

# INPUT 64

DAS ELEKTRONISCHE MAGAZIN 11/87

Infos · News · Programme · Unterhaltung · Tips

DM 19,80

Unverbindliche Preisempfehlung

Durchblick garantiert:

## Maschinensprache-Monitor

Für Rechner und Floppy

Spaß total:

## Das große Quiz

Ratespiel für alle

Fehler ade:

## Syntax-Check

Prüfung für BASIC

Mailing: Electrobrief

Updates: ID-Werkstatt

Drucken:

Multi-Hardcopy

Kalender-Programm

Serie: 64er Tips

Spiel: Time Race



# Hinweise zur Bedienung

INPUT 64 ist nicht nur einfach eine Programmsammlung auf Diskette, sondern ein Elektronisches Magazin. Es enthält ein eigenes Betriebssystem mit Schnelllader und komfortabler Programmauswahl. Die Bedienung ist kinderleicht.

Bitte entfernen Sie vor dem Laden eventuell vorhandene Steckmodule, und schalten Sie den Rechner einmal kurz aus und wieder ein. Geben Sie nun zum Laden der Diskette

## LOAD "INPUT\*", 8,1 und RETURN

ein. Alles Weitere geschieht von selbst.

Es wird nun zunächst ein Schnelllader initialisiert. Besitzen Sie ein exotisches Laufwerk oder ist Ihre Floppy bereits mit einem hardwaremäßigen Beschleuniger ausgerüstet, kann es zu Konflikten mit unserem SuperDisk kommen. In diesem Falle sollten Sie versuchen, die Diskette mit

## LOAD "LADER\*", 8,1 und RETURN

zu laden.

Nach der Titelgrafik springt das Programm in das Inhaltsverzeichnis des Magazins. Hier können Sie mit der Leertaste weiter und mit SHIFT und Leertaste zurückblättern. Mit RETURN wird das angezeigte Programm ausgewählt und geladen.

Das Betriebssystem von INPUT 64 stellt neben dem Inhaltsverzeichnis noch weitere Funktionen zur Verfügung. Diese werden mit der CTRL-Taste und einem Buchstaben aufgerufen. Sie brauchen sich eigentlich nur CTRL und H zu merken, denn mit dieser Tastenkombination erscheint eine Hilfsseite auf dem Bildschirm, die alle weiteren System-Befehle enthält. Nicht immer sind alle Optionen möglich. Befehle, die zur Zeit gesperrt sind, werden auf der Hilfsseite dunkel angezeigt. Hier nun die Befehle im einzelnen:

## CTRL und Q

Diese Tastenkombination hat nur während der Titelgrafik eine Bedeutung. Mit ihr wird

das Titelbild abgekürzt, und Sie landen sofort im Inhaltsverzeichnis.

## CTRL und H

Haben wir schon erwähnt – damit wird die Hilfsseite ein- und ausgeschaltet.

## CTRL und I

Sie verlassen das gerade laufende Programm und kehren ins Inhaltsverzeichnis zurück.

## CTRL und F

Ändert die Farbe des Bildschirmhintergrundes. Diese Option funktioniert immer, wenn ein Programm läuft oder Sie sich im Inhaltsverzeichnis befinden, aber nicht auf der Hilfsseite.

## CTRL und R

Wie CTRL-F wirkt auf die Rahmenfarbe.

## CTRL und B

Sie erhalten einen Ausdruck der Textseite eines laufenden Programmes auf einem angeschlossenen Drucker. Diese Hardcopy-Routine ist angepaßt für Commodore-Drucker und kompatible Geräte. Das Programm wählt automatisch die richtige Gerätadresse (4, 5 oder 6) aus. Sie können diese Routine mit der ←-Taste abbrechen.

## CTRL und S

Programme, die auch außerhalb von INPUT 64 laufen, können Sie mit diesem Befehl auf eine eigene Diskette überspielen. Wenn Sie diesen Befehl aktivieren, bekommen Sie unten auf der Hilfsseite angezeigt, wie viele Blocks das File auf der Diskette belegen wird. Geben Sie nun den Namen ein, unter dem das Programm auf Ihre Diskette geschrieben werden soll. In der Regel handelt es sich um Programme, die Sie ganz normal laden und mit RUN starten können. Ausnahmen sind in den jeweiligen Programmbeschreibungen erläutert.

## CTRL und D

Gibt das Directory der eingelegten Diskette

aus. Die Ausgabe kann mit der Leertaste angehalten und mit RETURN wieder fortgesetzt werden. Ein Abbruch ist mit der ←-Taste möglich. Wenn das Directory vollständig ausgegeben ist, gelangen Sie mit der RETURN-Taste zurück ins unterbrochene Programm beziehungsweise auf die Hilfsseite.

## CTRL und @

Disk-Befehle senden, zum Beispiel Formattieren einer neuen Diskette oder Umbenennen eines Files. Für den zu sendenden Befehls String gilt die übliche Syntax, natürlich ohne ein- und ausführende Hochkomata. CTRL-@ und RETURN gibt den Zustand des Fehlerkanals der Floppy auf dem Bildschirm aus. Weiter im Programm oder zurück auf die Hilfsseite führt ein beliebiger Tastendruck.

## CTRL und A

Sucht auf der Diskette nach einem INPUT 64-Inhaltsverzeichnis. Mit diesem Befehl ist es möglich, ohne den Rechner auszuschalten, Programme von anderen INPUT 64-Disketten zu laden. Das funktioniert aber nur bei den Ausgaben ab 4/86.

## Bei Ladeproblemen

Bei nicht normgerecht justiertem Schreib-/Lesekopf oder bei bestimmten Serien wenig verbreiteter Laufwerke (1570) kann es vorkommen, daß das INPUT-Betriebssystem eingebaute Schnellladeverfahren nicht funktioniert. Eine mögliche Fehlerursache ist ein zu geringer Abstand zwischen Floppy und Monitor/Fernseher. Das Magazin läßt sich auch im Normalverfahren laden, eventuell lohnt sich der Versuch:

## LOAD "LADER", 8,1

Sollte auch dies nicht zum Erfolg führen, senden Sie bitte die Diskette mit einem kurzen Vermerk über die Art des Fehlers und die verwendete Gerätekonstellation an den Verlag (Adresse siehe Impressum).

# Liebe(r) 64er-Besitzer(in)!

Wenn sie genügend lange zurückliegen, kann man ja sogar seine peinlichsten Anfängersünden souverän gestehen. Als ich vor einigen Jahren – der C64-Vorgänger VC 20 war gerade angeschafft – zum ersten Mal den Begriff „Monitor“ las und erfuhr, daß man damit Maschinenspracheprogramme eingeben kann, brachte mich das einigermaßen ins Schleudern. Schließlich hatte ich ja einen Monitor. Nebenbei: Ge nau genommen war es nur ein Fernseher, das konnte aber allenfalls mit der Bildqualität zu tun haben. Maschinenspracheprogramme ließen sich damit jedenfalls beim besten Willen nicht eingeben.

Es hat einige Zeit vorsichtigen Erkundens gekostet – wer steht schon gern als der blutige Laie da, der er ist –, bis ich herausfand, was es mit diesem merkwürdigen Ding auf sich hat. Bewährt hat sich letztendlich die Methode „So tun, als ob!“ Sich den Verkäufer der Computer-Fachabteilung eines Kaufhauses schnappen und mit Kennermiene fragen: „Welche Maschinensprache-Monitore haben Sie denn für mein Modell so auf Lager?“

Wenn der sich daraufhin von den Regalen mit den Bildschirmen ab- und der Software-Vitrine zuwendet, fällt es einem wie Schuppen von den Augen: Anscheinend gibt es solche und solche Monitore, und ein Maschinensprache-Monitor gehört im Gegensatz zu einem Schwarzweiß- oder Farbmonitor nicht in die Kategorie der Datensichtgeräte. Der Erfolg dieser Methode hängt natürlich davon ab, daß der Fachverkäufer diese Berufsbezeichnung zu Recht führt und nicht nach dem Regalfach sucht, wo die Bildschirme mit der Aufschrift „Maschinensprache“ stehen.

Daß diese Vorgehensweise überhaupt notwendig wurde, hängt damit zusammen, daß Fachzeitschriften und Fachliteratur das „Fach“ in ihrem Namen zwar meist zu Recht führen, aber sehr eng auslegen und sich nur an Fachleute wenden. Denen ist natürlich klar, was es mit den unterschiedlichen Begriffsbedeutungen des Wortes „Monitor“ auf sich hat. Der Einsteiger muß sehen, wie er weiterkommt.

Im unverständlichsten Fachjargon verfaßte Artikel und lückenhafte Bedienungsanleitungen – wahrscheinlich verfallen auch wir in INPUT 64 ab und an in diese „fachidiotische“ Schreibweise. Sie können sich dann ja aussuchen, ob Sie wie im geschilderten Beispiel vorgehen – nach dem Motto „Nichts anmerken lassen!“ Oder sich auf eine alte Volksweisheit beziehen: „Es gibt keine dummen Fragen, es gibt nur dumme Antworten.“ Soll ja meistens stimmen.

Jürgen Seeger

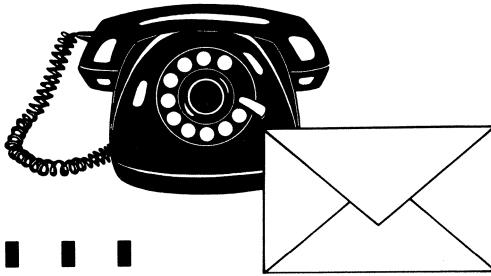
11/87

## INHALT



Leser fragen	2
News	
Neuer Btx-Decoder von Commodore	3
Maschinensprache-Monitor MLM64plus	
Floppy-Befehle, Assembler und Disassembler integriert	4
ID-Werkstatt	
Eigene Zeichensätze für Vokabeltrainer	10
Das große Quiz	
Familien-Ratespiel	11
Multi-Hardcopy	
Text, Sprites und Grafik drucken	12
Compiler-Know-how	
Tips zum Speedcompiler	14
Adventure-Wettbewerb	
3000 DM zu gewinnen	17
Syntax-Check	
Gegen Fehler in BASIC-Programmen	18
Neue CP/M-Karte für den C64	21
64er Tips	
Interrupt-Grundlagen	22
Kalender	
Geld sparen durch Eigendruck	26
Electrobrief	
Post per Diskette	28
Time Race	
Spiel gegen die Zeit	29
Vorschau	31
Impressum	32

# Leser fragen . . .



## Fehler im Assembler

Ist es richtig, daß ein 'dd-error' (doppelt definiertes Label) nicht immer richtig erkannt wird, wenn man hinter einem doppelt benutzten Label ein Makro aufruft?  
(tel. Anfrage)

Sie haben recht; danke für den Hinweis. Der Fehler im Assembler kann umgangen werden, wenn das Makro in die nächste, dem Label folgende Zeile geschrieben wird.  
(d. Red.)

Die neue, „nackte“ Gestalt von INPUT 64 halte ich für sowohl funktioneller als auch umweltbewußter, denn es werden jetzt keine Plastikmaterialien mehr verschwendet und unnötige Verpackungen werden auch vermieden. Vom Inhalt her entspricht INPUT 64 voll meinen Wünschen. Nur weiter so (aber bitte keine weiteren Preiserhöhungen).

C. Völker, Oersdorf

## Pro und Kontra zum neuen Heft

Als langjähriger Computer-User habe ich natürlich sofort von INPUT 64 Gebrauch gemacht (ich beziehe es seit 7/85).

Jedesmal wenn Sie am „Screen-Layout“ oder am Beiheft was veränderten, wurde es immer besser. Aber mit der neuesten Aufführung habt Ihr Euch wirklich selbst übertroffen! Kompliment!  
Boris Lorenz,  
Nürnberg

Was mich heute erzürnt, ist das neue Format des Beiheftes!!!

1. paßt es nicht mehr in A5-Klarsichthüllen, da es unnötigerweise vergrößert wurde.

2 ist die Verpackung der Diskette unpraktisch, weil sie nicht entfernt werden kann.

Sicher war die alte Verpackung nicht der Weisheit letzter Schluß, aber so bekomme ich beim Kauf ein Heft, welches schon etwas unansehnlich ist, weil es von Leuten mit schmutzigen Händen durchgeblättert wurde. Das dürfte auch nicht im Sinne des Erfinders sein!  
W. Schmidt, Berlin

zeugt. Deswegen muß das Strobe-Signal nicht explizit programmiert werden.

Manche Centronics-Kabel verbinden aber nicht diesen Pin mit dem Strobe-Signal, sondern Pin M; gleichbedeutend mit Bit 2 des Port-A-Datenregisters, in der Fachliteratur als 'PA2' bezeichnet. Diese Tradition entstammt der VC20-Ära, aber auch beim 128er unter CP/M wird dort Strobe erzeugt. Genau das wird durch die Änderung von Herrn Würdemann nachvollzogen; dadurch entfallen Probleme mit alten Kabeln und die Inkompatibilität zwischen C128-CP/M-Betrieb und C64.

Da wir damals auch den Source-Code des Treibers veröffentlicht haben, können Besitzer des INPUT-ASS (der im Juni 1986 in unserem Magazin erschienene Macro-Assembler) die Änderung leicht einbauen:

An zwei Stellen, bei der Initialisierung und der Datenausgabe, taucht der Befehl „*sta port*“ auf. Dadurch werden die Daten im Akku in das Ausgaberegister geschrieben. Hinter diesen beiden Befehlen „*jsr strobe*“ einfügen, das Unterprogramm „*strobe*“, um das der Druckertreiber ergänzt werden muß, sieht folgendermaßen aus:

```
:strobe
pha
lda $dd00
and #%11111011
sta $dd00
ora #%"00000100
sta porta
pla
rts
```

Dadurch wird ein negativer Impuls an Pin M des Userports erzeugt.  
(d. Red.)

## Dienstag ist Lesertag

Technische Anfragen:  
nur Dienstag von 9 — 16.30 Uhr  
 (05 11) 53 52-0

## Latinus-Trainer

Mit dem „Vokabeltrainer“ aus INPUT 64, Ausgabe 3/87 kann jetzt auch Latein gepunktet werden. Dazu ist ein „Cursus Latinus“ auf Diskette und C60-Kassette für 10 DM erhältlich. Die Kassette beziehungsweise Diskette enthält eine Vokabeltrainer-gerechte sequentielle Datei von 194 Blöcken (entspricht 48 KByte). Ungefähr 1600 Bytes sind noch für weitere, eigene Voka-belnen frei.

Bezugsquelle:

Horst Breitkreuz  
Chemnitzer Str. 5  
8900 Augsburg 1

## Noch mehr ICI

Wir werden immer wieder gebeten, weitere Beispiele für ICI-Batch-Dateien (ICI ist der INPUT-Command-Interpreter aus Ausgabe 7/87) vorzustellen.

Hier nun zwei Dateien. Die erste Batch-Datei ist ein Beispiel für eine Grundinitialisierung.

Also ICI laden und starten, danach nur noch „Batch start“ eingeben.

```
8>edit start
:poke d020 00
:poke d021 00
:poke 0286 0d
:poke 028a 80
:print guten tag lieber ....
:print bitte mal 'ne diskette
:print dann taste
:wait
:dir
:print taste go basic
:print stop go ici
:wait
:basic
:@
8>
```

Das zweite Beispiel dient der professionellen Programmentwicklung. Hierfür ist es unabdingbar, von Zeit zu Zeit Sicherungskopien der aktuellen Programme zu erstellen. Das folgende Batch-File muß sich auf der jeweiligen Programm-diskette befinden und setzt voraus, daß Sie für unterschiedliche Programme auch verschiedene Disketten benutzen.

```
8>edit sichern
:poke 002b 01
:poke 002c 08
:dos s:aktuell.bak
:dos r:aktuell.bak=aktuell
:save aktuell
:error
:basic
:@
8>
```

Durch Drücken von RESTORE aktivieren Sie ICI und geben danach 'Batch sichern' ein. Alles weitere geschieht von selbst.

## Kein Bonus mehr

Wenn ich einige Zeit mit dem Spiel „Bingo“ aus INPUT 64, Ausgabe 8/87 gespielt habe, springt der Superbonus von 9999 auf 0, das darf doch nicht sein, oder?

(tel. Anfrage)

Es ist leider so. Der Bonus zählt nicht ins Unermessliche hoch, sondern springt, nachdem er 9999 erreicht hat, wieder auf 0. Das ist durch den Bildschirmaufbau bedingt, daher also kein Fehler. (d. Red.)

# News

## Text für Bildschirme

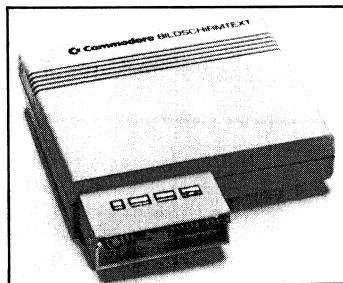
Auf der Internationalen Funkausstellung 1987 in Berlin stellte die Firma Commodore ihre zweite Generation von Btx-Decodern vor. Diese Neuentwicklung soll eine wirkliche Alternative zu den herkömmlichen Bildschirmtext-Systemen sein und durch den Preis von 399 DM praktisch jedem den Zugang zum aktuellen Service der Post öffnen.

Verglichen mit den herkömmlichen Systemen ist der Btx-Decoder von Commodore sehr preiswert. Dem Computer-Benutzer, sofern eine komplette C64- oder C128-Anlage vorhanden ist, könne nach Aussage von Commodore somit jeglicher Betriebskomfort zur Verfügung stehen, der sich in der Btx-Praxis als notwendig und wünschenswert erwiesen hat. Außerdem benötigte man nur noch das kompakte Steckmodul, das auf den Expansion-Port gesteckt wird, und eine Btx-Anschlußbox von der Bundespost.

Mit dieser Konfiguration lassen sich Standard-Eigenschaften eines Btx-Systems realisieren: „...das automatische Abfragen, mit dem Zeit und Kosten gespart werden, die Möglichkeit des Abspeicherns von Btx-Seiten, die Ausgabe von Texten auf einen Drucker sowie das Erstellen und Ausführen sogenannter Makro's“. (Commodore) Außerdem soll unter anderem auch das Abrufen von Telesoftware über Bildschirmtext möglich sein.

Große Hoffnungen setzt Commodore in das Potential der künftigen Btx-Anwender. Der

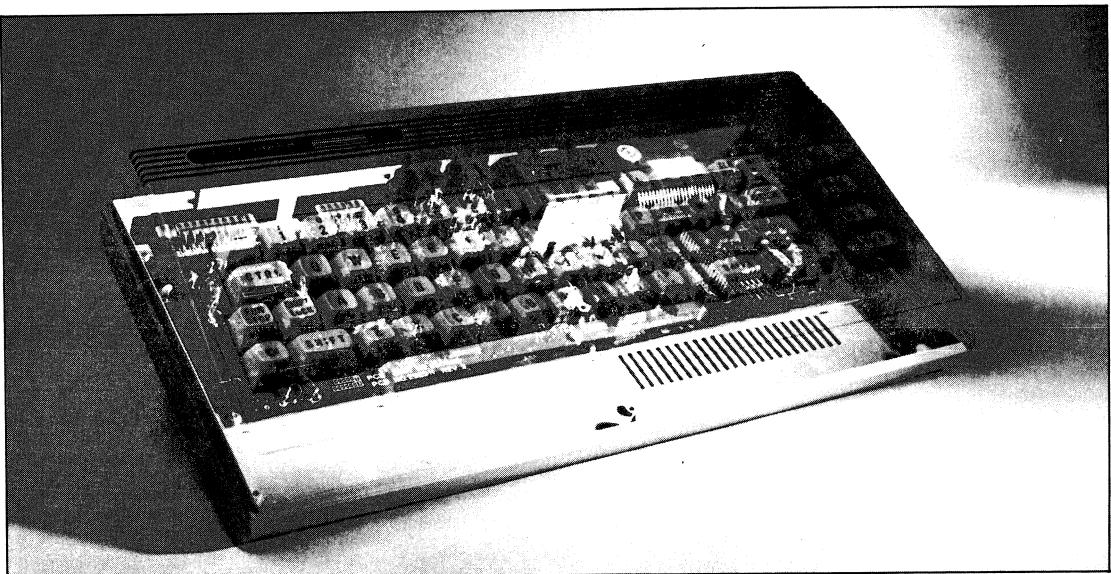
C64 ist allein in Deutschland mit rund 1,3 Millionen verkauften Geräten der meistgenutzte Rechner überhaupt. Zusammen mit dem ebenfalls sehr populär gewordenen C128 summieren sich die Zahl der installierten Systeme, mit denen das Btx-Modul Verwendung finden können, im Bundesgebiet auf über 1,5 Millionen. Bekäme auch nur ein kleiner Teil dieser großen Anwenderfamilie Lust auf den Btx-Anschluß, würde die Zahl der heutigen Bildschirmtext-Anwender binnen kürzester Zeit vervielfachen. kfp



Technische Daten:

- RAM-Puffer für mindestens zehn Seiten (20 KByte)
- beliebige Dateinamen mit max. 16 alphanumerischen Zeichen
- pro Diskette max. 36 Makrodateien
- bis zu 100 Btx-Seiten auf einer Diskette speicherbar.

