DAS ELEKTRONISCHE MAGAZIN 1/88

INPUT 64

Infos · News · Programme · Unterhaltung · Tips

DM 19,80

Unverbindliche Preisempfehlung

öS 170,- sfr 19,80

Struktur und Grafik

Ist BASIC

Spracherweiterung

Gewinn und Verlust

Lohnsteuer '87

Finanzvoraussage

Spiel und Jagd

Lady Duck

Video-Klassiker

Datensicherheit:
PW-Codierer
Disc Aid

Strategie:
StraMax

Serien:
64er Tips
Compiler-Know-how

Kontakt:
C64 lernt C16-BASIC

Über 140 KByte Software.
Ohne Abtippen.

INPUT — ein Magazin aus dem Verlag Heinz Heise GmbH & Co KG, Posttach 61 04 07, 3000 Hannover 61

Hinweise zur Bedienung

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

Bitte entfernen Sie vor dem Laden eventuell vorhandende 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 Schnellader 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-Drukker und kompatible Geräte. Das Programm wählt automatisch die richtige Geräteadresse (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 Formatieren einer neuen Diskette oder Umbenennen eines Files. Für den zu sendenden Befehls-String gilt die übliche Syntax, natürlich ohne ein- und ausführende Hochkommata. 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 ins INPUT-Betriebssystem eingebaute Schnelladeverfahren 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)!

„Wat de Bur nich kennt, dat fret he nich", sagen wir im Norden etwas despektierlich zu jenen, die mit „son neumodisch Krams nichs to don hebben wolln."

Nun kann man unter dem besagten "neumodisch Krams" natürlich fast alles und auch Grundsätzliches verstehen. Allerdings brauchen wir unter uns nicht über die Akzeptanz von Computern schlechthin zu sprechen. Um aber auf den Bauern zu sprechen zu kommen; ein wenig steckt er in jedem von uns, denn eingefahrene Gleise sind ja auch gleichzeitig so bequem.

Eine neue Programmiersprache oder Spracherweiterung zu lernen ist natürlich am Anfang mit mehr oder weniger viel Mühe verbunden, und ob sich die Arbeit gelohnt hat, kann leider erst überprüft werden, wenn der Lernprozeß abgelaufen ist und das Neue dann mit dem Altbekannten verglichen werden kann.

Nehmen Sie als Beispiel unsere BASIC-Erweiterung aus dieser Ausgabe. Der Kern dieser Erweiterung besteht aus Befehlen für Schleifen- und Abbruchbedingungen. Eigentlich nichts Neues, denn Sie konnten diese Strukturen bis gestern auch mit vielen, vielen GOTO-Befehlen nachbilden.

Bis gestern, denn eigentlich ist die GOTO-Hüpferei der Programmierstil der 60er Jahre und das nicht, weil die Programmierer das so praktisch fanden, sondern weil die älteren Interpreter ihnen keine anderen Möglichkeiten an die Hand gaben. Nur: so recht übersichtlich sind diese Programmgebilde trotz großer Mühen nicht gerade geworden. Und die – nicht nur für kommerzielle Software – unabdingbare Programmpflege und Wartung wurde durch den Spaghetti-Code auch nicht gerade erleichtert.

Unabhängig von einzelnen Programmiersprachen wurden neue Befehlssequenzen entwickelt. Ziel war, dem Programmierer Strukturbefehle zur Verfügung zu stellen und die Möglichkeit, modulare Programme zu entwickeln. Das Schlagwort von der „strukturierten Programmierung" war geboren.

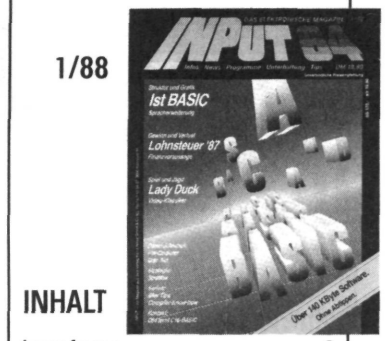
Diese positive Entwicklung ging leider an den meisten festinstallierten BASIC-Interpretern vorbei. Und so kann auch der C64 von Haus aus nur mit einem spartanischen Befehlsvorrat dienen.

Natürlich kann unsere BASIC-Erweiterung nicht alle „neumodischen" Anforderungen an strukturierte Programmiersprachen erfüllen, aber innerhalb der gegebenen Rahmenbedingungen (es bleibt ja nach wie vor ein BASIC-Interpreter) wird doch schon einiges geboten.

Auch wenn Ihre ersten Gehversuche mit IstBASIC vielleicht wiederholt mit der Meldung „syntax error" enden, sollten Sie sich doch durchbeißen, denn am Ende werden Sie ein Stückchen moderne Programmierung kennen gelernt haben. Dieses völlig neue BASIC-Gefühl sollten Sie sich jedenfalls nicht entgehen lassen.

Wolfgang Möhle

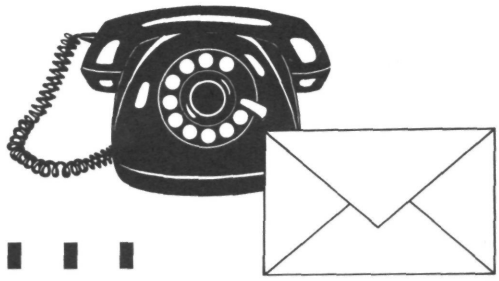
PS: Für alle südlich der Mainlinie; der erste Satz besagt ungefähr: Was der Bauer nicht kennt, das will (frißt) er nicht, weil er mit modernen Sachen nichts zu tun haben will.



1/88

INHALT

Leser fragen . . .



Aufmachung verärgert

Herr W. Schmidt, Berlin, hat den Nagel auf den Kopf getroffen, betreffs „zum neuen Heft". (Es geht um eine in Ausgabe 11/87 veröffentlichte Lesermeinung zum neuen Erscheinungsbild von INPUT 64, d. Red.) *Die jetzige Aufmachung ist nicht das Optimalste . . . Man sammelt die Hefte – vor allem haben sie alle wirklich schöne Titel, die jetzt nach vielem Gebrauch brechen. Und dann hat man nur so eine Art Loseblattsammlung.*

Etwas Nachdenken hätte gereicht, um diese Aspekte zu erkennen. Vielleicht wäre dann auch der Groschen gefallen: der Umschlag mit Diskette kann nochmals um das Heft gelegt und dann zum Heftesammeln einfach abgerissen werden . . . Ich sage: Mist mit den Neuerungen!

A. Miedreich, Mainz

Kalender druckreif

Im Terminkalender (Ausgabe 11/87) spielt der Drucker verrückt . . . Ich habe einen C64, Floppy 1541 und als Drucker den Star Sg-10C *M. Paudler, Backnang*

Das Programm ist für die Drucker der MPS-Serie von Commodore und Kompatible geschrieben. Wegen der auf diese Modelle angepaßten Steuerzeichen für Fettschrift und ähnliches kann es zu unschönen Effekten bei „Fremddruckern" kommen. Besitzer solcher Drucker können das Programm ändern, die entsprechenden Befehle befinden sich nach den Zeilennummern 2170, 2820 und 3290. Bei der Gelegenheit können auch die Zeilen 1083 und 1084 gelöscht werden, sie sind nur innerhalb von INPUT 64 wichtig; im „Hausgebrauch" kann es vorkommen, daß Programme, die eventuell im Adreßbereich ab 49152 liegen, durch die POKEs in diesen Zeilen zerstört werden.

Zu beachten ist außerdem folgendes: in das Programm sind die Maschinenspracheroutinen PrintAt und InLine eingebunden. Um das BASIC-Listing verändern zu können, muß das Programm mit RUN gestartet und dann mit RUN/STOP abgebrochen werden. Vor dem Abspeichern der veränderten Fassung **muß** man im Direkt-Modus die Befehlsfolge

POKE44,8:POKE43,1

eingeben, damit die Maschinenspracheteile nicht verlorengehen. (d. Red.)

Bildübertragung

Wie kann ich mit IMC-BASIC (der Multicolor-Grafikerweiterung aus Ausgabe 9/87) erstellte Bilder in eigene BASIC-Programme einbinden? *T. Bretz, München*

Dazu bedarf es etwas Programmieraufwands. Wenn vorausgesetzt wird, daß der Grafikbildschirm an Adresse 8192 liegen soll, reichen zunächst folgende BASIC-Zeilen:

```
10 POKE:56,31:CLR:REM SPEICHER BEGRENZEN
. . . Ihr Programm . . .
61000 SYS 57812 FI$,8:POKE 185,0
61005 REM PARAMETER FUER LOAD-FUNKTION
61010 POKE 780,0:POKE 781,0
61020 POKE 782,32:REM HIGH-BYTE GRAFIK
61030 SYS 65493:REM BILD LADEN
61040 V=53248:REM BASIS-ADRESSE VIC
61045 REM JETZT GRAFIK EINSCHALTEN
61050 POKE V+17,PEEK(V+17) OR 2↑5
61061 POKE V+22,PEEK(V+22) AND (255-2↑4)
61070 POKE V+24,PEEK(V+24) OR 2↑3
61080 REM GRAFIK IST AN
61090 RETURN
```

Dieses Unterprogramm kann mit GOSUB 61000 angesprungen werden, die String-Variable FI$ muß mit dem Namen des Bildes versorgt sein. Das ist allerdings nicht alles – der GSAVE-Befehl von IMC-BASIC speichert nämlich nur den Grafikbildschirm, nicht aber die Farbinformationen ab. Diese müßten durch entsprechendes 'POKEn' des Farb- und des Video-RAMs sowie der Adresse 33 im Video-Chip noch gesetzt werden. Da es bei komplizierten Bildern etwas mühselig werden kann, sich die notwendigen Farbinformationen zu notieren, sollte man diese Werte schon unter IMC-BASIC durch PEEK auslesen und auf Diskette ablegen. (d. Red.)

Dienstag ist Lesertag

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

Probleme, Tips, Debugging — Leseranfragen zum Speedcompiler

Bedingte Sprünge

Dieses Programm akzeptiert keine IF-GO-TO-Sprünge. Da ich ausschließlich auf diese Art programmiere, hat mich das sehr verärgert . . . Warum weist der Speedcompiler bedingte Sprungbefehle ab? *K. Zelt, Rohrhof*

Es handelt sich um ein typisches „Wie sag ich's ihm?"-Problem. Der C64-BASIC-Interpreter akzeptiert zwar die Syntax „IF AUSDRUCK GOTO ZEILENNUMMER", diese ist aber nicht normgerecht. Auch im Handbuch ist der bedingte Sprungbefehl mit der Syntax „IF AUSDRUCK THEN BEFEHLSBLOCK" erklärt. Einen Sonderfall bildet die Anweisung „IF AUSDRUCK THEN GOTO ZEILENNUMMER". Sie ist zwar syntaktisch korrekt, aber das „GO-TO" kann weggelassen werden, also „IF AUSDRUCK THEN ZEILENNUMMER". So steht's auch in den Beispielprogrammen im Handbuch, und der Speedcompiler mag's nicht redundant mit „THEN GOTO".

Übrigens: Für all diese Fälle sind die Befehle „FIND" und „CHANGE" der in Ausgabe 12/87 veröffentlichten BASIC-Erweiterung PLH wärmstens zu empfehlen.

(d. Red.)

DEF(FN) packt's nicht

Ich habe ein Programm, das in hochauflösender Grafik arbeitet. Es dauert sehr lange, bis die Grafik entsteht. Aber da gibt es ja Ihren Compiler . . . Mein kompiliertes Programm läuft zwar, aber es kommt keine 3-D-Grafik, sondern nur gerade Linien. *A. Wißkirchen, Köln*

Das Problem, das bei Herrn Wißkirchen auftauchte, hatte nicht mit dem Grafikmodus zu tun, sondern der Autor des Speedcompilers entdeckte einen Fehler in der DEF(FN)-Behandlung. Im besagten Grafikprogramm war eine Funktion folgender Form definiert:

```
DEF FNZ(X)=10-1/SIN(X)+1.08)-1/(SIN(Y)+1.08)
```

Der Speedcompiler behandelt die zweite, nicht als Funktionsargument übergebene Variable Y falsch; es darf in dem der Funktion zugewiesenen Ausdruck nur das Funktionsargument als Variable benutzt werden. Man kann diesen Fehler leicht umgehen, indem man Funktionen aufteilt, in diesem Fall folgendermaßen:

```
DEF FNZ1(X)=10-1/SIN(X)+1.08)
DEF FNZ2(X)=-1/(SIN(X)+1.08)
```

Die beiden Funktionen können dann bei Gebrauch additiv eingesetzt werden, etwa

```
A=FNZ1(X)+FNZ2(Y)
```

Beachten Sie aber, daß Funktionen als Argument von Funktionen nicht erlaubt sind. (d. Red.)

Negative DATAs unerwünscht

Ich habe ein Notendruckprogramm in INPUT-BASIC[1] geschrieben. Der Kompilationsvorgang wurde ohne Fehlermeldung abgeschlossen, und trotzdem kam am Anfang des Programms die Fehlermeldung „TYPE MISMATCH ERROR IN xxxx". Nach einigem Suchen fand ich heraus, daß der Compiler keine negativen DATA-Statements auslesen kann. Ich habe das Problem umgangen, indem ich die Werte bis in den positiven Bereich erhöht und dann in einem weiteren Rechenschritt durch Subtraktion die richtigen Zahlenwerte wiederhergestellt habe. P. Bücher, Wirges

Die von Herrn Bücher gewählte Methode ist ein Weg, diesen Bug im Speedcompiler zu umgehen. Eine andere Methode: die DATA-Statements erst einer String-Variablen zuweisen, die dann durch die VAL-Funktion umgewandelt wird. Ein für allemal kann das Problem behoben werden, wenn man den Compiler mit folgenden POKEs „entwanzt":

```
POKE 10345,43
POKE 10351,45
POKE 10931,45
```

Es hat sich noch eine weitere Inkompatibilität in der READ-Funktion ergeben: die Schreibweise, den Wert '0' durch ein Komma zu ersetzen, also „10,,10" statt „10,0,0,10", ist nicht möglich. (d. Red.)

DIM-Befehl mit Mißverständnissen

Nach der Begeisterung über den Speedcompiler gleich ein paar Fragen:

- *Kann man mehrere Felder hintereinander dimensionieren (DIM A$(50),B%(50),C(50) . . .?*
- *Was bedeutet „Der DIM-Befehl muß allein in einer Zeile stehen"? Im Beispielprogramm (Zeile 205) steht er nicht allein in der Zeile . . .*
- *Nachdem ein Speedcompiler-Programm beendet ist, würde ich die RUN/STOP- und RESTORE-Tasten gern wieder einschalten lassen. Wie geht das?*

S. de la Motte, München

Die Bemerkung, der DIM-Befehl müsse allein in einer Zeile stehen, ist offenkundig etwas mißverständlich. Vorgeschrieben ist folgendes:

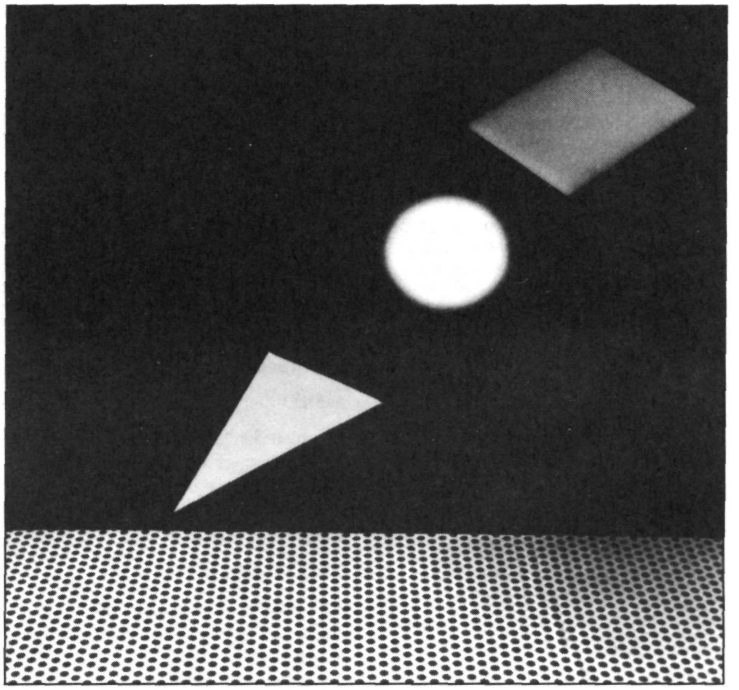
NACH dem DIM-Befehl dürfen keine anderen Befehle in der Zeile folgen, mit zwei Ausnahmen: REM-Compiler-Option-Stringlänge und weitere Argumente für den DIM-Befehl. Das REM darf natürlich nicht direkt nach dem DIM-Argument stehen, sondern durch einen Doppelpunkt abgetrennt. "Weitere Argumente für den DIM-Befehl" sind, wie in der üblichen BASIC-Syntax, die Namen und Dimensionierungen weiterer Arrays. Beispiele:

```
10 DIM A$(100): REM£10
10 DIM A(100,10),B(10,20)
```

Sollen die String-Längen mehrerer Felder festgelegt werden, muß jede Dimensionierung in einer neuen Zeile stehen.

Zum Problem des Sperrens der RUN/STOP-RESTORE-Taste durch die Compiler-Option 'REM£S2': nach Beendigung des kompilierten Programms wird diese Verriegelung wieder aufgehoben. (d. Red.)

Kurz, kompakt, kompatibel



BASIC-Erweiterung: Ist BASIC

Ist es nun BASIC oder Ist es doch kein BASIC, könnte man sagen. Ich kann Ihnen versichern, es Ist BASIC. Ziemlich verwirrend, oder? Ist aber ganz leicht zu entwirren: das Ist steht für INPUT-Struktur-BASIC. Wenn Sie sich aber erst mal mit der neuen Erweiterung beschäftigt haben, werden auch Sie zu der Meinung kommen, es könnte schon eine andere Programmiersprache sein, wenn nur die Zeilennummern nicht wären. Doch wir haben sie Ihnen gelassen, damit Ihnen wenigstens etwas vertraut vorkommt.

Schon wieder 'ne BASIC-Erweiterung. Ich höre schon, wie Sie gequält aufstöhnen. Was wir Ihnen mit Ist BASIC anbieten, ist zwar eine BASIC-Erweiterung, aber beim genaueren Hinsehen werden Sie staunend die Augen aufreißen und nicht mehr vom Rechner wegkommen. Eine neue Nachtschicht ist angesagt. Ab jetzt können Sie nämlich Ihre BASIC-Programme strukturiert programmieren und die Interrupts nicht nur den Assembler-Programmen überlassen.

Immer munter rauf und runter

Außer den neuen Befehlen (es sind immerhin mehr als fünfzig) enthält die neue Erweiterung einen erweiterten Bildschirm-Editor. Er funktioniert selbstverständlich nur im Direktmodus. Dabei muß bei allen aufgeführten Funktionen die jeweilige Taste und gleichzeitig die CTRL-Taste gedrückt werden. Bei den Cursor-Tasten ist dies allerdings nicht nötig.

CURSOR auf: Ein vorhandenes Listing wird in Richtung Programmanfang gescrollt. Es muß mindestens eine Zeile auf dem Bildschirm sichtbar sein.

CURSOR ab: Ein vorhandenes Listing wird in Richtung Programmende gescrollt. Auch hier muß mindestens eine Zeile des Listings auf dem Bildschirm sichtbar sein.

Funktionstaste F1: Drücken Sie die CTRL- und die F1-Taste gleichzeitig, wird der Bildschirm gelöscht, und am unteren Rand erscheint die erste Zeile des BASIC-Programmes.

Funktionstaste F3: Drücken Sie die CTRL- und die F3-Taste gleichzeitig, wird der Bildschirm gelöscht und am oberen Rand erscheint die letzte Zeile des BASIC-Programmes.

Funktionstaste F5: Drücken Sie die CTRL- und die F5-Taste gleichzeitig, wird die Zeile

des BASIC-Programmes, in der sich der Cursor befindet, ab der Cursor-Position gelöscht.

Funktionstaste F7: Drücken Sie die CTRL- und die F7-Taste gleichzeitig, wird im BASIC-Programm an der aktuellen Cursor-Position Platz für eine neue BASIC-Zeile geschaffen.

Delete: Drücken sie die CTRL- und die Delete-Taste gleichzeitig, wird der Anführungszeichenmodus aufgehoben. Steuerbefehle werden nicht mehr als reverse Zeichen dargestellt, sondern sofort ausgeführt.

Return: Drücken Sie die CTRL- und die Return-Taste gleichzeitig, wird das aktuelle BASIC-Programm automatisch mit einem RUN gestartet.

Die Befehle

MOD(Zahl1,Zahl2)
Zahl1 wird durch Zahl2 dividiert. Der dabei auftretende Rest kann entweder einer Variablen zugeordnet oder gleich ausgegeben werden. Das Ergebnis der Division wird nicht festgehalten.

