Nr. 8/86 August
DM 6.50, sfr 6.50, öS 50, Lit 5900, hfl 7.50

peeker

Netzwerk für Apple-Rechner

Pic-Edit – ein universeller Grafik-Editor.

Einblicke in Logo

Aussagenlogik für Programmierer

Imagewriter II und Unidisk

Hüthig PUBLIKATION

Leser werben Leser

»peeker« bietet Ihnen was!

Wer jetzt schenkt hat mehr von seinem Apple. Dafür schenkt »peeker« Ihnen etwas: Den praktischen Disk-Locher, der die Speicherkapazität Ihrer Disketten verdoppelt!

Sie wissen ja, wie gut der »peeker« Ihnen im täglichen Umgang mit Ihrem Apple behilflich ist. Und Sie brauchen Ihren »peeker« nicht mehr zu teilen oder auszuleihen.

Der blaue Disketten-Locher ist unser Geschenk an Sie für einen neuen »peeker«-Abonnenten. Denn wer einen Apple hat, der soll auch seinen »peeker« haben.

Peeker 8/86

Bestellcoupon

Ich habe den neuen Abonnenten geworben und erhalte kostenlos den Disk-Locher.

Name, Vorname

Straße, Postfach

PLZ, Ort

X

Datum, Unterschrift

Ich bin der neue Abonnent. Bitte liefern Sie mir bis auf Widerruf, zumindest aber für 1 Jahr, »peeker« zum Jahresbezugspreis von DM 72,– (Ausland plus DM 18,– Porto) an folgende Anschrift:

Name, Vorname

Straße, Postfach

PLZ, Ort

X

Datum, Unterschrift

Gewünschte Zahlungsweise
☐ gegen Rechnung
☐ bargeldlos durch Bankeinzug

Konto-Nr. | Bankleitzahl

Geldinstitut

Vertrauensgarantie:
Diese Bestellung kann ich innerhalb einer Woche bei Dr. Alfred Hüthig Verlag GmbH, Im Weiher 10, 6900 Heidelberg 1 widerrufen. Zur Wahrung der Frist genügt die rechtzeitige Absendung. Ich bestätige die Kenntnisnahme mit meiner Unterschrift:

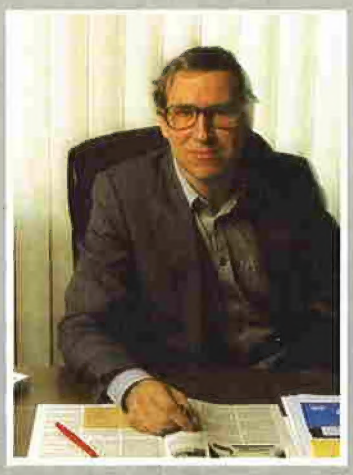
X

2. Unterschrift

Coupon ausschneiden und einsenden an:

»peeker«
Abonnementservice
Im Weiher 10
6900 Heidelberg 1

Hüthig
PUBLIKATION



EDITORIAL

Der depersonalisierte PC

Im vorletzten Peeker haben wir an dieser Stelle auf den paradoxen Trend in Richtung mehr Speicherkapazität bei weniger Mikro-Leistung aufmerksam gemacht, den man nicht nur bei Grafikrechnern (Macintosh usw.), sondern auch bei konventionellen Mikrocomputern (IBM-PC usw.) beobachten kann. Die Gründe für diese paradoxe Entwicklung sehen wir in den „auf die Schnelle" in Hochsprachen produzierten Betriebssystemen, Programmiersprachen und Anwenderprogrammen. An den in C geschriebenen BASIC-Interpreter haben sich Atari-Besitzer offensichtlich inzwischen gewöhnt. Und so wird man wohl nicht mehr lange warten müssen, bis eine Software-Firma einen BASIC-Interpreter X vorstellt, der in dem BASIC-Dialekt Y geschrieben ist.

Dieser verhängnisvolle Trend führt jedoch nicht nur zu aufgedunsenen und erschreckend langsamen Programmen (Adipositas-Syndrom), sondern auch dazu, daß solche Fat-and-Slow-Mikros selbst in Teilbereichen von einer einzelnen Person nicht mehr verstanden werden können. Der PC wird damit depersonalisiert (Depersonalisationssyndrom). Wer den guten alten Apple II verstehen wollte, brauchte nur zu entsprechenden Büchern und Zeitschriften zu greifen, denn die wichtigsten Betriebssysteme und Programmiersprachen wurden sprichwörtlich bis zum letzten Byte seziert:

– Den Anfang machten Steve Wozniak und Allen Baum, die den Quellcode des Monitors im Reference Manual abdrucken ließen.
– DOS 3.3 wurde von Randy Hyde sowie später von Don Worth und Pieter Lechner und noch später von Bernd Ruhland komplett kommentiert.
– Der Applesoft-Interpreter wurde von Glen Bredon und später von Matthias Buck vollständig disassembliert.
– ProDOS wurde von mir und später von Arne Schäpers bis in die letzten Feinheiten beschrieben.

All diese Analysen waren und sind nur deshalb möglich, weil die zitierten Betriebssysteme und Programmiersprachen

1. in Assembler geschrieben sind,
2. einen überschaubaren Umfang haben und
3. sich seit längerer Zeit im Einsatz befinden.

Genau diese drei Merkmale treffen auf die meisten Programme der „modernen" PCs nicht zu, und es fällt uns deshalb nicht schwer zu prophezeien, daß vollständig disassemblierte Betriebssysteme und Programmiersprachen von Rechnern wie Macintosh, Atari, Amiga usw. niemals in Buchform erscheinen werden. Damit haben EDV-Schüler, Informatik-Studenten und ambitionierte Mikro-Hobbyisten jedoch keine Chance mehr, die wirklichen Geheimnisse eines Mikrocomputers zu begreifen. Man wird den Personal Computer zwar weiterhin persönlich bedienen, aber nicht mehr persönlich verstehen können.

Aus der Großrechner-Historie wissen wir, daß die Trennung zwischen „denen da drinnen", den wissenden Hohepriestern in den Rechenzentren, und „denen da draußen", dem profan-unwissenden Volk, bewußt gepflegt worden ist. Einige Jahre lang, und dies ist nicht zuletzt dem Apple II zu verdanken, konnten die Geheimnisse der EDV breiten Kreisen zugänglich gemacht werden. Die zukünftigen PCs werden jedoch so komplex und undurchsichtig sein, daß eine Ent-Persönlichung der persönlichen Rechner unabwendbar sein wird.

Peeker-Umfrage

Wir möchten an dieser Stelle noch einmal auf unsere Umfrage-Karte auf Seite 45 hinweisen, mit der Sie beeinflussen können, ob der Peeker weiterhin eine reine Apple-II-Zeitschrift bleiben oder zusätzlich andere Rechner (Macintosh, IBM-PC, Atari) behandeln wird.

Ulrich Stiehl

inhalt

Impressum

Peeker
3. Jahrgang 1986
ISSN 0176–9200

Verleger und Herausgeber: Dipl.-Kfm. Holger Hüthig
Geschäftsführung Zeitschriften: Heinz Melcher
Chefredakteur: Ulrich Stiehl (us)
Redaktion: Dagmar Berberich

Telefonnummern:
Zentrale: 06221/489-1
Redaktion: 06221/489-352
Anzeigen: 06221/489-206
Abonnement: 06221/489-283
Software: 06221/489-231
Bücher: 06221/489-353
(Bestellungen bitte nur schriftlich)

Abonnement:
Der Abonnent kann seine Bestellung innerhalb von 7 Tagen schriftlich durch Mitteilung an den Dr. Alfred Hüthig Verlag GmbH, Postfach 10 28 69, 6900 Heidelberg 1, widerrufen. Zur Fristwahrung genügt die rechtzeitige Absendung des Widerrufs (Datum des Poststempels). Das Abonnement verlängert sich zu den jeweils gültigen Bedingungen um ein Jahr, wenn es nicht zwei Monate vor Jahresende schriftlich gekündigt wird. Die Abonnementsgelder werden jährlich im voraus in Rechnung gestellt, wobei bei Teilnahme am Lastschriftabbuchungsverfahren über die Postscheckämter und Bankinstitute eine vierteljährliche Abbuchung möglich ist.
Nichterscheinen infolge höherer Gewalt berechtigt nicht zu Ansprüchen gegen den Verlag.

Hüthig
PUBLIKATION

PEEKER

Heft 8/1986

UCSD

Pic-Edit
Ein universeller Grafik-Editor
in 128K-Apple-Pascal 1.2
von Jürgen Geiß 6

Logo

Einblicke in Logo
Apple Logo II
von Dipl.-Päd. Ernst F. Haas 22

Grundlagen

Das Wahre ist eins
Aussagenlogik für Programmierer
von Ulrich Stiehl 32

ProDOS

EDIT
Ein Luxus-Editor für ProDOS
Teil 3: Die Makrosprache
von Arne Schäpers 40

Hardware

Netzwerk für Apple-Rechner
Low-Cost-Rechner-Kopplung
für jedermann
von Alexander Niemeyer 50

Produkte

Imagewriter II und Unidisk
im Kurztest
von Ulrich Stiehl 56

Ein Heim für die Maus
Mouse Trap
vorgestellt von Thomas Bühner 61

Zeichnen mit der Maus
Maus-Grafiktablett – Mouse Tracer
getestet von Thomas Bühner 61

Grafik zu Papier bringen
Hires-Grafik-Druckprogramme
Imageprinter II, Printographer
und Zoom Grafix
getestet von Thomas Bühner 63

Hobby

Abenteurer in Green-Sky
Adventure-Spiel Below the Root
getestet von Thomas Bühner 67

Ein Tag im Stellwerk
Spiel Train Dispatcher
getestet von Thomas Bühner 68

Fehlerjagd im Computerlabor
Spiel I.O.Silver
getestet von Thomas Bühner 69

Anschrift:
Dr. Alfred Hüthig Verlag GmbH
Im Weiher 10, Postfach 10 28 69
6900 Heidelberg
Telefon (0 62 21) 4 89-1
Telex 4-6 17 27 hued d.
Telefax (0 62 21) 489 279
BTX * 51851 #

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

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

Bankverbindungen:
Zahlungen: an den Dr. Alfred Hüthig Verlag GmbH, D-6900 Heidelberg 1: Postgiro-konten: BRD: Ludwigshafen 4799-673, BLZ 545 100 67; Österreich: Wien 7555 88; Schweiz: Basel 40-24417; Niederlande: Den Haag 1 457 28; Italien: Mailand 5 968 92 08; Belgien: Brüssel 1084 1261; Dänemark: Kopenhagen 603 4969; Norwegen: Oslo 199 4243; Schweden: Stockholm 5477 76-5
Bankkonten: Landeszentralbank Heidelberg 67 207 341; BLZ 672 000 00; Deutsche Bank Heidelberg 02 65 041; BLZ 672 700 03; Bezirkssparkasse Heidelberg 204 51, BLZ 672 500 20.

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

Pic-Edit

Ein universeller Grafik-Editor in 128K-Apple-Pascal 1.2

von Jürgen Geiß

Hinweis: Die Objektcodes des nachfolgend beschriebenen UCSD-Pascal-Programmpakets nehmen die *gesamte* Sammeldisk #20 ein. Der Quelltext umfaßt über 7000 Zeilen und wird deshalb aus verständlichen Gründen nicht abgedruckt. Pic-Edit von Jürgen Geiß ist das Gegenstück zu dem Turtle-Graphics-Library-Paket von Dieter Geiß und kostet DM 48,–*.
Bei entsprechender Nachfrage werden die sehr umfänglichen Quelltexte für beide Programmpakete über den Hüthig-Software-Service erhältlich sein (Pic-Edit-Quelltext-Disketten DM 48,–*; Turtle-Graphics-Quelltext-Disketten DM 48,–*). Sie können Ihre unverbindliche Bestellung per Postkarte vormerken lassen und erhalten dann später eine Benachrichtigung.

* Für Fortsetzungsbezieher je DM 38,–. Wegen des höheren Preises wird die Disk #20 jedoch nur auf Anforderung verschickt.

I. Aufgabenstellung

II. Aufbau und Einzelheiten der Implementierung

1. Globals
   1.1. File-Handling
      1.1.1. Benutzerdateien
         – Workfile
         – Zeichensätze
         – Funktionstasten
         – Drucker
      1.1.2. Systemdateien
   1.2. Eingabegeräte
      1.2.1. Tastatur
      1.2.2. Joystick
      1.2.3. Maus
2. Init
   2.1. Die Cursoren
   2.2. Sonstige Initialisierungen
3. Menü
   3.1. Das Command-Fenster
   3.2. Das Text-Fenster
   3.3. Das Graphic-Sheet-Fenster
4. Draw
   4.1. Die Cursoren
   4.2. Allgemeiner Zeichenvorgang
   4.3. Die Undo-Funktion
   4.4. Seitenwechsel
   4.5. Lupe
5. Show
6. Print
   6.1. Drucken großer Bilder
   6.2. Verschiedene Drucker
7. Sonstige Module
   7.1. AppleStuff
   7.2. MouseStuff
   7.3. TurtleGraphics
   7.4. TurtleIO
   7.5. TurtleRealIO
8. Utilities
   8.1. Makemenu
      8.1.1. Symbol-Textfiles
      8.1.2. SYSTEM.SYMBOLS
   8.2. Cruncher
   8.3. Designer

III. Ergebnis und Probleme

1. Ergebnis
2. Probleme

IV. Benutzerhandbuch

1. Inbetriebnahme des Systems
   1.1. Ein Diskettenlaufwerk
   1.2. Zwei Diskettenlaufwerke
   1.3. Das Menü
   1.4. Filenamen
   1.5. Erste Annäherung
2. Bedienung
   2.1. Bewegung des Zeigers
      2.1.1. Maus
      2.1.2. Joystick
      2.1.3. Tastatur
      2.1.4. Repeat-Faktor
   2.2. Auswählen eines Kommandos
   2.3. Auswählen eines Unterkommandos
   2.4. Wechseln ins Draw-Modul
3. Komponenten des Menü-Moduls
   3.1. Commands
      3.1.1. Diskette
      3.1.2. Bleistift
      3.1.3. Pinsel
      3.1.4. Eindimensionales Achsenkreuz
      3.1.5. Zweidimensionales Achsenkreuz
      3.1.6. Bewegungsparameter
      3.1.7. Linien
      3.1.8. Polygone
      3.1.9. Rechtecke
      3.1.10. Gefüllte Rechtecke
      3.1.11. Kreise
      3.1.12. Funktionstasten
      3.1.13. Text
      3.1.14. Zeichensätze
      3.1.15. Radiergummi
      3.1.16. Lupe
      3.1.17. Strichstärke
      3.1.18. Farben
      3.1.19. Editorrahmen zum Verschieben und Kopieren
      3.1.20. Mülleimer
      3.1.21. Drucken
      3.1.22. Verkleinern
      3.1.23. Hilfe
      3.1.24. bis
      3.1.31. Erweiterungen
      3.1.32. Verlassen (Exit)
   3.2. Text-Input-Output-Window
   3.3. Graphics-Sheet
4. Komponenten des Draw-Moduls
   4.1. Verschiedene Hilfsmittel
      4.1.1. Diskette
      4.1.2. Bleistift
      4.1.3. Pinsel
      4.1.4. Eindimensionales Achsenkreuz
      4.1.5. Zweidimensionales Achsenkreuz
      4.1.6. Bewegungsparameter
      4.1.7. Linien
      4.1.8. Polygone
      4.1.9. Rechtecke
      4.1.10. Gefüllte Rechtecke
      4.1.11. Kreise
      4.1.12. Funktionstasten
      4.1.13. Text
      4.1.14. Zeichensätze
      4.1.15. Radiergummi
      4.1.16. Lupe
      4.1.17. Strichstärke
      4.1.18. Farben
      4.1.19. Editorrahmen zum Verschieben und Kopieren
      4.1.20. Mülleimer
      4.1.21. Drucken
      4.1.22. Verkleinern
      4.1.23. Hilfe
      4.1.24. bis
      4.1.31. Erweiterungen
      4.1.32. Verlassen (Exit)
   4.2. Seitenwechsel
5. Das Show-Modul
6. Das Print-Modul
7. Zusätzliche Tastaturfunktionen
   7.1. Globale Funktionen
   7.2. Funktionen im Menü-Modul
   7.3. Funktionen im Draw-Modul

Anhang A: Sammeldiskette
Anhang B: Literatur

I. AUFGABENSTELLUNG

Es sollte ein Grafik-Editor auf einem Apple IIe entwickelt werden, mit dem sich (fast) beliebig große Bilder editieren lassen. Diese sollten dann auf beliebigen Matrixdruckern ausgegeben werden können. Zum Editieren werden verschiedene Hilfsmittel wie Bleistift, Radiergummi, Lupe usw. zur Verfügung gestellt, wie man sie auch teilweise von anderen Systemen her kennt.

II. AUFBAU UND EINZELHEITEN DER IMPLEMENTIERUNG

Der Editor besteht aus 6 Modulen und einem Hauptprogramm, das diese 6 Module benutzt. Außerdem werden noch die Module AppleStuff, MouseStuff, TurtleGraphics, TurtleIO und TurtleRealIO benötigt. Diese sind aber nicht direkt Bestandteil des eigentlichen Editors.

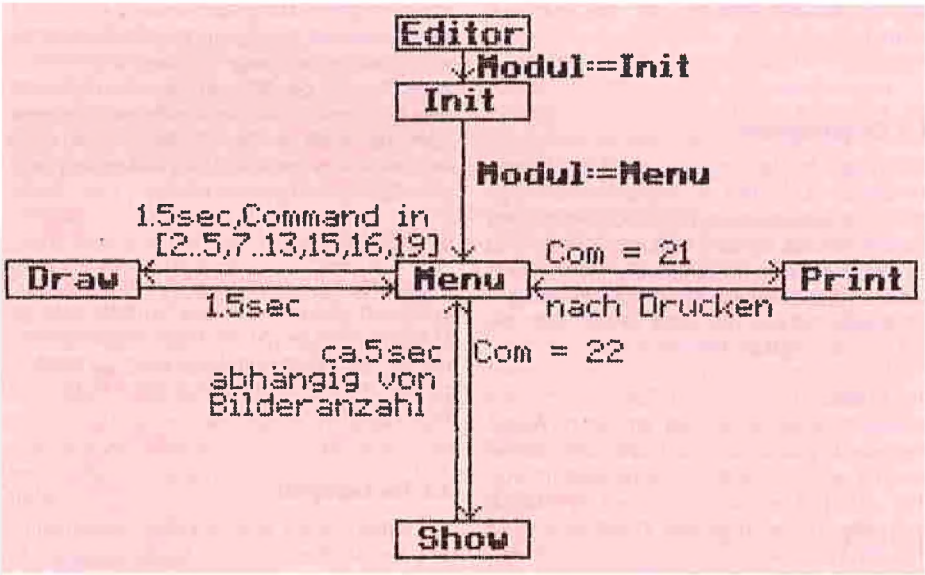


Abb. 1: Zusammenspiel der Module

1. Globals

Das Modul „Globals" dient vor allem als Schnittstelle zwischen den anderen Modulen und enthält alle globalen Variablen und Prozeduren, die von anderen Modulen aus benutzt werden. Dazu gehören z.B. die Diskettenverwaltung und die Treiber für die Eingabe.

1.1. File-Handling

1.1.1. Benutzerdateien

Workfile: Ein Grafik-File besteht immer aus 2 Teilen, der eigentlichen Grafik und einer *Informationsdatei* zu dieser Grafik. Dabei hat die Grafik einen gültigen Dateinamen gemäß UCSD-Syntax mit dem Suffix „.GRAF" und die Informationsdatei den gleichen Namen mit Suffix „.INFO".
Beispiel: „ BILD.1.GRAF" und „BILD.1.INFO".
Eine Grafik kann maximal 24 Bildschirmseiten in X-Richtung und 25 in Y-Richtung groß sein. Damit ergäbe sich eine Gesamtkapazität von 600 Bildschirmseiten. Aus praktischen Gründen wurde eine Begrenzung auf höchstens 255 Seiten (2M Pixeln) vorgenommen. Eine 1:1-Abbildung der Matrix auf Diskette hätte eine enorme Speicherplatzverschwendung zur Folge gehabt. Würde man beispielsweise das Bild mit den Koordinaten (0, 0) und dann eines mit (23, 24) editieren, so wäre die Datei 2M groß, obwohl in Wirklichkeit nur 16K (2 Bilder) davon benutzt wären. Deswegen wurde die Informationsdatei eingeführt. Der Zugriff auf ein Bild wird indirekt mit einer Tabelle vorgenommen. Die Reihenfolge der Bilder auf Diskette hängt dann direkt von der Reihenfolge des Editierens ab. **Abbildung 2** verdeutlicht dies noch einmal.
Geht die Informationsdatei einer großen Grafik einmal verloren, so kann man die Grafik mit einem Trick trotzdem weitereditieren. Dazu muß nur eine leere Grafik mit den gleichen Abmessungen und der gleichen Reihenfolge des Editierens aufgebaut werden. Die dabei erzeugte Info-Datei muß dann in die verlorene Info-Datei umbenannt werden.

Zeichensätze: Zeichensätze können mit speziellen Programmen editiert und gespeichert werden. Diese können dann im Menü geladen werden. Zur Standardausrüstung des Editors werden die Fonts „ASCII", „GERMAN", „GREEK", „MATH" und „SUPER.SUB" (hoch- und tiefgestellte Buchstaben und Zahlen) mitgeliefert. Der Aufbau eines selbst zu entwickelnden Zeichensatzes kann dem „Apple Pascal Language Reference Manual" entnommen werden. Zu ladende Zeichensätze müssen immer das Suffix „.FONT" tragen.

Funktionstasten: Die Files, die zur Speicherung von Funktionstasten dienen, können vom Benutzer selbst erzeugt werden. Dabei haben alle Files das Suffix „.KEYS". Bis zu acht Funktionstasten mit je 255 Zeichen Länge lassen sich in einem File abspeichern. Eine genaue Beschreibung des Vorgangs erfolgt im Benutzerhandbuch.

Drucker: Ein File, in dem sich alle benötigten Steuerzeichen für Drucker und Interface befinden, heißt „PRINTER.INFO". Dieser File muß vom Menü aus erzeugt werden. Eine ausführliche Beschreibung findet sich in Kapitel 6 (Print).

1.1.2. Systemdateien

Das Menü, von dem aus das Editieren gestartet wird, ist ebenfalls ein Grafik-File. Man sollte also niemals eine Grafik erzeugen, die den Namen „MENU.GRAF" trägt, da sonst Schwierigkeiten auftreten.
Das Laden und Speichern dieses Menüs erfolgt normalerweise im schnellen RAM des Apple-Hauptspeichers. Dadurch konnte die Zeit zum Wechseln von einem Modul ins andere auf die Hälfte verringert werden (ca. 1,5s). Es kann aber vorkommen – z.B. beim Drucken von Grafiken –, daß diese zusätzlichen 8K der RAM-Version des Menüs verloren gehen. Dann wird das Menü auf Diskette ausgelagert und beim erneuten Eintritt ins Menü-Modul von Diskette geladen. Es wird dann sofort wieder eine RAM-Version hergestellt.
Bei Lesefehlern wird eine Fehlermeldung ausgegeben und das Programm verlassen. Vorher werden alle Files geschlossen, damit eventuell vorhandene Grafiken nicht verlorengehen. Wird das Menü auf Diskette beschädigt, so muß eine

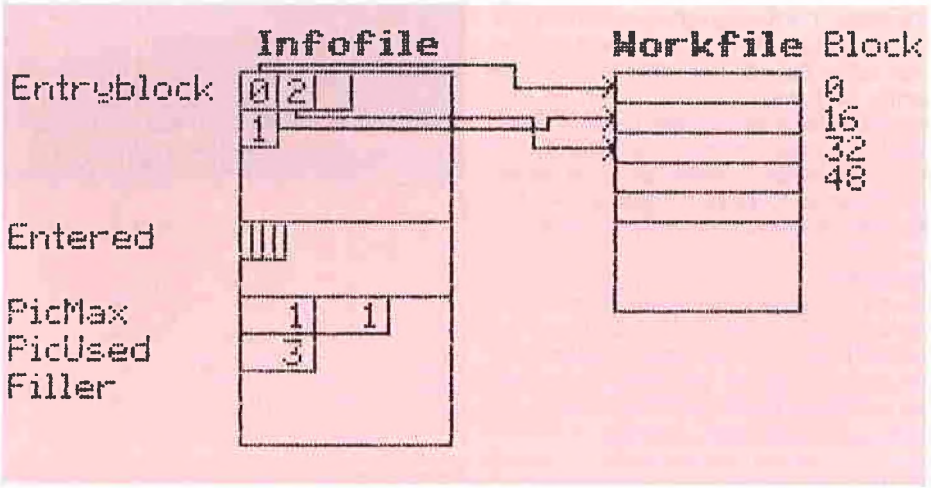


Abb. 2: Infofile und Workfile

Backup-Diskette benutzt und neu gebootet werden.

1.2. Eingabegeräte

Der Treiber für die Eingabe sieht 3 verschiedene Geräte vor: Tastatur, Joystick und Maus. Wenn sie angeschlossen sind, können sie alle parallel benutzt werden. Dadurch wird eine zu große Spezialisierung der Geräte, die am Computer angeschlossen sein müssen, ausgeschlossen. Ist z.B. die Maus defekt, kann mit der Tastatur weitergearbeitet werden.

1.2.1. Tastatur

Dieses Eingabemedium ist an jedem Apple-Rechner angeschlossen und kann daher immer benutzt werden. Für die Cursorbewegung wurden neun Tasten reserviert, die auf der Tastatur quadratisch angeordnet sind. Zusätzlich können die vier Pfeiltasten des Apple IIe benutzt werden. Die Geschwindigkeit, mit der sich der Cursor bei jedem Tastendruck fortbewegt, kann mittels eines Repeat-Faktors eingegeben werden. Dieser läßt sich leicht durch die Tasten 0..9 von 1 bis 999 einstellen. Außerdem werden noch die beiden Apfel-Tasten benutzt.
Ein besondere Eigenschaft des Tastaturtreibers ist die Möglichkeit, eigene Funktionstasten zu definieren. Bis zu acht verschiedene Operationen, bei der jede bis zu 255 Zeichen lang sein darf, können mittels <ESC>-Sequenzen aufgenommen und abgespielt werden. Damit lassen sich Muster, die öfter benutzt werden sollen, aufnehmen und an beliebiger Stelle wieder abspielen. Durch den Repeat-Faktor kann das Muster sogar vergrößert oder verkleinert werden. Außerdem lassen sich die Funktionstasten auf Diskette speichern und laden, so daß praktisch unbegrenzt viele Funktionstasten definiert werden können.

1.2.2. Joystick

Der Treiber für einen Joystick war ursprünglich schon im Modul AppleStuff implementiert; da dieses aber noch etwas erweitert wurde und unnötige Prozeduren entfernt wurden, handelt es sich hier nicht um das Original-AppleStuff von Apple, sondern um eine eigene Entwicklung.
Ist ein Joystick angeschlossen, so wird er automatisch benutzt. Man kann ihn aber bei Bedarf abschalten. Die Bewegung des Sticks wird *relativ* und nicht absolut an den Cursor übertragen. Das heißt: Bewegt man den Joystick nach rechts, so bewegt sich der Cursor mit einer konstanten, aber einstellbaren Geschwindigkeit gleichmäßig nach rechts. Entsprechendes gilt für alle anderen 7 Richtungen. Die beiden Knöpfe eines Joysticks entsprechen den beiden Apfel-Tasten auf der Apple-IIe-Tastatur.

1.2.3. Maus

Da die Apple-Maus nur einen Knopf besitzt, wurde ein Doppelklicken eingeführt, um eine weitere Funktion zur Verfügung zu stellen. Der Mausknopf entspricht der offenen Apfel-Taste auf dem Apple IIe bzw. dem Knopf Nummer 0 eines angeschlossenen Joysticks. Auch hier kann die Feinheit der Bewegung eingestellt werden. Soll pixelgenau gezeichnet werden, so sollte man die Tastatur der Maus und dem Joystick vorziehen. Mit einem Repeat-Faktor von 1 kann dann das Zeichengerät (z.B. Bleistift) jeweils pixelweise bewegt werden.
Beim Drücken des Mausknopfes wird die Aktion erst nach etwa einer halben Sekunde ausgeführt, da in dieser Zeit die Möglichkeit eines Doppelklicks besteht. Diese Verzögerung kann vom Benutzer eingestellt werden.

2. Init

Im Init-Modul werden alle globalen Variablen initialisiert und verschiedene Symbole erzeugt. Diese Routine wurde als Modul implementiert, um sie nach dem Aufruf auslagern zu können, da sie im weiteren Programmablauf nicht mehr benötigt wird.

2.1. Die Cursoren

Für jeden Cursor (z.B. Bleistift, Radiergummi usw.) existiert eine eigene Initialisierungsroutine. Dabei kann man meist zwischen 2 Typen von Cursoren unterscheiden, dem eigentlichen Abbild des Cursors und seiner Maske. Steht ein Cursor (z.B. Bleistift) auf einem Hintergrund, so wird dieser gespeichert, und der Cursor wird auf den Hintergrund gesetzt. Diese Methode ist ästhetischer als die XOR-Invertierung. Vor dem Zeichnen des Cursors muß dann mit Hilfe der Maske der Hintergrund weggeblendet werden. Dies dauert etwas länger, kann aber aus obengenanntem Grund hingenommen werden.

2.2. Sonstige Initialisierungen

Nach dem Initialisieren der globalen Variablen wird nach einer Maus gesucht. Wird eine gefunden, so wird diese initialisiert. Anschließend wird die Grafik initialisiert und nach dem „SYSTEM.CHARSET" gesucht. Dieser File wird unbedingt für die Darstellung von einigen Sonderzeichen benötigt. Fehlt er oder ist er defekt, muß mit einer anderen Diskette neu gebootet werden. Anschließend wird nach „PRINTER.INFO" gesucht, in dem Informationen für die Steuerzeichen des Druckers stehen. Wird dieser File gefunden, so wird er eingelesen. Näheres dazu unter Punkt 6 bei „Print". Schließlich wird noch das Menü geladen und eine RAM-Version desselben angelegt. Der File „MENU.GRAF" ist unbedingt notwendig. Bei Nichtvorhandensein ist neues Booten mit der Backup-Diskette erforderlich.

3. Menü

Das Menü besteht aus 3 Fenstern: Command, Text I/O, Graphic-Sheet. **Abbildung 3** zeigt das Menü, das von Diskette geladen wurde, mit einem Beispiel im Graphic-Sheet.

Für jedes Kommando aus dem Command-Fenster liegt eine eigene Prozedur vor, so daß noch leicht weitere Kommandos eingefügt werden können (insgesamt 32).

3.1. Das Kommando-Fenster

Bewegt man sich mit dem Cursor (ein Pfeil) in dieses Fenster und hat den Knopf der Maus gedrückt, so wird das entsprechende Kommado invers dargestellt. Läßt man den Mausknopf los, so wird ein Untermenü im Textfenster dargestellt, aus dem mit dem Cursor oder den Zifferntasten ausgewählt werden kann. Bei bestimmten Kommandos kann in andere Module (wie z.B. Print) verzweigt werden. Eine genaue Erklärung der einzelnen Kommandos folgt im Benutzerhandbuch.

3.2. Das Text-Fenster

Dieses Fenster dient der Ein- und Ausgabe von Filenamen, Fehlermeldungen und – wie schon oben besprochen – der Auswahl von Unterkommandos.

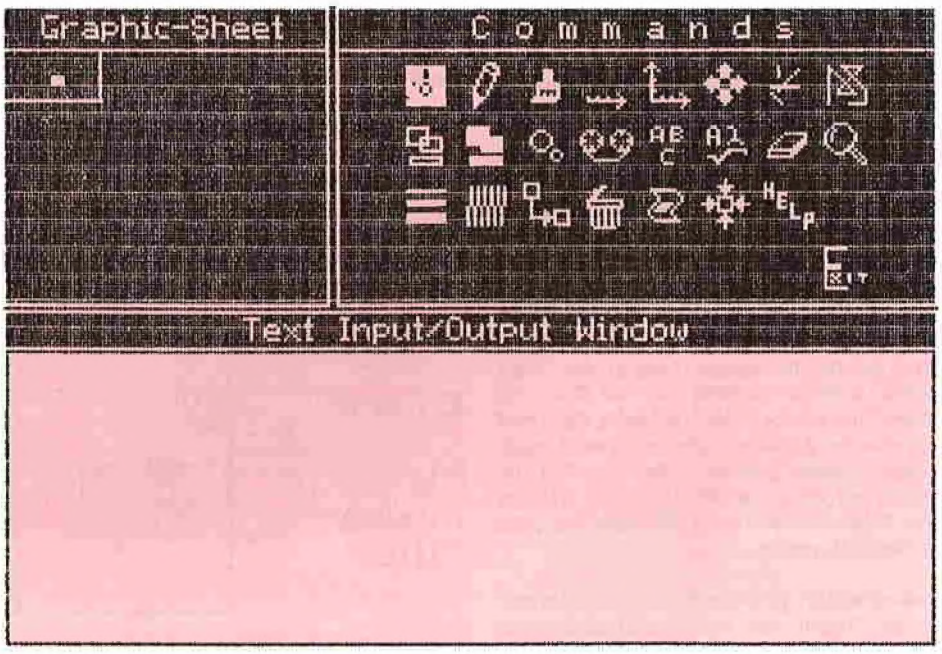


Abb. 3: Das Menü mit den drei Fenstern

3.3. Das Graphic-Sheet-Fenster

Eine Grafik kann in Pic-Edit aus bis zu 24x25 Teilbildern bestehen. In diesem Fenster wird sowohl die augenblickliche Größe der Gesamtgrafik als auch die Position der aktuellen Grafik angezeigt. (Die aktuelle Grafik ist die, in die man gelangt, wenn man das Menü verläßt und ins Draw-Modul wechselt.) Dabei kann mit dem Cursor die Größe der Gesamtgrafik bestimmt werden, indem man auf das Bild klickt, das am weitesten rechts unten stehen soll. In der obigen Abbildung erkennt man im Graphic-Sheet-Fenster den weißen Rahmen der Gesamtgrafik. In diesem befindet sich ein weißes Rechteck, das die Position der aktuellen Grafik anzeigt.

4. Draw

Im Draw-Modul können Grafiken erstellt werden. Dabei stehen dem Benutzer verschiedene Hilfsmittel wie Bleistift, Radiergummi, Lupe usw. zur Verfügung. Auch die Farbe des Zeichenstiftes kann bestimmt werden. Verlassen wird das Draw-Modul durch einen Doppelklick der Maus, was einem zweimaligen Drücken der offenen Apfel-Taste entspricht, durch einmaliges Drükken der geschlossenen Apfel-Taste oder durch zweimaliges Drücken von <ESC>.

Eine Besonderheit ist das Zeichnen von Koordinatensystemen. Dazu müssen nur die X- und Y-Achsen gezogen werden. Das Eintragen der Einheiten und der Beschriftung erfolgt automatisch. Die Skalierung kann durch Parameterangabe gänzlich vom Benutzer eingestellt werden. Es können auch Punkte in das Koordinatensystem eingetragen werden, die mit einer Geraden verbunden werden.

4.1. Die Cursoren

Ein aus dem Menü gewähltes Kommando kann eventuell durch den Cursor erkannt werden. Wurde der Bleistift ausgewählt, so erscheint auch im Draw-Modul ein Bleistift. Ein Auswählen des Radiergummis erzeugt im Draw-Modul ebenfalls einen Radiergummi. Die häufigste Art des Cursors aber ist das Fadenkreuz. Dieses wird bei 8 Kommandos benutzt. Somit läßt sich aus dem Fadenkreuz nicht auf das im Menü gewählte Kommando schließen.
Es gibt auch einen Pinsel, der verschiedene Formen annehmen kann. Diese reichen von einem Punkt über eine Linie bis zu einem quadratischen Klotz. Damit lassen sich größere Flächen schwarz einfärben.

4.2. Allgemeiner Zeichenvorgang

Allgemein sieht ein Zeichenvorgang immer folgendermaßen aus:
1. Drücken des Mausknopfes legt den Startpunkt fest.
2. Bewegen der Maus erzeugt die entsprechende geometrische Figur (z.B. Linie) in der Farbe Reverse.
3. Loslassen des Knopfes setzt die Figur fest (endgültiges Zeichnen mit der ausgewählten Farbe).

4.3. Die Undo-Funktion

Eine Undo-Funktion stellt man sich immer so vor: Nach jeder Aktion kann durch Auswählen von Undo diese wieder rückgängig gemacht werden. Leider konnte diese Vorgabe in der Implementierung nicht ganz eingehalten werden. Es müßte dann nach jeder Aktion die gesamte Grafikseite des Apple (8K) gerettet werden. Dies ist zwar mit einem einzigen Befehl (moveleft) in UCSD-Pascal möglich, aber es würde nach jeder Aktion zu einer Verzögerung kommen.
Damit man aber nicht ganz hilflos dasteht, gibt es doch eine eingeschränkte Undo-Funktion. Jedesmal, wenn man vom Draw- ins Menü-Modul wechselt, wird die Grafik auf Diskette gerettet. Wählt man nun irgendein Kommando aus und wechselt wieder zu Draw, so kann man die vorher abgespeicherte Grafik mittels <ESC> U wieder in den Bildschirm laden. Falls man sich verzeichnet hat, kann man sich damit wieder die vorherige Version der Grafik herholen.

4.4. Seitenwechsel

Die Besonderheit des Editors besteht in der Erzeugung von (fast) beliebig großen Grafiken. Dazu legt man innerhalb des Menüs im Graphic-Sheet die Größe der gesamten Grafik fest; diese läßt sich nachträglich noch vergrößern, falls man sie anfangs zu klein gewählt hat. Hat man also eine bestimmte Größe festgelegt, so kann im Draw-Modul mittels „P" ein Seitenwechsel herbeigeführt werden, wenn man auf einem der vier Ränder steht. (Ein Doppelklick der Maus ist auch möglich.)
Abbildung 4 zeigt die vier Fenstermöglichkeiten innerhalb einer Gesamtgrafik. In diesem Beispiel ist die Gesamtgrafik 5x5 Bilder groß (1400x960 Pixeln).

Steht der Cursor am rechten Rand, so verschiebt sich das Fenster nach rechts. Entsprechendes gilt für die anderen Richtungen. Das Zustandsübergangsdiagramm zeigt die Bewegung des Fensters, wenn eine der 4 Richtungen angewählt wurde.
Das Fenster Nummer 4 wurde nicht implementiert. Die Prozeduren dafür sind aber schon vorgesehen, so daß sie bei Bedarf erweitert werden können.

4.5. Lupe

Für Benutzer mit schlechten Augen oder für pixelgenaues Zeichnen wird eine Lupe zur Verfügung gestellt. Da das Vergrößern eines Bildausschnitts in Pascal zu langsam geworden wäre, wurde dieser Teil in Assembler programmiert. Trotzdem wurde zum besseren Verständnis eine entsprechende Pascal-Prozedur als Kommentar mit ins Draw-Modul eingebaut. Die Assemblerprozedur ist recht kurz, und die Antwortzeit liegt etwa bei einer zehntel Sekunde.
Befindet man sich in der Vergrößerung, so wird ein Bleistift zur Verfügung gestellt, mit dem schwarze oder weiße Pixeln – je nach vorherigem Hintergrund – gesetzt werden können. Da in einer Vergrößerung der Bildausschnitt relativ klein ist (40x24 Pixeln), wird hier die Möglichkeit gegeben, den Ausschnitt zu verschieben. Wie beim normalen Seitenwechsel (Kap. 4.4) kann ein „P" für Paging eingegeben werden. Der Bleistift wird dann zu einer Hand, um die Möglichkeit des Verschiebens anzuzeigen.

5. Show

Dieses Modul hat nur eine einzige Funktion: Das Erzeugen eines verkleinerten Bildes der Gesamtgrafik, das verzerrungsfrei ist und auf den Grafikbildschirm paßt. Dabei hängt die Verkleinerung von der Größe der gesamten Grafik ab. Ist sie z.B. 2x4 Teilbilder groß (dies entspricht einer DIN-A4-Seite bei einer 1:1-Grafikausgabe auf dem Drucker), so ergibt sich der Verkleinerungsmaßstab als max. [PicMax.X, PicMax.Y], also im Beispiel eine Verkleinerung von 1:4.
Man kann sich nun fragen, nach welchem Gesetz das verkleinerte Bild herzustellen ist. Die erste Idee war folgende (die Verkleinerung sei 1:n):
Berechne aus dem Originalbild die Summe der gesetzten Pixeln in einem Quadrat mit der Größe n ↑ 2. Ist die Summe größer als n ↑ 2/2, so setze auch ein Pixel in dem zu erzeugenden Bild. Dies scheint einleuchtend, ergab aber in der Praxis schon bei einer Verkleinerung von 1:3 und mehr ein verkleinertes Bild, das nicht mehr zu erkennen war. Es waren einfach zu wenige Pixeln in der Verkleinerung gesetzt.
Die nächste Idee war die, die Formel zu „tunen". Statt n ↑ 2/2 wurde n benutzt. Aber auch dies half nur bei der Verkleinerung von einigen wenigen geometrischen Figuren. Bei einer Freihandzeichnung mit der Maus war bei einer Verkleinerung kaum noch etwas zu sehen.
Das beste Ergebnis hatte folgende Formel: Ein Punkt wird gesetzt, wenn nur die Summe aus dem Quadrat größer als 1 ist. Auch bei Frei-

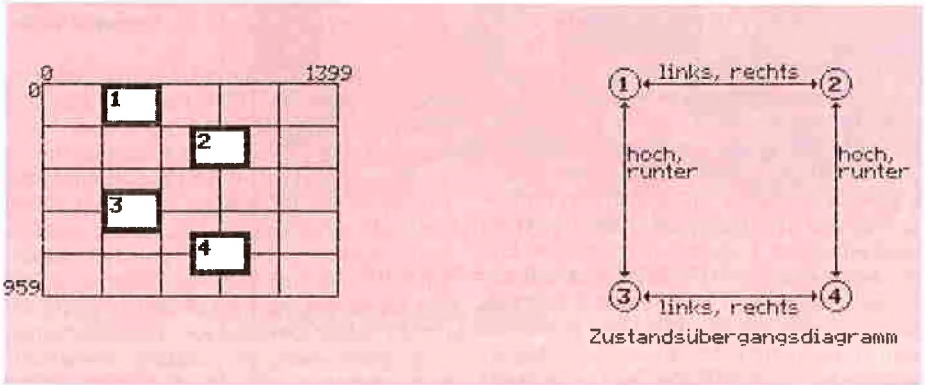


Abb. 4: Fenster in der Gesamtgrafik

handzeichnungen oder dann, wenn nur wenige Punkte gesetzt waren, konnte man nun in der Verkleinerung das Gesamtbild noch gut erkennen. Dieses Verfahren ist sogar bei Verkleinerungen von 1:n mit n > 10 mit guten Ergebnissen einzusetzen.
Das Show-Modul wurde aus Geschwindigkeitsgründen in Assembler implementiert. Auch hier ist eine Pascal-Version als Kommentar eingefügt, um den Assemblerteil besser verstehen zu können.

6. Print

Dieses Modul wurde zum größten Teil in Assembler geschrieben. Der Pascal-Teil besteht aus den 2 wichtigen Prozeduren „DumpActual" und „DumpWhole".

6.1. Drucken großer Bilder

Die Prozedur DumpActual bringt den augenblicklichen Bildschirminhalt auf einen angeschlossenen Drucker. Dies ist mit einigen Ausnahmen nicht weiter schwierig.
Bei größeren Bildern gibt es das Problem, nicht alle Teilbilder gleichzeitig im Speicher halten zu können (theoretisch maximal 2M). Deshalb wird ein Bild in Segmente zerlegt und nur jeweils ein Teil gedruckt. Wurden n Teilbilder in horizontaler Richtung editiert, so wird jedes Teilbild in N Segmente unterteilt. **Abbildung 5** zeigt ein Beispiel.

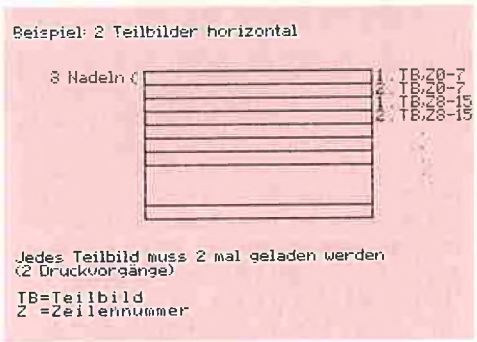


Abb. 5: Drucken von größeren Bildern

Ein Beispiel für N = 2 soll den Druckvorgang verdeutlichen:
Das 1. Teilbild wird geladen. Jeweils 8 Grafikzeilen werden zu einem Block zusammengefaßt. Block Nummer 0 ist dann Zeile 0..7, Block Nummer 1 Zeile 8..15 usw. Der 1. Block des 1. Teilbildes kommt in Zeile 0 des Grafikspeichers, der 2. Block in Zeile 16 usw. bis zum 12. Block. Dies entspricht der oberen Hälfte des 1. Teilbildes. Anschließend wird das zweite Teilbild geladen und dessen obere Hälfte in die Zeilen 8, 24 usw. eingetragen.
Beim anschließenden Drucken wird auf den Drucker Block 0 (Zeile 0..7) aus dem Grafikspeicher ausgegeben, wobei noch kein Wagenrücklauf und Zeilenvorschub erfolgt. Anschließend wird Block 1 (Zeile 8..15) gedruckt. Erst jetzt erfolgt ein CR und LF, da die jeweils ersten Blöcke von Teilbild 1 und Teilbild 2 nunmehr nebeneinander auf den Drucker ausgegeben wurden. Dies wird solange fortgesetzt, bis der gesamte Inhalt des Gafikspeichers ausgedruckt ist.
Mit den fehlenden unteren Bildhälften der beiden Teilbilder wird beim zweiten Druckvorgang entsprechend verfahren. Es gilt also folgende Regel: Die Anzahl der Lade- und Druckvorgänge ist proportional zur Anzahl der Teilbilder in X-Richtung.
Dieses Problem ist bei Teilbildern in Y-Richtung wesentlich einfacher zu lösen. Da der Drucker ja in diese Richtung druckt, können Teilbilder hintereinander auf den Drucker ausgegeben werden. Das Bild muß dann nicht segmentiert werden, es sei denn, es wurde schon in X-Richtung aufgetrennt.

6.2. Verschiedene Drucker

Beim Drucken von Bildern gibt es wegen der Druckervielfalt ein großes Problem: Viele Drukker benötigen unterschiedliche Steuerzeichen zum Umschalten auf den Grafikbetrieb, denn für Drucker gibt es keine Normen. Selbst bei Zeichen wie „CR" (Wagenrücklauf) verhalten sich Drucker verschieden. Die einen senden ein LF hinterher, die anderen nicht.
Beim Umschalten auf Grafikbetrieb gibt es noch viel größere Probleme. Die entsprechenden Zeichen sind in den meisten Fällen verschieden, und die Anzahl der Grafikbytes, die nach der Umschaltung folgen sollen, hat auch mindestens zwei verschiedene Formate. Epson verlangt z.B. für die Anzahl der Grafikbytes das „low/high-Format" (N mod 256, N div 256), während der Imagewriter von Apple einen String („0280") für 280 Punkte Grafik benötigt.
Für das Interface gelten ähnliche Probleme. Oft werden gewisse Codes (z.B. Tab = CHR (9)) als Steuerzeichen für das Interface interpretiert und dann von diesem weggefiltert. Da als Grafik-Bitmuster aber alle Werte von 0..255 vorkommen können, gibt es für verschiedene Interfaces einen Code zum Abschalten des Filterns, der ebenfalls in Form eines Strings angegeben werden kann. Aus diesen Gründen muß der File „PRINTER.INFO" angelegt werden. Er kann mit dem File „SYSTEM.MISCINFO" des UCSD-Systems verglichen werden, in dem sich die Steuerzeichen für den Bildschirm und die Tastatur befinden.

7. Sonstige Module

7.1. AppleStuff

Da das vom UCSD-System mitgelieferte AppleStuff teils Funktionen beinhaltet, die nicht benötigt werden, andere aber fehlen, wurde ein eigenes AppleStuff entwickelt. Es enthält nur noch die Prozeduren zum Abfragen der Knöpfe und des Joysticks sowie die Keypress-Funktion.
Da das gesamte System nur auf einem Apple IIe oder IIc unter der 128K-Pascal-Version 1.2 läuft, mußte noch die Funktion „codeavail" eingebaut werden. Sie gibt die Anzahl der freien Wörter (ein Wort hat 16 Bits) zurück, die in der 64K-Karte des IIe oder IIc liegen. Mit dieser Funktion ist es möglich zu testen, wieviele Codesegmente noch in den Speicher passen. Ein spezielles Programm von Dieter Geiß (der sog. „Cruncher", s. Peeker, Heft 10/1985) eliminiert Assemblerteile von Codesegmenten aus der 64K-Karte, da diese vom Betriebssystem schon in die unteren 64K kopiert werden, um dort ablaufen zu können. Ist der Cruncher, der als SYSTEM.ATTACH auf der PICEDIT-Diskette abläuft, einmal geladen, steht dem Grafik-Editor erheblich mehr Speicherplatz für den Programmcode zur Verfügung.

Ist der Cruncher nicht gestartet, lagert sich der Editor automatisch aus und lädt dann die einzelnen Module nach. Das dauert dann im Schnitt zwei Sekunden länger. Wer eigene Treiber mittels SYSTEM.ATTACH anschließen will, kann also auf den automatischen Start des Cruncher verzichten.

7.2. MouseStuff

Zu Beginn der Entwicklungszeit des Editors standen einem Pascal-Programmierer noch keine Maustreiber zur Verfügung, obwohl die Maus schon auf dem Markt war. Hier mußte also noch Vorarbeit geleistet werden, um eine Maus anschließen zu können. Der Maustreiber hält sich eng an die relativen Einsprungadressen im Firmware-ROM, so daß der Treiber auch noch bei neuen Versionen der Firmware funktionieren wird.
Inzwischen wird auch von der Firma Apple ein Maustreiber angeboten. Allerdings ist dieser nur für sog. Software-Entwickler zu haben.

7.3. TurtleGraphics

Mit dem beim UCSD-Pascal mitgelieferten TurtleGraphics-Programm war man bei weitem nicht in der Lage, komplexe Grafikpakete mit allem „Drum und Dran" zu entwickeln. So fehlten z.B. Prozeduren für die Ein- und Ausgabe im Grafikmodus, wie man sie vom Pascal-System gewohnt ist. Aber auch das gezielte Herausholen von Teilen aus dem Grafikspeicher in die Variablen hinein (wird z.B. beim Verschieben benötigt) und das Laden von verschiedenen Zeichensätzen fehlten völlig.
Dieter Geiß machte sich die Mühe, eine total neue, mächtige TurtleGraphics zu entwickeln, die anstelle der 16 Prozeduren der alten Form nunmehr 66 Prozeduren zur Grafikmanipulation und weitere zur Fensterverwaltung anbietet. Eine genaue Beschreibung aller einzelnen Befehle kann seiner Arbeit entnommen werden, die durch den Hüthig Software Service günstig zu beziehen ist. Diese TurtleGraphics ermöglicht dem IIe und IIc auch die doppelte Grafikauflösung mit bis zu 16 Farben (560x192 Pixeln).

7.4. TurtleIO

Dieses Modul beinhaltet alle Ein- und Ausgabeprozeduren, die man von der Pascal-Umgebung her gewohnt ist. So kann man Integer-Zahlen, Chars und Strings einlesen und ausgeben und die Feldbreite der Ausgabe angeben. Auch der Cursor kann programmiert werden. Zur genaueren Beschreibung siehe Punkt 7.3.

7.5. TurtleRealIO

Dieses Modul beinhaltet nur zwei Prozeduren, nämlich „ReadReal" und „WriteReal". Da die Ein- und Ausgabe von Real-Zahlen relativ kompliziert ist, wurde sie in ein eigenes Modul gesteckt, denn Real-Zahlen werden nicht so oft benötigt. (Der Editor benötigt Real-Zahlen nur bei der Eingabe von Punkten ins Koordinatensystem.) Das Modul kann also aus „Draw" ausgelagert werden.

8. Utilities

In diesem Kapitel werden einige Hilfsprogramme vorgestellt, die notwendig sind, um z.B. Erweiterungen vorzunehmen oder Zeichensätze zu editieren.

8.1. Makemenu

Dieses Hilfsprogramm wurde als allererstes Programm in Angriff genommen. Es hat die Aufgabe, den File MENU.GRAF herzustellen, also die Grafik für das Menü, in dem sich die drei Fenster Command, Text I/O und Graphic-Sheet befinden. Es erzeugt außerdem die Piktogramme für das Command-Fenster.
Nach dem Starten des Programms erscheint die Frage:
„Get what symbol textfile (without suffix .TEXT) ?"
Es gibt zwei mögliche Antworten:
– <Return> verläßt die Eingabe von weiteren Symbol-Files und geht zur nächsten Frage über.
– Eingabe des Namens eines Textfiles (wobei das Suffix „.TEXT" entfallen muß), genauer eines Symbol-Textfiles mit einem bestimmten Format (siehe 8.1.1.).
Nachdem mit <Return> die Eingabe(n) beendet wurde(n), kommt noch die Frage:
„Double Hires ?"
Diese Frage ist immer mit „n" zu beantworten. Sie wurde nur eingebaut, um ein eventuelles Erweitern auf doppelte Auflösung zu demonstrieren, und sollte daher auch nur für Demonstrationszwecke dienen.

8.1.1. Symbol-Textfiles
Ein Piktogramm des Command-Fensters besteht aus einer quadratischen Matrix der Dimension (1..Symbits, 1..Symbits), also aus einer 16x16-Matrix von booleschen Werten. Ist ein Wert „true", so wird ein Pixel in der Grafik auf weiß gesetzt.
Die Symbol-Textfiles bestehen aus zwei Teilen:
1. In Zeile 1 steht eine Integer-Zahl, die im Bereich [1..MaxSymbols] liegen muß. Sie gibt die Nummer des Piktogramms im Command-Fenster an.
2. In Zeile 2 bis 17 stehen Strings mit folgender Form: Delimiter – 16 beliebige Characters – Delimiter.
Ist der beliebige Character ein <Space>, so wird das entsprechende Bit der Matrix auf „false" gesetzt, andernfalls auf „true".
Abbildung 6 zeigt den Symbol-Textfile des Symbols Nummer 1 im Command-Fenster (eine 5 1/4" Diskette).

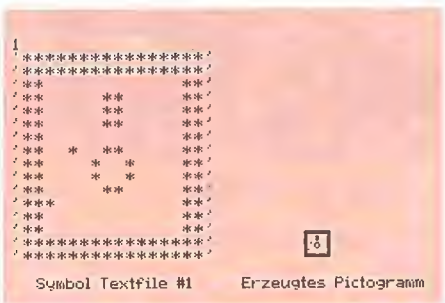


Abb. 6: Symbol-Textfile #1 und erzeugtes Piktogramm

8.1.2. SYSTEM.SYMBOLS
Das Hilfsprogramm MAKEMENU erstellt beim Einlesen der Symbol-Textfiles den File SYSTEM.SYMBOLS – sofern er noch nicht vorhanden war – und trägt dort die Symbole als Bitmuster in ein entsprechendes Array ein. Dies hat den Vorteil, daß lange Wartezeiten beim Einlesen der Symbol-Textfiles entfallen. Will man z.B. nur noch *ein* zusätzliches Symbol (Nummer 20) anhängen, so braucht man nicht wieder alle Symbole von 1 bis 19 einlesen, weil diese aus dem Datenfile SYSTEM.SYMBOLS geholt werden.