Commodore Büromaschinen GmbH  
Lyoner Str. 38  
6000 Frankfurt/Main 71



# Voller Durchblick

**MLM64plus:**  
**Maschinensprache-Monitor für Floppy und Rechner**

Die Bezeichnung „Monitor“ für dieses Programm ist bei Licht besehen krasstes Understatement. Der MLM64plus kann natürlich auch, wie jeder Monitor, Speicherinhalte lesen, verändern, laden oder abspeichern. Wie bei jedem guten Monitor stehen außerdem ein Disassembler, ein Zeilenassembler zum Eingeben von kleinen Maschinenprogrammen und ein Debugger zur Verfügung. Wie nur wenige der guten Monitore kann er auch rechnen. Und das „plus“ hat er sich mit seinen umfangreichen Floppy-Optionen verdient: Directory-Funktion, Disk-Befehle senden, Sektoren lesen und schreiben, Zugriffsmöglichkeit auf das Floppy-RAM.

Das „plus“ führt der MLM64plus noch aus einem weiteren Grund im Namen: im März

**Das wahrscheinlich meistbenutzte Werkzeug von allen, die ihren C64 selbst programmieren, ist ein Maschinensprache-Monitor. Obwohl die Bezeichnung dies nicht nahelegt, ist dieses Tool auch für BASIC-Programmierer interessant. Sei es zum Speichern oder Laden beliebiger Adreßbereiche mit Sprite- und Grafikdaten, sei es für so einfache, aber praktische Kleinigkeiten wie Directory und Disk-Befehle. Dieser Monitor kann sogar rechnen und in die Floppy gucken.**

1985 haben wir den von Pascal Dornier aus Zollikon/Schweiz geschriebenen MLM64 veröffentlicht. Dieser Monitor ist seit über zwei Jahren tagtäglich in der Redaktion im Einsatz. In den letzten Monaten hat Hajo Schulz alle uns bekannt gewordenen kleinen „Bugs“ beseitigt und über zehn neue Befehle eingebaut. Man mag einwenden, es sei nicht gerade ein Zeichen von Einfallsreichtum, ein Programm zu überarbeiten und noch einmal zu veröffentlichen. Nur: in dieses Programm sind dadurch die Erfahrungen zweijährigen professionellen Einsatzes geflossen. Und das wiegt man gelnde Phantasie allemal auf.

Speichern Sie eine der drei Monitor-Varianten (oder gleich alle drei) auf Ihre eigene Diskette ab. Geladen wird jede Version mit Sekundäradresse 1, also LOAD"NAME",8,1. Starten Sie den MLM64plus mit SYS Adresse, beispielsweise mit SYS 49152 die wohl am häufigsten benutzte Version C. (Diese Beschreibung bezieht sich im folgenden bei Adreßangaben nur auf diese Version, im Kasten „Drei gute Adressen“ sind die entsprechenden Werte für die anderen beiden Versionen nachzulesen.) Der Monitor meldet sich mit seinem Namen, zeigt die Inhalte der CPU-Register an und erwartet mit blinkendem Cursor Eingaben.

## Erstes Training

Zur Einübung kann man beispielsweise ein-tippen

'm a09e

und diese Zeile mit RETURN abschließen. (Das Zeichen vor dem 'm' wird über SHIFT und 7 erreicht.) Gezeigt wird einem ein Teil des BASIC-ROMs; dergestalt daß  
– zunächst ein Kürzel für die Zahlenbasis ausgegeben wird, in diesem Fall '\$', also hexadezimal

- man anschließend die Adresse erfährt, ab der die folgenden Ausgaben beginnen
- dann acht Bytes in hexadezimaler Form dargestellt sind
- und am rechten Bildschirmrand diese acht Bytes noch einmal in ASCII-Darstellung erscheinen. Mit der Beispieladresse sehen Sie einige BASIC-Befehle im Klartext.

## Von Farb-, Schwarzweiß- und Maschinensprache-Monitoren

Kaum ein Begriff hat so verwirrend viele Bedeutungen wie das Wort „Monitor“. Von dem computerfernen Inhalt „Studienüberwacher“ mal abgesehen, trägt auch die Begriffsbestimmung im DV-Metier selbst nicht gerade zur Klarheit bei. Immerhin gibt es Farb-, Schwarzweiß- und Maschinensprache-Monitore – Hardware das eine, Software das andere.

Wenn man sich auf die Kategorie „weiche Ware“ geeinigt hat, muß man sich längst nicht einig sein. Ein Monitorprogramm, ist in vielen DV-Lexika zu lesen, stellt nach dem Einschalten des Rechners grundlegende Funktionen zur Verfügung, wie Tastaturlauffrage, Bildschirmausgabe, Lesen und Beschreiben von Speicherstellen. So ein Monitorprogramm zu veröffentlichen, wäre für den C64 nun absolut überflüssig: nach dem Einschalten stehen nicht nur „grundlegende Funktionen“, sondern sogar ein betriebsbereites BASIC mit einem umfangreichen Betriebssystem zur Verfügung.

Doch in den letzten Jahren hat sich eine andere Definition von „Monitor“ in der Praxis durchgesetzt, und darunter fällt auch der MLM64plus. Neben BASIC braucht man ein Programm, mit dem man sich den Speicher unmittelbarer als durch 'PEEK' und 'POKE' ansehen und beschreiben kann. Außerdem möchte man Byte-Folgen suchen und Speicherbereiche vergleichen und verschieben können. Praktisch ist auch die Möglichkeit, definierte Adressbereiche laden und speichern zu können.

Und natürlich geht es um Maschinensprache. Und um das verbreitete Anfänger-Mißverständnis: „Wie kann ich Maschinenspracheprogramme LISTen?“ So

abwiegig dem DV-Spezi diese Frage klingen mag, so naheliegend ist sie eigentlich. Denn mit dem BASIC-Interpreter, der einen nach dem Einschalten des C64 in seiner ewigen Warteschleife empfängt, kann man Maschinensprache nicht sichtbar machen. Der LIST-Befehl erkennt nur BASIC-Zeilen. Wie also Maschinensprache sichtbar machen? Mit einem Unterprogramm der meisten Monitore, dem Disassembler. Der Disassembler liest nämlich die Inhalte des betreffenden Speicherbereichs und wandelt diese in entsprechenden Befehlstext, Mnemonics genannt, um. Beispielsweise die beiden Bytes '\$A9' und '\$00' in den Befehl 'LDA #00'.

Das funktionale Gegenstück ist der Assembler. Der wandelt eingegebene Befehle in hexadezimalen Code um und schreibt diesen in den Speicher. So kann man durchaus kleine Programme schreiben oder vorhandene Programme ändern; man wird also in die Lage versetzt, überhaupt Maschinbefehle einzugeben zu können.'

Soweit erst einmal zur Sprachregelung in Sachen Monitor. Man könnte etliche Punkte sicher noch ausführen. Aber das sollte hier eigentlich kein Maschinensprachekurs werden. Den gab's nämlich unter dem Namen „INPUT64-Assembler-Schule“ in den INPUT-Ausgaben 3 bis 8 dieses Jahres. JS

<sup>1</sup>Natürlich ist so ein Assembler, wie er im MLM64plus und anderen Monitoren eingebaut ist, nur ein unzureichendes Hilfsmittel. Ein „richtiger“ Assembler kann wesentlich mehr. Zum Beispiel Labels verarbeiten und an beliebigen Stellen Befehle einfügen. Dafür haben wir den Macro-Assembler INPUT-ASS (Ausgabe 6/86).

Der Monitor benutzt den Full-Screen-Editor des C64. Das heißt, man könnte jetzt mit dem Cursor auf die einzelnen Bytes wandern, eines ändern und mit RETURN diese Änderung durchführen. Das geht deswegen, weil der Monitor seine Ausgaben immer mit gültigen Eingaben beginnt. Das '\$'-Zeichen ist nämlich gleichzeitig der Befehl, mit dem an eine bestimmte Adresse ein bis acht Bytes geschrieben werden können.

## Eingaben über die Ausgaben

Naheliegend wäre, sich die Änderung jetzt mit einem erneuten „m a09e“ und RETURN anzusehen. Da wir es hier aber mit ROM zu tun haben, das bekanntlich nicht beschrieben werden kann, bliebe die Änderung wirkungslos. Man kann jetzt, weniger einer sinnvollen Anwendung wegen, sondern um die Bedienung des MLM64plus zu demonstrieren, zweierlei tun.

Möglichkeit 1: „t a000 a100 8000“ und RETURN eingeben; dann „m 809e“ und RETURN, mit dem Cursor hoch bis auf die '45' neben der Adresse \$809E; diese '45' mit '41' übertippen, mit RETURN eingeben, zwei Zeilen höher auf das „m 809e“ und dieses mit RETURN dem Monitor erneut zur Verarbeitung übergeben. Aus dem Wort 'end' ist 'and' geworden. 't' ist nämlich der Transfer-Befehl, mit dem in diesem Fall der Bereich von \$A000 bis \$A100 nach \$8000 verschoben wurde, im RAM an der Zieladresse konnten wir dann die Veränderungen durchführen. Als erwünschter Nebeneffekt ist deutlich geworden, daß der MLM64plus mehrere Parameter durch Leerzeichen getrennt sehen will, nicht etwa durch Komma.

Möglichkeit 2: Noch einmal „m a09e“ und RETURN eingeben, mit dem Cursor auf die Adreßangabe 'a09e' gehen, diese in '809e' und die nebenstehende '45' in '41' ändern, mit RETURN eingeben, sich mit „m 809e“ und RETURN das Ergebnis ansehen. Die er-

ste Zeile sieht genauso aus wie unter „Möglichkeit 1“ beschrieben. Man kann also beim '\$'-Befehl nicht nur bei der Byte-Ausgabe eigene Eingaben machen, sondern auch vorn bei der Adressausgabe. Was nicht geht: rechts direkt in den ASCII-Dump schreiben.

Das Prinzip, dem Monitor seine Wünsche mitzuteilen, dürfte jetzt klar sein. Ab sofort wird auch vorausgesetzt, daß der Leser daran denkt, alle Eingaben mit RETURN abzuschließen, da sie sonst wirkungslos bleiben.

## Kurz und knapp in drei Systemen

Für alle Befehle gelten, außer dem oben beschriebenen Full-Screen-Editor-Prinzip, bestimmte Regeln:

- Die Zahlenbasis ist nach dem Start des Monitors hexadezimal
- Zahlen können abgekürzt werden, so ist '\$8' gleichbedeutend mit '\$0008'.
- Als Trennzeichen sind ein oder mehrere Leerzeichen zulässig, zwischen Befehl und erstem Argument ist kein Trennzeichen notwendig.
- Die Kennungen für die Zahlensysteme sind \$=hexadezimal, @=dezimal, %=binär und als Sonderfall

' (Zeichen über der 7)=ASCII. „\$41“, „@65“,

## Die kleinen Unterschiede

Langjährige INPUT 64-Leser sind mit der Bedienung des Monitors wahrscheinlich schon im Schlaf vertraut. Extra für sie hier in geraffter Form die Unterschiede zwischen MLM64 und MLM64plus:

Neue dazugekommen sind die Befehle '>' (Diskbefehle, Directory), '0', '1' (Schreiben/Lesen von Floppy-Speicher), 'R', 'W' (Read/Write Track/Sektor), '+', '-' (Rechnen), '?' (Zahlensysteme wandeln), 'k' (Speicherkonfiguration einstellen).

Geändert hat sich die Ausgabe nach "m"; die Möglichkeit der Befehlwiederholung von 'd' und 'm', die Darstellung von wahlweise Sprite-/Zeichensatz-Mode durch '%' und die Parameter-Auswertung. Für letztere gilt bei allen Befehlen, die die Angabe von Start- und Endadresse verlangen: ist 'Ende' kleiner 'Start', wird 'Ende' als Anzahl interpretiert.

Entfallen ist die Prüfsummen-Option durch '+'. Die wesentlichste Änderung ist, daß mittlerweile die Aufrufadresse identisch mit der Startadresse ist.

„%01100101“ und „a“ sind also gleichbedeutend.

- Die Zahlenbasis ist umschaltbar durch Eingabe von nur '\$', '@' oder '%'. Zahlen in dieser bevorzugten Basis interpretiert der Monitor auch ohne Präfix richtig. 16-Bit-Zahlen können nicht binär aus- oder eingegeben werden, im Binär-Modus werden 16-Bit-Zahlen (Adressen) hexadezimal aus- und eingegeben.
- Eingegebene Zahlen dürfen nicht größer als \$FFFF (@65535) und nicht kleiner als Null sein.
- Für Befehle, die die Angabe einer Anfangs- und Endadresse verlangen, gilt: ist „Anfang“ größer „Ende“, so wird „Ende“ als Anzahl interpretiert, beginnend bei Null.

– Alle Ausgaben können mit der CTRL-Taste verlangsamt, mit der Leertaste angehalten, mit der C-Taste fortgeführt und mit RUN-STOP abgebrochen werden.

– Groß/Kleinschreibung ist signifikant: 'R' ist nicht gleich 'r'. Großbuchstaben beziehen sich in der Regel auf Operationen mit der Floppy.

– Falls Sie mal eine dieser Regeln mißachten oder einen falschen Befehl eingeben, zeigt Ihnen ein inverses Fragezeichen in der Eingabezeile den ungefähren Fehlerort an.

## Abkürzungen abgemacht

Für die Beschreibung der Befehle gelten folgende Vereinbarungen:

## MLM64plus-Befehlsübersicht

\$ [aa b0..b7]	Umschalten auf hexadezimal, 'POKE' Hexwert	h aa ea b0.bn	Hunt: nach Byte-Muster 'b0.bn' suchen
% [aa b0..b7]	Umschalten auf binär, 'POKE' Binärwert	l "name" dev [aa]	Laden eines Programms
+ wert1 wert2	Addieren	m [aa [ea]]	Memory-Dump: Speicherinhalt ausgeben
- wert1 wert2	Subtrahieren	p [befehl]	Protokollieren auf Datei
>[diskbefehl]	Diskbefehl senden	q [aa]	'Quick'-Trace
> \$	Directory	r	Register ausgeben
? wert	Zahlensystemwandlung	s "name" dev aa ea	Speichern eines Adreßbereiches
@ [aa b0..b7]	Umschalten auf dezimal, 'POKE' Dezimalwert	t aa ea za	Transfer: Speicherbereich verschieben
a aa : befehl	Assemblermodus	v "name" dev [aa]	Verify
b [aa]	Breakpoint 'n' nach 'aa'	x	Rückkehr zum BASIC
c aa ea za	Compare: Speicherbereiche vergleichen.	y [aa]	Wie 'g', aber 'RTS' geht in Monitor
d [aa [ea]]	Disassemblieren	I floppyadr c64addr	Liest Floppy-Speicher in Rechner
e [aa]	Einzelzrittzbetrieb	O c64addr floppyadr	Schreibt Rechner-Speicher in Floppy
f aa ea b	Fill: Speicherbereich füllen	R track sektor [aa]	Liest Block von Diskette
g [aa]	Go: Programm ab 'aa' oder PC starten	W track sektor [aa]	Schreibt Block auf Diskette

## Quellen-Service

Vielleicht möchten Sie das eine oder andere im Monitor ändern oder einfach nur wissen, wie man so'was programmiert. Oder Sie haben Geschmack an Maschinensprache gefunden und möchten sich eine komplette „Werzeugkiste“ zulegen, mit allem, was dazugehört.

Dann haben wir für Sie für 49,- DM (zuzüglich 3,- DM Versandkosten) eine doppelseitig bespielte Diskette mit dem in Ausgabe 6/86 veröffentlichten Macro-Assembler INPUT-ASS, diesem vorgestelltem Maschinensprache-Monitor, Library-Programmen zu I/O-Bedienung, Arithmetik und so weiter – und das alles im Source-Text.

aa: Anfangsadresse

ea: Endadresse

za: Zieladresse

b(n): Byte(s)

Parameter in eckigen Klammern ([]) sind optional, das heißt, sie können, müssen aber nicht eingegeben werden. Die folgende Übersicht ist nach Funktionsgruppen ge trennt.

### Sehen, suchen, ändern

In der Gruppe „Sehen, suchen, ändern“ sind sozusagen die Basis-Funktionen eines Monitors zusammengefaßt; alle Befehle, mit denen man sich Speicherinhalte anschauen, diese verändern, kopieren, vergleichen und so weiter kann.

c aa ea za Die Bereiche von 'aa' bis 'ea' werden mit einem Bereich gleicher Länge ab 'za' verglichen. Alle Unterschiede werden angezeigt. Eine häufige Anwendung ist der Vergleich verschiedener Programmversionen, die man sich an verschiedene Adressen ins RAM lädt. Beispiel:

c 1000 2000 3000

f aa ea b Füllt von Anfangsadresse 'aa' bis Endadresse 'ea' den Speicher mit dem Byte 'b'. Beispiel:

f 400 700 \*

**h aa ea b0 ... bn** Hunt-Befehl, zu deutsch Suchbefehl. Der angegebene Speicherbereich wird nach der durch die Bytes b0 ... bn definierten Zeichenfolge durchsucht, alle Fundstellen werden ausgegeben. Beispiele:

```
h a000 b000 45 'n 'd  
h a000 b000 20 d2 ff
```

**k [b]** Konfiguration für Speicherzugriff einstellen. 'b' enthält den Wert, der vor einem Speicherzugriff ins Register 1 geladen wird. Die eingestellte Konfiguration bezieht sich auf die Befehle 'm' und 'd' mit den dazugehörigen Änderungen sowie auf 'f', 't', 'c' und 'h'. Bei 't' und 'c' wird nur die Quelle beeinflußt.

'k' ohne Parameter zeigt die aktuelle Konfiguration ein. Siehe dazu auch den Kasten „Doppelt und dreifach“. Ein direktes Beschreiben der Register 0 und 1 ist deswegen tabu! Das folgende Beispiel schaltet das BASIC-ROM ab und macht den „darunter“ liegenden Adreßraum zugänglich:

k\$36

**m [aa [ea]]** Memory-Dump ausgeben, 22 Zeilen zu je acht Bytes ohne Parameter: 22 Zeilen ab Adresse 'aa' oder den Bereich von 'aa' bis 'ea'. Ein vorangestelltes ":" (wie schon erwähnt, das Zeichen über der ':') gibt rechts neben jeder Zeile die Bytes noch einmal in ASCII-Form aus. Steuerzeichen beziehungsweise Bildschirmcode werden als inverse Kleinbuchstaben angezeigt. Am Ende der Ausgabe wird ein 'm' in die folgende Bildschirmzeile geschrieben, so daß nur durch RETURN der m-Befehl mit den nächsten Adressen fortgesetzt wird Beispiel:

m 80 10

**t aa ea za** Transfer-Befehl. Überträgt die Daten von der Anfangsadresse 'aa' bis zur Endadresse 'ea' zur Zieladresse 'za'. 'za' darf nicht zwischen 'aa' und 'ea' liegen. Beispiel:

t 400 500 600

**\$ [aa b0 ... .b7]** Dieses beim Memory-Dump den Ausgaben vorangestellte Zeichen ist gleichzeitig der Befehl zur Umschaltung der Zahlenbasis auf hexadezimal und ein besonderer 'POKE'-Befehl. Mit Parametern wer-

den ab Adresse 'aa' die durch 'b0' bis 'b7' definierten ein bis acht Bytes geschrieben. Dieser Befehl wird in der Praxis selten zum Verändern von Speicherinhalten benutzt, meist läßt man sich durch 'm' den Speicherinhalt ausgeben und editiert dann den Dump. Beispiele:

```
$  
$ 8000 30 31 32 33
```

**% [aa b0..b7]** Umschaltung der Zahlenbasis auf binär. Beim m-Befehl wird nur ein Byte pro Zeile ausgegeben – praktisch für die Zeichensatzdarstellung. Ein nochmaliges '%' ohne Parameter schaltet um auf die Darstellung von drei Bytes pro Zeile – so kann man sich Sprites ansehen. Wiederholtes '%' schaltet zwischen diesen beiden Darstellungsmöglichkeiten, die sich nur auf die Bildschirmausgabe beziehen, um Alles andere wie '\$':

### Ein bißchen programmieren

Die Methode, Programme direkt mit einem Monitor „einzuhacken“, ist zwar endgültig out. Aber ein Disassembler und ein Zeilenassembler sind zum Analysieren und Entwickeln von Maschinenprogrammen unumgänglich.

**a aa :befehl** Assemblermodus. Mit diesem Befehl können Maschinenprogramme im 6502-Mnemonic-Format eingegeben werden. Nach Eingabe einer Zeile gibt der Monitor die nächste Adresse aus und bleibt im Assemblermodus, bis eine fehlerhafte Zeile oder eine Leerzeile eingegeben wird. Der Doppelpunkt ist zwingend. Mit dem d-Befehl disassemblierte Programme können editiert werden, da Hexdump, wie es dann zwischen Adresse und Doppelpunkt steht, ignoriert wird. So ist auch die Verschiebung von kurzen Programmstücken möglich, indem beim ersten Befehl vorn die Adresse verändert und dann mit RETURN durchgegangen wird. Natürlich müssen die Sprungadressen jeweils angepaßt werden. Das Leereichen zwischen Adresse und Doppelpunkt ist zwingend. Beispiel:

a 1000 :jmp 1000

**d [aa [ea]]** Disassembliert ohne Parameter 22 Zeilen ab aktuellem Programmzähler beziehungsweise von 'aa' bis 'ea'. Ausgege-

ben werden Hexdump und die üblichen 65xx-Mnemonics, illegale OpCodes ergeben drei Fragezeichen. Die Befehlwiederholung funktioniert analog zum m-Befehl. Beispiel:

d c000 c07f

## Emulieren und Entwanzen

Die folgende Befehlsgruppe dient dem Testen von und der Fehlersuche in Programmen.