Beispiel 1: A = MOD(8,3)

Beispiel 2: PRINT MOD(8,3)

In Beispiel 1 wird der Variablen 'A' der Wert 2 zugewiesen. In Beispiel 2 wird die 2 gleich ausgegeben.

HEX(dezimalwert)
Der angegebenen dezimale Wert wird in einen hexadezimalen Wert umgewandelt.

Achtung! Das Ergebnis ist ein String und kann demzufolge nur einer String-Variablen zugewiesen werden.

Beispiel: 10 A$ = HEX(40960)
20 PRINT A$

Es wird A000 ausgegeben. Dieser Befehl erleichtert ganz erheblich die Umrechnung von dezimalen in hexadezimale Adressen.

DEC(hexadezimalwert)

Der angegebene hexadezimale Wert wird in eine dezimale Zahl umgewandelt. Außerdem muß er vierstellig sein.

Achtung! Das Ergebnis ist ein numerischer Wert und kann daher nur einer numerischen Variablen zugewiesen werden.

Beispiel 1: 10 A = DEC("A000")
20 PRINT A

Beispiel 2: 10 A$ = "A000"
20 A = DEC(A$)
30 PRINT A

Auf dem Bildschirm wird in beiden Beispielen die dezimale Zahl 40960 ausgegeben. Wie man sieht, kann auch eine String-Variable eingesetzt werden. Auch dieser Befehl erleichtert die Umrechnung ganz erheblich.

MPEEK(adresse)
Liest den Inhalt der angegebenen Adresse. Schaltet dabei aber den ROM- und den I/O-Bereich ab, so daß nur aus dem RAM gelesen wird. Beim normalen BASIC ist das nicht möglich. Eine wirklich brauchbare Erweiterung durch Ist BASIC.

MDEEK(adresse)
Ähnlich dem Befehl MPEEK, liest aber einen Zwei-Byte-Wert aus der angegebenen und der folgenden Adresse. Sie wollen zum Beispiel ausfindig machen, wo sich zur Zeit der BASIC-Anfang befindet. Im Commodore-BASIC müssen Sie dazu die beiden Adressen 43 und 44 auslesen. Dabei müssen Sie beachten, daß sich in der Adresse 43 das Low- (niederwertige) und in der Adresse 44 das High- (höherwertige) Byte befindet. Anschließend muß der Wert aus der Adresse 44 mit 256 multipliziert und der Wert aus der Adresse 43 dazuaddiert werden. Mit PRINT **MDEEK** ist das nicht mehr nötig. Es wird sofort die vollständige Adresse ausgegeben.

BCODE(ASCII-Wert oder String)
Wandelt den angegebenen ASCII-Wert oder String in den Commodore-spezifischen Bildschirmcode um. Um reverse Zeichen zu erhalten, muß man zum erhaltenen Wert 128 addieren.

Beispiel:
PRINT BCODE("a")
PRINT BCODE(65)
PRINT BCODE(65)+128 (reverses 'a')

Beide Male erscheint eine 1 auf dem Bildschirm. 1+128=129 wäre demzufolge der Bildschirmcode für ein kleines, reverses 'a'. Probieren Sie es einfach mal aus, indem Sie eine 1 in die Anfangsadresse des Bildschirms schreiben (POKE 2048,1). Oben links in der Ecke des Bildschirms erscheint dann ein kleines 'a'.

FIRE p
Ergibt den Wert 1, wenn der Feuerknopf des an Port p angeschlossenen Joysticks gedrückt wird. Wird er nicht gedrückt, ergibt das den Wert 0. Der Wert kann auch einer Variablen zugewiesen werden.

Neues für die Sprites

MOBCOLL
Mit diesem Befehl können Sie das Sprite-Kollisionsregister (53278) auslesen. Er entspricht einem PEEK (53278) im normalen Commodore-BASIC.

BACKCOLL
Das Sprite-Hintergrundkollisionsregister wird ausgelesen. Dieser Befehl entspricht einem PEEK (53279) im normalen Commodore BASIC.

MOBCHAR(sn)
Ergibt die Adresse der Bildschirmspeicherzelle, die sich ungefähr in der Mitte des Sprites mit der Nummer sn(0-7) befindet. Kann zum Beispiel nützlich sein, wenn das Sprite mitten auf einen Punkt fixiert werden soll. Dabei muß das Sprite auf dem Bildschirm sichtbar sein.

JOY p,sn,g,x,y,b,h
Das Sprite mit der Nummer sn wird mit einem Joystick, der an Port p angeschlossen ist, mit der Geschwindigkeit g innerhalb eines Feldes, das durch x, y, b und h festelegt ist, gesteuert. X und y bezeichnen die linke obere Ecke, b und h die Breite und die Höhe des Feldes. Beide Ports können unabhängig verwaltet werden. Ausgeschaltet wird die Funktion über JOY p, wobei p den Port angibt, welcher ausgeschaltet werden soll.

SYNC sn1,sn2,mo
Sprite sn2 wird neben Sprite sn1 positioniert. Dabei bedeuten:

mo = 0 links neben Sprite sn1
mo = 1 rechts neben Sprite sn1
mo = 2 über Sprite sn1
mo = 3 unter Sprite sn1

Dieser Zustand wird erst durch den Befehl **SYNC OFF** wieder aufgehoben.

Farbe ins Programm

MULTI hf,pf1,pf2,pf3
Der Befehl schaltet den Multicolormodus

ein und setzt die angegebenen Farben; Hintergrundfarbe (hf), Punktfarbe (pf1, pf2, pf3).

MONO
Der Multicolormodus wird ausgeschaltet. Es ist zu beachten, daß der Farbspeicher nicht gesichert wird, so daß eventuelle Farbgebungen auf dem HiRes-Bildschirm durch Aufruf des 'MULTI'-Befehls verlorengehen.

MSETx,y,col
EEin Punkt der Farbe col (pf1, pf2, pf3) wird bei den Koordinaten, die durch x und y gekennzeichnet sind, gesetzt. Der Wert von col bezieht sich auf die mit 'MULTI' festgelegten Farben.

Achtung: Dieser Befehl funktioniert nur im Zusammenhang mit INPUT-BASIC. Ist INPUT-BASIC nicht im Speicher, wird ein SYNTAX ERROR ausgegeben.

Lange Zeichenketten

RPT$(az,string)
Es wird eine Zeichenkette produziert, die az-mal den angegebenen String enthält.

Beispiel 1: PRINT RPT$(4,"Igel")

Auf dem Bildschirm wird die Zeichenkette "IgellgellgellgeI" ausgegeben.

Beispiel 2: 10 A$ = RPT$(4,"Hase")
20 PRINT A$

Auch hier erscheint nach dem Start des kleinen Programmes "HaseHaseHaseHase" auf dem Bildschirm. Es ist also nicht mehr nötig einzugeben:

A$ = "Hase"+"Hase"+"Hase"+"Hase". Mit diesem Befehl genügt es, wenn man eingibt, wie oft eine bestimmte Zeichenfolge wiederholt werden soll.

INSTR (string1,string2)
Gibt die Position von string2 innerhalb von string1 an. Ist der String nicht enthalten, ergibt das den Wert 0.

GARBAGE
Schaltet die Rahmenfarbe auf Rot und ruft die 'Garbage-Collection-Routine' auf. Nach Beendigung der Routine erhält der Bildschirmrahmen seine ursprüngliche Farbe wieder. Bei der Garbage-Collection – zu deutsch: Müllabfuhr – werden nicht mehr gebrauchte Strings, die mal einer String-Variablen zugeordnet wurden, beseitigt. Das kann unter Umständen lange dauern. Durch die Garbage-Routine, die Ist BASIC zur Verfügung stellt, werden diese langen Wartezeiten abgebaut.

BASIC-Allerlei

OFF
Es wird von der Erweiterung auf das normale BASIC zurückgeschaltet. Außerdem wird die CBM80-Kennung aus der Speicherstelle $8000 und folgende wieder gelöscht, das heißt, Restore und Reset verlaufen wieder normal.

CLEAR az,ez
Löscht die Bildschirmzeilen von az bis ez. Sowohl az als auch ez können entfallen, dann wird für az der Wert 0 und für ez der Wert 24 angenommen.

Beispiele:
CLEAR löscht den ganzen Bildschirm
CLEAR 5 löscht nur die fünfte Zeile
CLEAR ,6 löscht bis Zeile 6
CLEAR 4, löscht ab Zeile 4
CLEAR 4,6 löscht von Zeile 4 bis Zeile 6

CURSOR on/off
Schaltet den Cursor ein beziehungsweise aus. Beim Ausschalten bleibt kein reverses Leerzeichen zurück wie bei bisherigen Lösungen mit POKE.

NO STOP
Sperrt die STOP-Taste und verhindert unerwünschte Programmabbrüche.

NO RESTORE
Wie 'NO STOP', nur wird hier die RESTORE-Taste gesperrt.

ADMIT STOP
Gibt die STOP-Taste wieder frei. Ein Programmabbruch ist wieder möglich.

ADMIT RESTORE
Gibt die RESTORE-Taste wieder frei. Sonst wie bei 'ADMIT STOP'.

SWAP var1,var2
Vertauscht den Inhalt der Variablen var1 mit dem Inhalt der Variablen var2. Die Variablen können String- oder numerisches Format haben, müssen aber beide vom gleichen Typ sein.

Beispiel 1: 10 A = 200
20 B = 5
30 PRINT A,B

Auf dem Bildschirm erscheint erst die 200 und dann die 5.
40 SWAP A,B
50 PRINT A,B

Jetzt erscheint auf dem Bildschirm erst die 5 und dann die 200. Der Inhalt beider Variablen wurde vertauscht.

Beispiel 2: 10 A = 40
20 B$ = "HASE"
30 SWAP A,B$

Achtung! Dieses Beispiel funktioniert nicht. Variablen verschiedenen Typs können nicht vertauscht werden.

Beispiel 3: 10 A$ = "HASE"
20 B$ = "IGEL"
30 SWAP A$,B$

Bei diesem Beispiel werden wieder Variablen gleichen Typs vertauscht. Das funktioniert.

SLEEP wert
Wartet eine bestimmte Zeit. Der angegebene Wert darf zwischen 0 und 65535 liegen. SLEEP 65535 zum Beispiel wartet 1 Minute und 35 Sekunden.

CLOCK "hhmmss"
Setzt die Uhrzeit auf Stunden (hh), Minuten (mm) und Sekunden (ss). Zu lesen ist die eingestellte Uhrzeit mit PRINT CLOCK.

UP wert
Legt das BASIC-Ende fest. Läßt man wert weg, so kann man das derzeitige BASIC-Ende lesen. Außerdem werden nach Ausführung eines 'CLR alle Variablen gelöscht.

Beispiel: UP 12288 setzt das BASIC-Ende auf 12288,

PRINT UP ergibt das BASIC-Ende (normal 32768).

START wert
Wie 'UP', jedoch wird hier der BASIC-Anfang festgelegt beziehungsweise ausgelesen.

Wenn der Interrupt kommt

IRQJOBzn,fr
Ein Pseudo-Interrupt wird definiert. Sobald der Befehl **IRQon/off** den Pseudo-Interrupt

zuläßt, wird die angegebene BASIC-Zeile mit der Nummer zn mit der Frequenz fr regelmäßig angesprungen. Der Pseudo-Interrupt stoppt bei den Befehlen INPUT, INKEY, SLEEP, KEYGET, WAIT und so weiter; er stoppt nicht bei Endlosschleifen. Je kleiner dabei fr ist, um so größer ist die Frequenz, mit der die angegebene Zeile angesprungen wird, und umgekehrt: je größer die Zahl, desto kleiner die Frequenz. Es werden auch nur ganzzahlige Werte für die Frequenz im Bereich von 0 bis 255 akzeptiert.

IRQ on/off
Der beim vorhergehenden Befehl definierte Interrupt wird eingeschaltet. Ab hier wird die angegebene Zeile mit der angegebenen Frequenz regelmäßig angesprungen.

```
10 IRQJOB 100,10
20 IRQ ON
30 B$ = "HASE"
40 PRINT B$
50 GOTO 40
...
...
100 CLEAR 0: REM löscht Zeile 0
110 PRINT CHR$(19): REM Cursor nach links oben
120 PRINT " ";
130 A1$ = LEFT$(A$,2)
140 A2$ = MID$(A$,3,2)
150 A3$ = RIGHT$(A$,2)
160 PRINT A1$;":";A2$;":";A3$
170 IRQEND
```

Dieses Beispielprogramm schreibt am linken Bildschirmrand immer das Wort 'HASE' untereinander. Sowie der Interrupt aktiv ist, verzweigt das Programm zur Pseudo-Interrupt-Routine und zeigt in der obersten Zeile rechts die Uhrzeit an.

IRQEND
Dieser Befehl kennzeichnet das Ende einer Pseudo-Interrupt-Routine. Das laufende Programm wird an der unterbrochenen Stelle fortgesetzt, ähnlich einem Unterprogramm, das mit GOSUB angesprungen und mit einem RETURN beendet wird.

PULLIRQ
Hebt den Interrupt-Status auf. Nach diesem Befehl kann das Programm an beliebiger Stelle fortgesetzt werden.

Namen: mehr als Schall und Rauch

KEYLABEL ft,Label
Der Funktionstaste mit der Nummer ft wird das angegebene Label zugewiesen. Unter Label ist ein Name mit höchstens sechs Zeichen zu verstehen. Ist hilfreich, wenn nach dem Befehl KEYGET ein WHEN LABEL THEN folgt. Der Name darf aber keinesfalls Steuerzeichen enthalten!

SHOW
Es werden alle Labels angezeigt, die den Funktionstasten zugewiesen wurden. Sehr hilfreich, wenn man die Labels nicht mehr im einzelnen kennt.

WHEN Label THEN
Ist die Funktionstaste mit dem angegebenen Label gedrückt, dann wird wie bei IF-THEN weitergearbeitet, sonst erfolgt ein Sprung zur nächsten Programmzeile.

Beispiel:

```
10 KEYLABEL 1,"hase"
20 KEYGET sv$
30 WHEN sv$ = "hase" THEN GOTO 300
```

CLEARLABEL
Löscht alle Labels, die den Funktionstasten zugeordnet wurden.

LABEL "Label"
Kennzeichnet eine Programmzeile mit dem angegebenen Label. Der Befehl wird in der Programmausführung wie ein 'REM' behandelt, das heißt die in dieser Zeile nachfolgenden Anweisungen werden ignoriert.

Beispiel: 300 LABEL "HASE"

Das Programm springt

JUMP Label/numerischer Ausdruck

a) Springt zu der mit diesem Label bezeichneten Zeile.

b) Springt zu der angegebenen beziehungsweise errechneten Adresse.

Beispiel:

```
10 A = 0
20 IF A$ = "HASE" THEN A = 1
30 IF A = 1 THEN JUMP "HASE"
40 A = A+300
50 JUMP A
...
...
200 LABEL "HASE"
210 PRINT"Sprung zur Zeile 200"
220 GOTO10
...
...
300 PRINT"Wenn A = 300 ist, wird zu dieser Zeile gesprungen."
310 GOTO10
```

CALL Label/numerischer Ausdruck
Ähnlich wie 'JUMP', führt aber ein GOSUB aus. Der Rücksprung erfolgt wie gewohnt mit RETURN.

PULLSUB
Holt den letzten 'GOSUB'-Datensatz vom Stack. Das jeweilige Unterprogramm wird aufgehoben, und man kann jetzt mit einem 'GOTO' oder 'JUMP' verzweigen (siehe Kastentext – Aufstapeln).

Oldies but Goldies

Schon bei der Planung hat sich der Autor von Ist BASIC, Elmar Dägele, Gedanken gemacht, wie man eine neue BASIC-Erweiterung erstellen und das bekannte INPUT-BASIC aus INPUT64 Ausgabe 1/86 integrieren kann. Das Ergebnis liegt Ihnen hier vor. Sie können diese neue Erweiterung eigenständig oder zusammen mit INPUT-BASIC verwenden.

Nach dem Laden wird Ist BASIC ganz einfach mit einem RUN gestartet. Muß ein Neustart des Programms durchgeführt werden, so geschieht das mit SYS 36474. Ist das Programm initialisiert, muß ihm durch ein POKE 39093,x mitgeteilt werden, ob sich INPUT-BASIC ohne SuperTape (x=0), INPUT-BASIC mit SuperTape (x=2) oder keine Erweiterung (x=0) im Speicher befindet.

Farbe zum zweiten

BORDER col
Setzt die Rahmenfarbe auf col.

PAPER col
Setzt die Hintergrundfarbe auf col.

INK col
Setzt die Schriftfarbe auf col.

Füllen, verschieben, tauschen

TAKE anfang,ende,ziel
Verschiebt den Inhalt des Speicherbereiches von der Adresse 'anfang' bis einschließlich der Adresse 'ende' nach 'ziel'. Ziel ist die Anfangsadresse des Zielbereichs.

LOFILL zl,sp,b,h,z
Füllt den durch zl, sp, b und h definierten Textbildschirm mit dem Zeichen z, wobei z der Bildschirmcode des Zeichens sein muß. Es dürfen Werte übergeben werden: für zl (0-39), für sp (0-24), für b (1-40) und für h (1-25).

Beispiel: 10 ZL = 10
20 SP = 3
30 B = 10
40 H = 7
50 Z = 1
60 LOFILL ZL,SP,B,H,Z

Der Bildschirm wird ab der Zeile 10 und der Spalte 3 bis zur Zeile 17 und zur Spalte 13 mit einem 'a' gefüllt.

INVERT zl,sp,b,h
Wie LOFILL, nur wird der angegebene Bereich invertiert.

COPY
Der Originalzeichensatz wird nach 12288 ($3000) bis 16383 ($3FFF) kopiert. Dieser Bereich sollte (mittels UP und START) vor dem Überschreiben geschützt werden.

CHANGE mo,zn,b1,b2,b3,b4,b5,b6,b7,b8
Definiert das Zeichen mit dem Bildschirmcode zn um. b1-b8 sind die Bytefolgen des neuen Zeichens. Es wird vorausgesetzt, daß sich an der Stelle von 12288 bis 16383 ein Zeichensatz befindet (siehe COPY). Der Modus mo gibt an, auf welchen Zeichensatz sich die Änderungen beziehen. Wird für mo eine 0 eingesetzt, ist der Groß-/Grafik-Zeichensatz gemeint. Setzt man für mo eine 1 ein, meint man den Groß-/Klein-Zeichensatz.

Ein Zeichen besteht ja bekanntlich aus acht Bytes. Das erste ist das obere, das letzte das untere Byte.

CHARS on/off
Schaltet einen Zeichensatz beziehungsweise den Originalzeichensatz ein/aus, der sich im Bereich von 12288 bis 16383 befinden muß.

Struktur in BASIC

REPEAT
Kennzeichnet den Anfang einer REPEAT-UNTIL-Schleife.

UNTIL bedingung
Kennzeichnet das Ende einer REPEAT-UNTIL-Schleife. REPEAT-UNTIL gehören zusammen wie FOR-NEXT. Die beiden Befehle bilden einen Block. Innerhalb dieser Schleife können mehrere andere Befehle stehen. Der Befehl UNTIL benötigt jedoch eine Bedingung, da aus dieser Schleife sonst eine Endlos-Schleife würde. Ist die Bedingung nicht zutreffend, springt das Programm zu dem Befehl, der gleich nach dem REPEAT steht, und wird dort fortgesetzt. Ist die Bedingung zutreffend, wird das Programm nach dem UNTIL ganz normal abgearbeitet.

Beispiel: 10 REPEAT
20 : KEYGET A$
30 UNTIL A$ = "Z"

DO
Kennzeichnet den Anfang einer DO-LOOP-Schleife.

LOOP
Kennzeichnet das Ende einer DO-LOOP-Schleife. Im Gegensatz zur REPEAT-UNTIL-Schleife können bei DO-LOOP mehrere Abbruchbedingungen innerhalb der Schleife definiert werden. Der Befehl LOOP muß immer am Anfang einer Zeile stehen.

Beispiel: 10 DO
20 IF A$ = "Z" THEN: EXIT
30 IF PEEK(2) = 7 THEN: EXIT
40 LOOP

Bei diesem Beispiel wartet das Programm so lange, bis entweder die Taste 'Z' gedrückt wird oder die Speicherstelle 2 den Wert 7 annimmt.

EXIT
Setzt das Programm hinter dem nächstfolgenden LOOP-Befehl fort (siehe LOOP).

PULL LOOP
Dieser Befehl hebt die aktuelle DO-LOOP- oder REPEAT-UNTIL-Schleife auf. Der entsprechende Datensatz wird vom Stack entfernt (siehe Kastentext – Aufstapeln). Sicherlich werden Sie sich jetzt fragen, wo eigentlich der unentbehrliche Befehl IF-THEN-ELSE bleibt. Den können wir Ihnen leider nicht bieten. Trotzdem hier ein Beispiel, wie man diesen Befehl nachbilden kann.

