DAS ELEKTRONISCHE MAGAZIN 9/87

INPUT 64

Infos · News · Programme · Unterhaltung · Tips

DM 19,80

Unverbindliche Preisempfehlung

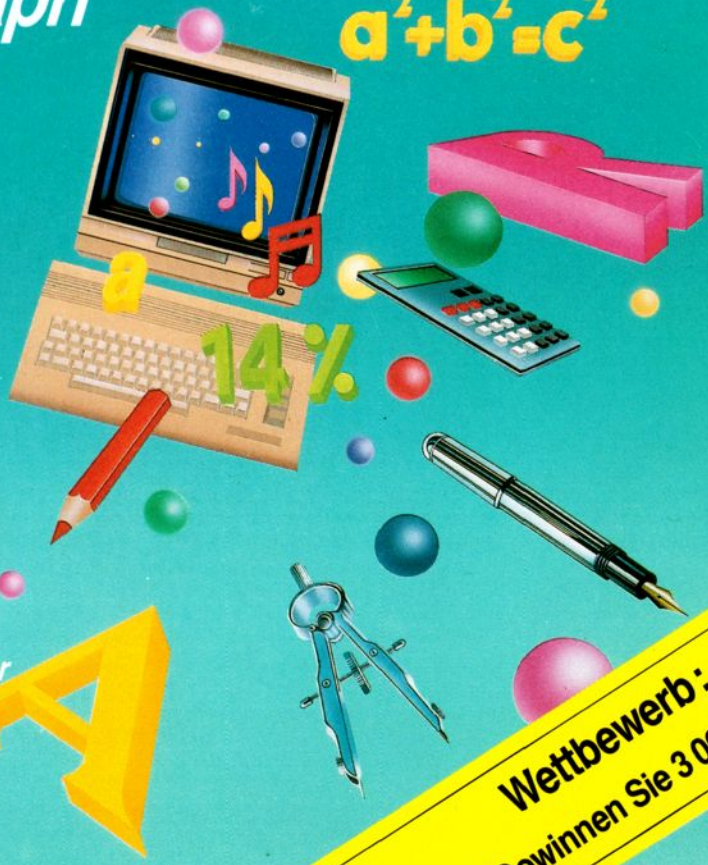
öS 170,- sfr 19,80

Multicolor-BASIC

Grafik programmieren

INPUT-Graph

Daten präsentieren



Spiele:
Farb-Basar
Tiere-Raten

Tool: Screen-Überblender

Serien:
64er Tips
Englische GRAMmatik
Einer gegen alle

Wettbewerb:
Gewinnen Sie 3 000 DM

INPUT-ein Magazin aus dem Verlag Heinz Heise GmbH, Postfach 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 vorhandene Steckmodule und schalten Sie den Rechner einmal kurz aus und wieder ein. Geben Sie nun zum Laden der Diskette

LOAD "INPUT*",8,1 und RETURN

ein. Alles Weitere geschieht von selbst.

Es wird nun zunächst ein 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-Drucker 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)!

Nicht immer tut der Volksmund Wahrheit kund, von Kindern und Betrunkenen wollen wir hier erst gar nicht reden. Die Rede ist von all den Sprichwörtern, die auf den Unterschied von äußerer Wirkung und innerer Funktion eingehen, „mehr Sein als Schein" empfehlen, unter der „rauhen Schale" einen „weichen Kern" vermuten und derlei.

Alles passé. Spätestens ein Blick auf die millionenschweren Werbeetats der Industrie und zugriffsgerechte Mitnahmepakkungen lehrt: auf die Hülle kommt es an. Produkte, denen die entsprechende Präsentation fehlt, verschwinden vom potentiellen Käufer unbemerkt in der Ladenhüterecke.

Womit wir beim Stichwort wären. „Daten-Präsentation" ist angesagt in der Datenverarbeitung. Früher begnügte man sich damit, daß Anwender-Daten eingegeben, berechnet und in Textform wieder ausgegeben wurden. Kennzeichnend waren über den Bildschirm laufende Zahlenkolonnen oder endlose Meter nüchtern bedruckten Papiers. Heutzutage gehört eine gefällige grafische Aufbereitung beinahe zu den Standard-Anforderungen. Zu Recht – schließlich ist die Darstellung von Funktionswerten als Kurven- oder Säulen-Diagramm wesentlich anschaulicher als das nackte Zahlenmaterial.

Im wissenschaftlichen Bereich stellt sich dieses Problem noch extremer: Ergebnisse von Meßreihen müssen als Funktionsgraphen, also in grafischer Form, dargestellt sein, um überhaupt mit vertretbarem Aufwand interpretiert werden zu können.

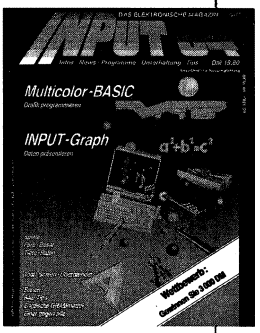
Programme, die ausschließlich der grafischen Präsentation von Daten dienen, sind die logische Konsequenz dieser Überlegungen. Mit INPUT-Graph in dieser Ausgabe stellen wir Ihnen Software zur Verfügung, die eben diesen Präsentationszwecken dient. Der kleine, aber feine Unterschied zu „reinen" Präsentationsprogrammen: Die Daten können nicht nur dargestellt, sondern auch auf grafischer Ebene verändert, zurückgeschrieben und als Benutzer-Daten weiterverarbeitet werden.

Obwohl wir insofern voll im Trend liegen, soll an dieser Stelle noch auf einige ganz und gar unäußerliche, das Innenleben von INPUT 64 angehende Veränderungen hingewiesen werden. Mit dem Wegfall der Kassettenversion unseres Magazins konnte das Betriebssystem der Diskettenversion noch einmal benutzerfreundlicher gestaltet werden. Der Platz, den die Kassetten-Routinen belegt haben, wird nun für Disk-Befehle benutzt, man kann jetzt innerhalb des Magazins andere INPUT-Ausgaben nachladen, und – eher eine Äußerlichkeit – das Layout der Hilfsseite wurde geändert.

Darum sollten auch langjährige INPUT-Kunden mal wieder einen Blick auf die Bedienungshinweise werfen. An dieser exponierten Stelle im Heft sind sie ja kaum noch zu übersehen. Und daß INPUT 64 die Hüllen abgeworfen hat und sich dem Käufer mit leicht veränderter Gestalt präsentiert, ist Ihnen ohnehin schon aufgefallen. Außerdem ist das nun wirklich ganz und gar äußerlich.