8.2. Cruncher

Der Cruncher von Dieter Geiß wurde schon in Kapitel 7.1. beschrieben. Er befindet sich auf der Boot-Diskette unter den Namen SYSTEM.ATTACH und CRUNCHER.CODE und wird automatisch beim Booten geladen. Dadurch erspart sich der Editor eine Menge Nachladezeit von Segmenten, die sonst ausgelagert werden müßten. Die genaue Beschreibung des Crunchers kann in der Ausgabe 10/85 des „Peeker" nachgelesen werden.

8.3. Designer

Der Designer von Herrn Gabor (s. Peeker, Heft 1/1986) ist ein Hilfsprogramm zum Entwickeln von eigenen Zeichensätzen. Er wurde nur der Vollständigkeit halber mit aufgenommen, da er nicht vom Autor erstellt, sondern lediglich überarbeitet wurde.
Die Bedienung des Designers ist menügesteuert. Nach dem Start muß der Filename eines Fonts angegeben werden. Ist dieser vorhanden, wird er eingelesen.

Weitere Einzelheiten kann man dem Aufsatz aus Heft 1/1986 entnehmen.

III. ERGEBNIS UND PROBLEME

1. Ergebnis

Das Ergebnis der Arbeit ist ein ca. 7000 Zeilen langes Programm. Der Code im einzelnen:

```
Globals       :  3414 Bytes (Pascal)
Init          :  3722 Bytes (Pascal)
Menu          : 14420 Bytes (Pascal)
Draw          :  8785 Bytes (Pascal,
                     wenig Assembler)
Show          :   998 Bytes (Assembler,
                     wenig Pascal)
Print         :  1736 Bytes (Assembler,
                     wenig Pascal)

AppleStuff    :   252 Bytes (Assembler)
MouseStuff    :   690 Bytes (Assembler)

TurteGraphics: 11190 Bytes (halb Pascal,
                     halb Assembler)
TurtleIO      :  2908 Bytes (Pascal,
                     wenig Assembler)
TurtleRealIO  :  1476 Bytes (Pascal)
```

Die letzten drei Module wurden nur der Vollständigkeit halber aufgeführt.
Die normalen Antwortzeiten liegen beim Wechseln vom Menü- in den Draw-Modus und umgekehrt im Bereich von 1,5s mit Cruncher und 3s ohne Cruncher. Die Druckzeit hängt natürlich von der Geschwindigkeit des Druckers und der Anzahl der Bilder ab, die in einer Gesamtgrafik editiert wurden. Ebenso hängt die Zeit für eine Verkleinerung von der Anzahl der Teilbilder ab.
Die Antwortzeiten können bei Benutzung einer RAM-Disk drastisch gesenkt werden.
Der Editor ist so aufgebaut, daß man weitgehend auf Betriebssystemfiles wie den Filer verzichten kann, denn ein Inhaltsverzeichnis kann auch vom Menü aus mit dem 1. Kommando (eine kleine Diskette) ausgegeben werden. Man benötigt nur noch den Pascal-Formatter, um wenigstens leere Datendisketten herstellen zu können.

2. Probleme

Die Entwicklung des Editors begann noch auf dem guten alten Apple II Plus unter 64K-Pascal 1.1. Als das Programm aber wuchs und wuchs, paßte es nicht mehr in den Speicher, denn man muß ja bedenken, daß schon die Grafik 8K beansprucht, abgesehen von TurtleGraphics, AppleStuff und MouseStuff. Außerdem braucht der Editor noch zusätzlich 8K Puffer zum Verschieben, Verkleinern und Drucken.
Die Entwicklung wurde dann auf einem Apple IIe mit 128K RAM unter der Pascal-Version 1.2 weitergeführt. Speicherplatzprobleme zur Übersetzungs- und Laufzeit traten damit nicht mehr auf.
Wie oben schon erwähnt, war die von Apple mitgelieferte TurtleGraphics total unbrauchbar. Dieses Problem beseitigte Dieter Geiß mit seiner hervorragenden neuen TurtleGraphics.
Ein größeres Problem stellte auch das Print-Modul dar, weil es weitgehend drucker- und interfaceunabhängig gestaltet werden sollte. (Diese Probleme sind in Kapitel 6 genau beschrieben.)
Daneben gab es noch viele andere kleinere Probleme, z.B. bei gewissen Interface-Karten wie der UltraTerm. Hier mußte der Type-ahead-Buffer abgestellt werden, da diese Karte sonst immer in den Textmodus umschaltet.

IV. BENUTZERHANDBUCH

Dieses Kapitel beschreibt das Starten und Benutzen des gesamten Systems anhand von Beispielen. Vor der Inbetriebnahme sollten alle Geräte angeschlossen werden, die später benutzt werden können (Joystick, Maus, Drucker).

1. Inbetriebnahme des Systems

Zunächst einiges zum System selbst:
Der Grafik-Editor läuft unter dem UCSD-Pascal-System (Apple-Pascal 1.2 mit 128K, also Apple IIe mit erweiterter 80-Zeichenkarte oder Apple IIc). Daher müssen Gerätenummern und Filenamen der UCSD-Syntax entsprechen. Im Prinzip handelt es sich um ein abgeschlossenes System, wenn man von der Herstellung von Leerdisketten (Formatierung, Anlegen eines Inhaltsverzeichnisses) absieht. Es empfiehlt sich, mindestens eine leere Pascal-Diskette zum Arbeiten bereitzulegen. Vor dem Start sollte man sich

auch eine Backup-Diskette anlegen, um gegen Überraschungen gewappnet zu sein.
Mit Hilfe des Grafik-Editors lassen sich – meines Wissens – erstmals Grafiken unter dem Pascal-System mit Hilfe einer Maus bearbeiten. Wer keine Maus besitzt, kann auch einen Joystick einsetzen. Besitzt man auch diesen nicht, oder ist eines der Geräte defekt, kann man immer noch die Tastatur zu Hilfe nehmen.

Hier einige Features des Editors:
– Editieren (fast) beliebig großer Grafiken (bis zu 6720x4800 Pixeln)
– Ausgabe auf vielen Druckern, da Steuerzeichen angegeben werden können
– Bearbeiten von Fremdbildern
– Laden verschiedener Zeichensätze (z.B. griechisch, mathematisch)
– Verschiedene Schriftarten (wie proportional, fett usw.)
– Definieren von Funktionstasten und Speichern auf Diskette

1.1. Ein Diskettenlaufwerk

Benutzer mit nur einem (Apple)-Diskettenlaufwerk (280 Blöcke) können auf der Systemdiskette höchstens zwei Bilder editieren. Es empfiehlt sich das Arbeiten mit einem größeren Laufwerk oder mindestens zwei Apple-Laufwerken.
Stecken Sie die Systemdiskette „PICEDIT:" in Laufwerk 1 und schalten Sie den Rechner ein. Nach 37s meldet sich das Menü mit der Aufforderung, einen Filenamen anzugeben.

1.2. Zwei Diskettenlaufwerke

Stecken Sie die Systemdiskette „PICEDIT:" in Laufwerk 1 und eine formatierte UCSD-Pascal-Datendiskette in Laufwerk 2 und schalten Sie den Rechner ein. Nach 37s meldet sich das Menü mit der Aufforderung, einen Filenamen anzugeben.

1.3. Menü

Zuerst erscheint ein Menü, in dem sich die Piktogramme der Kommandos befinden und in dem die Größe der gesamten Grafik angezeigt wird. In dieses Menü gelangt man immer nach Drucken, Verkleinern und nach dem Verlassen des Draw-Moduls.
Das Menü besteht aus drei Fenstern:
1. dem Command-Fenster zum Auswählen von verschiedenen Kommandos,
2. dem Graphic-Sheet-Fenster zum Einstellen der Gesamtgrafik,
3. dem Text-Input/Output-Fenster zum Auswählen von Untermenüs, der Eingabe von Dateinamen und der gesamten Textausgabe.
Abbildung 7 zeigt das Menü mit den drei Fenstern.

1.4. Filenamen

Ein UCSD-Filename sieht immer folgendermaßen aus:
Volumenummer + Filename + Suffix
Die Volumenummer stellt eine maximal 7 Zeichen umfassende Volume-Spezifikation dar, der Filename selbst darf maximal 15 Zeichen (inklusive Suffix „.S") umfassen. Beispiele:

```
#4:BILD                (entspricht Laufwerk 1,
                        Filename BILD)
*:TEST                 (entspricht der Boot-
                        Diskette, Filename Test)
PASCAL1:ARROWS.KEYS    (Name der Diskette
                        ist PASCAL1, Filename
                        ARROWS.KEYS)
```

Wird keine V-Spezifikation angegeben, so wird automatisch die Präfix-Diskette benutzt, was der Boot-Diskette PICEDIT: entspricht. Ein passendes Suffix wird vom System immer angehängt und braucht daher nicht vom Benutzer angehängt werden. Der Filename F kann somit maximal 10 Buchstaben lang sein.
Folgende Suffixe werden vom System angehängt:

F.GRAF (File enthält eine Grafik.)
F.INFO (File enthält zusätzliche Informationen zu einem Grafik-File.)
F.FONT (File enthält einen Zeichensatz, der geladen werden kann.)
F.KEYS (File enthält Funktionstasten, die geladen werden können.)

1.5. Erste Annäherung

Nachdem der Begriff Filename erklärt wurde, können wir die erste Frage des Grafik-Editors beantworten. Sie lautet:

```
"No workfile is present:
<Return> for SYSTEM.WRK.GRAF
? <Return> for directory listing"

>_
```

Das „_"-Zeichen steht für den Cursor. In Wirklichkeit ist der Apple-IIe-Cursor ein Schachbrett, CHR (127) oder ASCII DEL.
Geben Sie <Return> ein, wenn Sie eine neue Datei bearbeiten wollen, oder den Filenamen einer alten Datei. Wenn Sie sich vertippt haben, kann mit der „←"-Taste (Linkspfeil) jeweils ein Zeichen zurückgetippt werden. Die ganze Zeile kann mit <Ctrl-X> oder <DEL> gelöscht werden.
Wenn Sie nicht mehr wissen, welchen Namen die alte Datei hatte, geben Sie „? <Return>" ein. Es werden dann alle Files, die das Suffix „.GRAF" tragen, auf dem Bildschirm ausgegeben, nachdem eine Laufwerksnummer angegeben wurde.
Folgende Nummern entsprechen den Laufwerken:

```
#4 – Laufwerk 1 (Slot 6, Drive 1)
#5 – Laufwerk 2 (Slot 6, Drive 2)
#9 – Laufwerk 5 (Slot 4, Drive 1)
#1Ø– Laufwerk 6 (Slot 4, Drive 2)
#11– Laufwerk 3 (Slot 5, Drive 1)
#12– Laufwerk 4 (Slot 5, Drive 2)
```

Achtung!
Auf der Systemdiskette befindet sich ein File mit dem Namen „MENU.GRAF". Dies ist die Grafik des Menüs selbst; sie darf nie als Filename angegeben werden.

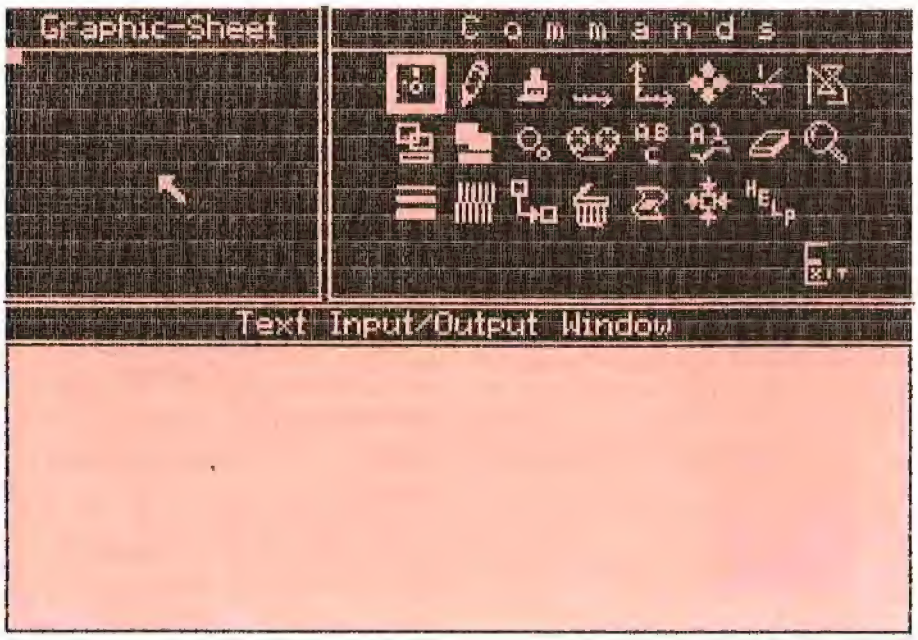


Abb. 7: Das Menü (vgl. Abb. 3)

2. Bedienung

In diesem Abschnitt wird der Umgang mit dem System geübt, insbesondere die Bewegungen des Zeigers, das Auswählen von Untermenüs usw.

2.1. Bewegung des Zeigers

Der Zeiger (ein Pfeil) wird das wichtigste Hilfsmittel sein, mit dem Sie im Menü zu tun haben. Mit ihm können Kommandos ausgewählt, die Größe einer Grafik bestimmt werden usw.
Bevor wir auf die einzelnen Möglichkeiten genauer eingehen, sollen noch drei wichtige Begriffe erläutert werden. Einer davon heißt im Englischen „drag", was soviel bedeutet wie ‚schleppen, schleifen, ziehen'. Damit ist gemeint, daß man den Knopf der Maus (oder des Joysticks oder die offene Apfel-Taste) gedrückt hält, während man sich bewegt. Wir wollen dies „ziehen" nennen.
„Ziehen Sie die Maus nach rechts unten" bedeutet also: Knopf drücken, gedrückt halten und

sich anschließend nach rechts unten bewegen.
Mit „Knopf drücken" ist entweder der Mausknopf gemeint oder der 1. Knopf des Joysticks oder aber die offene Apfel-Taste.
Mit „Doppelklick" ist folgendes gemeint: Sie drücken den Knopf zweimal und lassen ihn gleich wieder los, ohne die Maus dabei zu bewegen. Dies erfolgt in einem Zeitintervall, das Sie selbst bestimmen können.

2.1.1. Maus
Wenn Sie eine Maus an den Apple angeschlossen haben, so benutzen Sie diese auf jeden Fall. Mit ihr kann man sich am schnellsten in alle beliebige Richtungen fortbewegen. Fährt man mit der Maus nach oben, so bewegt sich der Zeiger auch in diese Richtung. Für alle anderen Richtungen gilt das Entsprechende.
Die Maus hat einen Druckknopf. Dieser ist in der Funktion identisch mit dem der offenen Apfel-Taste (links neben der Leer-Taste). Auf ihre Bedeutung wird später noch eingegangen.

2.1.2. Joystick
Da eine Original-Apple-Maus relativ teuer ist (wie leider alle Apple-Produkte), wird man sich manchmal nach anderen X-Y-Eingabegeräten umsehen. Ein billiger Ersatz für die Maus ist der Joystick. Er ist aber leider sehr ungenau in der Führung und kann für pixelgenaues Zeichnen kaum benutzt werden.
Bewegt man den Knüppel nach oben, so bewegt sich der Zeiger in diese Richtung. Man kann auch gleichzeitig nach links und oben fahren, indem man den Knüppel in die entsprechende Richtung bewegt. Der Stick hat zwei Druckknöpfe. Diese sind hardwaremäßig mit den beiden Apfel-Tasten (offener und geschlossener Apfel, links bzw. rechts neben der Leer-Taste) verbunden. Auf die Bedeutung der beiden Druckknöpfe kommen wir später noch zu sprechen.

2.1.3. Tastatur
Wer weder Maus noch Joystick besitzt, kann sich mit der Tastatur weiterhelfen, denn eine Tastatur hat jeder Apple, und auf sie kann man ohnehin nicht verzichten. Wer möchte denn schon Filenamen mit der Maus eingeben? Viele Tasten haben eine spezielle Bedeutung, aber hier werden wir nur auf die Bewegungstasten eingehen.
Auf einigen Tasten ist die Bewegungsrichtung schon angegeben – dies sind die 4 Pfeiltasten. Mit ihnen bewegt man sich jeweils um eine definierte, vom Benutzer einstellbare Anzahl von Pixeln fort. Eine Taste mit ähnlicher Bedeutung ist die Tabulatortaste. Mit ihr kann man sich um 8 Schritte nach rechts bewegen. Dies ist unabhängig vom sog. Repeat-Faktor. Mit ihm läßt sich die Anzahl der Schritte (Pixeln, Bildpunkte, Rasterpunkte) einstellen, um die man sich fortbewegen will. Dies gilt sowohl für die Tastatur als auch für den Joystick. Daneben gibt es noch weitere 9 Tasten für die Bewegung:

```
U I O
 J K L
  M , .
```

Von der Form, die diese 9 Tasten bilden, kann schon auf ihre Funktion geschlossen werden. Ein „U" bewegt den Pfeil nach links oben, während ein „I" ihn nach oben bewegt usw.
Eine Taste hat eine besondere Bedeutung. Das „K" wiederholt die zuletzt gewählte Richtung solange, bis wieder ein „K" eingegeben wird. Zwar hat der Apple auf den Tasten ein Auto-Repeat, aber es empfiehlt sich, das „K" zu benutzen, da die Wiederholungsrate dann genauso schnell ist wie die Hauptschleife der Tastaturabfrage des Systems.

2.1.4. Repeat-Faktor
Die Anzahl der Schritte, die der Pfeil in eine bestimmte Richtung geht, kann wie folgt eingestellt werden:
Man gibt einfach eine Zahl zwischen 1 und 999 ein, die den sog. „Repeat-Faktor" bestimmt. Jeder Tastendruck, der den Pfeil bewegt, bewegt ihn nun um diese Anzahl von Schritten fort. Gibt man z.B. „20" ein, gefolgt von einer nicht-numerischen Taste, so bewegt sich der Pfeil immer in Abschnitten von 20 Schritten vorwärts. Dieser Faktor hat auch für den Joystick die gleiche Wirkung. Damit kann man die Geschwindigkeit festsetzen, mit der sich der Pfeil fortbewegt, wenn man den Joystick in irgendeine Richtung bewegt.
Die Eingabe einer Ziffer kann nur erfolgen, wenn im Textfenster kein Untermenü angezeigt wird, das mit Ziffern ausgewählt werden kann.

2.2. Auswählen eines Kommandos

Nachdem klar ist, wie man seinen Zeiger bewegen kann, soll das Auswählen eines Kommandos beschrieben werden. Dazu bewegt man den Zeiger im Command-Fenster auf ein beliebiges Symbol (z.B. Bleistift) und drückt den Knopf (der Maus). Nach etwa einer halben Sekunde wird das angewählte Symbol invers dargestellt. Diese Zeitspanne wird benötigt, um einen eventuellen Doppelklick der Maus zu testen. Nach dem Loslassen des Knopfes erscheint im Text-Input/Output-Fenster eine Meldung, die vom ausgewählten Symbol abhängt. In diesem Fall nur die Meldung „Pencil:".
Soll ein anderes Symbol ausgewählt werden, so verfährt man wie oben beschrieben. Man kann auch den Knopf gedrückt halten und sich dann über die verschiedenen Symbole bewegen. Hat man ein bestimmtes Symbol ausgewählt, läßt man den Knopf einfach wieder los.

2.3. Auswählen eines Unterkommandos

Wählen Sie nun – wie unter Punkt 2.2. beschrieben – das Symbol für die Strichstärke aus. Dort sind 4 verschiedene Stricharten dargestellt (Symbol 17). Nachdem Sie den Knopf losgelassen haben, erscheint im Text-Fenster die Meldung „Linewidth:" mit einer Anzahl verschiedener Striche, die von 1 bis 4 durchnumeriert sind. Die augenblickliche Strichstärke wird durch ein besonderes Symbol (check mark), das Check-Zeichen (ein Haken), gekennzeichnet. Es gibt nun 2 Möglichkeiten, eine andere Strichstärke auszuwählen:
1. Eingabe einer Ziffer im Bereich der Durchnumerierung (hier „1" bis „4")
2. Bewegen des Zeigers in die Zeile, in der die gewünschte Strichstärke steht, anschließend Druck auf den Knopf und Loslassen.
Man kann die Maus auch „ziehen" (siehe oben), während man sich in den Zeilen mit der Angabe der Strichstärken befindet. Hat man gefunden, was man sucht, so läßt man den Knopf einfach wieder los. Während des „Ziehens" wird die ausgewählte Zeile invers dargestellt.

2.4. Wechseln ins Draw-Modul

Mit bestimmten Kommandos des Command-Fensters (z.B. Bleistift) kann ins Draw-Modul gewechselt werden. Nur in diesem Modul können alle graphischen Aktionen wie Freihandzeichnen, Beschriften, Verschieben von Grafiken usw. ausgeführt werden.
Ein Wechsel ins Draw-Modul (nachdem ein entsprechendes Piktogramm gewählt wurde) kann auf mehrere Arten erfolgen:
1. Doppelklick der Maus oder der offenen Apfel-Taste (oder des entsprechenden Knopfes auf dem Joystick).
2. Drücken der geschlossenen Apfel-Taste (oder des entsprechenden Knopfes auf dem Joystick).
3. Zweimaliges Drücken der <Esc>-Taste.

3. Komponenten des Menü-Moduls

3.1. Commands

Nachdem wir also wissen, wie man Kommandos und Untermenüs auswählt, werden wir alle Kommandos des Command-Fensters durchlaufen und uns genauer ansehen.

3.1.1. Diskette
Durch Auswählen des Diskettenzeichens erscheint folgendes Untermenü im Text-Fenster:
Filer:
Workfile: SYSTEM.WRK.GRAF (oder der Name einer gerade bearbeiteten Grafik)
1: Load Picture
2: Save Picture
3: List whole directory
4: List graphic files only

Möchte man ein Bild laden, so wählt man Punkt 1. Anschließend kann ein <Return> eingegeben werden, wenn man den File SYSTEM.WRK.GRAF laden möchte, oder ein anderer Filename, der höchstens 10 Buchstaben lang sein darf (ohne die weiter oben beschriebene Volume-Spezifikation). Ein Beispiel dazu ist „#5:Bild.3", wobei im Laufwerk 2 nach einem entsprechenden File gesucht wird. Wird er gefunden, so wird er geladen, ansonsten wird ein neuer File mit dem entsprechenden Namen angelegt.
Wer den Namen der zu editierenden Grafik vergessen hat, kann auch „?<Return>" eingeben. Es erscheint dann die Meldung:
List unit #
Dann muß eine der Zahlen 4, 5, 9, 10, 11 oder 12 eingegeben werden, die den Volume-Nummern des Pascal-Systems entsprechen. Die Zuordung zu den Laufwerken ist in Kapitel 1.5. beschrieben.
Es werden dann alle Files mit dem Suffix „.GRAF" aufgelistet. Danach wird wieder nach dem Filenamen gefragt.
Wählt man Punkt 2, so wird der Info-File des entsprechenden Grafik-Files auf Diskette gerettet. Der Grafik-File selbst ist schon gerettet. Beide Files werden dann geschlossen.

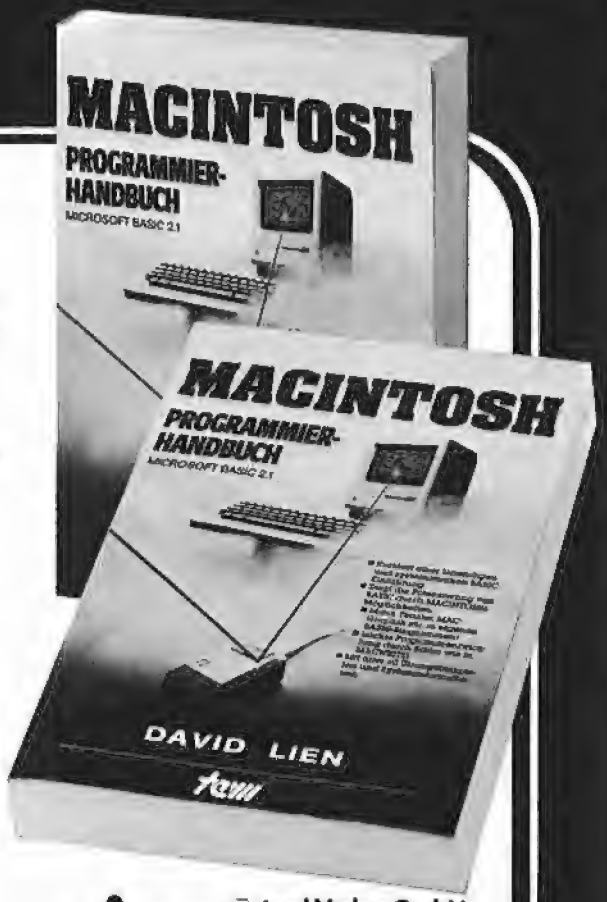
MS BASIC ist eine höchst moderne und leicht erlernbare Programmiersprache!

210 reservierte Begriffe...Programmsynthese aus Modulen (lokale Variablen, Wertübergaben mit COMMON)...Nachladen von Segmenten (CHAIN, mit Parameterübergabe)...Einbinden von MAC-Funktionen (Graphik, Maus, Fenster, Knöpfe usw.)...Fremddateizugriff (von MS-BASIC auf Dateien von z.B. MULTIPLAN, MACPAINT, WORD)...Programmablaufsteuerung (Ereigniserfassung wie ON TIMER, ON MOUSE)...Peripheriebefehle (wie COMI für DFÜ)...strukturierte BASIC-Programmierung ohne Zeilennummern...Gleitkommaarithmetik.....

MACINTOSH und MS BASIC bilden eine komfortable Programmierumgebung!

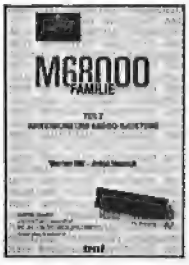
Mehrfachfenster für Simultanbeobachtung (z.B. Fenster 1: Hauptprogramm, Fenster 2: Unterprogramm; oder Fenster 1: Listing, Fenster 2: Ergebnisse)...Mausedition wie in Textprogrammen...Collagetechnik (Programmabschnitte ausschneiden und versetzen)...

HIER DER KURSTEXT einer lebendigen und systematischen BASIC-Einführung von dem US-Professor David Lien. Ein Text für das Selbststudium, das in anspruchsvolle BASIC-Programmierungen mit den Funktionen des MACINTOSH mündet. Ideal als Schultext.
450 Seiten, Softcover, DM 59,–



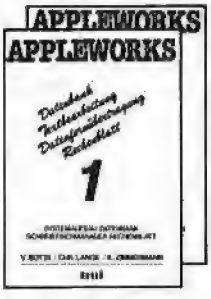
tewi Verlag GmbH
Theo-Prosel-Weg 1
8000 München 40

Weitere te-wi-Bücher



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

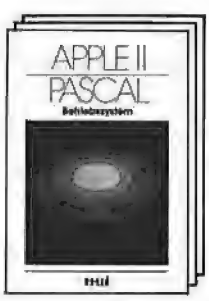
Das APPLE II/II+/IIe/IIc-Handbuch
L. Poole
Erst mit Hilfe dieses Leitfadens werden Sie Ihren Apple II erfolgreich einsetzen, denn Text und Bildmaterial gehen weit über das hinaus, was herstellerseitig an Literatur angeboten wird.
Neu überarbeitet und jetzt um die spezifischen Eigenheiten der Modelle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



APPLEWORKS integriert in APPLE II, IIe, IIc die Funktionen eines modernen Schreibtisches: Textverarbeitung, Datenbank, Rechenblatt, Datenfernübertragung. Sämtliche System/Anwendungsfragen in 2 Bänden.
Von Botta/Lange/Zimmermann, je 264 Seiten und je DM 49,–

DAS C-BUCH. **NEU**
Textbuch für C-Kurse und C-Anwendungen auf PCs. Beschreibt sämtliche Konstrukte der C-Sprache unter den Betriebssystemen MS DOS, CP/M, ISIS, UNIX und für die C-Compiler von MS, DR, LATTICE, INTEL. Didaktisch und typographisch außergewöhnlich. Mit über 100 laufähigen Beispielprogrammen für PCs. Zeigt Realisierungen neuester Softwarestrategien in „C".
Von Herold/Unger. Herbst 86.
Etwa 500 Seiten. Softcover. DM 79,–

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



Erstes deutsches Referenzwerk
sämtlicher Befehle und Systemroutinen von Apple II, IIplus, IIe
APPLE II PASCAL
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 4601

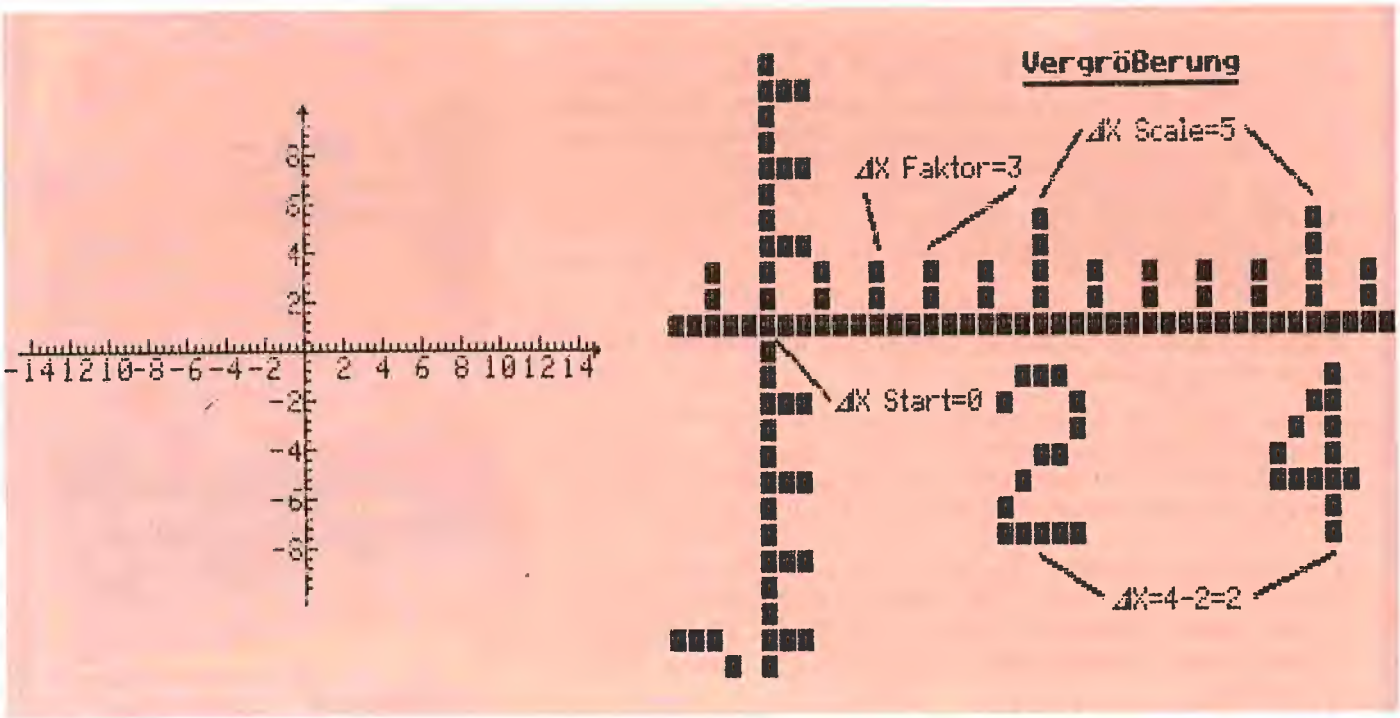


Abb. 8: Zweidimensionales Koordinatenkreuz

Bei Anwahl von Punkt 3 wird das gesamte Inhaltsverzeichnis einer Diskette ausgegeben, bei Punkt 4 nur alle Grafik-Files. Vorher muß natürlich das Laufwerk spezifiziert werden.

3.1.2. Bleistift
Das Wählen des Bleistifts hat im Menü keine besondere Bedeutung. Beim Wechseln ins Draw-Modul kann mit dem Bleistift gezeichnet werden.

3.1.3. Pinsel
Nach Anwählen dieses Piktogrammes erscheint eine Palette von 7 verschiedenen Pinselformen. Wählen Sie einen geeigneten Pinsel aus.

3.1.4. Eindimensionales Achsenkreuz
Zum besseren Verständnis der anzugebenden Parameter zeigt **Abbildung 8** ein zweidimensionales Koordinatenkreuz.

Die linke Hälfte der Abbildung zeigt ein Koordinatenkreuz mit den folgenden Parametern:

```
DeltaX-Start  = 0
DeltaX        = 2
DeltaX-Scale  = 5
DeltaX-Faktor = 3
```

Die Bedeutung der Parameter wird aus dem rechten Teil der Abbildung ersichtlich:
DeltaX-Start ist der Wert des Anfangspunktes, bei dem mit der Numerierung begonnen wird.
Sind X1 und X2 zwei aufeinanderfolgende Numerierungen, so gilt: DeltaX = X2 - X1 (im Beispiel 4 - 2 = 2).
DeltaX-Scale gibt an, nach wievielen Punkten jeweils ein etwas „längerer" Strich gesetzt werden soll.
DeltaX-Faktor gibt an, um wieviele Pixeln zwei Grundstriche jeweils voneinander entfernt sein sollen.

Die einzelnen Menüpunkte sehen wie folgt aus:
1: Setup DeltaX
2: Setup DeltaY
3: Set (x,y) input on
4: Set auto scaling off
5: Set auto numbers off
6: Show Setup

1. Mit Setup DeltaX können genau die vier oben angegebenen Parameter (DeltaX-Start usw.) eingestellt werden, wobei es sich um Integer-Werte handelt.
2. Entsprechendes gilt für die Y-Achse.
3. Mit diesem Punkt kann festgelegt werden, ob Punkte eingegeben werden sollen, die mit einer Geraden verbunden werden (hat nur bei zweidimensionaler Achse einen Sinn). Eine Beschreibung der Eingabe erfolgt im Kapitel über das Draw-Modul.
4. Hiermit kann die automatische Skalierung ein- und ausgeschaltet werden.
5. Hiermit kann die automatische Numerierung ein- und ausgeschaltet werden. Das ist sinnvoll, wenn man eigene Numerierungen vornehmen will, da die automatische Numerierung nur mit Ganzzahlen arbeitet.
6. Schließlich können alle Parameter und der Status der Punkte 3 bis 5 angezeigt werden.

3.1.5. Zweidimensionales Achsenkreuz
Innerhalb des Menüs hat dieses Kommando genau die gleiche Bedeutung wie 3.1.4.

3.1.6. Bewegungsparameter (Miscinfo)
Untermenü:
1. Set mousefactor to repeatfactor
2. Set mouse off
3. Set joystick on
4. Set max-time for double clicking

1. Damit läßt sich die Feinheit der Mausbewegung einstellen. Steht der aktuelle Repeat-Faktor auf 1, wird die Bewegung der Maus am gröbsten. Der Maus-Faktor verhält sich gerade umgekehrt wie der Repeat-Faktor. Die Einstellung des Repeat-Faktors wurde in 2.1.4. schon näher beschrieben. Zu Beginn des Programms steht der Maus-Faktor auf 2.
2. Ist eine Maus angeschlossen, so kann diese bei Bedarf abgeschaltet werden. Es ist sinnlos, den „Maus"-Parameter einzuschalten, wenn keine Maus vorhanden ist.
3. Für den Joystick gilt das Entsprechende.
4. Die Zeit, die für die Eingabe eines Doppelklicks zur Verfügung steht, wird Max-Time genannt. Eine hohe Max-Time läßt dem Benutzer viel Zeit für einen Doppelklick, aber die Zeit bis zu einer Antwort des Systems ist dann entsprechend hoch. Max-Time wird auf 8 initialisiert. Wird z.B. ein Accelerator-Karte benutzt (3.5-MHz-6502-CPU), so sollte die Max-Time entsprechend erhöht werden, da sonst kein Doppelklick mehr möglich ist. Dies gilt auch für Benutzer, die ihre Finger nicht schnell genug bewegen können.

3.1.7. bis 3.1.11.
Diese geometrischen Figuren können ausgewählt werden. Die Linien und Flächen werden dann mit der gewählten Farbe und Strichstärke gezeichnet.

3.1.12. Funktionstasten (record keys)
Soll eine beliebige Tastensequenz öfter wiederholt werden, so kann diese auf einfache Weise gespeichert werden. Bis zu acht solcher Funk-

tionstasten können jeweils 255 Tastendrücke aufnehmen.
Das Definieren dieser Funktionstasten geht wie folgt vor sich:
Eingabe von „<Esc> R <Esc> 1" definiert die Funktionstaste „<Esc> 1". Anschließend können bis zu 255 Tasten eingegeben werden. Zum Abschluß der Sequenz muß wieder ein „<Esc> 1" folgen.
Bsp: Definieren von viermal „Pfeil rechts", gefolgt von dreimal „Pfeil oben" in die Funktionstaste 2:

```
<Esc> R <Esc> 2 -> -> -> -> ↑ ↑ ↑ <Esc> 2
```

Durch Eingabe von „<Esc> 2" wird jedesmal die gespeicherte Sequenz „→ → → → ↑ ↑ ↑" abgespielt.
Wie wir noch sehen werden, haben Funktionstasten im Menü wenig Sinn, aber im Draw-Modul kann mit ihnen vorteilhaft gearbeitet werden.
Es können die Funktionstasten <Esc> 1 bis <Esc> 8 gespeichert und auf Diskette gerettet werden. Ebenso können einmal gerettete Funktionstasten wieder geladen werden. Das Untermenü hat die Form:
1. List record key files
2. Load function keys
3. Save function keys
4. Look function keys

1. Damit können alle Files, die Funktionstasten beinhalten, aufgelistet werden, nachdem die Laufwerksnummer (4, 5, 9..12) angegeben wurde.
2. Mit diesem Kommando können Funktionstasten geladen werden. An den Filenamen wird das Suffix „.KEYS" angehängt. Dieses Suffix darf also nicht mehr angegeben werden.
3. Es werden alle 8 Funktionstasten auf Diskette geschrieben. Der Filename darf maximal 10 Buchstaben lang sein. Das Suffix „.KEYS" wird angehängt.
4. Hat man vergessen, welche Tastendrücke den einzelnen Funktionstasten zugeordnet sind, so kann man diese ausgeben lassen. Nach Anwählen von Punkt 4 erscheint die Meldung: „Look what function key?"
Hier muß nur eine Ziffer zwischen 1 und 8 angegeben werden. Anschließend wird der Inhalt der entsprechenden Funktionstaste ausgegeben. Tastendrücke, bei denen der offene Apfel ebenfalls gedrückt wurde, werden in Breitschrift (d.h. ein Buchstabe erscheint doppelt so breit), normale Tastendrücke in Normalschrift dargestellt.

3.1.13. Text
Dem Benutzer stehen 3 verschiedene Schriftarten zur Verfügung, die beliebig miteinander gemischt werden können. Somit können insgesamt 8 Schriftarten erzeugt werden. Die 3 Grundschrifttypen, die mit 1, 2 oder 3 im Untermenü ausgewählt werden können, sind Fettschrift, Breitschrift und Proportionalschrift.

3.1.14. Zeichensätze
Damit die Grafiken auch vielseitig beschriftet werden können, stehen verschiedene Zeichensätze zur Verfügung. So existieren neben dem Standard-ASCII-Satz auch der entsprechende deutsche Zeichensatz und griechische und mathematische Symbole. Es können auch eigene Zeichensätze geladen werden. Alle Filenamen, die Zeichensätze darstellen, dürfen höchstens 10 Buchstaben lang sein, da noch das Suffix „.FONT" angehängt wird.
Mit Punkt 1 des Untermenüs kann ein Inhaltsverzeichnis aller Files auf Diskette ausgegeben werden, deren Suffix „.FONT" ist.
Mit Punkt 2 bis 5 können die entsprechenden Sätze geladen werden, und mit Punkt 6 kann ein eigener Zeichensatz geladen werden. Dabei darf das Suffix „.FONT" nicht angehängt werden.

3.1.15. Radiergummi
Dieses Kommando hat nur im Draw-Modul eine Funktion.

3.1.16. Lupe
Dieses Kommando hat nur im Draw-Modul eine Funktion.

3.1.17. Strichstärke
Dem Benutzer stehen 5 verschiedene Strichstärken zur Verfügung. Die Standardstärke ist unter Punkt 2 zu sehen. Die Strichstärke bestimmt, wie dick ein Strich bei Linien, Polygonen, Rechtecken und Kreisen werden soll. Bei gefüllten Rechtecken stellt die Strichstärke den Rand dar. Das Innere wird dann mit der gewählten Farbe gefüllt. Soll kein Rand gezeichnet werden, so ist die Strichstärke mit Punkt 1 zu wählen (eine gestrichelte, unsichtbare Linie).

3.1.18. Farben
Wer einen Farbmonitor an den Apple angeschlossen hat, kann dieses Piktogramm in 2 Farben erkennen. Es werden 8 verschiedene Farben angeboten. Standardfarbe ist Schwarz.
2 Farben verdienen besondere Beachtung:

Reverse:
Diese Farbe invertiert den Bildschirm an der entsprechenden Stelle. Aus Schwarz wird Weiß, aus Weiß wird Schwarz, die Farben kehren sich um.

Radar:
Diese Farbe kann nur bei Flächenfüllung benutzt werden. Es ist die Grundfarbe des Macintosh- bzw. des GEM-Desktop.

3.1.19. Editorrahmen zum Verschieben und Kopieren
Mit „Move only" können Bildausschnitte eingerahmt und verschoben werden, wobei der alte Hintergrund mit Weiß gefüllt wird.
Mit „Move & copy" bleibt der Hintergrund bestehen, d.h. es können Bildausschnitte kopiert werden.
Ein Beispiel zur Benutzung wird im Kapitel über das Draw-Modul vorgestellt.

3.1.20. Mülleimer
Untermenü:
1: Throw away actual picture
2: Throw away the whole graphics
3: Set whole sheet to actual picture

1. Damit kann das Teilbild (actual picture), das im Graphic-Sheet-Fenster angezeigt wird, gelöscht werden.
2. Hiermit wird die gesamte Grafik gelöscht.
3. Es kann vorkommen, daß man die Größe des Gesamtbildes versehentlich viel zu groß gewählt hat. Will man diese nachträglich verkleinern, setzt man das aktuelle Teilbild auf die rechte untere Ecke des Gesamtbildes, indem man mit der Maus auf dieses Teilbild klickt, und wählt Punkt 3 an. Das Gesamtbild schrumpft dann auf die entsprechende Größe. Auf die Teilbilder, die eventuell weiter rechts oder weiter unten editiert wurden, kann nicht mehr zugegriffen werden. Durch ein nachträgliches Vergrößern des Gesamtbildes können diese Bilder aber wieder zum Vorschein gebracht werden.

3.1.21. Drucken
Mit diesem Menüpunkt können die Druckparameter und Teilbilder oder die gesamte Grafik eingestellt werden. Untermenü:
1. Print whole graphics
2. Print actual picture
3. Change printer mode
4. Setup PRINTER.INFO

1. Hier wird sofort ins Print-Modul gewechselt und die gesamte Grafik auf dem Drucker ausgegeben. Besteht diese in horizontaler Richtung aus n Teilbildern, so werden n Lade- und Druckvorgänge benötigt. Dabei kann beobachtet werden, wie ein Teilbild in n Unterbilder zerlegt wird, bevor es gedruckt werden kann. Der Druck einer Grafik von 3x2 Teilbildern dauert etwa 110s, wenn ein Drucker mit einer Geschwindigkeit von 160 Zeichen/s zugrunde gelegt wird. Nach dem Drucken wird automatisch wieder ins Menü-Modul gewechselt.

2. Es wird ins Print-Modul gewechselt und nur dasjenige Teilbild gedruckt, das gerade im Graphic-Sheet-Fenster angewählt ist. Anschließend kommt man wie bei 1. wieder ins Menü-Modul.

3. Ein Drucker kann in bis zu acht verschiedenen Modi betrieben werden. So kann ein Epson FX-80 z.B. in alle 7 Grafikmodi geschaltet werden, ein Imagewriter der Firma Apple sogar in alle 8 Modi. Da bei verschiedenen Modi unterschiedliche Auflösungen erreicht werden (bei Epson FX-80 bis zu 1920 Pixeln/Zeile), können auch sehr große Grafiken gedruckt werden, die aus mehreren horizontalen Teilbildern bestehen. Ein FX-80 erlaubt hier bis zu 7xn Teilbilder (bei beliebigem n), da 7∗280 = 1960 ist. Es fehlen lediglich die 40 letzten Pixeln aus dem am weitesten rechts stehenden Teilbild.
Man sollte die Druckerauflösung immer so wählen, daß die gesamte Grafik in horizontaler Richtung noch auf eine Druckerbreite paßt. Man kann dies wie folgt berechnen: Anzahl der Teilbilder in horizontaler Richtung (ersichtlich aus dem Graphic-Sheet-Fenster), multipliziert mit 280.
Bsp: 3 Teilbilder → 3∗280 = 840 Pixeln/Zeile. Bei einem FX-80 sollte dann mindestens die Auflösung 960 Pixeln/Zeile eingeschaltet werden, was einem Druckmodus von (<Esc> L) entspricht.

4. Damit der Druck von Grafiken möglichst druckerunabhängig bleibt, muß der File PRINTER.INFO erzeugt werden. Wenn dieser File noch nicht angelegt wurde, aber schon gedruckt werden soll, erscheint die Meldung „PRINTER.INFO not ok. Create a new one before printing". Punkt 4 muß daher immer zuerst ange-

wählt werden, wenn dies vorher nicht schon einmal gemacht wurde. Damit wird PRINTER.INFO erzeugt und bei jedem Neustart benutzt. Der Infofile, der sich auf der Diskette befindet, ist auf den Epson FX-80 angepaßt. Daneben befindet sich auch der File „IMAGEWRTR.INFO" auf der Diskette, der für den Apple-Drucker Imagewriter angepaßt wurde. Dieser File muß in PRINTER.INFO umbenannt werden. Dies kann im Filer des UCSD-Systems geschehen.
Ein Beispiel eines Epson-Druckers mit 2 verschiedenen Print-Modi soll die Erstellung von PRINTER.INFO verdeutlichen.
Zunächst aber etwas Theorie: Ein Drucker benötigt mindestens 2 Angaben zur Ausgabe von Grafiken, den Zeilenabstand, der auf 8 Bits gesetzt werden muß, und die Steuerzeichen für das Umschalten auf den Grafikbetrieb. Außerdem ergeben sich manchmal Probleme mit dem Interface. Gewisse Zeichen (meist CHR (9)) werden als Steuerzeichen für das Interface interpretiert und von diesem „verschluckt". Beim Drucken aber können alle Werte zwischen 0 und 255 auftreten. Meist gibt es dann die Möglichkeit, dieses „Schlucken" zu verhindern. Bei der Super-Serial-Card von Apple besteht sie darin, daß vor dem Druckvorgang ein „<Ctrl-I> Z" an den Drucker gesendet wird, der dann die Weitergabe aller Bitmuster erlaubt. Dieser Code soll „Initialisierungscode des Interface" (init code of interface) genannt werden.
Um den Drucker für den Grafikbetrieb zu initialisieren, gibt es zwei Strings: Mit „Pinit" können z.B. der Zeilenabstand und alle sonstigen Initialisierungen eingestellt werden, die der Drucker nur einmal benötigt. Für die Umschaltung auf Grafik gibt es den String „Pline", der in jeder Zeile vor der Ausgabe der Grafikpixeln an den Drucker ausgegeben wird. Bei manchen Druckern ist dem Bit 0 die unterste Nadel zugeordnet („low" bei Epson), bei anderen die oberste („high" bei Imagewriter).
Die Anzahl der Grafikzeichen (Pixeln), die gedruckt werden sollen, müssen einmal als Ganzzahl (integer) und dann als String geschickt werden. Die Position dieser Grafikbytes in „Pline" ist meist 3 oder 4, da nach <Esc> K bei Epson die Grafikbytes an der 3. Stelle im String zu erwarten sind. Dies gilt auch für den Imagewriter.
Nun aber zum Beispiel Epson mit 2 Grafikmodi: Anwählen von Punkt 4 (Setup PRINTER.INFO). Danach kommt die Frage:
„Input modes as C(haracter, I(nteger?"
Wir antworten mit einem „C". Damit können die Druckersteuerzeichen als Zeichen von der Tastatur eingeben werden. Falls damit nicht alle ASCII-Codes erreicht werden können, sollte man als Eingabe „Integer" wählen. Die einzelnen Steuerzeichen können dann z.B. mit einem Komma getrennt werden. Näheres siehe **Tabelle 1**. Anschließend wird der File PRINTER.INFO erzeugt und auf Diskette gerettet. Nun kann in zwei verschiedenen Druckmodi gedruckt werden.

Tabelle 1

Meldungen:	Kommentar bzw. Eingabe
Input mode is character. Enter delimited string [e.g./stuff/] Control block #1:	
Pinit: /(Esc)3(24)/	Dies ist der erste Grafikmodus
	Eingabe ist /<Esc>3<Ctrl-X>/. Zeichen, deren ASCII-Codes zwischen 0 und 31 liegen, werden in Klammern dargestellt.
	Bei Input-mode Integer muß folgendes eingeben werden: 27,51,24 <Return>, was der ASCII-Codierung entspricht.
Pline: /(Esc)K(24)(1)/	Eingabe: /<Esc>K<Ctrl-X><Ctrl-A>/. Die zwei letzten Zeichen können beliebig sein, da diese vom Programm aus abhängig von den Teilbildern in horizontaler Richtung gesetzt werden.
Correct?	y
Another mode?	y
Control block #2:	Ein zweiter Grafikmodus
Pinit: /(Esc)3(24)/ Pline: /(Esc)L(24)(1)/	Siehe oben Siehe oben, statt K eben ein L
Correct? Another mode?	y n
Bit 0 = L(ow, H(igh needle? Name of printer: Graphic bytes: I(nteger, S(tring? Position of graphic bytes: Init code for interface?	L Epson FX-80 <Return> I 3 <Return> <Return> bei Super-Serial <Ctrl-I> Z <Return>

3.1.22. Verkleinern
Dieser Menüpunkt verkleinert die gesamte Grafik um (maximal) den Faktor [PicMax.X, PicMax.Y], wobei PicMax.X die Anzahl der Teilbilder in X-Richtung bedeutet, die im Graphic-Sheet-Fenster grafisch zu erkennen sind. Dasselbe gilt für PicMax.Y. Hat man seine Verkleinerung begutachtet, so kann mit dem unter Punkt 2.4. beschriebenen Verfahren wieder ins Menü-Modul gewechselt werden.

3.1.23. Hilfe
Nach Aufruf von Help kann jedes andere Piktogramm angewählt werden, um Hilfen zu den einzelnen Menüpunkten zu erhalten. Durch erneutes Anwählen von Help kann dieser Modus wieder verlassen werden.

3.1.24. bis 3.1.31. Erweiterung
Hier sind weitere Piktogramme für Erweiterungen vorgesehen.

3.1.32. Verlassen (Exit)
Untermenü:
1: Update the workfile and leave
2: Purge the workfile and leave
memavail: 4916 bytes
codeavail: 8232 bytes

1. Die augenblickliche Arbeitsdatei wird auf Diskette gerettet, und der Editor wird verlassen.
2. Die augenblickliche Arbeitsdatei wird gelöscht, und der Editor wird verlassen.
Die memavail- und codeavail-Angaben können von Rechner zu Rechner schwanken, je nachdem, ob weitere Treiber angeschlossen wurden oder der CRUNCHER benutzt wird. Der CRUNCHER befindet sich auf der Boot-Diskette unter dem Namen SYSTEM.STARTUP.

3.2. Text-Input/Output-Window

Wie der Name schon sagt, werden hier alle Aktionen abgehandelt, die eine Textein- oder -ausgabe verlangen. Ebenso werden dort die Untermenüs dargestellt, die man sich als Pulldown-Menüs wie beim Macintosh vorstellen kann und die man mittels Zeiger oder Zifferntasten auswählen kann.

3.3. Graphic-Sheet

Das Graphic-Sheet-Fenster hat primär zwei Funktionen:
1. Festlegen der Größe der gesamten Grafik durch Angabe des rechten unteren Teilbildes.
2. Orientierungshilfe zur Bestimmung der Position des aktuellen Teilbildes.
Ein Teilbild hat die Größe eines Apple-Bildschirmes (280x192 Pixeln) und wird im Graphic-Sheet-Fenster durch ein 4x3 Rechteck dargestellt. Im Grundzustand befindet sich dieses Rechteck an der linken oberen Ecke und wird von einer weißen Linie umrahmt.
Möchte man z.B. eine 3x2 große Grafik bearbeiten, so bewegt man den Zeiger auf dieses Rechteck und „zieht" dieses auf die 3. Position in horizontaler und auf die 2. Position in vertikaler Richtung. Anschließend läßt man den Knopf los. Ein weißer Rahmen zeigt dann die neuen Begrenzungslinien und damit die Größe der gesamten Grafik. Man kann nun das weiße Rechteck, das in der rechten unteren Ecke steht, wieder „nehmen" und nach links oben

bringen, um mit dem 1. Teilbild das Editieren zu beginnen, indem man z.B. den Bleistift wählt und ins Draw-Modul wechselt.
Wechselt man vom Draw-Modul ins Menü-Modul, so wird dieses Rechteck immer auf den neuesten Stand gebracht, falls man im Draw-Modul einen Seitenwechsel mittels „Paging-Befehl" vorgenommen hat. Näheres dazu im folgenden Abschnitt.

4. Komponenten des Draw-Moduls

Dieser Abschnitt beschreibt alle Möglichkeiten des Draw-Moduls. Die verschiedenen Hilfsmittel wie Bleistift usw. werden im Menü-Modul ausgewählt, und anschließend wird mittels Doppelklick oder geschlossener Apfel-Taste ins Draw-Modul gewechselt.

4.1. Verschiedene Hilfsmittel

4.1.1. Diskette
Diesem Zeichen entspricht im Draw-Modul keine Funktion.

4.1.2. Bleistift
Der Cursor ist jetzt ein Bleistift. Solange der Knopf der Maus gedrückt bleibt und man sich bewegt, wird in der ausgewählten Farbe – meist Schwarz – gezeichnet. Bewegt man sich per Tastatur mit Repeat-Faktor 1, so kann pixelgenau gezeichnet werden.

4.1.3. Pinsel
Im Menü-Modul wurde eine Pinselform ausgewählt, die jetzt als Cursor erscheint. Solange man den Mausknopf drückt und sich bewegt, wird der Hintergrund mit der Farbe Schwarz gefüllt. Dabei ist zu beachten, daß der Rechner bei großen Pinselformen der Bewegung nicht schnell genug folgen kann. Man sollte sich dann nur langsam mit der Maus bewegen oder die Tastatur zu Hilfe nehmen.

4.1.4. Eindimensionales Achsenkreuz
Hier muß nur eine Linie (die X-Achse) von links nach rechts gezogen werden. Die Skalierung und Numerierung wird entsprechend der Voreinstellung im Menü-Modul vorgenommen, sobald der Knopf der Maus losgelassen wurde.