Beim Start dieser Programme unter Kontrolle des Monitors werden die Register mit vom Benutzer bestimmten Werten geladen und das so gestartete Programm ausgeführt. Der Programmierer muß selbst sicherstellen, daß keine Arbeitsadressen des Monitors oder gar der Monitor selbst überschrieben werden. Deswegen ist vor dem „Debuggen“ ein Blick in den Kasten zum Thema „Speicherbelegung“ dringend angeraten.

**b [n aa]** Der Befehl 'b' ohne Parameter zeigt die Breakpoints an. Breakpoints sind die maximal vier setzbaren „weichen“ Halte-

punkte, die von den Trace-Routinen (siehe 'e';'q') erkannt werden. 'n' kann Werte von Null bis Drei annehmen und bezeichnet die Nummer, 'aa' die Adresse des Breakpoints. Gelöscht wird ein Breakpoint durch 'aa' gleich Null. Beispiel:

b 3 1003

**e [aa]** Einzelschrittbetrieb eines Programmes ab Programmzähler oder ab 'aa'. Dieses Programm wird ausgeführt, und nach jedem Befehl werden alle Register angezeigt und der Befehl disassembliert. Ein Breakpoint (siehe 'b'), ein illegaler Befehl oder die RUN/STOP-Taste führen zum Abbruch. Ein RTS-Befehl, dem kein vorheriges JSR im getesteten Programm entspricht, führt zu einem Einzelschrittbetrieb des Monitors selbst. Das geht meist nicht sehr lange gut. Beispiel:

e 2000

**g [aa]** Go-Befehl: das Maschinenprogramm ab 'aa' oder dem aktuellen Programmzähler starten, die Prozessor-Register werden mit den über '\*' beziehungsweise 'r' gesetzten Werten geladen. Zurück in den Mo-

nitor führt nur ein BREAK-Befehl im exekutierten Programm. Mit dem Go-Befehl kann man Programme in Echtzeit testen. Dieser Befehl sollte, genauso wie 'e', 'q' und 'y', nur mit etwas Überlegung eingesetzt werden, da es naturgemäß keine (sinnvolle) Möglichkeit gibt, den Monitor oder wichtige Adressen des Monitors vor Überschreiben zu schützen. Zu beachten ist auch, daß ein RTS-Befehl, dem kein JSR entspricht, sich die nächstbeste Adresse vom Stack holt und diese als Rücksprungadresse interpretiert. Beispiel:

g 1000

**q [aa]** Sogenanntes „Quicktrace“. Das Programm ab der angegebenen Adresse, beziehungsweise bei fehlender Parameterangabe ab aktuellem Programmzählerstand, wird kontrolliert gestartet. Genauer gesagt, das Programm wird Befehl für Befehl emuliert, so daß auch ROM-Routinen „getraced“ werden können. Ein Abbruch erfolgt durch gesetzte Breakpoints (siehe b-Befehl), einen illegalen oder den BRK-Befehl oder die RUN/STOP-Taste. Es wird als „History“ ein Disassembly der letzten vier Befehle vor Programmabbruch und der aktuelle Registerstand ausgegeben. Zu auftauchenden Problemen siehe auch 'g!'. Beispiel:

q a400

**r** Register anzeigen. Programmzähler, Register, Stackpointer und Status werden ausgegeben, letzterer immer binär. Es handelt sich dabei logischerweise nicht um Programmzähler und Register des Monitors selbst, sondern um die der mit 'e', 'q', 'g' und 'y' angesprochenen Programme. Die Adresse unter der Bezeichnung 'BRK' ist die des Break-Vektors (\$0316) und bezeichnet normalerweise den Einsprung im Monitor für die Service-Routine, wenn der Prozessor auf ein Null-Byte (BRK) läuft. Alle ausgegebenen Werte kann man durch Überschreiben und RETURN ändern, da die Zeile mit '\*' beginnt (siehe '\*').

**y [aa]** Fast der gleiche Befehl wie 'g', mit dem Unterschied, daß der Stackpointer nicht gesetzt wird, so daß ein RTS in den Monitor zurückführt. Beispiel:

y a3b7

**\* aa b b b b %b aa** Setzt die Register. Der Befehl ist hier nur aus Konsistenz-Gründen aufgeführt, da er allein nicht benutzt wird.

k-Wert \$8000-\$9FFF \$A000-\$BFFF \$C000-\$CFFF \$D000-\$DFFF \$E000-\$EFFF

\$37	evtl. Modul (ROM)	BASIC- ROM	RAM	I/O-Ports Farb-RAM	Kernal- ROM
\$36		RAM		I/O-Ports Farb-RAM	Kernal- ROM
\$35		RAM		I/O-Ports Farb-RAM	RAM
\$34		RAM			
\$33	evtl. Modul (ROM)	BASIC- ROM	RAM	RAM	Kernal- ROM

Diese RAM/ROM-Konfigurationen können durch den k-Befehl des Monitors eingestellt werden. Die Aufteilung gilt für lesende Zugriffe; geschrieben wird prinzipiell ins RAM. Ausnahme: sind die I/O-Ports eingeschaltet, werden diese beschrieben, nicht das darunter liegende RAM. Für den C128 gelten alle k-Werte mit zusätzlich gesetztem Bit 6, beispielsweise \$77 statt \$37.

sondern statt dessen die mit 'r' ausgegebene Register editiert werden.

## Sicher sichern, locker laden

Ein paar Befehle braucht man natürlich auch, um seine Daten mit der Außenwelt auszutauschen und nicht-flüchtig auf Diskette zu lagern. Mehr noch: Direktzugriffe auf Diskette und Floppy-RAM sind erlaubt. Alle Befehle, die sich ohne Angabe einer Gerätenummer auf die Floppy beziehen, gehen von Gerätenummer 8 aus. Diese Voreinstellung ist in Anfangsadresse+5 des Monitors hinterlegt und somit veränderbar.

**I "name" d [za]** Lädt das Programm namens "name" vom Gerät mit der Nummer d in den Speicher, gegebenenfalls an die durch 'za' vorgegebene Adresse. Fehlt 'za', wird mit Sekundäradresse 1 geladen, also an die auf Diskette oder Kassette (8 bzw. 1) eingetragene Adresse. Nach dem Laden gibt der Monitor Anfangs- und Endadresse aus. Beispiel:

I "test" 8

## Drei gute Adressen: Speicherbelegung der Monitore

### Version C

\$C000-\$CFFF Monitor-Programm  
\$9FOO-\$9FFF Default Block Buffer  
Start: SYS 49152

### Version 9

\$9000-\$9FFF Monitor-Programm  
\$CF00-\$CFFF Default Block Buffer  
Start: SYS 36864

### Version 6

\$6000-\$6FFF Monitor-Programm  
\$C000-\$COFF Default Block Buffer  
Start: SYS 24576

### Zero-Page-Belegung

\$57, \$58, \$65 bis \$70.  
Adresse \$00 sollte nicht, Adresse \$01 kann nicht verändert werden, da der MLM64plus sie für den k-Befehl benutzt.

**s "name" d aa ea** Save-Befehl. Der Speicherbereich von 'aa' bis 'ea' wird auf Gerät 'd' geschrieben. Beispiel:

s "grafikbild" 8 @8192 @16191

**v "name" d [aa]** Verify. Alle Parameter wie beim I-Befehl, nur eben statt zu Laden ein Vergleich. Beispiel:

v "hase" 8 1111

**p [Befehl]** Protokollieren. Die Ausgaben werden auf eine mit Kanalnummer 4 geöffnete Datei geschrieben. Falls dieses File noch nicht vorhanden ist, wird OPEN4,4,7 ausgeführt, die Ausgabe also auf den Drucker geleitet. 'p' ohne Parameter bewirkt einen Zeilenvorschub auf dem Drucker. Man kann vor dem Start des Monitors ein File eröffnen (etwa: OPEN4,8,2,"PROTO,S,W") und so ein Disassemblerlisting auf Diskette schreiben. Nach dem Verlassen des Monitors ist es zwingend notwendig, das File „von Hand“ zu schließen. Eine so geschriebene sequentielle Datei kann beispielsweise mit dem Macro-Assembler INPUT-ASS (Ausgabe 6/86) gelesen und weiterverarbeitet werden. Die Sekundäradresse und die Gerätenummer des Druckers stehen in Anfangsadresse+7 beziehungsweise Anfangsadresse+6 des Monitors und können so gegebenenfalls angepaßt werden. Beispiel:

pd c000

**R Track Sektor [aa]** Liest den durch 'Track' und 'Sektor' festgelegten Block an die Adresse 'aa' in den Speicher oder bei fehlender Adreßangabe in einen voreingestellten Puffer, dessen Adresse Sie dem entsprechenden Kasten („Drei gute Adressen“) entnehmen können. Diese Adresse ist übrigens auch im Low-/High-Byte-Format ab Anfangsadresse+3 der Monitore hinterlegt und kann verändert werden. Nicht mit 'r' verwechseln! Beispiel:

R 12 01 8000

**W Track Sektor [aa]** Schreibt ab Adresse 'aa' oder ab Puffer (siehe 'R') 256 Bytes in den durch 'Track' und 'Sektor' festgelegten Block auf Diskette. Beispiel:

W 12 07 8000

**>[Disk-Befehl]** Sendet einen Befehl an die Floppy. Ist dieser Befehl '\$', wird das Di-

rectory angezeigt. '>' ohne Parameter liest den Fehlerkanal aus. Beispiele:

>\$

>r:nameneu=namealt

**Q Quelladresse Zieladresse Anzahl „Output“** zur Floppy. Anzahl darf zwischen 1 und 35 sein, 'Anzahl' Bytes werden ab Rechneradresse 'Quelladresse' in das Floppy-RAM ab 'Zieladresse' geschrieben. Beispiel:

O 1000 @119 1

**I Quelladresse Zieladresse Anzahl „Input“** von der Floppy. 'Anzahl' Bytes werden aus dem Floppy-Speicher ab 'Quelladresse' in den Rechner nach 'Zieladresse' gelesen; 'Anzahl' darf zwischen 1 und 35 sein. Nach der Ausführung des Befehls – wie auch nach 'R' – ist die Standardadresse für 'm' und 'd' auf den Anfang des geladenen Bereiches eingestellt. Beispiel:

I @119 2000 2

## Individuell, aber nützlich

Manches läßt sich eben einfach nicht in vorgefertigte Rubriken pressen. Die folgenden Befehle zum Beispiel:

**x** Dieser Befehl bewirkt die Rückkehr zum BASIC. Kein Beispiel!

**+ wert1 wert2** Gibt die Summe von 'wert1' und 'wert2' im aktuellen Zahlenformat aus. Beispiel:

+ \$800 @90

**- wert1 wert2** Gibt die Differenz von 'wert1' und 'wert2' aus. Beispiel:

- ff fe

**? wert** Gibt den Wert in allen drei Zahlenformaten aus.

## Rückgriffe

Die Dezimal-Umwandlungsroutine – auch schon die des 85er Monitors – stammt aus mc 3/81<sup>1</sup>; die Syntaxprüfung ist durch einen „Syntax-Interpreter“ gelöst, wie man ihn im ROM des ZX-81 findet<sup>2</sup>. JS

<sup>1</sup> F. Laher, SED hilft bei Binär-Dezimal-Umwandlung, mc 3/81

<sup>2</sup> P. Dornier, ZX-81-ROM-Listing, Eigenverlag



**ID-Werkstatt:**

# Für europäische Nachbarn

**Vokabeltrainer lernt französisch**

Eigene Zeichensätze können mit dem Programm CHARMAKER in das im März dieses Jahres bei uns erschienene Vokabel-Lern-Programm „Vokabeltrainer“ eingebunden werden. Es geht dabei natürlich vor allem um nationale Sonderzeichen, deutsche Umlaute etwa, Accent circonflexe, Tremu und so weiter. CHARMAKER wurde freundlicherweise von Oliver Kraus, der auch den Vokabeltrainer selbst geschrieben hat, für die ID-Werkstatt zur Verfügung gestellt.

## Mit französischem Akzent

Einen französischen Zeichensatz bekommen Sie frei Haus mitgeliefert, er steht unter dem Namen FRANCE in der ID-Werkstatt zum Abspeichern bereit. Der Vokabeltrainer lässt aus programmtechnischen Gründen nicht mehr als neun Sonderzeichen zu, wir haben uns für die Auswahl der gebräuchlichsten entschieden: à, ê, î, ô, û, ç, è, à, und é. Alle Buchstaben mit Circonflexe sind über die gleichzeitige Betätigung der Commodore-Taste und des jeweiligen Buchstabens zu erreichen, dasselbe gilt für

Einen häufig geäußerten Leserwunsch können wir mit dieser Ausgabe der ID-Werkstatt erfüllen: der in INPUT 64, Ausgabe 3/87 veröffentlichte Vokabeltrainer bekommt Sonderzeichen.

ç, eine Sonderrolle spielen die Zeichen mit Accent:

é entspricht © und £  
â entspricht © und \*  
ê entspricht © und †

Will man seinen Vokabeltrainer auf diesen Zeichensatz umrüsten, geht man nach dem Abspeichern von CHARMAKER und FRANCE folgendermaßen vor: CHARMAKER laden und mit 'RÜN' starten, Diskette mit dem Original-Vokabeltrainer einlegen, wie gewünscht den alten Namen eingeben. Dann FRANCE nachladen lassen, die Frage nach dem Ersetzen des Doppelpunkts durch ein Leerzeichen mit 'Ja' beantworten (denn der Doppelpunkt gilt als Sonderzeichen) und

den neuen Vokabeltrainer unter einem sinnvollen neuen Namen abspeichern lassen.

## Eigeninitiative

Beim Einbinden eigener Zeichensätze ist folgendes zu beachten: Der Vokabeltrainer arbeitet immer im Groß/Kleinschrift-Modus, deshalb wird nur auf die zweite Hälfte des Zeichensatzes zugegriffen. Der CHARMAKER erwartet auf Diskette auch nur diese zweite, von der Adreßlage her betrachtet „obere“ Hälfte des Character Set. Deswegen belegt FRANCE nach dem Abspeichern auch nur neun Blöcke auf Diskette, ein kompletter Zeichensatz wäre doppelt so lang. Ein von einem Zeichensatzeditor<sup>1</sup> erstellter eigener Zeichensatz sollte also vorher aufgeteilt werden. Der für Groß/Kleinschrift zuständige Teil der Daten beginnt bei Basisadresse plus \$0800. JS

<sup>1</sup>Etwas der INPUT-SCE (6/87) oder der im Zusammenhang mit dem Video-Chip-Kurs im Februar 1985 veröffentlichte Zeichensatzeditor.

# Wer antwortet, gewinnt!

## Spiel: Das große Quiz

An diesem Spiel können zwei oder drei Kandidaten teilnehmen, die jeweils drei Runden durchstehen müssen. In der ersten Runde werden jedem Kandidaten – der Name steht immer in der Kopfzeile auf dem Bildschirm – vier Fragen gestellt, von denen er sich eine als Superfrage aussuchen kann (1–4). Wird diese dann richtig beantwortet, bekommt er 600,- DM dafür gutgeschrieben. Für jede andere richtig beantwortete Frage gibt es 200,- DM.

### Drei Kandidaten für drei Runden

Nachdem jeder Mitspieler seine vier Fragen beantwortet hat, geht es in die zweite Runde. Es werden fünf Wissensbereiche in fünf Spalten zu je vier Zeilen angezeigt. Darunter stehen die Namen der Kandidaten und ihre jeweilige Punktezahl. Links neben den Namen zeigt ein kleines Feld an, welcher Spieler aus den zwanzig Fragefeldern eines auswählen darf. Soll zum Beispiel aus dem Feld „Natur 60“ eine Frage gestellt werden, geben Sie „N6“ ein und bestätigen die Eingabe mit der RETURN-Taste. Erscheint jetzt eine Frage, kann jeder Mitspieler so schnell wie möglich seine Taste drücken (siehe Tabelle). Der schnellste bekommt den Zuschlag und darf die Frage beantworten.

Spieler Nr.	Taste	Joystick
1	←	Port 1
2	£	Port 2
3	N	–

Jedem Spieler seine eigene Taste.

**Die Quiz-Reise geht von Vaduz über Lissabon nach Tirana, so ungefähr jedenfalls. Aber nicht nur in der Geographie sollten Sie bewandert sein, sondern auch in Sachen Sport, Kfz-Technik, Natur und Musik. „Das große Quiz“, geschrieben von Thomas Langenkamp, fordert Ihnen einiges an Allgemeinwissen ab.**

2. Auf dem Bildschirm erscheint ein „Glücksschwein“ und beschert Ihnen 100,- DM.

3. Sie bekommen eine Gewinnfrage gestellt, für die es bei richtiger Beantwortung 500,- DM gibt.

4. Sie erhalten die „CHANCE“, Ihren Einsatz selber festzulegen. Er darf aber nicht höher sein, als Sie Punkte auf Ihrem Punktekonto haben.

Die Auswahl ist dabei zufällig. Es kann Ihnen also passieren, daß die eine Möglichkeit zehnmal und die andere nur einmal vorkommt.

Ist die Antwort falsch, werden die vorher festgelegten Punkte abgezogen und die Frage wieder freigegeben. Der Rest der Mitspieler kann sich jetzt in gleicher Weise um den Zuschlag bewerben. Beantwortet der Spieler die Frage richtig, erhält er die Punktzahl des jeweiligen Feldes und kann das nächste Feld bestimmen.

### Chancen, Schweine und Gewinne

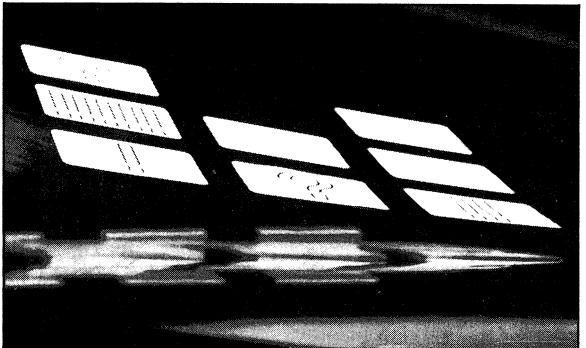
Wird eine Frage gestellt und es drückt keiner der Beteiligten auf seine Taste, kann sie mit der „f1“-Taste wieder zurückgenommen werden. Punkte werden dann nicht abgezogen.

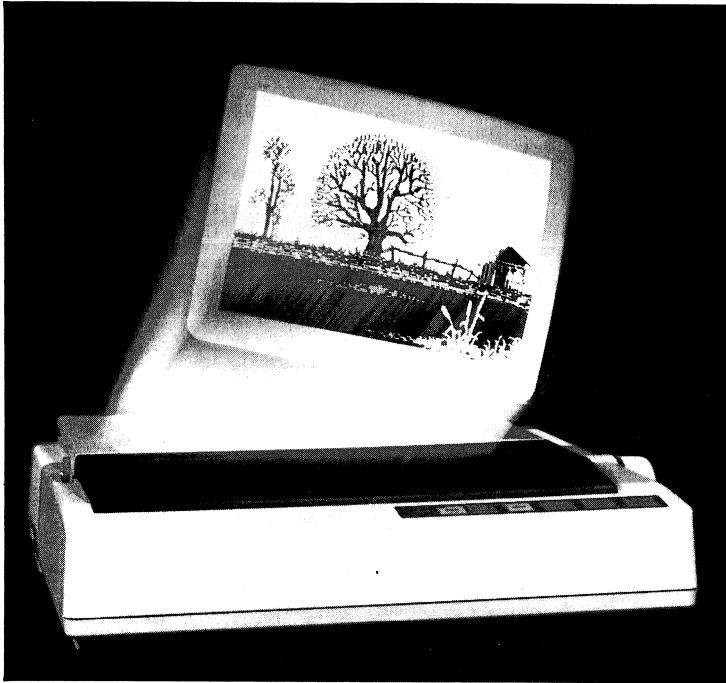
Bei der Auswahl eines Feldes müssen Sie mit folgenden Möglichkeiten rechnen:

1. Sie müssen eine ganz normale Frage beantworten.

Auf dem Feld „A-Z 6“ steht keine Zahl, sondern ein Fragezeichen, das immer als letztes ausgewählt werden muß. Dabei sollten Sie den Lautstärkeregler Ihres Fernsehers oder Monitors nicht zu leise gestellt haben, denn es ertönen an dieser Stelle die ersten Takte einer Nationalhymne, die es zu erraten gilt. Nachdem alle Felder ausgewählt sind, geht es zur dritten und letzten Runde.