Jürgen Seeger

9/87



INHALT

Leser fragen . . .

Byte-Compaktor ausgebremst

Wie kann ich verhindern, daß der Compaktor (Ausgabe 5/87) ein Programm nach dem Dekompaktieren automatisch anstartet? (telef. Anfrage)

In der Tat kann es – zum Beispiel bei nicht neustartfähigen Programmen – durchaus notwendig sein, die ansonsten sehr bequeme Autostart-Funktion zu verhindern. Sie sollten in diesem Fall das kompaktierte Programm laden und danach im Direkt-Modus 'POKE 2233,1' eingeben. Wenn Sie jetzt das Programm mit 'RUN' starten, wird es nur dekompaktiert, aber nicht gestartet.

(d. Red.)

GOLUM festgesetzt

. . . bei diesem guten Spiel tritt leider folgender Fehler auf: Wenn eine Spielfigur von zwei Blockaden eingeschlossen ist, eine Zahl größer als Eins gewürfelt wurde und nun mit dieser Figur versucht wird zu ziehen, ist dies nicht möglich. (H. Mai, Wien)

Ein Spiel, das in einer solch ausweglosen Situation endet, nimmt einem schon den Spaß. Zur Befreiung aus ausweglosser Sackgasse: laden Sie GOLUM in den Rechner, geben 'POKE 19839,105' ein und speichern unter GOLUM II wieder ab. Jetzt dürfte nur noch der Sieger das Spiel beenden können. (d. Red.)

SC-Toolbox komplett

Jedesmal, wenn ich mein „Gesamtprogramm" entsprechend der Anleitung auf Seite 15, Tabelle 2 aus INPUT 64, Ausgabe 8/87 abgespeichert habe, finde ich nur einen Block auf der Diskette wieder. Nach 'LIST' erscheint nur der BASIC-Teil. 'RUN' führt zum Absturz des Rechners. (telef. Anfrage)

In der Beschreibung wird stillschweigend vorausgesetzt, daß Sie vor dem Abspeichern Ihres Gesamtprogramms 'POKE 44,8' und 'POKE 43,1' im Direktmodus eingegeben haben, damit der Programmanfang wieder zurückgesetzt wird. Für Anwender, die mit Floppy-Beschleunigern arbeiten, noch ein Tip: Laden Sie das von der SC-Toolbox erzeugte Tool mit ' . . . ,8,0', damit es an den BASIC-Start bei $0801 (2 049) geladen wird. Die SC-Toolbox erzeugt das Tool im Speicherbereich ab $6000 (24 576) und schreibt diesen Wert natürlich auch in den ersten Sektor als Startadresse.

(d. Red.)

Tips über Null

Auf Seite 25 in INPUT 64, Ausgabe 8/87, schreiben Sie unter „Adresse der CIA 2: 56 576", die Bits 0/1 für den 16 K-Bereich bei $0000 (0) müßten auf '00' gesetzt werden, genauso wie für den 16 K-Bereich bei $C000 (49 152). Kann das denn sein? (telef. Anfrage)

Die Bits 0/1 in der Adresse 56 576 ($DD00) müssen selbstverständlich auf '11' gesetzt werden, will man den 16 K-Bereich ab $0000 (0) ansprechen. Die Darstellung im Heft ist an dieser Stelle eindeutig falsch, sozusagen „doppelt invertiert und umgedreht" . . . (d.Red.)

Tiefe Trauer

*Ich möchte mich in **tiefer Trauer** von Ihnen verabschieden und bedanke mich für die schönen Tage mit INPUT 64. Ein Kassettenanwender! H. Wulf, Lübeck*

INPUT-Calc 64/128 im Redaktionstest

Als wir während der Redaktionsarbeiten zu dieser Ausgabe INPUT-Calc 64/128 für Berechnungsprobleme bemühen wollten, entdeckten wir, daß das Programm sich anders verhielt, als es zu erwarten war. Im 128er Modus führten Lade- und Speicherversuche auf und von Diskette zu unerträglich langen Wartezeiten, beim Versuch, nur einzelne Blöcke zu behandeln, gar zum endgültigen Stillstand des Systems. In beiden Modi war die Suchfunktion über SHIFT-RETURN aussichtlos, wenn vorher über den Hilfseditor eine Formel eingegeben wurde.

Laden Sie INPUT-Calc 64/128 vom eigenen Datenträger im **64er Modus (!)**, starten es mit 'RUN' und unterbrechen das Programm innerhalb der ersten Menüs mit RUN/STOP-RESTORE. Danach können Sie diese beiden Fehler durch folgende kurze Eingaben beheben:

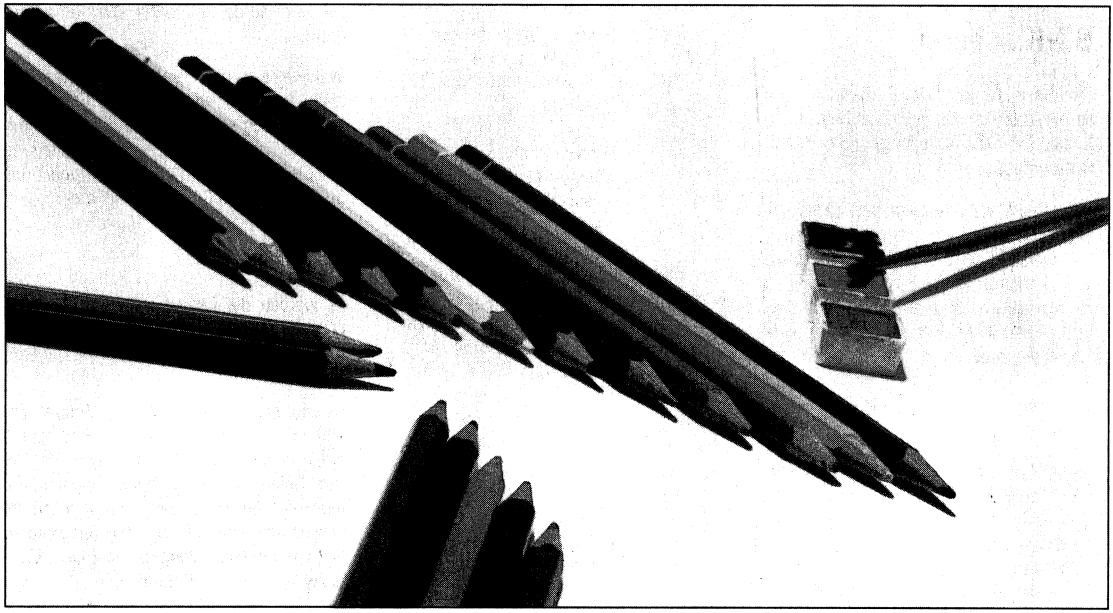
Korrektur zum Laden/Speichern für C128:

'POKE 2357,254:POKE 4249,212'

Korrektur der Suchfunktion

Ersetzen Sie im BASIC-Listing in den Zeilen 2 510 – 2 550 die Variable f0$ durch fq$ (siehe auch Listing 1).