Beispiel:
10 IF A = 1 THEN GOSUB 1000: GOTO 30
20 GOSUB 2000
30 REM Hier geht's weiter
...
...
1000 REM Hier ist der IF-THEN-Teil
1010 :
1020 RETURN
...
...
2000 REM Hier ist der ELSE-Teil
2010 :
2020 RETURN

Ist die Bedingung erfüllt, nämlich A = 1, dann geht es als normale IF-Bedingung zur Zeile 1000 und springt anschließend durch das GOTO zur Zeile 30. Ist die Bedingung nicht erfüllt, nämlich A <> 1, dann arbeitet das Programm bei der nächsten Zeile weiter (ELSE-Bedingung).

INPUT und die Eingabe

LINKEY z,s,"kommentar",l,df$,z$
Ermöglicht einen Kommentar bei dem Befehl 'INKEY' des INPUT-BASIC. Steht INPUT-BASIC nicht zur Verfügung, wird ein SYNTAX ERROR ausgegeben. Der Kommentar wird an der durch z und s angegebenen Zeilen- und Spaltenposition ausgegeben, unmittelbar dahinter erfolgt die Eingabe. Beim altbekannten Befehl INPUT konnte ein Kommentar eingegeben werden, was beim INKEY-Befehl nicht der Fall war. Das wird durch den LINKEY-Befehl verbessert.

KEYGET sv$
Wartet auf Tastendruck und weist der String-Variablen sv$ das Zeichen der gedrückten Taste zu. Folgt dem Befehl keine Variable, wird die gedrückte Taste aus dem Tastaturpuffer gelöscht.

Wichtig ist auch der herkömmliche BASIC-Befehl THEN. Folgt diesem Befehl ein Befehl des Ist BASIC, muß nach dem THEN ein Doppelpunkt (:) stehen. Sonst kommt es zu einem SYNTAX ERROR. K.-F. Probst

Zeitansage

Interrupt-Tips, Folge 3

Zum Einstieg sei kurz an die letzte Ausgabe erinnert: Dort lernten Sie den VIC als vielseitigen Baustein mit eigener IRQ-Leitung kennen und erfuhren, wie sich eigene VIC-Interrupt-Routinen in den System-Interrupt einbinden lassen. Diesmal rückt der CIA (Complex Interface Adapter) in den Vordergrund. Auf der Platine des C64 sind sogar gleich zwei dieser Bausteine vorhanden, von denen einer IRQ- und der andere NMI-Interrupts auslöst.

Versuchen Sie selbst einmal, die Häufigkeit der Interrupt-Aufrufe zu ändern, indem Sie in die Adresse 56325 ($DC05, die Highbyte-Adresse des Timer A) einen Wert zwischen 1 und 255 schreiben. Je größer Sie den Wert wählen, um so träger wird das Blinken des Cursors und die Abfrage der Tastatur. Ein kleiner Wert hingegen verhilft dem Cursor zu ungeahnter Beschleunigung, einem aktiven Programm zum Gegenteil: dessen Arbeitsgeschwindigkeit sinkt natürlich bei einem häufiger abgearbeitetem Interrupt, auch wenn dieser Effekt bei einem BASIC-Programm noch keine allzu auffälligen Verzögerungen mit sich bringt.

Der Complex Interface Adapter ist ebenso wie der VIC ein Peripheriebaustein aus der 65xx-Familie. Um die Synchronisation mit externen Geräten zu gewährleisten, ist er neben Ein-, Ausgabe- und Handshake-Leitungen mit zwei Intervall-Timern ausgerüstet.

Jeder dieser Timer verfügt über einen 16-Bit-Zähler und einen 16-Bit-Speicher. Außerdem kann jeder Timer über ein zugehöriges Steuerregister auf verschiedene Betriebsarten eingestellt werden. In Abhängigkeit von einem Taktimpuls, normalerweise der Systemtakt, wird der Timer wie bei einem Countdown heruntergezählt. Bei einem

Um das Phänomen Zeit in den Griff zu bekommen, bedient man sich im allgemeinen einer Uhr. Im C64 „ticken" gleich mehrere. Der Baustein, der sie verwaltet, ist aber auch in anderer Hinsicht interessant, stellt er doch ähnlich dem VIC mehrere Interrupt-Quellen zur Verfügung. Für Nicht-Programmierer fällt in dieser Folge der Serie ein praktisches Funktionstasten-Tool ab, womit sie Befehle auf Tastendruck abrufen können.

Unterlauf löst er einen Impuls aus, der je nach Einstellung der Steuerregister ausgewertet werden kann, zum Beispiel, um einen Interrupt zu erzeugen.

Neben diesen Timern beinhaltet der CIA auch noch eine Echtzeituhr. Diese Uhr verwendet als Taktgeber die Netzfrequenz und hat somit eine sehr viel höhere Ganggenauigkeit als die von BASIC aus über TI$ verfügbare Uhr. Zusätzlich kann diese CIA-Uhr mit einer Alarmzeit vorprogrammiert werden. Dieser „Alarm" erzeugt ebenfalls einen Interrupt. In der ersten Folge unserer Interrupt-Tips stellten wir Ihnen bereits eine mögliche Anwendung dieser Uhr vor.

Tempo gestoppt

Die beiden CIA-Bausteine im C64 werden vom Betriebssystem des C64 für verschiedene Aufgaben benutzt:

CIA #1 löst durch den Timer A jede 1/60 Sekunde den System-Interrupt aus, der so wichtige Dinge wie Tastaturabfrage und Cursor-Blinken steuert. Timer B wird bei Operationen am seriellen Bus und im Zusammenhang mit der Datassette verwendet. Über Port A und Port B wird die Abfrage der Tastaturmatrix und der Joysticks ermöglicht.

Die Timer A und B vom CIA #2 werden bei der Verwendung der RS232-Schnittstelle benötigt. Port A steuert darüber hinaus die Adreßauswahl der Videobank, Port B steht als Userport dem Benutzer zur Verfügung.

Für eigene Interrupt-Programme können Sie, vorausgesetzt Sie benutzen die RS232-Schnittstelle nicht, die Timer des CIA #2 nutzen. Der CIA #2 liefert kein IRQ-Signal, sondern löst statt dessen einen NMI aus, der im Gegensatz zu einem IRQ nicht unterbunden werden kann. Ist dies aber erwünscht und greifen Sie während Ihrer Interrupt-Routine nicht auf den seriellen Bus zu, können Sie auch den Timer B der CIA #1 benutzen. Der CIA #1 liefert nur einfache IRQ-Impulse an die CPU.

In dieser Serie war schon mehrmals die Rede davon, daß die CPU alle 1/60 Sekunden von der CIA #1 in ihrer Arbeit unterbrochen wird, um ein Interrupt-Programm abzuarbeiten. Dieses Zeitintervall wird durch den Timer A der CIA #1 über die Adressen 56324 und 56325 ($DC04, $DC05) gesteuert. Hier wird ein Wertepaar im Low-High-Byte-Format abgelegt und dieses mit der Systemtaktfrequenz von circa 1 MHz heruntergezählt. Bei Erreichen eines Unterlau-

Register 13 — Bit für Bit

Bit-Position	Funktion
7	Schreiben/Lesen Interrupt ein/aus
6	benutzt
5	unbenutzt
4	Interrupt -„Flag"
3	Interrupt - „SP"
2	Alarm von Uhr
1	Unterlauf Timer A
0	Unterlauf Timer B

Registrierte Adressen

fes (Wert < 0) wird ein Interrupt an die CPU gesendet. Gewöhnlich enthält der Timer A den Wert 16 421. Der Wert 16 421 errechnet sich aus folgender Formel, wobei **t** die gewünschte Zeit gemessen in Sekunden ist:

w = 98 5248 * t

Für t=1/60 ergibt sich für **w** gerundet der oben genannte Wert 16 421.

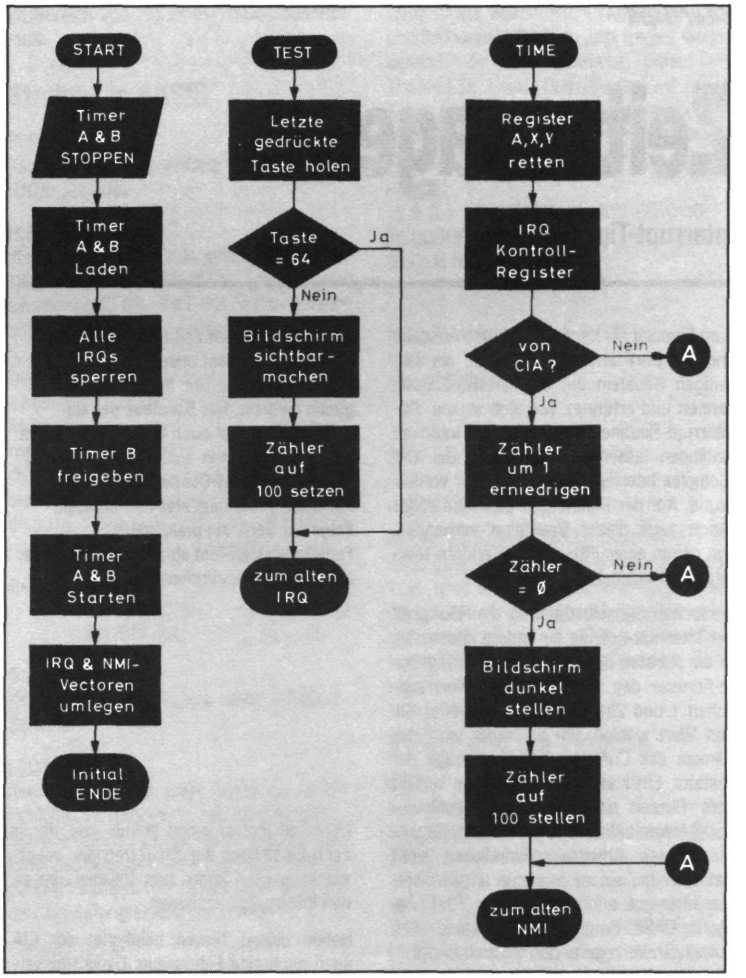
. . . und freigegeben

Um die Timer sinnvoll nutzen zu können, muß zuerst festgelegt werden, welche Operationen die Timer ausführen sollen. Genau wie beim VIC gibt es deshalb auch bei den CIA-Bausteinen ein Register, in welchem festgelegt wird, wer wie und wann einen Interrupt auslösen darf. Dieses geschieht im Register 13 bei beiden CIA-Bausteinen. Der interne Aufbau dieses Registers und der Adreßlage finden Sie im Bild „registrierte Adressen".

Register 13 besitzt eine Doppelfunktion. Durch Schreiben in das Register legen Sie fest, wer einen Interrupt auslösen darf. Das Lesen des gleichen Registers zeigt nach einem Interrupt an, wer den Interrupt ausgelöst hat. Diese Funktionen werden beim VIC über zwei getrennte Register geregelt.

Beim Schreiben in das Register 13 eines CIA müssen noch einmal zwei Fälle unterschieden werden, die vom Status des siebten Bits abhängen. Wird das siebte Bit geschrieben (Write-Modus), können damit alle Interrupts gesperrt oder wieder freigegeben werden.

Schreiben Sie eine Null in das siebte Bit, werden alle anderen Bits im Register, die auf Eins standen, auf Null gesetzt. Dies ist gleichbedeutend mit dem Sperren der betreffenden Interrupt-Quellen. Setzen Sie jetzt einzelne Quellen wieder auf Eins und schreiben anschließend eine Eins ins siebte Bit, werden nur diese Interrupts wieder freigegeben. Mit einem POKE 56333,127 werden alle Interrupts unterbunden. Der C64 stellt sich tot, weil keinerlei Peripherie mehr abgefragt werden kann, so auch nicht die Tastatur. POKE 56333,129 schaltet nur den Timer A als Interrupt-Quelle. Die RESTORE-Taste ist ohne Wirkung. Erst POKE 56333,130 läßt RUN/STOP-RESTORE wieder zu.



Wege ins Dunkle

Lesen Sie den Inhalt des siebten Bits im Register 13 aus, informiert Sie eine Eins im siebten Bit (Werte größer 127), daß ein Interrupt stattfand. Die übrigen Bits lassen erkennen, wer als Verursacher in Frage kam. Ein Lesen dieses Registers löscht jedoch dessen Eintragung. Brauchen Sie den Inhalt für weitere Anwendungen, so sollten Sie ihn an geeigneter Stelle zwischenspeichern.

Zeitvergleich

Nun zurück zum Setzen der Timer-Werte und deren Funktionen: Ein Timer ist nichts anderes als ein Zähler, der im Rhythmus des Systemtaktes heruntergezählt wird.

Der Timer A zählt bei jedem Taktimpuls ein Zähler-Register herunter; unterschreitet er den Wert null, löst er einen Interrupt aus,

setzt seinen ursprünglichen Startwert in seinen Zähler und startet von neuem. Dieses Zurückschreiben des Startwertes bezeichnet man als „Continues Mode" (fortlaufender Zustand), im Gegensatz dazu kann auch ein „One Shot Mode" („ein Schuß"-Zustand) eingestellt werden, bei dem nach einmaligem Herunterzählen kein weiterer Interrupt mehr ausgelöst wird.

Der gewünschte Modus wird in den Timer-Kontroll-Registern 14 und 15 der CIAs festgelegt. Der größtmögliche Timer-Wert von $FFFF reicht für eine Zeitdauer von cirka 1/15 Sekunde. Wollen Sie längere Zeitabstände mit Ihren Timern erzielen, müssen Sie Timer A und Timer B von CIA #1 oder CIA #2 zu einem 32Bit-Zähler zusammenschalten. Auch dieses wird über das Register 15 festgelegt. Dann lautet der größtmögliche Timer-Wert 2^{32} = 4 294 967 296. Dieser Wert läßt den Timer 1 Stunde, 12 Minuten und 40 Sekunden zählen. Das sollte für die meisten Anwendungen wohl reichen.

Im Gegensatz zu dem vorherigen Beispiel wird im letzten ein IRQ erst dann erzeugt, wenn der Timer B einen Unterlauf (Wert < 0) hat. Timer B wird nicht wie der Timer A im Systemtakt heruntergezählt, sondern zählt nur bei jedem Unterlauf des Timers A, wenn dieser also Null erreicht hat. Hat der Timer B eine Unterbrechungsanforderung an die CPU gesandt, laden beide Register wieder ihre Ausgangswerte.

Zeitbild

Nach so viel Theorie und Register-Schieberei wird es langsam Zeit, Ihnen anhand eines Programmbeispieles zu zeigen, wie ein CIA als Interrupt-Quelle im Einsatz zu Geltung kommt: Der Rechner soll die Tastatur jede Sekunde abfragen. Wurde länger als eine Minute keine Taste betätigt, soll der Bildschirm dunkel geschaltet werden. Das schützt den Monitor vor Einbrennschäden. Sobald Sie jedoch irgendeine beliebige Taste berühren, wird es sofort wieder hell. Sie kennen es vielleicht schon unter solch anheimelnden Namen wie „Bildschirm-Schoner", schon einmal als Beispiel zum IRAS (interaktiver Reassembler) in INPUT 64, Ausgabe 4/87 erwähnt.

In folgenden Schritten läßt sich diese Idee realisieren (wobei es natürlich auch andere Möglichkeiten geben mag, die zu anderen Programm-Lösungen führen): Die Timer A und B von CIA #2 sollen als Zähler und Interrupt-Quelle benutzt werden. Innerhalb der normalen Interrupt-Routine wird jedesmal, wenn eine Taste benutzt wurde, der Bildschirm eingeschaltet und der Intervall-Zähler wieder auf den Anfangswert gesetzt. In der CIA#2-Interrupt-Routine wird jede Sekunde der Intervall-Zähler erniedrigt. Ist der Intervall-Zähler abgelaufen, so wird der Bildschirm abgeschaltet, es wird dunkel.

Der unter „der Weg ins Dunkle" dargestellte Programmablaufplan zeigt Ihnen grafisch, wie so etwas aussehen kann.

Das abgedruckte Listing läßt sich mit fast jedem Assembler verarbeiten. Angepaßt ist es in der abgedruckten Form an den INPUT-ASS aus INPUT 64, Ausgabe 6/86.

Taste ohne Maske

Wie Sie bereits erfuhren, kann der CIA #2 einen NMI erzeugen. Neben dieser Quelle, besitzt der C 64 auch noch die Möglichkeit, einen NMI per RESTORE-Taste auszulösen. Diese ist direkt mit der NMI-Leitung der CPU verbunden und hat neben dem CIA #2 absolute Unterbrechungspriorität vor jedem anderen IRQ. Ein Druck auf diese Taste unterbricht alles und jeden Arbeitsvorgang unseres C64, egal ob sich der Rechner im Direkt- oder im Programmmodus befindet. Doch bei einigen C 64-Rechnern ist die mechanische und elektronische Auslegung dieser Taste nicht gerade überzeugend, wundern Sie sich also nicht, wenn Sie erst auf Ihre RESTORE-Taste länger Druck ausüben müssen.

Doch was passiert, wenn ein NMI ausgelöst wird? Besitzer eines ROM-Listings und Kenner desr Assembler-Dialekts können diese Odyssee quer durch den Speicher ja einmal Schritt für Schritt verfolgen, indem Sie den Bereich mit einem Monitor-Programm wie beispielsweise dem MLMplus aus INPUT 64, Ausgabe 12/87, benutzen.

Über den Vektor $FFFA/$FFFB wird die Adresse $FE43 (65091) angesprungen. Hier werden zuerst mit dem Assembler-Befehl SEI alle weiteren IRQ-Anforderungen gesperrt. Über den im RAM liegenden Vektor bei $0318 und $0319 (792 und 793) verzweigt die Routine normalerweise nach $FE47 (65095). Hier werden zuerst die Arbeitsregister gerettet und dann getestet, ob der NMI vom CIA #2 kam. In diesem Fall verzweigt der Prozessor in die RS232-Schnittstellenbehandlung. Ansonsten wird das Vorhandensein eines Moduls überprüft und dieses dann gestartet. Ist all dieses nicht der Fall, wird bei gedrückter RUN/

Registerplan der CIA

Register Nr.	Adressen CIA #1	(Hex/Dez) CIA #2	Funktion
04	$DC04/56324	$DD04/56580	Timer A Low-Byte
05	$DC05/56325	$DD05/56581	Timer A High-Byte
06	$DC06/56326	$DD06/56582	Timer B Low-Byte
07	$DC07/56327	$DD07/56583	Timer B High-Byte
08	$DC08/56328	$DD08/56584	1/10-Sekunden-Register
09	$DC09/56329	$DD09/56585	Sekunden-Register
10	$DC0A/56330	$DD0A/56586	Minuten-Register
11	$DC0B/56331	$DD0B/56587	Stunden-Register
12	$DC0C/56332	$DD0C/56588	serielles 8-Bit-Register
13	$DC0D/56333	$DD0D/56589	Interrupt-Kontroll-Register
14	$DC0E/56334	$DD0E/56590	Kontroll-Register A
15	$DC0F/56335	$DD0F/56591	Kontroll-Register B

STOP-Taste ein Warmstart durchgeführt. Wurde die RESTORE-Taste alleine betätigt, wird der Zustand vor Auslösung des NMI rekonstruiert und mit dem Programm fortgefahren; es passiert scheinbar gar nichts.

Echte Auswirkungen hat nur die Kombination mit der RUN/STOP-Taste. Die Vektoren-Verknüpfung wurde in der letzen Folge in einer Grafik gezeigt (Seite 12 in Ausgabe 12/87).

Wie aus dem eben Gesagten hervorgeht, passiert bei jedem Druck auf die RESTORE-Taste eigentlich eine ganze Menge, auch wenn dies dem Benutzer verborgen bleiben mag. Da einer der gerade beschriebenen Vektoren im RAM liegt, können wir diesen für eigene Zwecke verändern. Eine Anwendung einer eigenen Routine haben Sie schon kennengelernt. Ein Verbiegen des NMI-Vektors um 122 Byte auf $FEC1 (65217) läßt unseren NMI arbeitslos werden. Mit dieser „Vektoren-Umleitung" sperren Sie auf einfache Art die RUN/STOP-RESTORE-Tastenkombination und deren Auswirkung wie Bildschirm-Reset, Programm-Unterbrechung und ähnliches.

Von BASIC aus erreichen Sie dies mit **POKE 792,193**. Den Ursprungszustand können Sie mit **POKE 792,71** wiederherstellen.

f1 ruft LIST!