Die Kandidaten werden der Reihe nach aufgefordert, eine dreiteilige Frage zu beantworten. Dazu kann man einen von drei Bereichen wählen. Zur Beantwortung dieser dreiteiligen Frage haben Sie sechzig Sekunden Zeit. Die drei Antworten müssen durch ein Semikolon ohne Leerzeichen getrennt sein (Eskimo;Iglu;Fische). Inspiriert wurde der Autor zu diesem Spiel durch das Fernsehquiz „Der große Preis“. kfp





# Mixed Pixels

## Multi-Hardcopy für Text, Sprites und Grafik

Wenn man sich den tristen Textbildschirm durch ein paar Sprites aufgelockert hat, hört die Begeisterung meist dann auf, wenn der Bildschirminhalt zu Papier gebracht werden soll: von Sprites weit und breit keine Spur. Denn die gängigen Hardcopy-Programme drucken den Inhalt des Textbildschirms sozusagen „im engeren Sinne“: alles, was sich im Video-Speicher des C64 befindet. Die Informationen für Sprites werden bekanntlich getrennt abgelegt und deswegen „übersehen“.

Ein ähnliches Problem stellt sich bei der gemischten Darstellung von hochauflösender Grafik und Sprites. Zum Drucker geschickt

**Hardcops vom normalen Textbildschirm sind ein alter Hut, Grafik-Hardcops reißen auch keinen mehr vom Hocker, Sprites werden schon seltener zu Papier gebracht. Neuen Schwung in die Hardcopy-Riege bringt „Multi-Hardcopy“ – das kann nämlich alles zusammen, und zwar gleichzeitig.**

werden die 8000 Bytes des Grafikspeichers, eventuell vorhandene Sprites bleiben unberücksichtigt.

## Eigenintelligenz

Bei „Multi-Hardcopy“ ist nun folgende Idee in die Tat umgesetzt worden: Das genaue Aussehen des Bildschirms – also ob hochauflösende Grafik oder Text dargestellt wird, ob Sprites eingeschaltet sind und, wenn ja, wie viele und an welcher Bildschirmposition – ermittelt das Programm selbst aus den entsprechenden Videochip-Registern. Der Ausdruck erfolgt prinzipiell im Grafikmodus; ein Textbildschirm wird vorher in hochauflösende Grafik gewandelt. Die Daten eventuell entdeckter Sprites werden in den auszudruckenden Grafikspeicher hineinkopiert.

Soweit zur prinzipiellen Funktionsweise. Das Programm ist angepaßt für die meistverbreiteten Drucker in der 64er-Szene, den MPS801 und den MPS803 von Commodore. Inwieweit das Zusammenspiel mit Druckern anderer Hersteller klappt, hängt davon ab, inwieweit diese beziehungsweise deren Interfaces in der Lage sind, das Verhalten der Drucker der MPS-Serie zu simulieren. Mit einem Star NL10 zum Beispiel klappte der Ausdruck auf Anhieb, von dem Problem der leicht ovalen Kreise einmal abgesehen.

## In Wartestellung

Die Bedienung ist denkbar einfach. Nach dem obligatorischen Abspeichern auf die eigene Diskette wird die Multi-Hardcopy von dort geladen und mit RUN gestartet. Sichtbar geschieht daraufhin gar nichts, das Programm wartet nun darauf, daß jemand die Tastenkombination CTRL und 'B' drückt. Dann geht's los.

Zwei Dinge kann das Programm nicht: Ein durch Rasterzeilen-Interrupt „gesplitteter“ Bildschirm wird nicht ebenso „gesplittet“ wiedergegeben. Und die in dem Grafikbildschirm hineinkopierten Sprites werden nach dem Druck nicht wieder entfernt.

Belegt wird – nach dem Start durch RUN – nur der Adressbereich außerhalb des BASIC-Speichers: das Programm selbst braucht die Adressen von \$C000 bis

```

10 v=53248:h=56576
20 poke v+17.59
30 for a=3 to 0 step-1
35 : poke53280.1
45 : poke h,peek(h) and 252 or a
50 : poke v+24.29
60 : poke 648,(3-a)*64+4
70 : print chr$(147):
80 : poke 198,0:wait 198.1
90 next
95 poke h,peek(h) and 252 or 3
99 sys65409

```

Diese BASIC-Zeilen durchsuchen den Speicher nach Grafikbildern.

\$C052 (dezimal 49152 bis 50434), vor dem Druck wird eine HiRes-Seite entweder im RAM parallel zum BASIC-ROM (\$A000 bis \$BFFF/40960 bis 49151) oder im RAM parallel zum Kernel-ROM (\$E000 bis \$FFFF/57344 bis 65535) angelegt, je nachdem, welcher Block frei ist; außerdem dient das RAM von \$D000 bis \$D300 (53248 bis 54016) als Zwischenspeicher. Die CTRL-B-Abfrage läuft innerhalb der Interrupt-Rou-

tine durch „Umleitung“ des Vektors auf die Tastatur-Auswertung.

Durch 'RUN-STOP/RESTORE' wird die Abfrage nach CTRL-B „abgehängt“. Ein Einbinden der Hardcopy ohne Neustart ist durch POKE 655,0: POKE 656,192 möglich.

Die Arbeitsadresse des Programms ist übrigens in der letzten Zahl der BASIC-Programmzeile festgelegt und kann verändert werden.

## Bilder-Klau

So weit, so gut. Damit ist natürlich nicht das Problem gelöst, wie man beispielsweise aus Spielen oder anderen fremden Programmen die Grafikbilder auf den Drucker kriegt. Dazu ist zunächst festzustellen, daß es ohne unverhältnismäßig großen Aufwand nicht möglich ist, Sprites und Grafik schwarz auf weiß festzuhalten. Das funktioniert auch mit Multi-Hardcopy nur bei eigenen Programmen oder solchen, bei de-

nen man sicher ist, daß sie nicht die oben erwähnten Speicherbereiche überschreiben. In diesen Fällen wird Multi-Hardcopy vorher geladen und gestartet und dann im richtigen Augenblick CTRL-B gedrückt.

Für Fremdprogramme benötigt man meist einen Reset-Taster, um das Spiel abbrechen zu können, wenn das gewünschte Bild erscheint. Diese Stecker sind für wenige DM im Fachhandel erhältlich. Nach dem Reset lädt und startet man Multi-Hardcopy. Mit dem aus wenigen Zeilen bestehenden BASIC-Programm (siehe Kasten) kann man den Speicher nach Grafikbildern durchsuchen, die dann mit CTRL-B ausgedruckt werden können. Das Programm wählt die vier gängigsten Adressen, an denen HiRes-Bilder liegen können, aus. Falls diese Methode nicht zum Erfolg führt, handelt es sich bei dem gesuchten Bild wahrscheinlich um einen Textbildschirm mit verändertem Zeichensatz. JS

## Tips zum Speedcompiler

# Was tun, wenn es brennt?

## Teil 1: Fehlermeldungen

**BASIC-Programme durch einen Compiler zu etwas schnellerer Arbeit zu bewegen, ist nur die eher triviale Hälfte des Themas „Compiler-Einsatz in Theorie und Praxis“. Um wirklich das Letzte aus diesen Programmbeschleunigern herauszuholen, muß man schon etwas tiefer einsteigen. Diese andere, nicht ganz triviale Hälfte thematisiert die vom „Vater des Speedcompilers“ geschriebene Serie.**

Der Speedcompiler kennt etwa 50 Fehlermeldungen. Einige davon sind compilerspezifisch, es gibt aber auch etliche, für die der Interpreter nur das wenig aussagekräftige „Syntax Error“ kennt. Dadurch erhält der Programmierer vom Compiler wesentlich mehr Unterstützung als vom BASIC-Interpreter.

Als Hilfestellung habe ich im folgenden Text nach jeder Fehlermeldung den Pass angegeben, in dem die einzelnen Fehlermeldungen auftreten: '(1)' hinter der Fehlermeldung bedeutet Pass 1, '(2)' Pass 2.

## Variabel nach Maß

**String-Formel zu komplex(1):** Der string-verknüpfende Ausdruck in der angegebenen Zeile ist für die Verarbeitung durch den Runtime-Interpreter zu komplex. Um eine möglichst schnelle String-Verarbeitung zu gewährleisten, mußte die Fähigkeit, komplexe String-Formeln aufzulösen, reduziert werden. Abhilfe: eine so reklamierte String-Formel aufteilen. Beispiel:

110 A\$=B\$+MID\$(VAL(C\$),2,255)

muß man aufteilen zu:

110HILF\$=VAL(C\$):A\$=B\$+MID\$(HILF\$,2,255)

**Wiederholte Felddimensionierung(1):** Ist identisch mit dem „REDIM'D ARRAY“ des Interpreters: ein Array darf nur einmal durch den DIM-Befehl dimensioniert werden. Diese Fehlermeldung kann außerdem als Reaktion auf Konstruktionen auftreten, die unter „Programmierstil“ (siehe unten) näher beschrieben sind.

**Variablenname erwartet(1):** In einer FOR-NEXT-Schleife oder an einer anderen Stelle, an der eine Variable erwartet wird, steht kein zulässiger Variablenname. Beispielsweise ist es nicht erlaubt, eine Zuweisung an ST (Statusvariable) oder QA (interne Variable des Runtime-Moduls) zu machen.

**Falsche Dimensionierung(1):** Dimensionierung einer Variable, die kein Array ist.

Diese dreiteilige Serie soll eine genaue Übersicht über die Bedienung des Speedcompilers und seine verschiedenen Eigenschaften geben. Nebenbei ist einiges zu erfahren über den Aufbau eines BASIC-Compilers, vernünftige BASIC-Programmierung und C64-Know-how. Im einzelnen: der erste Teil dreht sich um die Ursachen der Compiler-Fehlermeldungen; im zweiten Teil gibt es Tips zur optimalen Compiler-Nutzung; Spezialitäten wie die Einbindung von Assemblerprogrammen in Kompilate behandelt Teil 3. Der „Speedcompiler“ selbst wurde in Ausgabe 10/87 veröffentlicht.

**Keine String-Verknüpfung(1):** Es wurde versucht, eine arithmetische Operation mit Strings als Operanden auszuführen, etwa, zwei Strings voneinander abzuziehen.

**Zu viele Variablen(1):** Das Programm enthält zu viele Variablen (Integer-Arrays, Strings, String-Arrays, Gleitkomma-Variablen oder Gleitkomma-Arrays). Es dürfen maximal 128 verschiedene Variablen deklariert werden. String-Arrays zählen dabei als zwei Variablen. Diese Zahl hat sich in der Praxis als ausreichend erwiesen, sogar ein so umfangreiches Programm wie der Speedcompiler selbst leidet nicht unter dieser Einschränkung. Entsteht wirklich ein Bedarf nach mehr als 128 Variablen im Programm, kann man zunächst überall dort, wo dies möglich ist, statt Gleitkomma-Variablen Integer-Variablen benutzen. Als zweiter Ausweg bietet sich an, selten benutzte Variablen zu einem Array zusammenzufassen. Diese Variablen können dann über die Indizierung angesprochen werden und benötigen nur einen Platz in der internen Variablenliste.

## Grenzwerte gezogen

**Numerischer Ausdruck erwartet(1):** Statt eines numerischen Ausdrückes wurde ein String-Ausdruck angegeben. So ist etwa „POKE A\$+B\$,.0“ unzulässig.

**Zu viele numerische Variablen(1):** Es werden im Programm zu viele Integer-Variablen (keine Arrays!) benutzt. Wie bei der Fehler-

meldung „Zu viele Variablen“ beschrieben, besteht die Möglichkeit, Variablen zusammenzufassen. Maximal sind 120 Integer-Variablen erlaubt, Schleifen-Variablen benötigen zwei Plätze.

**Zeilenende erwartet(1):** Bei manchen Befehlen (zum Beispiel DIM, ON GOTO, DEF) darf die Zeile nach dem Befehl nicht fortgesetzt werden, das heißt, es darf kein Befehl mehr folgen. Diese Ausnahmen sind in der Anleitung zum Compiler in Ausgabe 10/87 aufgeführt.

**Formel zu komplex(1):** Der Compiler oder das Runtime-Modul können einen numerischen Ausdruck nicht auflösen, da er zu komplex ist. Kommt nur sehr selten vor, Abhilfe: aufteilen.

**Verschiedene Typen(1):** Dieser Befehl hat starke Parallelen zum „TYPE MISMATCH ERROR“. Es wurde versucht, einen String mit einem numerischen Ausdruck zu verknüpfen, zum Beispiel: A\$+B.

**Unbekannter Befehl(1):** Der gefundene Befehl ist nicht im Speedcompiler implementiert. Das gilt insbesondere für einige spezielle INPUT-BASIC-Befehle. Die Liste der nicht implementierten Befehle befindet sich in der Anleitung.

## Passendes erwartet

**String-Ausdruck erwartet(1):** Anstelle eines String-Ausdrucks wurde ein numerischer Ausdruck angegeben.

**'£' erwartet(1):** Vor Compiler-Optionen muß '£' stehen. Wird statt dessen zum Beispiel 10 DIM A\$(9): REMInhalt angegeben, so vermutet der Compiler nach 'REM', daß die Länge des dimensionierten Strings folgt, es fehlt aber das '£'. Im übrigen sind REMs nach Dimensionierungen ohnehin verboten, wenn sie nicht der String-Anpassung dienen.

**Fehlerhafte String-Länge(1):** Als String-Länge wurde entweder Null oder eine Zahl größer 254 angegeben. Normale Strings dürfen bis 255 Zeichen haben, String-Arrays aus programmtechnischen Gründen aber nur 254.

**Hex-Zahl erwartet(1):** In einem Ausdruck wurde mit einem '\$' angezeigt, daß eine hexadezimale Zahl folgen soll. Diese Zahl muß

aus mindestens einer und maximal vier hexadezimalen Ziffern ('0' bis '9', 'A' bis 'F') bestehen.

**'=' erwartet(1):** Am Anfang einer Zeile oder nach einem Doppelpunkt steht eine Variable, der kein '=' für die erforderliche Zuweisung folgt.

**Befehlsende erwartet(1):** Der Befehl wurde syntaxwidrig fortgesetzt.

**'STEP' erwartet(1):** Der Ausdruck nach 'FOR' einer FOR-NEXT-Schleife wurde syntaxwidrig fortgesetzt, also mit mit einem anderen Befehl als 'STEP'.

**'-' oder '1' erwartet(1):** FOR-NEXT-Schleifen mit Integer-Schleifen-Variablen dürfen nicht mit einem anderen STEP-Wert als '1' (keine STEP-Angabe) oder '-1' (Angabe: STEP-1) benutzt werden. Wird bei einer derartigen Schleife STEP angegeben, so führt jede andere Zahl als '-1' dahinter zu diesem Fehler. Andere STEP-Werte lassen sich mit einer Gleitkomma-Schleifen-Variablen realisieren.

**'NEXT' ohne 'FOR' gefunden(1):** Der Compiler konnte dem 'NEXT' in dieser Zeile kein passendes 'FOR' zuordnen. Zu jeder FOR-Schleife gehört ein und nur ein 'NEXT'. Siehe auch unten unter „Programmierstil“!

**Fehler in Def(1):** Ein Syntax-Fehler in der DEF-Funktion liegt vor.

**Zeilennummer nicht gefunden(2):** Beim Auflösen einer Sprungadresse wurde die Zeilennummer nicht im Programmtext gefunden. Entspricht dem 'UNDEF'D STATEMENT' des C64-BASIC-Interpreters. Manche nicht aufgelöste Referenzen erkennt die veröffentlichte Compiler-Version nicht, siehe dazu „Fehlerfeuer“.

**Unerwartetes Zeilenende(1):** In einem Befehl wurde die Zeile beendet, ohne den Befehl abzuschließen.

**Sekundäradresse muß 1 sein in LOAD(1):** Als Sekundäradresse bei 'LOAD' ist aus programmtechnischen Gründen nur '1' oder '2' (als Konstante) zulässig. Dabei bedeutet '2', daß „das nachgeladene Programm nach dem Laden gestartet wird (kompiliertes BASIC ohne Runtime-Modul).“

**Unzulässige Compileroption(1):** Diese Option ist entweder nicht implementiert oder in dieser Form nicht zulässig.

## Schutz vor Überlastung

**Zu viele Zeilen(1):** Der Compiler kann bis 3000 Programmzeilen verarbeiten. Diese Anzahl kann theoretisch eigentlich nur durch die exzessive Verwendung sehr kurzer Zeilen überschritten werden.

**Floppy-Fehler(1),(2):** Bei der Arbeit mit dem Diskettenlaufwerk ist ein Fehler aufgetreten. Es kann sich auch um einen Übertragungsfehler zwischen C64 und Floppy han-

deln. In diesem Fall gibt die angezeigte Fehlermeldung der Floppy keinen Aufschluß über die Ursache des Fehlers. Sind zwei Laufwerke angemeldet, so erfolgt zunächst die Fehlermeldung vom Quelllaufwerk, dann die vom Ziellaufwerk. Nach 'Floppy-Fehler' wird die Arbeit des Compilers abgebrochen.

### 'Compiler überlastet'

oder

**'Überlauf im Compiler'(1),(2):** Ein interner Puffer des Compilers ist übergelaufen.

## Für RS-232- und DFÜ-Fans

Durch die Geschwindigkeit und die besiegte Garbage Collection ist der Speed-compiler prädestiniert, Programme für DFÜ-Anwendungen wie zum Beispiel Terminalprogramme zu kompilieren. Leider hat sich ein Fehler in die Open-Routine für die RS-232-Schnittstelle eingeschlichen, der zu spät entdeckt wurde. Aber zum Glück ist es kein dramatischer Fehler, der leicht umgangen werden kann.

Beim Öffnen eines RS-232-Kanals bricht das Programm ohne Fehlermeldung ab. Dies liegt an der speziellen internen Struktur der RS-232-Routinen. Das läßt sich leicht umgehen, indem man vor dem OPEN-Befehl die Option 'I04' gibt. Nach dem OPEN-Befehl kann man diese Option mit 'I00' wieder aufheben, falls sie nicht mehr gebraucht wird. Beim CLOSE-Befehl ist genauso vorzugehen. Das einzige Störende nach diesem OPEN oder CLOSE ist die Tatsache, daß die Status-Variable ST größer als 61440 wird; bis zum nächsten Befehl, der den Status ändert.

Grund ist die, gelinde ausgedrückt, etwas eigenartige Programmierung des RS-232-OPEN-Befehls im C64. Für Maschinen-sprache-Programmierer sei dies kurz erklärt: Diese Routine kommt mit gesetztem Carry-Flag und der Fehlernummer '240' im Akku zum Aufrufer zurück. Der BASIC-Interpreter nimmt dann diese Fehlermeldung zum Anlaß, die Ein- und Ausgabepuffer einzurichten.

Um durch solche Werte nicht bei der Abfrage von ST gestört zu werden, ist es immer ratsam, 'ST AND 255' einzusetzen, wo man ST benötigt. Hat das hö-

herwerte Byte von ST (ermittelbar durch INT(ST/256)) einen anderen Wert als 240, so ist ein echter Fehler beim Öffnen oder Schließen des Kanals aufgetreten.

Der RS-232-Kanal benötigt bekanntlich zwei Puffer im BASIC-Speicher. Diese liegen bei normaler RAM-Konfiguration bei \$9E00 (40448 dezimal) und \$9F00 (40704 dezimal). Dies beschränkt den verfügbaren Platz für kompilierte BASIC-Programme einschließlich der Variablen auf den Bereich unter \$9E00. Da dies der Compiler nicht prüft, muß der Programmierer selbst anhand der Compiler-Ausgabe sicherstellen, daß das „Ende der Variablen“ den Wert '40448' nicht überschreitet.

Um auch den Speicherplatz unter dem BASIC-ROM und ab \$C000 (49152 dezimal) nutzen zu können, kann ein Trick helfen, die RS-232-Puffer zu verlegen. Gibt man vor dem OPEN-Befehl folgende zwei POKEs ein, so sind die Puffer ab \$CE00 (52736 dezimal) und \$CF00 (52992 dezimal).

POKE 643,0  
POKE 644,208

Dadurch werden die Zeiger auf das Ende des für BASIC verfügbaren RAMs auf die Adresse 53248 verlegt, nach Abzug der 512 Bytes für die beiden Puffer können die Variablen dann den Adressbereich bis 52735 belegen. Vor Programmende ist es ratsam, mittels 'POKE 644,160' den Normalzustand wiederherzustellen.

Meist bringt die Aufteilung des Befehls, bei dessen Kompilation der Fehler auftaucht, Abhilfe. Dieser Fehler kommt äußerst selten vor und führt zum sofortigen Abbruch.

**Befehl nicht implementiert(1):** Wie "Unbekannter Befehl".

**RESTORE auf DATA-Ende(2):** Im Gegensatz zum BASIC-Interpreter erlaubt der Speed-compiler ein gezieltes 'RESTORE zeilenummer'. Diese Zeilennummer darf nicht hinter der letzten DATA-Zeile stehen, sonst kommt es zu dieser Fehlermeldung.

Alle weiteren, hier nicht noch einmal aufgeführten Meldungen sind entweder selbsterklärend oder mit diesen Erläuterungen meines Erachtens hinreichend erhellert.