4.1.5. Zweidimensionales Achsenkreuz
Hier müssen 2 Fälle unterschieden werden:
1. Set (x,y) input steht auf off
2. Set (x,y) input steht auf on

1. Dies ist der Standardfall, d.h. es müssen zuerst beide Achsen gezogen werden, bevor irgendwelche Punkte eingetragen werden können. Hat man diese gezogen, so wird entsprechend der Voreinstellung automatisch eine Skalierung und/oder eine Numerierung vorgenommen.
2. Hat man die Achsen gezeichnet, so können Punkte, die dann mit Geraden verbunden werden, wie folgt eingegeben werden:
Man geht zurück ins Menü-Modul und wählt Punkt 3 an. Damit schaltet man die (x,y)-Eingabe ein. Dann wechselt man wieder ins Draw-Modul. Ein Klicken mit dem Mausknopf bringt die Meldung „New point? [y/n]", die mit „y" zu beantworten ist. Ein <Return> oder <Space> gilt ebenfalls als „y". Man beachte, daß beim Klicken der Mausknopf mindestens so lange gedrückt bleibt, bis die Zeit für einen eventuellen Doppelklick verstrichen ist. Erst dann darf der Knopf wieder losgelassen werden.
Es erscheint ein „X:" und ein Cursor. Mit einer Gleitpunktzahl wird die X-Komponente des Punktes eingegeben; sie muß mit <Return> beendet werden. Anschließend wird nach der Y-Komponenten gefragt; sie wird analog eingegeben.
Der Punkt wird in das Koordinatensystem eingetragen und – falls es nicht der erste war – mit dem zuletzt eingegebenen Punkt durch eine Gerade verbunden.

4.1.6. Bewegungsparameter
Der Bewegungsparameter besitzt im Draw-Modul keine Funktion.

4.1.7. Linien
Hier können Linien in der ausgewählten Farbe und Strichstärke gezogen werden. Der Anfangspunkt wird durch Drücken des Knopfes bestimmt. Beim „Ziehen" verbindet ein „Gummiband" den Anfangspunkt mit der aktuellen Cursor-Position.

4.1.8. Polygone
Beim Zeichnen von Polygonen entspricht der Anfangspunkt einer Seitenlinie dem Endpunkt der zuletzt gezeichneten Linie.

4.1.9. Rechtecke
Durch „Ziehen" wird ein Rechteck mit der ausgewählten Strichstärke konstruiert. Sein Rand wird in der gewählten Farbe gezeichnet.

4.1.10. Gefüllte Rechtecke
Hier wird der Rand des Rechtecks mit der gewählten Strichstärke in Schwarz gezeichnet. Das Innere des Rechtecks wird mit der gewählten Farbe ausgefüllt.

4.1.11. Kreise
Der Rand des Kreises wird mit der gewählten Strichstärke in der entsprechenden Farbe gezeichnet.

4.1.12. Funktionstasten (record keys)
Zur Definition von Funktionstasten siehe Kapitel 3.1.12. Hier soll ein Anwendungsbeispiel vorgestellt werden:
Es sollen mehrere Rechtecke des gleichen Formats (30x20 Pixeln) gezeichnet werden. Dazu wählt man im Menü-Modul das Piktogramm für Rechtecke aus und wechselt anschließend ins Draw-Modul. Nun gibt man ein:

<Esc> R <Esc> 1 <offener Apfel drücken> 30 <Pfeil rechts> 20 <Pfeil unten> <offener Apfel loslassen> <Esc> 1

Damit ist die Funktionstaste <Esc> 1 definiert. Man kann nun den Cursor an eine andere Stelle placieren, und bei Eingabe von <Esc> 1 wird das gleiche Rechteck an diese Stelle gezeichnet.
Läßt man die Repeat-Faktoren „30" und „20" im obigen Beispiel weg, ergeben sich noch andere reizvolle Variationen. Man kann z.B. mit dem Bleistift Figuren zeichnen, während man eine Funktionstaste aufnimmt. Wählt man vor dem Abspielen einen anderen Repeat-Faktor, so wird die Figur automatisch vergrößert oder verkleinert, je nachdem, ob der neue Repeat-Faktor größer oder kleiner als derjenige beim Aufnehmen der Funktionstaste ist.

4.1.13. Text
Nachdem man im Draw-Modul die Text-Attribute gesetzt hat, können damit Zeichnungen beschriftet werden. Der Cursor besteht aus einer senkrechten Linie, wie sie auch vom GEM oder Apple-Macintosh bekannt ist. Um Texte einzugeben, klickt man an der einzugebenden Stelle mit dem Knopf. Jetzt können Texte geschrieben werden. Beim Zurücktippen kommt der alte Hintergrund wieder zum Vorschein. Ein <Return> beendet eine Zeile und stellt den unsichtbaren Schreibcursor um 8 Bits – das entspricht der Höhe eines Zeichens – nach unten an die horizontale Stelle, an welcher der Klick mit dem Knopf ausgeführt wurde. Ändert man den Repeat-Faktor bespielsweise auf 10, so wird der Zeilenabstand nach jedem <Return> entsprechend vergrößert.
Verlassen werden kann das Draw-Modul wie üblich mit einem Doppelklick oder mit der geschlossenen Apfel-Taste.
Hat man keine Maus zur Verfügung oder möchte man sich mit der Tastatur pixelgenau fortbewegen, kann der Schreibmodus mittels <Esc> <Esc> verlassen werden. Dann haben alle Tasten anstelle der Texteingabe wieder die üblichen Funktionen. So erzeugt ein „I" nicht den Text „I", sondern bewegt den Cursor nach oben.

4.1.14. Zeichensätze
Wurde ein bestimmter Zeichensatz im Menü-Modul gewählt, so kann mit diesem im Draw-Modul Text eingegeben werden, wie unter 4.1.13. beschrieben ist.

4.1.15. Radiergummi
Mit dem Radiergummi lassen sich Fehler wieder ausradieren. Beim Ziehen wird der Hintergrund mit der Grundfarbe Weiß gefüllt.

4.1.16. Lupe
Beim Wechseln ins Draw-Modul wird ein Rechteck der Größe 40x24 Pixeln als Cursor dargestellt. Der Inhalt dieses Rechtecks entspricht dem Ausschnitt der Vergrößerung. Man fährt nun den Cursor an die Stelle, die vergrößert werden soll, und drückt den Knopf. Die entsprechende Vergrößerung wird erzeugt, und der Cursor verwandelt sich in einen Bleistift. Nun können einzelne Pixeln bequem editiert werden. Drückt man den Knopf auf einem schwarzen Pixel, so wird dieses weiß und entsprechend umgekehrt. Will man zur Lupe zurück, so gibt man <Esc><Esc> ein.
Da der vergrößerte Ausschnitt in bezug auf das Gesamtbild relativ klein ist, wird die Möglichkeit gegeben, den Ausschnitt im vergrößerten Modus zu verschieben. Dazu gibt man ein „P" für Paging (Seitenwechsel) ein. Der Bleistift weicht einer Hand als Cursor, der anzeigt, daß nun das „Papier" unter der Hand verschoben werden kann. Dazu zieht man einfach die Maus um einige Zentimeter in die Richtung, um die verschoben werden soll. Möchte man zurück zum Bleistift, so gibt man <Esc><Esc> ein.
Zu jeder Zeit kommt man zurück ins Menü-Modul durch Drücken der geschlossenen Apfel-

Taste oder durch zweimaliges Drücken der offenen Apfel-Taste.

4.1.17. Strichstärke
Die im Menü-Modul ausgewählte Strichstärke bezieht sich auf Linien, Polygone, Rechtecke, gefüllte Rechtecke und Kreise.

4.1.18. Farben
Die im Menü-Modul gewählte Farbe bezieht sich auf Linien, Polygone, Rechtecke, Kreise und das Innere gefüllter Rechtecke.

4.1.19. Editorrahmen zum Verschieben und Kopieren
Wurde im Menü-Modul Punkt 1, also „Move only" gewählt, so kann nur verschoben werden. Dazu zieht man sich ein Rechteck, dessen Inhalt nach Loslassen des Knopfes gelöscht wird. Nun fährt man in das Rechteck hinein und zieht es an die gewünschte Stelle. Beim Loslassen des Knopfes wird der gelöschte Inhalt an die neue Stelle gesetzt.
Mit „Move & copy" können Teile aus Bildern kopiert werden. Man geht wie bei „Move only" vor mit dem Unterschied, daß der Inhalt des Rechtecks nach dem Definieren nicht gelöscht wird.
Hat man versehentlich ein falsches Rechteck definiert, so braucht man nur außerhalb dieses Rechtecks zu klicken, und der Originalinhalt wird wiederhergestellt. Man kann auch ins Menü-Modul wechseln.

4.1.20. Mülleimer
Dieses Zeichen hat im Draw-Modul keine Funktion.

4.1.21. Drucken
Keine Funktion im Draw-Modul.

4.1.22. Verkleinern
Keine Funktion im Draw-Modul.

4.1.23. Hilfe
Keine Funktion im Draw-Modul.

4.1.24. bis 4.31. Erweiterung
Diese Bereiche sind für zukünftige Erweiterungen reserviert.

4.1.32. Verlassen (Exit)
Keine Funktion im Draw-Modul.

4.2. Seitenwechsel

Um einen Seitenwechsel herbeizuführen, muß die gesamte Grafik mindestens aus 2 Teilbildern bestehen (siehe 3.3.). Dies kann im Menü-Modul definiert werden. Ein Beispiel soll den Seitenwechsel verdeutlichen.
Wir definieren 2 Teilbilder in X-Richtung im Menü-Modul, indem wir auf das 2. Teilbild rechts neben dem weißen Rechteck im Graphic-Sheet-Fenster klicken. Anschließend wählen wir wieder das erste Teilbild an, indem wir in dieses hineinklicken. Damit haben wir eine Grafik aus 2x1 Teilbildern definiert und starten mit dem Editieren im linken Teilbild. Wir wählen den Bleistift aus und wechseln durch Doppelklick ins Draw-Modul. Auf der rechten Bildschirmhälfte lassen wir unseren Ideen freien Lauf und zeichnen irgend etwas. Anschließend bewegen wir den Bleistift ganz an den rechten Rand und geben ein „P" für Paging (Seitenwechsel) ein.
Nun passiert folgendes: Der augenblickliche Bildschirminhalt wird auf Diskette gerettet und um eine halbe Bildschirmseite nach links verschoben. Dann wird die linke Hälfte des rechten Teilbildes geladen und in die rechte Bildschirmhälfte hineinkopiert. Man hat nun das Gefühl, sich zwischen den beiden Teilbildern zu befinden, und kann seine Zeichnung nahtlos fortsetzen, ohne den Anschluß an das linke Teilbild zu verlieren. Ein weiterer Seitenwechsel am linken oder am rechten Rand hat die entsprechende Wirkung. Man verschiebt damit quasi ein Fenster über der gesamten Grafik. Im Menü-Modul kann man das Piktogramm für „Show whole picture" anwählen und erhält eine entsprechende Verkleinerung der gesamten Grafik.

Entsprechendes gilt auch für den oberen und unteren Rand. Hier läßt sich das Fenster in der Vertikalen verschieben. Befindet es sich zwischen zwei Teilbildern, so kann man erst ins Menü-Modul zurückkehren, wenn man sich für ein Teilbild entschieden und in die entsprechende Richtung gewechselt hat. Das Fenster muß also dann immer genau auf einem Teilbild stehen.
Ist ein Seitenwechsel nicht erlaubt, z.B. weil der rechte Rand der gesamten Grafik erreicht wurde, so ertönt die Glocke als Fehlermeldung.

5. Das Show-Modul

Dieses Modul hat nur eine einzige Funktion: Erzeugen einer Verkleinerung der gesamten Grafik, falls diese mindestens aus 2 Teilbildern besteht. Es wird aufgerufen, indem man im Menü-Modul das entsprechende Piktogramm auswählt (siehe 3.1.22.). Hat man seine Verkleinerung begutachtet, so kann mit den üblichen Methoden wieder ins Menü-Modul gewechselt werden.

6. Das Print-Modul

Dieses Modul besorgt den gesamten Druckvorgang für die Gesamtgrafik und für Teilbilder. Es wird durch Auswählen des entsprechenden Piktogramms aufgerufen (siehe 3.1.21.) und automatisch nach Beendigung des Druckvorganges verlassen.

7. Zusätzliche Tastaturfunktionen

In diesem Kapitel werden zusätzliche Funktionen beschrieben, die mit der Tastatur eingegeben werden können. Die Tasten zur Steuerung des Cursors wurden ja schon in Kapitel 2 vorgestellt.
Fast alle Zusatzfunktionen werden durch <Esc> eingeleitet, d.h. Drücken der <Esc>-Taste mit anschließendem Drücken einer der folgenden Tasten.

7.1. Globale Funktionen

Diese Funktionen können von jedem Modul aus aktiviert werden.
<Esc>G schaltet den Rechner auf Grafikmodus um. Normalerweise befindet sich der Rechner während des Programmablaufs immer im Grafikmodus. Es können aber Fälle auftreten, bei denen der Rechner in den Textmodus zurückschaltet, z.B. wenn eine UltraTerm-Karte

der Firma Videx angeschlossen wurde und bei einem Zugriff auf Diskette vorausgetippt wird. Diese Tasten werden über die Prozedur „Console-Checking" abgefangen, die wiederum in die Firmware der Karte springt und ein Umschalten auf Textmodus erzwingt.
<Esc>Q verläßt das Programm, ohne irgendwelche Files zu schließen. Betätigt man diese Funktion im Laufzeitsystem, so startet das Programm von neuem, während man im Entwicklungssystem in die oberste Kommando-Ebene gelangt.
<Esc>W sollte nur für „Special Effects" benutzt werden und wurde nur eingebaut, um diese Dokumentation besser herstellen zu können. Der augenblickliche Inhalt des Bildschirms wird in das aktuelle Teilbild geschrieben und dieses somit gelöscht. Damit kann man z.B. eine gesamte Grafik nach dem Verkleinern in ein Teilbild kopieren. Das Teilbild, in das hineinkopiert werden soll, muß schon existieren, d.h. man muß sich mindestens einmal mit diesem Teilbild als aktuellem Bild im Draw-Modul befunden haben. Versuche mit dieser Funktion sollten nur mit nutzlosen Grafiken vorgenommen werden.

7.2. Funktionen im Menü-Modul

<Esc>N oder N erzwingt einen Newcommand-Befehl, d.h. das Textfenster wird mit dem Untermenü des gewählten Piktogrammes aktualisiert. Dies wird dann benötigt, wenn das Untermenü durch Eingaben des Benutzers oder Meldungen des Systems zerstört wurde. Das gleiche kann auch mit einem erneuten Klicken auf das gleiche Piktogramm erreicht werden.
<Esc>U schaltet die Spezialbehandlung einer angeschlossenen UltraTerm-Karte aus oder ein. Diese Spezialbehandlung, welche die Tastaturabfrage nicht über das eingebaute Console-Checking, sondern direkt über die Hardware-Adresse des Apple erledigt, ist bei angeschlossener Karte anfangs eingeschaltet, d.h. Console-Checking ist nur bei Diskettenzugriffen aktiv, wenn vorausgetippt wird.

7.3. Funktionen im Draw-Modul

<Esc>C löscht den augenblicklichen Bildschirminhalt mit der Grundfarbe Weiß.
<Esc>P führt einen Seitenwechsel durch. (Siehe 4.2.)
<Esc>U ist eine eingeschränkte Undo-Funktion. Der Bildschirminhalt wird mit dem Teilbild geladen, das bestand, nachdem man in das Draw-Modul gewechselt hatte. Wechselt man z.B. ins Draw-Modul und verzeichnet sich anschließend, so kann man dies mit <Esc>U wieder rückgängig machen. Das Bild, das dann geladen wird, ist immer dasjenige, das man auf dem Bildschirm hatte, als man ins Draw-Modul gelangt war. Hat man sich also zweimal verzeichnet, so kann man beide „Ausrutscher" nur gleichzeitig wieder rückgängig machen.

Anhang A: Sammeldiskette # 20

Auf der Sammeldiskette #20 mit dem Volume-Namen „PICEDIT:" befinden sich die untenstehenden Dateien außer SYSTEM.APPLE und SYSTEM.PASCAL, die Sie aus urheberrechtlichen Gründen selbst auf die hierfür reservierten Dummy-Files kopieren müssen. Dabei muß es sich um die 128K-Pascal-1.2-Files handeln.
Die Diskette läuft sofort mit Epson-Druckern. Für den Imagewriter muß im F(iler die Datei IMAGEWRITER mit dem T(ransfer-Befehl auf PRINTER.INFO[1] kopiert werden.
Nach dem Kopieren der zwei Pascal-Systemdateien kann die Diskette direkt gebootet werden.

```
PICEDIT:
SYSTEM.APPLE        32  Datafile Dummy!
SYSTEM.PASCAL       45  Codefile Dummy!
SYSTEM.MISCINFO      2  Datafile neu!
SYSTEM.CHARSET       2  Datafile neu!
ASCII.FONT           2  Datafile
GERMAN.FONT          2  Datafile
MATH.FONT            2  Datafile
GREECE.FONT          2  Datafile
SUPER.SUB.FONT       2  Datafile
MENU.GRAF           32  Graffile
PRINTER.INFO         1  Infofile
EPSON                1  Infofile
IMAGEWRITER          1  Infofile
SYSTEM.ATTACH        4  Codefile
CRUNCHER.CODE        4  Codefile
SYSTEM.STARTUP       3  Codefile
SYSTEM.LIBRARY     103  Datafile
ARROWS.KEYS          4  Datafile
DESIGNER.CODE        6  Codefile
```

Anhang B: Literatur

Apple Pascal Operating System Reference Manual
Apple Pascal Language Reference Manual
AppleMouse II User's Manual
Apple IIe Reference Manual
Apple Super-Serial-Card; Installation and Operating Manual
Imagewriter Reference Manual
Epson FX-80 Reference Manual
UltraTerm Installation and Operating Manual
Barry Haynes: Attach-BIOS Document for Apple II Pascal 1.1

Einblicke in Logo

Apple Logo II

von Ernst F. Haas

Der folgende Artikel soll an einigen einfachen Beispielen typische Merkmale und Eigenschaften der Programmiersprache Logo aufzeigen. Ganz bewußt wird dabei auf Beispiele aus der „Igel"(-Turtle)-Grafik verzichtet, um nicht der mit Sicherheit falschen Meinung weiteren Vorschub zu leisten, bei Logo handle es sich um eine „Kinder-Sprache", mit der allenfalls harmlos-einfache Figurengeometrie betrieben werden könne, obwohl – und dies nur am Rande – manches von dem, was in Logo tatsächlich sehr einfach und auch schon von Kindern programmiert werden kann, in anderen Programmiersprachen erhebliches Kopfzerbrechen bereiten kann.

Mit diesem Artikel kann nicht systematisch in Logo eingeführt werden. Auch wird von vielen Möglichkeiten, die die zugrundegelegte Logo-Version von Apple bietet, kein Gebrauch gemacht (hierzu zählt z.B. der ganze Befehlskomplex zur Dateiverwaltung und -bearbeitung). Von manchen Fähigkeiten wird auch höchstens beiläufig die Rede sein. Im Vordergrund steht auch nicht die Absicht, besonders elegante Programmlösungen vorzustellen; vielmehr sollen Einblicke in die Struktur, die „Philosophie" von Logo gegeben werden, soll gezeigt werden, über welch mächtiges Konzept diese Programmiersprache verfügt. Im einzelnen bedeutet dies eine eingegrenzte Auswahl, die sich mit den folgenden Stichworten beschreiben läßt:

– Modularität/Prozedurkonzept
– Rekursion
– Prozeduren als Funktionen
– Datentyp Liste
– Befehle RUN, TEXT, DEFINE

Diese Auswahl wurde unter der Annahme getroffen, daß der Leser Kenntnisse in BASIC und/oder Pascal besitzt und daß vor diesem Hintergrund die Charakteristika von Logo besonders deutlich werden.

Bei der Lektüre sollte außer acht bleiben, daß Logo ein *interpretierendes* System ist und sich in puncto Schnelligkeit z.B. mit Pascal nicht messen kann. Der Vorteil, während der Programmerstellung nicht ständig durch Compilierzeiten aufgehalten zu werden, ist wohl beabsichtigt; außerdem soll bereits ein Logo-Compiler auf dem Markt sein, der dem fertigen Programm auch seine nötige Ausführungsgeschwindigkeit verleihen wird. Auch der oft gehörte Vorwurf, daß wegen des relativ hohen Speicherbedarfs die Lauffähigkeit mancher Programme doch sehr stark beeinträchtigt werde, trifft nicht die eigentliche Idee Logo, sondern ist eher als eine Hardware-Schwäche zu betrachten, die überdies für die zur Zeit auf den Markt drängende Rechnergeneration kaum mehr zutrifft. (Das vom Verfasser verwendete Apple Logo II erfordert den IIc bzw. den IIe mit zusätzlichen 64K und meldet nur noch in ganz extremen Situationen einen vollen Speicher.)

1. Prozeduren als Programm-Bausteine

Ähnlich wie in Pascal ist ein Logo-Programm aus einzelnen, meist sehr kleinen Unterprogrammen, *Prozeduren* genannt, aufgebaut. Das Zusammenwirken dieser einzelnen Prozeduren, die einfach durch ihren – vom Programmierer erhaltenen – Namen aufgerufen werden, ergibt schließlich den Programmablauf. Es gehört jedoch zur Grundidee von Logo, daß die einzelnen Prozeduren, anders als in Pascal, unabhängig voneinander lauffähig und aufrufbar sein können.

Die Fülle der Logo-Grundbefehle (sog. *Logo-Primitives* oder *Logo-Grundwörter*) steht über einen einfachen Aufruf direkt oder auch innerhalb von Prozeduren zur Verfügung, d.h. daß z.B. der Befehl ROUND, der als Eingabe natürlich eine Zahl erfordert, unmittelbar nach dem Start des Logo-Systems im sog. *Direkt-Modus* verwendet werden kann, also

```
PRINT ROUND 13.321
--> 13
```

(nach RETURN wird der Befehl sofort ausgeführt). Er kann aber auch in genau derselben Weise innerhalb einer Prozedur verwendet werden; seine Ausführung ist dann lediglich durch die Reihenfolge der dort enthaltenen Einzelbefehle festgelegt.

Die Eingabe zu ROUND muß nicht unbedingt eine ausdrücklich notierte Zahl sein, sondern kann auch ein Logo-Befehl (genauer: eine Logo-Funktion – dazu unten mehr) sein, der seinerseits eine bestimmte Zahl zum Ergebnis hat, z.B. SQRT (Quadratwurzel):

```
PRINT SQRT 3.24
--> 1.8
PRINT ROUND 1.8
--> 2
```

läßt sich folgendermaßen aufrufen:

```
PRINT ROUND SQRT 3.24
--> 2
```

Genau in derselben Weise können auch eigene geeignete Prozeduren bzw. deren Namen, z.B.„EIGENE.PROZ" mit irgendeiner Wirkung auf eine eingegebene Zahl eingesetzt werden. Syntaktisch wird dabei keinerlei Unterschied gemacht:

```
PRINT ROUND EIGENE.PROZ irgendeine Zahl
```

Der mit Pascal vertraute Leser wird feststellen, daß die Beispiele dort mit *functions* programmiert werden, und in der Tat ist es auch bei Logo möglich, Prozeduren als Funktionen zu schreiben.

Wir haben jetzt allerdings schon etwas vorgegriffen, ohne die nötigen Grundlagen genauer betrachtet zu haben. Immerhin sollte deutlich geworden sein, weshalb das Programmieren in Logo auch als „Spracherweiterung" bezeichnet wird (vgl. Literatur-Angabe), was heißen soll, daß der Satz der vorgegebenen Logo-Grundwörter mit neuen, eigenen Befehlen, die durch entsprechende Prozeduren definiert wurden, beliebig erweitert werden kann, wobei der Gebrauch des eigenen Befehlssatzes sich in keiner Weise von dem der Grundwörter unterscheidet.

2. Einfache Prozedur-Beispiele

Doch nun etwas systematischer einige Beispiele für und zum Umgang mit Prozeduren. Ihr äußeres Kennzeichen besteht darin, daß ihre erste Zeile mit **TO** beginnt, gefolgt von dem gewählten **Prozedurnamen** und evtl. einer Aufzählung der für die Abarbeitung notwendigen Parameter, deren Kennzeichen wiederum der vorgestellte Doppelpunkt ist. Eine Prozedur ist mit **END** abzuschließen.

Zur besseren Orientierung werden im folgenden *alle Logo-Primitives* mit *GROßEN*, *alle selbstgewählten Ausdrücke* mit *kleinen* Buchstaben dargestellt.

```
TO halbiere :zahl
MAKE "haelfte QUOTIENT :zahl 2
PRINT :haelfte
IF :haelfte = INTQUOTIENT :zahl 2..
..[PRINT [gerade zahl] STOP]
PRINT [ungerade zahl]
END
```

Die Punkte gehören natürlich nicht zur Prozedur, sondern deuten nur an, daß der Text in *eine* Logo-Zeile gehört. Bei Zeilenüberschreitungen dieser Art deutet Logo mit einem Ausrufezeichen am Zeilenende selbständig an, daß der nachfolgende Text noch zur selben (logischen) Zeile zu zählen ist.

Der Aufruf der Prozedur erfolgt über die Nennung ihres Namens, gefolgt von einer Angabe für den Parameter „zahl", demnach liefert

```
halbiere 6Ø
--> 3Ø.Ø
    gerade zahl

halbiere 67
--> 33.5
    ungerade zahl
```

Die Prozedur zeigt u.a., wie in Logo *Variablen* vereinbart werden: Die in der Titelzeile vorkommende Variable „zahl" ist mit ihrer Nennung für diese und nur für diese Prozedur lokal vereinbart. Die in der zweiten Zeile mittels MAKE definierte Variable „haelfte" existiert global und wird im Arbeitsspeicher bis zur Löschung beständig abgelegt, d.h. nach ihrer Einführung kann auch von anderen Prozeduren auf sie zugegriffen werden (allerdings kann durch Einfügen von „LOCAL "haelfte" vor MAKE auch sie als nur lokale Variable erzeugt werden, die dann nach dem Ablauf von „halbiere" nicht mehr existiert).

Interessant ist der auch in der Schreibweise sichtbare unterschiedliche Gebrauch von „haelfte": Logo unterscheidet streng zwischen dem *Namen* und dem *Wert* einer Variablen. In Zeile 2 der Prozedur „halbiere" wird einer Variablen mit dem Namen „haelfte", geschrieben als Wort „haelfte" (*Wörter* werden mit vorausgehendem Anführungszeichen gekennzeichnet, deshalb „MAKE "haelfte"), der Wert zugeordnet, der sich als Quotient aus der eingegebenen Zahl und 2 ergibt. In der darauffolgenden Zeile soll mittels PRINT natürlich nicht das *Wort*, sondern eben der *Wert*, den das Wort repräsentiert, ausgedruckt werden. Dies wird durch den vorangestellten Doppelpunkt erreicht; deshalb also „PRINT :haelfte". Dasselbe Resultat ist zu erzielen, wenn dem Wort „"haelfte" das Logo-Grundwort THING vorangestellt wird; auch dadurch erhält man den Wert einer Variablen bestimmten Namens: „THING "haelfte" ist demnach gleichbedeutend mit „:haelfte".

INTQUOTIENT :zahl 2 ergibt den Ganzzahlquotienten seiner beiden Eingaben; bei Erfüllung der angegebenen Bedingung wird der Inhalt der nachfolgenden eckigen Klammer ausgeführt, also der Ausdruck von „gerade zahl" einschließlich des die Prozedur abbrechenden Befehls STOP. (Die eckige Klammer bezeichnet den äußerst wichtigen Datentyp *Liste*, auf den noch ausführlicher eingegangen wird.) Trifft die IF-Bedingung nicht zu, wird die Zeile einfach übersprungen und zur nächsten Zeile übergegangen.

Prüfwörter

Logo verfügt über eine Reihe von Grundwörtern, die zur Überprüfung der Korrektheit der verwendeten Daten herangezogen werden können. Um im obigen Beispiel bei falscher Eingabe (z.B. „halbiere "peeker") eine Fehlermeldung zu vermeiden, könnten die beiden folgenden Zeilen an zweiter und dritter Stelle in die Prozedur eingefügt werden. Da Logo einen sehr guten Editor besitzt, sind Änderungen an Prozeduren bequem möglich.

Die verwendeten *Prüfwörter* „NUMBERP" (steht für „numberproperty" = Zahleigenschaft) und „EQUALP" (für „equalproperty") liefern je nach Eingabe die Resultate „TRUE" oder „FALSE". Mit „COUNT" wird die Anzahl der Buchstaben eines Wortes bzw. der Ziffern einer Zahl (oder auch die Anzahl der Elemente einer Liste) ausgegeben. „INT" bestimmt den ganzzahligen Anteil einer Zahl:

```
IF (NOT NUMBERP :zahl) [meldung.1]
IF (NOT EQUALP (COUNT :zahl) (COUNT..
.. INT :zahl)) [meldung.2]
```

Dabei sind „meldung.1" und „meldung.2" selber wieder (einfache und eben zur Demonstration erstellte) Prozeduren, die vielleicht so aussehen könnten:

```
TO meldung.1
PRINT [nur zahlen eingeben]
abbruch
END

TO meldung.2
PRINT [nur ganze zahlen eingeben]
abbruch
END
```

In beiden Prozeduren wird erneut eine vom Benutzer formulierte weitere Prozedur aufgerufen:

```
TO abbruch
THROW "TOPLEVEL
END
```

„THROW "TOPLEVEL" bewirkt die Rückkehr des Systems in den Direktausführungsmodus, bricht also die Abarbeitung der Prozedur ab.

Anhand der genannten Beispiele wird einsichtig, wie flexibel man mittels des Prozedurkonzepts in Logo programmieren kann. Für manchmal erst im Laufe der Programmentwicklung auftretende Teilprobleme lassen sich Prozeduren schreiben, die dann innerhalb anderer Prozeduren an geeigneter Stelle durch einfache Angabe ihres Namens (einschließlich evtl. Parameter) aufgerufen werden. Es ist von nicht unerheblicher Bedeutung, daß jede Prozedur vor ihrer Einbettung in andere für sich allein ausgetestet werden kann.

Faßt man endlich auch noch die beiden Zeilen zur Überprüfung der Eingabe für unser Ausgangsbeispiel „halbiere" in eine Prozedur, also

```
TO eingabe.ok? :z
IF (NOT NUMBERP :z) [meldung.1]
IF (NOT EQUALP (COUNT :z)..
..(COUNT INT :z)) [meldung.2]
END
```

dann läßt sich eine Prozedur „halbiere.neu“ auch wie folgt formulieren:

```
TO halbiere.neu :zahl
eingabe.ok? :zahl
halbiere :zahl
END
```

Der Aufruf von Prozeduren innerhalb von Prozeduren ist auch rekursiv möglich, d.h. Prozeduren können sich auch selbst wieder aufrufen. Dies soll zunächst anhand des „Halbierungs“-Beispiels demonstriert werden.

3. Rekursive Prozeduren

Die Prozedur „halbiere“ soll der Reihe nach für eine bestimmte Folge von Zahlen aufgerufen werden; Logos Möglichkeit der Rekursion bietet für Aufgabenstellungen solcher Art einen eleganten Lösungsweg:

```
TO halbiere.folge :anfangszahl :endzahl
IF :anfangszahl > :endzahl [STOP]
halbiere :anfangszahl
halbiere.folge (:anfangszahl + 1):endzahl
END
```

Die Funktionsweise ist leicht verständlich: Ein Aufruf der Prozedur braucht zwei Zahleneingaben, also z.B „halbiere.folge 10 20“. Da die Bedingung zur Beendigung der Prozedur (noch) nicht zutrifft, wird „halbiere 10“ ausgeführt und anschließend „halbiere.folge“ erneut aufgerufen, wobei nun allerdings statt der Anfangszahl 10 eine um 1 erhöhte Anfangszahl, also 11, die Stelle von „:anfangszahl“ einnimmt. Der zweite Aufruf heißt nunmehr „halbiere.folge 11 20“ usw. Der Prozedurlauf endet beim Aufruf „halbiere.folge 21 20“.

Soll die Folge immer mit derselben Anfangszahl gestartet werden, z.B. mit 1, dann kann natürlich auch diese rekursive Prozedur in eine weitere Prozedur eingefügt werden, in der die 1 als feststehende Anfangszahl schon eingebaut ist:

```
TO halbiere.folge.neu :letzte.zahl
halbiere.folge 1 :letzte.zahl
END
```

Einen interessanten Einblick, wie Logo mit Rekursionen fertig wird, erhält man beim Vergleich der beiden folgenden Beispiele:

```
TO zaehle.a :zahl
IF :zahl = 0 [STOP]
TYPE :zahl
zaehle.a (:zahl - 1)
END
```

```
TO zaehle.b :zahl
IF :zahl = 0 [STOP]
zaehle.b (:zahl - 1)
TYPE :zahl
END
```

Der Aufruf „zaehle.a 5“ ergibt auf dem Bildschirm den Ausdruck „54321“ (im Gegensatz zu PRINT, wo jedesmal noch ein CR/LF hinzugefügt wird, unterbleibt dies bei TYPE), während beim Aufruf „zaehle.b 5“ der Ausdruck „12345“ erscheint

Dafür verantwortlich ist die jeweilige Stellung des rekursiven Aufrufs innerhalb der Prozedur. Die sog. *„last-line-“* oder auch *„tail-recursion“* in „zaehle.a“, von der auch in „halbiere.folge“ Gebrauch gemacht wurde, bewirkt lediglich wiederholende Durchgänge durch die entsprechende Prozedur, d.h. unmittelbar vor dem neuen Aufruf ist ein Durchgang beendet, mit dem erneuten Aufruf wird ein neuer Durchgang begonnen. Die Version „zaehle.b“ setzt den rekursiven Aufruf jedoch so, daß noch vor dem Abschluß eines Durchgangs bereits ein neuer Aufruf erfolgt. „zaehle.b 5“ bleibt also so lange „offen“, bis „zaehle.b 4“ abgeschlossen ist usw. Zum Schluß bleibt „zaehle.b 1“ unbeendet, bis „zaehle.b 0“ bei STOP regulär abbricht. Nun kann auch „zaehle.b 1“ mit TYPE 1 abgeschlossen werden usw., bis letztendlich, eben nach Abschluß aller „Unteraufrufe“, auch „zaehle.b 5“ mit dem Ausdruck von 5 zum Ende kommt.

In den nächsten Abschnitten wird von der Möglichkeit der Rekursion noch häufiger Gebrauch gemacht, so daß an dieser Stelle auf weitere Beispiele verzichtet werden kann. Es bleibt festzuhalten, daß Logo über bequem zu handhabende Möglichkeiten der Rekursion verfügt.

4. Prozeduren als Funktionen

Die bisher vorgestellten Prozeduren sind für eine unmittelbare Weiterverarbeitung ihrer Resultate ungeeignet. Ihre Ergebnisse können z.B. nicht an Variablen oder als Parameter weitergereicht oder innerhalb von Ausdrücken verwendet werden. Mit der oben vorgestellten Prozedur „halbiere“ sind also Formulierungen der folgenden Art nicht möglich (selbst wenn man die Meldungen „(un)gerade zahl“ wegließe): „SQRT (halbiere 50)“ oder „halbiere (halbiere 50)“. Für solche Zwecke müssen Prozeduren als *Funktionen* ausgedrückt werden, also als Vorschriften, die bestimmten Eingaben einen bestimmten Wert zuordnen, der an andere Prozeduren weitergegeben bzw. zurückgegeben werden kann.

Der entscheidende Logo-Befehl für funktionales Programmieren heißt OUTPUT. Wenn also die Prozedur „halbiere“ wie folgt umformuliert wird in

```
TO halbiere :zahl
MAKE "haelfte QUOTIENT :zahl 2
OUTPUT :haelfte
END
```

dann werden folgende Aufrufe möglich. Der einfache Aufruf, z.B. „halbiere 34“, führt bei der zugrundegelegten Logo-Version zur Fehlermeldung „you don't say what to do with ..ergebnis.. “. Andere Versionen reagieren hier freundlicher, deshalb die Voranstellung des PRINT-Befehls:

```
PRINT halbiere 34
PRINT SQRT (ROUND (halbiere 51))
PRINT halbiere (SQRT (halbiere..
..(RANDOM 20)))
```

Die Klammerung kann eigentlich unterbleiben, sie dient hier nur der besseren Lesbarkeit. Auch die die Eingabe überprüfende Prozedur „eingabe.ok?“ läßt sich, funktional formuliert, vorteilhafter einsetzen:

```
TO eingabe.ok? :z
IF (NOT NUMBERP :z) [OUTPUT "FALSE]
OUTPUT (COUNT :z) = (COUNT INT :z)
END
```

TRUE und FALSE gehören zum Logo-Wortschatz, werden also unmittelbar „verstanden“. Die dritte Zeile zeigt, daß die Gleichheitsaussage als Funktion mit den möglichen Werten TRUE bzw. FALSE verarbeitet wird. Außerdem ist zu erkennen, daß OUTPUT die Ausführung der Prozedur beendet. Danach wird zur aufrufenden Stelle zurückgekehrt, d.h. in der Regel zu einer Prozedur bzw. bei Rekursionen zum vorausgegangenen Aufruf derselben Prozedur.

„eingabe.ok?“ kann nun innerhalb eines Ausdrucks eingesetzt werden, wodurch nicht zuletzt die Lesbarkeit und Verständlichkeit einer Prozedur deutlich verbessert werden:

```
TO halbiere.neu :zahl
IF eingabe.ok? :zahl [OUTPUT halbiere
..:zahl] [PRINT [eingabefehler]]
END
```

Die gewählte Schreibweise der auf IF folgenden beiden eckigen Klammern, die zwei Listen darstellen, entspricht der Formulierung *IF-then-else*.

Typische Logo-Prozeduren

Im folgenden nun noch einige sich selbst erklärende Prozeduren, die für Logo typische Programmierlösungen zeigen.

```
TO dreifaches :z
OUTPUT :z * 3
END
```

```
TO div.durch.2 :x
MAKE "haelfte QUOTIENT :x 2
END

TO halbiere.ganz.neu :zahl
local "haelfte
IF eingabe.ok? :zahl [div.durch.2..
..:zahl] [PRINT [fehler]..
..abbruch]
OUTPUT :haelfte = INTQUOTIENT :zahl 2
END

TO gerade.zahl? :zahl
OUTPUT halbiere.ganz.neu :zahl
END

TO ungerade.zahlen :von :bis
IF :von > :bis [STOP]
IF NOT gerade.zahl? :von [PRINT :von]
ungerade.zahlen (:von + 1) :bis
END
```

Da Prozeduren insbesondere auch als Parameter für andere Prozeduren dienen können, wird auch die folgende Konstruktion möglich:

```
TO demo.bsp :x
ungerade.zahlen :x dreifaches :x
END
```

Unser Logo verfügt über keine Potenzfunktion; man wird sie sich also selber erstellen müssen:

```
TO x.hoch.pos.y :x :y
IF OR (:x = 1) (:y = 0) [OUTPUT 1]
OUTPUT :x * x.hoch.pos.y :x (:y - 1)
END

TO x.hoch.neg.y :x :y
OUTPUT 1 / x.hoch.pos.y :x (- :y)
END

TO x.hoch.y :x :y
IF :y < 0 [OUTPUT x.hoch.neg.y :x :y]
OUTPUT x.hoch.pos.y :x :y
END
```

Seine wahre Stärke zeigt Logo eigentlich erst im Zusammenhang mit dem von ihm unterstützten Datentyp *Liste*. Dieser Datentyp hebt Logo und seine Leistungen und Fähigkeiten auch deutlicher ab von z.B. Pascal, innerhalb dessen ja auch vieles von dem bisher Dargestellten, wenn auch mit gewissen Modifikationen, möglich ist.

5. Datentyp Liste

Rein äußerlich betrachtet ist eine *Liste* alles, was innerhalb eckiger Klammern steht. Liste ist also eine Zusammenfassung von einzelnen Elementen. Dabei ist jedoch die Art dieser Elemente völlig offen, die Zahlen, Wörter oder – und vor allem dies ist wichtig – auch selbst wieder Listen sein können. Innerhalb einer Liste ist man nicht auf einen Datentyp festgelegt, denn die Listenelemente können durchaus zu verschiedenen Typen gehören. Insbesondere existiert auch die Liste ohne Elemente, die *leere Liste*.

Natürlich gibt es eine Reihe von Logo-Grundwörtern, mit deren Hilfe Listen bearbeitet werden können. Die Anzahl dieser Befehle ist bei den verschiedenen Logo-Versionen unterschiedlich, jedoch erlaubt es das Prozedurkonzept, sich eine Sammlung von Prozeduren für Listenoperationen zu erstellen. Einheitlich vorhanden ist ein Stamm von wenigen einfachen, durch geeignete Kombination jedoch weitreichenden Funktionen, der nachfolgend vorgestellt wird. Der dabei mitverwendete Befehl SHOW beläßt im Gegensatz zu PRINT beim Ausdruck von Listen die sie einfassenden eckigen Klammern.

```
MAKE "bsp.liste [[apple logo]..
..fuer [apple [2 e] und [2c]]]

PRINT COUNT :bsp.liste
--> 3
(die Beispielliste hat 3 Elemente)
```

Den Zugriff auf das erste Listenelement erlaubt FIRST; auf das letzte Listenelement kann man mit LAST zugreifen:

```
SHOW FIRST :bsp.liste
--> [apple logo]

SHOW LAST :bsp.liste
--> [apple [2 e] und [2c]]

SHOW FIRST (LAST (LAST :bsp.liste))
--> 2
```

BUTFIRST bzw. BUTLAST haben als Resultat die eingegebene Liste ohne das erste bzw. das letzte Element:

```
SHOW BUTFIRST (FIRST :bsp.liste)
--> [logo]

SHOW BUTLAST (FIRST (BUTFIRST (LAST..
..:bsp.liste)))
--> [2]
```

Mit Hilfe von FPUT (steht für „first put") bzw. LPUT (last put) kann zu einer Liste an erster bzw. an letzter Stelle ein weiteres Element, also eine Zahl, ein Wort oder selbst wieder eine Liste, hinzugefügt werden:

```
SHOW LPUT [in peeker] FIRST :bsp.liste
--> [apple logo [in peeker]]

SHOW FPUT (FIRST (FIRST :bsp.liste))..
..FIRST (BUTFIRST (LAST :bsp.liste))
--> [apple 2e]
```

Mit SENTENCE und LIST können einzelne Objekte zu Listen zusammengefaßt werden:

```
SHOW LIST [apple] "logo
--> [[apple] logo]

SHOW SENTENCE [apple] "logo
--> [apple logo]
```

Weicht die Anzahl der zusammenzufassenden Objekte vom Default-Wert 2 ab, dann sind, wie unten gezeigt, runde Klammern zu schreiben:

```
SHOW (LIST "apple [likes] "logo [] )
--> [apple [likes] logo []]

SHOW (SENTENCE "apple [likes]..
.."logo [] )
--> [apple likes logo]
```

Nur am Rande soll darauf hingewiesen werden, daß die Befehle für Listenoperationen auch auf Wörter anwendbar sind. In Logo sind das Zeichenketten aus Buchstaben oder Ziffern, d.h. daß auch Zahlen zu den Wörtern gerechnet werden können. Hierzu ein kurzes Beispiel:

```
TO quersumme :zahl
IF (COUNT :zahl) = 1 [OUTPUT :zahl]
OUTPUT (FIRST :zahl) + quersumme..
..BUTFIRST :zahl
END
```

Die hier verwendete Logo-Version verfügt über sog. „property lists", die im nachfolgenden Beispiel für den Umgang mit Listen vorteilhaft einzusetzen wären. Da sie jedoch (noch) nicht zur Standard-Ausstattung gehören, werden sie an dieser Stelle nicht benutzt.

Beispiel Sportwettkampf

Für einen Sportwettkampf soll eine Teilnehmerliste erstellt werden. Die in den Disziplinen Lauf und Sprung erreichten Punktzahlen sollen in eine Punktliste aufgenommen werden. Schließlich werden noch die Gewinner einer Urkunde ermittelt, die ab einer bestimmten Gesamtpunktzahl verliehen wird.

In einem ersten Lösungsvorschlag wird mit globalen (Listen-)Variablen gearbeitet. Im Anschluß daran wird gezeigt, wie durch Abänderung einzelner Prozeduren in Funktionen die erzeugten Listen direkt als Eingabe-Parameter weitergegeben werden können.

Als bisher unerwähntes Logo-Grundwort taucht lediglich READLIST auf, das eine mit <RETURN> abzuschließende Eingabe über Tastatur als Liste weitergibt, außerdem wird die Hilfsprozedur „blank" verwendet, die einen Leerschritt ausführt:

```
TO blank
TYPE CHAR 32
END
```

Die folgende Prozedur „teilnehmerliste" erzeugt eine – zunächst natürlich leere – Liste zur Aufnahme der einzelnen Teilnehmer. Diese werden mit Hilfe der rekursiven Prozedur „teiln.einlesen", die durch Eingabe von „#" abgeschlossen werden kann, sukzessive ans Ende der Liste angehängt:

```
TO teilnehmerliste
MAKE "teiln.liste []
teiln.einlesen
END

TO teiln.einlesen
LOCAL "name
TYPE [Vorname Name:] blank
MAKE "name READLIST
IF :name = [*] [STOP] [MAKE..
..."teiln.liste LPUT :name :teiln.liste]
teiln.einlesen
END
```

Nach Beendigung von „teilnehmerliste" liegt also unter dem Namen „teiln.liste" beispielsweise folgende Liste global vor:

```
[[adam apple] [leo logo]..
..[paul peeker]]
```

Die untenstehende Prozedur „punktliste" ruft lediglich eine weitere rekursive Prozedur „punkte.einlesen" auf, die mit zwei formalen Parametern arbeitet:
einer „liste.1", deren Struktur der vorliegenden Teilnehmerliste entspricht, und einer „liste.2", in die die um die Punktzahlen erweiterten Teilnehmer-Elemente eingetragen werden.
Der Aufruf von „punktliste" übergibt dabei an „punkte.einlesen" als aktuelle Parameter zum einen die fertige Teilnehmerliste, zum anderen die leere Liste (da ja bislang noch keine Punktzahlen vergeben wurden). Ist die Teilnehmerliste ganz abgearbeitet (d.h. :liste.1 = [ ]), dann wird an eine global erzeugte „teiln.liste.mit.punkten" der Wert von „liste.2" übergeben:

```
TO punktliste
punkte.einlesen :teiln.liste []
END

TO punkte.einlesen :liste.1 :liste.2
IF :liste.1 = [] [MAKE..
..."teiln.liste.mit.punkten :liste.2..
..STOP]
PRINT FIRST :liste.1
TYPE [punkte fuer lauf und sprung:..
..] blank
LOCAL "punkte MAKE "punkte READLIST
MAKE "liste.2 LPUT (SENTENCE FIRST..
..:liste.1 :punkte) :liste.2
punkte.einlesen (BUTFIRST :liste.1) :liste.2
END
```

Als Teilnehmerliste einschließlich der erreichten Punktzahlen kann nun folgende Liste global vorliegen:

```
[[adam apple 25 15] [leo logo 20..
..10] [paul peeker 35 30]]
```

Da die einzelnen Teilnehmer durch Listen repräsentiert sind, kann auf ihre Merkmale z.B. folgendermaßen zugegriffen werden:

```
TO vorname :teilnehmer
OUTPUT FIRST :teilnehmer
END

TO zuname :teilnehmer
OUTPUT FIRST BUTFIRST :teilnehmer
END

TO punkte.sprung :teilnehmer
OUTPUT LAST :teilnehmer
END

TO punkte.lauf :teilnehmer
OUTPUT LAST BUTLAST :teilnehmer
END
```

Die folgende Prozedur „punktzahl" ist wegen des fehlenden Eingabeparameters nicht selbständig aufrufbar:

```
TO punktzahl
OUTPUT (punkte.sprung :teilnehmer)..
..+ (punkte.lauf :teilnehmer)
END
```

Sie ist auf die Einbettung in eine Prozedur mit dem Parameter „teilnehmer" angewiesen:

```
TO urkunde :teilnehmer :siegpunkte
IF punktzahl > :siegpunkte [(PRINT..
..[Urkunde fuer] zuname :teilnehmer)]
END
```

Da die einzelnen Teilnehmer in einer Liste „verpackt" sind, ist jetzt noch eine Prozedur nötig, die sie der Reihe nach mittels Rekursion wieder ausliest:

```
TO urkunden :liste :siegpunkte
IF :liste = [] [STOP]
urkunde FIRST :liste :siegpunkte
urkunden BUTFIRST :liste :siegpunkte
END
```

Die Übergabe der bereits erzeugten „teiln.liste.mit.punkten" geschieht mit der folgenden Prozedur:

```
TO sieger :siegpunkte
urkunden :teiln.liste.mit.punkten..
..:siegpunkte
END
```

Je nach festgelegter Punktzahl für die Sieger können nun z.B. durch einen Aufruf „sieger 50" die einzelnen Sieger ermittelt werden.

Wird die Aufgabe über Funktionen gelöst, d.h. werden die von den jeweiligen Prozeduren erzeugten Listen unmittelbar als aktuelle Parameter an die nächste Prozedur übergeben, dann sind die folgenden Änderungen nötig (sich nicht verändernde Zeilen werden nur mit ihrem ersten Wort erwähnt):

```
TO teiln.einlesen :teiln.liste
LOCAL ..
TYPE ..
MAKE ..
IF :name = [*] [OUTPUT :teiln.liste]
.. [MAKE ..]
OUTPUT teiln.einlesen :teiln.liste
END

TO teilnehmerliste
OUTPUT teiln.einlesen []
END

TO punkte.einlesen :liste.1 :liste.2
IF :liste.1 = [] [OUTPUT :liste.2]
PRINT ..
TYPE ..
LOCAL ..
MAKE ..
OUTPUT punkte.einlesen BUTFIRST..
..:liste.1 :liste.2
END

TO punktliste :liste
OUTPUT punkte.einlesen :liste []
END
```

Besonders wichtig ist das Hinzufügen von OUTPUT beim rekursiven Aufruf einer Prozedur (in den obigen Fällen also in den vorletzten Zeilen), denn OUTPUT bewirkt die Rückgabe des Ergebnisses nur an die („ihn") aufrufende Prozedur. Da der die Prozedur beendende „innere" OUTPUT demnach nur eine Rückgabe des Ergebnisses an den vorletzten Aufruf bewirkt, muß auch dieser Aufruf usw. mit OUTPUT versehen werden, um eine Weitergabe des Resultats bis zum ersten Prozeduraufruf zu erreichen.

Nach diesen Änderungen lassen sich nun in einer alles erfassenden Prozedur direkt die Sieger ermitteln:

```
TO sieger :siegpunkte
urkunden (punktliste teilnehmerliste)..
..:siegpunkte
END
```

Die Bedeutung des Datentyps Liste wird noch einmal ganz besonders unterstrichen in Verbindung mit den Befehlen RUN, TEXT und DEFINE. Einige Beispiele aus und zu diesem Kontext sollen abschließend vorgeführt werden.

6. Listenoperationen mit RUN, TEXT, DEFINE

Äußerst interessante Perspektiven in Verbindung mit Listen eröffnet der Befehl **RUN**. Mit seiner Hilfe kann der dafür geeignete Inhalt einer Liste so ausgeführt werden, als ob ihre Elemente über die Tastatur direkt eingegeben worden wären. So erzeugt z.B.

```
RUN [PRINT [apple logo]]
```

auf dem Bildschirm den Ausdruck „apple logo". Diesen Effekt erhält man auch zusammen mit Listen-Variablen:

```
MAKE "run.demo [PRINT [apple logo]]
RUN :run.demo
--> apple logo

PRINT RUN [5 * 5 > 30]
--> FALSE
```

Als Listeninhalte können alle sinnvollen, auswertbaren bzw. ausführbaren Befehlssequenzen, Prozeduren und Funktionen – natürlich auch die selbsterstellten – gewählt werden. Setzt man RUN dann noch innerhalb von Prozeduren ein, lassen sich in Anlehnung an Pascal z.B. folgende, ursprünglich nicht zu Logo gehörende, Kontrollstrukturen implementieren:

Für Ihre Unterlagen

Abonnement bestellt

am: _____

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

Peeker
Leserservice
Postfach 10 28 69
6900 Heidelberg

PEEKER

Abo-Karte

Ja, ich möchte **Peeker** abonnieren.
Liefern Sie mir **Peeker** ab Ausgabe ..... zum Jahresbezugspreis von z. Zt. DM 72,– (Inland) inkl. MwSt. Die Lieferung erfolgt frei Haus. Porto, Verpackung und Zustellgebühren übernimmt der Verlag. Der Jahresbezugspreis für das Ausland beträgt z. Zt. DM 72,– plus DM 18,– Versandspesen.

X _____
Datum 1. Unterschrift

Bitte lesen!

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

X _____
Datum 2. Unterschrift

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

Wir können nur Bestellungen mit zwei Unterschriften bearbeiten.

Für Ihre Unterlagen

Folgende Bücher bestellt:

_____
_____
_____
_____
_____

am: _____

bei:

Peeker
Versandbuchhandlung
Postfach 10 28 69
6900 Heidelberg 1

PEEKER

Buch-Karte

Bitte senden Sie mir gegen Rechnung folgende Bücher:

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

_____
Datum Unterschrift

Für Ihre Unterlagen

Folgende Disketten
und Programme bestellt:

_____
_____
_____
_____
_____

am: _____

bei:

Peeker
Softwareabteilung
Postfach 10 28 69
6900 Heidelberg 1

PEEKER

Software-Karte

Bitte senden Sie mir gegen Rechnung folgende Disketten:

- ☐ Peeker-Sammeldiskette, einzeln
  Disk# _____, Disk# _____
  Disk# _____, Disk# _____
  Peeker je Disk DM 28,– (einzeln)
- ☐ Peeker-Sammeldiskette, im Fortsetzungsbezug ab Disk# _____ (Mindestbezug 6 Disketten) Preis je Disk DM 20,–
- ☐ Apple DOS 3.3, Begleitdisk., DM 28,–
- ☐ ProDOS, Band 1, Begleitdisk., DM 28,–
- ☐ ProDOS, Band 2, Begleitdisk., DM 28,–
- ☐ Apple Assembler, Begleitdisk., DM 28,–
- ☐ ProDOS-Editor 1.0, Programm, DM 98,–
- ☐ MMU 2.0, Programm, DM 98,–
- ☐ INPPUT 2.0, Programm, DM 98,–
- ☐ Softbreaker 1.0, Programm, DM 48,–
- ☐ DB-Meister, Programm, DM 290,–
- ☐ Superquick, Programm, DM 48,–
- ☐ Turtle Graphics, Programm, DM 98,–
- ☐ Disk 40, Programm, DM 48,–
- ☐ Kyan-Pascal 2.0, Programm, DM 170,–
- ☐ [redacted] Fast-Writer, DOS 3.3, DM 128,–
- ☐ Fast-Writer, ProDOS, DM 128,–
- ☐ Double-Hires-Tools für Applesoft, DM 28,–

_____
Datum Unterschrift

Abo-Karte

Name

Firma

Straße

PLZ/Ort

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

Bank/PschA

Bankleitzahl

Kto.-Nr.