Anschließend geben Sie 'POKE 44,8' und 'POKE 43,1' im Direktmodus ein, damit der Programmanfang wieder zurückgesetzt wird, und speichern unter neuem Namen wieder ab. (d.Red)



Schnell und bunt mit Komfort

Multicolor-Grafikpaket IMC-BASIC

Die Programmierung der mehrfarbigen Grafik des C64 ist in BASIC eine komplizierte und zeitraubende Angelegenheit. Die Spracherweiterung INPUT-Multi-Color-BASIC beinhaltet neben den gängigen Grafikbefehlen noch Leckerbissen wie einen FILL-Befehl, der wirklich alles füllt, kinderleicht zu handhabende Sprite-Befehle, Maßnahmen zur Unterstützung des Disketten-Handlings und „intelligente" MERGE- und SAVE-Funktionen.

Im Multicolor-Modus des C64 steht dem Programmierer ein Zeichenschirm mit 32 000 einzeln ansprechbaren Punkten zur Verfügung. Je Punkt sind vier Farben wählbar. Die Ausdehnung beträgt in x-Richtung (waagerecht) 160 Punkte, in y-Richtung (senkrecht) 200 Punkte. Die vier möglichen Farben pro Punkt werden dadurch realisiert, daß jeder Punkt durch zwei Bits dargestellt wird, die bekanntermaßen vier

32 000 Punkte und 16 Farben

mögliche Zustände annehmen können. Wer weiteres Interesse an „Bit-Klauberei" hat, sei auf den gleichnamigen Kasten verwiesen; für die Arbeit mit IMC-BASIC sind derlei Kenntnisse nicht notwendig.

Statt dessen treffen wir eine einfache Vereinbarung: Der Nullpunkt des absprechbaren „Zeichenbretts" liegt oben links, gezählt wird computertypisch von 0 bis 159 in x- und von 0 bis 199 in y-Richtung.

Start wie gehabt

Das Programm IMC-BASIC wird mit LOAD"name",8,0 geladen und mit RUN gestartet. Das Programm setzt dann den BASIC-Start auf $2E01. Falls vor dem Abspeichern kein BASIC-Programm angehängt war, wird ein NEW ausgeführt, ansonsten wird das BASIC-Programm mit RUN gestartet. BASIC-Programm und -erweiterung können zusammen abgespeichert werden, indem man vorher den Befehl POKE44,8 eingibt.

Bit-Klauberei

Der 64er „kennt" zwei Grafik-Modi: den HiRes-Modus (HiRes steht für High Resolution, also hohe Auflösung) und den Multicolor-Modus.

Im HiRes-Modus repräsentiert jedes gesetzte Bit einen Bildpunkt, die Auflösung beträgt 320 Punkte in x- und 200 Punkte in y-Richtung. In diesem Modus können im Normalfall nur zwei Farben auf dem Bildschirm dargestellt werden: die eine Farbe, wenn der Punkt (das Bit) nicht gesetzt ist (Hintergrundfarbe) und die andere Farbe, wenn ein Punkt gezeichnet wurde.

Im Multicolor-Modus wird jeder Bildpunkt durch zwei nebeneinander liegende Bits dargestellt, da für einen Bildschirm aus hardware-bedingten Gründen aber nicht mehr Speicherplatz spendiert wird als für den HiRes-Schirm, reduziert sich die Auflösung in x-Richtung auf 160 Punkte.

Mit den vier möglichen 1/0-Kombinationen der zwei Bits pro Punkt lassen sich vier verschiedene Farben darstellen. Damit der Computer nun weiß, welche Farbe angezeigt werden soll, müssen den einzelnen Bildpunktwerten noch die eigentlichen Farben zugeordnet werden. Wo sich der Computer diese Informationen besorgt, zeigt folgende Tabelle.

%00	Hintergrundfarbe
%01	Video-RAM, untere 4 Bit
%10	Video-RAM, obere 4 Bit
%11	Farb-RAM

Wer diese Tabelle aufmerksam studiert, wird zwei Dinge festgestellt haben. Erstens: Genaugenommen kann ein Punkt nur drei Farben annehmen, da ein Punkt mit der gleichen Farbe wie der Hintergrund nicht sichtbar ist. Zweitens, sozusagen im Sinne der ausgleichenden Gerechtigkeit, können insgesamt auf dem Schirm 16 Farben dargestellt werden, da für jede 4*8-Matrix Video- und Farb-RAM mit anderen Werten belegt werden können. Jedem Byte dieser Farbquellen entspricht nämlich besagte 4*8-Matrix.

Weitere Informationen zum diesem Thema und zu den Möglichkeiten des Video-Chips finden Sie in den 64er Tips und der Sampler-Diskette „Bits & Bytes im Video-Chip", eine überarbeitete und erweiterte Zusammenfassung der gleichnamigen Serie aus INPUT 1–5/85.

In der anschließenden Befehlsbeschreibung gelten einige Syntax-Vereinbarungen:

● Optionale Parameter (also solche, die angegeben werden können, aber nicht angegeben werden müssen) stehen in eckigen Kammern ([]).

● Die Anfangsbuchstaben der Abkürzungen sind funktional gewählt, und zwar, falls nicht anders angegeben:

f für Farbe (0 bis 15)
x für x-Koordinate (0 bis 159)
y für y-Koordinate (0 bis 199)
ga für Geräteadresse der Floppy (8 oder 9)
sa für Sekundär-Adresse (0 oder 1)

Die Zahlen in Klammern bezeichnen die möglichen Grenzwerte, Überschreitungen werden mit einem „Illegal quantity error" quittiert.