Erwähnt werden muß noch, daß ein Fehler einen oder mehrere Folgefehler nach sich ziehen kann, auch wenn zu diesen Fehlermeldungen im einzelnen kein Anlaß besteht. Gehäuft kommt es zu dieser Erscheinung, wenn der erste Fehler direkt vor dem Zeilenende auftaucht. Diese Folgefehler können natürlich ignoriert werden. Eine weitere Quelle von Folgefehlern sind DIM- und FOR-Befehle. Tritt in diesen Befehlen nämlich ein Fehler auf, kann der Compiler in der Regel mit allen weiteren Ausdrücken, die sich auf diese Befehle beziehen (NEXT und Variablen-Benutzung), nichts mehr anfangen.

## Gelegentlich unwillig: das DOS

Wenn beim Kompilieren ein völlig unverständlicher Syntax-Fehler auftaucht und man beim Durchsehen des Quelltextes feststellen muß, daß ab der monierten Stelle das Programm zerstört ist, kommt häufig das DOS als Übeltäter in Frage. Die Floppy hat dann beim Kompilieren aus einem mir unbekannten Grund (DOS-Bug?) das erzeugte Programm (Objektcode) über den Quelltext geschrieben. Aus diesem Grund entstanden auch die in der letzten Ausgabe aufgestellten „Richtlinien zur Kompilation“, da beim Einhalten dieser Richtlinien solche oder ähnliche Fehler relativ wenig Schaden anrichten können und auch selten auftauchen.

## Programmierstil gefragt

Das BASIC des C64 ist bekanntlich hochgradig tolerant gegenüber Strukturfehlern

## Fehlerfeuer

Anscheinend waren die Programme, mit denen der Speedcompiler getestet wurde, nicht fehlerfrei genug. Folgendes ist nämlich vor der Veröffentlichung niemandem aufgefallen: gelegentlich wird ein GOTO oder GOSUB zu einer Zeilennummer, die nicht existiert, nicht wie erforderlich mit einer Fehlermeldung quittiert. Statt dessen kommt es dann zu einem Laufzeitfehler. Folgende fünf POKEs beheben das Problem bei der vorliegenden Version:

```
POKE 15186,0  
POKE 15187,0  
POKE 15188,0  
POKE 15189,44  
POKE 15190,1  
POKE 8661,50
```

Also den Compiler laden, diese POKEs eingeben und den Compiler wieder abspeichern (unter anderen Namen oder vorher alten Compiler „scratches“). Der sechste POKE-Befehl berichtigt dabei nur die Versionsnummer, die bei Benutzung eines Druckers ausgegeben wird; diese ist nämlich nicht auf dem neuen Stand.

Ein freundlicher Akt gegenüber dem Compiler ist es außerdem, nach einem NEXT-Befehl die Schleifen-Variable anzugeben. Das erleichtert dem Compiler die eventuell notwendige Fehlersuche ganz ungemein.

Das Ausspringen oder Einspringen (vom Hauptprogramm aus) in den Schleifenrumpf (alles das, was zwischen 'FOR' und 'NEXT' steht) mit 'GOTO' ist natürlich auch verboten. Die einzige Ausnahme bildet ein Aussprung aus einem Unterprogramm mit Berichtigung des System-Stacks mittels der Compiler-Option 'EB'.

## Von Beginn an festgelegt

Ein weiteres Beispiel für nicht erlaubte Konstruktionen ist das Dimensionieren von Arrays innerhalb von Unterprogrammen. Da der normale BASIC-Interpreter beim Durcharbeiten des Programms dem Programmverlauf folgt, der Compiler aber den Programmtext von vorn nach hinten durchliest, müssen die Dimensionierungen immer am Programmanfang stehen. Denn der Compiler legt die Adressen der einzelnen Variablen schon während der Kompilation fest, unter anderem um Zeit während der Laufzeit zu sparen. Wenn die Dimensionierung eines Arrays aber erst im Unterprogramm erfolgt, nachdem im laufenden Programmtext dieses Array möglicherweise schon angesprochen wurde, kann dies nicht gutgehen.

Aus dem gleichen Grund ist auch der „Trick“ mit dem Löschen der Variablen nicht erlaubt. Manche Programme legen nämlich einmal Variablen an, löschen sie dann später mit 'CLR' und legen einfach einen neuen Satz, anders gearterter Variablen an. Das ist natürlich für einen Compiler, der versucht, so viele Festlegungen wie möglich schon bei der Kompilation zu treffen und nicht erst während der Laufzeit (was Laufgeschwindigkeit kostet!), nicht durchschaubar. Darum sind solche Tricks verboten. Sollte es zu Schwierigkeiten mit dem Speicherplatz kommen, muß man die Datenstruktur so anpassen, daß sie den Erfordernissen ohne solche Tricks entspricht.

Ein weiterer Grund für Fehler, die obendrein nicht oder nur selten quittiert werden und darum schwer zu lokalisieren sind, ist der falsche Gebrauch von Compiler-Optionen.

Die meisten Compiler-Optionen sollen für das ganze Programm gelten. Darum müs-

sen sie auch ganz am Anfang, vor jedem anderen Befehl, stehen. Wird das nicht berücksichtigt, kann es zu un durchschau baren und schwerwiegenden Fehlern kommen.

Zum Beispiel:

```
10 a=5  
20 rem £V0  
30 print a
```

Dieses kleine Programm druckt kompiliert nicht etwa wie erwartet die Zahl '5' aus, sondern unverständlichweise '0'. Aber schauen wir uns doch mal an, was in Zeile 20 steht. '£V0 schaltet in den Integer-Modus. Das heißt, jede Variable ohne Zusatz wird als Integer-Variable behandelt. Der Knackpunkt aber ist, daß diese Option erst nach ihrem Erscheinen im Programm wirksam wird, nach Zeile 20 also. So wird 'a' in der Zeile 10 als Gleitkomma-Variable angenommen, während 'a' in Zeile 30 eine Integer-Variable ist. Man könnte also statt dessen schreiben:

```
10 a&=5  
30 print a%
```

Hier wird wohl jeder einsehen, daß es sich um zwei verschiedene Variablen handelt, die verschiedene Werte haben.

Dieses kleine Beispiel zeigt die Gefährlichkeit mancher Konstruktionen und die Wichtigkeit einer gewissen Struktur. Das gilt übrigens nicht nur für den Speed-Compiler.

## Optionen an der Kette

Mehrere Compiler-Optionen können in einer Zeile aneinandergehängt werden, allerdings nicht völlig beliebig.

Richtig ist:  
10 rem £io4: £v0

nicht zulässig dagegen:  
10 rem £io4: rem £v0

Nach dem Doppelpunkt muß außer dem Leerzeichen gleich das nächste Pfundzeichen kommen, kein weiteres 'REM':

Im nächsten Teil dieser Reihe geht es um optimales Programmieren mit dem Compiler. Im dritten Teil wird es dann um die Nutzung von Assemblerprogrammen gemeinsam mit kompilierten Programmen gehen.  
Jürgen Lindemeyer/JS



# Drei Riesen in zehn Wochen

## INPUT 64 sucht das Abenteuer

Mit etwas Glück bringt Ihnen der Einstieg dreitausend Mark. Das ist nämlich der Preis, den wir für das beste Abenteuerspiel zahlen, das uns bis zum 15. Januar nächsten Jahres erreicht. Zusätzlich zu unserem 'normalen' 3000-DM-Wettbewerb eine weitere Chance, uns diesen Betrag aus der Tasche zu ziehen.

### Riesenpreis

Noch 10 Wochen haben Sie Zeit, um sich eine spannende Geschichte auszudenken und daraus eine harte Nuß für Adventure-Freaks zu machen. Das Thema ist dabei völlig freigestellt. Nur kriegerische Auseinandersetzungen und Spiele nach dem Motto „Leichen pflastern seinen Weg“ haben von vornherein keine Chance auf den Wettbewerbssieg.

Wie es sich für gute Adventure-Spiele gehört, sollte Ihr Beitrag mit einer ansprechenden Grafik ausgestattet sein. Für Animationen, die womöglich noch mit „astreinen Sounds“ untermauert sind, vergibt die Jury natürlich Extrapunkte.

**Seit Erscheinen der letzten Ausgabe sitzen sie in ihren Hinterzimmern, Dachstuben und umgebauten Garagen an ihren Rechnern. Sie denken nur noch an eins: die drei großen braunen Scheine. Noch ist es auch für Sie nicht zu spät. Zum Einstiegen, zum Mitmachen, zum Mitgewinnen. Beim INPUT 64-Adventure-Wettbewerb.**

Großen Wert legen wir auf intelligente Reaktionen auf (Fehl-)Eingaben. Nichts ist nervtötender, als bei jedem zweiten Befehl mit „Das geht jetzt nicht.“ abgespeist zu werden. Je mehr Wörter Ihr Spiel versteht und je differenzierter es auf eingegebene Sätze antwortet, desto höher wird es in der Wertung liegen. Daß ein Programm, das auf Fehleingaben mit einem blauen Bildschirm und 'READY' reagiert oder dessen Bildschirm sich durch unkontrollierte Cursor-Bewegungen zerstören läßt, nicht in die engere Wahl kommt, braucht wohl nicht extra erwähnt zu werden.

### Riesenspaß

Abgesehen von der technischen Qualität Ihrer Einsendung ist natürlich vor allem der Spielspaß entscheidend für die Bewertung. Dieses Kriterium ist zugegebenermaßen subjektiv, wird aber doch von einigen meßbaren Faktoren beeinflußt. So freut sich der eingefleischte Adventure-Knacker, wenn er den Spielstand vor einer riskanten Aktion abspeichern kann. Großer Beliebtheit erfreuen sich auch Abenteuer, die man mit mehreren Spielern im Team oder im Wettlauf gegeneinander lösen muß.

Adventure-Spiele gewinnen an Spannung, wenn einige Aufgaben in einer vorgegebenen Zeit oder mit einer bestimmten Höchstanzahl von Eingaben gelöst werden müssen. Auch Einlagen, die ein geschicktes Hantieren mit dem Joystick erfordern, sind immer willkommen. Dagegen wird aus Abenteuerlust schnell Spielfrust, wenn der Lösungsweg zu stark eingehalten werden muß und jedes Abweichen auf dem berühmt-berüchtigten RIP-Bild endet.

### Kleine Bitte

Zum Schluß einige technische Hinweise: Senden Sie uns Ihren Beitrag bitte auf Datenträger ein – zum Abtippen von Listings haben wir keine Lust, und einen Drucker haben wir selbst. BASIC-Programme sind bitte weder kompiliert noch kompaktiert. Wollen Sie eine BASIC-Erweiterung verwenden, dann halten Sie bitte vorher mit uns Rücksprache. Bei Assemblerprogrammen gehört der kommentierte Quellcode dazu.

Die besten Einsendungen dieses Wettbewerbs sollen natürlich in INPUT 64 veröffentlicht werden und auch innerhalb des Magazins laufen. Dazu sind einige Einschränkungen in der Speicherplatzbelegung und im Umgang mit Betriebssystem-Vektoren erforderlich. Bei reinen BASIC-Programmen gibt es in der Regel keine Probleme, aber wenn Sie Ihren Beitrag in Maschinensprache schreiben wollen, sollten Sie vorsorglich unsere Autorenhinweise anfordern. Unsere Adresse finden Sie im Impressum. An diese schicken Sie bitte auch – mit dem Stichwort 'Adventure-Wettbewerb' versehen – Ihre fertigen Programme.

HS



# Gebt Bugs keine Chance!

## Fehlerfreie BASIC-Programme mit Syntax-Check

Das Programm Syntax-Check von Andreas Thiel überprüft BASIC-Programme hauptsächlich auf unkorrekten Sprachgebrauch, da Computer gerade in dieser Frage überaus penibel sind. PRINDT SIN(2,5) versteht doch jeder, nur der Rechner stellt sich absolut dumm! Eigentlich ist das ja noch kein Problem. Früher oder später präsentiert der Rechner ja seine Verständigungsschwierigkeiten – meistens allerdings später. Alle Eingaben sind bereits getägigt und die gewünschten Berechnungen durchgeführt; bei einer überaus selten benutzten Programmroutine passiert es dann: SYNTAX ERROR oder ähnliches ist da zu lesen – von einem Ergebnis keine Spur. Bei umfangreichen Grafikberechnungen kann es schon mal ein paar Stunden dauern, bis es soweit ist.

### Auf der sicheren Seite

Mit der INPUT-üblichen Steuersequenz CTRL-S wird das Programm aus dem Ma-

Jeder, der schon mal ein Programm geschrieben hat, kennt das: die notwendige Fehlerbeseitigungszeit ist oftmals um ein Vielfaches größer als die eigentliche Programmierzeit. Stupide Fleißarbeiten sollte man aber getrost dem Rechenknecht Computer überlassen; Syntax-Check hilft beim Debuggen von BASIC-Programmen und spart Zeit, die für kreative Betätigungen zur Verfügung steht. Damit kein unerkannter SYNTAX ERROR mehr das Programmieren vergällt!

gazin auf eine eigene Diskette gespeichert. Für Neulinge, Vergeßliche und seltene Gäste gibt es auf Seite 2 ausführliche Bedienungshinweise. Vor dem Speichern sollte man das Programm an den eigenen Druck-

ker anpassen! Mit der Auswahl ANPASUNG gelangt man ins Installationsmenü. Die Geräte- und Sekundäradresse lässt sich meist aus dem Kopf einstellen. Für die weiteren Einstellungen empfiehlt sich jedoch ein Blick ins Druckerhandbuch. Eine von der 'normalen' Schriftart verschiedene Einstellung wird gesucht, das kann Breit-, Kursiv- oder Fettschrift sein. Sollte der Drucker die Betriebsart 'automatisches Unterstreichen' kennen, ist das die 'optimale' Einstellung. Mit dieser zweiten Schriftart werden die gefundenen Fehler bei Ausgabe auf den Drucker markiert. Tragen Sie jetzt die vom Drucker gewünschte Steuersequenz für 'Markierung einschalten' an der entsprechenden Stelle ein. Direkt dahinter wird die 'Anzahl' der zu sendenden Bytes vermerkt. Sinngemäß ist das gleiche natürlich auch für 'Markierung ausschalten' nötig. Gehört der Drucker in die Familie der zu Commodore MPS 801 Kompatiblen, ist keine Anpassung erforderlich, da die benötig-

ten Werte der Voreinstellung entsprechen. Ist keine Hervorhebung der Fehler gewünscht (moderne Pädagogen werden diese Möglichkeit bevorzugen) oder sträubt sich der Drucker, wird einfach bei 'Anzahl' der Wert Null eingesetzt (an beiden Stellen!).

Bereich	Routinen
\$a000 - \$b5ff 40960 - 46591	SYNTAX-Prüfung
\$c000 - \$ceff 49152 - 52991	Steuer- u. Menüteil

#### Der BASIC-Speicher bleibt frei

Geladen und gestartet wird wie bei einem BASIC-Programm. Das RUN verschiebt zwei Maschinenprogrammteile in Speicherbereiche außerhalb des BASIC-Bereichs. Ist diese Prozedur erfolgreich verlaufen, erscheint auf dem Bildschirm eine Meldung mit einem Kurzinfo zum weiteren Vorgehen. Zuerst wird das zu untersuchende BASIC-Programm in den Rechner geladen und anschließend der Syntax-Check mit SYS 49152 aktiviert.

#### Von hinten aufgezäumt

Betrachten wir die einzelnen Möglichkeiten von hinten. Da wäre als erstes 'Programmende': hierzu bleibt wenig zu sagen. Das Programm wird verlassen, und man landet im Direktmodus, das vorhandene Programm bleibt zur weiteren Bearbeitung erhalten. Für eine erneute Überprüfung steht der Syntax-Check mit SYS 49152 weiterhin zur Verfügung.

Da dem Commodore-BASIC viele Befehle für Grafik und Sound fehlen, strotzen etliche BASIC-Programme nur so von PEEKs und POKEs. Wie leicht vertippt man sich bei den fünfstelligen Zahlen, und fortan wird nach Herzenslust ins eigene oder ein anderes Programm gepokt! Selbstzerstörerischen Programmtdendenzen stellt man am besten mit einer POKE-Bereichsüberprüfung nach. Im Untermenü von 'POKEBE-

REICH EINSTELLEN' sind die entsprechenden Optionen zu finden. Man kann für die Eingabe zwischen dem Dezimal- und dem Hexadezimalsystem wählen. Beim Eingeben einer Grenze erscheint der Cursor im Rahmen für Dezimaleingabe. Ist die Eingabe in Hex gewünscht, geht es mit ↓ (CRSR-Down) ins Hex-Feld.

Schwarz auf weiß erhält man die Fehler auf den Drucker gelistet, wählt man diesen im zweiten Punkt aus. Dabei wird die Fehlerbeschreibung und die entsprechend Programmzeile gedruckt. Hat man die Druckeranpassung gewissenhaft durchgeführt, wird die Fehlerstelle hervorgehoben. Alle Ausgaben gehen bis zum Ende des Checkdurchlaufs auf den Drucker. Anhand des Ausdrucks kann man dann in alle Ruhe die Fehler beseitigen.

Die eigentliche Arbeit nimmt das Programm auf, wählt man 'CHECKDURCHLAUF STARTEN'. Freude kommt auf, erhält man nach kurzem die Meldung: FEHLER: 0 und WARNSUNG: 0. Der nächste Tastendruck befördert einen zurück ins Hauptmenü. Selten passiert dieses beim ersten Versuch. Ein Fehler nach dem anderen wird da ausgegeben. Doch holla, was ist das? Differenzierte Fehlermeldungen, die deutlich sagen, was

wo genau falsch ist. Anders als bei der Druckerausgabe wird das Programm im bildschirmorientierten Modus nicht in einem Rutsch von Anfang bis Ende bearbeitet. Bei jedem Fehler stoppt das Programm und zeigt die BASIC-Zeile und die entsprechende Fehlermeldung am Monitor. Ein Druck auf W setzt die Prüfung fort, H bricht die laufende Prüfung ab und kehrt zurück zur nächsthöheren Ebene. E verläßt den Syntax-Check, und man befindet sich im BASIC-Editor am Anfang der betroffenen Zeile zur Beseitigung des gefundenen Fehlers. Danach wird der Cursor einfach auf die Zeile mit dem SYS 49152 gestellt, ein Druck auf RETURN startet den Syntax-Check aufs neue. Sind alle Fehler beseitigt, vergessen Sie nicht, die neue Version Ihres Programms zu sichern!

#### Nur verwarnt

Das Programm unterscheidet zwischen Fehlern und Warnungen. Warnungen werden beim Ansprechen des Poke-Bereichs gemeldet oder beim Fehlen des zweiten Anführungszeichen bei Strings. Dieses stellt zwar keinen Fehler dar, doch meckert der eine oder andere BASIC-Compiler. Bei Programmen die man später kompliert möch-

#### Überflüssigen Ärger vermeiden

Zu unangenehmen Überraschungen kann es kommen, startet man ein BASIC-Programm, bevor der Syntax-Check aufgerufen wird. Möglicherweise pokt das Programm in den Checker, und der läuft dann nicht mehr oder zumindest anders, als er soll. In solch einem Fall gab es einmal eine zweiteilige Fehlerliste bei einem fehlerfreien Programm. Zudem traten die Fehler nur in Zeilen auf, die im Programm gar nicht vorhanden waren. Manchmal scheint es allerdings unvermeidlich, mit RUN zu starten: Befinden sich vor dem Programm Maschinenroutinen, zum Beispiel PRINT AT oder INKEY, stellt sich die Frage: Wie soll's gehen? Der Syntax-Check untersucht immer die BASIC-Zeilen, die man auch mit LIST sehen kann. Im genannten Fall also eine Zeile mit SYS-Aufruf. Damit die noch verborgenen

Zeilen untersucht werden können, wird der BASIC-Anfang von Hand hochgesetzt. Die Speicherstellen 43/44 (\$2b/\$2c) müssen dafür die notwendigen Werte erhalten. Standardmäßig enthalten sie die Werte 1/8, der BASIC-Beginn liegt damit bei Adresse 2049 (\$0801). Die neuen Werte erhält man, indem das Programm nach dem Starten mit RUN/STOP abbricht. Die Adressen 43/44 kann man mit PEEK auslesen und notiert. Jetzt den Syntax-Check laden und initialisiert – vorher den BASIC-Anfang wieder auf \$0801 setzen! Nach dem Laden des Programms pokt man die genannten Speicherstellen mit den notierten Werten. Überzeugen Sie sich mit LIST, daß alles geklappt hat. Jetzt geht's weiter wie gehabt.

te, läßt sich durch Syntax-Check eine Menge Zeit sparen, da ein Compiler-Lauf im Regelfall recht lange dauert. "Gänsebeine" gibt's für die BASIC-Beschleuniger also besser paarig. Dafür mögen diese allerdings gar kein Spaghetti! Mehrere NEXT für ein FOR sind absolut unverdaulich, stellt es außerdem auch nicht gerade ein Ausbund an Struktur dar. Das BASIC V2 verführt zwar zu solchen 'Kavaliersdelikten', doch: „Wehret den Anfängen!“ Strukturierte Programme sind noch nach Wochen der Nichtbeschäftigung mit ihnen durchschaubar und auch für andere verständlich und wartbar. Hörte ich da gerade ein: „Moralapostel!“? Eigentlich nicht, doch beim Thema 'Struktur statt Spaghetti' – JA! Zu schwerwiegenden Programmabstürzen kommt es auch immer wieder durch Ansprünge von Zeilennummern, die nicht vorhanden sind. Bei allen Sprüngen wird deshalb auf Vorhandensein des Ziel's geprüft.