Das abspeicherbare Programm stellt Ihnen ein einfaches Werkzeug zur Verfügung. Es wird wie ein normales BASIC-Programm geladen und gestartet. Nach Eingabe von „RUN" verschiebt sich das Programm an das Ende des BASIC-Speichers, setzt die Speicherobergrenze auf 39999 und aktiviert den Funktionstasten-Generator.

Eine Übersicht über Belegung der einzelnen Funktionstasten erhalten Sie durch einmaligen Druck auf die RESTORE-Taste.

Die Funktionstasten können mit Hilfe folgender Syntax belegt werden:

*n,"STRING"
* = Befehlskennzeichen

n = Funktionstastennummer (1 – 8)
STRING = Ausdruck mit maximal 9 Zeichen

Ein '↑' als String-Abschluß gibt den Text mit anschließendem RETURN aus. Der String wird also gleich als Befehl ausgeführt.

Beispiel:

*1,"LIST↑"

Bei Druck auf 'f1' wird LIST ausgegeben und sofort ausgeführt.

Sollten Sie aus irgendeinem Grund die eventuell vorhandene RESET-Taste Ihres Rechners benutzt haben, können Sie den Funktionstastengenerator durch ein SYS 40000 zum Leben erwecken.

Tip:

Mit der Folge
load+CHR$(34)+$+chr$(34)+,8↑

erhalten Sie ein Directory, allerdings mit Programmverlust! CHR$(34) ersetzt das Zeichen"", das sich sonst nicht eingeben ließe. Thomas Stahmer/Frank Börnke/rh

Kontroll-Register 14

Bit-Positon	Funktion bei Bit=0	Bit=1
7	Hz 50 Hz	
	Taktfrequenz der Uhr	
	serielles Ein-Ausgabe-Register	
5	Systemtakt	externes Taktsignal
	für Timer	
4		Latch-Register laden
3	Continuous-Mode	One-Shot-Modus
2	PB6 Impulsschaltung	PB6 als Umschalter (Toggle)
1	PB6 Normal seriell[1]	Timer A an PB6
0	Stoppt Timer A	Startet Timer A

Kontroll-Register 15

Bit-Positon	Funktion bei Bit=0	Bit=1
7	Normal Uhr TOD	Alarm
6-5	00: Systemtakt	01: externer Takt
	10: Unterlauf Timer A	11: Unterlauf von A bei extern = 1
	für Timer B	
4		Latch-Register laden
3	Continuous-Mode	One-Shot-Modus
2	PB7 Impulsschaltung	PB7 als Umschalter (Toggle)
1	PB7 Normal seriell[1]	Timer A an PB7
0	Stoppt Timer A	Startet Timer A

[1]PB6/7 bezeichnet Port B-Leitungen sechs und sieben, die normalerweise der externen Ein-/Ausgabe dienen, aber über die Register auch als Signalausgänge für die Timer benutzt werden können.

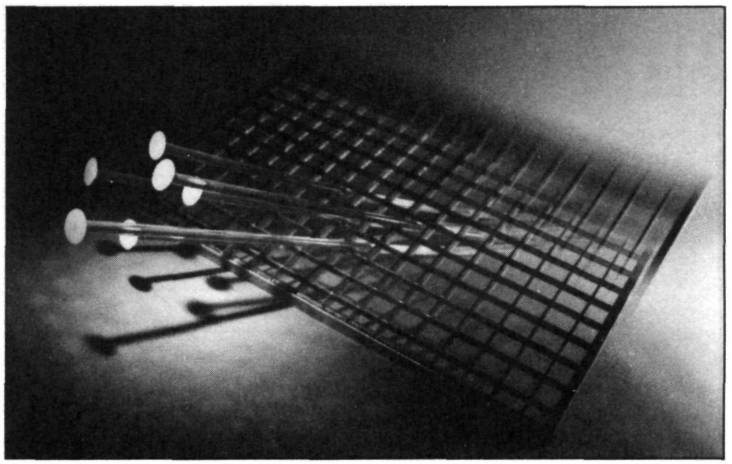
Register 14 und 15 – voll belegt

Literaturhinweise:

C64 INTERN – DATA BECKER
Düsseldorf 1983
C64 Computer – Handbuch Raeto West
München 1985

Drehen, wenden – paßt!

Strategiespiel: StraMax



In diesem Spiel – für einen bis vier kleine oder große Denker – kommt es darauf an, Muster zu erkennen und nach bestimmten Regeln zu verbinden. Da ein Part auch von Ihrem C64 übernommen werden kann, können Sie Ihrem Computer endlich mal beweisen, daß menschliche Vorstellungskraft, vor allem wenn außerdem noch strategisches Geschick gefragt ist, brutaler Rechengewalt überlegen ist.

Die angesprochenen Regeln sind sehr einfach aufgelistet. Gewonnen hat der Spieler, der als erster die vor Spielbeginn festgelegte Punktzahl erreicht. Punkte werden immer (automatisch) vergeben, wenn aus der Auswahl von sechs Spielsteinen ein Stein auf die große Spielfläche gelegt wird. Der ausgewählte Spielstein kann überall dort abgelegt werden, wo sein Bild ohne Überdeckung Platz findet.

Doppelt und dreifach

Erst wenn nach mehrfachem Ablegen eine Figur von drei mal drei Punkten entstanden ist, wird dieses Spielfeld vom Programm wieder freigegeben. In der Zwischenzeit kann munter gepunktet werden. Sie erhalten nämlich sowohl für den abgelegten Spielstein als auch für eine eventuell dort vorhandene Spielfigur Ihre Pluspunkte. Ihr Computer wird allerdings nur dann mit dabei sein, wenn Sie vor Spielbeginn einen Spielpartner als „C64" bezeichnet haben.

Die Steuerung des Spiels ist einfach über den Joystickport 2 oder über die normalen Cursor-Tasten (mit RETURN als Bestätigung) möglich. Hinzu kommen interessante Zusatzfunktionen, die über die Funktionstasten erreichbar sind. Sie können bei den Spiedsteinen zwar nicht die Anordnung der Punkte zueinander ändern, aber durch Drehen oder Spiegeln der Figuren selbst „neue" Spielsteine kreieren.

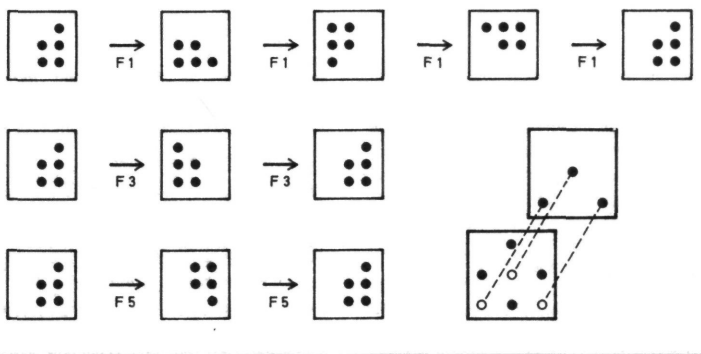
Auch wenn es neun (mehr oder weniger leere) Ablageflächen für einen Ihrer sechs Spielsteine gibt, kann der Fall eintreten, daß Sie trotz Drehung oder Spiegelung dieser Spielsteine keinen ablegen können.

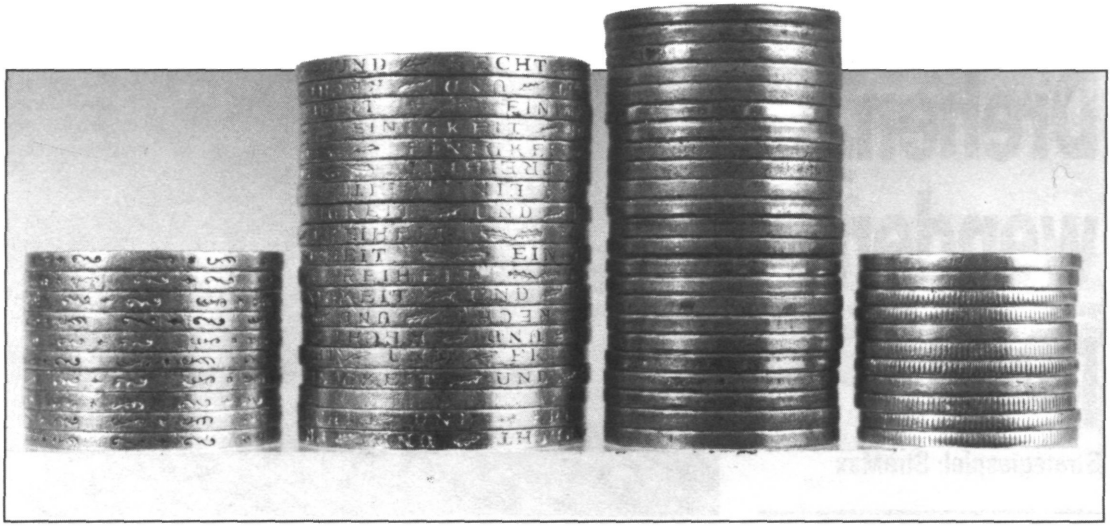
Wenn nichts mehr geht . . .

Jetzt, und nur jetzt, ist es erlaubt, die Funktion „tauschen" aufzurufen. Sie erreichen diese Funktion, wenn Sie einen beliebigen Spielstein nach unten führen und unmittelbar danach das Spielfeld rechts verlassen. Die angewählte Funktion muß noch mit RETURN bestätigt werden, und sofern der Funktionsaufruf wirklich erlaubt war, bekommen Sie andere Spielsteine. Anderenfalls erscheint ein entsprechender Hinweis in der oberen Bildschirmzeile.

T. Strahmer/WM

Spiegelfechtereien





Finanzielle Voraussagen

Lohn- und Einkommensteuer '87

Ohne Computerunterstützung ist die Antwort auf die Frage „Lohnt sich für mich ein Lohnsteuerjahresausgleich?" nur mühselig durch intensive Nutzung von Papier und Bleistift zu beantworten. Unser alljährlich aktualisiertes Programm zur Lohn- und Einkommensteuerberechnung macht eine mühelose Vorausschau möglich.

Das von Michael Hanke geschriebene Programm orientiert sich an den offiziellen Formularen der Finanzämter, dem Antrag auf Lohnsteuerjahresausgleich/Einkommensteuererklärung. Diese Bögen sollte man auch zur Hand haben, wenn man seine persönlichen Daten zur Berechnung eingibt. Denn die meisten Fragen werden nicht im Klartext gestellt, sondern unter Bezug auf die Zeilennummern der grau-grünen Formulare.

Im Eingangsmenü von „Lohnsteuer '87" steht zur Wahl, sich eine erste Übersicht durch Einsicht in die Einkommensteuertabelle zu verschaffen oder seine eigene Jahressteuer berechnen zu lassen. In letzterem Programmteil müssen zunächst die „Allgemeinen Angaben" entsprechend dem Formular gemacht werden, dann gelangt man in ein Inhaltsverzeichnis.

Menü à la Formular

Aus diesem Inhaltsverzeichnis sind die einzelnen Abschnitte der Steuererklärung abrufbar, wie „Anlage N", „Sonderausgaben" und so weiter, vor allem aber der Berechnungsteil, der den wahren Stand der Steuerschuld ans Licht bringt.

Wenn Sie einzelne Eingaben ändern wollen, müssen Sie nicht Ihre kompletten Steuerdaten neu eingeben. Durch Zurückspringen in das Inhaltsverzeichnis und neue Anwahl eines Abschnitts kann jeder Eintrag editiert und anschließend die Berechnung neu gestartet werden. Natürlich existiert auch die Möglichkeit, alle Daten komplett zu löschen; dann hat man wieder die Wahl zwischen Ausgabe der Einkommensteuer- und Splitting-Tabellen und Lohnsteuerberechnung.

Wer den Pfennig . . .

Das Programm gibt sich nicht mit Pfennigfuchserei ab, also immer volle DM-Beträge eingeben! Die eingetippten Daten müssen mit RETURN übergeben werden, RETURN ohne weitere Angaben steht für „0" oder „keine Angaben". Alle Bildschirmseiten, sowohl Eingaben als auch Berechnungsergebnisse, kann man durch Betätigen der Tastenkombination CTRL-B auf dem Drukker ausgeben lassen.

Ein besonderes Bonbon ist für die Besitzer eines C128/C128D eingebaut, die das Programm wie gewohnt im 64er Modus benutzen können: ein kleines Programm aktiviert die Zehnertastatur, die sonst nur in der 128er Betriebsart verfügbar ist[1]. JS

[1]Dieser Trick wurde erstmals im 64er-Sonderheft 1/86, Verlag Markt & Technik, Haar bei München, veröffentlicht.

Abzüge bei Einnahmen

Steuern sparen durch Computerkauf

Für nicht selbständige Arbeitnehmer gibt es grundsätzlich zwei Möglichkeiten, einen Computer von der Steuer abzusetzen: im Rahmen der Sonderausgaben oder bei den Werbungskosten.

Sonderausgabe Weiterbildung

Hat sich ein Arbeitnehmer im Rahmen einer Weiterbildung oder Berufsausbildung in einem nicht ausgeübten Beruf einen Computer angeschafft, sind die Kosten als Sonderausgaben absetzbar (§ 10 (1) 7 EStG), aber nur maximal 900 DM.

Beispiel: Herr K. ist in seinem erlernten Beruf als Industriekaufmann angestellt. Er möchte lieber als Programmierer arbeiten, nach Feierabend betreibt er eine Weiterbildung im DV-Bereich. Um das Gelernte zu vertiefen, schafft er sich einen geeigneten Computer an.

In diesem Fall sind die Aufwendungen für den Computer – und selbstverständlich auch noch weitere Ausbildungskosten – absetzbar, aber nur bis zur Höhe von 900 DM.

Privat oder beruflich?

Die interessantere Möglichkeit des Abzugs ist die „Verbuchung" unter der Rubrik Werbungskosten (§ 9 EStG), die jeder angestellte Arbeitnehmer geltend machen kann. Der Gesetzgeber hat hierfür schon einen Pauschbetrag in Höhe von 565 DM vorgesehen (Stand: 1987); sollten die Werbungskosten (dazu zählen auch Fahrten zwischen Wohnung und Arbeitsstätte, Gewerkschaftsbeiträge und so weiter) diesen Betrag übersteigen, vermindern alle Aufwendungen über 564 DM die Steuerschuld.

Das wäre nicht schlecht, durch den Lohnsteuerjahresausgleich das Finanzamt an den Kosten für die Anschaffung des neuen Geräts zu beteiligen.
Unter welchen Bedingungen dies möglich ist, erläutert Finanzinspektor Günter Müller.

Dazu kann in bestimmten Fällen auch der privat angeschaffte Computer gehören – wenn er als Arbeitsmittel von der Finanzbehörde anerkannt wird. Die Gesetze lassen hier wenig Spielraum: So sind beispielsweise Kosten, die nicht eindeutig dem privaten oder beruflichen Bereich zugeordnet werden können, grundsätzlich nicht absetzfähig (§12 Nr.1 EStG).

Glaubwürdigkeitsfragen

Zur Zeit aber lockert sich der Standpunkt der Finanzämter wieder: Geräte wie der C64 werden bei den abzugsfähigen Werbungskosten immer öfter anerkannt. Voraussetzung ist nicht nur, daß der Steuerzahler den Computer aus beruflicher Veranlassung benutzt. Er muß dies auch glaubhaft machen können.

Beispiel 1: Ein Außendienstmitarbeiter einer Versicherung muß seine Berichte abends zu Hause, eventuell in seinem Arbeitszimmer, schreiben. Anstelle einer herkömmlichen Schreibmaschine kauft er einen Computer mit allem, was zur Textverarbeitung dazugehört

Dieser Fall ist eindeutig. Der Kauf ist beruflich veranlaßt, und da dem Außendienstler durch seinen speziellen Job nur wenig Zeit für eine private Nutzung des Computers bleibt, wird er wenig Schwierigkeiten haben, die Anlage abzusetzen. Er kann die berufliche Nutzung glaubhaft machen.

Beispiel 2: Ein Industriekaufmann oder ein technischer Angestellter kann mit einem privat angeschafften Computer berufliche Berechnungen durchführen.

Obwohl hier eine berufliche Nutzung vorliegt, wird es sehr schwierig sein, diesen Computer bei den Werbungskosten geltend zu machen. Eine schriftliche Bestätigung des Arbeitgebers wäre von Vorteil, um dem Finanzamt die überwiegend berufliche Nutzung glaubhaft zu machen.

Die 800-DM-Grenze

Werbungskosten können im Prinzip in beliebiger Höhe geltend gemacht werden, denn das Finanzamt kann keinem vorschreiben, wie teuer er seine Arbeitsmittel zu erstehen hat. Aber: Sollte die Anschaffung eines Computers mehr als 800 DM betragen, so gilt er nicht mehr als sogenanntes „geringwertiges Wirtschaftsgut", das sofort im Anschaffungsjahr voll abgesetzt werden kann. Die Anschaffungskosten sind in diesen Fällen auf die „gewöhnliche Nutzungsdauer" zu verteilen, auf circa drei bis fünf Jahre.

Beispiel 1: Der Computer kostete 900 DM; 100 DM zuviel, um noch als geringwertiges Wirtschaftsgut eingestuft zu werden. Die Nutzungsdauer wird meist mit drei Jahren angesetzt, somit sind im Anschaffungsjahr und in den zwei darauf folgenden Jahren jeweils 300 DM als Werbungskosten absetzbar.

Beispiel 2: Der Rechner selbst kostete 600 DM, dazu wurde ein Drucker für 300 DM angeschafft. Computer und Drukker bilden eine wirtschaftliche Einheit, die generell zusammengefaßt wird, so daß es wieder zu der Verteilung wie in Beispiel 1 kommt.

Fazit: Jeder, der seinen Computer beruflich nutzt, sollte versuchen, das Gerät von der Steuer abzusetzen.

Vorsicht aber bei computererstellten Schreiben an das Finanzamt – denn dies ist immer ein Beweis für die private Nutzung!

Günter Müller/JS

Über Sprachen unter anderen

BASICon: Konverter für BASIC-Versionen

Umformung von Programmen, die für den C16/116 oder den Plus 4 geschrieben wurden, macht meist wegen des größeren BASIC-Sprachumfangs des BASIC V 3.5 Probleme. Gerade die interessanten Grafik-, Sound- und Struktur-Befehle sind der Version 2.0 völlig fremd. Es liegt daher nahe, in einer „höheren" BASIC-Sprache geschriebene Programme auf die Sprachsyntax einer BASIC-Erweiterung wie zum Beispiel INPUT-BASIC (aus Ausgabe 2/86) anzupassen.

C16, C116 und plus/4 unterscheiden sich genaugenommen nur in Äußerlichkeiten, das elektronische Innenleben ist identisch. Nur beim plus/4 kommen lediglich die in EPOROMs installierten vier Programme (Textverarbeitung, Dateiverwaltung, Tabellenkalkulation und Monitor) und ein größerer Speicher hinzu (64 KByte gegenüber 16 KByte). Mit etwas Hardware-Aufwand und Lötarbeiten können der C16 und der C116 für etwa 50,- DM auf 60 671 freie Bytes (für BASIC-Programmierer) hochgerüstet werden. Wer den Mehraufwand nicht scheut, erreicht sogar fast 128 KByte. Doch sind Speicheraufteilung und eben die BASIC-Version in wesentlichen Punkten anders als beim C64.

Wenn Rechner eines Herstellers die gleiche Sprache sprechen, sollten Programme auch zwischen ihnen austauschbar sein. Doch leider gibt es einige Übersetzungsprobleme bei den verschiedenen BASIC-Dialekten. Warum also nicht die „Intelligenz" der Rechner ausnutzen, statt in mühsamer Kleinarbeit selbst Listings umzuschreiben?

Adressvergleich wichtiger Speicherstellen:

Bis zur Adresse 75 sind alle für BASIC interessanten Speicherstellen in ihrer Funktion identisch.

Weitere Adressen für POKE, PEEK, WAIT:

C16	C64	Bemerkung
163-165	160-162	interne Uhr
194		REVERSE-Flag
198	203	letzte gedrückte Taste
200-201	209-210	Anfangsadresse der akt. Bildschirmzeile
202/205	211/214	Cursor-Spalte/-Zeile
203	212	Quotemode
239	198	Anzahl d. Zeichen im Tastaturpuffer
1319-1328	631-640	Tastaturpuffer
1343	649	Größe des Tastaturpuffers
819-1010	828-1019	Kassettenpuffer
1344	650	Tastenwiederholung
1347	653	Flag für SHIFT/CTRL und C=
2048-3071	55296-56295	Farb-RAM für Textmodus
3072-4095	1024-1023	Bildspeicher für Textmodus
4096-	2048-	BASIC-Programm-Speicher
7168-8191	1024-1023	Farb-RAM für HIRES
8192-16383	8192-16383	Bitmap HIRES
53248	53248	Startadr. d. Zeichengener.
65286 OR 64	POKE 53265,91	Extended Color Mode ein
65286 AND 191	POKE 53265,6	Extended Color Mode aus
65286 AND 239	POKE 53265,11	Bildschirm ausschalten
65286 OR 16	POKE 53265,27	Bildschirm einschalten
65409-65523	65409-65523	KERNAL-Sprungtabelle

Dialekt versteckt

In der zeitlichen Folge der Weiterentwicklung blieben die Versionen zueinander weitgehend aufwärtskompatibel; das heißt, die Version 2.0 des C64-BASIC ist vollständig in der weiterentwickelten V3.5-Version des C16/116-BASIC enthalten. Die Version 3.5 wiederum in der Version 7.0 des C128, bis auf geringfügige Sytax-Abweichungen.