Die Befehle sind nach Funktionsgruppen geordnet, nämlich allgemeine Grafikbefehle, spezielle Grafikbefehle, Programmierhilfen und Sprite-Befehle.

Für den Hausgebrauch: Allgemeine Grafikbefehle

MULTI ist die kürzeste Möglichkeit, die Grafik einzuschalten und alle notwendigen Einstellungen vorzugeben. Da dieser Befehl eine Zusammenfassung von vier anderen Befehlen ist, werden zunächst diese erklärt und anschließend wird noch einmal auf „MULTI" eingegangen.

COLDEF f0,f1,f2,f3 [,xl1,yl1,xl2,yl2]

Der COLDEF-Befehl muß unbedingt am Anfang des Programms stehen. Er legt die vier Farben (von schwarz=0 bis hellgrau=15) fest, auf die der Computer von nun an zugreift.

Wahlweise kann zusätzlich ein Rechteck angegeben werden, in dem die Farben gesetzt werden sollen, und so der Bildschirm in Zonen unterschiedlicher Farbfestlegung eingeteilt werden. xl kann Werte von 0 bis 39, yl Werte von 0 bis 24 annehmen.

Beispiel :
10 COLDEF 0,1,2,7

Es werden die Farben Schwarz (0), Weiß (1), Rot (2) und Gelb (7) festgelegt.

COLOR f

Dies ist sozusagen der zweitwichtigste Befehl, er sollte am Beginn eines Programms direkt dem COLDEF-Befehle folgen. Mit diesem Befehl wird die aktuelle Zeichenfarbe bestimmt. Der Parameter f gibt nämlich an, welche der vier mit COLDEF festgelegten Farben verwendet werden soll. f kann Werte zwischen 0 und 3 annehmen.

Beispiel :
10 COLDEF 0,1,2,7
20 COLOR 3

Alle folgenden Grafikausgaben zeichnen mit der Farbe Gelb (7).

MCON

Der Multicolor-Bildschirm wird eingeschaltet. Kehrt das Programm durch eine Fehlermeldung, durch den END-Befehl oder durch die STOP-Taste in den Direktmodus zurück, so wird die Grafik automatisch wieder abgeschaltet.

Bei eingeschaltetem Grafikbildschirm sollten keine Ausgaben mehr auf den normalen Bildschirm gehen, da sonst die Farben zerstört werden können. Der BASIC-Befehl STOP bewirkt KEINE automatische Rückkehr in den Textmodus.

MCOFF

schaltet wieder den Text-Bildschirm an.

CLS [f]

Dieser Befehl löscht den Grafikschirm. Da hinter dem Befehl angegeben werden kann, mit welcher Bit-Kombination (0–3) der Bildschirm vollgeschrieben werden soll,

könnte man auch sagen: CLS färbt den Bildschirm mit der Farbe f.

Beispiel :
30 CLS 0

CLS 0 füllt den Bildschirm mit Nullen, ist also gleichwertig mit CLS.

MULTI [f0,f1,f2,f3 [,xl1,yl1,xl2,yl2]]

Der Multibefehl hat die gleiche Syntax wie der COLDEF-Befehl. Dieser Befehl faßt die Befehle COLDEF, COLOR, CLS und MCON zusammen und erleichtert dadurch das Initialisieren. Es gibt zwei Formen:

1. MULTI ohne Parameter
ist gleichwertig mit
CLS:COLDEF 0,11,12,15:COLOR 3:MCON.

2. MULTI f0,f1,f2,f3,xl1,yl1,xl2,yl2
entspricht der Befehlsfolge
CLS:
COLDEFf0,f1,f2,f3,xl1,yl1,xl2,yl2:
COLOR 3:MCON

PSET x,y

Setzt einen Punkt in der durch COLOR bestimmten aktuellen Farbe.

PGET x,y,Variable

Einer numerischen Variablen wird der Bildpunktwert (von 0 bis 3) des Punktes x,y zugeordnet.

DRAW x,y

Zieht von der aktuellen Position des Grafik-Cursors aus eine Linie zu den angegebenen Koordinaten.

Beispiel :
```
10 COLDEF 0,1,2,7
20 COLOR 1
30 MCON
40 CLS
50 PSET 0,0
60 DRAW 159,199
70 GET A$:IF A$="" THEN 70
```

Dieses Programm zieht eine weiße (COLOR 1 !) Linie quer über den ganzen Bildschirm und wartet dann auf das Drücken einer Taste.

LINE x1,y1,x2,y2

Es wird eine Linie von den Koordinaten x1,y1 bis x2,y2 gezogen. Dieser Befehl ist gleichwertig mit der Befehlskombination PSET x1,y1 : DRAW x2,y2.

CIRCLE mx,my,rx,ry

Die Koordinaten mx und my geben den Kreismittelpunkt, rx den Radius in x-Richtung und ry den Radius in y-Richtung an.

Gleiche Werte für rx und ry ergeben keinen Kreis. Denn durch die Hardware-Zuordnung, daß einem Punkt zwei Bits in x-Richtung, aber nur ein Bit in der y-Ausdehnung entsprechen, ist das Zeichenbrett verzerrt. (Aus diesem Grund zeichnet der Befehl LINE0,0,100,100 auch keine 45-Grad-Diagonale.) Durch welchen Trick Kreise rund werden, ist dem Demo-Programm zu entnehmen: der y-Radius muß das 1.9fache des x-Radius betragen. Es gibt keine ausdrückliche Überlaufbehandlung. Stellt der Programmierer nicht sicher, daß der Kreis innerhalb des Zeichenbretts bleibt, kann es zu unangenehmen Wrap-around-Effekten kommen. (Wem das nichts sagt: Ausprobieren!)

HARDCOPY mo

Gibt eine Hardcopy in 4facher Größe auf den Drucker aus.

mo = 0 : 7-Nadel-Drucker
mo = 1 : 8-Nadel-Drucker

Für Adventure-Freaks: Die Spezial-Befehle

Der FILL-Befehl: Wer hat nicht schon neidisch professionelle Grafik-Adventures betrachtet, in denen nicht nur mit Farbe geLINEed und geFILLed wird, sondern sogar Schattierungen und Muster gezeichnet werden. Dies ist jetzt kein Problem mehr:

DESIGN f0,f1,f2,f3

Mit DESIGN wird das Muster festgelegt, mit dem gefüllt werden soll. Er ist vergleichbar mit dem COLOR-Befehl. Die Parameter können Werte zwischen 0 und 3 annehmen.