POSTKARTE

Peeker
Leserservice

Dr. Alfred Hüthig Verlag GmbH

Postfach 10 28 69

6900 Heidelberg

INPUT 2.0

Ein Bildschirm-Maskengenerator für DOS 3.3 und ProDOS

von U. Stiehl

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

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

Buch-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

POSTKARTE

Peeker
Buchabteilung

Dr. Alfred Hüthig Verlag

Postfach 10 28 69

6900 Heidelberg 1

MMU 2.0

Memory Managements Utilities

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

von U. Stiehl

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

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

Gerätevoraussetzung: Apple IIe mit 64K-Karte oder IIc

Software-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

POSTKARTE

Peeker
Softwareabteilung

Dr. Alfred Hüthig Verlag

Postfach 10 28 69

6900 Heidelberg 1

DISK 40

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

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

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

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

```
TO while :bedingung :befehlsfolge
IF (RUN :bedingung) [RUN :befehlsfolge]
                    [STOP]
while :bedingung :befehlsfolge
END
```

„bedingung“ und „befehlsfolge“ sind dabei jeweils als Listen einzugeben, damit sie von RUN richtig verarbeitet werden können. Folgender Aufruf sei als Beispiel für die Verwendung von „while“ anzusehen:

```
MAKE "x 1
while [:x < 1Ø]
[(PRINT :x :x * :x SQRT :x)
MAKE "x :x + 1]
—> 1 1 1
    2 4 1.41421
    3 9 1.732Ø5
    . . .
    9 81 3.Ø
```

Ganz entsprechend läßt sich auch eine *„repeat-until-Schleife“* formulieren. Da REPEAT als Logo-Grundwort bereits vorliegt, darf dieses Wort jedoch nicht als Prozedurname gewählt werden:

```
TO wiederhole :befehlsfolge :bis
RUN :befehlsfolge
IF NOT (RUN :bis)
 [wiederhole :befehlsfolge :bis]
 [STOP]
END
```

In derselben Weise können, wenn sie unbedingt gebraucht werden, auch weitere Kontroll-Befehle erzeugt werden (z.B. „for-to“, „case“, „case-else“).

Eine ganz besonders einfache Lösung ergibt sich mit Hilfe des RUN-Befehls bei Aufgaben, die z.B. Schaubilder oder Wertetafeln von (mathematischen) Funktionen verlangen. Es sind genügend viele Tricks bekannt, wie z.B. in BASIC ganze Funktionsterme als Quasi-Parameter an ein ja immer gleich ablaufendes Programm übergeben werden können; immerhin ist zu beachten, daß viele Programmiersprachen, wozu auch Pascal zählt, eine einfache Übergabe nicht unterstützen (vgl. Peeker, Heft 4/86, S.58: „Interaktive Funktionseingabe“). Logos Datentyp Liste erfordert zusammen mit RUN hier keinerlei „künstliche Verrenkungen“.

Es reizt, an dieser Stelle nun doch noch Logos Grafik einzusetzen, die bei aller Einfachheit dennoch reichhaltig ausgestattet ist. Da sich dieser Artikel jedoch auf typische Logo-Strukturen beschränken will, wird statt dessen die Funktionsübergabe zum Ausdruck von Wertetabellen benutzt.

Die Prozedur „w.tafel“ besitzt vier Parameter: die Funktion, die als Liste eingegeben werden muß, die untere und obere Intervallgrenze, innerhalb deren die Funktionswerte ermittelt werden sollen, und schließlich die Schrittweite, mit der das Intervall durchlaufen werden soll:

```
TO w.tafel :funktion :start
 :ende :schritt
IF :start > :ende [STOP]
(TYPE "f( :start ")= ) PRINT
       RUN ersetze "x :start :funktion
w.tafel :funktion (:start + :schritt)
       :ende :schritt
END
```

Ein möglicher Aufruf könnte so lauten:

```
w.tafel [abs x] -1Ø 1Ø Ø.5
```

wobei in diesem Fall als Funktion die benutzerdefinierte Absolutwert-Funktion verwendet wurde.

```
--> f(-1Ø) = 1Ø
    f(-9)=9
    ..
    f(1Ø)=1Ø
```

Eine kurze Anmerkung zu einer (manchmal etwas ärgerlichen) Syntax-Regel: Einige Zeichen sind von vornherein mit einer zusätzlichen Trennfunktion ausgestattet. Dies ist für eine Verwendung innerhalb von Listen durchaus sinnvoll – jedoch ist der Befehl „TYPE “f (..“ in dieser Form nicht zu notieren, da er so nicht ausgeführt wird. (Apple-)Logo macht daraus automatisch „TYPE “f (“. Die eigentlich geforderte Zeichenkette, also das Wort „f (“, wird getrennt in das Wort „f“ und die offene runde Klammer, die der Interpreter nun gesondert auszuwerten versucht. (Apple) Logo verwendet, um obige Notationen dennoch möglich zu machen, ein Sonderzeichen („backslash“: \), das bei der Reihe von Zeichen mit Trennfunktion deren Sondereigenschaft unterdrückt.

Viel entscheidender ist jedoch die Sequenz „RUN ersetze “x :start :funktion“ in der gleichen Prozedurzeile: Hier wird von einer Prozedur mit dem Namen „ersetze“ Gebrauch gemacht, die das Funktionsargument x, also ein Element aus der nachfolgenden Liste „funktion“, jeweils durch den aktuellen Zahlenwert, der in „start“ enthalten ist, ersetzt.

```
TO ersetze :zchn.1 :zchn.2 :liste
IF :liste = [] [OUTPUT []]
IF (FIRST :liste) =
    :zchn.1 [OUTPUT FPUT :zchn.2 ..
.. ersetze
 :zchn.1 :zchn.2 BUTFIRST :liste]
OUTPUT FPUT FIRST :liste ersetze
    :zchn.1 :zchn.2 BUTFIRST :liste
END
```

Ein Beispiel-Aufruf von „ersetze“ kann die Wirkung noch einmal verdeutlichen:

```
SHOW ersetze "a "peeker [a b c a]
--> [peeker b c peeker]
SHOW ersetze "x 25 [(sin x) / (cos x)]
--> [( sin 25 ) / ( cos 25 )]
```

Nachdem nun mit großer Selbstverständlichkeit immer wieder rekursive Prozeduren verwendet werden, kann zwischendurch wenigstens mitgeteilt werden, daß Logo über einen sehr ordentlichen Protokoll-Modus verfügt. In der Apple-Version kommen hierfür die Befehlspaare STEP/UNSTEP bzw. TRACE/UNTRACE in Frage. Befiehlt man beispielsweise „STEP “ersetze“ zusammen mit „TRACE “ersetze“, dann läßt sich am Bildschirm sehr anschaulich und schrittweise ein konkreter Aufruf von „ersetze“ verfolgen.

Die für „w.tafel“ nötigen aktuellen Parameter lassen sich über eine zweite Prozedur „wertetafel“ z.B. so erzeugen und übergeben:

```
TO wertetafel
LOCAL [funktion start ende schritt]
TYPE [funktion:] blank
       MAKE "funktion READLIST
TYPE [startwert:] blank
       MAKE "start READWORD
TYPE [endwert:] blank
       MAKE "ende READWORD
TYPE [schrittweite:] blank
       MAKE "schritt READWORD
w.tafel :funktion :start :ende :schritt
END
```

Die Eingabe der Funktion kann wegen READLIST ohne eckige Klammern erfolgen, die weiteren Werte werden mit READWORD als Worte eingelesen.

Zur späteren Verwendung muß die Prozedur „ersetze“ noch etwas erweitert werden, denn bislang ersetzt sie entsprechende Zeichen nur, wenn diese als unmittelbare Elemente in der Liste vorkommen. Sollen sie auch ersetzt werden, wenn sie innerhalb einer Liste in der Liste vorkommen, dann benötigt man z.B. die folgende Version (sie verwendet die Logo-Grundfunktion LISTP, die bei Eingabe einer Liste TRUE ausgibt):

```
TO ersetze.alle :zchn.1 :zchn.2 :liste
IF :liste = [] [OUTPUT []]
IF LISTP (FIRST :liste) [OUTPUT ..
 ..FPUT (ersetze.alle :zchn.1 :zchn.2..
 ..FIRST :liste) ersetze.alle :zchn.1..
 ..:zchn.2 BUTFIRST :liste]
IF (FIRST :liste) = :zchn.1 [OUTPUT ..
 ..FPUT :zchn.2 ersetze.alle :zchn.1..
 .. :zchn.2 BUTFIRST :liste]
OUTPUT FPUT (FIRST :liste) ersetze.alle..
 .. :zchn.1 :zchn.2 BUTFIRST :liste
END
```

Damit funktioniert auch der folgende Aufruf:

```
MAKE "ersetze.demo [[[[[x]
 x] x] ..
 ..apple x] x [x logo [x [x [x]]]]]
SHOW ersetze.alle "x "u :ersetze.demo
—> [[[[[u] u] u]
 apple u] u [u logo..
 .. [u [u [u]]]]]
```

Letztendlich soll nun auch noch eine Prozedur geschaffen werden, mit der mehrere Elemente einer Liste durch entsprechend viele andere Zeichen(ketten) ersetzt werden können; dazu werden die zu ersetzenden Elemente und die Ersatzelemente jeweils in einer der Prozedur zu übergebenden Liste zusammengefaßt. Außerdem werden die Ausgangsliste und die Liste mit den getauschten Elementen parallel geführt, wobei sich letztere mit dem Fortschreiten des Tauschvorgangs laufend verändert und für den nächsten Tauschvorgang als Ausgangsliste Verwendung findet. Die Prozedur endet mit dem OUTPUT der „endliste", wenn die Liste der zu ersetzenden Elemente abgearbeitet, also leer ist:

```
TO ersetze.liste :elemente.raus ..
..:elemente.rein :ausgangsliste :endliste
IF :elemente.raus = []  ..
 .. [OUTPUT :endliste]
MAKE "endliste ..
 ..ersetze.alle FIRST :elemente.raus ..
 ..FIRST elemente.rein :ausgangsliste
MAKE "ausgangsliste :endliste
OUTPUT ersetze.liste BUTFIRST ..
..:elemente.raus BUTFIRST :elemente.rein
 .. :ausgangsliste :endliste
END
```

Ein Aufruf von „ersetze.liste" muß natürlich den Parameter „endliste" als leere Liste enthalten, die erst während der Rekursionen gefüllt wird:

```
SHOW ersetze.liste [g o l] [r u g]..
.. [l o g o] []
--> [g u r u]

SHOW ersetze.liste [l o g] [g u r]..
.. [l o g o] []
--> [r u r u]
```

Die Prozedur „ersetze.liste" wird im Zusammenhang mit den nun noch anzusprechenden Logo-Grundwörtern TEXT und DEFINE noch einmal Verwendung finden.

TEXT erwartet als Eingabe einen Prozedurnamen und gibt dann diese Prozedur als eine Liste (von Listen) aus: ihr erstes Element ist immer eine Liste, die die formalen Prozedurparameter als Elemente enthält. Arbeitet die Prozedur ohne Eingabe-Parameter, dann wird dieses Element als leere Liste aufgenommen; sämtliche weiteren Elemente sind wiederum Listen, die jeweils eine Prozedurzeile zum Inhalt haben, z.B.

```
TO abs :x
IF :x < Ø [OUTPUT - :x]
OUTPUT :x
END

SHOW TEXT "abs
--> [[x] [IF :x < Ø
 [OUTPUT - :x]]..
.. [OUTPUT :x]]
```

Da natürlich auch TEXT innerhalb von Prozeduren verwendet werden kann, wird offensichtlich, daß sich hier gewaltige Möglichkeiten auftun. Prozeduren liegen nun im Listenformat vor und können, wie jede andere Liste auch, mit Hilfe anderer listenver- und -bearbeitender Prozeduren, also durch Programmierung, verändert werden. Um dann die geänderte Prozedur-Liste wieder in eine eigenständige, lauffähige Prozedur zurückzuverwandeln, bedient man sich des Grundworts **DEFINE**, wobei lediglich darauf zu achten ist, daß das erste Element als Liste die Eingabe-Parameter der zu definierenden Prozedur enthält. Der neue Prozedurname wird als erste Eingabe an DEFINE übergeben:

```
DEFINE "absolutwert [[x] [IF
 :x < Ø ..
.. [OUTPUT - :x]] [OUTPUT :x]]
```

DEFINE erzeugt also eine in diesem Fall zu „abs" identische Prozedur namens „absolutwert", die sich in nichts von den herkömmlich erstellten Prozeduren unterscheidet.

Es würde inhaltlich zu weit führen und auch den Rahmen dieses ohnehin schon lang genug geratenen Beitrags übersteigen, wollte man nun dieses neue Feld mit einem voll überzeugenden Beispiel genauer vorstellen. Nur *ein* Anwendungsaspekt soll noch ansatzweise demonstriert werden, nämlich die Übersetzung von Prozeduren mit den hier verwendeten englischen Logo-Befehlen in gleichbedeutende deutsche mit Hilfe einer Übersetzungsprozedur:

```
MAKE "engl [BUTFIRST FIRST FPUT IF..
.. MAKE OUTPUT PRINT RUN SENTENCE STOP]
MAKE "dtsch [OHNEERSTES ERSTES..
.. MITERSTEM WENN SETZE RUECKGABE..
.. DRUCKEZEILE TUE SATZ RUECKKEHR]
```

Die beiden gerade vorgestellten Listen sind natürlich unvollständig. Für unsere Demonstration reichen sie jedoch aus. Die Wörter der Liste „dtsch" entsprechen in der Reihenfolge der Liste „engl" den Befehlen der deutschen IWT-Logo-Version.

```
TO uebersetze :neuer.prozname ..
.. :alter.prozname
DEFINE :neuer.prozname FPUT FIRST TEXT..
.. :alter.prozname ..
.. ersetze.liste (SENTENCE :engl..
.. :alter.prozname) ..
.. (SENTENCE :dtsch :neuer.prozname) ..
 .. BUTFIRST TEXT :alter.prozname []
END
```

Wird „uebersetze" auf z.B. „abs" angewendet, dann ergibt sich (der erstmals verwendete Befehl **PO** steht für „print out", er veranlaßt den Ausdruck einer Prozedur):

```
uebersetze "absolutwert "abs
PO "absolutwert
--> TO absolutwert :x
 WENN :x < Ø [RUECKGABE - :x]
 RUECKGABE :x
 END
```

Die Syntax in der zweiten Zeile ist allerdings noch nicht stimmig. Für die deutsche Version der Prozedur soll die Formulierung „WENN-DANN-SONST" verwendet werden. Zu einer Teillösung dieses Spezialproblems taugen die folgenden Prozeduren.

Da es sich bei TEXT einerseits um eine Liste von Listen handelt und andererseits auch IF selber wieder die Notation von Listen verlangt, kann nicht einfach die sich öffnende eckige Klammer mit DANN bzw. an späterer Stelle mit SONST übersetzt werden. Schon der Zugriff auf die Klammerzeichen klappt nicht, da sie ja nicht als einfache Zeichen, also nicht als Elemente in der Liste auftreten, sondern eben als Bezeichner einer Liste. Es ist also eine Prozedur vonnöten, die zum einen in der Lage ist, bis zu einer bestimmten Tiefe und unter bestimmten Bedingungen die inneren Listen zu öffnen, und die zum anderen die Fähigkeit besitzt, die Vokabeln DANN und SONST an der richtigen Stelle einzufügen. Um das Vorhaben nicht allzusehr zu komplizieren, wird zunächst einmal davon ausgegangen, daß neben den zu IF gehörenden „then"- und „else"-Listen nur noch die leere Liste als Element der durch TEXT erzeugten Zeilenliste vorkommen soll (damit läßt sich über eine einfache Abfrage deren Öffnung – und das bedeutete ja ihr Verschwinden! – vermeiden).

Die Prozedur „dann.od.sonst" sorgt dafür, daß jeweils die passende Vokabel an eine Variable „dos" übergeben wird.

```
TO dann.od.sonst
IF :dos = "SONST [MAKE "dos "DANN]..
.. [MAKE "dos "SONST]
END

TO oeffnen :liste
LOCAL "dos
MAKE "dos "SONST
OUTPUT aufmachen :liste
END

TO aufmachen :liste
IF :liste = [] [OUTPUT []]
IF LISTP FIRST :liste..
.. [IF (FIRST :liste) = [] [OUTPUT..
.. FPUT FIRST :liste aufmachen BUTFIRST
.. :liste][dann.od.sonst..
.. OUTPUT (SENTENCE :dos aufmachen..
.. FIRST :liste aufmachen BUTFIRST..
.. :liste)]]
OUTPUT SENTENCE FIRST :liste oeffnen..
.. BUTFIRST :liste
END
```

Es ist zu beachten, daß in Logo auch wechselseitig rekursive Aufrufe möglich sind, wie dies bei „oeffnen" und „aufmachen" gezeigt wurde.

Die Prozedur „oeffnen" kann natürlich nur dann sinnvoll arbeiten, wenn sie auf eine Prozedur-Zeile, also eine Liste aus TEXT, angewandt wird, die mit IF, d.h. übersetzt mit WENN, beginnt (auch dies sei der Ein-

fachheit halber unterstellt). Die folgende Prozedur „wenn.zeile“ ist in der Lage, solche Zeilen festzustellen und auf sie „oeffnen“ anzuwenden:

```
TO wenn.zeile :liste
IF :liste = [] [OUTPUT []]
IF NOT EQUALP (FIRST :liste) [] ..
..[IF (FIRST FIRST :liste) = "WENN ..
..[OUTPUT FPUT oeffnen FIRST :liste..
.. wenn.zeile BUTFIRST :liste]]
OUTPUT FPUT FIRST :liste wenn.zeile..
.. BUTFIRST :liste
END
```

Unter den genannten einschränkenden Bedingungen kann nun die folgende Prozedur „translate“ bestimmte bestehende Prozeduren unter neuem Namen „in deutscher Übersetzung“ syntaxgerecht erzeugen:

```
TO translate :neuer.name :alter.name
DEFINE :neuer.name wenn.zeile..
.. uebersetze :neuer.name :alter.name
END
```

In „uebersetze“ muß lediglich noch die Sequenz „DEFINE :neuer.prozname“ herausgenommen und durch OUTPUT ersetzt werden.

Ungelöst ist jetzt noch das Problem, wenn IF-Abfragen nicht mit der leeren Liste arbeiten oder wenn innerhalb der „then“- bzw. „else“-Liste selber wieder Listen auftreten (z.B. im Zusammenhang mit dem PRINT-Befehl). Diese Schwierigkeit soll hier nicht bis zum letzten ausgeräumt werden. Ein relativ anschaulicher, allerdings nur partiell aus der Misere führender Ausweg wird aber wenigstens skizziert: Es wird eine Variable „letztes“ eingeführt, die jeweils das letzte bearbeitete Listenelement konserviert. Je nachdem, welchen Wert diese Variable gerade hat, wird entschieden, ob die nachfolgende Liste geöffnet und DANN oder SONST eingefügt wird. Die Entscheidung benützt OR für verschiedene mögliche Fälle; nur ein paar davon werden nachfolgend berücksichtigt.

Die Variable „letztes“ kann lokal in „wenn.zeile“ angekündigt werden; die erforderlichen Änderungen betreffen nur die Prozedur „aufmachen“:

```
TO aufmachen :liste
IF :liste = [] [OUTPUT []]
IF LISTP FIRST :liste [IF OR..
..(MEMBERP :letztes [= WENN DRUCKEZEILE
..RUECKGABE]) (NUMBERP :letztes)..
..[MAKE "letztes FIRST :lisste OKTPUT..
..weiter wie oben ..][MAKE "letztes..
..FIRST :liste dann.od.sonst ..weiter..
..wie oben ..]]
MAKE "letztes FIRST :liste
OUTPUT SENTENCE .. weiter wie oben ..
END
```

MEMBERP untersucht das Enthaltensein eines Elements in einer nachfolgend eingegebenen Liste bzw. eines Zeichens in einem Wort. Bei den obigen Änderungen wurde davon ausgegangen, daß eine Liste dann nicht geöffnet werden darf, wenn vor ihr das Gleichheitszeichen oder die Wörter WENN, DRUCKEZEILE bzw. RUECKGABE stehen, oder auch nur eine Zahl.

Noch einmal: Es wurde höchstens ein an dieser Stelle genügender Ausweg skizziert, der einige häufig auftretende Fälle berücksichtigt. Besser wäre sicherlich ein Vorgehen, das z.B. in einer Prozedur „logischer.ausdruck“ die zu öffnenden von den anderen Listen unterscheidet, oder man könnte über runde Klammern eine Festlegung der Syntax verlangen.
Folgende Aufgabe wird immerhin schon zur Zufriedenheit gelöst:

```
TO dummy
IF :x = [] [IF :b = [a] [PRINT..
.. [tue dies]] [IF :c = "ok [PRINT..
..[tue nix]]]]
END

Translate "nonsense "dummy
PO "nonsense
--> TO nonsense
 WENN :x = [] DANN WENN :b = [a]..
..DANN DRUCKEZEILE [tue dies] ..
..SONST WENN :c = "ok DANN DRUCKEZEILE..
..[tue nix]
 END
```

7. Zusammenfassung

Logo ist ein LISP-Abkömmling (was nicht unbedingt und von vornherein schon eine qualifizierte Programmiersprache bedeuten muß) und hat davon viel Interessantes, Wichtiges und v.a. Leistungsfähiges abbekommen. Hoffentlich hat dieser Artikel in Ansätzen zu dieser Auffassung beitragen können. Es ist durchaus einzuräumen, daß man manche Aufgabe auch anders hätte angehen können. Auch wäre es möglich gewesen, manche Prozedur komprimierter zu fassen, doch dies lag nicht immer im Sinne dieses Beitrags.

An einigen für Logo charakteristischen Stellen sollte über die Arbeitsweise mit diesem System Auskunft gegeben werden. Vielleicht ist damit einsichtig geworden, daß Logo wirklich keine „Kindersprache“ ist. Denn diese als vermutlich geringschätzige Abqualifizierung gemeinte Einstufung entpuppt sich bei näherem Hinsehen eigentlich eher als Auszeichnung, denn zweifellos steht fest, daß – z.B. in dem hier nicht behandelten Grafik-Bereich – auch nicht besonders fortgeschrittene Programmierer in Logo sehr gute Erfolge erzielen können. Die dem natürlichen Problemlösevorgang angepaßte Sprachverwendung erlaubt sehr niederschwellige Einstiege, die gerade auch von Kindern ohne weiteres bewältigt werden können. Entscheidend ist aber doch, daß Logo seinen Benutzer nicht auf diesem Niveau festhält, sondern erhebliche Steigerungen im Komplexitätsgrad der zu bearbeitenden Probleme angemessen zuläßt – oft genug angemessener als manch andere Programmiersprache.

Literatur

Abelson, Harold: Einführung in Logo (übers. u. bearb. v. Herbert Löthe) Vaterstetten 1983
Apple Logo II Reference Manual
Hoppe, Ulrich/Löthe, Herbert: Problemlösen und Programmieren mit Logo, Stuttgart 1984
Ziegenbalg, Jochen: Programmieren lernen mit Logo, München/Wien 1985

Neue BASIS-User-Group

Aufgrund einer Meldung im Peeker, Heft 10/85, S. 68 hat sich eine kleine BASIS-User-Group gebildet, die Kontakte zu anderen Usern oder User-Gruppen sucht, Probleme löst, Tips, Tricks und Erfahrungen weitergibt. Diese Gruppe hat inzwischen zwei Rundschreiben herausgegeben.
Im ersten Rundschreiben (März/April 1986) wird über folgende Themen berichtet:
– Erfahrungen mit dem Erphi-Controller unter BASIS-CP/M 3.0
– Neue Zusatzkarte der Firma BASIS, um Software-Kompatibilität zwischen BASIS 108 und Apple IIe herzustellen
– Wordstar-Patches und Labeladressen
– CP/M-3.0-Patches
– Turbo-Pascal-Tips

Das zweite Rundschreiben vom Juni 1986 befaßt sich mit den Themen:
– ProDOS 1.1.1 auf BASIS 108 (Anpassung durch Modifizieren von ProDOS 1.1.1 s. Kyan-Club-Nachrichten, Peeker, Heft 5/86, S. 60)
– Kyan-Pascal Version 2.0, ein vollcompilierendes 6502-Pascal (Hüthig Software Service, Heidelberg)
– BASIS-108-Club-Mailbox
– CP/M-3.0-Patches
– CP/M-3.0-Fehler
– Preislisten BASIS-Computer

Die BASIS-User-Group ist weiterhin an neuen Mitgliedern interessiert, deshalb nochmals die vollständige Adresse:
BASIS-User-Group
Rolf Gachnang
Neue Jonastraße 81
CH-8640 Rapperswil

Das Wahre ist eins

Aussagenlogik für Programmierer

von Ulrich Stiehl

1. Was ist Logik?

Logik kommt vom griechischen Wort „logos", das eigentlich „Wort" und „Rede", dann aber auch „Denkkraft" und „Vernunft" bedeutet. Somit können wir Logik als Lehre vom richtigen oder genauer folgerichtigen Denken definieren. Des näheren unterscheiden wir die *traditionelle Logik*, die sich an die natürliche (Umgangs)-sprache und den „normalen" Menschenverstand anlehnt, von der *modernen Logik*, die sich mit kunstsprachlichen Kalkülen befaßt, die der „normalen" Wirklichkeit teilweise stark „entrückt" sind. Es verwundert deshalb nicht, daß es kein einheitliches System der modernen Logik gibt, sondern nur unterschiedliche Logiken von unterschiedlichen Logikern. Dies beginnt bereits bei der Terminologie und der Symbolik und endet bei den divergierenden Kalkülen (zwei/dreiwertiger Aussagenkalkül; Klassenkalkül mit/ohne Nullklasse usw.). Nehmen wir als Beispiel das einfache Wörtchen „oder": Manche sprechen von Disjunktion (I.M.Bochenski), manche von Adjunktion (P.Lorenzen), manche von Alternative (G.Klaus) usw. Angesichts dieser mißlichen Ausgangssituation werden wir im nachfolgenden einige radikale Vereinfachungen vornehmen müssen, damit der *praktische* Nutzen der Logik für die Programmierung überhaupt erkenntlich wird. Unser Streifzug durch die Logik will deshalb auch keine systematisch-umfassende Darstellung sein, sondern lediglich einige der nützlicheren Aspekte beleuchten.

Das Kernstück der Logik ist die Lehre vom folgerichtigen Schließen (Grund-Folge-Beziehung), d.h. der Ableitung (= Deduktion) von Folgerungen (= Schlüssen = Konklusionen) aus Voraussetzungen (= Prämissen), z.B. „Wenn alle Haie Fische sind (1. Prämisse) und alle Karpfen Fische (2. Prämisse), dann sind alle Karpfen Haie (Konklusion)". Auch wenn wir aufgrund unseres gesunden Menschenverstandes erkennen, daß dieser Schluß nicht „konkludent" ist, so fällt es uns doch schwer nachzuweisen, ob dieses Schlußschema *stets* zu falschen Schlüssen führt. Am Ende unseres kleinen Streifzuges durch die Welt der Logik werden Sie sicherlich „konkludenter" argumentieren können.

2. Programm und Logik

Insbesondere die (zweiwertige) Aussagenlogik als Teilgebiet der Logik ist von der modernen Datenverarbeitung nicht wegzudenken. Zur Einstimmung zählen wir einige EDV-Begriffe auf:
Schaltalgebra: NAND-Gatter, NOR-Gatter, Flip-Flop, Inverter usw.
Assembler: AND, ORA, EOR, BIT, BCC, BCS usw.
BASIC: IF-THEN-ELSE, ON-GOTO/GO-SUB, FOR-NEXT, WHILE-WEND, NOT, AND, OR, EOR, IMP, EQV usw.
Pascal: BOOLEAN, TRUE, FALSE, REPEAT-UNTIL, FOR-DO, WHILE-DO, CASE, EOF, EOLN usw.

Ein Programm ist eine Folge von Befehlen. Normalerweise werden diese Befehle sequentiell abgearbeitet, z.B.
10 PRINT "1. Befehl"
20 PRINT "2. Befehl" usw.

Es gibt jedoch auch Befehle, mit denen man den sequentiellen Befehlsfluß durchbrechen kann. Dabei muß man zwischen unbedingten und bedingten Befehlen unterscheiden. So ist etwa der Sprungbefehl
10 GOTO 100
ein *unbedingter* Befehl, der stets ausgeführt wird. Demgegenüber wird etwa der *bedingte* Befehl „THEN PRINT ..." in
10 IF A = B THEN PRINT "Stimmt!"
nur dann ausgeführt, wenn die Bedingung „IF A = B" erfüllt, richtig oder *wahr* ist. Wenn wir das Miniprogramm
10 A = 2: B = 3
20 PRINT (A = B)
30 PRINT (A < B)
unter Applesoft-BASIC laufen lassen, so werden die Werte 0 und 1 ausgegeben. Daß Computer als „Binärrechner" nur mit Nullen und Einsen umgehen können, haben wir schon in unserem zweiteiligen Beitrag über „Binäres Rechnen mit Papier und Bleistift" (Peeker, Hefte 2/86 und 4/86) deutlich gemacht. Eine binäre 0 oder 1 kann man jedoch nicht nur als *Zahl* interpretieren, sondern auch als **Wahrheitswert**. Danach steht eine 0 bzw. ein gelöschtes Bit für „falsch" und eine 1 bzw. ein gesetztes Bit für „wahr". Wir könnten auch formelhaft sagen: *Das Wahre ist eins.* Damit hätten wir endlich eine Definition für den Begriff der Wahrheit gefunden, um den die philosophischen Köpfe des Abendlands seit mehr als 2000 Jahren vergeblich gerungen haben! Zum Glück sind die Dinge nicht ganz so einfach, denn mit einem Computerprogramm in der Art
IF "Strahlenschäden sind harmlos"
THEN PRINT "Ich hab's gewußt!"
hat noch keiner die Wahrheit gefunden.

3. Elementare Ontologie

Unter Ontologie versteht man die Lehre vom Seienden oder von den Seinsformen. Für den Logiker gibt es nur zwei Arten von *Seinsheiten* (= Seienden = Wesen = Entitäten), nämlich Dinge (= Gegenstände = Objekte) und Eigenschaften (= Bestimmungen = Prädikate). Das **Ding** (z.B. „Baum") ist das, was bestimmt wird, und die **Eigenschaft** (z.B. „groß") ist das, was dem Ding zukommt.
Dinge lassen sich in *Klassen* (= Mengen) einteilen („Würfel", „Kugeln" usw.). Wenn nun eine Eigenschaft *allen* Dingen einer Klasse zukommt, so sprechen wir von einer *essentiellen* Eigenschaft, z.B. „*Alle* Kugeln sind rund". Wenn umgekehrt eine Eigenschaft nur *einigen* Dingen einer Klasse zukommt, so sprechen wir von einer *akzidentellen* Eigenschaft, z.B. „*Einige* Kugeln sind rot".
Die Verbindung von Ding und Eigenschaft nennen wir **Sachverhalt** (= Tatsache = Fall). Wenn dabei eine Eigenschaft nur *einem* Ding oder *einer* Klasse von Dingen zukommt, so liegt ein *attributiver* Sachverhalt vor, z.B. „Bäume sind *groß*". Wenn umgekehrt eine Eigenschaft *mehreren* Dingen oder *mehreren* Klassen von Dingen gleichzeitig bzw. gemeinsam zukommt, so liegt ein *relationaler* Sachverhalt vor, z.B. „Bäume sind *größer als* Sträucher".
Ein Sachverhalt kann selbst als Ding oder Eigenschaft eines komplexeren Sachverhalts fungieren, der selbst wiederum als Bestandteil eines noch komplexeren Sachverhalts fungieren kann usw. Aus der Sicht des Logikers ist deshalb die Welt ein riesiger Sachverhalt oder die Gesamtheit aller „Fälle": „Die Welt ist alles, was der Fall ist" (L.Wittgenstein).

4. Elementare Semantik

Der vorangehende Abschnitt dürfte Ihnen als recht abstrus erscheinen, doch lassen sich alle logischen Kalküle („Logiken") auf die ontologischen Kategorien Ding, Eigenschaft und Sachverhalt reduzieren, wenn man zusätzlich noch die entsprechenden semantischen Grundbegriffe hinzuzieht.
Unter Semantik versteht man die Lehre von den Bedeutungen. In der elementaren Semantik werden nur zwei Arten von *Bedeutungsgebilden* unterschieden, nämlich Begriff und Aussage. Der **Begriff** (= Gedanke) ist ein Bedeutungsgebilde, das Dinge oder Eigenschaften bezeichnet, und die **Aussage** (= Urteil) ist ein Bedeutungsgebilde, das Sachverhalte bezeichnet. Dabei nennen wir die Dinge oder Eigenschaften bzw. Sachverhalte, die von Begriffen bzw. Aussagen bezeichnet werden, **Designate** (= „Bezeichnete") der Begriffe bzw. Aussagen.
Des näheren müßten wir noch zwischen „Wort und Satz" („auf dem Papier") und „Begriff und Aussage" („im Kopf") unterscheiden. Ferner müßten wir die erkenntnistheoretischen Zusammenhänge (Sachverhalt → Erkenntnis → Aussage → Satz) herausarbeiten. Dies würde jedoch den Rahmen unserer Darstellung sprengen. Wir begnügen uns deshalb mit einer kleinen Tabelle:

```
Ding       +  Eigenschaft  = Sachverhalt
D-Begriff  +  E-Begriff    = Aussage
D-Wort     +  E-Wort       = Satz
```

Begriffe haben einen Inhalt und Umfang. Der *Inhalt* des Begriffs „Baum" ist die Summe seiner Merkmale („groß", „wurzelhaltig", „hölzern" usw.), und der *Umfang* des Begriffs „Baum" ist die Summe seiner Designate („Buchen", „Tannen", „Fichten" usw.). Je größer der Begriffsumfang („Berberlöwe" → „Löwe" → „Raubtier" → „Tier" → „Lebewesen" → „Wesen"), desto geringer ist der Begriffsinhalt und umgekehrt.
Aussagen lassen sich in wahre und falsche Aussagen aufteilen. Eine Aussage ist **wahr bzw. falsch**, wenn die von der Aussage bezeichnete Klasse von Sachverhalten *existiert bzw. nicht existiert*. Der Nachweis der Wahrheit einer (elementaren, s.u.) Aussage (= **Verifikation**) erfolgt indessen nicht durch die Logik selbst, sondern durch die jeweilige Spezialwissenschaft (Physik, Biologie, Informatik usw.).

5. Klassen- und Prädikatenlogik

Die **Klassenlogik** („Dingbegriffslogik") befaßt sich mit dem Umfang von Begriffen bzw. mit den Klassen von Dingen und ihren Zusammenhängen. Hinsichtlich des Umfangs können sich zwei Begriffe teils überschneiden („Mann" und „Deutscher"), ganz überschneiden („Rotz" und „Nasenschleim") oder gar nicht überschneiden („Mann" und „Frau" in bezug auf den Oberbegriff „Mensch" oder „Mann" und „Nicht-Mann" (= Komplementbegriff) in bezug auf den Oberbegriff „Wesen"). Ferner können die Designate des einen Begriffs in der Klasse der Designate des anderen Begriffs enthalten sein („Tanne" – „Baum": Art – Gattung) oder umgekehrt („Baum" – „Tanne": Gattung – Art).
Aus der Klassenlogik ist später die **Mengenlehre** entstanden, die von der *Menge* statt von der Klasse, vom *Element* statt vom Designat, von der *Teilmenge* statt vom Artbegriff, von der *Obermenge* statt vom Gattungsbegriff, von der *Grundmenge* statt vom Gegenstandsbereich („universe of discourse") usw. spricht.
Ein Beispiel für eine Schlußfolgerung im Rahmen der Klassenlogik: „Wenn die Klasse der Löwen in der Klasse der Katzen enthalten ist und die Klasse der Katzen in der Klasse der Tiere, dann ist die Klasse der Löwen in der Klasse der Tiere enthalten." Dafür würde man normalerweise sagen: „Wenn Löwen Katzen und Katzen Tiere sind, dann sind Löwen Tiere."

Die **Prädikatenlogik** („Eigenschaftsbegriffslogik") befaßt sich mit dem Inhalt von Begriffen bzw. mit den Eigenschaften von Dingen und ihren Zusammenhängen. Da eine Eigenschaft *einem*, *einigen* oder *allen* Designaten eines Begriffs zukommen kann, spielen in der Prädikatenlogik die sog. Quantoren oder Quantifikatoren (Partikularisatoren „ein" und „einige"; Generalisator „alle") eine wichtige Rolle: Einzelaussagen beziehen sich auf einzelne Designate oder ein einziges Designat („Kant starb 1804"), während All(gemein)aussagen auf alle Designate eines Begriffs abheben („Alle Philosophen sind weise").
Ferner wird zwischen monadischen Eigenschaften („Elefanten sind groß") als attributiver Sachverhalt und dyadischen Eigenschaften („Elefanten sind größer als Nashörner" als relationaler Sachverhalt) unterschieden. Schließlich wird noch hinsichtlich der essentiellen und akzidentellen Eigenschaften zwischen analytischen (Erläuterungs)aussagen („Hunde sind Vierbeiner") und synthetischen (Erweiterungs)aussagen („Dieser Hund ist bissig") differenziert.
Ein Beispiel für eine Schlußfolgerung im Rahmen der Prädikatenlogik: „Wenn alle Wesen, denen die Eigenschaft „menschlich" zukommt, zugleich Wesen sind, denen die Eigenschaft „sterblich" zukommt, und wenn Sokrates eines der Wesen ist, denen die Eigenschaft „menschlich" zukommt, dann ist Sokrates eines der Wesen, denen auch die Eigenschaft „sterblich" zukommt." Dafür würde man normalerweise sagen: „Wenn alle Menschen sterblich sind und Sokrates ein Mensch ist, dann ist Sokrates sterblich."

6. Aussagenlogik

Im Gegensatz zur Klassen- und Prädikatenlogik, welche die Aussagen (= Urteile) in Ding- und Eigenschaftsbegriffe zerlegen und somit deren Komponenten untersuchen, werden in der Aussagenlogik die *Aussagen als Ganzes* betrachtet. Dabei

wird jedoch zwischen den einfachen oder *elementaren* Aussagen (= **Teilaussagen,** z.B. „Reden ist Silber") und den zusammengesetzten oder *komplexen* Aussagen (= **Gesamtaussagen,** z.B. „Reden ist Silber *und* Schweigen ist Gold") unterschieden, die auch als Aussagenverbindungen bezeichnet werden. Eine elementare Aussage ist in der zweiwertigen (oder binären) Aussagenlogik entweder wahr (eins, 1) oder falsch (null, 0). Ein Drittes gibt es nicht („tertium non datur"). „Halbwahrheiten", z.B. Aussagen mit Zusätzen wie „möglicherweise", „teils", „weitgehend" usw. scheiden damit aus (= mehrwertige Logik, Modallogik usw.).

6.1. Vergleichsaussagen

Wie in Abschnitt 4 erwähnt, wird in der Aussagenlogik die Wahrheit oder Falschheit einer elementaren Aussage als gegeben vorausgesetzt. In einem Computerprogramm, und darum geht es letztlich in diesem Beitrag, müssen jedoch praktische Entscheidungskriterien für die Verifikation von elementaren Aussagen bereitgestellt werden, denn ein Programm kann bei dem Befehl
IF 3 = 1 THEN ...
nicht über die „philosophische Identität der Dreieinigkeit" ins Grübeln geraten, weil es sonst schlechterdings „hängen" würde.
Für welche Typen von Aussagen lassen sich nun computermäßige Verifikationskriterien implementieren? Aussagen wie
„Alle Menschen sind sterblich"
„Pi ist eine transzendente Zahl"
„Heidelberg liegt am Neckar"
scheiden offenkundig aus, denn sie bezeichnen Sachverhalte, die jenseits der binären Welt des Computers liegen. Um sich dem Problem zu nähern, greifen wir einen der logischen Computerbefehle, und zwar den IF-THEN-Befehl, heraus und untersuchen, welche Aussagen zwischen IF und THEN eingesetzt werden können. Wir gelangen dann zu der Erkenntnis, daß nur Vergleichsaussagen zulässig sind:
1. Zu den **Vergleichsoperatoren** zählen „=" (gleich), „<" (kleiner) und „>" (größer) sowie deren Kombinationen „<=" (kleiner/gleich), „>=" (größer/gleich) und „<>" (ungleich bzw. kleiner oder größer).
2. Zu den **Vergleichsgrößen** zählen (a) von der Form her Literals, d.h. „wörtliche Werte" (123, „abc"), sowie Konstanten, denen feste Werte (PI = 3,1415), und Variablen, denen veränderliche Werte (X = 10) zugeordnet werden, und (b) vom Inhalt her Ganz- und Fließkommazahlen (12, 12.5), Zeichen und Zeichenketten („A", „ABC") und ggf. weitere Datentypen (Hires-Shapes usw.).

Vergleichsaussagen können neben den Vergleichsoperatoren noch datentypspezifische Operatoren sowie die weiter unten beschriebenen aussagenlogischen Operatoren enthalten. Zwischen IF und THEN können damit u.a. folgende Aussagen stehen (Beispiele):

Nur Literals:
IF 1 = 1 THEN...
IF "abc" <> "abd" THEN...
Literal + Variable/Konstante:
IF X < 1 THEN...
IF Y > "a" THEN...
Nur Variablen/Konstanten:
IF X <= Y THEN...
Datentypspezifische Operatoren:
IF (A + B) = (X * Y) THEN...
Aussagenlogische Operatoren:
IF (A = B) AND (X >= Y) THEN...

Vergleichsaussagen sind ein Sonderfall der relationalen Aussagen (vgl. die relationalen Sachverhalte in Abschnitt 3) und werden übrigens in der sog. mehrstelligen Prädikatenlogik näher untersucht (G.Klaus, Moderne Logik, S. 259ff.). Allerdings muß nachdrücklich betont werden, daß eine „normale", umgangssprachliche Vergleichsaussage, z.B. „Wenn Hans größer als Willi ist, dann...", erst nach einer entsprechenden Formalisierung, z.B. „IF H > W THEN...", in ein Computerprogramm eingebaut werden kann. Der Wahrheitswert einer computermäßigen Vergleichsaussage ergibt sich dann aus den in der jeweiligen Programmiersprache implementierten Vergleichsroutinen, die letztlich – implizit oder explizit – den binären Wert 1 für „wahr" und 0 für „falsch" liefern. So ist etwa der Wahrheitswert 1 bei
A = 10: IF A = 10 THEN...
implizit und bei
A = 10: PRINT (A = 10)
explizit gegeben.

Weitere Feinheiten

1. Null-eins-Konvention: Bei der Zuordnung von 0 = falsch und 1 = wahr handelt es sich um eine Vereinbarung, die keineswegs für alle Computersprachen gilt. Beispielsweise wird in MBASIC im Gegensatz zu Applesoft-BASIC der Wahrheitswert durch -1 und nicht durch +1 definiert. Im übrigen ist auch eine „negative Logik" möglich, also 0 = wahr und 1 = falsch.

2. Andere Vergleiche: Wir haben bislang nur den IF-THEN-Befehl behandelt. Analoges gilt auch für die anderen logischen Befehle:
ON-Beispiel (Applesoft):
10 ON A = B GOTO 30
20 ON A < B GOSUB 40
30 ... : END
40 ... : RETURN
WHILE-Beispiel (MBASIC):
10 WHILE F = 0
20 ...
30 IF ... THEN F = 1
40 WEND
REPEAT-Beispiel (Pascal):
REPEAT ...
UNTIL A >= 100;
WHILE-Beispiel (Pascal):
WHILE F = 0 DO
BEGIN ... END;

3. Versteckte Vergleiche: Diese findet man insbesondere bei den FOR-NEXT-Schleifen in BASIC und FOR-DO-Schleifen in Pascal. So wird etwa bei dem Applesoft-Programm
10 FOR I = 1 TO 10
20 PRINT SIN (I)
30 NEXT I
anstelle von „NEXT I" interpreter-intern „IF I <= 10 THEN GOTO 20" eingesetzt. Das gleiche gilt für die FOR-DO-Schleife in Pascal, z.B.

FOR I := 1 TO 10 DO WRITELN;
bei der jedoch kaum erkennbar ist, an welcher Stelle der Vergleich stattfindet. Weitere Pascal-Beispiele für versteckte Vergleiche sind die Dateibefehle
REPEAT... UNTIL EOLN (F); und
IF EOF (F) THEN...;
bei denen das Betriebssystem prüft, ob das Zeilen- oder Dateiende erreicht ist, sowie der Auswahlbefehl
CASE I OF
1: ...;
2: ...;
END;
bei dem geprüft wird, welche der Auswahlkonstanten mit dem Case-Selektor übereinstimmt.

4. Verkürzte Vergleiche: Bei einigen BASIC-Befehlen, z.B.
10 ON B GOTO 10, 20, 30
20 IF B THEN...
fehlt scheinbar der Vergleich. In Wirklichkeit wird etwa „IF B THEN.." nur dann ausgeführt, wenn B <> 0 ist, so daß man auch „IF B <> 0 THEN..." formulieren könnte.
Etwas anders liegen die Dinge bei einigen Pascal-Befehlen, z.B.
REPEAT ... UNTIL B;
WHILE NOT (B) DO...;
Diese Ausdrücke sind zulässig, weil Pascal über einen besonderen Wahrheitswert-Datentyp verfügt (BOOLEAN), der den Wert „wahr" (TRUE) bzw. „falsch" (FALSE) annehmen kann.

5. Kurioses: In Applesoft-BASIC ist es möglich, den Befehl
IF "Das Wahre ist eins"
THEN PRINT "Stimmt!"
auszuführen. Man könnte also meinen, daß Computerprogramme auch Nicht-Vergleichsaussagen verarbeiten können. Dem ist jedoch nicht so, denn bei „Das Wahre ist null" würde ebenfalls „Stimmt!" angezeigt werden.

6.2. Wahrheitswerttabellen

Um die elementaren Aussagen p und q zu komplexen Aussagen zu verknüpfen, bedient man sich der aussagenlogischen Operatoren NOT, AND, OR, IMP, EOR, EQV, NAND, NOR usw.

6.2.1. Monadischer Operator

p	NOT p
1	Ø
Ø	1

Eine einzelne Aussage p kann mit Hilfe des Negationsoperators NOT in ihr Gegenteil verkehrt werden. Wenn p wahr ist, dann ist NOT p falsch, und wenn p falsch ist, dann ist NOT p wahr. Beispiele:
Wenn p = „Kant starb 1804" eine wahre Aussage ist, dann ist NOT p = „Kant starb nicht 1804" = „Es ist nicht wahr, daß Kant 1804 starb" eine falsche Aussage. Wenn p = „3 ist kleiner als 2" eine falsche Aussage ist, dann ist NOT p = „Es ist nicht wahr, daß 3 kleiner als 2 ist" eine wahre Aussage.

6.2.2. Dyadische Operatoren

p	dOp	q	
1	x	1	1. Kombination
1	x	Ø	2. Kombination
Ø	x	1	3. Kombination
Ø	x	Ø	4. Kombination

Eine einzelne Aussage p ist entweder wahr oder falsch. Wenn man jedoch zwei Teilaussagen p und q mit Hilfe der sog. dyadischen Operatoren AND, OR usw. zu einer Gesamtaussage verknüpft, so ergeben sich 4 Kombinationen für die Teilaussagen, denn es können
p und q beide wahr oder
p wahr und q falsch oder
p falsch und q wahr oder
p und q beide falsch sein.
Nun können wir jedoch nicht nur den Teilaussagen, sondern auch den aus den Teilaussagen gebildeten Gesamtaussagen Wahrheitswerte zuweisen. Beispielsweise hat die Gesamtaussage „Ich stehe im Regen *und* warte auf dich" einen Wahrheitswert, der (a) von den Wahrheitswerten der Teilaussagen „Ich stehe im Regen" und „Ich warte auf dich" sowie (b) vom jeweiligen dyadischen Operator (hier „und") abhängt. Dies wird sofort einsichtig, wenn wir für „und" andere dyadische Operatoren substituieren, z.B.
„Ich stehe im Regen *oder* warte auf dich."
„*Entweder* stehe ich im Regen, *oder* ich warte auf dich."
„*Wenn* ich im Regen stehe, *dann* warte ich auf dich."
„Ich stehe *weder* im Regen, *noch* warte ich auf dich."
Da es in der obigen Tabelle 4 Leerstellen (markiert mit x) gibt, die je einen von 2 Werten (0 oder 1) annehmen können, gibt es insgesamt 2 hoch 4 = 16 Kombinationen und damit 16 dyadische Operatoren.

6.2.2.1. Seltenere Operatoren

Tautologie und Antilogie

p	q	Tautologie
1	1	1
1	Ø	1
Ø	1	1
Ø	Ø	1

p	q	Antilogie
1	1	Ø
1	Ø	Ø
Ø	1	Ø
Ø	Ø	Ø

Von den 16 dyadischen Operatoren scheiden zwei von vornherein aus, nämlich die Tautologie („gilt immer") und die die Antilogie („gilt nie"). Die Tautologie wird jedoch bei der Wahrheitswertentwicklung (s.u.) benötigt.

Präpendenz und Pränonpendenz
Postpendenz und Postnonpendenz

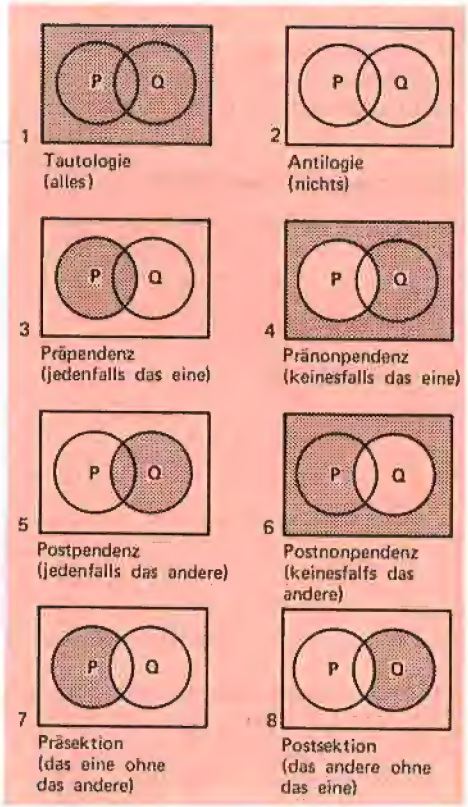
p	q	Präpendenz
1	1	1
1	Ø	Ø
Ø	1	1
Ø	Ø	Ø

p	q	Pränonpendenz
1	Ø	1
1	Ø	Ø
Ø	1	1
Ø	1	Ø

p	q	Postpendenz
1	1	1
1	Ø	Ø
Ø	1	1
Ø	Ø	Ø

p	q	Postnonpendenz
1	Ø	1
1	1	Ø
Ø	Ø	1
Ø	1	Ø

Diese vier dyadischen Operatoren werden mit „jedenfalls/keinesfalls" sowie mit „gleichviel ob" konstruiert. Beispielsätze:
Präpendenz: „Ich gehe *jedenfalls* morgen in die Schule, *gleichviel ob* ich in die Disco gehe oder nicht."
Pränonpendenz: „Ich gehe morgen *keinesfalls* in die Disco, *gleichviel ob* ich in die Schule gehe oder nicht."
Postpendenz: „*Gleichviel ob* ich morgen in die Disco gehe oder nicht, *jedenfalls* gehe ich in die Schule."
Postnopendenz: „*Gleichviel ob* ich morgen in die Schule gehe oder nicht, *keinesfalls* gehe ich in die Disco."



Wahrheitswerttest
Um sich von der Richtigkeit einer Wahrheitswerttabelle zu überzeugen, braucht man nur für die jeweilige Aussage alle vier Kombinationen durchzuspielen. Betrachten wir hierzu die Präpendenz-Gesamtaussage „Ich gehe jedenfalls morgen in die Schule, gleichviel ob ich morgen in die Disco gehe oder nicht" in Verbindung mit den vier Kombinationen in der Präpendenz-Tabelle:
1 1 1 - 1. Kombination: wahr
1 1 0 - 2. Kombination: wahr
0 0 1 - 3. Kombination: falsch
0 0 0 - 4. Kombination: falsch
Die Gesamtaussage ist offenbar nur dann wahr, wenn die Teilaussage „Ich gehe morgen in die Schule" wahr ist. Dies gilt für die ersten beiden Kombinationen.

Wenn dagegen die Teilaussage „Ich gehe morgen in die Schule" falsch ist, dann ist es bei der Präpendenz irrelevant, ob die Teilaussage „Ich gehe morgen in die Disco" wahr oder falsch ist. Dies gilt für die letzten beiden Kombinationen.
Wenn wir für die falschen Teilaussagen deren Negationen einsetzen, erhalten wir:
1. „Ich gehe morgen in die Schule" + „Ich gehe morgen in die Disco" → Gesamtaussage wahr.
2. „Ich gehe morgen in die Schule" + „Ich gehe morgen nicht in die Disco" → Gesamtaussage wahr.
3. „Ich gehe morgen nicht in die Schule" + „Ich gehe morgen in die Disco" → Gesamtaussage falsch.
4. „Ich gehe morgen nicht in die Schule" + „Ich gehe morgen nicht in die Disco" → Gesamtaussage falsch.

Präsektion und Postsektion

p	q	Präsektion
1	Ø	1
1	Ø	Ø
Ø	1	1
Ø	Ø	Ø

p	q	Postsektion
1	Ø	1
1	1	Ø
Ø	Ø	1
Ø	Ø	Ø

Diese dyadischen Operatoren werden mit „zwar (nicht), aber dafür (nicht) bzw. sondern" konstruiert. Beispielsätze:
Präsektion: „Ich gehe morgen *zwar nicht* in die Disco, *aber dafür* in die Schule" oder „Ich gehe morgen *nicht* in die Disco, *sondern* in die Schule."
Postsektion: „Ich gehe *zwar* morgen in die Schule, *aber dafür nicht* in die Disco."

6.2.2.2. Häufigere Operatoren

Während die bislang diskutierten Operatoren in keiner mir bekannten Computersprache implementiert sind, findet man die nachfolgenden Operatoren zumindest ansatzmäßig in den meisten Programmiersprachen.

Konjunktion und Rejektion

p	q	Konjunktion AND
1	1	1
1	Ø	Ø
Ø	Ø	1
Ø	Ø	Ø

p	q	Rejektion NOR
1	Ø	1
1	Ø	Ø
Ø	Ø	1
Ø	1	Ø

Die Konjunktion wird mit „und" (AND: „beide") und die Rejektion mit „weder noch" (NOR: „beide nicht", „keines von beiden") konstruiert. Beispielsätze:
Konjunktion: „Bonn liegt am Rhein *und* (Bonn) ist die Hauptstadt Deutschlands."
Rejektion: „Frankfurt liegt *weder* am Rhein, *noch* ist Frankfurt die Hauptstadt Deutschlands."

Disjunktion und Exklusion

p	q	Disjunktion OR
1	1	1
1	1	Ø
Ø	1	1
Ø	Ø	Ø

p	q	Exklusion NAND
1	Ø	1
1	1	Ø
Ø	1	1
Ø	1	Ø

Die Disjunktion wird durch das einschließende „oder" (OR; „mindestens eines"; „oder ... oder beides"; „und/oder") und die Exklusion durch das ausschließende „oder" (NAND; „höchstens eines"; „nicht beides"; vgl. Kontravalenz) konstruiert. Beispielsätze:
Disjunktion: „Morgen gehe ich in die Schule *oder* in die Disco (oder beides)."

Exklusion: „Das Auto dort drüben ist ein Opel *oder* ein Ford (oder keines von beiden)."