Reine BASIC-Programme, die ohne SYS, PEEK, POKE, USR oder WAIT auskommen, also keine Befehle enthalten, die auf bestimmte Speicherstellen zugreifen, können Sie direkt vom C64 auf den C16 übernehmen. Ein einfacher LOAD-Befehl reicht, da diese Rechner allesamt ohne Schwierigkeiten auf die 1541 zugreifen können. Einzig bei längeren Programmen kann es wegen des größeren Speicherplatzbedarfs zu Problemen kommen.

Umgekehrt gilt das gleiche für Programme, die auf einer höheren BASIC-Version basieren, solange sie nur den eingeschränken Befehlsvorrat der niedrigeren Version zum Beispiel des C64-BASIC nutzen. Programme von anderen Rechnern sollten Sie möglichst mit LOAD "name",8,0 laden. Durch die Sekundäradresse Null wird das Programm auf jeden Fall an den BASIC-Anfang geladen. Manche Betriebssystem-Erweiterungen, wie zum Beispiel Floppy-Beschleuniger, laden Programme absolut, also an die Speicherstelle, an der sie vor dem Abspeichern lagen.

Verstehen und sehen

Um zu nachvollziehen zu können, wie die Rechner die BASIC-Befehle „verstehen", ist es wichtig zu wissen, daß die Befehle in Kennziffern, in sogenannten Token, verschlüsselt sind. Der Token-Satz des C16 und des C64 sind bis zum Wert 203 gleich. Höhere Codes im Programmspeicher werden beim C64 vom normalen Betriebssystem nach einem RUN mit SYNTAX ERROR quittiert. Die Token-Tabelle des C16 reicht aber bis 253. Bis hierhin ist sogar der C128 mit dem C16 identisch.

Wie kann man nun ein BASIC-Programm, das Token über 203 verwendet, auf dem C64 laufen lassen? Hat man das Programm als Ausdruck vorliegen, kann man beim Abtippen diejenigen Befehle anpassen, die der C64 nicht kennt. Sie verwenden entweder eine BASIC-Erweiterung oder setzen SYS-Aufrufe für die entsprechenden Tools oder umschreiben den fehlenden Befehl mit aufwendigen Unterprogrammen oder POKE-Sequenzen.

Übersetzer in Sicht

Liegt das Programm jedoch auf Datenträger vor, können Sie es zwar wie oben beschrieben problemlos einladen, aber die Token ab 204 werden fehlerhaft gelistet, und Sie benötigten eine Vergleichstabelle, um sie zu identifizieren. Einfacher ist ein Programm, welches die entsprechenden Codes in ASCII-Zeichen „ausschreibt". Liegt das Programm dann in lesbarer Form im Speicher vor, so daß COLOR 0,2,3 nicht in LIST 0,2,3 übersetzt wird, kann man darangehen, die Token zu ersetzen, welche der C64 nicht kennt.

BASICon hilft Ihnen, indem er alle Token über 203 anhand einer eigenen Tabelle in Befehlstext (ASCII-Code) umwandelt. Damit bei einem versuchsweisen „RUN" nichts schiefgeht, steht vor jeder Zeile, die einen solchen Befehl enthält, REM **. Sie können mit einem Blick erkennen, wo Änderungen notwendig sind. Um die „Handarbeit" noch etwas mehr zu erleichtern, stellt Ihnen INPUT 64 noch ein zweites Programm zur Verfügung: den AB-Generator (ASCII-Befehlstext-Generator), mit dem Sie die Befehlstexte in der Tabelle des Wandlers verändern können. Dieses Programm ist einfachstes BASIC. Sie müssen nur darauf achten, daß keine DATA-Zeilen verlorengehen und jeder Text-String mit '13' abgeschlossen wird. Der AB-Generator ist bereits mit einer Anpassung für INPUT-BASIC ausgerüstet.

Sie speichern sich die beiden Programme mit CTRL-S auf eigenen Datenträger. Nach-

BASIC-Vergleiche

C16	C64	Bemerkung
COLOR 0,F,L[1]	POKE 53280,F	Hintergrundfarbe setzen
COLOR 4,F,L[1]	POKE 53281,F	Rahmenfarbe setzen
COLOR 1,F,L[1]	POKE 646,F	Zeichenfarbe setzen

[1]'L' bedeutet LUMINANZ und kann beim C64 nicht eingesetzt werden. Sind Zeichenfarbe und Hintergrundfarbe nur durch 'L' unterscheidbar, muß man eine andere Farbe verwenden.

GETKEY A$	POKE198,0:WAIT198,1:GET A$	wartet auf Tastendruck
SCNCLR	PRINT CHR$(147)	löscht Textbildschirm
10 IF A=0 THEN 40: ELSE 50	10 IF A=0 THEN 40 20 IF A<>0 THEN 50	Die ELSE-Anweisung muß durch eine zweite IF- Abfrage ersetzt werden.

DO/LOOP/WHILE/UNTIL/EXIT-Schleifen:

10 DO: X=X+1 20 IF X=10 THEN EXIT 30 LOOP	10 FOR I=0 TO 1 STEP 0 20 X=X+1:IF X=10 THEN I=1 30 NEXT	Die Schleife zwischen DO und LOOP wird solange durchlaufen, bis EXIT erfüllt ist.
10 DO WHILE X<10 20 X=X+1 30 LOOP	10 IF X =10 THEN 40 20 X=X+1 30 GOTO 10	Die DO WHILE/LOOP wird durchlaufen, solange die WHILE-Bedingung wahr ist.
10 DO 20 X=X+1 30 LOOP UNTIL X=10	10 X=0 20 X=X+1 30 IF X<10 THEN GOTO 20	Die DO/LOOP UNTIL Schleife wird durchlaufen, bis die UNTIL-Bedingung wahr wird.

Für weitere Anpassungen benötigt man beim C64 entsprechende Programme oder eine BASIC-Erweiterung. Viele der C16 Befehle kann man mit Hilfe des INPUT-BASIC aus der Ausgabe 1/86 ersetzen, (die genaue Beschreibung siehe dort).

Wichtige SYS-Adressen

SYS 32768	SYS 64738	Kaltstart
SYS 34578	SYS 65126	Warmstart
SYS 61192	SYS 65511	Schließen aller Kanäle
SYS 62219	SYS 65409	Videoreset
POKE205,z:POKE202,s: SYS 55363 POKE2035,z:POKE2036,s: SYS65520	POKE214,z:POKE211,s: SYS 58640	Cursor auf Zeile z und Spalte s positionieren auch Cursor-Positionieren

(statt 65520 kann auch 55355 oder 55253 stehen)

ESC-Funktionen kann man ersetzen:

C16	C64	Bemerkung
ESC-D	SYS 59903	löscht Zeile in CURSOR-Pos.
ESC-I	SYS 59749	fügt Zeile ein
ESC-J	PRINTCHR$(13)CHR(145)	CURSOR auf Zeilenanfang
ESC-K	PRINTCHR$(13)CHR(157)	CURSOR ans Zeilenende
ESC-O	POKE 212,0	hebt QUOTE-Mode wieder auf
ESC-V	SYS 59626	scrollt Bildschirm nach oben
ESC-W	PRINT CHR$(19);:SYS59749	scrollt Bildschirm nach unten

ESC-T und ESC-B legen die linke obere und die rechte unter Ecke des Bildschirmfensters fest.

dem Sie Ihren Rechner durch Ein-/Ausschalten „erfrischt" haben, laden Sie zuerst BASICon und starten ihn mit RUN. Er verschiebt sich danach an das obere Ende des BASIC-RAM. Von dort können Sie das Programm mit einem Monitor-Programm absolut abspeichern (zum Beispiel mit dem MLMplus aus INPUT 64, Ausgabe 12/87). Der Wandler liegt im Bereich von 38100 bis 40959.

Mit **SYS 38100** initialisieren Sie den Wandler, um ihn gegen String-Übergriffe von BASIC-Programmen zu schützen. Das BASIC-Ende liegt jetzt bei 38099.

Sie könnten nun ein V3.5-BASIC-Programm mit LOAD " . . .",8,0 in den C64 laden. LIST führt zu wenig lesbaren Ergebnissen. Die „Fremd-Token" erscheinen als unleserliche Großbuchstaben und Sonderzeichen.

List ist alles

Nach **SYS 38100,1** steht Ihnen ein zweiter LIST-Befehl zur Verfügung: !LIST. Hiernach erscheinen die Befehle im Klartext der V3.5-Version auf dem Bildschirm. Dies dient aber nur der Anschauung! Jedes Editieren oder RETURN in einer Zeile führt unweigerlich zum Verlust des V3.5-Textes. RUN würde mit einem SYNTAX ERROR beantwortet.

Für die eigenen Unterlagen können Sie diese Form der Listings ausdrucken:

```
OPEN 4,4,0: CMD4: !LIST
PRINT#4,CHR$(12): CLOSE 4
```

Der letzte PRINT-Befehl führt bei den meisten Druckern zu einem Seitenvorschub und gibt den Druckerkanal wieder frei.

Aber mit SYStem

Um vernünftig weiterarbeiten zu können, stehen Ihnen jetzt zwei weitere Wege offen:

SYS 38100,9

Damit erzeugen Sie ein umgewandeltes BASIC-Programm. Vor jeder Zeile mit einem fremden Befehl steht jetzt ein REM gefolgt von dem Befehlstext des jeweiligen Befehls. Dieses Programm läßt sich abspeichern und jederzeit wieder einladen und editieren. Die Befehle müssen dann entsprechend der BASIC-Syntax umgeschrieben werden, wie Sie es der Tabelle „BASIC-Vergleiche" entnehmen können.

Der zweite Weg führt über den AB-Generator: Wollen Sie sich einen Teil der Vorarbeit ersparen, laden Sie dieses Programm und starten es mit RUN. BASICon wird jetzt auf die ASCII-Schreibweise der INPUT-BASIC-Befehle angepaßt. Sie können auch diese Variante mit einem Monitor abspeichern. Laden Sie ein C16-BASIC-Programm und geben SYS 38100,9 ein, erscheinen hiernach im Listing Befehlstexte, die für INPUT-BASIC vorbereitet sind. Das gewandelte Programm sollten Sie sicherheitshalber auch erst einmal abspeichern.

Wer einen anderen BASIC-Dialekt verwenden möchte, kann im AB-Generator die Befehlstexte in den DATA-Zeilen entsprechend ändern. Wird diese Abwandlung eingesetzt, ändern sich die Tabelle im BASICon und die damit bearbeiteten Programme.

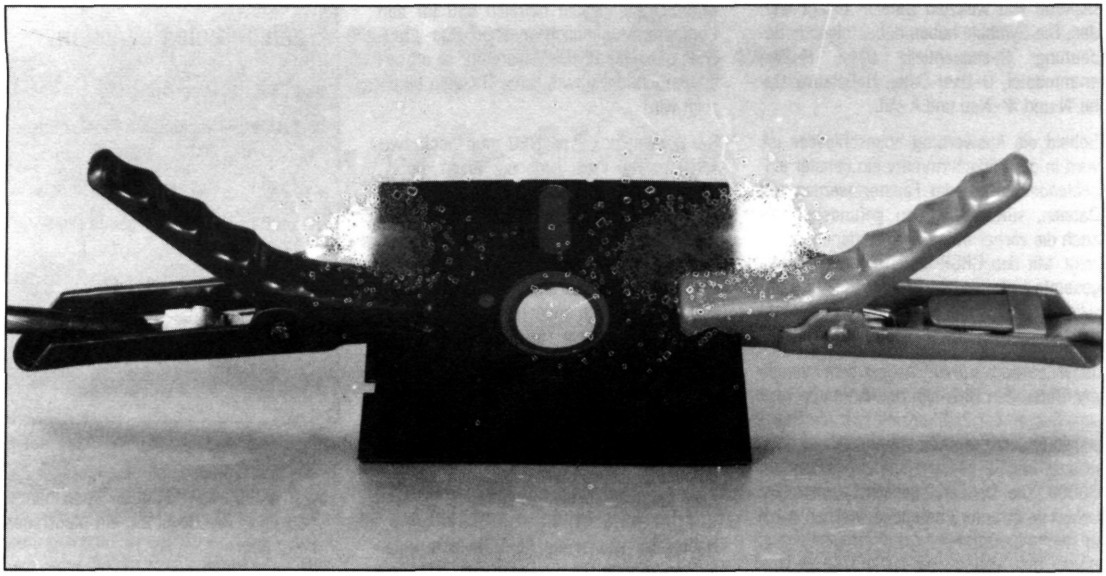
Mit der Grundversion des BASICon können Sie auch INPUT-BASIC-Programme in V3.5-BASIC umändern. Der Weg folgt der obengenannten Beschreibung, nur in umgekehrter Richtung: Sie laden ein INPUT-BASIC-Programm und erhalten nach der Konvertierung ein für V3.5-BASIC vorbereitetes Programm.

Das einzige, was wir Ihnen nicht ersparen konnten, ist die notwendige Arbeit von Hand, da eine Reihe von Befehlen in der Schreibweise und der Parameter-Zuweisung völlig unterschiedlich sind. Von einem „vollautomatischen" Wandler haben wir wegen des unverhältnismäßigen Aufwandes Abstand genommen.

Rudolph Schmid-Fabian/rh

Schutz-POKEs

C16	C64	Bemerkung
772-775	POKE 775,200	LIST-Schutz
806/807	POKE 808,225	RUN/STOP-RESTORE
788/789	POKE 788,52	STOP-Taste



Unsichtbar, aber doch vorhanden

Disktool: Relo-Diskuss

„Herrgottsakramentkruzifixnochmal . . . genau das natürlich nicht!" So oder ähnlich hat sicherlich jeder schon einmal geflucht. Oder haben Sie noch nie ein Programm versehentlich gelöscht? Nun, kann man nichts machen — was weg ist, ist weg — oder vielleicht doch nicht?

Wenn Sie den Scratch-Befehl (oder den New-Befehl ohne ID-Kennung) angewendet haben, sind die gelöschten Daten physikalisch noch vorhanden, denn es wird tatsächlich nur der Eintrag im Inhaltsverzeichnis der Diskette (Directory) gelöscht beziehungsweise so verändert, daß das File nicht mehr angezeigt wird. Gleichzeitig werden die bisher von der Datei belegten Blökke für neue Files wieder freigegeben. Der Inhalt dieser Blöcke ist jedoch nach wie vor derselbe.

Nur Programm laden . . .

Im Laufe eines Diskettenlebens sammelt sich, durch Löschen und Neubeschreiben, einiges an Datenresten auf einer Diskette an. Manche dieser Reststücke sind brauchbare Dateien. Mit den herkömmlichen Hilfsmitteln wäre es aber selbst für einen Fachmann sehr umständlich und zeitraubend, diese Daten wiederzubeleben.

Nicht so bei Relo-Diskuss! Sie laden das Programm und starten es normal mit RUN und RETURN. Danach brauchen Sie nur noch die zu bearbeitende Diskette ins Laufwerk zu schieben und anzuwählen, ob Sie den Schnelldurchlauf (F2) oder den etwas mehr Zeit in Anspruch nehmenden Durchlauf (F1) ausführen wollen.

Der Unterschied zwischen den beiden Modi besteht darin, daß im Schnellgang das Suchen und das Anzeigen der Startadressen der Programme entfällt. Die Startadressen sind jedoch eine wichtige Information, wenn es darum geht, ein Programm von einer Datei oder einem Programmfragment zu unterscheiden. Wenn Sie nur eine zuvor aus Versehen gelöschte Datei wiederherstellen wollen, so können Sie auf die Startadressen getrost verzichten.

. . . und starten, . . .

Nachdem Sie F1 oder F2 gedrückt haben, beginnt das Programm die Diskette zu lesen. Bereits nach einigen Sekunden sind sämtliche für das Wiederherstellen benötige Daten gelesen, und die Auswertung beginnt. Dabei wird auf dem Bildschirm durch Symbole angezeigt, welche Bereiche der

Diskette von welchen Dateien belegt werden. Die Symbole haben dabei folgende Bedeutung: S=sequentielle Datei, P=Programmdatei, U=User-Datei, R=Relative Datei, N und *=Neu und A=Alt.

Sobald die Auswertung abgeschlossen ist, wird in der Bildschirmmitte ein Fenster eingeblendet. In diesem Fenster werden alle Dateien, sowohl die neu gefundenen als auch die vorher bereits vorhandenen, angezeigt. Mit den CRSR-Tasten können Sie die gesamte Liste in dem Fenster auf- und abrollen lassen, wobei jeweils der in der Mitte des Fensters stehende Eintrag editierbar ist.

Jeder Eintrag enthält, neben dem Namen der Datei, den Dateityp, den Anfangs-Track und Sector, die Blockanzahl und die Startadresse. Letztere steht, wenn Sie das Programm mit F2 gestartet haben, immer auf $0000. Die Dateien, die sich auch zuvor schon im Directory befanden, werden dabei in normaler Darstellung angezeigt. Es sind dieses die Dateien vom Typ SEQ, PRG, USR oder REL.

. . . RETURN drücken . . .

Es werden sich aber auf vielen Disketten noch andere Dateien finden. Diese Dateien werden invers dargestellt. Und das sind nun auch die Dateien, die sich bisher unerkannt auf der Diskette befanden! Sobald sich eine solche Datei in der Mitte des Anzeigefensters befindet, können Sie sie durch Betätigen der RETURN-Taste aus ihrem Schattendasein befreien und für den Computer zugänglich machen. Sie brauchen dabei die RETURN-Taste nur so oft betätigen, bis der gewünschte Dateityp angezeigt wird.

Funktionen auf einen Blick

F1	- Programmstart mit Anzeige der Programmstartadressen
F2	- Programmstart ohne Anzeige der Startadressen
F3	- Abspeichern des neuen Directory
F7	- Neustart des Programms
F8	- Programm beenden
RETURN	- Ändern des File-Typs
⇑ / ⇓	- Rollen des Directory

Bei Dateien des Typs NEU muß noch zwischen zwei verschiedenen Arten unterschieden werden. Wenn der Name einer NEU-Datei aus einem Pfeil links und einem Buchstaben besteht, dann gab es zu dieser Datei keinen Eintrag im Directory. Ist jedoch ein anderer Dateiname angezeigt, dann können Sie davon ausgehen, das der angegebene Dateiname keinerlei Bedeutung mehr hat. In diesem Fall ist die Wahrscheinlichkeit, daß Sie eine nutzbare Datei gefunden haben, sehr gering.

Wenn der Dateityp ALT im Feld Typ steht, handelt es sich um ein gelöschtes File, das noch komplett vorhanden ist.

Haben Sie einmal die RETURN-Taste betätigt, so können Sie die Dateitypen NEU und ALT bei dieser Datei nicht mehr einstellen, was auch sinnvoll ist, da NEU und ALT ja keine Typangaben sind, mit denen der C64 etwas anfangen könnte. Wollen Sie diese Datei dennoch löschen, so stellen Sie einfach den Dateityp DEL (deleted) ein. Der Eintrag wird dann wieder invers dargestellt und bleibt auch später, nach dem Neuschreiben des Directory, gelöscht.

. . . und aufatmen!

Die Entscheidung, ob Sie ein File wieder zugänglich machen wollen, kann Ihnen das Programm natürlich nicht abnehmen, und leider ist es dem Programm auch nicht möglich, festzustellen, welchem Typ die Datei früher angehörte. Deshalb werden die „versteckten" Einträge nur mit NEU oder ALT im Feld Dateityp angezeigt. In den meisten Fällen empfiehlt es sich, den Dateityp in PRG umzuwandeln.

Die Chance, ein ehemaliges Programm erwischt zu haben, ist besonders groß, wenn folgendes eintritt:

- Im Feld SADR (StartADResse) steht $0801. In diesem Fall handelt es sich mit großer Wahrscheinlichkeit um ein Programm, das normal geladen und mit RUN gestartet werden kann.

Speicherbelegungsplan

ab **$0801**	- Programmcode
ab **$2000**	- Backup von Blockpointer-Liste
ab **$3000**	- das eingelesene Directory
ab **$4000**	- die eingelesenen Blockpointer
ab **$5000**	- RAM-Directory
ab **$6000**	- aufbereitetes Directory für das Editierfenster
ab **$7500**	- frei

- Die Startadresse ist eine gerade Zahl (z.B. $8000 oder $C000). Dann handelt es sich in der Regel um ein Maschinenprogramm.