FILL x,y

Die Fill-Routine ist zwar nicht besonders schnell, füllt aber sehr zuverlässig jede um-

Drei mal SAVE

Innerhalb von INPUT 64 läuft ein Demonstrationsprogramm, daß einige der Fähigkeiten von IMC-BASIC plastisch darstellt. Nach dem Start dieser Demo können Sie wählen, ob Sie eines der angebotenen Programme auf eine eigene Diskette überspielen wollen oder erst die Demonstration genießen möchten. Im letztgenannten Fall drücken Sie einfach 'D'.

Wollen Sie die Files 'SAVE'n, sind drei Versionen wählbar; naheliegenderweise durch die Tasten '1' bis '3'. Die jeweils aktuelle Version wird in der untersten Bildschirmzeile angezeigt, diese wird dann wie üblich über die CTRL-S-Funktion abgespeichert.

Version 1 schränkt den BASIC-Speicher durch Herabsetzung des BASIC-RAM-Endes um 9,5 KByte ein, ist aber „INPUT 64-verträglich".

Version 2 schränkt den Platzbedarf für BASIC nicht weiter ein, außer natürlich durch die Grafikroutinen selbst (siehe unten).

Version 3 meint die mit den neuen Befehlen geschriebene Demonstration, die als Beispiel für eigene Programmierungen sehr nützlich ist. Dieses Programm ist natürlich nur lauffähig, nachdem eine der beiden Versionen von IMC-BASIC geladen und mit RUN gestartet ist!

Das Grafikpaket IMC-BASIC kann zusammen mit einem eingegebenen oder nachgeladenen BASIC-Programm abgespeichert werden, da es den Anfang des BASIC-RAMs belegt. Wenn Sie Ihr eigenes Programm zusammen mit dem Grafikpaket abspeichern wollen, geben Sie vor dem SAVE-Befehl POKE44,8 ein. Ansonsten reicht der übliche SAVE-Befehl.

Die genaue Speicherbelegung finden Sie am Ende dieses Artikels.

f Ø	f 1
f 2	f 3

Durch den DESIGN-Befehl kann ein beliebiges Muster aus vier quadratisch angeordneten Farbpunkten gebildet werden.

randete Fläche. Gefüllt wird mit dem aktuellen, durch DESIGN festgelegten Muster.

Der FILL-Befehl arbeitet nicht mit der üblichen Methode, die Koordinaten der noch zu füllenden Bereiche auf einem Stack zu speichern, sondern es wird eine neue Grafikseite angelegt. Jeder gesetzte Punkt entspricht dort einer noch zu bearbeitenden Koordinate. Dadurch kann das Programm jede noch so komplexe Fläche füllen, da es zu keinem Stack-Überlauf kommen kann.

Außerdem richtet sich der Befehl noch eine weitere Grafikseite ein, in der er sich merkt, ob der Punkt schon gefüllt wurde oder nicht. Dies ist notwendig, da zwischen dem zu füllenden Bereich und dem Muster unterschieden werden muß.

Beispiel :

```
10 COLDEF 0,1,2,7
20 COLOR 3
30 DESIGN 3,0,0,3
40 CLS
50 MCON
60 FILL 0,0
70 GET A$ : IF A$="" THEN 70
```

Der Bildschirm wird mit folgendem Muster gefüllt :

Gelb	Schwarz
Schwarz	Gelb

Der COLOR-Befehl in Zeile 20 hat keinen Einfluß auf den FILL-Befehl.

FRAME x1,y1,x2,y2

Der FRAME-Befehl zeichnet ein Rechteck, das durch den Punkt links oben (x1/y1) und rechts unten (x2/y2) festgelegt ist. Mit dem FRAME-Befehl kann man zum Beispiel verhindern, daß mehr gefüllt wird als gewünscht.

Dieses Grafikfenster kann mit dem FRAME-Befehl erstellt werden.	Hier kann Text stehen.
Oder etwas anderes.	

So kann der Grafikschirm durch den FRAME-Befehl eingeteilt werden.

Der Text muß natürlich direkt in den Grafikbildschirm geschrieben werden. Und das geht so:

TEXT x,y,Zeichensatz,Breite,Höhe

Aufgabe des TEXT-Befehls ist es, wichtige Informationen an die Grafikroutine zu übergeben.

x und y : Koordinaten, an denen der Text ausgegeben werden soll.

„Zeichensatz" kann die Werte 1 und 2 annehmen :
1 ist der normale Commodore-Zeichensatz,
2 spricht einen speziellen Multicolor-Zeichensatz an (siehe Demo im Magazin).

Breite und Höhe :
Es handelt sich dabei um Faktoren, mit denen die Zeichenbreite und Zeichenhöhe multipliziert werden. Die Werte müssen mindestens 1 sein und dürfen 39 beziehungsweise 79 nicht überschreiten.

IMC-BASIC-Kurzübersicht

BOX x1,y1,x2,y2	Ausgefülltes Rechteck
CIRCLE mx,my,rx,ry	Kreis
CLS [f]	Grafik löschen
COLDEF f0,f1,f2,f3 [,xl1,yl1,xl2,yl2]	Farbauswahl
COLOR f	Aktuelle Punktfarbe (0–3)
DIR [ga]	Directory
DC [b$,[ga]]	Disk-Befehl senden
DRAW x,y	Linie nach x,y
FRAME x1,y1,x2,y2	Rahmen
GLOAD n$,ga	Grafikbild laden
GSAVE n$,ga	Grafikbild speichern
HARDCOPY mo	7-/8-Nadel-Drucker-Hardcopy
LINE x1,y1,x2,y2	Linie ziehen
MCOFF	Grafik aus
MCON	Grafik ein
MERGE n$,ga	Programm einfügen
MLOAD sadr,n$,ga,sa	An Adresse laden
MODE gmode,tmode	Modus für Grafik/Text
MSAVE sadr,eadr,n$,ga	Speicherbereich SAVEn
MULTI [f0,f1,f2,f3 [,xl1,yl1,xl2,yl2]]	MCON/CLS/COLDEF/COLOR
OLD	BASIC-Programm zurückholen
PGET x,y,Var	Punktfarbe holen
PSET x,y	Punkt setzen
SBLOCK nr,adr	Sprite-Startadresse setzen
SCROLL z	Bildschirm scrollen
SPOS nr,xk,yk	Sprite-Position setzen
SPRITE nr,mode,xbr,ybr,prio,color	Sprite einschalten
TEXT [x,y,Satz,Breite,Höhe]	Text-Parameter setzen
ZSAVE sz,ez,n$,ga,sa	Zeilenbereich abspeichern