Nie wieder Fehler in Programmen? Nein, das kann Syntax-Check nicht leisten, denn die untersuchten Programme werden nicht

simuliert. Fehler, die erst zur Laufzeit auftreten, bleiben unerkannt. Den besten Schutz dagegen bietet ein durchdachtes Programmkonzept – sinnvollerweise ist das eine Arbeit, die vorm Eintippen der ersten Programmzeile liegt. Das Vorgehen ist etwa dem des Schreibens eines Aufsatzes vergleichbar. Schulaufsätze (erst schreiben, dann die Gliederung fertigstellen) sind ausdrücklich nicht gemeint! Auch ist es gar

## Ab und an eine Maus melken

nicht so einfach, ein fertiges Programm auszutesten. Ohne dieses hier weiter vertiefen zu wollen, sollte man versuchen, das Programm unter den sogenannten Extrembedingungen zu testen. Ein durchaus ernstgemeinter Vorschlag ist der sogenannte Monkey-Test: alle zehn Finger leicht krümmen und mehrfach die Hände auf die Tastatur fallen lassen. Aber mit Gefühl bitte! Versuchen sollte man auch die Kombination **Ctrl** und **SHIFT**. Manchmal stellen sich interessante, aber unerwünschte Effekte ein. Nach dieser eher mechanischen Pro-

grammprüfung sollte es überlegt weitergehen. Man sollte ruhig mal versucht zu drücken und dabei so tun, als wäre das notwendige Druckerkabel gerade verliehen. Der Versuch Dateien auf eine volle Diskette abzuspeichern verläuft manchmal anders als erwartet. Bei Eingaben ruhig mal einem Buchstaben versuchen, auch wenn eine Ziffer gewünscht wird. Der Programmentwickler kennt die Feinheiten, bei denen sein Programm streikt. Doch woher soll der zukünftige User wissen, daß keine zu großen oder zu kleinen Zahlen erlaubt sind? Das wird alles dem Bereich Plausibilitätskontrolle zugerechnet. Und dieser liegt wiederum im Verantwortungsbereich des Programmierers!

Und was tun, wenn alles nicht klappt, der Fehler nicht zu finden ist, obwohl er sich immer wieder störend bemerkbar macht? Ab und an eine Maus melken, das beruhigt – und nach einer Pause mit Abstand und System(!) weitersuchen. Der Einsatz von Programmierhilfen wie SYNTAX-CHECK, FIND, TRACE oder ähnlichen ist ausdrücklich empfohlen!

pan

## Assembler-Know-how für alle!

Ab sofort direkt beim Verlag erhältlich: Ein Leckerbissen für jeden Assembler-Programmierer und alle, die es werden wollen.

Eine Diskette mit dem Macro-Assembler INPUT-ASS aus INPUT 64, Ausgabe 6/86, und dazu

- der komplette Source-Code dieses Assemblers
- der Source-Code des Maschinensprache-Monitors MLM64plus aus INPUT 64, Ausgabe 11/87
- Library-Module: I/O-Routinen, Hex/ASCII/Dezimal-Wandlung, Multiplikation, Division
- Konvertierungsprogramme zur Format-Wandlung von PROFI-ASS- und MAE-Texten in das Source-Code-Format des INPUT-ASS

**Preis: 49,— zuzüglich 3,— DM für Porto und Verpackung (nur gegen V-Scheck)**

**Bestelladresse:** Verlag Heinz Heise GmbH & Co KG  
Postfach 61 04 07 · 3000 Hannover 61

## INPUT 64 BASIC-Erweiterung

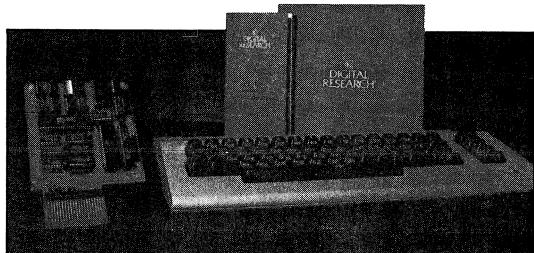
Die BASIC-Erweiterung aus INPUT 64 (Ausgabe 1/86), gebrannt auf zwei 2764er EPROMs für die C-64-EPROM-Bank.

Keine Ladezeit mehr – über 40 neue Befehle und Super-Tape integriert.

**Preis: 49,— DM  
zuzüglich 3,— DM für Porto und Verpackung (V-Scheck)**

**Bestelladresse:**  
Verlag Heinz Heise  
GmbH & Co KG  
Postfach 61 04 07  
3000 Hannover 61

# Back to the roots



## Das neue c't-Projekt: CP/M-Karte für C64

Die Karte kann mit einem seriellen Schnittstellen-Baustein bestückt werden (6551), der eine V.24 mit fünfzehn Baudraten von 50 bis 19 200 Baud zur Verfügung stellt. Außerdem ist eine parallele IEEE-488-Schnittstelle vorgesehen. Der Anschluß von Einzel- und Doppellaufwerken der CBM-Serie, die mit Kapazitäten von bis zu 1 MByte unter CP/M besonders interessant sind, ist daher kein Problem. Es besteht auch Kompatibilität zu anderen CP/M-Karten.

Das CP/M-Projekt eröffnet den Zugang zum leistungsfähigen Z80-Prozessor, der mit seinen 16-Bit-Registern und den über 500 Prozessor-Befehlen gegenüber der 6510-CPU den Maschinen-Programmierern ganz neue Möglichkeiten bietet. Zahlreiche unter CP/M laufende Werkzeuge erleichtern die Entwicklung von Z80-Assemblerprogrammen. Mit der c't-CP/M-Karte läßt sich auf dem C64 ein 52-KByte-System betreiben, damit stehen für Anwendungen 45 KByte Programmspeicher zur Verfügung.

CP/M selbst gibt dem Programmierer von Anwender-Programmen über 40 Systemfunktionen in die Hand, über die auf die Schnittstellen und auf die Diskette zugegriffen werden kann. Systemnahe Befehle zum Lesen und Schreiben einzelner Disketten-Sektoren sind ebenso selbstverständlich wie der sequentielle oder Random-Zugriff auf Dateien. Die Anzahl der gleichzeitig geöffneten Dateien ist dabei unbeschränkt.

Entscheidend für die Leistungsfähigkeit eines CP/M-Systems ist das BIOS, also der

CP/M ist als Urvater aller Mikrocomputer-Betriebssysteme zwar nicht mehr der Jüngste, gehört aber noch lange nicht zum alten Eisen. Schließlich existiert eine Fülle bewährter Anwender-Software, man denke nur an WordStar oder an Turbo-Pascal. Das neue c't-Projekt ist folglich eine leistungsfähige CP/M-Karte, deren mit 8 MHz betriebene Z80-CPU alle bisher dagewesenen Lösungen leicht abhängt.

Teil, der die Verbindung zur Hardware herstellt. Bei der Entwicklung des BIOS für die c't-CP/M-Karte wurde besondere Sorgfalt auf einen schnellen Diskettenzugriff gelegt. Dank eines speziellen Sector-Read-Ahead-Mechanismus ist die 1541 ein ausreichend schneller Massenspeicher für zügiges Arbeiten, wobei die Kompatibilität zum C64-CP/M-Normalformat gewährleistet ist. Der Kaltstart des Systems, also das Booten des kompletten Betriebssystems, benötigt 9 Sekunden, ein Warmstart dauert 4 Sekunden, und WordStar ist in 10 Sekunden geladen. Insgesamt unterstützt das BIOS zwei 1541-Drives (Laufwerk A: und B:) mit einer Kapazität von je 136 KByte, sowie als Laufwerke C: und D: zwei Drives mit je 856 KByte Kapazität (8050, 8250 oder SFD 1001).

Professionelles Arbeiten setzt eine 80-Zeilchen-Darstellung voraus. Aus diesem

Grund ist das BIOS kompatibel zu einer hochwertigen 80-Zeilchen-Karte, die sich durch störungsfreies Scrollen auszeichnet. Ihr Bildschirmspeicher liegt im Adreßraum des Z80-Prozessors. Dadurch kann beispielsweise WordStar direkt auf den 80-Zeilchen-Bildschirm zugreifen und läuft dann in puncto Geschwindigkeit zu wahrer Hochform auf.

Das Projekt startet in c't 12/87 (Erscheinungsdatum 16. November) mit der ausführlichen Beschreibung der Hardware. Der in c't 1/88 folgende Software-Teil stellt das BIOS vor und zeigt unter anderem, wie man mit anderen CP/M-Rechnern, etwa dem Apple II, Daten austauscht.

## Training total

### Sammeldiskette: Englische GRAMmatik

Ab sofort beim Verlag erhältlich: Eine Zusammenfassung aller bislang erschienenen Folgen des interaktiven Lernprogramms „Englische GRAMmatik“.

Eine Diskette mit 800 Übungssätzen in 10 abgeschlossenen Einheiten zu den wichtigen Problemen der englischen Grammatik.

Erweitert um flexible Druck-Optionen und die Möglichkeit, eigene Übungen zu erstellen.

Aus dem Inhalt: „If-clauses“, „Reported Speech“, „Irregular plural forms“, „Personal and reflexive pronouns“, „Will-future and going-to-future“, „Prepositions of place, movement and time“, „Question tags and short answers“ u.v.a.m.

**Preis: 19,80 DM inklusive Porto und Verpackung (nur gegen Verrechnungsscheck)**

**Bestelladresse:**  
Verlag Heinz Heise GmbH & Co KG  
Postfach 61 04 07  
3000 Hannover 61

# Unterbrechen Sie mal!

## Interrupts — Prinzip und Programmierung

„Lassen Sie sich doch mal unterbrechen!“ Unter diesem Motto steht die jetzt beginnende Serie. Sie befaßt sich mit einfachen Techniken der Interrupt-Programmierung. Was ein Interrupt überhaupt ist? Ein Interrupt ist nichts weiter als eine Unterbrechung. Zu wenig, meinen Sie? Nun gut, ich werde etwas weiter ausholen.

In den Anfangszeiten der Computertechnik reichte es aus, wenn der Mikroprozessor immer eine Aufgabe zur Zeit bewältigen konnte. Im Prinzip ist das auch heute noch so. Doch im Gegensatz zu damals, als Computer noch nicht über Terminals verfügten, müssen die heutigen Mikroprozessoren eher auf Einflüsse von außen reagieren können, wie zum Beispiel zur Annahme von aktuellen Tastatureingaben. Dies kann ohne großen Aufwand mit Hilfe der Interrupt-Programmiertechnik gelöst werden, in der Regel in Maschinensprache. Da es aber nun nicht jedem unserer Leser liegt, solche Sachen in Assembler zu programmieren, stelle ich Ihnen auch Alternativen vor, mit denen die Interrupt-Routinen aus BASIC herausbeeinflußt werden können.

### Abfrage ohne Aufsehen

Wie reagiert nun der Mikroprozessor auf besagte Unterbrechungen? Betrachten wir ein Beispiel aus dem täglichen Programmieralltag. Sie sitzen vor Ihrem C64 und geben ganz entspannt Zeile für Zeile ein BASIC-Programm ein. Während Sie so fleißig auf die Tasten hauen, muß die CPU ja Ihre Eingaben an den entsprechenden Empfänger weiterleiten. Und genau hier fängt die Unterbrechung Ihrer Arbeit an. „Wieso Unterbrechung“ werden Sie sich vielleicht fragen, „ich kann doch laufend eingeben, und das Zeichen der gedrückten Taste er-

Daß mit dem C64 ohne die periodischen Unterbrechungen namens „Interrupts“ wenig los wäre, hat sich wahrscheinlich mittlerweile herumgesprochen. Warum das so ist, was eigentlich ein Interrupt bedeutet und wie man sogar aus BASIC heraus diese Interrupts nutzen kann, verrät die erste Folge unserer Interrupt-Tips.

scheint sofort auf dem Bildschirm?“ Richtig! Doch zwischen dem Tastendruck und dem Anzeigen der gedrückten Taste vergeht doch eine winzige Zeitspanne. Und in diesen Bruchteilen von Sekunden werden die Interrupt-Routinen abgearbeitet.

Das BASIC-Programm in der Demonstration soll das nochmals modellhaft verdeutlichen; es findet ein ständiger Wechsel zwischen Hauptprogramm und Interrupt-Programm statt. Letzteres ist für all die Aufgaben zuständig, von denen der Benutzer normalerweise nichts bemerkte: Tastatur abfragen, Zeichen auf dem Bildschirm „echoen“ und so fort.

Wie das im Prinzip läuft? Die CPU (Central Processing Unit = Zentraleinheit, im C64 der 6510) reagiert auf Unterbrechungsanforderungen folgendermaßen:

- Unterbrechen des laufenden Programmes
- Abarbeiten des Interrupt-Programmes (IRQ)
- Fortsetzen des unterbrochenen Programmes an der Stelle, wo es verlassen wurde.

Wie das im einzelnen vor sich geht, können Sie dem Kommentar zu dem abgedrucktem Listing entnehmen.

### Eingriffe ins Innere

Auch ohne Maschinensprache-Kenntnisse kann man einige Experimente mit den Interrupt-Routinen anstellen. Wie erwähnt, muß der C64 viele Aufgaben gleichzeitig erledigen: Bearbeiten des Hauptprogramms, Erledigen des System-Interrupts und Senden des Video-Signals an Monitor oder Fernseher. Wir können durch einige Eingriffe jeweils einige Teile dieser Arbeit beschleunigen.

Als erstes soll probehalber der System-Interrupt-Routine das Abfragen der STOP-Taste und das Stellen der TI-Uhr abgewöhnt werden, und zwar durch den Befehl

POKE 788,52.

Hierbei ist zu beachten, daß die interne Uhr dann nicht korrekt läuft, da sie ja nicht mehr „gestellt“ wird. Der Ursprungszustand kann durch

POKE 788,49

wiederhergestellt werden. Der Sinn dieser POKEs besteht in einer Beschleunigung des abgearbeiteten BASIC-Programms und dem Ausschalten der STOP-Taste.

### Schnell, . . .

Auch die Anzahl der Unterbrechungsaufrufe kann man verändern. Allerdings sollte man sich die Konsequenzen vor Augen führen: bei zu wenigen Aufrufen hinken Uhr, Cursor und Tastaturabfrage nach; bei zu vielen werden sie zu schnell. Mit

POKE 56325,0

erreicht man extrem viele, mit  
POKE 56325,255

extrem wenig Interrupt-Aufrufe. Logische Folge: je weniger häufig die Interrupt-Routine aufgerufen wird, desto schneller wird die Arbeitsgeschwindigkeit des BASIC-Programms. (Die Adresse 56325 ist ein Zähler in der CIA 1, dem „Complex Interface Adapter“. Dieser Baustein hat unter anderen einen eingebauten Timer, der Interrupts auslösen kann.) Eine mögliche Anwendung dieser POKEs: Arbeitsbeschleunigung bei Programmteilen ohne Eingaben.

### ... aber dunkel

Der wohl bekannteste Eingriff zur Arbeitsbeschleunigung ist die Unterbrechung der Video-Signalausgabe. Der Nachteil ist sozusagen augenfällig: der aktuelle Bildschirminhalt ist nicht zu sehen, geht aber auch nicht verloren. Ausgeschaltet wird der Bildschirm mit

POKE 53265,PEEK(53265) AND 239 und wieder eingeschaltet mit  
POKE 53265,PEEK(53265) OR 16.

Die Adresse 53265 ist das Register 17 im VIC (Video Interface Controller) und für die Steuerung der Bildschirmausgabe verantwortlich. Der Vorteil dieser Möglichkeit der Arbeitsbeschleunigung in Programmen besteht im der weiterhin korrekt laufenden Uhr und einer funktionierenden STOP-Taste. In unserer Demonstration können Sie die einzelnen Zeitverschiebungen gut erkennen und für eigene Anwendungen auswerten und nutzen.

### Quellenstudium

Aufmerksamen Lesern wird schon beim Durchdenken der Beispiele aufgefallen sein, daß anscheinend mehrere Verursacher einer Interrupt-Anforderung möglich sind. Als Interrupt-Quelle kommen eine Reihe von Geräten und Bausteinen in Betracht. Genannt seien zunächst die sogenannten „sekundären Quellen“, wie Diskettenstation, Datasette, Drucker, Modem, Schaltelemente et cetera. Sekundär deswegen, weil sie nicht direkt mit der CPU in Verbindung stehen.

Die Diskettenstation beispielsweise kann eine Unterbrechungsanforderung über den seriellen Port des Computers an die CIAs geben. Diese leiten das „Ansinnen“ des Peripherie-Gerätes an die CPU weiter, und erst dort wird diese Unterbrechungsanforderung bearbeitet.

Soviel zum Thema „sekundäre“ Interrupt-Quellen, befassen wir uns nunmehr mit den primären Unterbrechungsquellen. „Primär“, weil sie im direkten Kontakt zur CPU stehen. Das sind:

- der VIC-II-CHIP (MOS 6566/6567 Video Interface Controller)
- die beiden CIAs (MOS 6526 Complex Interface Adapter)
- die RESTORE-Taste
- der Expansions-Port
- als Sonderfall: die RESET-Leitung, die wir aber bei unseren Betrachtungen weglassen. Schließlich geht es um Unterbrechungen, nicht um einen Abbruch mit anschließendem Kaltstart.

### Wer darf?

Doch schön der Reihe nach. Wodurch werden bei welchem Baustein die Unterbrechungen herbeigeführt?

1. Der **VIC-II-CHIP** als Unterbrechungsquelle. Es sind vier Ereignisse vorgesehen, die zu einer Unterbrechungsaufforderung führen.

- Die Rasterzeilen-Unterbrechung
- Die Kollision eines Sprites mit einem Hintergrund
- Die Kollision von Sprites untereinander
- Die Lichtgriffelanwendung (deren Verwendung ich aber zurückstelle)

Es sei kurz erwähnt, daß in dem VIC-Register 26 (Adresse 53274) festgelegt wird, welche der obengenannten Unterbrechungsmöglichkeiten nun auch wirklich eine Unterbrechung hervorrufen darf. In diesem „Interrupt-Enable-Register“ (Unterbrechungs-Zulassungs-Register) bestimmen dies die einzelnen Bits: Die Bitposition 7 bis 4 sind unbenutzt und haben daher immer den Wert Eins. Bit 3 entspricht der Lichtgriffelunterbrechung, Bit 2 der Kollision von zwei Sprites, Bit 1 der Kollision eines Sprites mit dem Hintergrund, Bit 0 der Rasterzeilen-Unterbrechung. Ein gesetztes Bit be-

### Maske aufgesetzt

Wenn Sie gerade mit irgendeiner Arbeit beschäftigt sind und das Telefon klingelt, unterbrechen Sie Ihre augenblickliche Tätigkeit, gehen zum Telefon, heben den Hörer ab, führen das Gespräch und fahren anschließend mit der unterbrochenen Tätigkeit fort. Dieser Ablauf entspricht ziemlich genau den Vorgängen beim Auftreten eines Interrupts: Programm unterbrechen, Interrupt-SERVICE-Routine abarbeiten, unterbrochenes Programm fortsetzen.

Nun kann es vorkommen, daß Sie bei der Arbeit nicht gestört werden wollen. Wie Sie das im einzelnen anstellen, sei jetzt dahingestellt – ob Sie das Telefon leise stellen, den Hörer von der Gabel nehmen oder wie auch immer. Der Prozessor hat für solche Gelegenheiten einen bestimmten Befehl, nämlich SEI. SEI steht für Set Interrupt Disable und setzt ein Flag im Status-Register. Ist dieses Flag gesetzt, wird auf die Unterbrechungsanforderung nicht reagiert. Aufgehoben wird dieser Zustand durch den Befehl CLI – Clear Interrupt Disable. Im Fachjargon nennt man diesen Vorgang „maskieren“ oder „ausmaskieren“ des Interrupt-Flags, der „gewöhnliche Interrupt“ ist „maskierbar“.

Allerdings gibt es noch eine zweite Sorte Interrupts, die NMIs. NMI ist die Abkürzung für Non Maskable Interrupts und meint, daß diese Unterbrechungsanforderung nicht maskierbar ist, der Zustand des Interrupt-Flags im Statusregister bleibt unberücksichtigt. Diese Art Interrupt kommt immer durch, Beispiel: RESTORE-Taste.

deutet, daß die Unterbrechung zugelassen ist.

Zur Erinnerung ein Hinweis zu allgemeinen „Logeleien“ mit Bits und Bytes in einer angenommenen Adresse „AD“:

Setzen eines Bits durch  
POKE AD,PEEK(AD) OR 2<sup>BITPOSITION</sup>  
Löschen eines Bits durch  
POKE AD,PEEK(AD) AND 255- 2<sup>BITPOS</sup>

## Hintergrundarbeit

Anhand des disassemblierten ROM-Listings lässt sich der Weg, den das Unterbrechungsprogramm im Commodore 64 durchläuft, genau verfolgen.