Äquivalenz und Kontravalenz

```
p   q Äquivalenz EQV
─────
1 1 1
1 Ø Ø
Ø Ø 1
Ø 1 Ø

p   q Kontravalenz EOR
─────
1 Ø 1
1 1 Ø
Ø 1 1
Ø Ø Ø
```

Die Äquivalenz wird mit „genau dann, wenn“ oder „stets und nur dann, wenn“ (EQV) und die Kontravalenz mit „entweder, oder“ (EOR, XOR, EXOR = „exclusive or“) konstruiert. Die Äquivalenz ist eine Grund-Folge-Beziehung, bei der die hinreichende Bedingung zugleich notwendige Bedingung ist (Implikation und Replikation zugleich, s.u.). Beispielsätze:
Äquivalenz: „*Wenn* morgen Dienstag ist, ist übermorgen Mittwoch.“
Kontravalenz: „*Entweder* ich bestehe die Prüfung, *oder* ich falle durch.“

Implikation und Replikation

```
p   q Implikation IMP
─────
1 1 1
1 Ø Ø
Ø 1 1
Ø 1 Ø

p   q Replikation (REP)
─────
1 1 1
1 1 Ø
Ø Ø 1
Ø 1 Ø
```

Die Implikation wird mit „wenn, dann“ (IMP) und die Replikation mit „Nur wenn, dann“ konstruiert. (Hinweis: REP kommt als Befehl offenbar in keiner Programmiersprache vor.) Implikation und Replikation sind Grund-Folge-Beziehungen: Die Implikation entspricht der „hinreichenden Bedingung“ – die Folge kann mehrere Gründe haben –, während die Replikation der „notwendigen Bedingung“ entspricht – die Folge hat nur den einen genannten Grund. Beispielsätze:
Implikation: „*Wenn* die Sonne scheint, ist es Tag.“
Replikation: „*Nur wenn* Sauerstoff vorhanden ist, brennt Schwefel.“

IMP versus IF

Man verwechsle nicht den Wenn-Teil der IMP-*Aussage* mit dem IF-Teil des IF-THEN-*Befehls*. Die Konstruktion „IF Aussage THEN Befehl“ ist *insgesamt ein Befehl*, wobei zwar der IF-Teil eine Aussage ist, aber der THEN-Teil einen Befehl darstellt. Dagegen ist die Konstruktion „Wenn Aussage, dann Aussage“ *insgesamt eine Aussage*, wobei Wenn-Teil und Dann-Teil beides Aussagen sind. Der IF-Teil des IF-THEN-Befehls wird nicht nur durch eine elementare Vergleichsaussage, sondern auch durch eine beliebige Aussagenverbindung (IMP-Aussage, AND-Aussage, OR-Aussage usw.) erfüllt, doch der THEN-Befehl wird nur dann ausgeführt, wenn der IF-Teil wahr ist. So wird etwa der THEN-Befehl in
IF (P IMP Q) THEN PRINT “Befehl“
nicht ausgeführt, wenn gilt P = 1 und Q = 0.

6.3. MBASIC-Operatoren

Im Gegensatz zu Applesoft-BASIC, das nur über die Operatoren NOT (Negation), AND (Konjunktion) und OR (Disjunktion) verfügt, gibt es in CP/M-MBASIC noch die zusätzlichen Operatoren XOR (Kontravalenz), IMP (Implikation) und EQV (Äquivalenz). Das nachfolgende Beispielprogramm **MLOGIK** erzeugt die entsprechenden Wahrheitswerttabellen (s.o. Abschnitt 6.2.2). Man beachte insbesondere, daß die FOR-NEXT-Schleifen mit -1 beginnen, weil in MBASIC -1 für „wahr“ steht. In den Zeilen 190 usw. wird das Minuszeichen dann wieder entfernt.

```
1ØØ HOME: REM *** MLOGIK ***
11Ø PRINT "MBASIC-Wahrheitswerttabellen"
12Ø PRINT
13Ø FOR A = 1 TO 5
14Ø PRINT " P ";
15Ø READ X$: PRINT X$;
16Ø PRINT " Q"
17Ø FOR P = -1 TO Ø
18Ø FOR Q = -1 TO Ø
19Ø PRINT ABS (P); " ";
2ØØ ON A GOSUB 25Ø, 26Ø, 27Ø, 28Ø, 29Ø
21Ø PRINT " "; ABS (Q)
22Ø NEXT Q,P
23Ø PRINT: NEXT A
24Ø END
25Ø PRINT ABS (P AND Q);: RETURN
26Ø PRINT ABS (P OR  Q);: RETURN
27Ø PRINT ABS (P XOR Q);: RETURN
28Ø PRINT ABS (P IMP Q);: RETURN
29Ø PRINT ABS (P EQV Q);: RETURN
3ØØ DATA AND,"OR ",XOR,IMP,EQV
```

6.4. Reduktion der Operatoren

Da Applesoft-BASIC und Pascal über weniger Operatoren als MBASIC verfügen und auch in MBASIC nur wenige der 16 möglichen dyadischen Aussagenoperatoren implementiert sind, stellt sich die Frage, ob und durch welche Operatoren die nicht-vorhandenen Operatoren ausgedrückt werden können. Es läßt sich zeigen, daß man beispielsweise mit „NOT und OR“ oder „NOT und AND“ alle anderen Operatoren umschreiben kann. Es ist sogar möglich, sowohl den monadischen Operator NOT als auch alle anderen dyadischen Operatoren auf den Exklusionsoperator NAND zu reduzieren, wie M.H.Shefer bereits 1913 nachgewiesen hat.

Das nachfolgende Applesoft-Programm **LOGIK** reduziert alle dyadischen Operatoren auf AND und NOT und erzeugt die entsprechenden Wahrheitswerttabellen (s.o. Abschnitt 6.2.2). Man beachte dabei insbesondere, wie die Tautologie und die Antilogie konstruiert worden sind.

```
1ØØ HOME :
    PRINT "Wahrheitswerttabellen":
    PRINT "auf Basis von AND+NOT":
    PRINT: REM *** LOGIK ***
11Ø FOR A = 1 TO 16
12Ø PRINT "p   q ";
13Ø READ A$: PRINT A$
14Ø PRINT "-----"
15Ø FOR P = 1 TO Ø STEP - 1
16Ø FOR Q = 1 TO Ø STEP - 1
17Ø PRINT P;" ";
18Ø ON A GOSUB 25Ø,26Ø,27Ø,28Ø,29Ø,
    3ØØ,31Ø,32Ø,33Ø,34Ø,35Ø,36Ø,
    37Ø,38Ø,39Ø,4ØØ
19Ø PRINT " ";Q
2ØØ NEXT Q,P: PRINT
21Ø GET X$
22Ø NEXT A
23Ø END
24Ø DATA Tautologie,Disjunktion,
    Replikation,Präpendenz,
    Implikation,Postpendenz,
    Äquivalenz,Konjunktion,
    Exklusion,Kontravalenz,
    Postnonpendenz,Postsektion,
    Pränonpendenz,Präsektion,
    Rejektion,Antilogie
25Ø PRINT NOT ((P AND Q) AND
    (NOT P AND NOT Q));: RETURN:
    REM Tautologie
26Ø PRINT NOT ( NOT P AND NOT Q);:
    RETURN : REM Disjunktion
27Ø PRINT NOT ( NOT P AND Q);: RETURN:
    REM Replikation
28Ø PRINT P;: RETURN : REM Präpendenz
29Ø PRINT NOT (P AND NOT Q);: RETURN:
    REM Implikation
3ØØ PRINT Q;: RETURN: REM Postpendenz
31Ø PRINT NOT (P AND NOT Q)
    AND NOT ( NOT P AND Q);: RETURN:
    REM Äquivalenz
32Ø PRINT P AND Q;: RETURN:
    REM Konjunktion
33Ø PRINT NOT (P AND Q);: RETURN:
    REM Exklusion
34Ø PRINT NOT (P AND Q) AND NOT
    (NOT P AND NOT Q);: RETURN:
    REM Kontravalenz
35Ø PRINT NOT Q;: RETURN:
    REM Postnonpendenz
36Ø PRINT P AND NOT Q;: RETURN:
    REM Postsektion
37Ø PRINT NOT P;: RETURN :
    REM Pränonpendenz
38Ø PRINT NOT P AND Q;: RETURN:
    REM Präsektion
39Ø PRINT NOT P AND NOT Q;: RETURN:
    REM Rejektion
4ØØ PRINT (P AND Q) AND
    (NOT P AND NOT Q);: RETURN:
    REM Antilogie
41Ø REM U.Stiehl/Jan.86
(Schulen usw. können zu Übungszwecken
dieses Programm so umschreiben,
daß nur NOT und OR verwendet werden.)
```

6.5. Logische Gleichungen

Logische Gleichungen sind Äquivalenzen zwischen Aussagenverbindungen, die in allen Fällen wahr sind (= Tautologien). Aus der Fülle dieser sog. „logischen Gesetze“ greifen wir vier Gruppen heraus:

Symmetriegesetze

Bei bestimmten Operatoren können p und q vertauscht werden, z.B.

```
(p AND q) EQV (q AND p)
(p OR  q) EQV (q OR  p)
(p EQV q) EQV (q EQV p)
(p EOR q) EQV (q EOR p)
(p NOR q) EQV (q NOR p)
```

Umkehrgesetze

Bei bestimmten Operatoren können p und q vertauscht werden, wenn gleichzeitig die entsprechenden Umkehroperatoren eingesetzt werden, z.B. die Implikation-Replikation-Umkehrung:

```
(p IMP q) EQV (q REP p)
```

Kontrapositionsgesetze

Bei bestimmten Operatoren können p und q vertauscht werden, wenn zusätzlich Negationen vorgenommen werden. Für die Implikation ergeben sich dann beispielsweise vier Kombinationen:

```
(     p IMP     q) EQV (NOT q IMP NOT p)
(     p IMP NOT q) EQV (     q IMP NOT p)
(NOT p IMP     q) EQV (NOT q IMP     p)
(NOT p IMP NOT q) EQV (     q IMP     p)
```

Diese sog. „Kontraponibilität der Implikation" wollen wir an einem Beispielsatz verdeutlichen. Wenn gilt „Wenn die Sonne scheint, dann ist es Tag", dann gilt auch „Wenn nicht Tag ist, dann scheint die Sonne nicht".

de-Morgansche Gesetze

Dem Logiker Augustus de Morgan (1806-1870) werden fälschlicherweise in mathematischen Schulbüchern vier Gesetze zugeschrieben, die in Wirklichkeit bereits den Scholastikern bekannt waren (vgl. Bochenski/Menne, Grundriß der Logistik, S. 38). Es handelt sich dabei um OR-AND-Umwandlungen mit „distributiver" Änderung der Negation:

```
(NOT (p OR  q)) EQV (     NOT p AND NOT q)
(NOT (p AND q)) EQV (     NOT p OR  NOT q)
(     p OR  q)  EQV (NOT (NOT p AND NOT q))
(     p AND q)  EQV (NOT (NOT p OR  NOT q))
```

Das 1. de-Morgansche Gesetz reduziert die Rejektion (NOR) auf eine NOT-OR- (linke Hälfte der Gleichung) bzw. auf eine NOT-AND-Kombination (rechte Hälfte der Gleichung). Das 2. Gesetz reduziert die Exklusion (NAND) auf eine NOT-AND- bzw. NOT-OR-Kombination. Das 3. Gesetz umschreibt OR durch eine NOT-AND-Kombination und das 4. Gesetz schließlich umschreibt AND durch eine NOT-OR-Kombination. Man vergleiche hierzu die entsprechenden Zeilen des Applesoft-Progamms LOGIK.

6.6. Logische Schlüsse

Logische Schlüsse sind Implikationen zwischen Aussagenverbindungen, die in allen Fällen wahr sind (= Tautologien). Aus der Fülle der logischen Schlüsse greifen wir vier „Modi" heraus:

```
1. ((p IMP  q) AND     p) IMP     q
2. ((p IMP  q) AND NOT q) IMP NOT p
3. ((p OR   q) AND NOT p) IMP     q
4. ((p NAND q) AND     p) IMP NOT q
```

Nachfolgend bringen wir einige (zur Verdeutlichung bewußt restriktiv gewählte) Beispielsätze für diese Modi:

1. Modus ponendo ponens: „Wenn heute Dienstag ist, dann ist morgen Mittwoch, und da nun einmal heute Dienstag ist, so muß wohl morgen Mittwoch sein."
2. Modus tollendo tollens: „Wenn heute Dienstag ist, dann ist morgen Mittwoch, und da nun einmal heute kein Mittwoch ist, dann war gestern wohl auch kein Dienstag."
3. Modus tollendo ponens: „Mein Hemd ist eingegangen oder ich habe zugenommen, und da mein Hemd nun einmal nicht eingegangen ist, so habe ich wohl zugenommen."
4. Modus ponendo tollens: „Mein Hund ist ein Männchen oder ein Weibchen, und da

Ausgewählte Äquivalenzen

```
1000  AND   Konjunktion   beides
0001  NOR   Rejektion     keines
111Ø  OR    Disjunktion   mindestens eines von beiden
Ø111  NAND  Exklusion     höchstens eines von beiden
1ØØ1  EQV   Äquivalenz    beides oder keines
Ø11Ø  EOR   Kontravalenz  eines von beiden
1Ø11  IMP   Implikation   das eine nicht ohne das andere
11Ø1  REP   Replikation   das andere nicht ohne das eine
```

Es gelten folgende Prioritätsregeln:
1. Klammern
2. NOT
3. AND
4. NOR, OR, NAND, EQV, EOR, IMP, REP

1 Aussage P

```
P EQV P                          Identität
(P AND P) EQV P                  Idempotenz
(P OR P) EQV P                   Idempotenz
```

Negation NOT

```
(NOT NOT P) EQV P                Doppelte Verneinung
(NOT NOT NOT P) EQV (NOT P)      Dreifache Verneinung
```

2 Aussagen P und Q

Konjunktion AND

```
(P AND Q) EQV (Q AND P)          Symmetrie (Kommutativität)
(P AND Q) EQV (NOT (P NAND Q))   Reduktion auf Exklusion
(P AND Q) EQV (NOT (P IMP NOT Q)) Reduktion auf Implikation
(P AND Q) EQV (NOT (NOT P OR NOT Q) Reduktion auf Disjunktion
(P AND (P OR Q)) EQV P           Absorption
(P AND (P AND Q)) EQV (P AND Q)  Absorption
```

Rejektion NOR

```
(P NOR Q) EQV (Q NOR P)          Symmetrie (Kommutativität)
(P NOR Q) EQV (NOT (P OR Q))     Reduktion auf Disjunktion
(P NOR Q) EQV (NOT P AND NOT Q)  Reduktion auf Konjunktion
```

Disjunktion OR

```
(P OR Q) EQV (Q OR P)            Symmetrie (Kommutativität)
(P OR Q) EQV (NOT P NAND NOT Q)  Reduktion auf Exklusion
(P OR Q) EQV (NOT P IMP Q)       Reduktion auf Implikation
(P OR Q) EQV (NOT Q IMP P)       Reduktion auf Implikation
(P OR Q) EQV (NOT (NOT P AND NOT Q)) Reduktion auf Konjunktion
(P OR (P OR Q)) EQV (P OR Q)     Absorption
(P OR P AND Q) EQV P             Absorption
```

er nun einmal ein Männchen ist, so kann er wohl kein Weibchen sein."

Für den letzten Modus (Exklusionsschluß) führen wir eine sog. **Wahrheitswertentwicklung** durch:

```
 1  2   3    4    5   6      7  8
(p NAND q)  AND   p)  IMP  (NOT  q)
 1  Ø   1    Ø    1   1      Ø (1)
 1  1   Ø    1    1   1      1 (Ø)
 Ø  1   1    Ø    Ø   1      Ø (1)
 Ø  1   Ø    Ø    Ø   1      1 (Ø)
    *        *    *   *      *
    ***************  *      *
                 *   *      *
                 ***********
```

In den p-Spalten 1 und 5 setzen wir die übliche 1100-Folge und in der q-Spalte 3 die übliche 1010-Folge ein. In der NOT-q-Spalte 7 setzen wir jedoch wegen NOT nicht die 1010-Folge von Spalte 8 (eingeklammert), sondern deren 0101-Umkehrfolge ein. Dann tragen wir in Spalte 2 die 0111-NAND-Folge ein.
Nunmehr ermitteln wir die AND-Folge in Spalte 4 als Kombination aus den Spalten 2 und 5. AND ist nur 1, wenn p = 1 und q = 1 sind, folglich ergibt sich 0100.
Schließlich ermitteln wir die IMP-Folge in Spalte 6 als Kombination aus den Spalten 4 und 7. IMP ist nur dann 0, wenn p = 1 und q = 0. Diesen Fall finden wir hier nicht. Folglich ist die Implikation *allgemeingültig* und somit ein korrekter logischer Schluß.

Literatur

K.Ajdukiewicz: Abriß der Logik, Berlin 1958.
I.M.Bochenski und A. Menne: Grundriß der Logistik, Paderborn 1965.
G.Klaus: Moderne Logik, Berlin 1967.
F.v.Kutschera und A.Breitkopf: Einführung in die moderne Logik, Freiburg 1979
O.Neufang: Digitale Systeme, Teil 1: Schaltnetze 1976, Teil 2: Schaltwerke 1979.
U.Stiehl: Einführung in die allgemeine Semantik, Bern 1970.
M.Troitzsch: Mikrocomputer-Schaltungstechnik, München 1984.
L.Wittgenstein: Tractatus logico-philosophicus, Frankfurt 1960.

Kurzhinweise

1. Zweck: Darstellung der (aussagen)logischen Grundlagen für Programmierer.
2. Konfiguration: Apple II+/e/c
3. Test: RUN LOGIK (unter Applesoft und DOS 3.3 oder ProDOS); RUN MLOGIK (unter CP/M-MBASIC)
4. Sammeldisk: LOGIK (Applesoft) MLOGIK (MBASIC-Textfile)
5. Sonstiges: MLOGIK muß zunächst mit APDOS von der Sammeldisk auf Ihre CP/M-MBASIC-Arbeitsdiskette konvertiert werden.

Exklusion NAND

```
(P NAND Q) EQV (Q NAND P)                    Symmetrie (Kommutativität)
(P NAND P) EQV NOT P                         Reduktion auf Negation
(P NAND Q) EQV (P IMP NOT Q)                 Reduktion auf Implikation
(P NAND Q) EQV (NOT P OR NOT Q)              Reduktion auf Disjunktion
(P NAND Q) EQV (NOT (P AND Q))               Reduktion auf Konjunktion
```

Äquivalenz EQV

```
(P EQV Q) EQV (Q EQV P)                      Symmetrie (Kommutativität)
(P EQV Q) EQV (NOT P EQV NOT Q)              Inversion
(P EQV Q) EQV (NOT Q EQV NOT P)              Kontraposition der Äquivalenz
(P EQV Q) EQV (NOT (P EOR Q))                Reduktion auf Kontravalenz
(P EQV Q) EQV (P AND Q OR NOT P AND NOT Q)   Umschreibung mit AND/OR
```

Kontravalenz EOR

```
(P EOR Q) EQV (Q EOR P)                      Symmetrie (Kommutativität)
(P EOR Q) EQV (NOT P EOR NOT Q)              Inversion
(P EOR Q) EQV (NOT Q EOR NOT P)              Kontraposition der Kontravalenz
(P EOR Q) EQV (NOT (P EQV Q))                Reduktion auf Äquivalenz
(P EOR Q) EQV (P AND NOT Q OR NOT P AND Q)   Umschreibung mit AND/OR
```

Implikation IMP

```
(P IMP Q) EQV (NOT Q IMP NOT P)              Kontraposition der Implikation
(P IMP Q) EQV (Q REP P)                      Reduktion auf Replikation
(P IMP Q) EQV (P NAND NOT Q)                 Reduktion auf Exklusion
(P IMP Q) EQV (NOT P OR Q)                   Reduktion auf Disjunktion
(P IMP Q) EQV (NOT (P AND NOT Q))            Reduktion auf Konjunktion
```

Replikation REP

```
(P REP Q) EQV (NOT Q REP NOT P)              Kontraposition der Replikation
(P REP Q) EQV (Q IMP P)                      Reduktion auf Implikation
(P REP Q) EQV (NOT P NAND Q)                 Reduktion auf Exklusion
(P REP Q) EQV (P OR NOT Q)                   Reduktion auf Disjunktion
(P REP Q) EQV (NOT (NOT P AND Q))            Reduktion auf Konjunktion
```

3 Aussagen P, Q und R

```
(P OR (Q OR R)) EQV ((P OR Q) OR R)          Assoziativität der Disjunktion
(P AND (Q AND R)) EQV ((P AND Q) AND R)      Assoziativität der Konjunktion
(P OR Q AND R) EQV ((P OR Q) AND (P OR R))   Distributivität der Disjunktion
(P AND (Q OR R)) EQV (P AND Q OR P AND R)    Distributivität der Konjunktion
```

Wenn eine Gesamtaussage aus 3 Teilaussagen besteht, dann ergeben sich 8 Wahrheitswertkombinationen, z.B. bei der Distributivität der Disjunktion:

```
FOR P = 1 TO 0 STEP -1:
FOR Q = 1 TO 0 STEP -1:
FOR R = 1 TO 0 STEP -1:
PRINT (P OR (Q AND R)) =
((P OR Q) AND (P OR R)):
NEXT R: NEXT Q: NEXT P
```

Man beachte, daß in Applesoft-BASIC „=" für EQV eintreten kann.
Die Wahrheitswertentwicklung sieht hier folgendermaßen aus:

```
P  Q  R  (P OR (Q AND R)) EQV ((P OR Q) AND (P OR R))
1  1  1     1       1       1      1     1     1
1  1  Ø     1       Ø       1      1     1     1
1  Ø  1     1       Ø       1      1     1     1
1  Ø  Ø     1       Ø       1      1     1     1
Ø  1  1     1       1       1      1     1     1
Ø  1  Ø     Ø       Ø       1      1     Ø     Ø
Ø  Ø  1     Ø       Ø       1      Ø     Ø     1
Ø  Ø  Ø     Ø       Ø       1      Ø     Ø     Ø
            *               *             *
            *               *             *
            ********************************
```

EDIT

Ein Luxus-Disk-Editor für ProDOS

von Arne Schäpers

Teil 3: Die Makrosprache

1. Kommando-Übersicht

Dieser dritte und letzte Teil befaßt sich mit einigen weiteren Funktionen und mit den Makrobefehlen des Diskketteneditors. Zuerst die neuen Funktionen:

PSx: Setzt die Slotnummer des Druckers, x im Bereich 1..7; Beispiel: PS1.
PD: druckt den Bildschirm aus;
Beispiel: PD.

N: schaltet das DUMP-Display ab;
Beispiel: N.
O: schaltet das DUMP-Display wieder an;
Beispiel: O.

Fxxx:yy yy yy ... sucht BUFFER ab xxx nach Hexzahlen yy yy yy ... ab; Beispiel: F000:20 58 FC.

Fxxx'aaa... sucht BUFFER ab xxx nach ASCII aaa.. mit gelöschtem MSB ab;
Beispiel: F100'COPYRIGHT.
Fxxx"aaa... sucht BUFFER ab xxx nach ASCII aaa.. mit gesetztem MSB ab;
Beispiel: F080"COPYRIGHT.
Fxxx= wiederholt die letzte FIND-Operation ab Adresse xxx; Beispiel: F000=.

Zyy: wartet yy/16 Sekunden und tut sonst nichts; Beispiel: Z0F (s.u. Routine SLEEP).

!Yx ist ein „Label", x im Bereich von 0..9;
J <+/–>x springt zu diesem Label;
% mit folgendem „A"..„Z" ist eine Variablenfunktion.

Einige Anmerkungen

„N" und „O" schalten das DUMP-Display an bzw. aus und sind nur für Schleifen gedacht, die blockweise auf einer (RAM) Disk suchen, austauschen usw. Wenn das Display „O" („on") ist, wird ansonsten nach einem „R"ead oder „M"ove der Inhalt von BUFFER neu gedruckt und dadurch das Arbeitstempo von EDIT stark herabgesetzt. Das Kommando „N" löscht das Display-Feld und schreibt „NO DISPLAY" in die Zeile zwischen der INFO und dem Display-Feld.

Die Funktion „PSx" sollte ebenfalls klar sein. Mit ihr kann die Slotnummer eines angeschlossenen Druckers gesetzt werden. Beim ersten Aufruf von EDIT hat der

Drucker standardmäßig die Slotnummer 1. „PD“ druckt den Bildschirm aus, und zwar mit einigen Besonderheiten:
– Es wird immer auf 80 Zeichen Breite ausgedruckt, auch wenn die Schirmbreite nur 40 Zeichen beträgt.
– Wenn mit dem letzten Kommando eine Obergrenze angegeben wurde, also z.B. „D0,30“, dann wird bis zu dieser Obergrenze gedruckt, ansonsten der gesamte BUFFER.

Die Funktion „F“ind ist von der Syntax her in drei Fällen mit „S“et identisch, das Format ist für
„F“ind <Start>:Hex Hex ..,
„F“ind <Start>’ASCII.. und
„F“ind <Start>“ASCII.. dasselbe.
„Start“ steht hier in spitzen Klammern und ist genauso wie bei „S“et optional. Bei angegebener Startadresse wird ab „Start“ gesucht, ansonsten ab der Startadresse + 1(!) des letzten „F“ind-Kommandos.
„F<Start>=“ benutzt das bei einem vorherigen „F“ind angegebene Argument noch einmal.

Die etwas merkwürdige „Startadresse + 1“ hat folgenden Grund:
Wenn Sie innerhalb eines Blocks alle Stellen finden wollen, in denen z.B. die Zeichenfolge „***“ steht, reicht damit ein „F000’***“. Das nächste „F=“ sucht dann (vorausgesetzt, es wurde mindestens einmal „***“ gefunden) nach dem nächsten Auftreten von „***“ innerhalb dieses Blocks.
Es gibt noch eine letzte (sehr komplizierte) Syntax für „F“ind:
„F<Start>:Hex MHex Hex Hex MHex Hex MHex....“
Ein vorangestelltes „M“ innerhalb der Folge von Hexzahlen wird als AND-Maske für das zu findende Hexbyte genommen, bevor mit dem davor stehenden Byte verglichen wird. Einige Beispiele:
F000:0D M7F findet die Hexbytes $0D und $8D;
F000:00 M00 ist ein „Wildcard“, d.h. jedes Zeichen wird „gefunden“;
F000:01 M00 würde nie gefunden.

Das Kommando „Z“ tut überhaupt nichts – hier wird lediglich eine gewisse Zeit verzögert. „Z10“ wartet eine Sekunde, „Z20“ wartet zwei Sekunden usw. Dieses Kommando ist manchmal ganz nützlich, wenn man innerhalb einer Schleife Daten liest und dabei pro Block einen Blick auf den Bildschirm werfen will.

Mit den Funktionen „Y“ps und „J“ump kommen wir endlich zum Sinn des Flags SUCCEED:
„J“ump kann auf drei Arten erfolgen:
Jx: springt unkonditional zum Label x;
J+x: springt zum Label x, wenn SUCCEED auf „+“ gesetzt ist;
J-x: springt zum Label x, wenn SUCCEED auf „-“ gesetzt ist.
SUCCEED selber wird durch die folgenden Operationen gesetzt:
„F“ind: „+“, wenn FIND erfolgreich war, sonst „-“;
„C“ompare: „+“, wenn HLBUF und BUFFER im verglichenen Bereich vollständig übereinstimmen, sonst „-“;
Variablen- und Wertvergleiche: „+“ wenn gleich, sonst „-“;
Variablenaddition und -subtraktion. Wenn der Wert 00 erreicht wird, wird SUCCEED auf „+“, ansonsten auf „-“ gesetzt.
„Y“ps selber ist eine leere Funktion und wird nur auf Syntax geprüft, wenn der Command-Interpreter zu diesem Kommando verzweigt.

Die Variablen

In EDIT sind insgesamt 26 16-Bit-Variablen definiert, ihre Namen gehen von „A“..„Z“. Beim Start von EDIT sind diese Variablen undefiniert. Da die Argument-Berechnung für sämtliche Funktionen außer „Printer Slot“ und „Read/Write Slot/Drive“ über EVALHEX geht, ist damit jeder beliebige Zahlenwert einer Funktion durch eine Variable ersetzbar. Eine Variable kann somit wiederum durch eine andere Variable gesetzt werden.
Eine Variable wird durch ein vorangestelltes „%“ gekennzeichnet. Alle „Literals“, d.h. feste Zahlenwerte, sind in Hex.
Folgende Zuordnungen sind definiert:
%A=xx ordnet der Variablen A den Wert auf der rechten Seite der Gleichung zu. Dieser Wert kann wieder eine Variable sein.
%A+xx addiert xx zum Wert von A und setzt SUCCEED auf „+“, wenn dabei der Wert 0 erreicht wird.
%A-xx subtrahiert xx von A und setzt ebenfalls SUCCEED auf „+“, wenn dabei Null erreicht wird. In beiden Fällen kann xx eine weitere Variable sein.
%A(xx ordnet A den Wert des xxten Bytes in BUFFER zu, sinngemäß ordnet
%A(%B A den Wert des Bten Bytes in BUFFER zu.
%A?xx vergleicht A mit xx und setzt SUCCEED entsprechend auf „+“ oder „-“. Genauso ist die Anweisung
%A?%B zulässig, die die Variablen A und B vergleicht.
Was hier nicht unterstützt wird, sind zusammengesetzte Ausdrücke wie z.B. „%A?%B+100“, hier muß erst die rechte Seite einzeln berechnet werden, also in diesem Fall mit
%C=%B!%C+100. Erst danach kann ein Vergleich mit
%A?%C erfolgen.

2. Makrobefehle

– Eine einfache Schleife ohne Inhalt („FOR X= 0 to 16 : NEXT“):
%A=0 !Y0!%A+!%A?11!J-0
Dieser Loop springt solange wieder zu Y0, bis A den Wert $11 erreicht hat.
– Eine Schleife mit Inhalt, die die Blocks $000 bis $010 liest und auf den Bildschirm druckt („FOR X= 0 to 16: ’READ BLOCK X’ : NEXT“):
%A=0 !Y0!R%A!D!%A+!A?11!J-0

Variablen können wirklich überall anstelle von Festwerten verwendet werden. Das folgende Beispiel setzt die ersten $100 Bytes von BUFFER auf die Werte $00, $01, $02, $03, .. $FE, $FF:
%A=0 !Y0!S%A:%A!%A+1!%A?100 !J-0
Hier wird die Variable A sowohl als Startadresse von SET als auch als Wert, der gesetzt werden soll, benutzt.

Etwas komplizierter ist das Lesen und Ausdrucken eines gesamten Files mit Storage Type 2 – dieses Beispiel ist schon einmal im ersten Teil dieser Serie aufgeführt worden, aber vielleicht jetzt etwas verständlicher:

§/Volname/Filename
liefert den Indexblock dieses Files. Wir halten diesen Indexblock im Hold-Buffer fest:
§/Volname/Filename!H!
Als nächstes brauchen wir die einzelnen Blocks, aus denen dieser File besteht. Die Blocknummern stehen als erstes, zweites, drittes... Byte im Indexblock – leider auch als $100tes, $101tes...., denn hier stehen die höherwertigen Teile. Deshalb muß eine primitive Multiplikationsroutine aufgebaut werden.
Zuerst einmal die Zuordnung:
%A=0!%B=100 !Y0!%C(%A!%D(%B
C enthält jetzt den niederwertigen Teil, D den höherwertigen Teil der Blocknummer. Wenn D gleich Null ist, können wir die Multiplikation überspringen:
%D?0!J+2
ansonsten wird „multipliziert“, d.h. C um jeweils $100 erhöht, D am Ende der Schleife erniedrigt und geprüft. Falls D noch nicht Null ist, wird die Schleife wiederholt.
!Y1!%C+100!%D-1!J-1
Danach enthält C die Blocknummer. Wenn sie Null ist, ist dieser Block nicht belegt:
Y2!%C?0!J+3
Ansonsten wird der Block gelesen und gedumpt:
R%C!D!..
Die Indizes innerhalb des Indexblocks werden erhöht:
!Y3!%A+1!%B+1

Jetzt können wir noch einen Test auf „Indexblock-Ende“ einbauen:
%B?200!J-0
hier wird nur wieder zum Start der ersten Schleife gesprungen, wenn B noch nicht den Wert $200 erreicht hat.

Zusammengefaßt ergibt sich ohne die Returns, die hier aus drucktechnischen Gründen eingefügt wurden:

```
§/Volname/Filename
!%A=0!%B=100
!HY0!M!%C(%A!%D(%B!%D?0!J+2
!Y1!%C+100!%D-1!J-1
!Y2!%C?0!J+3!R%C!D!...
!Y3!%A+1!%B+1!%B?200!J-0
```

Nachdem wir auf diese Weise schon die nicht belegten Blocks eines Files ausgeklammert haben, könnte man diesen File natürlich in eine zusammenhängende Datei verwandeln. Hierzu wird eine weitere Variable benötigt und eine Schleife, die innerhalb des Indexblocks gefundene Blocks nicht liest, sondern mittels der „S“et-Funktion die Blocknummern zusammenschiebt.
Wenn Sie Zeit und Lust haben, können Sie mit diesem Mini-Interpreter ganze Forscher-Wochenenden verbringen.

3. ED.FUNCS2

3.1. Allgemeine Beschreibung

Dieser Programmteil enthält die oben besprochenen Funktionen und ansonsten eigentlich keine weiteren Besonderheiten.

3.2. Problemstellungen

Hier ist nur die Funktion FIND erwähnenswert: Bedingt durch die Möglichkeit „F=“ braucht FIND einen eigenen Puffer, denn es kann nicht direkt mit dem Inhalt von CMDSTR verglichen werden. Darüber hinaus kann theoretisch jedem FIND-Byte eine AND-Maske zugeordnet werden. Um den Ablauf innerhalb von FIND programmiertechnisch so einfach wie möglich zu gestalten, wird ein zweiter Buffer gleicher Länge zugeordnet, der die AND-Masken enthält. Wenn diese Maskenbytes nicht spezifiziert sind, erhalten sie den Wert $FF.
Genauso wie bei der Funktion SET ergibt sich das Problem des „Delimiters“, für den das Aufrufezeichen („!“) verwendet wird: Wenn ein solches Zeichen mit „F“ind ASCII gefunden werden soll, muß es (analog zur SET-Funktion) als „!!“ geschrieben werden.
Damit ergibt sich ein ungelöstes Problem: Wenn die Kommandozeile eine „J“ump-Anweisung zum Label Yx enthält und gleichzeitig innerhalb derselben Kommandozeile vor dem Ziel-Label ein „F“ind ASCII definiert ist, das die Zeichenfolge „...!!Yx..“ enthält, wird die Routine innerhalb von „J“ump, die den CMDSTR nach der Zeichenfolge „!Yx“ absucht, das „!Yx“ innerhalb des FIND-Strings als Sprungziel interpretieren. Dieses Problem wäre nur durch eine komplette Analyse der Kommandozeile innerhalb der Such-Funktion von „J“ump zu lösen, die die Bedingung „innerhalb eines FIND-Strings“ erkennt und sich dementsprechend verhält.

3.3. Ausgewählte Routinen

PRINTER SLOT/DUMP (Z. 6-66): Wenn das nächste Zeichen nach dem Kommando „P“ ein „S“ ist, wird das übernächste Zeichen als Slotnummer ausgewertet und die Speicherstelle PRSLOT entsprechend gesetzt (Z. 11-23).
Bei „PD“, also „Printer Dump“, wird das Flag IS40 (40-Zeichendarstellung) zwischengespeichert und danach zurückgesetzt, d.h. der Dump erfolgt immer auf 80 Zeichen Breite. Die OUT-Vektoren ($0036-$0037) werden ebenfalls zwischengespeichert und danach durch die entsprechende Slotadresse des Drucker-Interfaces ersetzt (Z. 36-39). Wenn innerhalb von EDIT ein PINIT-String für das Drucker-Interface definiert wurde (z.B. Ctrl-I80N), wird dieser String dann zum Drucker geschickt.
Nach dem Setzen des PRINTER-Flags wird PRINFO aufgerufen, das „Source ...Block..“ aufgrund des gesetzten PRINTER-Flags ohne Leerzeichen druckt. BPLINE (Z. 49) ist die Anzahl der Bytes pro Zeile im momentanen DUMP-Modus ohne eventuelle Halbierung für 40-Z/Z-Bildschirme. Der folgende Aufruf von SETDUMP berechnet die CRMASK und andere Parameter für DUMP neu, bevor von DSPSTART bis DSPEND im gesetzten Modus gedumpt wird. Je nachdem, ob im letzten DUMP-Kommando DSPEND explizit gesetzt war oder nicht, geht dieser DUMP bis zur angegebenen Endadresse oder bis zum Ende von BUFFER.
Ab Zeile 50 wird es etwas komplizierter: Nach dem Zurücksetzen des PRINTER-Flags und dem Zurückspeichern der alten OUT-Vektoren muß geprüft werden, ob der Bildschirmaufbau von EDIT diese Aktion unbeschadet überstanden hat. Bei 80 Z/Z ist dies immer der Fall. Befinden wir uns dagegen in 40 Z/Z und ist kein PINIT-String definiert, dann haben die meisten Drucker-Interfaces den DUMP auf den Bildschirm reflektiert, und deshalb wird der Bildschirm über REPRINT (in ED.FUNCS1 bei „MONITOR“) neu gedruckt.

DSPON/DSPOFF (Z. 73-95) setzen das DUMP-Display on bzw. off. DSPON setzt nur das Flag NODSP zurück und benutzt einen Teil von TITLE (in ED.FRAME), um die von DSPOFF gedruckte Meldung „NO DISPLAY“ wieder aus dem Bildschirm zu entfernen. DSPOFF löscht das gesamte DUMP-Display, setzt das Flag NODSP und schreibt „NO DISPLAY“ über das Display-Feld.

FIND (Z. 103-226) prüft zuerst über EVALHEX, ob eine Startadresse angegeben wurde. Wenn ja, wird die FIND-Startadresse FNADDR entsprechend gesetzt. Danach wird geprüft, ob das folgende Zeichen entweder ein „:“ (FIND Hex), ein „=“ (Wiederholung), ein „'“ (ASCII, MSB gelöscht) oder ein Anführungszeichen (“) (ASCII, MSB gesetzt) ist, und dann entsprechend verzweigt.
FIND Hex (Z. 118-137) holt über GETHEX die einzelnen Hex-Bytes (oder Variablen!) aus CMDSTR in FNBUF. Wenn GETHEX mit „keine Hexzahl“ returnt, wird geprüft, ob dieser Fehler durch ein vorangestelltes „M“ (Maskenbyte) hervorgerufen wurde, und gegebenenfalls das folgende Maskenbyte ausgewertet.

FNSAME (Z. 136-140) prüft CMDSTR auf „=“ und springt direkt zu FNSET. Die FIND-Zeichenfolge sollte bei einem vorherigen FIND bereits gesetzt worden sein.

FNASC (Z. 142-163) unterscheidet zwischen (') und (“) und setzt eine interne AND-Maske (ASCMASK), bevor der String aus CMDSTR nach FNBUF kopiert wird. Die dazugehörigen Maskenbytes in FNMASK werden sämtlich auf $FF gesetzt.
Der folgende Loop zum Absuchen des BUFFER-Inhalts wird von allen vier FIND-Arten benutzt und holt über GNBYTE (aus den DUMP-Funktionen in ED.FUNCS1) jeweils das nächste Byte. Da sich GNBYTE auf NXTADDR bezieht und mit DSPEND endet, muß vorher DSPEND zwischengespeichert und auf MAXBUF gesetzt werden.
Wenn das erste aus BUFFER geholte Byte nach einem AND mit dem ersten FNMASK-Byte mit dem FNBUF-Byte übereinstimmt, wird solange verglichen, bis entweder FNTOP (FIND-Stringende) erreicht ist oder die Bytefolgen nicht mehr übereinstimmen. Wenn die Bytefolge in diesem Durchlauf nicht gefunden wurde, wird die Startadresse (FNADDR) um eins erhöht. Ist danach MAXBUF erreicht, wird SUCCEED auf „-“ gesetzt und FIND endet mit dem Restore des ursprünglichen DSPEND-Wertes.

YPS (Z. 234-243) ist lediglich ein Syntax-Check und bewegt den CMDIDX über die folgende Labelnummer hinaus.

JUMP (Z. 250-300) überprüft zuerst, ob das nächste Zeichen ein „+“ oder „-“ ist, ansonsten erfolgt der Sprung unbedingt, und das interne Flag JMPCOND wird auf den Wert 00 gesetzt. Danach wird die Nummer des Ziel-Labels ausgewertet. Wenn der Wert von SUCCEED mit JMPCOND übereinstimmt oder JMPCOND auf 00 gesetzt ist, wird danach der gesamte CMDSTR nach dem Ziel-Label abgesucht. Dieses Absuchen hat das Problem, daß ein „!!Yx“ innerhalb eines SET- oder FIND-Strings falsch behandelt wird, wenn es vor dem gesuchten Ziel-Label in CMDSTR auftaucht. Die Analyse des allerersten Zeichens in CMDSTR erfolgt durch einen zweiten Einsprung in die Schleife SCANJMP. Wenn ein Jump-Label den CMDSTR einleitet, ist ein vorangehendes „!“ natürlich nicht nötig.

SETVARS (Z. 308-397) benötigt als erstes die Adresse der Variablen: VARIDX wertet das nächste Zeichen („A“..„Z“) in CMDSTR aus und berechnet daraus einen Index im Bereich 0-51 (dezimal), der dann auf die entsprechende Variable innerhalb des Bereichs VARS zeigt. Danach wird das nächste Zeichen in CMDSTR auf „+“, „-“, „?“ oder „(“ geprüft und entsprechend verzweigt.
Alle Auswertungen der „rechten“ Gleichungsseite benutzen wiederum die Routine EVALHEX, die zu GETVAR springt, wenn anstelle einer Konstanten eine Variable gefunden wurde. Je nach Zuweisungsart wird danach verglichen, addiert und subtrahiert und SUCCEED entsprechend gesetzt.

GETVAR (Z. 399-405) wird von EVALHEX her angesprungen, wenn anstelle einer Konstanten das Zeichen „%“ gefunden wurde. Über VARIDX wird die Adresse der Variablen berechnet, und der Wert der Variablen wird EVREG (und EVREG+1) „untergeschoben“. Auf diese Weise ist das Einsetzen von Variablen anstelle von Konstanten für jedes Programm, das EVALHEX aufruft, vollkommen transparent, denn der Wert befindet sich nachher immer in EVREG.

VARIDX (Z. 407-420) berechnet den Index zu einer Variablen durch einfache Subtraktion des ASCII-Wertes von „A“ und anschließender Multiplikation mit 2. Dadurch ergibt sich für „A“ der Index 0 im Speicherbereich VARS, für „B“ der Index 2 usw.

SLEEP (Z. 429-446) ist eine Verzögerungsroutine, deren Zeitbedarf durch das Argument nach „Z“ bestimmt wird. „Z00“ wartet etwa 1/16 Sekunde, „Z10“ wartet 1/1 Sekunde usw. Die Maximalverzögerung von SLEEP beträgt rund 16 Sekunden und wird mit „ZFF“ gesetzt. Wenn das Argument größer als $100 ist, wird nur der niederwertige Teil berücksichtigt, „ZFFFF“ dauert also genauso lange wie „ZFF“. Man hätte hier noch eine Tastaturabfrage einbauen können, um diese Routine bei Bedarf abzubrechen.

HELP (Z. 454-475) löscht das Display-Feld und versucht, einen „HLPTXT“ auszudrucken. Dieser Text besteht allerdings im Moment nur aus einem „Ende“-Zeichen (00) in ED.CMDS, und deshalb taucht HELP auch überhaupt nicht in der Kommandobeschreibung am Anfang dieses Artikels auf.

4. ED.CMDS

Hier finden Sie die neuen Kommandos und ihre Startadressen. Die Definition der Startadressen von FNBUF und FNMASK hat jetzt ihren Sinn bekommen. Die Funktion HELP, aufgerufen durch ein „?“, ist hier noch nicht definiert, auch wenn sie bereits in ED.FUNCS2 enthalten ist. Damit sich der Assembler nicht beschwert, enthält ED.CMDS noch das Label HLPTXT mit einer folgenden „Text-Ende“-Markierung.
In einem guten Programm sollte zumindest eine minimale Hilfestellung nicht fehlen. Andererseits benötigen Hilfemenüs u.U. sehr viel Speicherplatz. Ein schlechtes Beispiel ist der FILER, denn hier ist der Speicher mit HELP-Texten so vollgestopft, daß beim Kopieren einer Diskette nur noch 4 Tracks auf einmal gelesen werden können gegenüber 8 Tracks beim guten alten COPYA.
Wenn Sie auf HELP ganz verzichten wollen, sollten die Zeilen aus ED.CMDS gestrichen werden, in denen der Help-Befehl momentan definiert wird (Z. 66-67).
Das Programm EDIT ist damit komplett.

Fast-Writer im Kit(t)chen?

Mit der Bestellkarte auf Seite 27 kann man jetzt einen Kitchen Fast-Writer ordern ... Oder doch nicht?
Harald Grumser ist natürlich weder im Kittchen, noch befaßt sich sein Programm mit der Küche (“kitchen”). Woher kommt der „Dreckfuhler“? Ganz einfach: Ich gab telefonisch an die Druckerei durch, daß auf der Bestellkarte noch eine Zeile einzufügen sei: Ein Kästchen setzen und dann Fast-Writer, DOS 3.3, DM 128,–. Und was aus dem Kästchen geworden ist, sehen Sie jetzt auf Seite 27.
Wie heißt es doch so schön bei Kunfuzius: Fordere viel von dir selbst und erwarte wenig von anderen, dann wird dir Ärger erspart bleiben. us

ED.FUNCS.TXT

```
1775:             1         PAGE
1775:             2 *********************
1775:             3 *-PRINTER_SLOT/DUMP-*
1775:             4 *********************
1775:             5 *
1775:AE 9E ØC     6 SETPRT: LDX CMDIDX
1778:BD ØØ Ø2     7         LDA CMDSTR,X
177B:2Ø DF ØA     8         JSR SETUC
177E:C9 C4        9         CMP #'D'        ;Printer DUMP?
178Ø:FØ 1D       1Ø         BEQ PDUMP
1782:C9 D3       11         CMP #'S'        ;Printer Slot?
1784:DØ 16       12         BNE PDERR
1786:E8          13         INX
1787:BD ØØ Ø2    14         LDA CMDSTR,X    ;Printer Slot: Slotno
178A:69 ØF       15         ADC #$ØF        ;+ $1Ø: $Bx => $Cx
178C:C9 C1       16         CMP #$C1        ;Slot 1
178E:9Ø ØC       17         BCC PDERR
179Ø:C9 C8       18         CMP #$C8        ;> Slot 7?
1792:BØ Ø8       19         BCS PDERR
1794:8D EE 17    2Ø         STA PRSLOT      ;Drucker: Slotnummer
1797:E8          21         INX
1798:8E 9E ØC    22         STX CMDIDX
179B:6Ø          23         RTS
179C:            24 *
179C:4C AA ØA    25 PDERR: JMP CMDERR
179F:            26 *
179F:E8          27 PDUMP: INX
17AØ:8E 9E ØC    28         STX CMDIDX      ;Command 'abgehakt'
17A3:AD 9A ØC    29         LDA IS4Ø
17A6:48          3Ø         PHA             ;PRINTER immer mit
17A7:4E 9A ØC    31         LSR IS4Ø        ;"8ØZ"
17AA:A5 36       32         LDA COUTV       ;der Vektor
17AC:48          33         PHA             ;'Character out'
17AD:A5 37       34         LDA COUTV+1     ;wird gespeichert
17AF:48          35         PHA
17BØ:AD EE 17    36 PRSTART: LDA PRSLOT     ;durch "PDx" gesetzt
17B3:85 37       37         STA COUTV+1
17B5:A2 ØØ       38         LDX #ØØ
17B7:86 36       39         STX COUTV       ;=> $CxØØ
17B9:86 24       4Ø         STX CH          ;HTAB ØØ für Printer
17BB:BD EF 17    41 PRINIT: LDA PINIT,X     ;Initstring, z.B:
17BE:FØ Ø6       42         BEQ PDMP2       ;Ctrl-I 8ØN
17CØ:2Ø ED FD    43         JSR COUT        ;<ØØ> = Ende
17C3:E8          44         INX
17C4:DØ F5       45         BNE PRINIT
17C6:A9 FF       46 PDMP2: LDA #$FF        ;PRINTER-Flag wird
17C8:8D 9B ØC    47         STA PRINTER     ;gesetzt
17CB:2Ø 75 1Ø    48         JSR PRINFO      ;"Source... Blockno"
17CE:AD AE ØC    49         LDA BPLINE      ;Bytes per Line
17D1:2Ø 7F 13    5Ø         JSR SETDUMP     ;=> DUMP
17D4:A9 ØØ       51         LDA #ØØ         ;PRINTER-Flag zurück
17D6:8D 9B ØC    52         STA PRINTER
17D9:68          53         PLA
17DA:85 37       54         STA COUTV+1     ;Restore der 0-Vecs
17DC:68          55         PLA
17DD:85 36       56         STA COUTV
17DF:68          57         PLA
17EØ:8D 9A ØC    58         STA IS4Ø        ;Restore von IS4Ø
17E3:1Ø Ø8       59         BPL PRDONE      ;8Ø Zeichen, Schirm ok
17E5:AD EF 17    6Ø         LDA PINIT       ;Init-String
17E8:DØ Ø3       61         BNE PRDONE      ;definiert
17EA:2Ø 3A 11    62         JSR REPRINT     ;4ØZ: Schirm zerstört
17ED:6Ø          63 PRDONE: RTS
17EE:            64 *
17EE:Cl          65 PRSLOT: DFB $C1        ;Printer-Slot 1
17EF:ØØ ØØ ØØ    66 PINIT: DFB Ø,Ø,Ø,Ø,Ø,Ø ;Init für Printer
17F5:            67 *
17F5:            69 ******************
17F5:            7Ø *---DSPON/OFF---*
17F5:            71 ******************
17F5:            72 *
17F5:A9 ØØ       73 DSPON: LDA #$ØØ
17F7:8D 9C ØC    74         STA NODSP       ;Flag OFF
17FA:A9 Ø4       75         LDA #4
17FC:4C DE Ø8    76         JMP TTL3        ;"-------" auf VTAB 4
17FF:            77 *
17FF:2Ø 63 ØB    78 DSPOFF: JSR CLRDSP     ;Clear Display
18Ø2:A9 Ø4       79         LDA #4
18Ø4:2Ø 55 ØB    8Ø         JSR SETVTB      ;VTAB 4
18Ø7:AD 97 ØC    81         LDA HMAX        ;Zeilenlänge: 4Ø/8Ø
18ØA:38          82         SEC
18ØB:6E 9C ØC    83         ROR NODSP       ;Flag ON
18ØE:38          84         SEC
18ØF:E9 ØA       85         SBC #1Ø
```

```
1811:4A          86         LSR A           ;Mittenzentrierung
1812:2Ø 89 ØB    87         JSR SETCH
1815:A2 ØØ       88         LDX #Ø
1817:BD 23 18    89 SAYOFF: LDA OFFMSG,X
181A:2Ø ED FD    9Ø         JSR COUT
181D:E8          91         INX
181E:E0 ØA       92         CPX #1Ø
1820:9Ø F5       93         BCC SAYOFF
1822:6Ø          94         RTS
1823:CE CF AØ    95 OFFMSG: ASC 'NO      DISPLAY'
182D:            96 *
182D:            99 ************
182D:           1ØØ *---FIND---*
182D:           1Ø1 ************
182D:           1Ø2 *
182D:AE 9E ØC   1Ø3 FIND:  LDX CMDIDX
1830:2Ø 36 ØC   1Ø4        JSR EVALHEX     ;folgt Startadresse?
1833:BØ ØE      1Ø5        BCS FINDZ       ;nein
1835:2Ø 2D ØC   1Ø6        JSR TSTARG      ;Start > MAXBUF?
1838:9Ø Ø3      1Ø7        BCC SFSTART     ;in EVREG/EVREG+1
183A:4C BF 18   1Ø8        JMP FNERR       ;Startadresse > $1FF
183D:8D C5 ØC   1Ø9 SFSTART: STA FNADDR+1 ;spezifizierter Start
1840:8C C4 ØC   11Ø        STY FNADDR      ;=> FIND_ADDRESS
1843:AØ ØØ      111 FINDZ: LDY #ØØ
1845:AE 9E ØC   112        LDX CMDIDX
1848:BD ØØ Ø2   113        LDA CMDSTR,X
184B:C9 BA      114        CMP #':'        ;Find HEX?
184D:DØ 2E      115        BNE FNSAME      ;Wiederholung/ASCII?
184F:8C 3D 19   116        STY FNTOP       ;Obergrenze in FNBUF
1852:E8         117        INX             ;und FNMASK
1853:A9 FF      118 FNHEX1: LDA #$FF       ;Default: Maske $FF
1855:AC 3D 19   119 FNHEX2: LDY FNTOP
1858:99 ØØ 25   12Ø        STA FNMASK,Y
185B:2Ø DA 15   121        JSR GETHEX      ;holt ein Hex-Byte
185E:AC 3D 19   122        LDY FNTOP       ;aus CMDSTR
1861:BØ Ø8      123        BCS MPARM       ;kein Hex-Byte, "M"?
1863:99 ØØ 24   124        STA FNBUF,Y
1866:EE 3D 19   125        INC FNTOP
1869:DØ E8      126        BNE FNHEX1
186B:BD ØØ Ø2   127 MPARM?: LDA CMDSTR,X
186E:2Ø DF ØA   128        JSR SETUC
1871:C9 CD      129        CMP #'M'        ;"M"ask?
1873:DØ 44      13Ø        BNE FNSET       ;nein, Hex zuende
1875:E8         131        INX
1876:2Ø DA 15   132        JSR GETHEX      ;nach "M" folgt Maske
1879:9Ø DA      133        BCC FNHEX2      ;Maske einsetzen
187B:BØ 42      134        BCS FNERR       ;"Command Error"
187D:           135 *
187D:C9 BD      136 FNSAME: CMP #'='       ;Wiederholung?
187F:DØ Ø7      137        BNE FNASC
1881:E8         138        INX
1882:AC 3D 19   139        LDY FNTOP       ;alter Index
1885:4C B9 18   14Ø        JMP FNSET
1888:           141 *
1888:A9 FF      142 FNASC: LDA #$FF        ;default:
188A:8D 3E 19   143        STA ASCMASK     ;ASCII-Maske $FF
188D:BD ØØ Ø2   144        LDA CMDSTR,X
189Ø:C9 A2      145        CMP #'"'        ;ASCII, MSB set?
1892:FØ Ø7      146        BEQ FNASC1
1894:C9 A7      147        CMP #$A7        ;"'"?
1896:DØ 27      148        BNE FNERR       ;"Command Error"
1898:4E 3E 19   149        LSR ASCMASK     ;=> $7F: MSB off
189B:E8         15Ø FNASC1: INX
189C:BD ØØ Ø2   151        LDA CMDSTR,X
189F:FØ 18      152        BEQ FNSET       ;CMDSTR zuende
18A1:C9 A1      153        CMP #STOP       ;ein "!"?
18A3:DØ Ø6      154        BNE FNASC2      ;nein, ist ok
18A5:DD Ø1 Ø2   155        CMP CMDSTR+1,X ;noch ein "!"?
18A8:DØ ØF      156        BNE FNSET       ;nein, Ende
18AA:E8         157        INX             ;"!!" => "!"
18AB:2D 3E 19   158 FNASC2: AND ASCMASK    ;für "'": MSB off
18AE:99 ØØ 24   159        STA FNBUF,Y
18B1:A9 FF      16Ø        LDA #$FF
18B3:99 ØØ 25   161        STA FNMASK,Y    ;keine AND-Maske
18B6:C8         162        INY             ;für ASCII
18B7:DØ E2      163        BNE FNASC1      ;"always"
18B9:           164 *
18B9:8E 9E ØC   165 FNSET: STX CMDIDX      ;Command 'abgehakt'
18BC:98         166        TYA
18BD:DØ Ø4      167        BNE FNSET2      ;
18BF:CA         168 FNERR: DEX             ;FIND-String ist leer:
18CØ:4C AA ØA   169        JMP CMDERR      ;"..Parameter Error"
18C3:           17Ø *
18C3:8C 3D 19   171 FNSET2: STY FNTOP      ;Obergrenze in FNBUF
18C6:2Ø 4E 12   172        JSR SAVEND      ;GNBYTE geht von
18C9:A9 Ø2      173        LDA #MAXBUF     ;NXTADDR..DSPEND:
18CB:8D B7 ØC   174        STA DSPEND+1    ;DSPEND wird auf
```

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 5,50 zuzügl. ges. MwSt. Mindestens 2 Druckzeilen
Beispiel:

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

nur DM 18,81 inkl. MwSt.