Um nun auch Zeichen ausgeben zu können, ändert der TEXT-Befehl den BASIC-Befehl PRINT (Vektor $0326). Dabei mußten aber einige Kompromisse eingegangen werden.

a) In der neuen PRINT-Routine fallen einige Steuercodes weg – nur die folgenden sind möglich :

- Die Cursor-Steuerzeichen (aber jetzt auf die eingestellte Breite und Höhe bezogen)
- Das Home-Zeichen
- Revers ein- und ausschalten
- Der Return-Code (chr$(13))

b) Es ist nur der Groß-/Kleinschrift-Zeichensatz erreichbar und

c) es fallen alle Zeichen weg, die mit der Commodore-Taste erreicht werden.

Außerdem wirkt ein geshiftetes Leerzeichen (ASCII-Wert 160) als Endemarkierung. Der PRINT-Befehl erhält entweder durch Rückkehr in den Direktmodus oder durch TEXT ohne Parameter seine ursprüngliche Funktion zurück. Diese Wiederherstellung des Originalzustandes ist zwingend, da sonst keine weiteren Bildschirmausgaben möglich sind!

Beispiel:

```
10 COLDEF 0,1,2,7
20 COLOR 1
30 CLS
40 MCON
50 TEXT 0,0,1,1,1
60 T$="hier steht ein Text"
70 PRINT T$
80 TEXT
90 GET A$ : IF A$="" THEN 90
```

Das Programm schreibt in weißer Schrift auf schwarzem Grund einen Text mit dem Commodore-Zeichensatz (aber doppelt so breit !). Zeile 80 schaltet den TEXT-Befehl wieder ab.

Aus verschiedenen Gründen mußte auf ein automatisches Scrollen beim TEXT-Befehl verzichtet werden. Deshalb gibt es dafür einen besonderen Befehl.

SCROLL z

Dieser Befehl scrollt den Grafikbildschirm ab der angegebenen Zeile z bis zum Schluß um 8 Grafikeinheiten nach oben. Mit z ist eine Zeile zwischen 0 und 23 gemeint. Da eine Zeile 8 Grafikeinheiten dick ist, kann man z auch dadurch berechnen, indem man die Grafikkoordinate y ohne Rest durch 8 teilt. Die unterste Bildschirmzeile bleibt immer erhalten.

Beispiele:
SCROLL 0 scrollt den ganzen Bildschirm, SCROLL 12 nur die untere Hälfte.

BOX x1,y1,x2,y2

Dieser Befehl entspricht dem FRAME-Befehl, nur daß diesmal das Rechteck in der aktuellen Farbe ausgefüllt wird. Mit diesem Befehl kann auch die unterste Zeile nach dem SCROLL-Befehl gelöscht werden:

Beispiel:
```
200 SCROLL 0
210 COLOR 0
220 BOX 0,192,159,199
```

MODE gmode,tmode

Mit MODE können zusätzlich die Grafikausgaben geregelt werden. Die Voreinstellung

Grafik auch in Assembler

Das Programm besteht aus zwei Teilen: der BASIC-Erweiterung und den Grafikroutinen. Die Grafikroutinen liegen ab $1700 im Speicher. Sie sind völlig unabhängig vom anderen Teil lauffähig. Am Anfang befindet sich eine umfangreiche Einsprungtabelle, die auch das Ansprechen der Grafikroutinen aus Assembler erleichtert:

Vor dem Aufruf einer Zeichenroutine muß in der Speicherzelle 2 der Zero-Page der Pixel-Code (also die Farbe) übergeben werden.

Speicherzelle 2 : %0000XX00
ΔΔ
hier steht der Code !

Adresse	Funktion	Parameter
$1700	Grafik ein	–
$1703	Grafik aus	–
$1706	Bild löschen	Akku: Byte, mit dem gefüllt wird
$1709	Farben setzen	Farben 1 und 2 im Akku, Farbe 3 im X-Reg.
$170C	Punkt setzen	$FD=xk, $FE=yk, $02=Pixel-Code*4
$170F	Punkt holen	$FD=xk, $FE=yk, –> Akku : PixelCode
$1712	Linie ziehen	$33C=x1, $33D=y1, $33E=x2, $33F=y2
$1715	Bereich füllen	$FD=xk, $FE=yk, 4-Bytes-Muster ab $1727
$1718	Kreis zeichnen	$33C=mx, $33D=my, $33E=rx, $33F=ry
$171B	Multicol.-Text	$FD=xk, $FE=yk, Akku: Zeichen, X-Reg: Breite, Y-Reg: Höhe (gibt nur ein Zeichen aus)
$171E	Normaler Text	wie bei Multicolor-Text
$1721	Scrollen	Akku : Startzeile
$1724	Modus	Akku : gmode, X-Reg : tmode
wichtige Speicherzellen ↓		
$1727 bis $172A		Muster,mit dem gefüllt wird. Jedes Byte enthält einen Pixel-Code*4 (Bits 2 u. 3)
$172B	Grafikseite	enthält das Hi-Byte der Anfangsadresse des Grafikbildschirms
$172C	Video-RAM	neues Video-RAM der Grafik
$172D	Farb-RAM	genauer: Zwischenspeicher des Farb-RAMs

ist MODE 0,0. Dennoch sollte am Beginn eines Programms der MODE-Befehl stehen.

gmode = 0 ist der normale Modus zum Zeichnen von Grafikpunkten.

gmode = 1
setzt nicht die Farben, sondern invertiert sie (aus Farbe 0 wird Farbe 3, aus Farbe 1 wird Farbe 2 und vice versa).

tmode ist nur für Textausgaben zuständig:
tmode = 0 Die Buchstaben werden komplett in den Grafikbildschirm geschrieben; zum Beispiel löscht das Leerzeichen den darunter liegenden Bereich.

tmode = 1
Nur die gesetzten Bits der Buchstaben werden gezeichnet. Das Leerzeichen hätte in diesem Fall keinerlei Auswirkungen.

Für alle: Die Tool-Befehle

GLOAD n$,ga
und
GSAVE n$,ga

funktionieren wie LOAD oder SAVE, nur wird nicht das Programm, sondern die Grafikseite abgespeichert. „n$" steht für den File-Namen, „ga" für die Geräteadresse.