Es ist jedoch in beiden Fällen nicht sicher, daß das Programm noch komplett und damit lauffähig ist.

Wenn Sie mit dem Editieren des Directory fertig sind, können Sie durch Druck der Taste F3 das neue Inhaltsverzeichnis (nach einer Sicherheitsabfrage) abspeichern. Neben dem Directory wird auch die BAM (Block Available Map) neu berechnet und abgespeichert.

Das Erstellen der BAM entspricht dem Validate-Befehl des DOS, nimmt jedoch nur einen Bruchteil einer Sekunde in Anspruch. Da der gesamte bisher besprochene Vorgang höchstens ca. 24 Sekunden in Anspruch nimmt (ohne Editieren), lohnt es sich auch, den Relo-Diskuss nur zum Validieren der Disketten zu benutzen.

Nebenbei bemerkt können Sie durch Ändern des File-Typs auf DEL auch ganz normal Dateien aus dem Directory löschen, was mit Relo-Diskuss einfacher und bei größeren Dateien zudem schneller geht als mit dem Scratch-Befehl.

Nachdem Sie das Directory abgespeichert haben, können Sie mit F7 wieder an den Anfang des Programms gelangen und eine weitere Diskette bearbeiten oder mit F8 das Programm verlassen. R. Lowack/WM

In den Tiefen des DOS – Relo-Diskuss in Aktion

Die File-Verwaltung des Disketten-Operating-Systems

Die Einteilung der Diskette in Spuren (Tracks) und Blöcke (Sektoren) findet bereits beim Formatieren statt. Es wird dabei jeweils ein Track geschrieben, wobei dessen Nummer im sogenannten Header (Kopf) des Tracks steht. Je nach Track-Nummer werden zwischen 16 und 20 Sektoren gleichmäßig auf dieser Spur verteilt. Auch jeder Sektor erhält dabei einen Header.

Aufteilung der Sektoren

Tracks	1-17	18-24	25-30	31-35
Sektoren	20	18	17	16

Um verstehen zu können, wie es möglich ist, daß ein Programm, das eigentlich bereits gelöscht war, wieder hervorgezaubert werden kann, muß man ein wenig tiefer in die Funktionsweise der Diskettenspeicherung eindringen.

Wenn das DOS nun eine Datei schreiben soll, so trägt es den Namen der Datei in das Directory ein. Dann sucht es einen freien Block auf der Diskette. Dabei müssen nicht alle Sektoren gelesen werden, denn ob ein Sektor bereits von einer anderen Datei belegt ist, erfährt das DOS aus der Block Available Map (BAM), was frei übersetzt soviel bedeutet wie „Block-Verfügbarkeits-Liste". Dort werden vom Betriebssystem der Floppy nach Abschluß eines jeden Schreibzugriffes die entsprechenden Sektoren als belegt gekennzeichnet.

Files schreiben

Hat das DOS einen freien Sektor für die neue Datei gefunden, trägt es im Directory sowohl die Track- als auch die Sektornummer dieses ersten Blocks ein. Danach wird der nächste freie Block gesucht. Ist er gefunden, so wird in den ersten gefundenen Block die Track- und Sektornummer des zweiten Blocks, gefolgt von den ersten 254 Datenbytes der Datei, geschrieben.

Danach wird der dritte freie Block gesucht, in den zweiten Block die "Adresse" des dritten und die nächsten 254 Bytes der Datei geschrieben, und so weiter. Wenn das DOS feststellt, daß nur noch weniger als 254 Bytes zu schreiben sind, dann sucht es keinen freien Block mehr, sondern schreibt in den letzten gefundenen Block anstatt der Track-Nummer des nächsten Blocks eine Null und anstatt der Sektornummer die Anzahl der noch verbleibenden Bytes gefolgt von den Bytes selbst.

Damit ist die ganze Datei physikalisch vollständig auf die Diskette geschrieben worden. Während des Schreibens zählt das DOS noch die Anzahl der Blöcke mit und schreibt diese nun ebenfalls in den Directory-Eintrag der Datei. Zudem wird dort noch der Dateityp eingetragen.

Wie bereits beschrieben, enthält der Eintrag neben dem Namen der Datei den Anfangstrack und -Sektor, die Blockanzahl und den File-Typ. Der Eintrag für ein File belegt jeweils 32 Bytes. Die genaue Aufteilung entnehmen Sie bitte der Tabelle.

Der Directory-Eintrag

von Byte	bis Byte	Inhalt
0	0	Dateityp
1	1	erster Track
2	2	erster Sektor
3	18	File-Name
19	21	nur bei REL benutzt
22	25	frei
26	27	@-Zeiger
28	29	Blockanzahl
30	32	frei

Wenn das DOS vom Benutzer den Befehl zum Löschen einer Datei bekommt, wird zuerst im Directory nach einem Dateieintrag mit dem angegebenen Namen gesucht.

Files löschen

Wenn der Eintrag gefunden wurde, so wird der erste Block (Track- und Sektornummer stehen ja im Eintrag) gelesen und in der BAM als frei gekennzeichnet. Aus diesem ersten Block entnimmt das DOS die Zeiger auf den nächsten Block. Auch dieser wird in der BAM freigegeben. Es werden wieder die Zeiger geholt, der nächste Block freigegeben und so weiter. Bis in einem Block anstatt der nächsten Track-Nummer eine Null

steht, denn das ist ja offensichtlich der letzte Block der Datei.

Nachdem auch dieser in der BAM als frei gekennzeichnet wurde, wird noch im Directory-Eintrag der Dateityp DEL eingetragen, und bei der nächsten Ansicht des Inhaltsverzeichnisses dieser Diskette ist die Datei verschwunden. Auch die "Blocks free"-Angabe am Ende des Directory weist genau so viele Blöcke mehr auf.

Das nullte Byte

Bit 7	wenn 0, dann '*'
Bit 6	wenn 1, dann '<'
Bit 5	keine Funktion
Bit 4	keine Funktion
Bit 3	keine Funktion
Bit 2	Dateityp
Bit 1	Dateityp
Bit 0	Dateityp

Der Dateityp

%000	$00+$80=$80	DELeted
%001	$01+$80=$81	SEQuentiell
%010	$02+$80=$82	PRoGramm
%011	$03+$80=$83	USeR
%100	$04+$80=$84	RELativ

Solange keine neuen Daten geschrieben wurden, kann man die alte Datei, die ja zur Zeit noch, wenn auch unsichtbar, im Directory steht, durch Ändern von Byte Null des Eintrages (File-Typ) zurückholen. Das läßt sich mit einem Floppy-Monitor ganz einfach machen. Schwieriger wird es, wenn die alte Datei teilweise durch neue Daten überschrieben wurde.

Natürlich sind die Inhalte der überschriebenen Blöcke geändert. Wenn die neue Datei kleiner als die vorher gelöschte war, kann doch wenigstens noch ein Teil der alten Datei gerettet werden. Aber selbst wenn der alte Dateieintrag im Directory noch vorhanden sein sollte, so stimmen in diesem Fall doch Anfangs-Track und Sektor nicht mehr.

Files retten

Die leeren Blöcke erkennt Relo-Diskuss daran, daß sie den Wert $4B (dez.75) im ersten Byte (hier steht die Track-Nummer des Folgeblocks) enthalten. Verwechslungen können dabei nicht auftreten, da es einen Track 75 nicht gibt. Endblöcke haben im ersten Byte eine Null stehen und sind somit auch einwandfrei zu identifizieren. Folgeblöcke haben in den ersten beiden Bytes eine gültige Track- und Sektornummer stehen.

Um eine zusammengehörende Datei wiederherstellen zu können, muß herausgefunden werden, welcher der erste Block der Datei ist. Würden Sie dieses mit einem Floppy-Monitor untersuchen, so würden Sie bald das Handtuch werfen. Sie müßten erst einmal einen Block finden, dessen Inhalt Ihnen zeigt, daß er zu der alten Datei gehört. Dann würden Sie versuchen, den Block rückwärts zu verfolgen. Sie müßten also in allen Blöcken der Diskette nachsehen, ob einer auf diesen Block verweist.

Bei über 600 Blöcken kann das sehr lange dauern. Haben Sie den Vorblock dann gefunden, so müßte wieder der Vorblock dieses Blocks gesucht werden. Wenn die Datei aus 100 Blöcken bestand und Sie ausgerechnet den letzten Block gefunden haben, so wird das zu einer wahrhaft gigantischen Aufgabe. Aber es geht auch anders.

Die Geschwindigkeit, mit der Relo-Diskuss diese Aufgabe meistert, basiert grundlegend auf der Tatsache, daß er erst einmal die Bytes 0 und 1 aller Diskettenblöcke in den Computer liest. Das Programm braucht nun, um alte Dateien zu verfolgen, nicht mehr auf die Diskette zuzugreifen.

Schon während des Einlesens der Blockzeiger werden leere Blöcke und Blöcke mit ungültigen Zeigern gar nicht erst berücksichtigt. Als nächstes löscht Relo-Diskuss aus dieser Liste alle Blöcke, die von bestehenden Programmen belegt werden. Dazu muß zuvor noch das Directory gelesen werden, um die Anfangsblöcke der Dateien zu erfahren.

Dann werden die gelöschten Dateien auf Vollständigkeit geprüft. Es wird dabei davon ausgegangen, daß eine Datei, wenn die Anzahl der tatsächlichen Blöcke mit der Anzahl im Eintrag übereinstimmt (die Blokkanzahl der alten Datei ist beim Löschen der Datei nicht verändert worden) noch komplett ist.

Ist das der Fall, so bekommt der Dateieintrag im Speicher den Typ ALT zugewiesen. Wenn nicht, dann erhält die Datei den Typ NEU. In beiden Fällen werden die von diesen Dateien belegten Blöcke ebenfalls aus der Liste gestrichen.

Resteverwertung

Die jetzt noch in der Liste befindlichen Blöcke gehören zu alten Dateien. Die Erstblöcke werden erkannt, indem alle Blöcke, auf die ein anderer Block verweist, ebenfalls aus der Liste gestrichen werden. Die übriggebliebenen Blöcke können dann weder Folge- noch Endblöcke sein.

Die gefundenen Anfangsblöcke müssen jetzt nur noch darauf geprüft werden, ob sie auch gültige Folgeblöcke besitzen, und wenn ja, wie viele. Gültige Folgeblöcke deshalb, weil es sein könnte, daß sie auf Blökke verweisen, die bereits aus der Liste gestrichen worden sind, weil sie zu bestehenden Dateien gehören.

Die neu gefundenen "alten" Dateien werden nun mit einem künstlichen Namen versehen (Pfeil links und ein Buchstabe, durchgezählt ab "A") und in das im Speicher befindliche Directory eingetragen.

Directory rekonstruieren

Das Directory auf der Diskette ist zu diesem Zeitpunkt noch unverändert. Erst wenn Sie das RAM-Directory abspeichern, sind die neuen Dateien zugänglich. Stellen Sie also fest, daß keine (oder keine brauchbaren) Dateireste gefunden wurden, so brauchen Sie das Directory auch nicht neu zu schreiben, sondern können das Programm verlassen oder neu starten und die nächste Diskette inspizieren.

An dieser Stelle soll noch einmal darauf hingewiesen werden, daß Relo-Diskuss nicht auf kopiergeschützte Disketten angewandt werden sollte, denn es kann dazu führen, daß die Programme danach nicht mehr geladen werden können. Bei „normalen" Disketten kann Relo-Diskuss jedoch keine Schäden verursachen. Solange das neue Directory nicht zurückgeschrieben wird, ändert das Programm kein Byte auf der Diskette.

R. Lowack/WM

Tips zum Speedcompiler

Geordnetes Nebeneinander

Teil 3: Speedcompiler und Maschinensprache

In dieser Folge unserer kleinen Serie geht es noch einmal um ein heißes Thema: die Benutzung von Assembler-Routinen in kompilierten Programmen. Aber nichtsdestotrotz ist diese Folge auch hochinteressant für Nicht-Assemblerfreaks, denn es geht so richtig ans Eingemachte — die Struktur kompilierter Programme wird entblättert und enträtselt, und wir lüften das Geheimnis der „Overlay-Technik".

Ein kompiliertes Programm hat einen vollkommen anderen internen Aufbau als sein Ursprungsprogramm für den Interpreter. Dadurch sind die Variablen dem Maschinenspracheprogrammierer besser zugänglich als unter dem Interpreter. Zudem wird die Speicherplatz-Organisation dadurch vereinfacht, daß der Compiler seine Variablen in einem festen Bereich ablegt, den er ja auch ausgibt. Dadurch wird es möglich, Maschinenprogramme etwa einfach über das Ende der Variablen zu legen oder vor den (heraufgesetzten) Code-Start, ohne mit verschiedenen POKEs Speichergrenzen verschieben zu müssen. Kompilierte Programme benutzen bekanntlich nur den Speicherbereich, den sie wirklich brauchen, und nicht wie der Interpreter den gesamten zur Verfügung stehenden Speicherplatz.

Normal übergeben

Nach einem SYS-Befehl übergibt das Runtime-Modul die Kontrolle an das Maschinenprogramm. Dazu wird die BASIC-SYS-Routine benutzt; so kann man an den bekannten Stellen im RAM (siehe Kasten) die Inhalte der Prozessor-Register vorbesetzen und erhält nach dem Rücksprung dort auch eventuelle Rückgabewerte. Der wichtigste Unterschied zum „Interpreter-SYS": das BASIC-ROM ist während der meisten Zeit der Programmausführung ausgeschaltet ist. Deswegen sind direkte Einsprünge ins BASIC-ROM ($A000 bis $BFFF) nicht möglich! Natürlich kann man sich fragen, ob das sinnvoll ist. Ich meine: „Ja!". Es ist nämlich in der Regel überhaupt nicht ratsam, aus einem kompilierten Programm heraus in den BASIC-Interpreter zu springen. Es geht dabei vor allem um mögliche Kollisionen auf der Zero-Page. Werden nämlich von ROM-Routinen vom Runtime-Interpreter benutzte Zero-Page-Adressen verändert, hat dies natürlich verheerende Folgen.

Will man trotzdem in eine Interpreter-Routine einspringen, muß man zunächst sicherstellen, daß keine der wichtigen Werte in der Zero-Page verändert werden (siehe „Zero-Page-Belegung"). Ist das sichergestellt, kann das BASIC-ROM eingeblendet, die Routine abgearbeitet und vor dem Rücksprung das BASIC-ROM wieder ausgeblendet werden.

Diese dreiteilige Serie soll eine genauere 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.

Runtime-Modul von innen

Aber nun zur Praxis. Eine genaue Beschreibung des Aufbaus des Runtime-Interpreters finden Sie im entsprechenden Kasten.

Wie man auf den ersten Blick sieht, belegen die P-Code-Interpreter-Routinen nicht die gesamten sechs Kilobyte Speicherplatz, sondern in diesen sechs Kilobyte sind noch viele zusätzliche nützliche Speicherbereiche enthalten, die zur Verwaltung benötigt werden. Zum Beispiel werden normale Integer-Variablen in einem reservierten Speicherbereich am Beginn des Runtime-Interpreters gehalten; auch die Adressen der anderen Variablen sind so gespeichert und verbrauchen keinen zusätzlichen Speicherplatz im BASIC-Speicher.

Zwei String-Puffer dienen als Zwischenspeicher für interne String-Operationen. Die Beschränkung auf nur zwei Puffer ist übrigens einer der Gründe, warum manche komplexere String-Operationen aufgeteilt werden müssen.

Der „gemischte" Speicherbereich ab $1E00 beinhaltet sehr verschiedene Daten und darf *niemals* verändert werden. Mit dem sogenannten „Black Board" kommt ein für Maschinensprachefreunde und „POKE-Fans" äußerst interessanter Bereich. Dort kann man verschiedene Daten des Systems erfragen oder verändern. Im einzelnen:

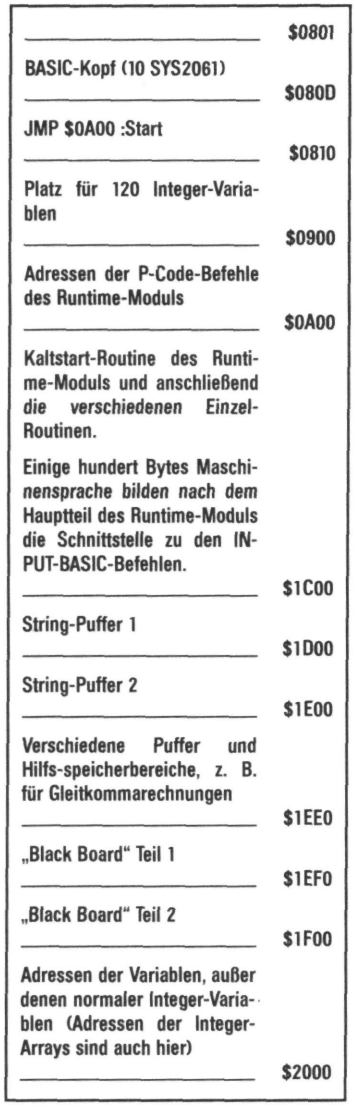
$1EE0 (7904):
aktuelles Stopp-Zeichen für INPUT

$1EE1 (7905):
letzter aufgetretener Gleitkommafehler

$1EE2 und $1EE3 (7906/7907):
reserviert für spätere Erweiterungen

$1EE4 und $1EE5 (7908-7909):
Statusvariable QA (Low-/High-Byte-Format)

$1EE6 (7910):
INPUT-Mode, internes Merkregister



$1EE7 (7911):
I/O-Error, aufgetretener Fehler bei einer Ein/Ausgabe-Operation. Entspricht dem High-Byte der Status-Variable ST.

$1EE8 (7912):
Anzahl der ausgewerteten Zeichen bei der VAL-Funktion

$1EE9 bis $1EEF (7913-7919):
reserviert für spätere Erweiterungen

$1EF0 bis $1EF3 (7920-7923):
Diese vier Bytes dienen als Start-Up-Sequenz für Overlay-Programme. Sie enthalten folgende vier Werte:

$6A, $48: Start-Low/-high; Startadresse des zu startenden Programms

$1EF4 bis $1EF8 (7924-7928):
reserviert für spätere Erweiterungen

$1EF9 (7929):
Gleitkommafehler-Flag (0 oder 128)

$1EFA (7930): I/O-Error-Flag (0,1,2,3,128)
$1EFB (7931): Break-Flag (0,1,128)
$1EFC (7932): zweites Trennzeichen des INPUT-Befehls. Normal: $2C = Komma
$1EFD (7933): INPUT-BASIC-Flag (0,1)
$1EFE und $1EFF (7934/7935): Start-Vektor für das Programm

Das Black Board ist in zwei Teile ($1EE0-$1EEF und $1EF0-$1EFF) aufgeteilt, weil der zweite Teil jeweils vom Compiler angefügt wird. Der erste Teil enthält hauptsächlich Meldungen und Merkregister des Runtime-Modules, der zweite Einstellungen, die den Betrieb direkt modifizieren.

Verrechnungsmelder

Das „aktuelle Stopp-Zeichen" sowie der „INPUT-Mode" sind interne Merkregister, mit denen der Anwender im allgemeinen wenig anfangen kann. Die Status-Variable QA hat auch hier im Black Board Platz gefunden. Bei dem Byte Nummer 2, dem „letzten aufgetretenen Gleitkommafehler", wird es schon interessanter. Ist nämlich die Möglichkeit zum Programmabbruch durch einen Gleitkommarechenfehler abgeschaltet (REM£E2), wird hier die Fehlernummer abgelegt. Die Bedeutung der Fehlernummern kann mit einem kleinen Programm ermittelt werden, dessen Listing wir auf diesen Seiten abgedruckt haben. Will man diese Fehlernummer abfragen, so sollte man am Anfang des Programms '0' in diese Speicherstelle schreiben. Das „I/O-Error-Flag" zeigt die Fehlernummer nach einem Ein-/Ausgabefehler an und ist identisch mit dem High-Byte der Variable ST.

„Anzahl der ausgewerteten Zeichen der VAL-Funktion" ist sehr nützlich, um bei einer Eingabe von Zahlen festzustellen, wie viele Ziffern tatsächlich in eine Zahl umgewandelt werden konnten. Eventuell übriggebliebene Zeichen wurden nicht berücksichtigt und können weiterverarbeitet werden.

„Gleitkommafehler-Flag", "I/O-Error-Flag" und "Break-Flag" sind Flags, die die Abbruchbedingungen des Programms festlegen. Sie können direkt mit den Compiler-Optionen "£E", "£IO" und "£S" beeinflußt werden. Das „INPUT-BASIC-Flag" zeigt die Benutzung von INPUT-BASIC-Befehlen an.