Musteranzeige gewerblich

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

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

nur DM 62,70 inkl. MwSt.

PEEKER

Peeker-Börse

AUFTRAG FÜR KLEINANZEIGEN

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

☐ suche Hardware ☐ suche Software ☐ Verschiedenes
☐ biete Hardware ☐ biete Software ☐ Chiffre

Bitte jeweils 32 Buchstaben pro Zeile – einschließlich Satzzeichen und Wortzwischenräume. Bitte Absender nicht vergessen. Mindestens 2 Zeilen. Chiffregebühr DM 6,– zuzügl. MwSt.

PEEKER

Produkt-Karte

Zu der im **Peeker**, Heft _____, Seite _____ erschienenen

Anzeige über _____

_____

bitte ich um detaillierte Information.

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

Menge	Produkt und Bestellnummer	à DM	gesamt DM

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

_____
Unterschrift (für Jugendliche unter 18 Jahren der Erziehungsberechtigte)

PEEKER

Umfrage-Karte

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

Ich besitze einen

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

Ich interessiere mich für den

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

Ich bin

☐ mehr Anwender
☐ mehr Programmierer

Peeker-Börse

Vorname, Name

Firma

Straße

Wohnort

PLZ/Ort

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

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

Datum Unterschrift

Produkt-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

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

Umfrage-Karte

Karte bitte vollständig ausfüllen

Vorname, Name

Firma

Straße

PLZ/Ort

Telefon mit Vorwahl

POSTKARTE

Peeker-Börse
Anzeigen-Service

Dr. Alfred Hüthig Verlag

Postfach 10 28 69

6900 Heidelberg 1

POSTKARTE

Inserent

Straße

PLZ/Ort

POSTKARTE

Peeker
Redaktion

Dr. Alfred Hüthig Verlag

Postfach 10 28 69

6900 Heidelberg 1

Produkt-Karte

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

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

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 und Ihren Absender ein.

PEEKER

```
18CE:A9 ØØ     175         LDA #ØØ       ;MAXBUF gesetzt
18DØ:8D B6 ØC  176         STA DSPEND
18D3:          177 *
18D3:AD C5 ØC  178 FNEXT:  LDA FNADDR+1  ;Start nächster
18D6:AC C4 ØC  179         LDY FNADDR    ;Vergleich
18D9:8D B5 ØC  18Ø         STA NXTADDR+1 ;für GNBYTE: Start
18DC:8C B4 ØC  181         STY NXTADDR
18DF:A2 ØØ     182         LDX #ØØ
18E1:2Ø CC 14  183 FLOOP:  JSR GNBYTE    ;ein Byte aus BUFFER
18E4:3D ØØ 25  184         AND FNMASK,X  ;AND MASK
18E7:DD ØØ 24  185         CMP FNBUF,X   ;CMP FIND_BUFFER
18EA:DØ ØD     186         BNE NXTBASE
18EC:E8        187         INX           ;bis hierher gleich
18ED:EC 3D 19  188         CPX FNTOP     ;FNBUF-Ende?
18FØ:FØ 1F     189         BEQ FOUND     ;ok - Pattern gefunden
18F2:2C ØØ 15  19Ø         BIT DDONE     ;BUFFER-Ende erreicht?
18F5:3Ø 16     191         BMI FNOT      ;ja, nicht gefunden
18F7:1Ø E8     192         BPL FLOOP     ;nächstes Byte
18F9:          193 *
18F9:EE C4 ØC  194 NXTBASE: INC FNADDR   ;Vergleichsadresse+1
18FC:DØ D5     195         BNE FNEXT
18FE:EE C5 ØC  196         INC FNADDR+1
19Ø1:AD C5 ØC  197         LDA FNADDR+1
19Ø4:C9 Ø2     198         CMP #MAXBUF   ;BUFFER-Ende?
19Ø6:9Ø CB     199         BCC FNEXT     ;nein
19Ø8:A9 ØØ     2ØØ         LDA #ØØ       ;FIND-Ende: Start
19ØA:8D C5 ØC  2Ø1         STA FNADDR+1  ;zurück auf ØØØØ
19ØD:          2Ø2 *
19ØD:A9 AD     2Ø3 FNOT:   LDA #'-'
19ØF:DØ 25     2Ø4         BNE FNDONE    ;"always"
1911:          2Ø5 *
1911:AD C5 ØC  2Ø6 FOUND:  LDA FNADDR+1  ;FNADDR=> DSPSTART
1914:AC C4 ØC  2Ø7         LDY FNADDR    ;und DUMP ab gefun-
1917:8D B3 ØC  2Ø8         STA DSPSTART+1 ;denem Pattern. Gibt
191A:8C B2 ØC  2Ø9         STY DSPSTART  ;falsche Anzeige,
191D:EE C4 ØC  21Ø         INC FNADDR    ;falls CMODE!
192Ø:DØ ØC     211         BNE FOUNDZ    ;für ein evtl.
1922:EE C5 ØC  212         INC FNADDR+1  ;nächstes "F=" wird
1925:C9 Ø1     213         CMP #MAXBUF-1 ;FNADDR um eins
1927:9Ø Ø5     214         BCC FOUNDZ    ;erhöht
1929:A9 ØØ     215         LDA #Ø
192B:8D C5 ØC  216         STA FNADDR+1
192E:2Ø 63 ØB  217 FOUNDZ: JSR CLRDSP    ;Display clear
1931:2Ø A2 13  218         JSR DUMP      ;DUMP ganze Seite
1934:A9 AB     219         LDA #'+'      ;FIND erfolgreich
1936:          22Ø *
1936:8D BA ØC  221 FNDONE: STA SUCCEED   ;"+" oder "-"
1939:2Ø 61 12  222         JSR RESEND    ;RESTORE DSPEND
193C:6Ø        223         RTS
193D:          224 *
193D:ØØ        225 FNTOP:  DFB ØØ        ;FNBUF/MASK-Obergrenze
193E:          226 ASCMASK: DS 1         ;Scratch
193F:          227 *
193F:          23Ø ***********
193F:          231 *---YPS---*
193F:          232 ***********
193F:          233 *
193F:AE 9E ØC  234 YPS:    LDX CMDIDX    ;"!Yx" = Label x
1942:BD ØØ Ø2  235         LDA CMDSTR,X  ;das Command selber
1945:2Ø 7Ø ØC  236         JSR SETHEX    ;ist "leer", d.h. es
1948:BØ Ø9     237         BCS YPSERR    ;findet nur ein
194A:C9 ØA     238         CMP #1Ø       ;Syntax-Check statt.
194C:BØ Ø5     239         BCS YPSERR    ;Labels von "Ø"-"9"
194E:E8        24Ø         INX
194F:8E 9E ØC  241         STX CMDIDX    ;Command 'abgehakt'
1952:6Ø        242         RTS
1953:4C AA ØA  243 YPSERR: JMP CMDERR    ;keine Zahl nach "!Y"
1956:          244 *
1956:          246 ************
1956:          247 *---JUMP---*
1956:          248 ************
1956:          249 *
1956:A9 ØØ     25Ø JUMP:   LDA #ØØ       ;Annahme: JMP always
1958:8D BE 19  251         STA JMPCOND   ;(unkonditional)
195B:AE 9E ØC  252         LDX CMDIDX
195E:BD ØØ Ø2  253         LDA CMDSTR,X  ;folgt Bedingung?
1961:C9 AB     254         CMP #'+'
1963:FØ Ø4     255         BEQ SETCOND
1965:C9 AD     256         CMP #'-'
1967:DØ Ø7     257         BNE JMPNUM    ;nein
1969:8D BE 19  258 SETCOND: STA JMPCOND
196C:E8        259         INX
196D:BD ØØ Ø2  26Ø         LDA CMDSTR,X
197Ø:2Ø 7Ø ØC  261 JMPNUM: JSR SETHEX    ;nach J<+/-> muß die
1973:BØ 36     262         BCS JMPERR    ;Labelnummer folgen
1975:8D BF 19  263         STA JMPARG    ;Labelnummer
1978:E8        264         INX
```

```
1979:8E 9E ØC  265         STX CMDIDX    ;Command 'abgehakt'
197C:A Ø FF    266         LDY #$FF      ;Scan-Index in CMDSTR
197E:AD BE 19  267         LDA JMPCOND
1981:FØ ØE     268         BEQ SCANØØ    ;unkonditional
1983:CD BA ØC  269         CMP SUCCEED   ;"+" oder "-"
1986:DØ 22     27Ø         BNE JMPDONE   ;nicht erfüllt
1988:FØ Ø7     271         BEQ SCANØØ    ;Test 1.Zeichen CMDSTR
198A:          272 *
198A:2Ø AE 19  273 SCANJMP: JSR NXTJMP   ;Absuchen von CMDSTR
198D:C9 A1     274         CMP #STOP     ;nach dem Ziel-Label
198F:DØ F9     275         BNE SCANJMP   ;nächstes Zeichen
1991:2Ø AE 19  276 SCANØØ: JSR NXTJMP    ;Test 1.Zeichen CMDSTR
1994:2Ø DF ØA  277         JSR SETUC     ;"Y" nach "!"?
1997:C9 D9     278         CMP #'Y'
1999:DØ EF     279         BNE SCANJMP
199B:2Ø AE 19  28Ø         JSR NXTJMP
199E:2Ø 7Ø ØC  281         JSR SETHEX    ;Labelnummer
19A1:CD BF 19  282         CMP JMPARG    ;gesuchtes Label?
19A4:DØ E4     283         BNE SCANJMP
19A6:C8        284         INY           ;"hinter" Labelnummer
19A7:8C 9E ØC  285         STY CMDIDX    ;weiter ab <LABEL>
19AA:6Ø        286 JMPDONE: RTS
19AB:          287 *
19AB:4C AA ØA  288 JMPERR: JMP CMDERR    ;"Command Error"
19AE:          289 *
19AE:C8        29Ø NXTJMP: INY           ;holt ein Zeichen aus
19AF:B9 ØØ Ø2  291         LDA CMDSTR,Y  ;CMDSTR und prüft auf
19B2:FØ Ø1     292         BEQ LBLBAD    ;CMDSTR-Ende
19B4:6Ø        293         RTS
19B5:AE 9E ØC  294 LBLBAD: LDX CMDIDX    ;für Fehleranzeige
19B8:CA        295         DEX
19B9:A Ø 9Ø    296         LDY #BADLBL-ERRMSGS
19BB:4C AC ØA  297         JMP DSPERR    ;"JMP-Label not found"
19BE:          298 *
19BE:          299 JMPCOND: DS 1         ;JUMP-Bedingung:+/-/Ø
19BF:          3ØØ JMPARG: DS 1          ;Ziel-Labelnummer
19CØ:          3Ø1 *
19CØ:          3Ø4 ***************
19CØ:          3Ø5 *---SETVARS---*
19CØ:          3Ø6 ***************
19CØ:          3Ø7 *
19CØ:AE 9E ØC  3Ø8 SETVARS: LDX CMDIDX   ;nach "%" muß
19C3:2Ø 99 1A  3Ø9         JSR VARIDX    ;"A".."Z" folgen
19C6:BØ 55     31Ø         BCS VARERR    ;nicht der Fall
19C8:8C 94 ØC  311         STY TEMP      ;Index in VARS
19CB:BD ØØ Ø2  312         LDA CMDSTR,X
19CE:C9 AB     313         CMP #'+'
19DØ:FØ 73     314         BEQ ADDVAR    ;Variable + ARG
19D2:C9 AD     315         CMP #'-'
19D4:DØ Ø3     316         BNE SVAR1
19D6:4C 57 1A  317         JMP SUBVAR    ;Variable - ARG
19D9:C9 BF     318 SVAR1:  CMP #'?'      ;Vergleich?
19DB:FØ 2Ø     319         BEQ CMPVAR
19DD:C9 A8     32Ø         CMP #'('      ;Adressen-Zuordnung?
19DF:FØ 3F     321         BEQ ADDRVR
19E1:C9 BD     322         CMP #'='      ;direkte Zuordnung?
19E3:DØ 38     323         BNE VARERR
19E5:          324 *
19E5:E8        325 LETVAR: INX           ;"LET" <Var> = xx
19E6:2Ø 36 ØC  326         JSR EVALHEX   ;Zuweisung, kann
19E9:BØ 32     327         BCS VARERR    ;wieder eine Variable
19EB:AC 94 ØC  328         LDY TEMP      ;sein!
19EE:AD A6 ØC  329         LDA EVREG
19F1:99 B1 1A  33Ø         STA VARS,Y
19F4:AD A7 ØC  331         LDA EVREG+1   ;EVREG hält das Ergeb-
19F7:99 B2 1A  332         STA VARS+1,Y  ;nis der "rechten"
19FA:4C 76 1A  333         JMP SVARQT    ;Seite der Gleichung
19FD:          334 *
19FD:E8        335 CMPVAR: INX           ;"IF <Var> = xx THEN
19FE:8E 9E ØC  336         STX CMDIDX    ;SUCCEED = '+'"
1AØ1:2Ø 36 ØC  337         JSR EVALHEX   ;holt Argument, kann
1AØ4:BØ 17     338         BCS VARERR    ;eine Variable sein!
1AØ6:AC 94 ØC  339         LDY TEMP      ;Index in VARS
1AØ9:AD A6 ØC  34Ø         LDA EVREG
1AØC:D9 B1 1A  341         CMP VARS,Y
1AØF:DØ 6Ø     342         BNE VAREXX    ;SUCCEED auf "-"
1A11:AD A7 ØC  343         LDA EVREG+1
1A14:D9 B2 1A  344         CMP VARS+1,Y
1A17:DØ 58     345         BNE VAREXX
1A19:A Ø AB    346         LDY #'+'      ;SUCCEED auf "+"
1A1B:DØ 56     347         BNE VARZRO    ;"always"
1A1D:          348 *
1A1D:4C AA ØA  349 VARERR: JMP CMDERR    ;"Command Error"
1A2Ø:          35Ø *
1A2Ø:E8        351 ADDRVR: INX           ;"<Var> = PEEK
1A21:8E 9E ØC  352         STX CMDIDX    ;(xx+BUFFER)"
1A24:2Ø 28 ØC  353         JSR EVALARG   ;maximal MAXBUF!
```

```
1A27:B0 F4     354         BCS VARERR
1A29:AE A6 0C  355         LDX EVREG      ;Adresse kann
1A2C:AD A7 0C  356         LDA EVREG+1    ;wieder eine
1A2F:18        357         CLC            ;Variable sein!
1A30:69 20     358         ADC #<BUFFER   ;Hi-Adresse
1A32:8D 3A 1A  359         STA ADDRV1+2
1A35:AC 94 0C  360         LDY TEMP       ;Index in VARS
1A38:BD 00 20  361 ADDRV1: LDA BUFFER,X ;holt Wert
1A3B:99 B1 1A  362         STA VARS,Y
1A3E:A9 00     363         LDA #0
1A40:99 B2 1A  364         STA VARS+1,Y   ;Hibyte auf 00
1A43:F0 31     365         BEQ SVARQT     ;"always"
1A45:          366 *
1A45:20 77 1A  367 ADDVAR: JSR VARARG     ;holt ARG
1A48:B0 D3     368         BCS VARERR     ;aus CMDSTR
1A4A:6D A6 0C  369         ADC EVREG      ;und returnt mit
1A4D:9D B1 1A  370         STA VARS,X     ;Index
1A50:98        371         TYA            ;und altem Wert
1A51:6D A7 0C  372         ADC EVREG+1    ;in A-Y
1A54:4C 67 1A  373         JMP SUBADD
1A57:          374 *
1A57:20 77 1A  375 SUBVAR: JSR VARARG     ;holt ARG
1A5A:B0 C1     376         BCS VARERR     ;aus CMDSTR
1A5C:38        377         SEC            ;und returnt mit
1A5D:ED A6 0C  378         SBC EVREG      ;altem Wert in A-Y
1A60:9D B1 1A  379         STA VARS,X
1A63:98        380         TYA
1A64:ED A7 0C  381         SBC EVREG+1
1A67:A0 AB     382 SUBADD: LDY #'+'
1A69:9D B2 1A  383         STA VARS+1,X
1A6C:1D B1 1A  384         ORA VARS,X     ;Null erreicht?
1A6F:F0 02     385         BEQ VARZRO
1A71:A0 AD     386 VAREXX: LDY #'-'       ;nein, SUCCEED="-"
1A73:8C BA 0C  387 VARZRO: STY SUCCEED
1A76:          388 *
1A76:60        389 SVARQT: RTS
1A77:          390 *
1A77:E8        391 VARARG: INX            ;für "+" und "-":
1A78:20 36 0C  392         JSR EVALHEX    ;holt Argument
1A7B:B0 09     393         BCS NOTARG     ;kein Argument folgt
1A7D:AE 94 0C  394         LDX TEMP       ;Index zur Var
1A80:BD B1 1A  395         LDA VARS,X
1A83:BC B2 1A  396         LDY VARS+1,X
1A86:60        397 NOTARG: RTS            ;X-Reg: CMDIDX
1A87:          398 *
1A87:20 99 1A  399 GETVAR: JSR VARIDX     ;holt Variablen-Index
1A8A:B0 91     400         BCS VARERR     ;nicht "A"..."Z"
1A8C:B9 B1 1A  401         LDA VARS,Y
1A8F:8D A6 0C  402         STA EVREG      ;der Variablenwert
1A92:B9 B2 1A  403         LDA VARS+1,Y   ;wird EVREG unter-
1A95:8D A7 0C  404         STA EVREG+1    ;geschoben
1A98:60        405         RTS
1A99:          406 *
1A99:BD 00 02  407 VARIDX: LDA CMDSTR,X ;nach einem "%"
1A9C:20 DF 0A  408         JSR SETUC      ;Shift nach UC
1A9F:38        409         SEC
1AA0:E9 C1     410         SBC #'A'
1AA2:90 0B     411         BCC BADVAR     ;kleiner "A"
1AA4:C9 1B     412         CMP #27        ;größer "Z"?
1AA6:B0 07     413         BCS BADVAR
1AA8:0A        414         ASL A          ;16 Bit, Index * 2
1AA9:A8        415         TAY
1AAA:E8        416         INX
1AAB:8E 9E 0C  417         STX CMDIDX     ;Variable 'abgehakt'
1AAE:60        418         RTS
1AAF:38        419 BADVAR: SEC
1AB0:60        420         RTS
1AB1:          421 *
1AB1:          422 VARS:  DS  52          ;26 Variablen (16 Bit)
1AE5:          423 *
1AE5:          425 *************
1AE5:          426 *---SLEEP---*
1AE5:          427 *************
1AE5:          428 *
1AE5:AE 9E 0C  429 SLEEP: LDX CMDIDX
1AE8:20 36 0C  430         JSR EVALHEX    ;Argument für SLEEP
1AEB:B0 18     431         BCS SLPERR
1AED:AD A6 0C  432         LDA EVREG
1AF0:09 01     433         ORA #01        ;mindestens 1
1AF2:A8        434         TAY            ;maximal rund 25 Sek.
1AF3:          435 *
1AF3:A9 0F     436 SLP1:  LDA #15
1AF5:A2 EB     437 SLP2:  LDX #235
1AF7:20 04 1B  438 SLP3:  JSR SLQUIT      ;innerer Loop:
1AFA:CA        439         DEX            ;3999 uSec pro
1AFB:D0 FA     440         BNE SLP3       ;Durchlauf
1AFD:E9 01     441         SBC #1
1AFF:D0 F4     442         BNE SLP2
1B01:88        443         DEY
1B02:D0 EF     444         BNE SLP1
1B04:60        445 SLQUIT: RTS
1B05:4C AA 0A  446 SLPERR: JMP CMDERR     ;"Command Error"
1B08:          447 *
1B08:          450 *************
1B08:          451 *---HELP---*
1B08:          452 *************
1B08:          453 *
1B08:20 42 12  454 HELP:  JSR NOMODE      ;alle Modes OFF
1B0B:20 63 0B  455         JSR CLRDSP     ;Display clear
1B0E:A9 1B     456         LDA #<HLPTXT   ;höherwertiger Teil
1B10:A0 81     457         LDY #>HLPTXT   ;Low-Byte
1B12:8D 1C 1B  458         STA GETHLP+2
1B15:8C 1B 1B  459         STY GETHLP+1
1B18:          460 *
1B18:A2 00     461         LDX #0
1B1A:BD 81 1B  462 GETHLP: LDA HLPTXT,X
1B1D:F0 16     463         BEQ HLPGOT     ;Text-Ende
1B1F:C9 0D     464         CMP #$0D       ;fakultatives <CR>?
1B21:D0 07     465         BNE PRTHLP
1B23:2C 9A 0C  466         BIT IS40       ;40 Zeichen?
1B26:30 05     467         BMI HSKIP      ;ja, nicht drucken
1B28:09 80     468         ORA #$80       ;=> $8D
1B2A:20 E6 0A  469 PRTHLP: JSR PRCHAR
1B2D:E8        470 HSKIP: INX
1B2E:D0 EA     471         BNE GETHLP
1B30:EE 1C 1B  472         INC GETHLP+2   ;nächste $100 Bytes
1B33:D0 E5     473         BNE GETHLP     ;"always"
1B35:          474 *
1B35:60        475 HLPGOT: RTS
1B36:          476 *
1B36:          478         CHN ED.CMDS.TXT
```

ED.CMDS.TXT

```
1B36:           1         PAGE
1B36:           2 ****************
1B36:           3 *---COMMANDS---*
1B36:           4 ****************
1B36:           5 *
1B36:           6 CMDS:   EQU *
1B36:C0         7         ASC '§'        ;Lädt den ersten Block
1B37:28 0F      8         DW  ATFILE     ;eines Files
1B39:D8         9         ASC 'X'        ;MONITOR-Aufruf
1B3A:F6 10     10         DW  XMON
1B3C:D1        11         ASC 'Q'        ;Programm-Ende
1B3D:4F 11     12         DW  QUIT
1B3F:D2        13         ASC 'R'        ;Read Block/Prefix
1B40:          14 *
1B40:9B 0D     15         DW  READ
1B42:D7        16         ASC 'W'        ;Write Block/Prefix
1B43:A3 0D     17         DW  WRITE
1B45:          18 *
1B45:D3        19         ASC 'S'
1B46:3A 15     20         DW  SET        ;Bytes verändern
1B48:          21 *
1B48:C1        22         ASC 'A'
1B49:36 13     23         DW  DUMPA      ;ASCII-Dump
1B4B:C4        24         ASC 'D'
1B4C:3C 13     25         DW  DUMPH      ;Hex/ASCII-Dump
1B4E:D6        26         ASC 'V'
1B4F:42 13     27         DW  DUMPV      ;VBM-Dump
1B51:          28 *
1B51:BC        29         ASC '<'
1B52:58 11     30         DW  HALFUP     ;halbe Seite aufwärts
1B54:AC        31         ASC ','
1B55:63 11     32         DW  LINEUP     ;Zeile aufwärts
1B57:BE        33         ASC '>'
1B58:94 11     34         DW  HALFDN     ;halbe Seite abwärts
1B5A:AE        35         ASC '.'
1B5B:9F 11     36         DW  LINEDN     ;Zeile abwärts
1B5D:          37 *
1B5D:C8        38         ASC 'H'
1B5E:B2 16     39         DW  HOLD       ;HLBUF setzen
1B60:CD        40         ASC 'M'
1B61:29 17     41         DW  MOVE       ;aus HLBUF kopieren
1B63:C3        42         ASC 'C'        ;Vergleich
1B64:17 16     43         DW  COMPARE    ;BUFFER mit HLBUF
1B66:          44 *
```

```
1B66:D0          45         ASC 'P'
1B67:75 17       46         DW  SETPRT     ;Printer SLOT & Dump
1B69:CE          47         ASC 'N'
1B6A:FF 17       48         DW  DSPOFF     ;DUMP-Display OFF
1B6C:CF          49         ASC 'O'
1B6D:F5 17       50         DW  DSPON      ;DUMP-Display ON
1B6F:            51 *
1B6F:C6          52         ASC 'F'
1B70:2D 18       53         DW  FIND       ;Absuchen von BUFFER
1B72:            54 *
1B72:D9          55         ASC 'Y'
1B73:3F 19       56         DW  YPS        ;JUMP-Label
1B75:CA          57         ASC 'J'
1B76:56 19       58         DW  JUMP       ;JUMP-Anweisung
1B78:            59 *
1B78:A5          60         ASC '%'
1B79:C0 19       61         DW  SETVARS    ;Variablen...
1B7B:            62 *
1B7B:DA          63         ASC 'Z'
1B7C:E5 1A       64         DW  SLEEP      ;WAIT-Routine
1B7E:            65 *
1B7E:BF          66         ASC '?'
1B7F:00 1B       67         DW  HELP       ;Kommando-Erklärungen
1B81:            68 *
1B81:            69 LCMD:   EQU *-CMDS
1B81:            70 *
1B81:            71 HLPTXT: EQU *
1B81:            72 *INCLUDE ED.CMDS.ADD ;falls erwünscht
1B81:00          73         DFB 00      ;Text-Ende
1B82:            74 *
1B82:            76 ******************************
1B82:            77 *
0200:            78 BUFLEN: EQU MAXBUF*$100
1B82:            79 *
2000:            80 BUFFER: EQU $2000
2200:            81 HLBUF:  EQU BUFFER+BUFLEN
2400:            82 FNBUF:  EQU HLBUF+BUFLEN
2500:            83 FNMASK: EQU FNBUF+$100
2600:            84 EDITEND: EQU FNMASK+$100
```

ED.CMDS.ADD

```
0000:22 40 2F 56    1        ASC '"$/Volname": Read 1st Block'
001B:20 6F 66 20    2        ASC '          of VolDIR'
0025:8D             3        DFB $8D
0026:22 40 2F 56    4        ASC '"$/Volname/Fname": Get 1st'
0040:20 42 6C 6F    5        ASC ' Block of File'
004E:0D             6        DFB $0D
004F:8D             7        DFB $8D
0050:22 52 50 78    8        ASC '"RPx,y": READ from Sx,Dy.'
0069:53 65 74 73    9        ASC 'Sets "WRITE to"
0078:0D             10       DFB $0D
0079:8D             11       DFB $8D
007A:22 52 20 78    12       ASC '"R xxx": READ Block xxx'
0091:8D             13       DFB $8D
0092:22 57 20 78    14       ASC '"W xxx": WRITE Block xxx'
00AA:8D             15       DFB $8D
00AB:8D             16       DFB $8D
00AC:22 44 78 78    17       ASC '"Dxx,yy": DUMP from xx '
00C3:74 6F 20 79    18       ASC 'to yy'
00C8:8D             19       DFB $8D
00C9:22 3C 22 2F    20       ASC '"<"/","/">"/".": Move '
00DF:43 75 72 73    21       ASC 'Cursor in DUMP'
00ED:8D             22       DFB $8D
00EE:8D             23       DFB $8D
00EF:22 53 78 78    24       ASC '"Sxx:Hex Hex...": Set'
0104:20 48 65 78    25       ASC ' Hex from xx...'
0113:8D             26       DFB $8D
0114:22 53 78 78    27       ASC '"Sxx"
0119:A7             28       DFB $A7        ; "'"
011A:41 53 43 49    29       ASC 'ASCII..": Set ASCII,'
012E:20 4D 53 42    30       ASC ' MSB off'
0136:8D             31       DFB $8D
0137:22 53 78 78    32       ASC '"Sxx"ASCII..": Set'
0149:20 41 53 43    33       ASC ' ASCII, MSB on'
0157:8D             34       DFB $8D
0158:8D             35       DFB $8D
0159:46 6F 72 20    36       ASC 'For more Help,'
0167:20 72 75 6E    37       ASC ' run EDIT.DOC.'
0175:8D             38       DFB $8D
```

Ein Patch für den „enhanced" Apple IIe

Der zusammen mit den neuen Monitor-ROMs gelieferte Maus-Zeichensatz benutzt leider andere Bildschirm-Codes für die inverse Darstellung. Die Folge: Ein ASCII-Dump mit EDIT produziert Grafikzeichen anstelle von Dateinamen und Texten – aber nur dann, wenn die 80-Zeichen-Karte aktiviert ist. Die folgende Veränderung schafft Abhilfe:

BLOAD EDIT
CALL -151
0AFD: EA EA
<Control-C> <RETURN>
BSAVE EDIT

Dieser Patch bewirkt das Setzen des höchstwertigen Bits innerhalb der Ausgaberoutine PRCHAR (Datei ED.FRAME.TXT, Zeile 410).

Weitere EDIT-Übungen

Wir werden auch hier wieder von einer Kopie der BENUTZER.DISK ausgehen (wie in Heft 6/86 beschrieben). Starten Sie EDIT mit *BRUN EDIT oder -EDIT* und legen Sie Ihre Kopie der BENUTZER.DISK in Laufwerk 1 ein. EDIT setzt das „Read-Prefix" automatisch auf „S6, D1" und zeigt ansonsten einen leeren Bildschirm.

1. Suchen von Dateieinträgen mit dem Befehl „F"ind

Die Befehlsfolge *$/BENUTZER.DISK!D* liest den ersten Block des Volume Directory und stellt ihn auf dem Bildschirm dar. Mit *F0'<Dateiname>* wird dieser Block nach einem bestimmten Dateieintrag abgesucht. *F0'ANIMALS* bewirkt die Darstellung des Blocks ab der Adresse $013D, d. h. ab der Stelle, an der sich der Dateieintrag ANIMALS befindet. Um mehrere Blocks eines Directory nach einem Eintrag abzusuchen, tippen Sie:

$/VolName ← z. B. BENUTZER.DISK
F0'Dateiname ← z. B. ANIMALS
R+!F0= ← Wenn nicht gefunden, nächster Block

Der Befehl *F0=* wiederholt die letzte Suchaktion ab der Adresse 000 im momentan geladenen (neuen) Block.

Innerhalb eines Subdirectory ist der Befehl „F"ind in derselben Weise anwendbar. Mit *$/Volname/Subdirectory-Name* wird der erste Block des Subdirectory geladen, mit *F0'Dateiname* der Eintrag gesucht. Leider stehen die Blocks eines Subdirectory nicht direkt hintereinander auf der Diskette: Sie müssen die jeweils nächste Blocknummer über die „Forward Reference" (Bytes 02 und 03 des Blocks) bestimmen. Beispiel: Wenn die Adressen 002/003 des gerade gelesenen Blocks die Werte $0D, $01 haben, dann hat der nächste Block die Nummer $010D und kann mit *R010D* gelesen werden.

Selbstverständlich kann man auch eine ganze Diskette nach einer Zeichenfolge absuchen. *R0* liest den ersten Block, *F'Textstelle* sucht diesen Block nach der Zeichenfolge *Textstelle* ab, R+ liest den nächsten Block etc. Damit Sie diesen Prozeß nicht 280 Mal wiederholen müssen, „verpacken" wir ihn in eine Schleife:

R<Blocknummer ← z.B. 007, d.h. nach dem Volume Direcotry
F"ProDOS ← Suche nach Text ProDOS, mit MSB gesetzt
Y0!R+!F0=!J-0

Eine „Übersetzung":
10 „Lese nächsten Block": „Suche nach *ProDOS*: „Wenn nicht gefunden", GOTO 10

Diese Schleife braucht immerhin 55 Sekunden, bis der gesuchte Text innerhalb der Hilfestellung von CONVERT gefunden wird (Blocknummer $008B, Adresse $18E). Der Grund dafür liegt darin, daß jeder gelesene Block auf dem Bildschirm dargestellt wird. Wenn wir den Bildschirm während der Suche mit dem Befehl *N* abschalten, reduziert sich der Zeitbedarf auf die Hälfte (knapp 30 Sekunden):

N!Y0!R+!F0=!J-0!O!F0=

2. Wiederherstellung gelöschter Dateien

Eigentlich geht das unter ProDOS nicht ... Tatsache ist, daß ein „Undelete" recht kompliziert ausfällt und sich in den meisten Fällen auch nur für Textdateien des „Storage Type" 2 (Dateigröße: 513 bis 131072 Byte) lohnt.

Verlassen Sie EDIT mit *Q* und erstellen Sie die Datei TEST mit dem folgenden Applesoft-Programm:

```
10 PRINT CHR$ (4); „OPEN TEST":
   PRINT CHR$ (4); „WRITE TEST"
20 FOR X = 65 TO 70: FOR Y = 1 TO
   100
30 PRINT CHR$ (X);: NEXT: NEXT
40 PRINT CHR$ (4); „CLOSE"
50 PRINT CHR$ (4); „DELETE TEST"
NEW
```

Die gelöschte Datei TEST enthält hundertmal den Buchstaben „A" (d. h. CHR$65)), hundertmal den Buchstaben „B" usw. Starten Sie EDIT erneut und lassen Sie uns probieren, TEST wiederherzustellen.

R6!V

liest die Volume Bit Map der Diskette und stellt ihren Inhalt in einzelnen Bits dar. Ab Adresse $018 sind die Bits gesetzt, d. h. ab der Blocknummer $C0 (=$18 * $08) sind die Blocks der Diskette frei.

RC0!D

Und tatsächlich: Der Block $00C0 enthält die ersten 512 Byte der Datei TEST. Schreiben Sie sich diese Blocknummer auf. (Wenn Sie mehrere Dateien auf einer Diskette gelöscht haben, müssen Sie alle freien Blocks überprüfen und dabei mit der niedrigsten Blocknummer beginnen.) Suchen Sie nun den Dateieintrag von TEST innerhalb des Volume-Directory mit den unter Punkt 1) beschriebenen Befehlsfolgen. (Block $0002, ab Adresse $163). Das achtzehnte und neunzehnte Byte des Eintrags (hier: Adressen $174/75) enthält die Blocknummer des Indexblocks: $00C1.

Mit der Befehlsfolge:

D163!S163:24!W

wird der Eintrag von TEST wiederhergestellt (Storage Type $02, Namenslänge $04 Byte). Die Anzahl der „aktiven Einträge" des Directory muß nun wieder um eins erhöht werden. Sie findet sich auf den Bytes $008/09 im ersten Block des Directory und hat momentan den Wert $0008. Mit

R2!D!S25:9!W

wird sie auf den Wert neun erhöht und der Directory-Block zurückgeschrieben. Wie Sie mit der Befehlsfolge

X
CAT
<Control-Y>

feststellen können, existiert nun der Eintrag wieder.

Jetzt müssen wir nur noch die restlichen Blocks der Datei finden und im Indexblock eintragen. Mit

R00C1!D!H

wird der Indexblock eingelesen und festgehalten. Wie zu sehen, ist der erste Datenblock von TEST ($00C0) noch verzeichnet. Prüfen Sie nun die folgenden Blocks mit R+ und schreiben Sie die Blocknummern auf, wenn der momentan gelesene Block zur Datei gehört. (Bei der Datei TEST ist das nur noch für den Block $00C2 der Fall.) Mit dem Befehl *M* holen Sie den Indexblock wieder in den Arbeitspuffer. Hier müssen Sie jetzt die aufgeschriebenen Blocknummern eintragen, also:

S001: C2 ←der niederwertige Anteil
S101: 00 ←der höherwertige Anteil

Mit dem Befehl *W00C1* (nicht *W* allein!) wird der Indexblock auf die Diskette zurückgeschrieben.

Sie haben nun zwei Möglichkeiten: Entweder kopieren Sie die gerettete Datei sofort auf eine andere Diskette und löschen sie danach auf der Originaldiskette mit DELETE TEST – oder Sie setzen auch noch die entsprechenden Bits innerhalb der Volume Bit Map *(R6!S18:1F!W)* und stellen die Datei damit komplett wieder her.

Arne Schäpers

Netzwerk für Apple-Rechner

Low-Cost-Rechner-Kopplung für jedermann

von Alexander Niemeyer

Für viele Anwendungen kann es sinnvoll sein, mehrere Apples (oder Kompatible) miteinander kommunizieren zu lassen. Dies kann durch herkömmliche Netzwerke und Interfaces geschehen, wie zum Beispiel durch den IEEE-488-Bus. Das ist jedoch teuer und aufwendig.
Ich möchte Ihnen hier ein Netzwerk vorstellen, das im Eigenbau erstellt werden kann und sich durch seine minimalen Kosten auszeichnet. Natürlich ist dieses Netzwerk nicht annähernd so komfortabel wie ein herkömmliches; die Interfaces sind nämlich nicht intelligent. Es ist jedoch ohne größere Schwierigkeiten möglich, Software zu schreiben, die sich des Netzwerkes bedient.
Der Zeitaufwand hierfür wird durch die niedrigen Kosten mehr als wettgemacht. So hat die Installation des hier beschriebenen Systems an einer Schule mit 7 Apples (und Kompatiblen) ca. 160,- DM gekostet, einschließlich *aller* mechanischen Teile (ca. 50m Buskabel). Die Übertragungsleistung ist dabei recht beachtlich, es können Daten mit 40 KBaud von jedem beliebigen Rechner zu allen anderen Rechnern gleichzeitig gesendet werden. (Die Beschränkung auf 40 KBaud ist durch Software gegeben.)

1. Die Hardware des Netzwerkes

Das Netzwerk besteht aus einer Busleitung, an die sämtliche Rechner mittels einfachster Interfaces angeschlossen werden. Es gibt dabei keine Unterschiede bei den einzelnen Interfaces, so daß alle Computer hardware-mäßig völlig gleichberechtigt sind.
Die Busleitung ist ein 4adriges Kabel mit Abschirmung. Es gibt zwei Datenleitungen und eine Steuerleitung, so daß immer zwei Bits parallel übertragen werden. Eine Ader ist unbelegt, da Masse an der Abschirmung liegt. Diese Ader kann dann für spätere Erweiterungen benutzt werden, z.B. für +5V oder für eine Interrupt-Leitung. Es ist besser, ein Kabel mit einer Ader zuviel zu verlegen, als später ein zweites Kabel neben die alte Leitung kleben zu müssen.

1.1. Das Interface

Jeder anzuschließende Rechner benötigt ein Interface, das ihn mit dem Bus verbindet. Das Interface basiert dabei auf den 1-Bit-Ein/Ausgängen des Game-Connectors, der für solche Zwecke wie geschaffen ist. Die drei Eingänge des Game-I/O werden direkt mit den Datenleitungen und mit der Steuerleitung verbunden. So können immer alle Rechner die Informationen auf dem Bus gleichzeitig empfangen, sie müssen es aber nicht.
Da die Ausgänge des Game-Connectors Standard-TTL-Ausgänge sind, dürfen sie nicht alle gleichzeitig mit den Busleitungen verbunden werden, da nämlich ein Kurzschluß entsteht, wenn Rechner 1 eine logische 1 z.B. auf die Steuerleitung legt, während Rechner 2 gleichzeitig eine logische 0 schickt. Das könnte die Ausgänge beschädigen. Die Ausgänge werden daher über Tri-State-Treiber mit den Busleitungen verbunden. Mit Hilfe der Steueranschlüsse dieser Treiber können gezielt die Ausgänge eines Rechners auf den Bus gelegt werden, während die Ausgänge der anderen Computer keinen Einfluß auf den Zustand der Leitungen haben. Jeder der drei benötigten Ausgänge muß einen eigenen Treiber haben. Für die Steueranschlüsse der Treiber ist noch ein zusätzlicher Ausgang nötig, der ebenfalls ein Game-Ausgang ist; der Gameport besitzt glücklicherweise 4 Ausgänge. Somit sind lediglich drei Treiber nötig, um ein Interface zu realisieren. Als Treiber-IC wird der 74125 (vier Tri-State-Treiber) benutzt. Bitte verwenden Sie den Standard-Typ und nicht die LS-Ausführung, da der Standard-Typ erstens billiger und zweitens leistungsfähiger ist.
Damit ist das Interface fertig. **Abb. 2** zeigt den Schaltplan und die Pin-Belegung des 74125.
Die Schaltung (Schaltplan in **Abb. 1**) kann auf einer winzigen Lochraster-Platine von 16 x 8 Lötaugen bequem aufgebaut werden, wenn man nicht zwei linke Hände hat. Der IC sollte gesockelt werden, damit er im eventuellen Schadensfalle leicht ausgetauscht werden kann. Der Gameport sollte nicht mit fliegenden Leitungen bestückt werden, da eine herausrutschende Leitung den Computer beschädigen könnte. Am besten eignet sich ein richtiger 16poliger DIL-Stecker zum Anpressen oder Anlöten. Wird die Platine geätzt, so sollte auch der Steuereingang des 4. Treibers mit den anderen Steuereingängen

verbunden werden, um spätere Erweiterungen zu erleichtern.

1.2. Anschluß an den Bus

Der Bus sollte mit den Computern auf jeden Fall durch Steckverbindungen gekoppelt sein, damit die Computer mobil bleiben. Die 5poligen Diodenstecker bzw. -buchsen aus dem Rundfunk- und Hi-Fi-Bereich bieten sich hier besonders an, da sie billig und in vielen Variationen zu erhalten sind.
Eine Diodenbuchse ist allerdings nicht leicht auf der Rückseite des Apple unterzubringen. Eine große Bohrung oberhalb der Kassettenanschlüsse ist jedoch eine sehr gute Lösung. Auf jeden Fall sollte die Buchse verschraubt werden, damit sie nicht herausgerissen werden kann.
In diese Buchse kommt dann ein Diodenstecker, der an das Buskabel gelötet wird. Da ein normales Kabel meist nur zwei Enden hat, sind Abzweigungen erforderlich. Ideal dafür geeignet sind einfache Aufputz-Abzweigdosen, wie sie im Elektrofachhandel für sehr wenig Geld zu erhalten sind. Nach dem Ausschneiden der vorgesehenen Öffnungen lassen sich bis zu sechs Diodenbuchsen einpressen, die auch ohne weitere Befestigung sehr gut sitzen. In der Abzweigdose werden dann die entsprechenden Anschlüsse aller Buchsen miteinander verbunden. Zwischen den einzelnen Abzweigdosen können dann einfache Kabel mit Diodenbuchsen an beiden Enden verwendet werden.
Durch diese Konstruktion ist das Netzwerk sehr mobil. Es kann leicht vergrößert oder verkleinert werden, außerdem läßt es sich bequem an die räumlichen Gegebenheiten anpassen (z.B. Ringleitung auf der Fußbodenleiste mit einer Abzweigdose für jeden Rechner; sternförmiges Netzwerk usw.).
Durch Ziehen eines Steckers läßt sich das Netzwerk auch aufspalten – kurzum, die Möglichkeiten sind sehr vielfältig.
Die Busleitungen müssen allerdings immer auf definiertem Pegel liegen. Am einfachsten ist das dadurch zu realisieren, daß man jeden Anschluß eines Diodensteckers über 1 KOhm mit Masse verbindet. Das ergibt einen sehr handlichen Stecker. Bei einer Aufspaltung muß für jedes Teilnetzwerk ein solcher Stecker vorhanden sein.

1.3. Die Kosten

Pro Interface belaufen sich die Kosten auf etwa DM 10,- (je nach örtlichen Preisunterschieden). Ein Meter Buskabel kostet etwa DM 1,50, eine Diodenbuchse oder -stecker etwa DM 0,80, und eine Abzweigdose schlägt auch nur mit DM 1,- zu Buche. Die investierte Arbeitszeit ist allerdings erheblich, da etliche Steckverbindungen gelötet werden müssen.
Die Kosten können weiter gesenkt werden, wenn man auf abgeschirmtes Kabel verzichtet und normales nimmt. Es ist mir jedoch nicht bekannt, ob bei den durch Software erreichbaren 40 KBaud ein Übersprechen oder ähnliche Effekte auftreten. Die abgeschirmten Leitungen funktionieren immerhin auch bei 200 KBaud noch einwandfrei (über eine Distanz von 35m getestet).

2. Das Übertragungsprotokoll

Es werden immer 2 Bits parallel übertragen. Die Übertragung erfolgt dabei weder synchron noch asynchron, da hierbei Zeitverzögerungsroutinen benutzt werden müßten, die die Übertragungsrate minderten. Statt dessen wird durch die Steuerleitung signalisiert, wann auf den Datenleitungen gültige Daten liegen. Die Empfängerroutine wartet ab, bis die Steuerleitung ein „gültig“ signalisiert, dann übernimmt sie die Daten. Anschließend wartet sie darauf, daß die Steuerleitung wieder „ungültig“ sagt, um anschließend auf die nächsten zwei Bits zu warten.
Ein „gültig“ wird durch eine logische 1 auf der Steuerleitung (STRB) dargestellt. Damit ist auch klar, warum die Busleitungen auf Masse gezogen werden müssen, wenn kein Rechner sendet. Wenn nämlich die Leseroutine, die schon den Bus „abhorcht“, eine zufällige 1 auf der Steuerlei-

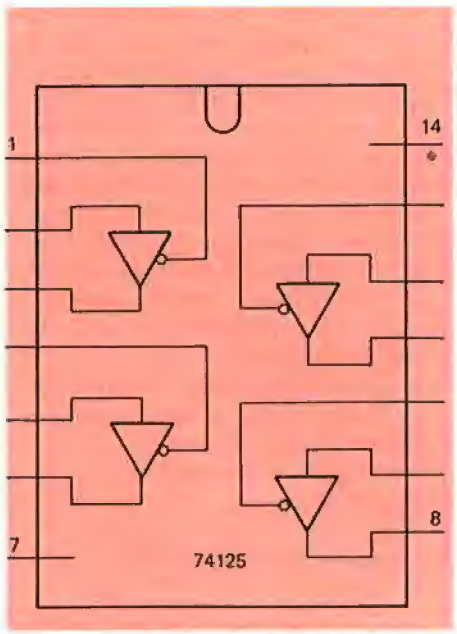


Abb. 1: Blockschaltbild und Pinbelegung des SN 74125

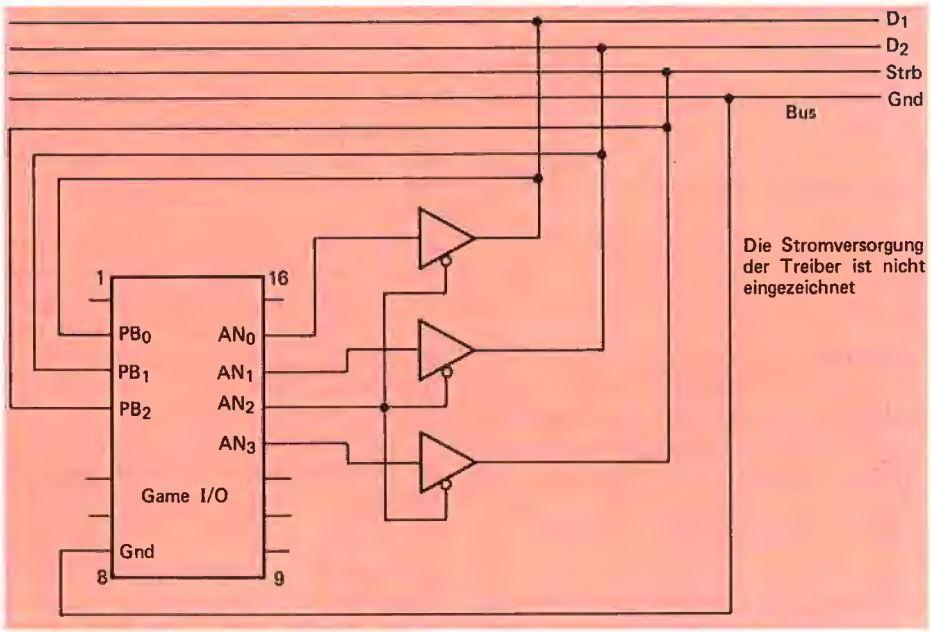


Abb. 2: Schaltbild des Interfaces

tung findet, so übernimmt sie die Zustände auf den Datenleitungen als gültige Daten.

Die Senderoutine geht davon aus, daß die Empfangsroutine schon „läuft", wenn die Sendung beginnt. Es besteht also keine Möglichkeit für einen Handshake, denn es können ja beliebig viele Rechner lesen, während keiner lesen *muß*.

Zusätzlich schaltet die Senderoutine vor jeder Sendung die Treiber durch, um die Ausgänge des eigenen Rechners auf den Bus zu legen. Sendet zur gleichen Zeit bereits ein anderer Computer, so ist ein Datenchaos die Folge. Obwohl dabei theoretisch die Treiber kaputtgehen könnten, ist mir das bisher nicht gelungen.
Glücklicherweise setzt der Apple die 1-Bit-Ausgänge beim Reset auf definierte Pegel, so daß nach dem Einschalten oder nach Reset alle Interfaces abgeschaltet sind. Aber Vorsicht! Nicht alle Kompatiblen initialisieren auch alle Ausgänge. Insbesondere der 4. Ausgang wird bei Kompatiblen mit ASCII/Deutsch-Umschaltung oft nicht gesetzt. Das Netzwerk verwendet deshalb den 3. Ausgang als ENABLE und nicht Ausgang Nr. 4.
Es läßt sich jedoch nicht vermeiden, daß beim Einschalten eines Computers das Interface kurz eingeschaltet wird, da vom Einschalten bis zum Selbst-Reset des Rechners einige Zehntelsekunden vergehen. Datenübertragungen können deshalb durch das Einschalten eines weiteren Apple gestört werden.

Das abgedruckte Programm **NETZWERK** implementiert verschiedene Kommunikationsfunktionen als BASIC-Erweiterung, die mit dem &-Befehl aufgerufen werden. Die darin enthaltenen Unterroutinen (READBYTE, SENDBYTE, TESTBUS) können für andere Programme eingesetzt werden.

Die Funktionen sind:

&SAVE – sendet das momentane BASIC-Programm
&LOAD – holt ein BASIC-Programm vom Bus
&STORE a,e – sendet den Speicherbereich von Adresse a bis Adresse e
&RECALL a,e – füllt den Speicherbereich von Adresse a bis Adresse e mit Busdaten
&PR# – sendet alle Ausgaben auf den Bus
&IN# – empfängt Eingaben vom Bus

Die beiden letzten Funktionen können z.B. verwendet werden, um Daten von einem Programm an das andere mit PRINT- und INPUT-Statements zu übergeben. Man beendet diese Funktionen wie üblich mit PRINT CHR$(4)"PR#0" bzw. "IN#0".
Die &PR#-Funktion gibt die Ausgaben sowohl auf den Bildschirm als auch auf den Bus aus, wenn die Speicherstelle 1656 einen Wert > 127 hat, sonst gehen die Ausgaben nur auf den Bus.
Die &IN#-Funktion arbeitet nur dann vernünftig, wenn die Daten nicht schneller gesendet als empfangen werden. Außerdem wird ein eventuell gesendeter DOS-Befehl zuerst auf dem sendenden Rechner ausgeführt, wenn dieser im Befehlsmodus ist. Das ist darauf zurückzuführen, daß DOS erst dann das Return ausgibt, wenn der DOS-Befehl fertig ausgeführt ist. (*Vorsicht:* Initialisieren Sie niemals eine Diskette, wenn Sie vorher &PR# eingegeben haben. Alle Rechner mit einem laufenden &IN# initialisieren dann ihre Disketten ebenfalls!)

3. Variationsmöglichkeiten für das Netzwerk

Der beschriebene Hardware-Aufbau ist natürlich nur ein Vorschlag. Zwei mögliche Variationen möchte ich hier noch aufzeigen:
– Mit dem vierten Treiber und mit dem Utility-Strobe des Game-Connectors sollte es möglich sein, bei allen Computern gezielt ein Interrupt auszulösen. Diese Busleitung muß dann aber einen Ruhepegel von +5V haben, da dies auch der Ruhepegel des Strobe ist. Die Busleitung müßte dann mit den IRQ-Eingängen des Mikroprozessors irgendwo auf der Hauptplatine verbunden werden. Allerdings würde ein solcher Interrupt immer auf allen Rechnern ausgelöst, die die I-Flags nicht gesetzt haben, also auch auf dem eigenen Rechner! Bei zu großer Leitungslänge könnte ich mir jedoch vorstellen, daß der nur eine halbe Mikrosekunde lange Impuls des Strobe durch die Kapazität der Leitungen „verschlungen" wird.
– Wenn man auf eine Datenleitung verzichtet (Software umschreiben!), so kann man die beiden Treiber, die nur noch nötig sind, einzeln steuern. Es ist somit möglich, zwei völlig getrennte Kanäle zu realisieren, die von verschiedenen Rechnern gleichzeitig benutzt werden können.

3.1. Hardware-spezifische Einschränkungen

Einige Apple-kompatiblen Rechner benutzen einen ihrer Ausgänge schon (z.B. für Deutsch/ASCII-Umschaltung). Hier muß man entweder eine Datenleitung opfern oder die Deutsch/ASCII-Umschaltung in Zukunft über einen Schalter betätigen.

80-Zeichenkarten schränken das Netzwerk ebenfalls ein. Meine 80-Zeichenkarte kann aus dem 40-Zeichen-Modus durch den ersten Ausgang ein- und ausgeschaltet werden. Im 80-Zeichenmodus hat dieser Ausgang dagegen keine Wirkung mehr. Ist das auch bei den 80-Zeichenkarten der beteiligten Rechner der Fall, so müssen diese bei einer Sendung im 80-Zeichenmodus sein, sonst flimmert der Bildschirm unerträglich. Sie können natürlich auch auf die entsprechende Datenleitung verzichten.

Leider gibt es mindestens einen kompatiblen Rechner, der keine 1-Bit-Ausgänge hat. Es ist dies der BASE 64A. Er kann in diesem Netzwerk niemals senden, sondern nur lesen (Sie brauchen gar keinen IC in den Sockel des Interfaces zu stekken). Verwenden Sie ihn als Drucker-Spooler.

4. Anwendungsmöglichkeiten

Anwendungsmöglichkeiten für das Netzwerk sind zum Beispiel:
1. Im Informatikunterricht (Programme und Daten können frei verteilt werden, ohne daß alle mit verschiedenen Disketten durch die Gegend laufen. Außerdem kann auf einem Rechner ein Programm ausgeführt werden, dessen Ausgaben dann auf den Monitoren aller Computer erscheinen.)
2. Zur Schulverwaltung (besonders geeignet zum Zeugnisdrucken: Es werden z.B. auf zwei Rechnern Noten eingegeben, die Ausgabe der Zeugnisse erfolgt dann auf anderen Computern, die als Drucker-Spooler dienen.)
3. Sogar zum Spielen kann man das Netzwerk verwenden (Schiffeversenken für 2, Skat für 3 oder 4 Rechner. Auch kann ein Apple auf dem anderen Flipper spielen.)