DIR [ga]

zeigt das Directory der Diskette in der Floppy mit der Geräteadresse ga an. Das Auflisten des Directory kann mit SPACE angehalten und mit STOP abgebrochen werden.

DC ["Diskbefehl" [,ga]] (DiskCommand)

sendet einen Befehl an das Laufwerk mit der Geräteadresse ga und liest den Fehlerkanal aus.

OLD

Wurde ein BASIC-Programm mit NEW gelöscht, kann es mit OLD wiederhergestellt werden.

MERGE n$,ga

Parameter wie LOAD, nur daß das Programm in ein bereits vorhandenes eingefügt wird. Bereits existierende Zeilennummern werden überschrieben, ansonsten ist beliebiges „Mischen" möglich!

ZSAVE sz,ez,na$,ga

Speichert die BASIC-Zeilen ab der Zeilennummer sz bis zur Zeilennummer ez ab. In Verbindung mit dem MERGE-Befehl wird hierdurch die modulare Programmentwicklung komfortabel gemacht.

Speicherbelegung und Zero-Page-Benutzung

Version 1

$0000–$07FF	–
$0800–$16FF	Neue BASIC-Routinen
$1700–$26FF	Grafikroutinen
$2780–$297F	Die ersten acht Sprite-Blöcke
$2A00–$2DFF	Zwischengespeichertes Farb-RAM
$2E00–$79FF	Neuer BASIC-Speicher
$7A00–$89FF	Speicher für FILL (Teil 1)
$8A00–$8BFF	Die zweiten acht Sprite-Blöcke
$8C00–$8FFF	Video-RAM für die Grafik
$9000–$9FFF	Speicher für FILL (Teil 2)
$A000–$BFFF	Grafikbildschirm
$C000–$FFFF	–

Version 2

$0000–$07FF	–
$0800–$16FF	Neue BASIC-Routinen
$1700–$26FF	Grafikroutinen
$2780–$297F	Die ersten acht Sprite-Blöcke
$2A00–$2DFF	Zwischengespeichertes Farb-RAM
$2E00–$9FFF	Neuer BASIC-Speicher
$A000–$BFFF	BASIC-ROM / Speicher für FILL u. GSAVE/LOAD
$C000–$C9FF	–
$CA00–$CBFF	Die zweiten acht Sprite-Blöcke
$CC00–$CFFF	Video-RAM für die Grafik
$D000–$DFFF	–
$E000–$FFFF	Grafikbildschirm

Version 1 und 2

Benutzte Vektoren

$0300	Warmstartvektor (um die Grafik abzuschalten)
$0304	Umwandlung in Interpretercode
$0306	Umwandlung in Klartext
$0308	Neue Befehle ausführen
$0326	Output-Vektor für den TEXT-Befehl
$0328	STOP-Vektor

Zero-Page-Adressen

$01	ROMs ausblenden
$02	Farbe übergeben (s.o.)
$22 / $23	Zeiger auf den Zeichensatz
$24 / $25	dx und dy für Line-Routine
$AC – $AE	Diverse Zwischenspeicher
$FD	x-Koordinate
$FE	y-Koordinate
$FB / $FC	Zeiger auf die Grafikseite

MSAVE sadr,eadr,na$,ga

Speichert den Speicherbereich von sadr (Startadresse) bis eadr–1 (Endadresse–1) ab; weitere Parameter siehe LOAD.

MLOAD sadr,na$,ga,sa

Parameter siehe MSAVE. Wenn sa gleich Null ist, wird ein Maschinenprogramm an die angegebene Adresse (sadr) geladen. Ist sa ungleich Null, wird das Programm an die Orginaladresse geladen.

Für mehr Bewegung: Die Sprite-Befehle

Die Sprite-Befehle sind notwendig geworden, da ein problemloses Ansprechen der Sprites im Grafikmodus nicht mehr möglich ist. Das Programm richtet zwei mal acht Sprite-Blöcke ein und verwaltet diese selber.

Mit diesen Sprite-Befehlen ist man nicht mehr durch den eingeschränkten VIC-Adressierungsbereich behindert. Es wird einfach die Startadresse des Sprites, dessen Definitions-Muster an einer beliebigen Stelle im RAM steht, übergeben. Das Programm kopiert die Sprites dann an die richtige Stelle.

SBLOCK nr,adresse

Teilt dem Programm mit, daß das Sprite mit der Nummer nr (0 bis 7) an der angegebenen Adresse zu finden ist.

Beispiel:
SBLOCK 0,704

entspricht dem sogenannten Sprite-Block 11 (11 * 64 = 704). Man ist in der Wahl der Adresse übrigens nicht mehr auf die 64-Byte-Schritte eingeschränkt.

SPRITE nr,mo,x,y,priorität,farbe

Schaltet das Sprite mit der Nummer nr ein.

mo = 0 Einfarbiges Sprite
mo = 1 Multicolor Sprite

farbe 0–15

x Vergrößerung in x-Richtung

y Vergrößerung in y-Richtung

priorität Sprite vor dem Text sichtbar oder nicht
Anmerkung: bei Multicolor-Sprites müssen die gemeinsamen Farben aller Sprites wie bislang durch Beschreiben der entsprechenden VIC-Register gesetzt werden.

SPOS nr,x,y

Sprite Nummer nr soll an den Koordinaten x und y dargestellt werden. x und y sind in diesem Fall die üblichen Sprite-Koordinaten, nicht die Grafik-Koordinaten.

Beispiel :

```
10 SPRITE 0,0,1,1,0,15
20 SPOS 0,100,100
30 FOR I=0 to 65535 STEP 3
40 SBLOCK 0,I
50 NEXT I
```

Das Programm stellt ein vergrößertes Sprite in der Farbe Hellgrau an den Koordinaten 100,100 dar. Anschließend wird der komplette Speicherinhalt durch das Sprite gescrollt.

Cold and old

Sollte aus irgendeinem Grund der C64 abstürzen und ein Hardware-Reset notwendig werden, so läßt sich das Programm mit einem SYS 2128 neu starten. Ein BASIC-Programm kann dann mit OLD wieder sichtbar gemacht werden. Oliver Kraus/JS

Fauna-Kenntnisse gefragt

Spiel: Tiere-Raten

Lehrreich für Kinder, unterhaltsam für Kindgebliebene: Testen Sie Ihre Kenntnisse in Sachen Tierwelt — über Springmäuse, Feldhasen und Pottwale.