Das „Trennzeichen für den INPUT-Befehl" legt fest, welches Trennzeichen als „schwaches" Trennzeichen benutzt wird, also anstelle des sonst üblichen Kommas. Man kann diese Einstellungen auch mittels der Compiler-Option "£Z" vornehmen, solch nützliche Zeichen wie '$0D' (Carriage Return, was quasi eine Abschaltung dieses Trennzeichens bewirkt, da $0D schon erstes Trennzeichen ist) oder '$00' sind aber nur durch direktes 'POKEn'n möglich.

Zweiklassen-Variablen

Der Speedcompiler unterscheidet hauptsächlich zwei Klassen von Variablen: die normalen Integer-Variablen (keine Arrays), die in einem speziellen Speicherbereich gehalten werden und auf die dadurch besonders schnell zugegriffen werden kann, und alle anderen Variablen, die im Hauptspeicher hinter dem Hauptprogramm abgespeichert werden. Damit man auch auf sie möglichst schnell zugreifen kann, gibt es eine Liste mit Zeigern (den sogenannten Vektoren), die jeweils auf den Anfang der Variable oder des Arrays zeigen.

Beim Programmlauf wird intern nur über eine Nummer auf die einzelnen Variablen oder Arrays zugegriffen. Diese Nummern legt der Speedcompiler beim Kompilieren fest, sie entsprechen genau der Reihenfolge ihres Auftauchens im Programmtext. Wenn man sich die Variablen ausgeben läßt, so entspricht die Ausgabereihenfolge diesen Nummern. Die erste Variable trägt die Nummer Null.

Die Liste der Integer-Variablen ist in eine Liste mit den niederwertigen und eine mit den höherwertigen Bytes unterteilt, jede Integer-Zahl verbraucht somit Bytes. Das Low-Byte wird an der Adresse $0810+Platznummer (2064+Platznummer) abgespeichert, das High-Byte an Adresse

dez.	hex.	6502-Register
780	030C	Akku
781	030D	X-Register
782	030E	Y-Register
783	030F	Status-Register

Über diese Adressen kommunizieren BASIC- und Maschinenprogramme beim SYS-Befehl.

$0888+Platznummer. Zu beachten ist dabei, daß eine Schleifen-Variable, die in einer FOR-NEXT-Schleife benutzt wird, zwei Listenplätze verbraucht: der erste beinhaltet die Variable selbst, der zweite dient als Zwischenspeicher für den Schleifenendwert. Dieser Listenplatz wird bei der Ausgabe der Variablen durch zwei Doppelpunkte signalisiert.

Die Adressen der anderen Variablen (nicht die Werte!) werden ebenfalls getrennt nach Low- und High-Byte gespeichert, Low-Bytes ab $1F00, High-Bytes ab $1F80. Natürlich gibt es auch hier eine Besonderheit. String-Arrays benötigen zwei Listenplätze, auf dem zweiten Listenplatz steht die feste maximale String-Länge (plus 1). Ohne Kenntnis dieser Länge könnten die einzelnen Teil-Strings eines Arrays nicht angesprochen werden. Bei der Ausgabe der Variablen wird diese zusätzliche Position durch ":!" und den Variablen-Typ '0' angezeigt.

Ordentliche Zeiger

Im Hauptspeicher hinter dem Hauptprogramm herrscht im allgemeinen geordnete Unordnung. Integer-Arrays, Strings, String-Arrays, Gleitkomma-Variablen und Gleitkomma-Arrays erscheinen wüst durcheinandergewürfelt. Die Variablen wurden einfach vom Compiler ohne Rücksicht auf ihre Typen und Funktion im Hauptspeicher in der Reihenfolge, in der er die Variablen im Programmtext vorgefunden hat, abgelegt. Durch die Adreßliste ab $1F00 kann sie das Runtime-Modul trotzdem jederzeit finden, und zwar nach fest vereinbarten Konditionen.

```
10 for i=41374 to 41767
20 a=peek(i):printchr$(a);
30 if a<127 then 50
40 n=n+1:printtab(25)n
50 next
```

Diese fünf BASIC-Zeilen geben Aufschluß über die Bedeutung der BASIC-Fehlernummern.

Da sind zunächst einmal die Integer-Arrays. Wie die normalen Integers sind sie in Low- und High-Byte aufgeschlüsselt. Aber hier sind die beiden Bytes nicht getrennt, sondern immer Low-, dann High-Byte hintereinander. Die Adresse eines Integer-Wertes innerhalb des Arrays errechnet sich also wie folgt:

Adresse Low-Byte =
Vektor-Adresse+Positionsnummer im Array*2

Strings werden grundsätzlich so dargestellt, daß das erste Zeichen (auf den der Vektor zeigt) die aktuelle String-Länge und die weiteren Bytes den String selbst beinhalten. Man braucht also immer ein Byte mehr als die String-Länge, um einen String zu speichern. Bei String-Arrays sind diese Strings jeweils hintereinander gespeichert. Die Adresse eines Strings ergibt sich durch die Formel:

Adresse=
Vektor-Adr.+Pos.-Nr.*(String-Länge+1)

Eine besondere Art des String-Arrays ist das „Array of Char" (siehe zweite Folge). Bei diesem Datentyp wird nur das Zeichen ohne String-Länge mitgespeichert. Die String-Länge (0 oder 1) ergibt sich daraus, ob das Zeichen '0' oder einen anderen Wert enthält. Die Adresse eines Zeichens im Array ergibt sich aus der Summe von Vektor-Adresse und Positionsnummer.

Gleitkomma-Variablen werden immer als fünf hintereinanderfolgende Bytes gespeichert. Das Format entspricht dem des BASIC-Interpreters. Schleifen-Variablen benötigen allerdings 15 Bytes: die ersten fünf stellen die Variable dar, dann folgt der STEP-Wert, zuletzt der Schleifenendwert. Gleitkomma-Arrays sind aufgebaut wie Integer-Arrays, nur daß immer fünf Bytes eine Variable ergeben.

```
10      a$=""
...
1000    load"utility 1",8,1
...
2500    input a$
2510    poke 780,74: rem asc("j")
2520    sys 50000
2530    if peek(782)=1 then ...
...

:start  ldx $1f00
        stx $50
        ldx $1f80
        stx $51
        pha
        ldy #$00
        lda ($50),y
        sta $02
        pla
:loop   iny
        cmp ($50),y
        beq end
        cpy $02
        bne loop
        ldy #$00
:end    rts
```

So werden String-Variablen des Kompilats dem Assembler-Programmierer zugänglich. Die BASIC-Zeilen zeigen beispielhaft, wie der entsprechende Hochsprachen-Partner aussehen könnte.

Hände weg von den Zero-Page!

Mehrdimensionale Arrays werden im Speicher wie eindimensionale Arrays abgelegt, mit dem einzigen Unterschied, daß sich die Positionsnummer aus den einzelnen Index-Werten durch Multiplikation einzelner Indizes mit festen Faktoren und anschließender Addition ergibt.

Von der Zero-Page-Belegung war oben schon die Rede; natürlich werden noch andere Adressen verändert, wenn die Rechenroutinen des Interpreters genutzt werden oder INPUT-BASIC arbeitet. INPUT-BASIC arbeitet beispielsweise mit den beliebten Adressen ab $FB, die der Speedcompiler in Ruhe läßt.

Zum Abschluß noch ein kleines Utility zur Demonstration. Es ist ein Programm, das

Die Belegung der Zero-Page

$02	Temporärer Speicher
$08-$10	String-Verwaltung
$11	Temporärer Speicher
$13	I/O-aktiv-Flag
$14-$15	Rechen-Akku 1 für Integer
$16-$17	Laufzeiger in P-Code
$18-$19	DATA-Zeiger
$1A	Stapelzeiger für Integer
$1B	„Benutzung des FAC"-Flag
$20	Gleitkomma-Stapelzeiger
$21	„Benutzung der String Akkus"-Flag
$22-$23	Hilfszeiger 2
$24-$25	Hilfszeiger 3
$2B-$44	Integer-Rechen-Stack
$50-$51	Rechen-Akku 2 für Integer

verschieblich gehalten ist und mit dem INPUT-Ass (dem in Ausgabe 6/86 veröffentlichten Macro-Assembler) für jede Adresse assembliert werden kann. Es kann dann von einem kompilierten Programm nachgeladen und benutzt werden. Es handelt sich um eine einfache String-Suchroutine. Der String muß als erste Variable im Programmtext des Programms auftauchen (wegen des Platzes in der Vektorliste). Die Routine kann einfach mit SYS aufgerufen werden, nachdem man in Speicherstelle 780 den ASCII-Wert des zu suchenden Zeichens übergeben hat. Anschließend ist in der Speicherstelle 782 die Position auslesbar, an der das Zeichen in String 1 steht. Ist es nicht enthalten, steht hier eine '0'. Die beiden Listings zeigen ein Beispielprogramm in BASIC und die Suchroutine in Assembler.

Das Programm benutzt dabei nur drei Zero-Page-Adressen, die im allgemeinen für Maschinenunterprogramme unbedenklich sind.

Wenn's eng wird

Wenn der vorhandene Speicherplatz für umfangreiche Programmprojekte nicht mehr ausreicht, werden in der Regel Programmteile ausgelagert und bei Bedarf nachgeladen. Diese Programmteile werden als „Overlays" oder „Nachlader" bezeichnet.

Solche Konstruktionen werden auch vom Speedcompiler unterstützt. Zum Nachladen von Programmen gibt es nämlich zwei Möglichkeiten, den LOAD-Befehl einzusetzen. Die Sekundär-Adresse '1' hat ihre bekannte Bedeutung behalten: Mit

LOAD"Name",Gerät,1

kann man ein Programm oder Daten, die als Programm-File abgespeichert sind, an die Adresse, die im File selbst angegeben ist, laden. Sekundäradresse '2' bewirkt, daß ein Overlay-Programm, das als „Nachlader" ohne Runtime-Modul kompiliert wurde, nach dem Laden sofort gestartet wird. Dies ist dann notwendig, wenn das ursprünglich im Speicher befindliche Programm durch das nachgeladene überschrieben wird.

Wenn der Nachlader das nachladende Programm nicht überschreibt, so lassen sich auch zwei unabhängige Programme gleichzeitig im Speicher halten. Man kann dann mittels „GOTO#" direkt in die einzelnen Programmteile beider Programme einspringen. Das Argument von „GOTO#", die Sprungadresse, muß vorher ermittelt worden sein, indem man beim Kompilieren die Zeilenadressen speichern läßt.

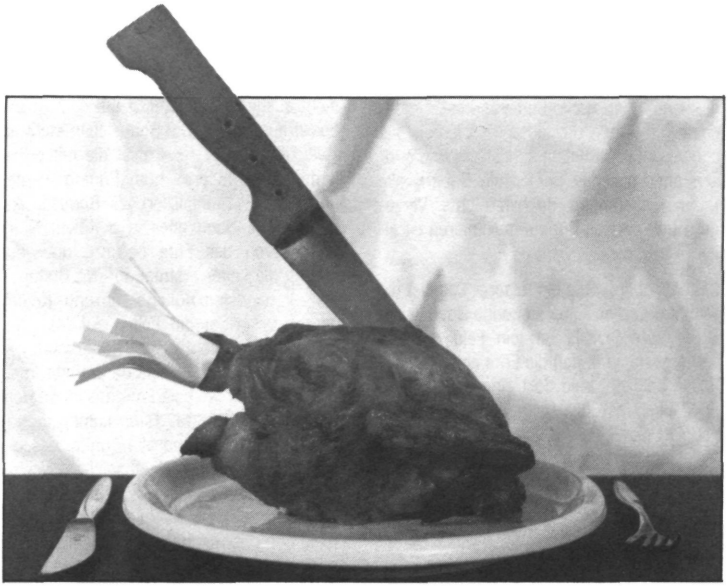
Um die Variablen des Programms beim direkten Einsprung zu löschen, muß entweder ganz zum Anfang („Beginn Programmcode") oder direkt auf einen CLR-Befehl gesprungen werden. Damit beide Programme völlig unabhängig und resident im Speicher liegen können, ohne daß die Variablen des ersten das zweite Programm löschen, muß der Code-Start des zweiten Programms über dem Variablen-Ende des ersten liegen.

Damit die Variablen-Werte der einzelnen Programme beim Umschalten erhalten bleiben, darf man nicht mit „GOTO#" zum Code-Anfang springen, da sonst die Variablen des jeweiligen Programms gelöscht werden.

Jürgen Lindemeyer/JS

	$0800
Runtime-Module für beide Programme	
	$2000
Programm-Code 1	
Variablen für Programm 1	
Programm-Code 2 (Nachlader)	
Variablen für Programm 2	

So können sich zwei Kompilate den Speicher teilen



Paccen wir's an

Spiel: Lady Duck

Das Spiel gehört zur Gattung der bekannten PacMan-Spiele: Ein konzentrisches Labyrinth, in dessen quadratischen Bahnen wir uns, vertreten zum Beispiel durch unseren PacMan, bewegen, um die auf den Wegen gleichmäßig verstreuten Punkte zu fressen oder aufzusammeln. Sind alle Punkte gesammelt, erscheint der nächste Level: jeder Punkt ergibt dann mehr Zähler auf dem mitlaufenden Scoreboard. Aber damit Langeweile erst gar nicht aufkommt: auch die Feinde unseres PacMans gewinnen pro Level (Stage oder Bild) an Schnelligkeit, List und Tücke.

Das legendäre Ur-PacMan hatte bekanntlich berühmte Nachfolger, wie „Mrs. PacMan" und den Spielhallenhit „Lady Buck". Hier war PacMan zum ebenso niedlichen Marienkäferchen geworden, das in gleicher Manier Punkte im Labyrinth fraß, durch Aufsammeln eingestreuter Herzchen Extrazähler erreichte und von allerlei unangenehmem Getier gejagt wurde. Dieses Spiel, das vor allem Kinder in seinen Bann zog, liegt nun überarbeitet und originell abgewandelt, brandneu als 'Lady Duck' für den C64 vor.

Lauf Ente lauf

Hierbei ist wiederum das Ur-PacMan-Labyrinth beibehalten, als Akteur aber watschelt diesmal eine erkennbare Entenfrau durch die Gänge, um weiße Körner aufzupicken, die laufend als Punktgewinn gutgeschrieben werden. Die herumlaufenden Männchen verteilen zufällig hier und da noch rote Küken, die bei Berühren ein Weiterschalten der rechts unten im Bilde stehenden Bonusleiter ergeben. Ein ganz neues Feature dieses Spieles gegenüber den Vorläufern, dazu am Schluß noch ausführlicher.

Die Feinde unserer Ente sind ebenfalls die Männchen, die man als amerikanische Farmer (am Hut) erkennen kann und die unserer Dame nachstellen, um sie vermutlich dem Küchenchef zu übergeben. Jedenfalls kostet jede Berührung „Ente mit Farmer" ein Leben, und ein Spiel hat nur vier Entenleben.

Hat aber unsere laufende Spielente trotz der Bauern alle Körner endlich aufgepickt, dann erscheint das nächst höhere, schwerere, schnellere und die Punkte höher bewertende Bild (hier genannt Stage). Die Spannung steigt, bis auch das Leben der letzten Ente ausgehaucht und das laufende Spiel beendet ist.

Giftige Pillen

Nach dem Start des Spieles kann man zunächst ein vierstelliges (zu ratendes) Code-Wort eingeben, das dann größere Freiheiten in der Wahl der Level zulässt, im Grunde aber als ein Hack-Versuch die Spielfreudigkeit wecken soll. Dazu auch das Versprechen, daß das Code-Wort als Belohnung im 10. Bild verraten wird – aber bis dahin muß man bei all den emsigen Farmern erst mal kommen! Alsdann kann zwischen der Steuerung der Lady über (T) Tastatur und alternativ über (J) wie Joystick (bitte in Port 2) gewählt werden. Hierzu der Tip, daß sich zu Anfang die Joystick-Version aus dem Stand spielen läßt, die Tastatur-Variante aber den Vorteil der größeren Schnelligkeit bietet. Man kann allein oder auch zu zweit spielen.

Zusammen mit der Ente läuft auch schon unter melodischem Klang unser erster Farmer los, er startet aus dem Zentrum des Irrgartens, unsere Ente darunter, also Obacht, daß man sie nicht gleich in seine Arme steuert. Die Enten watscheln, die Farmer aber laufen recht geschwind.

Die Spiel-Level steigern dieses Tempo: In höheren Leveln watscheln die Enten schneller, die Farmer aber laufen noch schneller, dem Spieler schließlich verbleibt weniger Zeit zu gucken, zu taktieren, zu lenken, auszuweichen und zu verschwin-

Taste	Funktion
A	Ente nach oben
Z	Ente nach unten
/	Ente nach rechts
.	Ente nach links
Space	Fire

Damit die Ente auch weiß wo es lang geht.

den. Die Enten werden mit höherem Level tendenziell auch schneller gefangen.

Schnelle Bauern

Jedenfalls als generellen Tip: da die Enten langsamer als die Farmer laufen, nehmen Sie keinen Wettlauf auf gerader Strecke an, sondern steuern Sie die Enten geschickt weg, ehe sie gefangen werden. Besondere Vorsicht auch, wenn es um's Eck geht, da sind die Enten gegenüber den Farmern noch unbeholfener.

Den Enten helfen aber, im Labyrinth überall gut sichtbar, verteilte Klappen. Den Bogen zugunsten der Ente hat der Spieler schnell heraus: Die Ente schlüpft durch die Klappe und dreht diese dabei so, daß der nacheilende Landmann vor einer ihm unüberwindlichen Barriere steht und umkehren muß. Er kann das Drehkreuz nicht bewegen, der Spieler sollte es aber mit seiner Ente oft und geschickt tun, das hilft zu überleben. Ein weiterer Verbündeter der Enten sind hier und da herumliegende Giftpillen (Totenköpfe), wobei jegliche Berührung das jeweilige Leben (Ente oder Bauer) kostet, nur die durch uns gesteuerte Ente meidet die Pillen bewußt, die Bauern aber rennen so, als ob es keine Pillen gäbe, und das kostet manchem von ihnen mit lauter Detonation das Leben. Dennoch haben schließlich und endlich am Spielende immer die Bauern alle Enten eingefangen, da hilft nichts. Verzögern und derweil Punkte maximieren ist alles.

Ein besonderer Gag bei unserer Lady ist die rechts unten im Bilde stehende Bonusleiter. Diese wird jeweils um ein Feld dadurch weitergeschaltet, daß die Ente eines der per Zufall bald da, bald dort eingestreuten roten Küken berührt: beim ersten Berühren schaltet die Leiter auf '2x', beim zweiten auf '3x', beim dritten auf 'No Bonus', beim vierten auf '4x', beim fünften auf 'No Bonus', beim sechsten auf 'Special' (eine Überraschung!), beim siebenten Kükentreffen schließlich auf 'Extraduck'.

Noch mehr Punkte

Klar, daß diese Leiste den Spielspaß deutlich steigert, und mit seinen 'No Bonus'-Feldern der eigentliche Clou des Spieles ist. Man stelle sich vor, daß man schon zwei Küken und dazu eine Menge Körner eingesammelt hat. Eine Menge Guthaben ist also schon auf dem Bonuskonto, aber noch nicht ausgezahlt (das heist auf dem Scoreboard aufaddiert). Die Bonusleiste steht inzwischen auf '3x', das heist die mit dieser laufenden Ente erreichten Punkte werden noch mit '3' multipliziert als Bonuspunkte vorgemerkt. Dann aber wird Küken Nummer 4 von der Ente berührt, und jetzt springt die Leiter erstmal auf 'No Bonus' – bis zur nächsten Kükenberührung. Kommt es dazu, springt die Leiter auf '4x'. Was aber, wenn es nicht mehr dazu kommt, weil die Ente doch noch einem der Farmer in die Arme lief oder weil die Ente das letzte Korn noch eben aufpickte? Dann kommt man in das nächste Bild, und alle Punkte des Bonuskontos wären in einer solchen 'No Bonus'-Position verloren. Deshalb sollte man unbedingt erst das nächste Küken einsammeln und das Bild oder das Entenleben keinesfalls vorher beenden (leicht gesagt, oft schwer getan).

So richtig Punkte gibt es beim Spiel (10 000 und mehr) ohnehin nur durch konzentriertes Bonuspunktesammeln oder durch mehrmaliges Hochwechseln ins nächste Bild.

Alles in allem ist Lady Duck eine würdige und durchaus packend spielfreudige Tochter im Familienkreis der Pac-Männer.

W. Schmidt-Pabst/kfp

ID-Werkstatt:

Treueprämien

Dateien für Vokabeltrainer, INPUT-CAD und INPUT-Calc 64/128

Die erste Datei wurde in Ausgabe 11/87 veröffentlicht, es handelt sich um einen französischen Zeichensatz für den **Vokabeltrainer** (aus 3/87). Leider war die Belegung der C=-Tasten mit den Umlauten keine sehr glückliche Idee, weil diese nämlich Funktionen wie Drucken und ähnliches auslösen.