Kurzhinweise

1. Zweck:
Kostengünstige Hard- und Software zum Aufbau eines Apple-Netzwerks

2. Konfiguration:
II+ oder IIe (Kompatible mit Einschränkung);
Interface und Buskabel;
DOS 3.3

3. Test
Nach Installierung:
BRUN NETZWERK auf zwei Geräten
&PR# bei Gerät 1
&IN# bei Gerät 2
CATALOG bei Gerät 1

4. Sammeldisk:
T.NETZWERK
(Big-Mac-Quelltext)
NETZWERK
(Maschinenprogramm)

NETZWERK

BSAVE NETZWERK, A$9400, L$01C5

```
1   ********************************
2   *                              *
3   *          NETZWERK            *
4   *                              *
5   *   Einbindung des Netzwerks   *
6   *         in Applesoft         *
7   *                              *
8   * Von Alexander Niemeyer, 1985 *
9   *                              *
10  ********************************
11
12  ADR       EQU  $50
13  TXTTAB    EQU  $67
14  VARTAB    EQU  $69
15  HIMEM     EQU  $73
16  PROGEND   EQU  $AF
17  CHRGET    EQU  $B1
18  CHRGOT    EQU  $B7
19  XSAV      EQU  $FD
20  ADR1      EQU  $FE
21  AMPER     EQU  $3F5
22  OUTVECT   EQU  $AA53
23  INVECT    EQU  $AA55
24  DAT1OFF   EQU  $C058
25  DAT1ON    EQU  $C059
26  DAT2OFF   EQU  $C05A
27  DAT2ON    EQU  $C05B
28  ENABLE    EQU  $C05C
29  DISABLE   EQU  $C05D
30  STRBOFF   EQU  $C05E
31  STRBON    EQU  $C05F
32  DAT1      EQU  $C061
33  DAT2      EQU  $C062
34  STROBE    EQU  $C063
35  RESTART   EQU  $D43C
36  LINKSET   EQU  $D4F2
37  FRMNUM    EQU  $DD67
38  CHKCOM    EQU  $DEBE
39  GETADR    EQU  $E752
40  COUT      EQU  $FDF0
41
42  * TOKENS der Befehle
43
44  SAVE      EQU  $B7
45  LOAD      EQU  $B6
46  STORE     EQU  $A8
47  RECALL    EQU  $A7
48  PR        EQU  $8A
49  IN        EQU  $8B
50
```

```
                    51          ORG  $9400
                    52
9400: A9 4C         53  BEGIN   LDA  #$4C       ;JUMP
9402: 8D F5 03      54          STA  AMPER
9405: A9 18         55          LDA  #<ENTRY
9407: 8D F6 03      56          STA  AMPER+1
940A: A9 94         57          LDA  #>ENTRY
940C: 8D F7 03      58          STA  AMPER+2
940F: A9 00         59          LDA  #<BEGIN
9411: 85 73         60          STA  HIMEM
9413: A9 94         61          LDA  #>BEGIN
9415: 85 74         62          STA  HIMEM+1
9417: 60            63          RTS
                    64
9418: C9 B7         65  ENTRY   CMP  #SAVE      ;Einzel-
941A: F0 5E         66          BEQ  SAVERT      ;vergleich
941C: C9 B6         67          CMP  #LOAD
941E: F0 1D         68          BEQ  LOADRT      ;Tabelle
9420: C9 A8         69          CMP  #STORE      ;lohnt
9422: F0 0D         70          BEQ  STORERT1    ;nicht
9424: C9 A7         71          CMP  #RECALL
9426: F0 0C         72          BEQ  RECRT1
9428: C9 8A         73          CMP  #PR
942A: F0 0B         74          BEQ  PRRT1
942C: C9 8B         75          CMP  #IN
942E: F0 0A         76          BEQ  INRT1
9430: 60            77          RTS              ;SYNTAX
                    78  *                        ;ERROR
9431: 4C F9 94      79  STORERT1 JMP  STORERT
9434: 4C 2D 95      80  RECRT1   JMP  RECRT
9437: 4C C6 94      81  PRRT1    JMP  PRRT
943A: 4C B8 94      82  INRT1    JMP  INRT
                    83
943D: 20 90 95      84  LOADRT   JSR  READBYTE   ;Anfangs-
9440: 85 67         85          STA  TXTTAB      ;adresse
9442: 8D 5F 94      86          STA  COUNT+1     ;holen
9445: 20 90 95      87          JSR  READBYTE
9448: 85 68         88          STA  TXTTAB+1
944A: 8D 60 94      89          STA  COUNT+2
944D: 20 90 95      90          JSR  READBYTE
9450: 85 AF         91          STA  PROGEND     ;und End-
9452: 85 69         92          STA  VARTAB      ;adresse
9454: 20 90 95      93          JSR  READBYTE
9457: 85 B0         94          STA  PROGEND+1
9459: 85 6A         95          STA  VARTAB+1
945B: 20 90 95      96  STORELP  JSR  READBYTE   ;Daten
945E: 8D FF FF      97  COUNT    STA  $FFFF       ;lesen
9461: EE 5F 94      98          INC  COUNT+1     ;selbst-
9464: D0 03         99          BNE  NOUPCNT     ;modifi-
9466: EE 60 94      100         INC  COUNT+2     ;zieren-
9469: AD 5F 94      101 NOUPCNT LDA  COUNT+1     ;der
946C: C5 AF         102         CMP  PROGEND     ;Zähler
946E: D0 EB         103         BNE  STORELP
9470: AD 60 94      104         LDA  COUNT+2
9473: C5 B0         105         CMP  PROGEND+1
9475: D0 E4         106         BNE  STORELP
9477: 4C F2 D4      107         JMP  LINKSET
                    108
```

```
947A: 20 B2 95  109  SAVERT   JSR  TESTBUS
947D: A5 67     110           LDA  TXTTAB     ;Anfangs-
947F: 8D 98 94  111           STA  SNDCNT+1   ;und End-
9482: 20 5E 95  112           JSR  SENDBYTE   ;adresse
9485: A5 68     113           LDA  TXTTAB+1   ;senden
9487: 8D 99 94  114           STA  SNDCNT+2
948A: 20 5E 95  115           JSR  SENDBYTE
948D: A5 AF     116           LDA  PROGEND
948F: 20 5E 95  117           JSR  SENDBYTE
9492: A5 B0     118           LDA  PROGEND+1
9494: 20 5E 95  119           JSR  SENDBYTE
9497: AD FF FF  120  SNDCNT   LDA  $FFFF      ;Programm
949A: 20 5E 95  121           JSR  SENDBYTE   ;senden
949D: EA        122           NOP             ;warten
949E: EE 98 94  123           INC  SNDCNT+1
94A1: D0 03     124           BNE  NOUPSND
94A3: EE 99 94  125           INC  SNDCNT+2
94A6: AD 98 94  126  NOUPSND  LDA  SNDCNT+1
94A9: C5 AF     127           CMP  PROGEND
94AB: D0 EA     128           BNE  SNDCNT
94AD: AD 99 94  129           LDA  SNDCNT+2
94B0: C5 B0     130           CMP  PROGEND+1
94B2: D0 E3     131           BNE  SNDCNT
94B4: 20 B1 00  132           JSR  CHRGET
94B7: 60        133           RTS
                134
94B8: A9 EB     135  INRT     LDA  #<INDO     ;INDO
94BA: 8D 55 AA  136           STA  INVECT     ;wird zur
94BD: A9 94     137           LDA  #>INDO     ;KEYIN-
94BF: 8D 56 AA  138           STA  INVECT+1   ;Routine
94C2: 20 B1 00  139           JSR  CHRGET
94C5: 60        140           RTS
                141
94C6: 20 B2 95  142  PRRT     JSR  TESTBUS
94C9: A9 D7     143           LDA  #<OUTDO    ;OUTDO
94CB: 8D 53 AA  144           STA  OUTVECT    ;wird zur
94CE: A9 94     145           LDA  #>OUTDO    ;COUT-
94D0: 8D 54 AA  146           STA  OUTVECT+1  ;Routine
94D3: 20 B1 00  147           JSR  CHRGET
94D6: 60        148           RTS
                149
94D7: 48        150  OUTDO    PHA             ;Register
94D8: 08        151           PHP             ;sichern
94D9: 86 FD     152           STX  XSAV
94DB: 20 5E 95  153           JSR  SENDBYTE
94DE: A6 FD     154           LDX  XSAV
94E0: 28        155           PLP
94E1: 68        156           PLA
94E2: 2C 78 06  157           BIT  1656       ;Echo auf
94E5: 10 03     158           BPL  NOECHO     ;CRT?
94E7: 4C F0 FD  159           JMP  COUT
94EA: 60        160  NOECHO   RTS
                161
94EB: 86 FD     162  INDO     STX  XSAV
94ED: 20 90 95  163           JSR  READBYTE
94F0: A6 FD     164           LDX  XSAV
94F2: 60        165           RTS
                166
94F3: 20 67 DD  167  ADRGET   JSR  FRMNUM     ;Adr. holen
94F6: 4C 52 E7  168           JMP  GETADR
                169
94F9: 20 B2 95  170  STORERT  JSR  TESTBUS
94FC: 20 B1 00  171           JSR  CHRGET
94FF: 20 F3 94  172           JSR  ADRGET     ;Anfang
9502: A6 50     173           LDX  ADR        ;ADR1 =
9504: 86 FE     174           STX  ADR1       ;Anfang
9506: A6 51     175           LDX  ADR+1
9508: 86 FF     176           STX  ADR1+1
950A: 20 BE DE  177           JSR  CHKCOM
950D: 20 F3 94  178           JSR  ADRGET     ;Ende
9510: A0 00     179           LDY  #0
9512: B1 FE     180  TRANSLP  LDA  (ADR1),Y
9514: 20 5E 95  181           JSR  SENDBYTE   ;Senden
9517: A5 FE     182           LDA  ADR1
9519: C5 50     183           CMP  ADR
951B: A5 FF     184           LDA  ADR1+1
951D: E5 51     185           SBC  ADR+1
951F: B0 08     186           BCS  ENDTRANS
9521: E6 FE     187           INC  ADR1
9523: D0 ED     188           BNE  TRANSLP
9525: E6 FF     189           INC  ADR1+1
9527: D0 E9     190           BNE  TRANSLP
9529: 20 B7 00  191  ENDTRANS JSR  CHRGOT
952C: 60        192           RTS
                193
952D: 20 B1 00  194  RECRT    JSR  CHRGET
9530: 20 F3 94  195           JSR  ADRGET     ;Anfang
9533: A6 50     196           LDX  ADR
9535: 86 FE     197           STX  ADR1
9537: A6 51     198           LDX  ADR+1
9539: 86 FF     199           STX  ADR1+1
953B: 20 BE DE  200           JSR  CHKCOM
953E: 20 F3 94  201           JSR  ADRGET     ;Ende
9541: A0 00     202           LDY  #0
9543: 20 90 95  203  READLOOP JSR  READBYTE   ;lesen
9546: 91 FE     204           STA  (ADR1),Y
9548: A5 FE     205           LDA  ADR1
954A: C5 50     206           CMP  ADR
954C: A5 FF     207           LDA  ADR1+1
954E: E5 51     208           SBC  ADR+1
9550: B0 08     209           BCS  ENDREC
9552: E6 FE     210           INC  ADR1
9554: D0 ED     211           BNE  READLOOP
9556: E6 FF     212           INC  ADR1+1
9558: D0 E9     213           BNE  READLOOP
955A: 20 B7 00  214  ENDREC   JSR  CHRGOT
955D: 60        215           RTS
                216
                217  * Universell verwendbare Sende-
                218  * und Empfangsroutinen
                219
                220  ************************************
                221  * Sendet Byte im Akku auf den Bus *
                222  ************************************
955E: 2C 5E C0  223  SENDBYTE BIT  STRB0FF
9561: 2C 5C C0  224           BIT  ENABLE
9564: A2 04     225           LDX  #4         ;4 * 2 Bits
9566: 0A        226  SENDLP   ASL
9567: 90 05     227           BCC  LOG0
9569: 8D 59 C0  228           STA  DAT1ON
956C: B0 03     229           BCS  NOLOG
956E: 2C 58 C0  230  LOG0     BIT  DAT1OFF
9571: 0A        231  NOLOG    ASL
9572: 90 05     232           BCC  LOG01
9574: 8D 5B C0  233           STA  DAT2ON
9577: B0 03     234           BCS  NOLOG1
9579: 2C 5A C0  235  LOG01    BIT  DAT2OFF
957C: 2C 5F C0  236  NOLOG1   BIT  STRBON
957F: EA        237           NOP             ;READBYT
9580: EA        238           NOP             ;muß
9581: EA        239           NOP             ;STROBE-
9582: EA        240           NOP             ;Wechsel
9583: EA        241           NOP             ;fest-
9584: EA        242           NOP             ;stellen
9585: EA        243           NOP             ;können
9586: 2C 5E C0  244           BIT  STRB0FF
9589: CA        245           DEX
958A: D0 DA     246           BNE  SENDLP
958C: 2C 5D C0  247           BIT  DISABLE
958F: 60        248           RTS
                249
                250  ********************************
                251  * Hole ein Byte vom Bus        *
                252  ********************************
9590: A2 04     253  READBYTE LDX  #4         ;4 * 2 Bits
9592: 2C 63 C0  254  READLP   BIT  STROBE     ;STRB = 1
9595: 10 FB     255           BPL  READLP     ;abwarten
9597: 2C 61 C0  256           BIT  DAT1       ;D1 lesen
959A: 30 02     257           BMI  SETZC      ;und
959C: 18        258           CLC             ;rotieren
959D: 24        259           HEX  24         ;Dummy-BIT
959E: 38        260  SETZC    SEC
959F: 2A        261           ROL
95A0: 2C 62 C0  262           BIT  DAT2       ;wieder-
95A3: 30 02     263           BMI  SETZC1     ;holen
95A5: 18        264           CLC             ;mit D2
95A6: 24        265           HEX  24
95A7: 38        266  SETZC1   SEC
95A8: 2A        267           ROL
95A9: 2C 63 C0  268  WAITLP   BIT  STROBE     ;STRB = 0
95AC: 30 FB     269           BMI  WAITLP     ;abwarten
95AE: CA        270           DEX             ;weiter
95AF: D0 E1     271           BNE  READLP
95B1: 60        272           RTS
                273
                274  ********************************
                275  * Testet, ob der Bus frei ist  *
                276  ********************************
95B2: A0 00     277  TESTBUS  LDY  #0          ;wenn Bus
95B4: 2C 63 C0  278  TESTLP   BIT  STROBE     ;bereits
95B7: 30 04     279           BMI  SENDERR    ;aktiv,
95B9: 88        280           DEY             ;dann
95BA: D0 F8     281           BNE  TESTLP     ;Warmstrt
95BC: 60        282           RTS
95BD: A9 87     283  SENDERR  LDA  #$87       ;BELL
95BF: 20 F0 FD  284           JSR  COUT
95C2: 4C 3C D4  285           JMP  RESTART


453 Bytes
```

Imagewriter II und Unidisk

im Kurztest

von Ulrich Stiehl

Zum ersten Mal seit dem nunmehr fast zweijährigen Bestehen des Peekers hat uns die Firma Apple zwei Testgeräte ausgeliehen, jedoch im Gegensatz zu anderen Firmen leider nur mit einer Ausleihfrist von 4 Wochen einschließlich Hin- und Rücksendung. Ich habe Verständnis dafür, daß man sich zu keiner längeren Ausleihfrist durchringen konnte, denn dadurch wird Kapital gebunden, das dann für andere Zwecke nicht mehr zur Verfügung steht. Indes müßte man sich bei Apple eigentlich denken können, daß man in dieser kurzen Zeit keinen umfassenden Testbericht mit entsprechenden Utilities schreiben kann. Es sollte vielleicht an dieser Stelle einmal in Richtung München gesagt werden: Auch dieser vorliegende Testbericht kostet Geld, nämlich für anteilige Satz-, Druck- und Papierkosten etwa soviel, wie man für eine Unidisk mit Controller im Laden bezahlen muß, von den Redaktionskosten ganz zu schweigen. Muß man sich bei dieser verqueren Marketing-Politik wundern, daß die Firma Apple seit einem Monat in der Statistik der führenden europäischen Mikrocomputer-Firmen nur noch in der Sammelrubrik „Sonstige Unternehmen" geführt wird?

1. Imagewriter II

Der Matrixdrucker Imagewriter II ist der Nachfolger zum Imagewriter, den man im nachhinein als Imagewriter I bezeichnen könnte. Das futuristische Design geht wahrscheinlich wieder auf den Schwarzwälder „Frog-Designer" zurück, der auch das Gehäuse zum Apple IIc entworfen hat. So ästhetisch das Design auch sein mag, der Imagewriter II vibriert wegen der Schräglage so stark, daß man keine Kaffeetasse mehr auf den Tisch stellen kann. Dies gilt insbesondere, wenn relativ kurze Zeilen gedruckt werden, weil der Nadelkopf dann nicht mehr bidirektional druckt, sondern quasi im freien Flug an den linken Anschlag knallt (s.u. „Kaffeetasse-Rappeltest").
Die Verwendung von perforiertem Endlospapier funktioniert tadellos. Von der Einzelblattzuführung ist hingegen abzuraten, weil sich die Blätter nur mit viel Geschick richtig einführen lassen, denn es gibt keine Vorder- und Seitenanlage, wie dies bei Druckmaschinen üblich ist.

1.1. Schriftarten

Der Imagewriter II verfügt hinsichtlich der Anzahl der benutzten Nadeln über drei Matrixdruckerschriften:
Entwurfschrift: Sie soll angeblich die Schrift mit der niedrigsten Druckqualität sein und kann über die Sequenz ESC a 1 eingestellt werden. Vgl. **Abb. 1**.

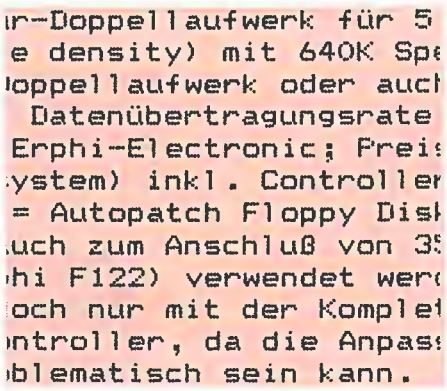


Abb. 1. Entwurfschrift (ESC a 1)

Standardschrift: Diese Schrift, die über die Sequenz ESC a 0 erzeugt wird, soll angeblich der Entwurfschrift qualitativ überlegen sein. Wenn man sich jedoch beide Schriften mit einem Fadenzähler ansieht, so liegt bei der Entwurfschrift eine größere Konturenschärfe vor. Man kann also getrost die Enwurf- statt der Standardschrift verwenden, zumal die Entwurfschrift eine höhere Druckgeschwindigkeit aufweist. Vgl. **Abb. 2**

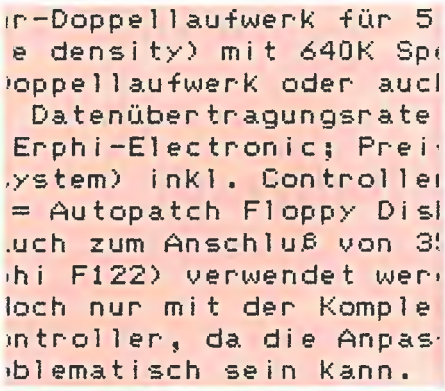


Abb. 2. Standardschrift (ESC a 0)

Schönschrift: Das Druckbild kommt bei dieser im Doppeldruck erzeugten Schrift, die über die Sequenz ESC a 2 eingestellt wird, der Qualität eines Typenrads oder Kugelkopfs sehr nahe. Allerdings ist die Druckgeschwindigkeit dann niedriger als bei einem Typenraddrucker in der Preislage des Imagewriters II, so daß ein richtiger Schönschreibdrucker vorzuziehen ist. Vgl. **Abb. 3**

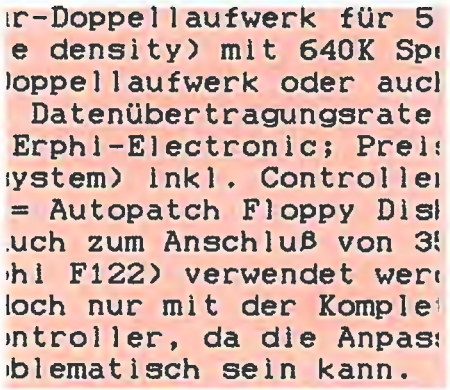


Abb. 3. Schönschrift (ESC a 2)

Innerhalb der von der Anzahl der benutzten Nadeln abhängigen Druckschriften gibt es verschiedene Schriftschnitte, die sich hinsichtlich der Laufweite, der Schräglage usw. unterscheiden. Ob die diesbezüglichen Steuerzeichen alle funktionieren, konnte aus Zeitgründen nicht ausprobiert werden.

1.2. Druckgeschwindigkeit

Soweit ich mich noch erinnere, stand in dem Handbuch, das mir nicht mehr vorliegt, eine Druckgeschwindigkeit von etwa 250 Z/s (Zeichen/s). Wie dem auch sei, bei meinen eigenen Messungen habe ich für die Entwurfschrift folgende Werte ermittelt:

– Wenn man gleiche Zeilen mit jeweils 80 „A" ohne Spaces ausgibt, werden 180 Z/s gedruckt.
– Wenn man gleiche Zeilen mit jeweils 80 „." ohne Spaces ausgibt, werden ebenfalls 180 Z/s gedruckt.
Fazit: Die Druckgeschwindigkeit wird nicht von der Anzahl der Nadeln beeinflußt, die für den Aufbau eines ASCII-Zeichens erforderlich sind. Wenn man hingegen Grafiken ausdruckt, so hängt die Druckgeschwindigkeit sehr wohl vom Grauwert der Vorlage ab, d.h. vom Verhältnis der gedeckten Fläche zur Gesamtfläche.
– Wenn man gleiche Zeilen mit einem „." am Anfang, 78 Spaces und einem „." am Ende ausgibt, werden 180 Z/s gedruckt.
– Wenn man gleiche Zeilen mit einem „." am Anfang und 79 Spaces ausgibt, werden 438 Z/s gedruckt („Kaffeetasse-Rappeltest")
– Wenn man gleiche Zeilen mit 80 Spaces ausgibt, werden 463 Z/s gedruckt.
Fazit: Eine Druckweg-Optimierung findet *zwischen* dem ersten und letzten druckbaren Zeichen einer Zeile nicht statt, dagegen sehr wohl nach dem letzten druckbaren Zeichen.

Die obigen Tests sind stark gekünstelt und deshalb nur von geringer, praktischer Aussagekraft. Deshalb wurden noch zwei praxisorientierte Drucktests durchgeführt:

1. Als typische Textverarbeitungsaufgabe wurde ein 30.000 Zeichen umfassender normaler Text im Flattersatz mit folgender Einstellung ausgedruckt:
linker Rand 5,
rechter Rand 75,
damit 70 Zeichen/Zeile,
ferner 60 Zeilen/Seite.
Nicht-proportionale Pica (Sequenz ESC N). Die Druckgeschwindigkeiten betrugen hier:
– Entwurfschrift: 145 Z/s
– Standardschrift: 112 Z/s
– Schönschrift: 27 Z/s
Fazit: Von der Schönschrift sollte man nur dann Gebrauch machen, wenn sie unbedingt erforderlich ist, z.B. für Kopiervorlagen o.ä., denn im Schönschrift-Modus ist der Imagewriter II langsamer als eine Typenrad-Schreibmaschine.

2. Als typische Dateiverwaltungsaufgabe wurde 715 Adressen (Kyan-Club-Liste) in Listenform ausgedruckt:
121 Zeichen/Zeile,
50 Zeilen/Seite.
Enge Entwurfschrift (Sequenz ESC q): Die Druckgeschwindigkeit betrug hier 182 Z/s.
Fazit: Bei Listen und Etiketten machen sich „Leerfelder" positiv bemerkbar. Da das Dateiverwaltungsprogramm die Adressen als Records „at random" von der Diskette einlesen mußte, konnte der Imagewriter II gerade noch optimal bedient werden, denn der 2K-Druckerpuffer war nach der letzten Adresse nur etwa zur Hälfte gefüllt. Bei 182 Z/s wird es also durchaus Dateiverwaltungsprogramme geben, bei denen der Imagewriter II zwischendurch vergeblich auf „Futter" wartet.

1.3. Gesamteindruck

Insgesamt hinterließ der Imagewriter II einen positiven Eindruck, wenn man einmal von der nicht unerheblichen Vibration während des Druckens absieht. Das SUPERDUMP-Programm von Jürgen Geiß funktioniert übrigens auch auf dem Imagewriter II, zumindest im Default-Modus, denn die Varianten mit Mehrfachdruck usw. habe ich aus Zeitgründen nicht mehr ausprobieren können. Dies ist insofern bedeutsam, als der Imagewriter II ohne grafikfähiges Interface geliefert wird. Beim Apple IIe muß man die Super-Serial-Card verwenden, während beim IIc und Macintosh ein Direktanschluß des Kabels möglich ist.

Abb. 4. Grafikausdruck mit SUPERDUMP

Wie ersichtlich entstehen bei dem bidirektionalen Grafikausdruck unschöne weiße Linien.

2. 800K-Unidisk

Die Unidisk ist ein 3,5-Zoll-Diskettenlaufwerk, das mit einem Controller geliefert wird, an den sich im Gegensatz zu den alten Disk-II-Laufwerken nur 1 Drive anschließen läßt. Doch kann man über ein Verbindungskabel das 1. Drive mit einem 2. Drive koppeln (Daisy-Chain-Verfahren), so daß indirekt wiederum pro Controller 2 Drives angeschlossen werden können. Die Unidisk hat eine Speicherkapazität von 800K netto oder von umgerechnet 1600 Blöcken zu je 512 Bytes.

2.1. Pseudo-ROM

Anstelle eines technischen Handbuchs wird eine Art Bilder- oder Kinderbuch geliefert, in dem nur der Anschluß der Unidisk mit entsprechenden Illustrationen beschrieben wird. Die nachfolgenden Angaben beruhen deshalb auf eigenen Untersuchungen und Mutmaßungen:
Wenn man den Apple IIe mit angeschlossener Unidisk einschaltet, so wird von der Controller-Karte offenbar ein Interrupt ausgelöst. Wenn der Controller beispielsweise in Slot 5 steckt, so wird der Bereich $C500-$C5FF mit *absoluten* Slot-Adressen aktiviert. Offenbar ist in einem Controller-ROM der Bereich $C100-$C1FF, $C200-$C2FF ... $C700-$C7FF insgesamt siebenfach mit den jeweils absoluten Slot-Adressen abgelegt. Schaut man dann im Monitor mit

C514L
nach, so findet man
LDX #$C5 ;Slot 5
während man nach
C414L
LDX #$C4; Slot 4
vorfinden würde, wenn der Controller im Slot 4 stecken würde. Halten wir also fest: Der Bereich $Cn00-$CnFF wird offenbar beim Kaltstart eingeblendet und dann ähnlich wie bei der Accelerator IIe per Softswitch zum Pseudo-ROM umfunktioniert. Dagegen hat das über „LDA $CFFF" einblendbare Expansion-ROM $C800-$CFFF stets denselben Inhalt.

2.2. Disk-Driver

Die Unidisk ist seitens der Firma Apple für ProDOS und ferner für das neue Pascal 1.3 gedacht, das uns allerdings nicht vorlag. Darüber hinaus gibt es bereits Utilities, die 1 Unidisk als 2 virtuelle Drives unter DOS 3.3 ansprechen können (vgl. „Apple Assembly Lines", Mai 1986 ff.).
Wenn man eine ProDOS-3,5-Zoll-Diskette anlegen will, so bootet man zunächst von einem Disk-II-Laufwerk eine 35-Spur-ProDOS-Systemdiskette, startet den FILER und ruft dann den Format-Befehl auf. Danach wird die eingelegte 3,5-Zoll-Sony-Leerdiskette, z.B. in Slot 4, formatiert und mit einem ProDOS-Directory versehen.
Der Disk-Driver der Unidisk, der sich im Gegensatz zum Disk-II-Driver nicht im RAM, sondern auf dem Controller selbst befindet ($Cn00-$CnFF und $C800-$CFFF), kann in der gleichen Art angesprochen werden, wie dies in „ProDOS für Aufsteiger", Bd. 1, S. 32, geschildert worden ist. Das Demo-Programm **UNIFORMAT** zeigt, wie beispielsweise eine Unidisk initialisiert werden kann. UNIFORMAT führt jedoch nur die nackte Formatierung aus und legt keine Directory-Blocks an. Es dürfte jedoch nicht schwerfallen, das Programm FORMAT.O aus „ProDOS für Aufsteiger", Bd. 2, S. 165ff., entsprechend abzuändern.

2.3. Blockorganisation

Die Blockorganisation konnte aus Zeitgründen nicht genau untersucht werden. Ich vermute jedoch, daß das gleiche Verfahren wie beim Macintosh angewandt worden ist (vgl. Peeker, 3/85, S. 48). Dies würde für jeweils die obere und untere Seite der 3,5-Zoll-Diskette bedeuten:

```
16 Spuren mit 12 Blöcken = 192 Blöcke
16 Spuren mit 11 Blöcken = 176 Blöcke
16 Spuren mit 1Ø Blöcken = 16Ø Blöcke
16 Spuren mit  9 Blöcken = 144 Blöcke
16 Spuren mit  8 Blöcken = 128 Blöcke
-----------------------------------
8Ø Spuren                800 Blöcke
```

Der Programmierer muß sich allerdings normalerweise nicht um die Blockorganisation kümmern, da der Disk-Driver die erforderlichen Umrechnungen automatisch vornimmt.

2.4. Übertragungsrate

In anderen Zeitschriften, z.B. „Incider" usw., war bereits zu lesen, daß die Unidisk eine höhere Datenübertragungsrate als das Disk-II-Drive aufweisen soll. Dies ist aufgrund meiner eigenen Untersuchungen falsch. Wenn man eine ProDOS-Unidisk bootet, so werden tatsächlich die Dateien PRODOS und BASIC.SYSTEM schneller geladen. Dies rührt jedoch daher, daß sich diese Dateien auf den äußeren Spuren befinden, die nach der obigen Tabelle 12 Blöcke pro Spur aufweisen. Bei den Spuren mit geringerer Blockanzahl sinkt die Datenübertragungsrate spürbar ab. Im einzelnen wurden folgende Werte gemessen:
– sequentieller Block-Read (0-1599): 95s, d.h. ca. 8,4K/s
– sequentieller Block-Write (0-1599): 95s, d.h. ca. 8,4K/s
– Formatieren einer Diskette (mit UNIFORMAT): 48s, d.h. ca. 17K/s

Fazit: Abgesehen von der Formatierung, die von einer ROM-Routine des Controllers vorgenommen wird, ist die Unidisk nicht schneller als ein Disk-II-Drive.

Relativ günstig, wenngleich schwer meßbar, scheinen die Spur-Wechsel-Zeiten zu sein. Zu diesem Zweck wurde eine Random-Access-Datei mit 500 Records zu je 512 Bytes angelegt. Jeder Record enthielt als ersten Eintrag eine Zufallszahl, und die gesamte Datei wurde dann mit dem Quicksort-Programm QUICK.DISK (Peeker 1/86, S. 14) sortiert. Zur Erinnerung: Ein externes Quicksort-Programm muß über weite Distanzen Blöcke lesen und schreiben, so daß hier die Spur-Wechsel-Zeiten besonders deutlich zutage treten.

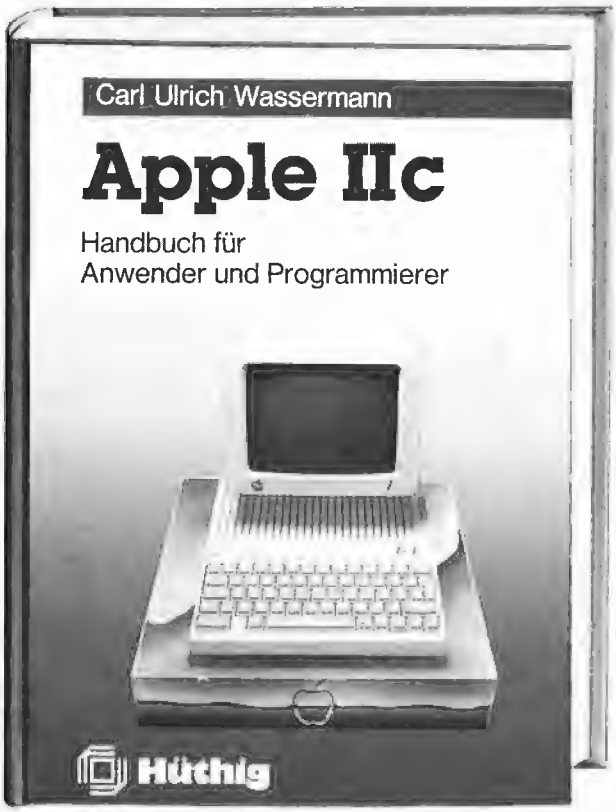
Bei der Unidisk wurde für diese „Knochenarbeit", die die Laufwerksmechanik erheblich strapaziert, eine Zeit von ca. 32 Minuten gemessen, während beispielsweise die Megaboard-MDB-Festplatte für denselben Test ca. 10 Minuten benötigte. Damit ist die MDB nur dreimal schneller als die Unidisk, was für die flotten Zugriffszeiten der Unidisk spricht.

Demoprogramm UNIFORMAT

BSAVE UNIFORMAT, A768, L122

```
                  1             ORG  $Ø3ØØ
                  2     *
                  3     * Formatbefehl für Unidisk
                  4     * ------------------------
                  5     *
                  6     * 1Ø PRINT CHR$(4)"BLOAD UNIFORMAT"
                  7     * 2Ø S = 4 : REM Slot
                  8     * 3Ø POKE 769, S: CALL 768
                  9     *
                  1Ø    IND       EQU  $CE
                  11    BEFEHL    EQU  $ØØ42
                  12    UNITNO    EQU  $ØØ43
                  13    PUFFER    EQU  $ØØ44       ;LLHH
                  14    BLOCKNO   EQU  $ØØ46       :LLHH
                  15    BELL      EQU  $FBDD
                  16    *
                  17    * Slot-Adresse poken (hier S4)
                  18    *
Ø3ØØ: A9 Ø4       19    SLOT      LDA  #$Ø4        ;3Ø1:S
Ø3Ø2: AA          2Ø              TAX
Ø3Ø3: 69 CØ       21              ADC  #$CØ        ;C4
Ø3Ø5: 85 CF       22              STA  IND+1       ;High
Ø3Ø7: 8D 72 Ø3    23              STA  ENTRY+2
Ø3ØA: A9 ØØ       24              LDA  #$ØØ        ;ØØ
Ø3ØC: 85 CE       25              STA  IND         ;Low
Ø3ØE: 8A          26              TXA              ;Ø4
Ø3ØF: ØA          27              ASL
Ø31Ø: ØA          28              ASL
Ø311: ØA          29              ASL
Ø312: ØA          3Ø              ASL
Ø313: 8D 79 Ø3    31              STA  UNIT        ;4Ø
                  32    *
                  33    * Zero-Page retten
                  34    *
Ø316: AØ Ø5       35    ZERO1     LDY  #5
Ø318: B9 42 ØØ    36    ZERO2     LDA  BEFEHL,Y
Ø31B: 99 73 Ø3    37              STA  ZER05,Y
Ø31E: 88          38              DEY
Ø31F: 1Ø F7       39              BPL  ZERO2
                  4Ø    *
                  41    * Controller im richtigen Slot?
                  42    *
Ø321: AØ ØA       43    CONTROL?  LDY  #$ØA        ;$CXØA
Ø323: B1 CE       44              LDA  (IND),Y
Ø325: C9 38       45              CMP  #$38        ;"SEC"?
Ø327: DØ Ø9       46              BNE  FEHLER
Ø329: AØ 15       47              LDY  #$15        ;$CX15
Ø32B: B1 CE       48              LDA  (IND),Y
Ø32D: CD 72 Ø3    49              CMP  ENTRY+2     ;"C4"?
Ø33Ø: FØ Ø6       5Ø              BEQ  STATUS?
                  51    *
Ø332: 2Ø 64 Ø3    52    FEHLER    JSR  ZERO3
Ø335: 4C DD FB    53              JMP  BELL
                  54    *
                  55    * $Ø64Ø = 16ØØ Blocks?
                  56    *
Ø338: A9 ØØ       57    STATUS?   LDA  #$ØØ        ;Status
Ø33A: 85 42       58              STA  BEFEHL      ;Command
Ø33C: AD 79 Ø3    59              LDA  UNIT
Ø33F: 85 43       6Ø              STA  UNITNO      ;S4,D1
Ø341: A9 ØØ       61              LDA  #$ØØ
Ø343: 85 44       62              STA  PUFFER      ;Puffer
Ø345: 85 45       63              STA  PUFFER+1
Ø347: 85 46       64              STA  BLOCKNO     ;Block
Ø349: 85 47       65              STA  BLOCKNO+1
Ø34B: 2Ø 7Ø Ø3    66              JSR  ENTRY
Ø34E: EØ 4Ø       67              CPX  #$4Ø        ;LL=4Ø
Ø35Ø: DØ EØ       68              BNE  FEHLER
Ø352: CØ Ø6       69              CPY  #$Ø6        ;HH=Ø6
Ø354: DØ DC       7Ø              BNE  FEHLER
                  71    *
                  72    * Unidisk formatieren
                  73    *
Ø356: A9 Ø3       74              LDA  #$Ø3        ;Format
Ø358: 85 42       75              STA  BEFEHL
Ø35A: AD 79 Ø3    76              LDA  UNIT
Ø35D: 85 43       77              STA  UNITNO
Ø35F: 2Ø 7Ø Ø3    78              JSR  ENTRY
Ø362: BØ CE       79              BCS  FEHLER
                  8Ø    *
                  81    * Zero-Page wieder herstellen
                  82    *
Ø364: AØ Ø5       83    ZER03     LDY  #5
Ø366: B9 73 Ø3    84    ZER04     LDA  ZER05,Y
Ø369: 99 42 ØØ    85              STA  BEFEHL,Y
Ø36C: 88          86              DEY
Ø36D: 1Ø F7       87              BPL  ZER04
Ø36F: 6Ø          88              RTS
Ø37Ø: 4C ØA C4    89    ENTRY     JMP  $C4ØA       ;gepokt
                  9Ø    *
                  91    ZER05     DS   6
Ø379: ØØ          92    UNIT      HEX  ØØ
```

FACHBUCHTIP Hüthig



Apple IIc

Handbuch für Anwender und Programmierer

von Carl-Ulrich Wassermann

1985, 324 S., zahlr. Abb., kart.,
DM 35,—
ISBN 3-7785-1157-2

Wenn Sie die Leistungsfähigkeit ihres Apple IIc bisher noch nicht ausschöpfen konnten, brauchen Sie dieses Buch.

Leicht verständlich, trotzdem ausführlich wird die Sprache Applesoft BASIC dargestellt. Eine Vielzahl von Programmen, die speziell auf den Apple IIc zugeschnitten wurden, zeigen die Wirkung der einzelnen Befehle bis zu Programmiertechniken für Diskettenzugriff, Maus und andere Anwendungen. Apple IIc Maschinensprache, Programmeingabe und -kontrolle mit Monitorbefehlen in PASCAL und FORTH geben dem Leser eine Basis für die Programmierung des Apple IIc in anderen Sprachen.

Die Konfigurierung der seriellen Ports und weitere spezifische Beschreibungen des Computers ermöglichen die gezielte Kontrolle über die Maschine.

Mehr als 90 PEEK-, POKE-und Maschinenprogrammadressen, Speicheraufteilung und weitere Tabellen sind für den Programmierer eine konzentrierte, wichtige Grundlage.

BESTELLCOUPON

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

Buchtitel

Name

Straße

Ort

Unterschrift

Ein Heim für die Maus

Mouse Trap

vorgestellt von Thomas Bühner

Wer häufig mit der Apple Maus arbeitet, kennt das Problem: Wenn man sie braucht, liegt sie meist unter einem zwanzig Zentimeter hohen Stapel aus Arbeitsunterlagen und Disketten; und sobald es gelungen ist, sie darunter hervorzuziehen, beginnt das Ganze mit Sicherheit in Richtung Schreibtischkante zu rutschen. Mit Mouse Trap hat man die Maus immer griffbereit.

Aufbau

Diese „Mausgarage" ist ein 55 mm hohes, oben offenes Kästchen, das aus 2,5 mm dickem beigem Plastik besteht. Auf einer Innenseite ist es mit einem Streifen von 7 mm starkem Schaumstoff gepolstert. Rückwärtig befindet sich eine 5 x 5 cm große Fläche, die mit Kletthaken besetzt ist. Sinnvoll wäre es gewesen, wenn der Hersteller die Vorderseite der Mouse Trap so mit einem senkrechten Schlitz versehen hätte, daß man die Maus mit nach unten gerichtetem Kabel aufbewahren könnte.

Befestigung

Mouse Trap sollte so an einer der beiden Monitorseiten angebracht werden, daß man die Maus bequem von oben hineinstecken und wieder herausnehmen kann. Ein selbstklebendes 5 x 5 cm messendes Textilstück wird mitgeliefert, in dessen Fasern die Kletthaken des Kästchens festen Halt finden. Die Mouse Trap wird nicht direkt am Monitor befestigt, damit sie ohne Probleme wieder abgenommen werden kann, wenn – etwa im Falle eines Umzugs – die Computeranlage verpackt werden muß.

Fazit

Wer ständig einen mit Arbeitspapieren überladenen Schreibtisch hat, sollte sich diese Anschaffung durch den Kopf gehen lassen – oder in den Hobbykeller hinabsteigen, selbst aus Sperrholz eine „Mausefalle" bauen und sich von den gesparten $11.00 einen gemütlichen Abend machen.

Ein-Blick

Name	Mouse Trap
Einsatz	Aufbewahrungskästchen für die Apple Maus
Gesamtwertung	****
Zweckdienlichkeit	****
Optischer Eindruck	****
Verarbeitung	*****
Preis-Leistungs-Verhältnis	***
Preis	$11.ØØ
Bezugsquelle	Raex Enterprises, Beloit, USA
Bewertungsschlüssel:	***** Hervorragend
	**** Überdurchschnittlich
	*** Akzeptabel
	** Schlecht
	* Katastrophal

Wenn Sie Fragen zu den von unserem Tester, Herrn Thomas Bühner, vorgestellten Produkten haben, so können Sie ihn jederzeit in Berlin unter der Nummer 030 / 625 26 42 anrufen.

Zeichnen mit der Maus

Maus-Grafiktablett – Mouse Tracer

getestet von Thomas Bühner

Von Zeit zu Zeit würden viele Anwender gerne auf Papier vorliegende Zeichnungen auf den Computerbildschirm übertragen, um sie als hochauflösende Grafik in eigenen Programmen zu verwenden. Mouse Tracer macht dies in beschränktem Umfang möglich.

Am einfachsten geschieht das Digitalisieren von Bildern mit einem Video-Digitizer, der die Signale einer Fernsehkamera für den Apple umwandelt. Etwas aufwendiger ist es, die Umrisse der Strukturen mit einem Grafiktablett nachzuzeichnen und das Ergebnis dieses Vorgangs mit einem Grafikprogramm – wie z.B. Mouse Paint – weiterzuverarbeiten. Diese Wege erfordern aber den Einsatz von teuerer Peripherie; beides ist unter DM 1.000 kaum zu machen.

Viele Anwender haben bereits eine Maus in ihrem Besitz, mit der sie Grafik- oder Textverarbeitungsprogramme steuern. Befestigt man einen einfachen Pappzeiger an der Maus und fährt damit die Umrisse der Strukturen nach, ist das Ergebnis wenig befriedigend: Da man die Maus nicht ständig genau im rechten Winkel zu der Vorlage halten kann, „verbiegt" sich die auf dem Bildschirm sichtbare Zeichnung so stark, daß man diese Versuche schnell aufgibt.
Brimark Innovations entwickelten mit Mouse Tracer ein Hilfsgerät, mit dem eine kontrollierte Führung der Maus möglich wird.

Aufbau

Mouse Tracer besteht aus einem 30 x 30 cm großen Sperrholzbrett, auf das ein rechtwinkliger, beweglicher Stahlarm aufgeschraubt wurde. Die Platte bekommt so Ähnlichkeiten mit dem Reißbrett eines technischen Zeichners. Am Ende des Arms befindet sich eine Aufnahmevorrichtung für die Apple Maus, die darin mit einem einfachen Haushaltsgummi befestigt wird.

Bild 1

Einsatz

Um eine Zeichnung zu digitalisieren, befestigt man sie mit Tesafilm auf der Platte. Nachdem das Grafikprogramm, mit dem man arbeitet, auf „fortlaufend freihandzeichnen" eingestellt wurde, fährt man die Umrisse der gewünschten Strukturen mit der Spitze des Stahlzeigers langsam nach, der im **Bild 1** auf den rechten Ellbogen des Malers zeigt.

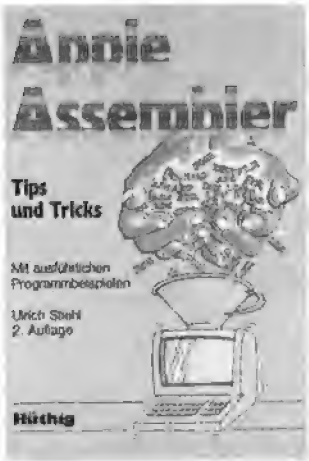
Da mit der Apple Rollmaus kein sehr exaktes Arbeiten möglich ist, wird die Strichführung auf dem Bildschirm nach einiger Zeit nicht mehr genau mit den Umrissen des Originals übereinstimmen. Die Maus muß dann angehoben und der Zeichenstift auf dem Monitor entsprechend verstellt werden, bevor man fortfährt. Auf diese Weise

Verlangen Sie mehr

Computerbücher aus dem Fachbuchverlag Hüthig:
Präzise Informationen, aktuelle Themen,
gut lesbarer gesetzter Text.

Arne Schäpers
ProDOS-Analyse
Versionen 1.0.1, 1.0.2, 1.1.1

Arne Schäpers
ProDOS-Analyse
Versionen 1.0.1, 1.0.2, 1.1.1
1985, 470 S., kart., DM 68,—
ISBN 3-7785-1134-3
Dies ist die umfangreichste und detaillierteste Darstellung, die jemals ein Apple-Betriebssystem erfahren hat. So etwas gibt es nicht einmal in Amerika!



Ulrich Stiehl
Apple Assembler
1984, 200 S., 3 Abb., kart., DM 34,—
ISBN 3-7785-1047-9
Alle wichtigen ROM-Routinen sowie eine Vielzahl von Hilfsprogrammen werden in diesem Buch für den Programmierer mit Anfängerkenntnissen vorgestellt. Insgesamt über 40 Utilities mit mehreren völlig neuartigen Programmen sind gelistet.

Arne Schäpers
Bewegte Apple-Grafik
DOS Toolkit-Erweiterungen

Arne Schäpers
Bewegte Apple-Grafik
1985, 305 S., 6 Abb., kart., DM 58,—
ISBN 3-7785-1150-5
Ein Kursus für alle, die auf dem Apple hochaufgelöste und bewegte Grafiken in Maschinensprache programmieren wollen. Schrittweise wird ein Arcade-Spiel entworfen, das käuflichen Action-Spielen in der meisterhaften Grafik als Vorbild dienen kann.

Ulrich Stiehl
Apple DOS 3.3
Tips und Tricks
2., durchges. Auflage 1984, 216 S.,
kart., DM 28,—
ISBN 3-7785-1049-5
Das Standardwerk zum DOS 3.3 sowohl für BASIC- als auch für Assemblerprogrammierer. Enthüllt einige bislang noch niemals publizierte Techniken und viele Tricks aus der langjährigen Praxis des Autors. Dieses Buch ist der unentbehrliche Begleiter für jeden Apple-Programmierer.

Frank Bühler
Applesoft BASIC

Frank Bühler
Applesoft BASIC
Tips und Tricks
1985, 241 S., 40 Abb., kart., DM 38,—
ISBN 3-7785-1094-0
Das Buch enthält eine komplette Beschreibung aller möglichen Applesoft-Befehle und zeigt an einem Beispiel die erforderliche Syntax auf. Ausgearbeitete Unterroutinen können leicht in eigene Programme übernommen werden.

Ulrich Stiehl
ProDOS für Aufsteiger
Band 1
2., geänderte Auflage 1985, 208 S.,
kart., DM 28,—
ISBN 3-7785-1098-3
Das Schwergewicht des ersten Bandes liegt auf der Darstellung der Pro-DOS-internen Systemadressen und der Assemblerprogrammierung unter diesem neuen Betriebssystem. Das Werk enthält auf über 70 Seiten eigens hierfür entwickelte Programme.

Jürgen Kehrel
Apple-Assembler lernen
Band 1: Einführung in die Assembler-Programmierung

Jürgen Kehrel
Apple-Assembler lernen
Band 1: Einführung in die Assembler-Programmierung des 6502
1985, 235 S., kart., DM 38,—
Noch nie war es so einfach, die Assembler-Programmierung auf dem Apple zu erlernen, wobei auch der neue 65C02-Prozessor behandelt wird.

Carl-Ulrich Wassermann
Apple IIc
Handbuch für Anwender und Programmierer

Carl-Ulrich Wassermann
Apple IIc
Handbuch für Anwender
und Programmierer
1985, 324 S., kart., DM 35 —
ISBN 3-7785-1157-2
Ein Handbuch für alle, die die besonderen Eigenschaften der Apple IIc voll ausschöpfen wollen.

Benutzen Sie die Bestellkarte im PEEKER!

Weitere Titel und Informationen finden Sie in unserem Computerbuch-Katalog:
Dr. Alfred Hüthig Verlag, Postfach 10 28 69, 6900 Heidelberg 1

Hüthig

vermindert man größere horizontale oder vertikale Abweichungen.
Zur Feststellung der Genauigkeit wurde folgender Versuch unternommen: Mit Hilfe des Fingerprint Plus Interface (Testbericht in Peeker 4/86, Seite 66 f.) wurde das Bild ausgedruckt, das sich dem Anwender beim bekannten Grafikprogramm Mouse Paint bietet. Es diente als 65 x 90 mm große Vorlage für den Genauigkeitstest. Alle Ränder der Mouse Paint „Konsole" wurden – ohne abzusetzen – mit dem Mouse Tracer nachgezeichnet. Das Ergebnis sieht man in **Bild 2**.
Wenn man den Strukturen ohne ein Zittern der Hand folgen könnte und die Maus exakt arbeiten würde, dann wäre die innere Skizze ein genaues, verkleinertes Abbild der äußeren „Konsole". Wie man erkennt, ist ein recht aufwendiges Überarbeiten der Grafik nötig, wenn ein befriedigendes Endprodukt erreicht werden soll. Im allgemeinen sind die Strukturen natürlich weniger regelmäßig als im abgebildeten Beispiel.

Nacharbeiten

Sobald die groben Umrisse eingegeben wurden, wird man sich normalerweise daranmachen, die Ränder zu glätten und dem Bild insgesamt ein gefälligeres Aussehen zu geben. Diese Arbeit wird am besten mit einem leistungsfähigen Punkt-Grafik-Editor vorgenommen, wie er z.B. bei dem Trickfilmprogramm „Take 1" mitgeliefert wird (ein Testbericht darüber folgt).

Fazit

Die mit Hilfe von Mouse Tracer digitalisierten Strukturen sind im allgemeinen eine brauchbare Ausgangsbasis für ein Bild. Die Genauigkeit dieses Maus-Grafiktabletts ist vergleichbar mit der des „Koala Pad", das zwar zusätzlich als Joystick-Ersatz Verwendung finden kann, dafür aber auch dreimal so teuer ist.

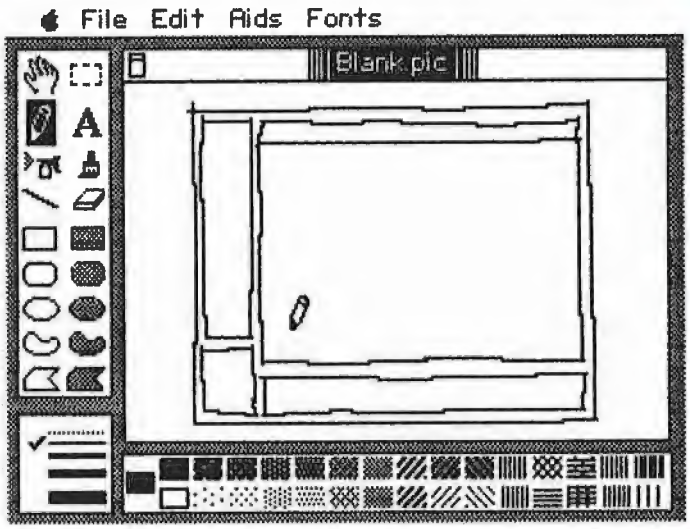


Bild 2

Ein-Blick

Name	Mouse Tracer
Notwendige Ausstattung	Apple II, Apple Maus, Monitor
Einsatz	Nachzeichnen der Umrisse von Strukturen mit Hilfe der Maus
Gesamtwertung	****
Genauigkeit	***
Optischer Eindruck	***
Verarbeitung	****
Allgemeine Bedienbarkeit	****
Preis–Leistungs–Verhältnis	****
Preis	$ 35
Bezugsquelle	Brimark Innovations, Northridge, USA

Bewertungsschlüssel:

***** Hervorragend
**** Überdurchschnittlich
*** Akzeptabel
** Schlecht
* Katastrophal

Grafik zu Papier bringen

Hires-Grafik-Druckprogramme

Imageprinter II, Printographer und Zoom Grafix

getestet von Thomas Bühner

Obwohl die sog. „doppelt hochauflösende Grafik" von einigen neueren Programmen eingesetzt wird, kommt der herkömmlichen HGR-Grafik – mit sechs Farben und 280 x 192 Rasterpunkten – bei den Computern der Apple-II-Serie immer noch die weitaus größte Bedeutung zu.

1. Problem

Die meisten Software-Pakete, mit deren Hilfe man Diagramme oder Zeichnungen auf dem Bildschirm entwerfen kann, unterstützen nur wenige Drucker und Interfaces. Das Programm Mouse Paint etwa, das im Kaufpreis der Apple Maus inbegriffen ist, arbeitet nur mit dem Imagewriter zusammen. Viele Anwender sind daher in der wenig beneidenswerten Situation, die gezeichneten Bilder nicht ausdrucken zu können.

2. Alternativen

Interfaces wie Print-It oder Fingerprint (Testbericht in Peeker 4/86, Seite 66 f.), die den Inhalt des Bildschirms direkt an den Drucker ausgeben können, sind verhältnismäßig teuer. Möchte man weniger Geld ausgeben, empfiehlt sich die Anschaffung von Software, die bereits auf Diskette gespeicherte Bilder mit einer Vielzahl von Drucker- und Interface-Modellen zu Papier bringt. In Peeker 11/85, Seite 66 f. wurde bereits das sehr flexible Triple Dump kurz vorgestellt; hier folgt ein vergleichender Leistungstest dreier weiterer Druckprogramme: Imageprinter II, The Printographer und Zoom Grafix.

3. Hardware

Alle drei Programme arbeiten mit den Apple-Modellen II+, IIe und IIc. Zur nötigen Geräteausstattung gehören ein Farb- oder Schwarz-Weiß-Monitor, ein Diskettenlaufwerk und ein Drucker. Da die Hersteller in regelmäßigen Abständen neue Programmversionen veröffentlichen, die mit weiteren Interfaces und Druckern zusammenarbeiten, wäre eine entsprechende Liste vermutlich bereits zum Zeitpunkt der Veröffentlichung dieses Artikels überholt. Im allgemeinen läßt sich jedoch sagen, daß diejenige Hardware unterstützt wird, die sich gut verkauft. Die Drucker der Firmen Apple, Epson und Itoh gehören ebenso dazu wie die Interfaces von Apple und Orange Micro.