Der Prozessor bekommt eine Interrupt-Anforderung (woher und von wem, klären wir später) und holt sich den Anfang des Programms, das daraufhin abgearbeitet werden soll, aus den Adressen \$FFFE/\$FFFF (dezimal 65534/65535; alle folgenden Angaben erfolgen nur noch in hexadezimaler Form und können mit dem in dieser Ausgabe veröffentlichten Monitor am Rechner verfolgt werden), den beiden höchsten Speicheradressen im C64. Dort ist ein Verweis auf die Adresse \$FF48 eingetragen. Nach dem Retten der Prozessor-Register wird getestet, ob der Prozessor wegen eines Interrupts an diesem Programmzählerstand ankam oder weil er auf einen BREAK-Befehl gelau- ften war. Die Behandlung von BREAK

und Interrupt ist nämlich bis hierher gleich. War es wirklich ein Interrupt, geht es wieder indirekt weiter, mit einem Sprung zum Inhalt des „Vektors“ \$0314/\$0315, zur Routine ab \$EA31.

Diese Routine wird normalerweise 60 mal pro Sekunde verarbeitet. Die erste aufgerufenen Subroutine überprüft, ob die STOP-Taste gedrückt ist, und stellt die interne Uhr TI weiter. Die folgenden Befehle erledigen das Cursor-Blinken (was, wie man sieht, eine nicht ganz anspruchslose Aufgabe ist), anschließen werden die Recorder-Tasten abgefragt. Zum Schluß geht es in die Tastaturabfrage, das IRQ-Flag in der CIA wird gelöscht und nach dem Zurücksetzen der Register mit RTI ins unterbrochene Programm zurückgekehrt.

Eingriffsmöglichkeiten existieren durch „Verbiegen“ des Vektors \$0314/\$0315 und des – im Listing nicht mehr dargestellten – Vektors in der Tastaturauswertung (\$028F/\$0290).

Das Gegenstück zu diesem „Maskenregister“ ist das Interrupt-Latch-Register (53273) des VIC-II-CHIP („Unterbrechungs-Einrast-Register“). In ihm wird das Eintreten eines der oben genannten Ereignisse durch das Setzen des entsprechenden Bits angezeigt. Wurde diese Unterbrechung in dem „Maskenregister“ (VIC-Reg.26) zugelassen, so führt das zu einer Unterbrechung des laufenden Programmes. Testen kann man das durch einfaches Lesen von Bit 7 im VIC-Register 25; dieses zeigt an, ob überhaupt ein VIC-Interrupt eingetreten ist.

Anwendungsmöglichkeiten der VIC-Interrupts sind:

- Mischen von hochauflösender Grafik mit Textteilen
- Verwendung zweier Zeichensätze auf einem Bildschirm (zum Beispiel Groß-/Kleinschrift und Großschrift/Blockgrafik)
- Einbringen von Laufschriften in normalen Bildschirmtext

```
; interrupt-service-routine
; im commodore 64

;einstiegs-behandlung sowohl fuer
;BRK-befehl als auch fuer interrupt
ff48 48 :pha ;alle
ff49 8a :txa ;register
ff4a 48 :pha ;des
ff4b 98 :tya ;unterbrochenen
ff4c 48 :pha ;progs retten
ff4d ba :tsx ;status vom
ff4e bd 04 01 :lda 0104,x ;stack holen
ff51 29 10 :and #10 ;war's BRK?
ff53 f0 03 :beq ff58 ;z-flag=nein
ff55 6c 16 03 :jmp (0316) ;zum BRK-prg.
ff58 6c 14 03 :jmp (0314) ;zum IRQ-prg.

;irq-routine im engeren sinne
ea31 20 ea ff :jsr ffea ;stop-taste
;abfragen und
;TI stellen

;cursor-behandlung
ea34 a5 cc :lda cc ;wenn der
ea36 d0 29 :bne ea61 ;cursor blinkt
ea38 c6 cd :dec cd ;blinkzaehler
ea3a d0 25 :bne ea61 ;erniedrigen
ea3c a9 14 :lda #14 ;bzw. neu setzen
ea3e 85 cd :sta cd
ea40 a4 d3 :ldy d3 ;nun
ea42 46 cf :lsl cf ;das blinken
ea44 ae 87 02 :ldx 0287 ;selbst
ea47 b1 d1 :lda (d1).y ;erledigen
ea49 b0 11 :bcs ea5c ;im video-ram
ea4b e6 cf :inc cf ;und
```

```
:sta ce ;im
ea4f 20 24 ea :jsr ea24 ;color-ram
ea52 b1 f3 :lda (f3).y ;nach der
ea54 8d 87 02 :sta 0287 ;berechnung
ea57 ae 86 02 :idx 0286 ;werte
ea5a a5 ce :lda ce ;jeweils
ea5c 49 80 :eor #80 ;einsetzen
ea5e 20 1c ea :jsr ea1c ;farbe auch

;recorder-tasten auswerten
ea61 a5 01 :lda 01 ;port laden
ea63 29 10 :and #10 ;auf taste
ea65 f0 0a :beq ea71 ;pruefen
ea67 a0 00 :ldy #00 ;und
ea69 84 c0 :sty c0 ;je nach
ea6b a5 01 :lda 01 ;ergebnis
ea6d 09 20 :ora #20 ;strom fuer
ea6f d0 08 :bne ea79 ;bandmotor
ea71 a5 c0 :lda c0 ;ausschalten
ea73 d0 06 :bne ea7b ;oder
ea75 a5 01 :lda 01 ;den strom
ea77 29 1f :and #1f ;einschalten
ea79 85 01 :sta 01 ;ende der recoder-behandlung

ea7b 20 87 ea :jsr ea87 ;tastatur-abfrage
ea7e ad 0d dc :lda dc0d ;irq-flag loeschen

;irq-ende-behandlung
ea81 68 :pla ;alle
ea82 a8 :tay ;register des
ea83 68 :pla ;unterbrochenen
ea84 aa :tax ;programms
ea85 68 :pla ;wieder einsetzen
ea86 40 :rti ;und zurueck
```

## Uhren, Flaggen und Alarme

2. Die beiden CIA-Bausteine können durch mehrere Ereignisse zur Unterbrechungsquelle führen.

- Unterlauf der internen Uhr A
- Unterlauf der internen Uhr B
- Erreichen einer eingestellten Alarmzeit
- Am seriellen Port wurde eine Unterbrechungsanforderung abgestellt.

Am Eingang namens FLAG wurde ein bestimmter Zustand erreicht. Auch hier gibt es Unterbrechungs-Kontroll-Register für jede der beiden CIAs. Gemeint ist das Register 13 der beiden CIAs (56333-CIA 1 / 56589 CIA 2). Von hier wird zum Beispiel das Timing des Systeminterrupts gesteuert. Über die genaue Belegung und Handhabung dieser Register werden wir noch ausführlich berichten. Diesmal sei nur erwähnt, daß CIA 1 maskierbare Interrupts, CIA 2 nicht maskierbare Interrupts (NMIs) erzeugt. Siehe dazu auch den Kasten „Maske aufgesetzt“. Mögliche Anwendungen der CIAs als eigene Interrupt-Quellen sind:

- Nutzen der internen Uhren (besonders die der CIA 2) für eigene Timer-Aufgaben
- Nutzung der Echtzeituhren.

## Einer kommt immer durch

3. Die RESTORE-Taste ist eine „nicht maskierbare“ Unterbrechungsquelle. Hier müssen wir außerdem das erste Mal von einer „willkürlichen“ Unterbrechung sprechen. Die anderen Unterbrechungsarten waren meist periodischer Art (System-Interrupt und so weiter). „Willkürlich“ deswegen, weil die Unterbrechung erst auf Grund unseres Tastendruckes stattfindet. Da die RESTORE-Taste direkt mit dem nicht maskierbaren Interrupt-Eingang der CPU verbunden ist, können wir hier durch einfachen Druck auf die RESTORE-Taste jederzeit ins Geschehen eingreifen. Unsere Demonstration wird es Ihnen nochmals verdeutlichen. Wie auch bei den CIAs werden wir uns später noch ausführlich mit der Nutzung der RESTORE-Taste befassen. Nutzbar ist diese Möglichkeit zur Unterbrechung jeder Aktivität mit anschließender Ausführung eines eigenen Unterbrechungsprogrammes, wie etwa Ausgeben des Disketten-Inhaltsverzeichnis oder Wechsel der Bildschirmrahmenfarbe oder . . .

Zum Schluß dieser Folge möchte ich Ihnen noch Hinweise zum Gebrauch der abspeicherbaren Demonstration geben. Es handelt sich bei dem ersten Teil des BASIC-Programmes um eine Initialisierungs-Routine der internen Uhr der CIA 1. In ihr wird die aktuelle Uhrzeit im BCD-Format („binary coded decimals“) abgelegt und anschließend der NMI-Vektor (Sprungzielzeiger) um einige Bytes versetzt, so daß bei einem Druck auf die RESTORE-Taste das Initiali-

sierungs-Programm übersprungen wird, das die System-Vektoren „normalisiert“. Im zweiten Teil des BASIC-Programmes steht das Maschinenprogramm, das nach seiner Initialisierung in den System-Interrupt eingebunden ist. Es liest auf Tastendruck die Uhrenregister der CIA 1 aus und POKEt sie dann auf den Bildschirm. Sie haben somit nach dem Start des Uhrenprogrammes jederzeit „auf Knopfdruck“ die aktuelle Uhrzeit vor Augen.

Th. Stamer

Verlag  
Heinz HEISE GmbH  
Helstorfer Straße 7  
3000 Hannover 61

# Schlagkräftig!

## software



Der Schachtrainer  
auf Diskette:

Eine Vielzahl von Trainingsmöglichkeiten für alle begeisterten Schachspieler und solche, die es werden wollen. Der Anfänger entwickelt sich schnell zum starken Gegner; dem Fortgeschrittenen gelingt es, mit immer wieder neuen und intelligenten Spielvarianten zu überraschen.

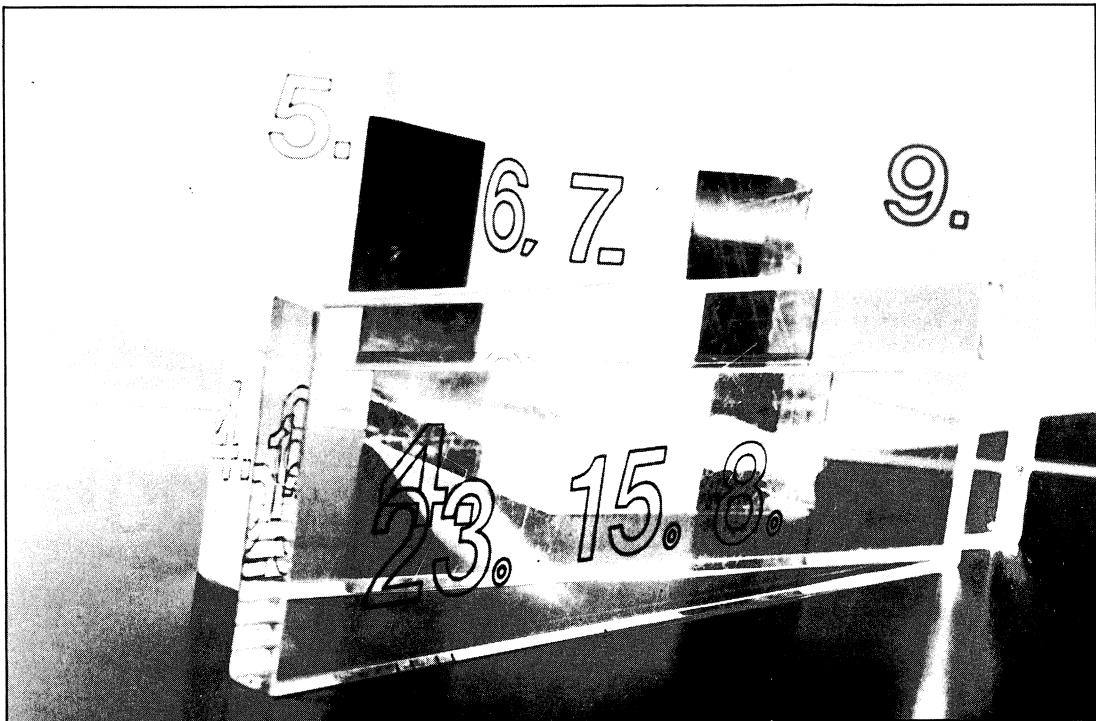
Best.-Nr. 13138 für C64

Best.-Nr. 51138 für IBM

DM 68,- unverbindliche Preisempfehlung

HEISE-Bücher und Software erhalten Sie bei Ihrem Computer-, Elektronik- oder Buchhändler.

138/1.4



# Gregor XIII.

Ein Kalender-Programm für viele Jahre

Sie haben sicherlich – je nach Einsatzbereich – unterschiedliche Vorstellungen von dem äußerem Erscheinungsbild eines Kalenders. So kann es sinnvoll sein, alle Tage eines Jahres auf einen Blick zu sehen, langfristig bekannte Termine gleich mit ausdrucken zu lassen oder aber auch Platz für nachträgliche (aktuelle) Eintragungen zu haben.

Diesen unterschiedlichen Anforderungen wird unser Kalender-Programm gerecht, indem es drei unabhängige Druckbilder generiert. Sie haben die Wahl zwischen einem 'Wandkalender', einem 'Terminkalender' und einem 'Merkkalender'. Das Programm ist

**Rechtzeitig, bevor das jährliche Kalenderkaufen akut wird, stellen wir Ihnen ein flexibles Kalender-Druck-Programm vor. Was Sie brauchen, um die Geldausgaben dieses Jahr zu sparen, ist neben diesem Programm nur noch ein beliebiger Drucker.**

durchgehend menüorientiert und von daher einfach zu bedienen.

## Der Überblick über die Zeit, und . . .

Vom Hauptmenü aus erreichen Sie mit den Cursor-Tasten und einer Bestätigung mit RETURN die Option "Ausgabe eines Wandkalenders". Für das eingestellte Jahr wird ein übersichtlicher Jahreskalender auf eine Seite gedruckt (siehe Abbildung). Da hier kein Platz mehr für Einträge vorhanden ist, haben Sie auch keine weiteren Bedienun-

gen zu beachten. Mit 'f2' gelangen Sie wieder in das Hauptmenü.

Wenn Sie die "Auszgabe eines Merkkalenders" angewählt haben, können Sie vor dem Ausdruck für jeden Tag eine kurze Bemerkung (11 Zeichen) eintragen. Bei den Tagen, wo schon „amtliche“ Vermerke stehen, können Sie diese selbstverständlich auch überschreiben.

Sie sehen einen Ausschnitt von 10 Tagen auf dem Bildschirm. Mit den Cursor-Tasten beziehungsweise mit 'f5' können Sie diesen Ausschnitt wie ein Fenster über das ganze

Jahr schieben. Haben Sie die Stelle für Ihren Eintrag erreicht, gelangen Sie mit 'f3' zu der eigentlichen Eingabe. Innerhalb des Fensters können Sie jetzt den Cursor frei bewegen und über die Tastatur Ihre Einträge vornehmen. Beachten Sie bitte, daß jede Zeile mit RETURN abgeschlossen werden muß. Sie erkennen eine abgeschlossene Eingabe an der reversen Darstellung. Den Eingabe-Modus können Sie nur mit 'f2' wieder verlassen.

Wenn Sie auf diese Weise alle Eintragungen vorgenommen haben, können Sie diese Kalenderform mit 'f7' ausdrucken. Der Jah-

reskalender benötigt jetzt zwei Druckseiten und ist in dieser kompakten Form immer noch sehr übersichtlich.

Die dritte und letzte Form der Darstellung ist monatsweise orientiert. Sie erreichen diese Möglichkeit durch Anwahl des Menüpunktes "Auszgabe eines Terminkalenders". Sie können für jeden Tag einen kurzen Text eingeben (30 Zeichen), wobei die Programmbedienung identisch mit der vorher beschriebenen Option ist.

Der Ausdruck (wieder mit 'f7' erreichbar) benötigt jetzt für jeden Monat eine Druckseite. Für jeden Tag des eingestellten Monats ist ein kleines Kästchen vorgesehen, wobei nur die ersten sieben Zeichen Ihrer Tageseingabe direkt im Kasten ausgedruckt wird. So bleibt auch für nachträgliche handschriftliche Einträge genug Platz. In der Aufstellung unterhalb der Tageskästchen werden Ihre Einträge selbstverständlich in voller Länge ausgegeben.

## ... der Blick hinter die Kulissen

Das Programm geht unmittelbar nach dem Start von der Jahreszahl 1988 aus. Es werden die Wochentage, die festen und die beweglichen Feiertage berechnet. Nach rund 30 Sekunden meldet sich ein blinkender Cursor, und es wird Ihre Eingabe erwartet.

Sollten Sie das voreingestellte Jahr nicht wünschen, können Sie auch ein anderes Jahr eingeben, das mit dem gregorianischen Kalender berechnet werden kann. Diese kalendarischen Berechnungen wurden von Papst Gregor XIII. im 16. Jahrhundert eingeführt und basieren auf dem davor üblichen Julianischen Kalender.

Außerhalb von INPUT 64 können Sie das Programm über den dafür vorgesehenen Menüpunkt verlassen und danach auflisten. Da das Programm gut dokumentiert und recht gut modularisiert ist, werden Sie die einzelnen Routinen schnell erkennen und gegebenenfalls spezielle (Drucker-)Anpassungen vornehmen können.

Sollten Sie das Programm modifiziert haben, vergessen Sie bitte vor dem Abspeichern nicht die obligatorischen Pokes (44,8 und 43,1; Sie wissen schon ...).

T. Stahmer/wm

2083						
<b>JANUAR</b>				<b>FEBRUAR</b>		
MONTAG 6 13 20 27				3 10 17 24		
DIENSTAG 7 14 21 28				4 11 18 25		
MITTWOCH 1 8 15 22 29				5 12 19 26		
DONNERSTAG 2 9 16 23 30				6 13 20 27		
FREITAG 3 10 17 24 31				7 14 21 28		
SAMSTAG >> 4 11 18 25				8 15 22 29		
SONNTAG >> 5 12 19 26				1 8 16 23		
				2 9 16 23 30		
<b>APRIL</b>						
MONTAG 7 14 21 28				5 12 19 26		
DIENSTAG 1 8 15 22 29				6 13 20 27		
MITTWOCH 2 9 16 23 30				7 14 21 28		
DONNERSTAG 3 10 17 24				1 8 15 22 29		
FREITAG 4 11 18 25				2 9 16 23 30		
SAMSTAG >> 5 12 19 26				3 10 17 24 31		
SONNTAG >> 6 13 20 27				4 11 18 25		
<b>JULI</b>						
MONTAG 7 14 21 28				4 11 18 25		
DIENSTAG 1 8 15 22 29				5 12 19 26		
MITTWOCH 2 9 16 23 30				6 13 20 27		
DONNERSTAG 3 10 17 24 31				7 14 21 28		
FREITAG 4 11 18 25				1 8 15 22 29		
SAMSTAG >> 5 12 19 26				2 9 16 23 30		
SONNTAG >> 6 13 20 27				3 10 17 24 31		
<b>OKTOBER</b>						
MONTAG 6 13 20 27				3 10 17 24		
DIENSTAG 7 14 21 28				4 11 18 25		
MITTWOCH 1 8 15 22 29				5 12 19 26		
DONNERSTAG 2 9 16 23 30				6 13 20 27		
FREITAG 3 10 17 24 31				7 14 21 28		
SAMSTAG >> 4 11 18 25				1 8 15 22 29		
SONNTAG >> 5 12 19 26				2 9 16 23 30		
<b>NOVEMBER</b>						
MONTAG 6 13 20 27				3 10 17 24		
DIENSTAG 7 14 21 28				4 11 18 25		
MITTWOCH 1 8 15 22 29				5 12 19 26		
DONNERSTAG 2 9 16 23 30				6 13 20 27		
FREITAG 3 10 17 24 31				7 14 21 28		
SAMSTAG >> 4 11 18 25				1 8 15 22 29		
SONNTAG >> 5 12 19 26				2 9 16 23 30		
<b>DEZEMBER</b>						
MONTAG 6 13 20 27				3 10 17 24		
DIENSTAG 7 14 21 28				4 11 18 25		
MITTWOCH 1 8 15 22 29				5 12 19 26		
DONNERSTAG 2 9 16 23 30				6 13 20 27		
FREITAG 3 10 17 24 31				7 14 21 28		
SAMSTAG >> 4 11 18 25				1 8 15 22 29		
SONNTAG >> 5 12 19 26				2 9 16 23 30		

**Aktuell:** Wert für einen funktionstüchtigen Computer Commodore C64 (Baujahr 87)  
37 500 DM. Bezahlte wurde dieser Preis auf einer Antiquitäten-Auktion in Hamburg.

# Gelbe Post – bitte kommen!

**Electrobrief erledigt Disketten-Mail**

Zeitgeistberührte wissen es ja eh schon, Briefe auf Papier sind out! Mailing heißt das neue Zauberwort, trotz seines jugendlichen Alters gibt es heute schon unzählige verschiedene Formen. Der Vielfalt ist eines gemein: der waldfreundliche Verzicht auf Bütten oder ähnlichem. Eine verbreitete Version ist die per Akustikkoppler (oder illegalem Modem) und Telefon in eine sogenannte Mailbox. Sollten Sie das jetzt mit einer Blackbox verwechseln, liegen Sie gar nicht so falsch. Bei diesen 'Dingern' handelt es sich um einen Zwitter zwischen Briefkasten und Postfach. Der Absender hinterlegt seine Message (veraltet auch: Mitteilung) in einer Box, der Adressat holt sich selbige dort ab.