Sie finden unter dem Namen „france.ok" jetzt einen französichen Zeichensatz mit folgender Tastaturbelegung:

Was ist die Datenverarbeitung ohne Daten? Sinnlos. Darum liefern wir Ihnen hier Anwenderdaten für bereits veröffentlichte Programme.

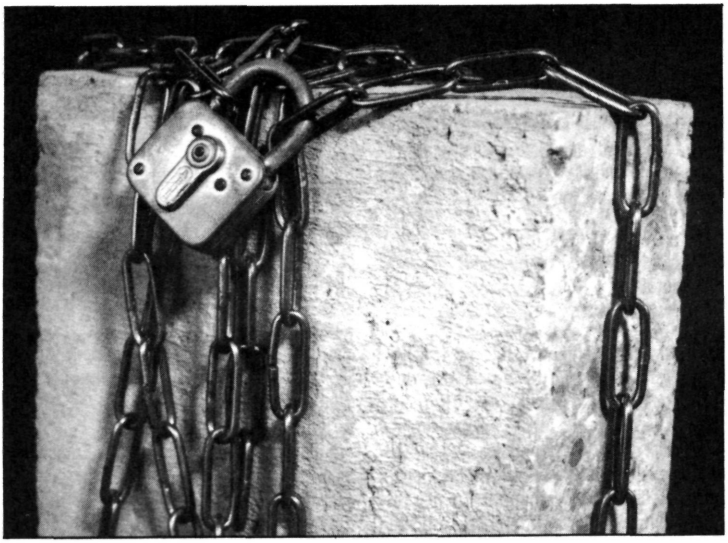
â = [, ê =], î = <, ô = >, û = @, ç = #, é = £, è = *, à = ↑

Als Trostpflaster gibt es einen deutschen Zeichensatz, die Belegung entspricht der deutschen DIN-Tastatur:

ä = ;, Ä =], ö = :, Ö = [, ü = @, Ü = Shift/@, ß = £

Einige Mißverständnisse gab es bei der Bedienung des Programms CHARMAKER, mit dem die Zeichensätze in den Original-Vokabeltrainer eingebunden werden. Bei der Abfrage, welche Sonderzeichen selbstdefiniert belegt werden sollen, sind neun Zeichen vorgegeben, zu denen auch der Doppelpunkt zählt. Diese müssen mit DEL gelöscht werden, um eigene Zeichen eingeben zu können. Die Meldung „SYNTAX ERROR IN 1987" nach Beendigung des Programms ist ein reiner Schönheitsfehler und kann ignoriert werden.

Unter dem Namen „ANALOG-CONSTRUCT" können Sie für das in den Ausgaben 11/86 bis 2/87 veröffentlichte Zeichen- und Konstruktionspaket **INPUT-CAD** eine Datei mit elektronischen Symbolen zum Schaltungsaufbau abspeichern.



Ein Wort genügt

Paßwort-Datenschutz mit Patron Coder

Manch originelle Ideen und mühsam gesammelte Informationen sind geistige Schätze, die es wert sind, vor unerwünschten Fremdzugriffen geschützt zu werden, so daß nur Eingeweihte, die das „Sesam öffne dich" kennen, Hand anlegen dürfen. Beim Patron Coder wird ein Paßwort zum Zauberschlüssel, der Dateien und auch komplette Programme vor allzu Neugierigen verbirgt und erst für den, der den Schlüssel besitzt, den Zugang freigibt. Selbst ein chiffrierter Diskettenbrief kann dann nur noch vom Empfänger geöffnet werden.

Ist Ihnen das auch schon einmal passiert? Sie stecken mitten in der Entwicklung eines Programms oder einer Datei und geben eine Vorversion an einige Bekannte weiter, nur um zu erfahren, was an Ihrer Idee noch verbesserungswürdig ist. Einige Wochen später dann der Ärger: Sie halten eine Version in den Händen, an der andere weitergeschrieben haben oder Sie finden Teile Ihrer Idee in anderen Veröffentlichungen wieder. Kann man sich dagegen schützen? Verschlüsseln Sie Ihre Daten auf Ihrem Datenträger mit Patron Coder so, daß nur derjenige Sie lesen kann, der das richtige „Geheimwort" kennt.

Geheimnisträger

Einige Methoden, Texte zu chiffrieren, konnten Sie bereits in INPUT 64 in den Ausgaben 3, 6, 9, 12 des Jahres 1987 in der Serie „Einer gegen Alle" kennenlernen. Es ging darum, eine möglichst intelligente Umwandlungsformel zu finden, mit der ein Text kodiert und dekodiert werden kann.

Patron Coder gibt sich jedoch mit solch einfachen Verfahren nicht zufrieden. Hier geht es um mehr: Sinn der Sache ist es, nur dem Absender und dem Empfänger der geschützten Informationen den Zugriff zu ermöglichen. Deshalb verwendet diese Chiffrierung ein sogenanntes Paßwort (Schlüsselwort). Mit diesem Wort wird das File, das Sie verschlüsseln wollen, chiffriert und ist danach selbst mit einem Diskettenmonitor nicht mehr zu ‚entziffern". Jeder Versuch, ein chiffriertes File zu laden und zu starten, scheitert, wenn das Paßwort nicht genau so angegeben wird, wie vorher festgelegt.

Sie sehen, wer das Zauberwort nicht kennt, steht vor verschlossener Tür. Erst wenn Sie oder ein anderer Eingeweihter das richtige Paßwort in der BASIC-Zeile eingeben, die nach dem Laden des chiffrierten Files mit einem LIST-Befehl erreichbar ist, startet der Dekodierer und dechiffriert das File. Haben Sie eine Datei verschlüsselt, erzeugt der Patron Coder eine Datei auf Ihrer Diskette, die mit der Ursprungsdatei identisch ist. Handelte es sich um ein Programm, startet dieses automatisch nach der Rückverwandlung.

Innerhalb von INPUT 64 können Sie sich anhand einer Vorführung ansehen, wie Sie Patron Coder bedienen müssen. Wenn Sie das Programm benutzen wollen, müssen Sie sich natürlich Patron Coder mit CTRL-S erst einmal auf eigenen Datenträger sichern. Benutzen Sie Patron Coder, um ein eigenes File zu chiffrieren, geht dies natürlich nur außerhalb von INPUT 64 und am besten nach kurzem Ein- und Ausschalten Ihres Rechners. Laden Sie zuerst den Coder und starten ihn mit „RUN". Geben Sie den Namen des Files ein. Verwenden Sie keine Namenszusätze, geht der Coder davon aus, daß es sich um ein Programm handelt, und bindet einen Programmstarter mit in das erzeugte File ein. Anschließend kommen Sie der Aufforderung nach, Ihre Diskette einzulegen, auf der das File zu finden ist, das Sie wandeln möchten.

Wollen Sie eine Datei vor fremden Zugriffen schützen, können Sie Patron Coder dazu veranlassen, indem Sie an den Dateinamen ein Komma und anschließend das Kürzel für den File-Typ anhängen: Sie schreiben beispielsweise ein „ ‚p" bei einer Programmdatei, „ ‚s" bei einer sequentiellen und „ ‚u" bei einer User-Datei („ ‚r" ist natürlich nicht zulässig, es würde auch keinen Sinn ergeben). Wenn Sie die eine derart kodierte Datei wieder laden und die De-

Strategie

1. Namen Ihres zu bearbeitenden Programmes eingeben und Diskette mit Ihrem Programm einlegen.

   Das Programm wird nun geladen.

2. Adresse eingeben, an der Ihr Programm angesprungen werden soll (als Hexadezimalzahl). Wenn Sie „0" eingeben, wird ein RUN ausgeführt.

3. Inhalt von Speicherstelle $01[1] angeben.

4. Geben Sie nun "j" oder "n" ein. War es ein "n", dann lesen Sie bei Punkt 6 weiter.

5. Sie können nun ein Paßwort eingeben, daß Sie vor dem Starten des Programmes anstelle Ihres Textes in der BASIC-Zeile zwischen die beiden Anführungszeichen einfügen müssen.

6. Hier geben Sie den Text ein, der hinter dem SYS-Befehl in der ersten BASIC-Zeile steht.
   Das Programm wird nun kodiert.

7. Letztendlich müssen Sie einen Namen für das kodierte File eingeben und eine Zieldiskette einlegen.

8. Ende.

[1]Für BASIC ist es $F7 (247), $F6(246), wenn der BASIC-Interpreter ausgeschaltet werden soll, $F4 (244), wenn Sie 64 KByte-RAM wünschen.

chiffrierung starten, wird eine echtes Duplikat Ihrer ursprünglichen, unkodierten Datei auf Diskette erzeugt.

Doppelmaske

Normalerweise genügt es, ein Programm zu kodieren, um anderen die Einsicht zu verwehren. Manchmal jedoch möchte man möglichst niemanden an eine neue „top-secret"-Version heranlassen. Der beste Schutz ist dann die Paßwort-Funktion: Sie geben ein Paßwort ein, dann den Text, der im chiffrierten Programm in der BASIC-Zeile erscheinen soll.

Beispiel:
Paßwort: Alea jacta est!
BASIC-Zeile: Was ist gefallen?

Wenn Sie ein Programm mit Paßwort-Schutz versehen haben, können Sie das Programm nach dem Laden erst starten, wenn Sie den Text in der BASIC-Zeile durch Ihr Paßwort ersetzt haben:

Zeile nach Laden und „LIST":
2001 sys(2088) "was ist gefallen?"
Zeile lauffähig:
2001 sys(2088) "alea jacta est! "

Wenn Sie das Programm starten, wird der Bildschirm während der Dechiffrierung ausgeschaltet. Am Ende der Prozedur schaltet sich der Bildschirm wieder ein, jedoch mit der Hintergrundfarbe Schwarz. Anders, wenn das Paßwort falsch eingegeben wurde: Das Programm stürzt mit Sicherheit ab und ein LIST-Befehl zeigt höchsten Unsinn auf dem Bildschirm.

Spezialist

Die Paßwort-Option muß sich allerdings auf Programme bis 154 Blöcke beschränken, sonst würde dem BASIC-Interpreter allzu viel zugemutet. Ihr Programm darf zwar den Bereich $01C0-$FFFF (448 – 65535) belegen, allerdings darf es – außerhalb der Pass-Wort-Option – „nur" maximal 240 Blöcke lang sein. Wobei Ihnen das Problem, Programme zu laden, die länger als 202 Blöcke sind, nicht erspart bleibt.

Hintergründiges

Chiffrierung ist ein geschichtsträchtiges Thema: Schon um 800 vor Christi Geburt waren Informationen für die Spartaner so kostbar, daß sie sich intensiv um Chiffrierungen bemühten. Durch alle nun folgenden Epochen waren Fachleute gefragt, die sich mit der Ver- und Entschlüsselung von Texten befaßten. Größtenteils spielten Geheimschriften bei militärischen Auseinandersetzungen eine Rolle. Vor allem mit der Entwicklung der Datenübertragung per Telegraf und Funk wuchs das Bedürfnis, unerwünschte Lauscher auszuschalten.

Ein Beispiel ist die ENIGMA, die im Zweiten Weltkrieg entwickelt wurde. Sie ist eine Art Schreibmaschine, deren Alphabet sich mechanisch nach einem bestimmten Schlüssel umordnen ließ. Immerhin ermöglichte Sie mehr als eine Billion verschiedener Einstellungen.

Als die beiden ersten elektronische Rechenmaschinen entwickelt wurden, um den Code der ENIGMA zu dechiffrieren, gelang es auch diese Geheimcodes zu knacken. Einen entscheidenden Sprung nach vorne machte diese Verschlüsselungstechnik, die man auch Kryptologie nennt, durch die Erfindung und Weiterentwicklung von elektronischen Geräten, die es erlaubten, in kurzer Zeit viele Operationen auszuführen: die Computer.

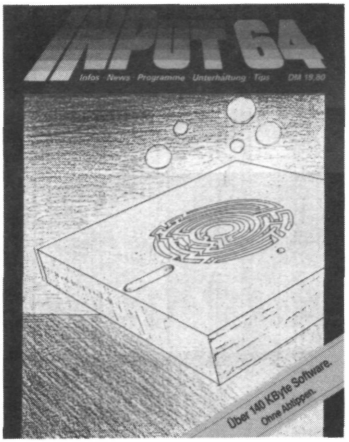
Sie beschleunigten nicht nur die Entschlüsselung, sondern ließen es auch zu, kompliziertesten Algorithmen in kurzer Zeit abzuarbeiten, das heißt, man konnte darangehen, Programme besser zu verschlüsseln. In den USA wurde sogar ein landesinternes System, DES genannt, entwickelt, um Daten zu schützen. Ein Zweig der modernen Mathematik, die Komplexitätstheorie, bewirkte neuen Aufruhr. Dieser Bereich behandelt auch die sogenannten Falltür-Funktionen, mathematische Gebilde, die sich in der einen Richtung "relativ" leicht berechnen lassen. Bei Versuchen den Rechenweg umzukehren, müssen jedoch die mächtigsten Computersysteme aufgeben.

Ein typischer Fall für das ausgeprägte Interesse von Staat und Wirtschaft an diesem Thema ist die Entwicklung des „Data Encryption Standard", eine 1977 festgelegte Norm.

Es handelt sich um eine Chiffrierung, bei der Blöcke zu je 56 Bit durch Vertauschen einzelner Glieder sowie Ersetzen durch andere und Anfügen einer 8 Bit langen Fehlerkennung verschlüsselt werden. IBM entwickelte hierfür einen speziellen Chip, der diesen Vorgang unterstützt. Im Ausschuß, der diese Norm festlegte, saßen NBS (Norm-Kontroll-Behörde), IBM und die NSA. Die NSA (nicht NASA) ist die höchste Geheimdienstbehörde der USA. Der DES-Code soll für 15 Jahre sicher sein, so heißt es der Norm nach, die von NBS, dem National Bureau of Standards, aufgestellt wurde.

Michael Niemeier/rh

Am 7. Februar an Ihrem Kiosk: INPUT 64, Ausgabe 2/88



Wir bringen unter anderem:

Digitest

Simulieren statt Löten! Digitest ermöglicht Aufbau, Verdrahtung und Test digitaler Schaltungen. Die wichtigsten Logik-Bausteine – Konverter, BCD-Decoder, verschiedene Gatter und Flipflops – sind als Bausteine vorhanden, aufgebaut wird auf einer „theoretischen Platine", die sich über mehrere Bildschirme ausdehnt. Nicht nur für Bastler, sondern auch als Einübung in digitale Logik.

Labyrinth

Gesucht wird die beste Strategie, sich in einem Labyrinth gegen zwei Katzen durchzusetzen. „Durchsetzen" ist genaugenommen der falsche Ausdruck, „Ausweichen und Durchwurschteln" heißt die Devise in diesem Spiel, das verschiedene Schwierigkeitsgrade mit unterschiedlichen Ent- und Verschleierungstaktiken bereithält.

Scrange

Scrange steht für die Kombination von „Screen" und "Change"; es geht um ein Tool zur Ansteuerung von verschiedenen Textbildschirmen ohne komplizierte PEEKs und POKEs. Damit stehen verschiedene Notizzettel jederzeit abrufbar im Rechner bereit, man kann zwischen verschiedenen Listing-Teilen hin- und herschalten, schnell zwischen verschiedenen Ausgaben wechseln und und und . . .

c't – Magazin für Computertechnik

Ausgabe 2/88 – ab 15. Januar am Kiosk

Software-Know-how: 3-D-Darstellungen – Von Vektoren und Ebenengleichungen * Praxistip: Inversionen – Hardware-Schaltung für inverse Bildschirme für Model 30 oder PC1512 * Prüfstand: Ataris Laserdrucker SLM 804 * Preiswerte Festplatten über OMTI-Controller an Ataris DMA-Port * u.v.a.m.

elrad – Magazin für Elektronik

Ausgabe 2/88 – ab 22. Januar am Kiosk

Bauanleitung Stromversorgung: Netzgerät 0 . . .16 V/20 A * Report Satelliten-Direktempfang: Schüsseln und Empfänger für Fernmeldesatelliten * Der Weg zum eigenen Meßlabor (3): Oszilloskope * Die elrad-Laborblätter: Schaltungstechnik Infrarot-Fernbedienungen * Bauanleitung Meßtechnik: Effektivwert-Wandler * u.v.a.m

INPUT 64 - Abonnement Abruf-Coupon

Ja, übersenden Sie mir bis auf Widerruf alle künftigen INPUT-64-Ausgaben

ab Monat _____

(Kündigung ist jederzeit mit Wirkung ab der jeweils übernächsten Ausgabe möglich. Überbezahlte Abonnementgebühren werden sofort anteilig erstattet.)

Das Jahresabonnement kostet auf Diskette DM 198,— inkl. Versandkosten und MwST.

Absender und Lieferanschrift:
Bitte in jedes Feld nur einen Druckbuchstaben (ä = ae, ö = oe, ü = ue)

Vorname / Zuname

Beruf / Funktion

Straße / Nr.

PLZ Wohnort

_____
Datum/Unterschrift

Von meinem Recht zum schriftlichen Widerruf dieser Order innerhalb einer Woche habe ich Kenntnis genommen. Zur Wahrung der Frist genügt die rechtzeitige Absendung.

Unterschrift: _____

Bitte beachten Sie, daß diese Bestellung nur dann bearbeitet werden kann, wenn beide Unterschriften eingetragen sind.

INPUT64 - Abonnement Abruf - Coupon

Ich wünsche Abbuchung der Abonnement-Gebühr von meinem nachstehenden Konto. Die Ermächtigung zum Einzug erteile ich hiermit.

Name des Kontoinhabers

Bankleitzahl Konto-Nr.

Geldinstitut Ort des Geldinstitus

Bankeinzug kann nur innerhalb Deutschlands und nur von einem Giro- oder Postscheckkonto erfolgen.

IMPRESSUM:

INPUT 64 Das elektronische Magazin

Verlag Heinz Heise GmbH & Co KG
Helstorfer Straße 7 · 3000 Hannover 61
Postfach 61 0407 · 3000 Hannover 61
Telefon: (05 11) 53 52-0

Technische Anfragen:
nur dienstags von 9.00 — 16.30 Uhr

Postgiroamt Hannover, Konto-Nr. 93 05-308 (BLZ 250 100 30)
Kreissparkasse Hannover, Konto-Nr. 000-01 99 68 (BLZ 250 502 99)

Herausgeber: Christian Heise

Redaktion:
Christian Persson (Chefredakteur)
Jürgen Seeger (stv.)
Ralph Hülsenbusch, Wolfgang Möhle
Karl-Friedrich Probst

Redaktionsassistent: *Wolfgang Otto*

Ständige Mitarbeiter:
Peter S. Berk, Irene Heinen, Peter Sager
Hajo Schulz, Eckart Steffens

Vertrieb: Anita Kreutzer

Grafische Gestaltung:
Wolfgang Ulber, Dirk Wollschläger,
Ben Dietrich Berlin

Herstellung: Heiner Niens

Lithografie: Reprotechnik Hannover

Druck: SONOPRESS GmbH, Gütersloh

Diskettenherstellung:
SONOPRESS GmbH, Gütersloh

INPUT 64 erscheint monatlich.
Jahresabonnement Inland DM 198,—
Einzelpreis DM 19,80

Redaktion, Abonnementverwaltung:
Verlag Heinz Heise GmbH & Co KG
Postfach 61 04 07 · 3000 Hannover 61
Telefon: (05 11) 53 52-0

Abonnementverwaltung Österreich:
Erb-Verlag GmbH & Co KG Abt. Zeitschriftenvertrieb
z. Hd. Frau Pekatschek
Amerlingstraße 1 · A-1061 Wien
Telefon: (00 43 2 22) 56 62 09
(00 43 2 22) 57 94 98
(00 43 2 22) 57 05 25

Jahresabonnement: Diskette DM 210,—

Vertrieb (auch für Österreich, Niederlande, Luxemburg und Schweiz):
Verlagsunion Zeitschriften-Vertrieb
Postfach 57 07 · D-6200 Wiesbaden
Telefon: (0 61 21) 2 66-0

Verantwortlich:
Christian Persson
Helstorfer Straße 7 · 3000 Hannover 61

Eine Verantwortung für die Richtigkeit der Veröffentlichungen und die Lauffähigkeit der Programme kann trotz sorgfältiger Prüfung durch die Redaktion vom Herausgeber nicht übernommen werden.

ISSN 0177-3771

Titelidee: **INPUT 64**
Titelillustration: S. Wustmann, Dortmund
Fotografie: Lutz Reinecke, Hannover
Titel-Grafik und -Musik:
Tim Pritlove, Fabian Rosenschein
Betriebssystem: Hajo Schulz

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