4. Arbeitsablauf

Das Ausgangsmaterial für den Druck sind HGR-Bilder, die bereits auf einer DOS 3.3 Diskette abgespeichert wurden. Im Catalog kann man diese Dateien daran erkennen, daß sie vom Typ „B" (binär) sind und eine Größe von 33 oder 34 Sektoren haben. Das Programm lädt zunächst ein Bild von der Diskette in den Arbeitsspeicher und präsentiert es auf dem Monitor. Bei Imageprinter II und The Printographer kann man es nun modifizieren und die veränderte Grafik eventuell wieder abspeichern.
Danach wird festgelegt, ob das ganze Bild oder nur ein Ausschnitt gedruckt werden soll. Weiterhin bestimmt man, ob ein Punkt, der auf dem Bildschirm weiß ist, auf dem Papier als schwarze („normaler Druck") oder weiße Stelle („inverser Druck") erscheinen wird.

Peeker-Sammeldisketten #18, #19 und #20

Disk #18

(Kombinierte DOS-3.3- und UCSD-Diskette; Heft 6/1986; Einzelpreis DM 28,-; Fortsetzungspreis DM 20,-) Achtung: Solange Diskette schreibgeschützt bleibt, kann sie sowohl unter DOS 3.3 als auch unter UCSD-Apple-Pascal 1.1/1.2 benutzt werden.
(1) = Zweck; (2) = Heft/Seitenzahl; (3) = Gerätekonfiguration; (4) = Betriebssystem; (5) = Programmstart; (6) = Sonstiges

UCSD-Teil

CALLDUMP.TEXT
CALLDUMP2.TEXT
SUPERDUMP.TEXT
SUPERDUMP.LIB
EPSON.TEXT
EPSON.CODE
IMAGEWRITR.TEXT
IMAGEWRITR.CODE
LIB.TEXT
FUELLER

(1) Druckprogramm (analog zu dem SUPERDUMP-Programm unter DOS 3.3) für einfache und doppelte Hires-Grafik unter Apple-Pascal 1.1 und 1.2 für wahlweise Imagewriter I oder Epson FX-80; (2) Heft 6/86, S. 40; (3) Apple II+; für Double-Hires IIc oder IIe mit 64K-Karte; (4) Apple Pascal 1.1 oder 1.2 (64K- oder 128K-Version); (5) E(xec EPSON.CODE oder IMAGEWRITR. CODE (nach Einbindung in Library; siehe Heft!)

DOS-3.3-Teil

EDIT
(1) Diskette neditor für ProDOS; Quellcode befindet sich auf Sammeldisk #17; (2) Heft 6/86, S. 24 (ausführliche Übungsbeispiele; siehe auch Hefte 5/86, 7/86 und 8/86); (3) Apple II+/e/c mit und ohne 80-Zeichenkarte; (4) ProDOS 1.0.1, 1.0.2, 1.1.1; (5) Objektcode EDIT muß zunächst mit CONVERT oder DOSTOPRO auf ihre ProDOS-Arbeitsdiskette konvertiert werden. Dann Start mit BRUN EDIT.

MDB.KOPY.SPEZIAL
MDB.KOPY
T.MDB.KOPY.OBJ
MDB.KOPY.OBJ

(1) Backup-Programm für die MDB-Megaboard-Festplatte unter DOS 3.3; nach Änderungen im Programm ggf. auch bei anderen Festplatten verwendbar, deren DOS-Volumes 140K umfassen; ferner als 140K-Kopierprogramm für 2-Drive-Besitzer benutzbar. (2) Heft 5/86, S. 61; (3) Apple IIe/+, eine 140K-Disk-II und eine MDB10-Festplatte; (4) DOS 3.3; (5) RUN MDB.KOPY (Warnung: Nicht starten, wenn keine Festplatte existiert!); (6) Die Spezialversion MDB.KOPY.SPEZIAL kann an die Anzahl der vorhandenen MDB-Volumes angepaßt werden.

PLOT.3.PRO
PLOT.3.E
PLOT.BX
T.PLOT.BX
PLOT.HELP.1 bis PLOT. Help.5

PLOT.PATCH
SUPERDUMP.PATCH

(1) Funktionsplotprogramm für IIe/c mit Double-Hires (Besitzer des Apple II+ nehmen bitte die Version PLOT.2.0 von Sammeldisk #5!); (2) Heft 6/86, S. 6; (3) IIc oder IIe mit 64K-Karte; (4) DOS 3.3; (5) EXEC PLOT.3.PRO; (6) Zum Ausdrucken der Funktionskurven kann SUPERDUMP (Sammeldisk #5) verwendet werden.

PRODOS.BACKUP
T.PRO.BACKUP.O
PRO.BACKUP.O
KOPY.160.SPUR
T.KOPY.160.SPUR.O
KOPY.160.SPUR.O

(1) Backup-Programm PRODOS. BACKUP für MDB-Megaboard-Festplatte und 160-Spur-Erphi-Laufwerk oder 140K-Laufwerk unter ProDOS (im Ggs. zum obigen MDB.KOPY, das für DOS 3.3 gedacht ist); nach Änderungen im Programm ggf. auch bei anderen Festplatten verwendbar; ferner ist Programm KOPY.160. SPUR als 160-Spur-Erphi-Kopierprogramm (ohne Formatbefehl!) verwendbar; (2) Heft 6/86, S. 30; (3) Wie bei MDB.KOPY, ferner für PRODOS. BACKUP ggf. 160-Spur-Erphi-Drives; (4) ProDOS, alle Versionen; (5) Zunächst müssen die Programme mit CONVERT oder DOSTOPRO auf Ihre ProDOS-Arbeitsdiskette kopiert werden; danach RUN PRODOS. BACKUP oder RUN KOPY.160. SPUR (Warnung: PRODOS.BACKUP nicht starten, wenn keine Festplatte existiert!)

Disk #19

(DOS-3.3-Diskette; Heft 7 1986; Einzelpreis DM 28,–; Fortsetzungspreis DM 20,–)

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

A 002 STIEHL
A 003 HELLO
B 006 DDMOVER
B 050 REGISTER.UTILITIES
B 024 REGISTER.RUNTIME
B 009 REGISTERSTARTER. OBJ
B 030 REGISTERREDIGIERER. OBJ
B 018 REGISTERSORTIERER. OBJ
B 019 REGISTERMISCHER. OBJ
B 020 REGISTERDRUCKER. OBJ
B 018 REGISTERUMWANDLER. OBJ
A 007 REGISTERTESTER
A 008 REGISTERUMDREHER
A 007 REGISTERTEILER

(1) Programmpaket zum Erstellen, Sortieren, Ausdrucken und Pflegen von Buchregistern (= Seitenregistern) und (zweisprachigen) Glossaren; (2) Heft 7/86, S. 6; (3) Apple II+/e/c mit 2 Diskettenlaufwerken (Disk II oder Erphi-160-Spur-Drive); auch 256K-RAM-Disk oder MDB-Festplatte verwendbar; beliebiger Slot, doch müssen beide Drives an demselben Slot angeschlossen sein; (4) Diversi-DOS 2C oder ersatzweise mit Einschränkungen DOS-3.3; (5) RUN STIEHL von Drive 1; zuvor formatierte Leerdiskette in Drive 2 einlegen.

A 005 MOUSORY
A 013 MOUSORY. DEMO
A 019 MOUSORY. GAME
B 002 MOUSORY. BELL
B 024 MOUSORY. SET1
B 022 MOUSORY. SET2
B 026 MOUSORY. SET3
B 034 MOUSORY.BILD

(1) Memory-Spiel mit Hires-Spielkarten; (2) Heft 7/86, S. 12; (3) Apple II+/e/c; Apple-Maus optional, d. h. auch Cursortasten verwendbar; (4) DOS 3.3 oder ProDOS; (5) RUN MOUSORY

A 002 START
A 005 HP
A 003 UP1
A 003 UP2
T 007 T.RUN.FILE
B 002 RUN.FILE

(1) Demo-Programm für Chain -Routine (= Übergabe von Variablen an das nächste Programm-Modul) unter Applesoft; (2) Heft 7/86, S. 18; (3) Apple II+/e/c; (4) DOS 3.3; aus verschiedenen Gründen (HIMEM u. a.) nicht lauffähig unter ProDOS!; (5) RUN START

A 015 CLOCK
T 020 CLOCK.EXEC

(1) Anpassungsprogramm für Hardware-Uhr TIME II; (2) Heft 7/86, S. 22; (3) Apple II+/e mit TIME-II-Uhrenkarte; (4) ProDOS (nur unter Version 1.0.1 getestet); (5) RUN CLOCK oder vorher Original-START-UP laden und dann EXEC CLOCK. EXEC; (6) Programme müssen zuvor von der Sammeldisk mit CONVERT oder DOSTOPRO auf Ihre ProDOS-Arbeitsdiskette konvertiert werden; Programme nur starten, wenn TIME II im Rechner installiert ist!

A 006 PRODOS.LIB.DEMO
T 074 T.PRODOS.LIB
B 009 PRODOS.LIB

(1) Muster-ProDOS-Runtime-Library für Stand-alone-Programme (Assembler, Applesoft oder Kyan-pascal), die aus speicher- oder programmtechnischen Gründen auf das BASIC.SYSTEM verzichten müssen; (2) Heft 7/86, S. 40; (3) Apple II+/e/c; (4) ProDOS (jede Version); (5) RUN PRODOS.LIB. DEMO; (6) Programme müssen zuvor von der Sammeldisk mit CONVERT oder DOSTOPRO auf Ihre ProDOS-Arbeitsdiskette konvertiert werden.

Disk#20

(UCSD-Apple-128K-Pascal-1.2-Diskette; Heft 8/1986; Achtung: Umfangreiches Softwarepaket, deshalb Einzelpreis DM 48,–; Fortsetzungspreis DM 38,–; (wird nicht automatisch an Fortsetzungsbezieher verschickt)

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

PICEDIT:
SYSTEM.MISCINFO
SYSTEM.CHARSET
ASCII.FONT
GERMAN.FONT
MATH.FONT
GREECE.FONT
SUPER.SUB.FONT
MENU.GRAF
PRINTER.INFO
EPSON
IMAGEWRITER
SYSTEM.ATTACH
CRUNCHER.CODE
SYSTEM.STARTUP
SYSTEM.LIBRARY
ARROWS.KEYS
DESIGNER.CODE

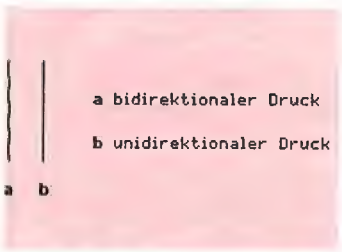
(1) PIC-EDIT von Jürgen Geiß ist das Gegenstück zum Turtle-Graphics-Library-Paket, das bekanntlich über den Hüthig Software Service für DM 98,– erhältlich ist. PIC-EDIT ist ein universeller Grafik-Editor, der dem Macpaint-Programm nachempfunden ist und über ungewöhnlich leistungsfähige Befehle verfügt; (2) Heft 8/86, S. 6; (3) Apple IIc oder IIe mit erweiterter 80-Zeichenkarte; (4) UCSD-Apple-Pascal 1.2; (5) Auf der Sammeldiskette #20 befinden sich die obenstehenden Dateien außer SYSTEM.APPLE und SYSTEM.PASCAL, die Sie aus urheberrechtlichen Gründen selbst auf hierfür reservierte Dummy-Files kopieren müssen. Dabei muß es sich um 128K-Pascal-1.2-Files handeln.

Die Diskette läuft sofort mit Epson-Druckern. Für den Imagewriter muß im Filer die Datei IMAGEWRITER mit den Transfer-Befehl auf PRINTER. INFO 1 kopiert werden.

Nach dem Kopieren der zwei Pascal-Systemdateien kann die Diskette direkt gebootet werden.

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

Läßt man die Grafik beim Ausdruck um 90 Grad drehen, kann für die Bilder ein größeres Format gewählt werden.
Preiswertere Matrixdrucker, die oft nicht sehr belastbar sind, können „heißlaufen", wenn im Bild viele schwarze Flächen enthalten sind; um dem vorzubeugen, kann man das Programm anweisen, am Ende jeder Druckzeile eine kurze Pause zu machen. Die Nadeln des Druckkopfes kühlen dann schnell wieder ab.
Vergrößert man das ganze Bild um das Vier- oder Fünffache, paßt es nicht mehr auf DIN A4 große Blätter. Es kann zu Störungen im Druckbetrieb kommen, wenn die Software diese Übergröße nicht erkennt und das Gesamtformat entsprechend beschneidet.
Wenn man Text zu Papier bringt, sparen die meisten Drucker Zeit, indem eine Zeile von links nach rechts und die folgende von rechts nach links gedruckt wird. Dieses „bidirektionale" Drucken führt bei grafischen Darstellungen aber zu Unsauberkeiten (siehe **Abb.**), daher läßt es sich meist per Druckerbefehl abstellen. Von den drei getesteten Programmen kann jedoch nur Zoom Grafix diese Option einschalten. Bei Imageprinter II und The Printographer muß der entsprechende Druckerbefehl vor dem Booten per Hand eingegeben werden.
Außer den bisher geschilderten Eigenschaften stehen bei jedem der drei Programme zusätzliche Wahlmöglichkeiten zur Verfügung:



5. Imageprinter II

Besitzt man ein Interface, das nicht vom Imageprinter II unterstützt wird, kann man sich selbst ein kleines Maschinenprogramm schreiben, das die Kommunikation zwischen Software und Interface abwickelt. Eine ausführliche Anleitung dafür befindet sich im Imageprinter-Handbuch. Voraussetzung für ein Gelingen ist jedoch, daß im Handbuch des Interface die notwendigen Informationen nachgeschlagen werden können.

Man kann das Bild, das gedruckt werden soll, an beliebigen Stellen mit Text versehen. Es steht nur eine Typenform zur Verfügung; zusätzlich sind eine Reihe mathematischer Symbole vorhanden. Um für die Beschriftung genügend freien Platz zu schaffen, kann das Bild nach links, rechts, oben oder unten verschoben werden. Die Buchstaben liegen auf der Diskette als Applesoft Shapes vor; der Zeichenvorrat ist also erweiterbar, wenn man ein leistungsfähiges Shape-Editor-Programm besitzt.
Mit Hilfe einiger Sonderoptionen kann das Aussehen des Bildes verändert werden: Es ist möglich, einen beliebigen Teil der Grafik mit einem rechteckigen Rahmen zu umgeben, oder ihn mit einer der Apple-HGR-Farben zu füllen. Das Löschen von Bildteilen wird bewerkstelligt, indem man sie mit Schwarz überdeckt. Ist die ausgedruckte Grafik für einen bestimmten Anwendungszweck zu groß, läßt man das Bild auf ein Viertel der originalen Größe schrumpfen. Das Ergebnis all dieser Modifikationen kann auf Diskette abgespeichert werden.
Für EDV-Neulinge befindet sich im Handbuch ein kurzer Abschnitt, in dem auf die Speicherverteilung der Grafik eingegangen wird. Auch werden Ratschläge gegeben, wie man Bilder aus kopiergeschützten Programmen auf Diskette abspeichern kann.
Die Druckroutinen von Imageprinter II können auch in selbstgeschriebenen Applesoft-Programmen verwendet werden. Da der Aufruf der Funktionen jedoch über eine CALL-Parameterliste erfolgt (CALL IMGPRT,A%,B%,C%), ist ein späteres Compilieren nicht möglich.

6. The Printographer

Ebenso wie bei Imageprinter II kann man selbst ein Interface-Treiberprogramm schreiben, wenn es nötig sein sollte. Zusätzlich bietet das Programm die Option, Matrixdrucker anzusteuern, die nicht serienmäßig unterstützt werden. Zu diesem Zweck gibt man allerlei technische Daten ein, die man dem Druckerhandbuch entnimmt. Auf diese Weise arbeitet The Printographer auch mit „Exoten".
Bei der Bestimmung des zu drukkenden Bildausschnitts zeigt nicht – wie bei den anderen Programmen – ein Strichfenster die getroffene Wahl an; stattdessen wird der nicht zum Druck bestimmte Teil vom Bildschirm gelöscht. Da nicht nur eine rechtwinklige Begrenzung möglich ist, sondern auch Ovale und Rauten zur Verfügung stehen, kann ein Bild so mit einem gefälligen Rahmen versehen werden.
Die Buchstaben, mit denen man an beliebiger Stelle einen Text auf dem Bild unterbringen kann, liegen im DOS Toolkit-Format vor. Sie können also mit der Editier-Software, die diesen Standard verwendet (wie z.B. Higher Text), ohne Mühe verändert werden.
Als einziges der drei getesteten Programme überprüft The Printographer *nicht*, ob das Gesamtbild auf das Papier paßt. Bei einer Vergrößerung kann es daher bei manchen Druckern zu Problemen kommen.
Falls zwei Bilder geschaffen wurden, die (horizontal aneinandergelegt) zusammenpassen, kann man sie nebeneinander ausdrucken.
Manche preiswerten Matrixdrucker werden von diesem Programm stark belastet, da hier am Ende einer Zeile keine Druckpause eingelegt werden kann. Wenn viel Schwarz im Bild vorkommt, können die Drucknadeln unter Umständen sehr heiß werden.
Die Einbettung der Routinen von The Printographer in Basic-Programme erfolgt ebenso wie bei Imageprinter II. Zusätzlich besteht jedoch die Möglichkeit, die Parameterübergabe mit POKEs zu bewerkstelligen (POKE A,A% : POKE B,B% : POKE C,C% : CALL PRTGPH), so daß die Applesoft-Programme compiliert werden können.
Lores-Grafiken, die mit einer Auflösung von 40 x 48 Punkten gezeichnet wurden, kann man mit The Printographer in Hires-Bilder verwandeln und anschließend abspeichern oder ausdrucken.
Eine sehr nützliche Option ist die Umwandlung von Grafik in Textdateien. Die entstehenden Daten können in Dokumente aufgenommen werden, die mit Textverarbeitungsprogrammen erstellt wurden. Somit ist es möglich, beim Ausdruck eines Schriftstücks gleich die dazugehörigen Bilder zu verwerten.

7. Zoom Grafix

„Exotische" Drucker oder Interfaces arbeiten mit diesem Programm nicht, da die Diskette kopiergeschützt ist und der Anwender daher keine weiteren Treiberprogramme hinzufügen kann. Wieder einmal stellt sich Kopierschutz als anwenderfeindlich heraus. Um die allgemeine Verbreitung seines Werks zu verhindern, ermöglicht der Hersteller auch nicht die Benutzung der Zoom Grafix-Routinen in den selbstgeschriebenen Programmen des Benutzers.
Das Handbuch ist nur sehr knapp bemessen, dennoch sind einige wichtige Hinweise für Anfänger enthalten, die das Abspeichern von Bildern aus kopiergeschützten Programmen betreffen.
Nur bei Zoom Grafix kann der horizontale Vergrößerungsfaktor unabhängig vom vertikalen eingestellt werden, so daß man (entsprechend verzerrte) breite oder hohe Bilder drucken kann.
Auch bieten die beiden anderen Programme nicht die Möglichkeit, dem Drucker zu signalisieren, daß unidirektional – also nur von links nach rechts – gedruckt werden soll. Bei der anfänglichen Festlegung von Drucker- und Interface-Modell muß man sich bei Zoom Grafix außerdem dafür entscheiden, ob man uni- oder bidirektionalen Druck wünscht; während des Programmbetriebs kann dieser Punkt nicht geändert werden.

Fazit

Obwohl Zoom Grafix ein im Grunde leistungsfähiges Programm ist, kann ein Kauf nicht empfohlen werden. Da man die Diskette nicht kopieren kann, ist jederzeit die Möglichkeit gegeben, daß sie aus irgendwelchen widrigen Umständen die Funktion einstellt. Für Basic-Programmierer ist Zoom Grafix uninteressant, da man die Druckroutinen nicht in eigenen Programmen verwenden kann. Bei diesen Nachteilen wird man eher zu der Software anderer Hersteller greifen, selbst wenn man dann den Unidirektionaldruck vor dem Start des Programms von Hand (per Tastatur) einstellen muß.
Imageprinter II und The Printographer haben beide ihre Vor- und Nachteile, deren Bedeutung jeder Anwender individuell beurteilen wird. Für den Autor ist es z.B. unerheblich, ob während des Ausdrucks Pausen eingelegt werden können, da er über einen robusten Itoh-Drucker verfügt. Der Besitzer eines 600-Mark-Matrixdruckers dagegen wird diesem Punkt wesentlich mehr Beachtung schenken. Übrigens: Wie man sieht, wurde das schon im November getestete Triple Dump in die Tabelle aufgenommen, um einen direkten Vergleich zu erleichtern.

Ein-Blick

Name	Imageprinter II	The Printographer	Zoom Grafix	Triple Dump
Zahl der unterstützten Drucker	über 2Ø	über 6Ø	über 2Ø	über 3Ø
Anpassung an weitere Drucker	–	einfach	–	–
Zahl der unterstützten Interfaces	über 2Ø	über 4Ø	über 3Ø	über 5Ø
Anpassung an weitere Interfaces	möglich	möglich	–	–
Dokumentation	***** (36 Seiten)	***** (53 Seiten)	*** (12 Seiten)	***** (37 Seiten)
Umfang (in Diskettenseiten)	1	2	1	4
Betriebssystem	DOS	DOS	DOS	ProDOS & DOS
Allgemeine Bedienung	*****	*****	*****	*****
Mitgelieferte Bilder	9	5	11	18
Form der Bildausschnitte	Rechteck	Rechteck, Oval, Raute	Rechteck	Rechteck
Bild mit Text versehen	ja, 1 Typenform und math. Symbole	ja, 21 Typenformen (DOS Toolkit Format)	–	ja, 1 Typenform
Modifiziertes Bild abspeichern	ja	ja	–	–
Automatisches Beschneiden bei Übergröße	ja	–	ja	ja
Normal / invers drucken	ja	ja	ja	ja
Vergrößert drucken	2- bis 6-fach	2- bis 99-fach	2- bis 99-fach	2- bis 255-fach
Unterschiedliche horizontale und vertikale Vergrößerungsfaktoren	–	–	ja	ja
Horizontal / vertikal drucken	ja	ja	ja	ja
Horizontale Bildposition auf Papier	beliebig	beliebig	beliebig	beliebig
HGR1 und HGR2 nebeneinander drucken	–	ja	–	–
Druckpause nach jeder Zeile	Ø bis 12 Sekunden	–	1/4 bis 6Ø Sekunden	1/1Ø bis 25 Sekunden
Unidirektionaler Druck eingebaut	–	–	ja	–
Tips für das Abspeichern von Bildern kopiergeschützter Programme	***	–	****	*****
Einbettung in eigene Programme	nur für Applesoft	ja	–	ja
Besonderheiten	Bildausschnitt rahmen oder mit Farbe füllen, Linien zeichnen, Bild verschieben oder auf 1/4 schrumpfen	Bildausschnitt als Textdatei abspeichern für Drucken mit Textverarbeitungsprogramm, Lores Bilder drucken	–	Drucken von Lores, Double Lores, Text, 8Ø-Zeichen-Text, Double Hires
Kopierbar	ja	ja	–	ja
Gesamtwertung	****	****	***	*****
Preis/Leistungs-Verhältnis	***	***	**	*****
Preis	$ 5Ø	$ 4Ø	$ 5Ø	$ 4Ø
Bezugsquelle	Sensible Software Birmingham, USA	Roger Wagner Publishing Santee, USA	American Eagle Lincolnwood, USA	Beagle Brothers San Diego, USA

Bewertungsschlüssel:
***** Hervorragend
**** Überdurchschnittlich
*** Akzeptabel
** Schlecht
* Katastrophal

Abenteuer in Green-Sky

Adventure-Spiel
Below the Root

getestet von Thomas Bühner

Vorgeschichte

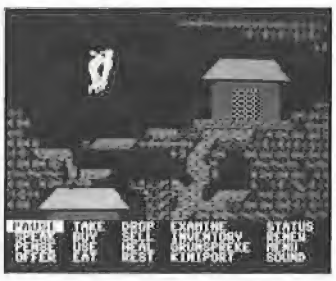
Green-Sky ist ein Landstrich, in dem das Gras noch saftig grün und die Borke der riesenhaften Bäume kräftig braun ist. (Zumindest, wenn man einen Farbmonitor sein eigen nennt.) Zwei Völker leben dort seit langer Zeit in friedlichem Einvernehmen: Die Erdlinge in einem jahrhundertealten Höhlensystem, und ihre Verwandten, die Kindar, in den Kronen der oft mehr als hundert Meter hohen Bäume.
Vor kurzem kamen zwar Spannungen zwischen den beiden Gruppen auf, doch Raamo, ein junger Kindar, verstand es zu vermitteln, und so herrscht nun ein Verhältnis vorsichtigen Abwartens zwischen Erdlingen und Kindar. In beiden Völkern haben sich Gruppierungen gebildet, deren Mitglieder es gerne sähen, wenn die Auseinandersetzungen neu aufflammen würden. Eines Tages nun verschwindet Raamo spurlos; man flüstert hinter vorgehaltener Hand, einer der Geheimbunde habe ihn entführt und getötet. Ohne seinen persönlichen Einsatz droht jetzt endgültig ein Krieg zwischen den Völkern auszubrechen.

Ausgangssituation

Fünf junge Freunde – zwei Erdlinge und drei Kindar – haben sich vorgenommen, alles daranzusetzen, Raamos Schicksal aufzuklären. Als sie eben losziehen wollen, um nach Hinweisen für seinen Aufenthaltsort zu suchen, erreicht sie eine Botschaft des alten Weisen Falla: Wer Raamo finden will, muß alle Gefahren des Weges allein auf sich nehmen. Die Fünf beraten nun, wer von ihnen gehen soll.
An diesem Punkt beginnt das Spiel. Zunächst muß die Entscheidung getroffen werden, in welche der fünf Rollen man schlüpfen will. Da die geistigen und körperlichen Fähigkeiten der jungen Leute recht unterschiedlich sind, gilt es, klug zu wählen. Wer noch nie ein Computer-Abenteuer hinter sich gebracht hat, sucht vielleicht den starken, impulsiven Neric aus, der – wie die meisten Kindar – die Empfindungen anderer Menschen schon auf große Entfernung wahrnehmen kann. Daher wird keiner der Geheimbunde in der Lage sein, ihm einen Hinterhalt zu legen. Genaa dagegen – einem Kindar-Mädchen – wurde die Gabe des sechsten Sinns nicht in die Wiege gelegt. Wählt man sie, muß man sich also auf das eigene Urteilsvermögen verlassen, wenn man die Handlungen anderer vorhersehen will.

Spielverlauf

Der Spieler macht sich auf die Suche, ausgestattet mit etwas Geld, Nahrung und einem in Green-Sky sehr wichtigen Utensil: dem Shuba. Dies ist ein hauchdünner magischer Umhang, der seinem Träger die Fähigkeit verleiht, bei einem Fall aus großer Höhe wie eine Feder zum Erdboden herabzuschweben (siehe **Abb.**). Mit Hilfe der Karte, die das Fünfergespann aufgetrieben hat, orientiert man sich und versucht, im Gespräch mit anderen Menschen Hinweise für das rätselhafte Verschwinden Raamos zu erhalten. Doch stellt man bald fest, daß nur die Hälfte von Green-Sky auf der Karte verzeichnet ist; um die Aufgabe zu lösen, muß jedoch auch der höchste Ast und die tiefste Höhle durchsucht werden.
Mitunter wird man auf freundliche Leute stoßen, die Speis' und Trank oder ein Lager anbieten... Doch kann man ihnen vertrauen? Manche sind Mitglieder der Geheimbunde und warten nur darauf, über den Spieler herzufallen, wenn er ihnen Gelegenheit dazu gibt; denn sie kennen das Ziel seiner Suche und wissen, daß er Frieden zwischen den beiden Völkern anstrebt.
Traditionell war Green-Sky ein Land ehrlicher und rechtschaffener Menschen. Das schlägt sich natürlich im Charakter der Handelnden nieder. Wenn ein Spieler auf den Gedanken kommen sollte, sich ohne die Erlaubnis der Besitzers einen Gegenstand „auszuleihen", wird ihm schlicht und einfach mitgeteilt: „Das gehört Dir nicht!" Auch eine Mütze voll Schlaf zu nehmen, gestaltet sich schwierig. Da Kindar und Erdlinge niemals im Freien zu ruhen pflegen, gilt es zunächst einmal, jemanden zu finden, der dem Spieler ein Lager anbietet – oder man muß den langen Weg zurück in das eigene Bett antreten. Während des Schlafs erneuert sich unter anderem die geistige Kraft, die man zuvor vielleicht durch den häufigen Einsatz des sechsten Sinns arg strapaziert hat.
Doch wie kann Genaa, die junge Kindar, von der schon die Rede war, ihre Suche zu einem guten Ende bringen, wenn sie die Emotionen anderer nicht wahrnimmt? Ist sie den Mitgliedern der Geheimbunde hilflos ausgeliefert? Natürlich lautet die Antwort: Nein. Viele, die in Green-Sky leben, glauben an die Macht des Guten – unter ihnen mächtige Magier, die Genaas Geisteskräfte stärken werden.
Begegnet sie einem solchen Wesen, so nimmt die Stärke ihres sechsten Sinns jeweils um eine Stufe zu: Zunächst wird Genaa in die Lage versetzt werden, wie Neric die Gefühle der Menschen wahrzunehmen. Hat sie den nächsten Magier gefunden, wird sie auch Gedanken lesen können.



Zum Schluß werden ihre Fähigkeiten sogar das Bewegen lebloser Objekte einschließen. Vielleicht können ja auch die Beeren des Wissens, auf die sie von Zeit zu Zeit stößt, hilfreich wirken?
All seine geistige Macht muß der Spieler zur richtigen Zeit und am richtigen Ort einsetzen – sonst schlägt die Suche fehl. Fünfzig Tage bleiben, um Raamos Schicksal aufzuklären; danach wird sich die Lage so zugespitzt haben, daß wahrscheinlich ein Krieg ausbricht...

Bedienung

Die Bewegung der Spielfigur erfolgt entweder durch die Tastatur oder – angenehmer – mit Hilfe des Joysticks: Ein Neigen des Knüppels nach links oder rechts bewegt die Figur horizontal; beim Hinauf- oder Hinunterklettern geschieht das entsprechend mit senkrechten Steuerbewegungen. Verläßt man den Bereich, der auf dem Bildschirm zu sehen ist, wird von der Diskette ein neuer Hintergrund geladen. Green-Sky besteht insgesamt aus etwa 400 Einzelbildern.
Möchte der Spieler eine bestimmte Handlung ausführen, wird der Joystick-Knopf gedrückt, und ein Menü erscheint (siehe **Abb.**). Das Geschehen wird so lange unterbrochen, bis man seine Wahl getroffen hat, und das Menü wieder verschwindet. Befindet man sich in einer ausweglosen Situation, kann man durch Geisteskraft nach Hause zurückkehren; dabei verliert man allerdings einen Spieltag.
„Below the Root" ist ein weitgehend gewaltfreies Spiel: Mord und Totschlag werden von den Verfassern zu Recht auch im Computer-Spiel als den Charakter korrumpierend angesehen. Die höchste Stufe der Gewalt ist das Entführtwerden durch Mitglieder eines Geheimbundes in ein Versteck, aus dem man leicht wieder entkommen kann.

Spielbarkeit

Zunächst erkundet man die in der Karte verzeichneten Bezirke und bekommt dabei allerhand Hinweise auf wichtige Orte des Geschehens. Danach versucht man, in die unbekannten Gegenden vorzudringen und die Karte so zu vervollständigen. Hat man schließlich alle Hinweise verwertet, liegt die Lösungs des Rätsels nahe.
Wichtig ist, die meist nur aus einem Satz bestehenden Unterhaltungen mit Erdlingen und Kindar genau zu protokollieren. Ganz

gleichgültig, wie nichtssagend ein Hinweis klingen mag: Fast alle Aussprüche und alle gelesenen Gedanken haben eine Bedeutung, die oft nur in Kombination mit einem oder mehreren anderen Dialogen zutage tritt.
Zwar können im Lauf des Spiels Situationen auftreten, in denen man glaubt, den gesamten Bereich abgesucht zu haben, ohne die zur Beendigung der Suche nötigen Hilfsmittel zu entdecken. Bei genauer Betrachtung stellt sich jedoch immer heraus, daß ein oder mehrere Hinweise, die man in Unterhaltungen erhalten hat, nicht genügend beachtet wurden.

Fazit

„Below the Root" ist ein Rollenspiel neuer Art, in dem die Faszination des Rätselratens der Text-Adventures, die gute optische Gestaltung der Grafik-Adventures und die leichte Bedienbarkeit der Actionspiele mit einem phantasievollen Handlungshintergrund so gelungen verwoben sind, wie man es bisher nicht kannte. Die Zusammenarbeit zwischen erfahrenen Programmierern, einem talentierten Künstler und einer begabten Schriftstellerin hat reiche Früchte getragen.
Das Foto wurde mit Hilfe des Polaroid Palette Systems aufgenommen.

Literatur

„Green-Sky Trilogy":
Zilpha Keatley Snyder: Below the Root, And All Between, Until the Celebration; Tor Books, Kanada, 1985

Ein-Blick

Name	Below the Root
Notwendige Ausstattung	Apple II+ / IIe / IIc, 1 Diskettenlaufwerk, Monitor (Joystick empfehlenswert)
Typ	Adventure-Rollenspiel mit animierter Grafik
Gesamtwertung	*****
Spielerzahl	1 Spieler
Dokumentation	*** (Handbuch, 16 Seiten)
	**** (Landkarte, 3Ø x 5Ø cm)
Optischer Eindruck	*****
Akustische Untermalung	***
Einfallsreichtum	*****
Schwierigkeitsgrad	Für Anfänger und fortgeschrittene Abenteurer verschiedene Schwierigkeitsgrade wählbar
Gewaltfreiheit	*****
Spieldauer	Mehrere Tage
Anteilnahme der Spieler	*****
Allgemeine Bedienbarkeit	*****
Preis-Leistungs-Verhältnis	*****
Kopierbar	Nein
Preis	$27.ØØ
Bezugsquelle	Windham Classics, Cambridge, USA

Bewertungsschlüssel:
***** Hervorragend
**** Überdurchschnittlich
*** Akzeptabel
** Schlecht
* Katastrophal

Ein Tag im Stellwerk

Spiel Train Dispatcher

getestet von Thomas Bühner

Spielidee

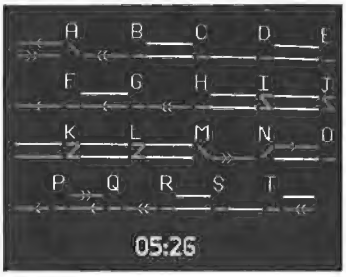
Wer gerne mit der Modelleisenbahn hantiert, kann in Zukunft auch am Mikrocomputer seiner Leidenschaft frönen. Ziel des Spieles „Train Dispatcher" ist es, in einer Arbeitsschicht von acht Stunden eine Anzahl von Güterzügen zwischen zwei Bahnhöfen verkehren zu lassen.
Weil jedoch nur ein Schienenstrang verlegt ist, geht das nicht ohne Komplikationen ab. Zwar verlaufen entlang der Hauptstrecke eine ganze Reihe von kurzen Ausweichgleisen, doch da die Züge mit unterschiedlicher Geschwindigkeit vorankommen, ist eine gut geplante Koordination nicht leicht zu bewerkstelligen. Kompliziert wird das Vorhaben außerdem durch einige dringend notwendige Reparaturen an den Gleisen und den einen oder anderen Schichtwechsel der Zugführer – beides kostet wertvolle Zeit.

Spielablauf

Zunächst wirft man einen Blick auf die Überblickstafel des Stellwerks (siehe **Abb.**). Mit ihrer Hilfe kann man feststellen, wie die Weichen geschaltet sind, für welche Teilstrecken „Freie Fahrt" gegeben wurde und wo sich gerade Züge befinden.

Der Hauptstrang wird von der durchgezogenen Linie zwischen den Weichen A und T gebildet und ist auf der Tafel dreigeteilt worden, um die Darstellung übersichtlicher zu gestalten. Ein Zug – symbolisiert durch einen Doppelpfeil – bewegt sich entweder von A über B, C etc. nach T oder in der entgegengesetzten Richtung. Rückwärtsfahren ist nicht möglich. Eine Teilstrecke – im Bild z.B. der Strang von der Weiche H nach G – ist dann zur Fahrt nach rechts oder links freigegeben, wenn darauf ein einfacher Pfeil erscheint.

Hat man sich einen Überblick verschafft, wählt man die Zugtabelle an. Auf ihr kann man alle wichtigen Betriebsdaten der Züge ablesen, z.B. zu welchem Zeitpunkt sie über die vor ihnen liegende Weiche fahren werden. Nun entschließt man sich, welche Weichen als nächstes gestellt und welche Teilstrecken freigegeben werden müssen. Wenn für einen Zugführer bald der Schichtwechsel bevorsteht, tut man gut daran, seinen Triebwagen mit den angehängten Waggons auf ein Ausweichgleis zu lenken, damit die Strecke während des Wechsels befahrbar bleibt.
Mit einem Blick auf den Reparaturzeitplan vergewissert man sich jetzt besser, daß der geplante Fortschritt nicht durch eine Baukolonne aufgehalten wird. Dieser Plan



wird von den Kolonnen strikt eingehalten. Zwar arbeiten sie an einem Schienenstrang immer nur für kurze Zeit, doch ist dann ein Passieren des betroffenen Streckenabschnitts vollkommen unmöglich. Langsam wird es höchste Zeit, die geplante Weichenstellung vorzunehmen und dem Zug auf einem weiteren Streckenabschnitt freie Fahrt zu erteilen. Dazu holt man sich eine Vergrößerung der entsprechenden Weiche ins Bild und erteilt seine Anweisungen. Es lohnt sich vorauszuplanen, da die Züge eine Geschwindigkeit von 100 km/h erreichen können, wenn eine längere Strecke vor ihnen frei ist, während sie sich im Stop-and-Go-Verkehr mit gemächlichen 50 km/h fortbewegen.
Für die Betriebssicherheit sorgt eine Automatik, die keine Anweisungen akzeptiert, die eine Kollission zur Folge haben könnten. Das Spiel ist zu Ende, wenn alle Züge abgefertigt worden sind oder wenn nach acht „Stunden" die Ablösung im Stellwerk anrückt. Dann berechnet das Programm die erzielten Erfolgspunkte nach verschiedenen Kriterien wie Schnelligkeit und Aufenthaltsdauer der Züge.

Technisches

Alle Anweisungen werden durch einen einzigen Tastendruck gegeben. Damit man nicht – wie bei anderen Spielen – die Befehle auswendig lernen muß, ist im Lieferumfang eine Papp-Schablone für die Tastatur enthalten. Benützt wird nur die oberste Tastenreihe und Return. Für Apple II+ und IIe sind verschiedene Belegungen vorgesehen, die Rücksicht auf die unterschiedliche Tastatur nehmen.

Grafische Darstellung

Da man ständig zwischen Tabellen und Grafiktafeln hin und her schaltet, spielt der übersichtliche Aufbau dieser Darstellungen eine große Rolle. Zunächst wird jeder die Bilder als „langweilig" empfinden. Nach einigen Minuten des Spiels stellt man jedoch fest, daß durch die strenge Form eine gute Übersicht über das Geschehen ermöglicht wird. Bei einer Auflockerung durch Lokomotiven-Bilder oder ähnliches wäre es nicht möglich, den Inhalt einer Tafel in wenigen Sekunden zu erfassen. Gerade das aber ist notwendig, um schnell Entscheidungen treffen zu können. Die Ästhetik mußte hier der Funktion weichen. Obwohl bei den Diagrammen Farbe eingesetzt wird, spielt es sich ebenso einfach, wenn man nur einen Schwarz-Weiß-Monitor anschließt.

Schwierigkeitsstufen

Fünf verschiedene Schwierigkeitsstufen stehen zur Verfügung. Je nach Wahl dauert das Spiel maximal zwischen 15 und 30 Minuten. Variabel sind dabei die Zahl der auftauchenden Züge, Ort und Dauer der notwendigen Reparaturen und der Zeitpunkt des Schichtwechsels der Zugführer. Während bei einer Spielzeit von 30 Minuten das Abfertigen von acht Zügen kein Problem darstellt, ist es fast unmöglich, innerhalb von 15 Minuten zwölf Züge bei ständigen Gleisreparaturen erfolgreich zu rangieren.

Tonerzeugung

Es findet keine akustische Untermalung des Geschehens statt. Um die Aufmerksamkeit des Spielers zu erwecken, ertönt beim Passieren einer Weiche und bei der Erteilung einer gültigen Anweisung ein Piepsen. Beide Ereignisse finden aber fortwährend statt; der Computer piepst also alle paar Sekunden, so daß darauf nicht mehr geachtet wird. Allenfalls wirkt es bereits nach kurzer Zeit störend.

Programmfehler

Da die Züge nicht in den Rückwärtsgang geschaltet werden können, sind Situationen möglich, die ein Weiterspielen verhindern: drei Lokomotiven stehen sich dann gegenüber – zwei auf dem Hauptstrang und eine am Ende der Ausweichstrecke – und können weder vor noch zurück. Das Programm ist dann nicht in der Lage zu erkennen, daß der Spieler „Mist gebaut" hat und zählt brav die Minuten bis zum Schichtwechsel, selbst wenn sich kein Zug mehr bewegen kann. Eine Unterbrechung des Spiels ist auch in solch aussichtslosen Fällen nur durch einen Neustart des Apple zu bewerkstelligen.

Kopierschutz

Unternimmt man den Versuch, die Diskette mit dem Standard-Kopierprogramm COPYA zu duplizieren, stellt man ebenso erstaunt wie erfreut fest, daß keine Probleme dabei auftreten. Nach kurzer Untersuchung ergibt sich, daß keinerlei Schutzmaßnahmen getroffen wurden; das Disketten-Betriebssystem ist originales DOS 3.3, das vom Benutzer durch ein schnelleres DOS ersetzt werden kann. Hoffentlich wird dieses kundenfreundliche Verhalten dadurch honoriert, daß niemand das Programm raubkopiert. Schon der geringe Preis gebietet, dem fairem Verhalten des Herstellers mit einem fairen Kaufverhalten zu begegnen.

Spielgefühl

Während man sich mit „Train Dispatcher" befaßt, vergißt man gewöhnlich seine Umgebung schon aus dem Grund, weil der Inhalt der verschiedenen Tabellen und Grafiktafeln in Sekunden erfaßt werden muß. Die folgenden Entscheidungen über die Freigabe von Streckenabschnitten trifft man ebenso schnell – oder das Spiel ist verloren. Manche der „Oldies" unter den Apple-Besitzern kennen ein ähnliches Gefühl von nächtlichen Sitzungen mit dem etwa 1978 entstandenen Programm „Air Traffic Controller".

Ein-Blick

Name	Train Dispatcher
Notwendige Ausstattung	Apple II+ / IIe / IIc,
Typ	Strategisches Action-Spiel
Gesamtwertung	****
Spielerzahl	1 Spieler
Dokumentation	**** (12 Seiten)
Optischer Eindruck	****
Akustische Untermalung	*
Einfallsreichtum	*****
Spieldauer	15–30 Minuten
Anteilnahme der Spieler	****
Allgemeine Bedienbarkeit	*****
Verhalten bei Bedienungsfehlern	****
Preis–Leistungs–Verhältnis	****
Kopierbar	Ja
Preis	$30.00
Bezugsquelle	Signal Computer Consultants, Pittsburgh, USA
Bewertungsschlüssel:	***** Hervorragend **** Überdurchschnittlich *** Akzeptabel ** Schlecht * Katastrophal

Fehlerjagd im Computerlabor

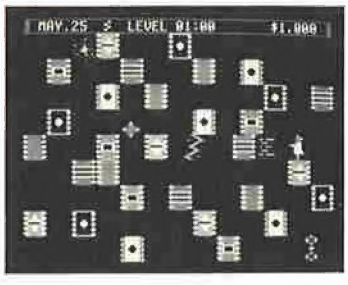
Spiel I.O. Silver

getestet von Thomas Bühner

Oft langweilt man sich schon nach kurzer Zeit bei Spielen, die spannende Unterhaltung durch schnelle Action bieten, weil nur geistlose Reaktion gefordert wird. Bei I.O. Silver gibt es auch viel zu knobeln.

Vorgeschichte

Professor Silver ist ein in Ehren ergrauter Ingenieur, der endlich seinen lang gehegten Traum erfüllen kann: Den Bau eines Supercomputers, wie ihn die Welt noch nicht gesehen hat. Alle Chips, die dazu notwendig sind, hat er bereits fertiggestellt. Es bleibt nur noch eines zu tun: Er muß seinen Traumrechner zusammenbauen. Dazu zieht er sich in sein neues, perfekt ausgestattetes Labor zurück und ist für Niemanden mehr zu sprechen.



Das strategische Spiel

Sechs verschiedene Typen von Chips hat der Professor konstruiert, deren Zusammenbau auf ungewohnte Weise vor sich geht. Wenn man nämlich zwei Chips derselben Bauart aufeinanderlegt, verschmelzen sie mit einem scharfen Klicken unlösbar miteinander. Auch das neue Labor hat es in sich: Von den Mitarbeitern Professor Silvers wurde es mit einem Fußboden ausgestattet, der so glatt ist, daß sich eine Eislaufbahn dagegen wie Sandpapier ausnimmt. Will man also einen Chip verschieben, muß man ihm nur einen kleinen Stoß geben, und er

wird erst wieder zum Stillstand kommen, wenn er auf ein Hindernis prallt.
Da die Chips aber sehr groß sind und der Wissenschaftler recht betagt ist, kann er nicht mehrere gleichzeitig wegschieben. Liegen zwei Bauteile direkt nebeneinander, ist es unmöglich, sie beide in Bewegung zu setzen. Den einfarbigen Chip in der linken unteren Hälfte des Bildes etwa könnte er im Augenblick nicht verrücken, da sein Weg horizontal und vertikal versperrt ist. Zuerst müßte er eines der beiden blockierenden Stücke beiseite schaffen. Professor Silver muß demnach um jeden Preis verhindern, daß vier Chips im Quadrat aneinander stoßen, da sie sich dann gegenseitig im Weg liegen und nicht mehr verschoben werden können.
Die Aufgabe des Professors besteht nun darin, alle Chips des gleichen Typs zusammenzubringen. Sobald die letzten beiden gleichartigen Bauteile verschmelzen, spielt sich ein von dem Wissenschaftler entwickelter, geheimnisvoller Prozeß ab, und aus der Fusion geht – musikalisch untermalt – eine Leiterplatine hervor, die ebenso groß wie ein einzelner Chip ist. Der Forscher hat sich dann ein wenig Platz im Labor geschaffen, denn wo vorher fünf Stücke der gleichen Farbe waren, befindet sich jetzt eine einzige Platine.
Im **Bild** hat Professor Silver gerade mit der Arbeit begonnen. Er schiebt soeben einen gestreiften Chip nach links. Sobald der zum Stillstand gekommen ist, wird er ihm noch einen Stoß nach unten versetzen, und das Bauteil wird mit einem anderen seiner Art verschmelzen.
Ist es dem Wissenschaftler gelungen, durch verschiedene Fusionen zwei Platinen zu erzeugen, kann er diese beiden zusammenbringen und ein Taschenrechner wird entstehen. Fügt er dem Taschenrechner eine weitere Platine hinzu, verwandelt sich das Gerät in einen Mikrocomputer. Die Reihe geht dann weiter über Minicomputer und Großrechner bis zum Traumziel des Professors: dem Supercomputer.
Die Schwierigkeit ist nur, daß bei jeder Fusion ein Bauteil verloren geht, und so immer weniger Hindernisse im Labor sind, die einen gleitenden Chip zum Stehen bringen könnten. Denn wie man sieht, hat der Raum, in dem der Professor arbeitet, keine Wände. Durch einen technischen Kniff betritt alles, was auf einer Seite verschwindet, das Zimmer auf der gegenüberliegenden Seite wieder. Das macht es mit zunehmender Zeit nicht nur schwieriger, die richtigen Teile miteinander zu kombinieren; es ist sogar gefährlich. Würde der Forscher etwa den Chip, über dem er sich im Augenblick befindet, nach unten wegdrücken, so würde der am unteren Rand des Bildschirms verschwinden, am oberen wieder auftauchen und den Wissenschaftler durch seinen Schwung von den Beinen reißen. Da Professor Silver nicht mehr der Jüngste ist, bedeutet ein solcher Unfall natürlich das Ende seiner Karriere.
Verfügt man nicht über viel Erfahrung – oder eine hervorragende Fähigkeit zur logischen Planung –, wird das Spiel meist dadurch beendet, daß es keinen Baustein mehr gibt, der verschoben werden könnte: Alle Teile liegen dann im Quadrat aneinander, oder es gibt keine Hindernisse mehr, die einen gleitenden Chip, Taschenrechner oder Computer aufhalten könnten. Schließlich streicht man seine Bezahlung ein – auch Wissenschaftler leben nicht von der Luft allein – und startet einen weiteren Versuch, der dann vielleicht von mehr Erfolg gekrönt ist.

Das Action-Spiel

Für Spieler, die auch bei strategischen Planungen einen gewissen Nervenkitzel schätzen, befindet sich eine zweite Version von I.O. Silver auf der Diskette. In diesem Szenario hat der Professor das Labor nur für ein Jahr zur Verfügung. Dieses „Jahr" spielt sich im Lauf von etwa zwölf Minuten auf dem Bildschirm ab. Außerdem wurde seine Arbeit um eine realistische Komponente erweitert: Während des Konstruktionsprozesses können vier Fehler auftreten – im Computer-Jargon „Wanzen" genannt –, die es zu vermeiden gilt: der Logische Fehler, die Unendliche Schleife, der Overflow und schließlich der Stromausfall. All jene, die schon einmal Programme geschrieben haben, kennen diese Gesellen und ihre verheerende Wirkung: Jedesmal, wenn Professor Silver auf einen der Vier trifft, verliert er zwischen ein und vier Wochen Zeit.
Aber jeder Fehler kann durch sorgfältige Arbeit ausgeschlossen werden, so auch hier. Das „Entwanzen" geschieht, indem der Forscher einen der unangenehm pulsierenden Burschen zwischen zwei Bauteilen zerquetscht, worauf hin der betroffene Fehler mit einem häßlichen Quietschen sein Unwesen einstellt. In der **Abb.** hat das letzte Stündlein des zickzackförmigen Stromausfalls geschlagen. Gelingt es dem Professor, allen Vieren auf diese Weise den Garaus zu machen, kann er für den Rest des Jahres in Ruhe seiner Arbeit nachgehen. Wenn er vorher aber von einer der verbleibenden „Wanzen" getroffen wird, nehmen alle schon Totgeglaubten wieder ihr Treiben auf.
Verliert der Wissenschaftler einmal den Überblick, kann er sich Urlaub nehmen und eine neue Strategie entwerfen. Das Geschehen auf dem Bildschirm friert dann ein und man hat beliebig viel Zeit, um die nächsten Schritte zu planen. Nach dem vierten Urlaub jedoch ist keine Entspannung mehr möglich; dann läuft die Zeit bis zum 31. Dezember unerbittlich und ohne weitere Unterbrechung.
Dieses I.O. Silver Szenario ist recht schwierig zu beherrschen. Man braucht viel Spielerfahrung, bis man auch nur annähernd so gute Ergebnisse erzielt wie zuvor im rein strategischen Teil. Die Action-Version ist aber sehr unterhaltsam und wird bei Vielen schlaflose Nächte verursachen, in denen man immer wieder versucht, endlich den Supercomputer zu konstruieren.

Technisches

Wenn man das Geschehen im Action-Szenario unterbrechen will, kann man den Stand auf der Diskette abspeichern und das Spiel zu einem späteren Zeitpunkt weiterführen. Die besten Ergebnisse werden auf der Diskette festgehalten.
Die Steuerung des Professors geschieht mit vier beliebigen Tasten; wer will, kann auch einen Joystick benutzen. Musik und Töne sind abschaltbar (was jeder Nachbar zu später Stunde schätzen wird).
Um das Erscheinungsbild der einzelnen Chip-Typen klar unterscheiden zu können, gibt man dem Programm an, ob ein Farb- oder Schwarz-Weiß-Monitor an den Apple angeschlossen ist; die Farbe bzw. Textur wird dann entsprechend geregelt.
Wer selbst in 6502-Assembler programmiert und an guten Arbeitsvorlagen interessiert ist, kann sich für weitere $30 den kommentierten Source Code von I.O. Silver schikken lassen. Der Code umfaßt sechs Diskettenseiten und wurde mit dem Merlin Assembler erstellt.

Ein-Blick

Name	I.O. Silver
Notwendige Ausstattung	Apple II+ / IIe / IIc,
Typ	Strategisches Action-Spiel und Strategiespiel
Gesamtwertung	*****
Spielerzahl	1 Spieler
Dokumentation	***** (2Ø Seiten)
Optischer Eindruck	****
Akustische Untermalung	****
Einfallsreichtum	*****
Spieldauer	Action-Spiel: ca. 2Ø Minuten Strategiespiel: bis ca. 1 Stunde
Anteilnahme der Spieler	*****
Allgemeine Bedienbarkeit	*****
Preis-Leistungs-Verhältnis	****
Kopierbar	Ja
Preis	$3Ø.ØØ
Bezugsquelle	Beagle Bros., San Diego, USA
Bewertungsschlüssel:	***** Hervorragend **** Überdurchschnittlich *** Akzeptabel ** Schlecht * Katastrophal

FACHBUCHTIP Hüthig

Das Buch zum Apple-Writer II/IIe

Hans Gabriel

1986, 155 S., kart., DM 35,—
ISBN 3-7785-1234-X

„Das Buch zum Apple Writer II/IIe" wendet sich an alle, die dieses Textverarbeitungssystem schon einsetzen oder einsetzen wollen.
In einzelnen, in sich geschlossenen Kapiteln erlernt der Leser alle Funktionen und erzielt schnelle Erfolgserlebnisse.
Im ersten Kapitel werden typische Arbeitsstellungen der Textverarbeitung und ihre systematische Bearbeitung vorgestellt. Das zweite Kapitel stellt in logischen Funktionsgruppen die Befehle vor, mit denen der Applewriter während der Textarbeit direkt gesteuert werden kann. Kapitel 3 zeigt die Programmierung des Applewriters in der Text-Kommando-Sprache TKS (WPL). Dazu werden Beispiele analysiert. Auf der Begleitdiskette zum Buch, die separat bezogen werden kann, sind darüber hinaus umfangreiche Schwerpunkterklärungen enthalten, die über die Help-Funktion vom Benutzer abgerufen werden können. Eine kleine Adreßdatenbank mit den dazugehörigen Dienstprogrammen zur Pflege der Daten und zu ihrem Einsatz in Einzel- und Serienbriefen befindet sich ebenfalls auf Diskette.
„Das Buch zum Apple Writer II/IIe" behandelt sowohl die alte Programmversion für den Apple IIplus als auch die neue Ausgabe, die 128-kByte auf dem Apple IIe oder Apple IIc unterstützt.

BESTELLCOUPON

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

Buchtitel

Name

Ort

Straße

Unterschrift

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

Fast-Writer

von Harald Grumser
Programmdiskette und Handbuch
Gerätevoraussetzung: Apple IIe oder IIc (nicht II+)
DOS-3.3-Version. Auslieferung ab 1.6.86
Normalpreis DM 128,– (ISBN 3-7785-1419-9)
Sonderpreis für Peeker-Abonnenten DM 98,–

ProDOS-Version. Auslieferung ab 1.9.86
Normalpreis DM 128,– (ISBN 3-7785-1421-0)
Sonderpreis für Peeker-Abonnenten DM 98,–
Kombinationspreis für Bezieher der DOS-3.3-Version DM 28,–

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

Flexibilität

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

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

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

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

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

Befehlsvorrat

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

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

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

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

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

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

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

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

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

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