Eine andere Möglichkeit ist das Verschicken von Disketten. In der einfachsten Form benutzt man unzählige BASIC-Zeilens im PRINT-Text-Stil aneinander. Achtet man dabei auf zuviel Qualität, ist die Message in unserer schnellebigen Zeit mit Chance schon wieder veraltet. Damit Ihnen solches nicht widerfahrt und die Mode nicht zur Qual wird, entwickelte der Autor Gerrit Knoef den Electrobrief. Speichern Sie diesen mit CTRL-S auf eine eigene Diskette ab, so haben Sie ihn jederzeit fix zur Hand. Noch ein Tip für die ganz Eiligen: durch einen Compaktor<sup>1</sup> gejagt lassen sich rund 50% Ladezeit sparen!

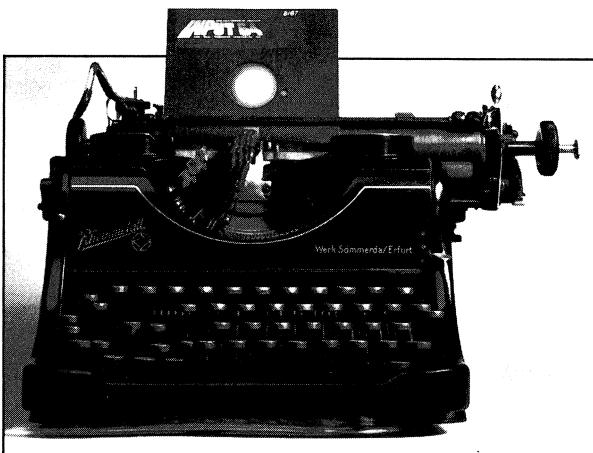
Ohne Zeitverlust kann es nach dem Programmstart mit dem Schreiben losgehen. Die Funktion 'Text aufnehmen' mit der '1' anwählen und einfach losschreiben. Die Bedienung gleicht der des BASIC-Editors. Schreiben Sie erst mal den Text in einem

**Immer häufiger gehen Disketten via Postsack auf die Reise. Ein kurzes Anschreiben soll dabei sein – das Label scheidet aus: einfach zu klein! Will man die Spieleerung des Briefkastens noch erreichen, dauer's mit der Textverarbeitung einfach zu lange. Mit Electrobrief zaubert man in Null Komma nichts einen hübschen Brief direkt auf die Scheibe.**

Rutsch runter; solange die Zeilen noch auf dem Bildschirm sichtbar sind, lassen sich Fehler wieder ausbügeln. Die Schreibmarke läßt sich mit den Cursor-Tasten an jede beliebige Stelle bewegen, mit Delete oder durch Überschreiben vertuscht man fast jedes Malheur. Haben Sie viel Zeit oder gehören Sie zu den Leuten, die viel mitzuteilen haben (mehr als eine Bildschirmseite), schreiben Sie nur! Zeilen, die oben raus'scrollen', sind nicht verloren, der Empfänger bekommt die ganze Sendung. Farbenpracht erreicht man mit der CTRL- oder ⌘-Taste in Verbindung mit den Zahlen. Ist

Taste	Bedeutung
£	ß
@	ü / Û
: / [	ö / Ö
; / ]	ä / Ä
<	:
>	:

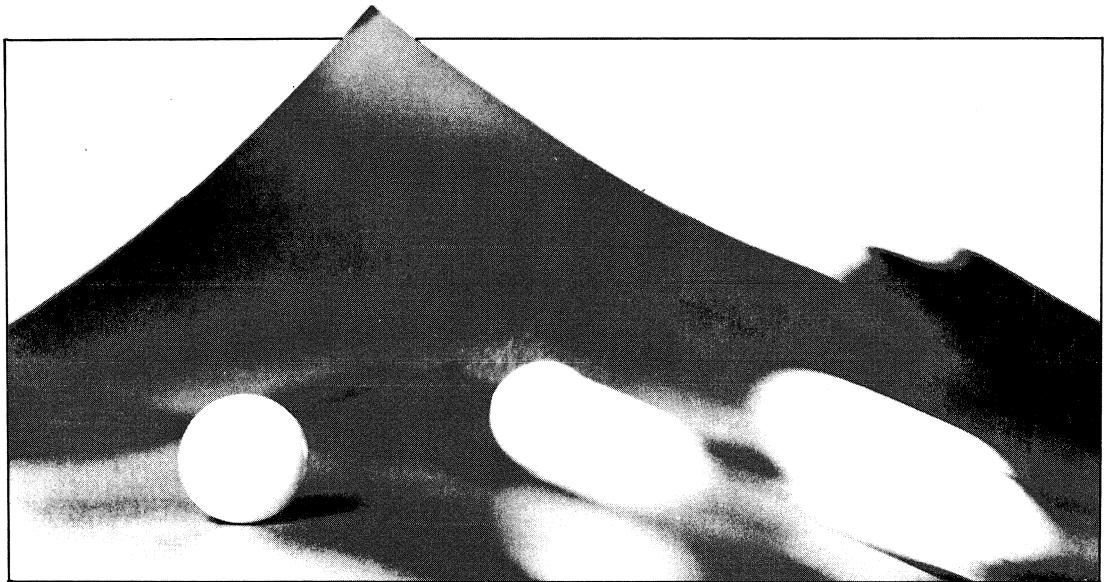
Umlaute  
à la 'VIZA'



alles gesagt (besser geschrieben), geht es mit RUN/STOP zurück ins Menü. Für Skeptiker, die sich vor dem Eintüten noch mal von der Güte Ihres Schreibens überzeugen wollen, ist der zweite Menüpunkt unabdingbar. Ansehen ist erlaubt, doch was steht, das steht! Vielleicht läßt sich der Inhalt noch ein bißchen besser verpacken. In Anlehnung an ein verbreitetes Sprichwort: Die Farbe macht das Bild. Und wenn das nicht reicht, wähle man einfach eine andere Schriftart. Einige der zur Verfügung stehenden Zeichensätze sind des Deutschen mächtig. Im Kasten finden alle, vor allem diejenigen, die in weiser Vorausschau auf den Electrobrief keine Textverarbeitung ihr eigen nennen, die Tastenbelegung für die Umlaute. Wählt man den entsprechenden Zeichensatz vor dem Schreiben, erleichtert das die Sache ungemein.

Damit endlich die Post abgehen kann, bleibt als letztes die Übertragung des Textes auf die Scheibe. Einen eindeutigen Namen unter 'Text abspeichern' eingeben, zum Beispiel 'Lies mich!', und ab damit! Natürlich sollten nur stabil verpackte Disketten in die große weite Welt entlassen werden, damit der Empfänger die Botschaft auch entziffern kann. Wider allen Unkenrufen stärkt der elektronische Brief das Briefgeheimnis, muß der Empfänger doch über einen C64 mit Floppy verfügen und(I) diesen auch bedienen können. Trifft das alles zu, weiß er bestimmt auch, was er tun muß: LOAD "Lies mich!", 8,0 und RUN. pan

<sup>1</sup>Ein entsprechender Byte-Compactor ist in der Ausgabe 5/87 veröffentlicht.



# Gegen Zeit und Raum

## Spiel: Time Race

Ziel des Spiels ist es, einen kleinen Ball über eine Rennstrecke möglichst schnell ins Ziel zu bringen. Der Ball darf weder die Streckenbegrenzung noch eines der bewegten oder unbewegten Hindernisse berühren. Tut er es doch, verliert man den Ball und fängt wieder beim Startpunkt an; für alle fünf Levels liegen insgesamt sechs Bälle bereit. Ist der Ball, von unten kommend, in das mit „Goal“ bezeichnete Loch gerollt, geht's weiter mit dem nächsten Level.

Taste	Funktion
A	hoch
Z	unten
.	links
/	rechts

**Es geht auch ohne Joystick:**  
mit diesen Tasten wird der Ball gelenkt.

**Blinder Eifer und übertriebene Hektik** bringen in diesem Videospiel den Ball nie ins Ziel. Eine schnelle, ruhige Hand, genügend Übersicht und die obligatorische Portion Gelassenheit sind die Voraussetzungen, um Level für Level sein Punktekonto nach oben zu treiben.

Das klingt einfacher, als es ist. Hauptgegner sind nämlich nicht die diversen bewegten Objekte, Hauptgegner ist die Zeit. Circa 25 Sekunden, aufgeteilt in sechs „Zeitpunkte“, stehen zur Verfügung – ohne die gewisse zielsichere Eile geht's nicht vorwärts. Ist die Zeit abgelaufen, hat man das Spiel verloren. Je schneller man eine Runde schafft, über desto mehr Punkte darf man sich freuen, die unter „Score“ angezeigt werden.

Speicherbelegung nach dem Start	
Adresse (hex.)	Adresse (dez.)
\$27FD	10237
\$3905	14597
\$3e80-\$46B0	16000-18096

Zeichensatz,  
Sprites  
Programm

Wer alle fünf Levels geschafft hat, kann das ganze Spiel mit erhöhtem Schwierigkeitsgrad wiederholen. Dem zuständigen Redakteur ist dies bis zum Zeitpunkt des Schreibens dieser Zeilen nicht gelungen, aber bevor das Spiel auf die INPUT-Diskette kommt, wird's ja vielleicht noch mal was. Eher neugierige als zielsichere Menschen können sich die verschiedenen Spielstufen nach Betätigen der Pfeil-links-Taste ansehen; und wenn einem das Ganze zu hektisch wird, bringt die F1-Taste Ruhe bis zur nächsten Tastaturbetätigungen.

Gespielt wird wahlweise mit Joystick in Port 2 oder mit den Tasten 'A';'Z';' und '/', die (in dieser Reihenfolge) den Richtungen oben, unten, links und rechts entsprechen.

K. Liebig/J.S.

# Computer für Mucker

# COMPUTER & ELEKTRONIK

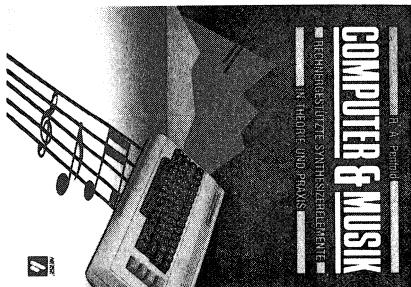
Der Homecomputer als Hilfsmittel zur elektronischen Klangerzeugung

- Stichworte Sequenzer, MIDI – Schnittstellen, Soundgeneratoren, Digitalumsetzer, Komander, Mehrkanal-Generatoren.

Sämtliche Themen werden leicht nachvollziehbar behandelt. Vorausgesetzt wird etwas Erfahrung in der Programmierung von Computern und im Aufbau einfacher Schaltungen.

Broschur, 108 Seiten  
DM 18,80  
ISBN 3-922705-37-5

Im Buch-, Fachhandel oder beim Verlag erhältlich.  
Verlag  
H. Heise GmbH  
Postfach 61 04 07  
3000 Hannover 61



# Information schafft Wissens- vorsprung.

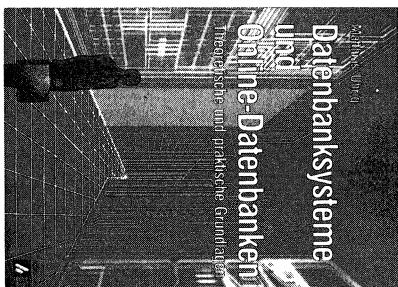
# COMPUTER- BUCH

Datenbanksysteme speichern und verarbeiten Informationen. Das Buch vermittelt Einblick in die Thematik und unterstützt Entscheidungen in allen Wirtschaftsbereichen.

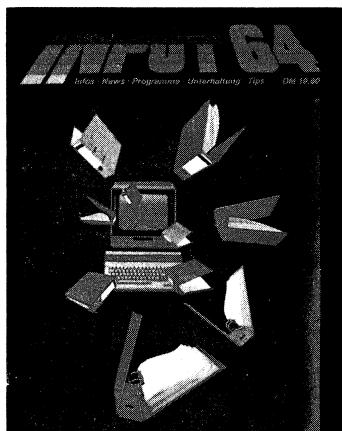
Am Beispiel dBASE III, dem Marktführer bei PC-Datenbanksystemen, werden die dargestellten theoretischen Aspekte verdeutlicht. Ein eigener Teil ist dem hochaktuellen Thema „Online-Datenbanken“ gewidmet.

Broschur, 173 Seiten  
DM 36,80  
ISBN 3-88229-133-8

Im Buch-, Fachhandel oder beim Verlag erhältlich.  
Verlag  
H. Heise GmbH  
Postfach 61 04 07  
3000 Hannover 61



## Am 7. Dezember an Ihrem Kiosk: INPUT 64, Ausgabe 12/87



Wir bringen unter anderem:

### Dateiverwaltung

Jetzt, sozusagen als Weihnachtsgeschenk, ist sie da: die indexsequentielle Dateiverwaltung für jedermann. Mit ihr können Sie so gut wie alles verwalten, von Adressen bis hin zu Ihren gesammelten Werken von INPUT 64. Apropos INPUT: mitgeliefert wird ein Inhaltsverzeichnis über drei Jahre, Monat für Monat geballte Software. Alles, was bisher in diesem Magazin erschienen ist – von Januar 85 bis Dezember 87. Damit Sie sich einen Überblick verschaffen können.

### PLH

Eine hilfreiche Spracherweiterung ist diese Zusammenstellung von Tools, die Sie mit „Programmers Little Helper“ erhalten. Zeilen können mit 'Renumber' umnummiert werden, einen bestimmten Bereich Ihres BASIC-Programmes (von Zeilennummer bis Zeilennummer) können Sie abspeichern und mit 'Merge' in ein anderes BASIC-Programm integrieren. Außerdem haben Sie die Möglichkeit, mit 'Delete' Zeilennummern zu löschen, Disk-Befehle zu senden, mit 'Find-Change' bestimmte Variablen, Wörter oder BASIC-Befehle zu suchen und gegebenenfalls auszutauschen und vieles mehr.

### Fighting Hardware

Bei diesem Spiel wird gekämpft. Aber nicht, daß Sie denken, Ihr C64 macht sich selbstständig. Man rangelt hier um Punkte. Die Hardware, die auf dem Bildschirm bei diesem Spiel gegeneinander antritt, ist ein Joystick und eine Maus, die Sie und Ihr Spielpartner bedienen müssen. Im ersten Moment sieht es zwar sehr einfach aus, aber wehe, wenn es blitzt . . .

### c't – Magazin für Computertechnik

Ausgabe 12/87 – ab 13. November am Kiosk

Projekt: CP/M-Karte mit 8-MHz-Z80 für den C64 \* Software-Know-how: Digitale Filter in der Theorie, in BASIC und in Maschinensprache \* Prüfstand: 24-Nadel-Drucker unter 2500 DM im Vergleich \* Programm: Fraktale Landschaften für Amiga, Hardcopy-Routine für alle PC-Grafik-Adapter \* CP/M+ liest MSDOS-Disketten \* u.v.a.m

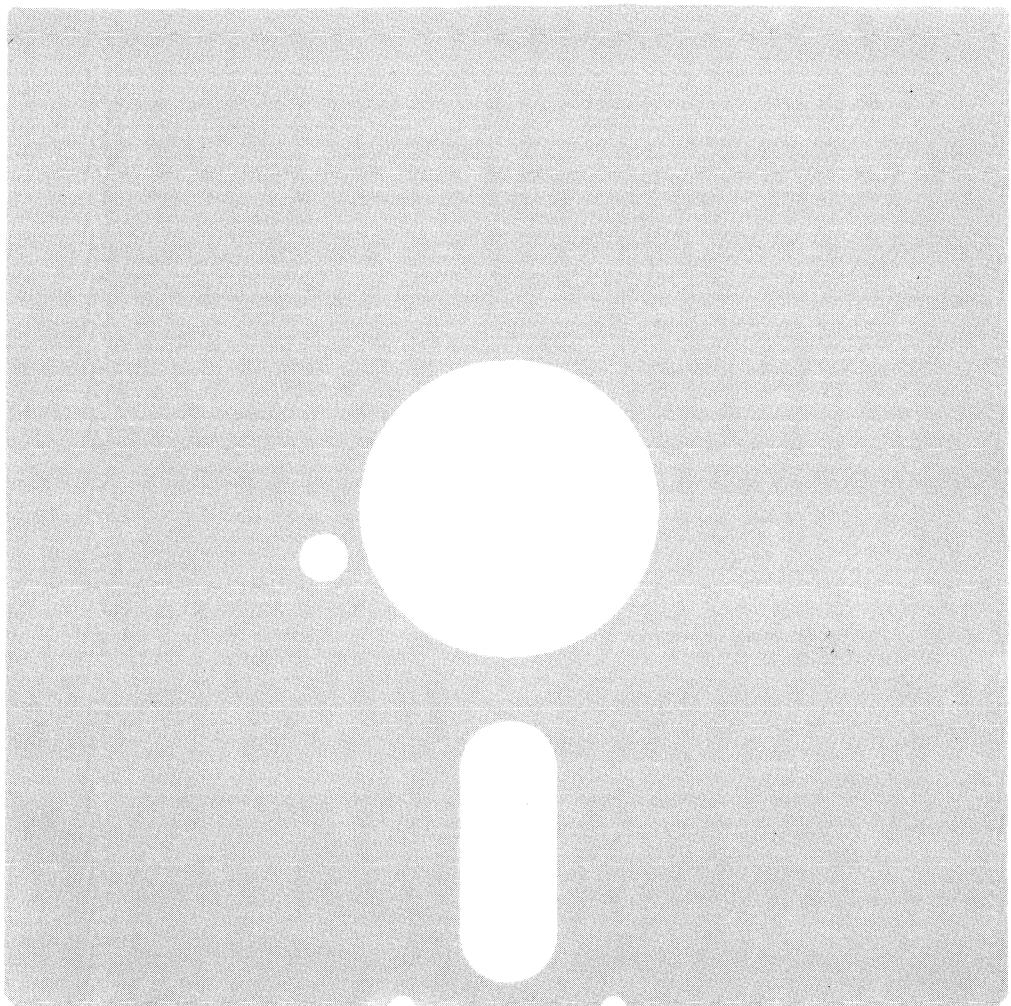
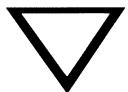
### elrad – Magazin für Elektronik

Ausgabe 12/87 – ab 30. November am Kiosk

Sonderteil „elrad 13“: Computer-Kochbuch mit Hard-, Software und Praxistips \* Stromversorgung: Abwärtsregler 12 V . . . 57 V/9,5 A \* Haus & Hof: Marderscheuche mit Ultraschall \* Marktreport: Schrittmotoren \* Grundlagen: Leistungsendstufen \* Bühne/Studio: MIDI-Sender \* u.v.a.m

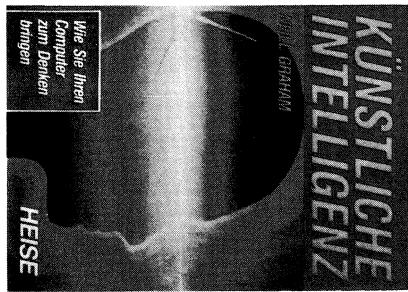


**Bitte zum Entnehmen der Diskette die Perforation  
an den markierten Stellen aufreißen.**



# K I Die Computer- anwendung von morgen.

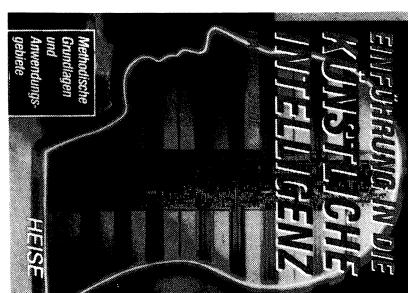
# COMPUTER- BUCH



Eine solide Einführung in die Hauptprinzipien der KI-Programmierung. Beschrieben wird, was künstliche Intelligenz ist und wie sich die Entwicklung Schritt für Schritt dahin vollzogen hat. Die Problem-Definition ist ein Schwerpunkt und wird an zahlreichen Beispielen und Methoden aufgeführt.



Theoretische Informationen über künstliche Intelligenz werden in konkrete Programme umgesetzt, die der Leser ausprobieren, verstehen und erweitern kann. Zum Experimentieren dienen dem fortgeschrittenen Hobby-Programmierer vor allem die Beziehe Suchverfahren und Spielstrategie.



Der umfassende Einblick in diesen hochaktuellen Bereich der Computerprogrammierung ermöglicht es dem Leser, sich sein eigenes Urteil über Chancen und Grenzen der künstlichen Intelligenz zu bilden. Die methodischen Grundlagen der KI und ihre wichtigsten Anwendungsfelder werden vorgestellt.

Broschur, 243 Seiten  
DM 44,80  
ISBN 3-88229-012-9

Broschur, 219 Seiten  
DM 44,80  
ISBN 3-88229-126-5

Broschur, 267 Seiten  
DM 49,80  
ISBN 3-88229-018-8

Verlag  
H. Heise GmbH  
Postfach 61 04 07  
3000 Hannover 61

**HEISE**