Vom „Anfänger" bis zum „As" kann der interessierte Tierfreund in diesem Spiel aufsteigen. Es geht darum, anhand vorgegebener Merkmale eines von 18 Tieren zu erraten. Das Programm gibt Eigenschaften vor, etwa: „Es liebt Wälder mit dichtem Unterholz" oder: „Es frißt Schnecken und Frösche"; der Spieler gibt als Antwort den Namen des vermuteten Tieres ein.

Bei falschen Antworten oder Leereingabe (nur RETURN) wird ein weiterer Hinweis gegeben. Bis zu sieben Mal versucht das Programm, einem auf die Sprünge zu helfen. Je eher das Tier erraten wird, desto mehr Punkte erhält man.

Das Programm bietet umfangreiche Hilfestellungen an, um auch kleineren Kindern Erfolgschancen zu geben. So kann man sich zum Programmanfang die Beschreibungen aller Tiere zeigen lassen („Willst Du eins kennlernen?" mit 'J' beantworten), verschiedene Schwierigkeitsstufen wählen und testweise ohne Bewertung (ohne Punkte) spielen.

„Tiere-Raten" kann durch CTRL-S auf eine eigene Diskette überspielt werden, ist aber auch innerhalb von INPUT 64 uneingeschränkt lauffähig.

BASIC-Programmierer können das Spiel mit neuen Tier- (oder auch Pflanzen-) Daten versehen. Alle Tiere und die dazugehörigen Merkmale sind in den DATA-Statements ab Zeile 2100 abgelegt. Falls die Anzahl der Tiere oder der Eigenschaften geändert werden soll, muß die Dimensionierung des String-Arrays T$ in Zeile 53 und die READ-Schleife ab Zeile 2000 angepaßt werden.

Da in das Programm die Tools PrintAt und Inline zur Ein-/Ausgabesteuerung eingebunden sind, ist bei Änderungen im BASIC-Teil folgendes zu beachten:

Programm mit RUN starten und mit RUN/STOP abbrechen. Dann ist das BASIC-Listing zugänglich. Vor dem Abspeichern im Direktmodus POKE44,8 eingeben, um die am BASIC-Anfang liegenden Tools mit abzuspeichern. JS

Tips zur Assemblerprogrammierung

Assembler als Hochsprache

Teil 3: Makros und bedingte Assemblierung

Assembler ist, im Gegensatz zur in der Überschrift aufgestellten Behauptung, natürlich keine Hochsprache. Im Gegenteil: Assembler ist heutzutage die maschinennahste Form der Programmierung, die dem Benutzer eines Computers zur Verfügung steht. Deswegen ist es auch in der Regel so, daß zur Nachbildung eines einzigen BASIC-Befehls eine ganze Folge von Maschinensprachebefehlen nötig ist; und jedes größere Maschinenspracheprogramm besteht aus sehr, sehr häufigen Wiederholungen ähnlicher Befehlssequenzen.

Parallel zum Maschinensprachekurs begann diese Artikelfolge, die Anfängern und Fortgeschrittenen Hilfen zur Erstellung überschaubarer Assemblerprogramme gibt. In der dritten und vorerst letzten Folge dieser Serie geht es darum, der Software nicht nur stupide Tipparbeit, sondern auch die Anpassung eines Programms an verschiedene Bedingungen zu überlassen.

Makros: Arbeit sparen . . .

Ein relativ triviales Beispiel soll diese Behauptung verdeutlichen: die Ausgabe eines Strings auf dem Bildschirm. In BASIC ist so etwas sehr unkompliziert zu erledigen: PRINT"TEXT", und das Wort „TEXT" erscheint auf dem Monitor. In Assembler ist das um einiges aufwendiger. Um das Problem nicht unnötig zu verkomplizieren, gehen wir einmal davon aus, daß das Betriebssystem eine Routine zur Textausgabe enthält. Beim C64 gilt dafür die Vereinbarung, daß der String maximal 255 Zeichen lang sein darf, mit einem Null-Byte abgeschlossen sein und die Adresse im Akku (Low-Byte) und Y-Register (High-Byte) übergeben werden muß.

Weiter angenommen, der Text sei unter dem Label TEXT abgelegt, sähe die entsprechende Routine so aus[1]:

```
LDA # < TEXT
LDY # > TEXT
JSR $AB1E
```

Das ist wesentlich umständlicher als die BASIC-Lösung, und es soll schon mancher Programmierer durch die Verwechslung von Low- und High-Byte an den Rand der Verzweiflung geraten sein.

Hier helfen Makros weiter. Es ist nämlich auch in Assembler erlaubt, PRINT TEXT zu sagen. Vorausgesetzt, wir haben vorher unsere Wünsche mitgeteilt, indem wir ein Makro definiert haben. Eine Makro-Definition besteht aus:

- dem Namen des Makros
- einem Befehl, der angibt: Dies ist eine Makro-Definition
- eventuell einer Angabe über die Anzahl der Parameter, die überge ben werden sollen
- dem sogenannten „Rumpf" des Makros, also der dem Makro zuzu ordnenden Befehlsfolge
- und einem Endekennzeichen.

Für den INPUT-ASS sähe das PRINT-Makro folgendermaßen aus:

```
:PRINT M 1
LDA #< @0
LDY #> @0
JSR $AB1E
/
```

```
;makro fuer zwei-byte
;integer-vergleich
;mit 500
:bge mf 2
lda @0+1
cmp #>500
bcc no
bne jp
lda @0
cmp #<500
bcc no
:jp jsr @1
:no nop
/
```

So sieht eine Makro-Definition für einen Zwei-Byte-Intreger-Vergleich aus.

Nach dem Namen folgt das M als Makro-Kennung; der eine zu übergebende Parameter ist die Adresse (nicht etwa dieser Text selbst!) eines Textes, also das Label TEXT, in der Form :TEXT B "TEXT",0

```
:makro fuer zwei-byte
;integer-vergleich
;mit konstante
:vgl mf 3
lda @0+1
cmp #>@2
bcc no
bne jp
lda @0
cmp #<@2
bcc no
:jp jsr @1
:no nop
/
```

Noch einmal der Zwei-Byte-Integer-Vergleich; so kann auch der Fest-wert übernommen werden.

Der „Klammeraffe" (@) mit der nachfolgenden Null gibt an, daß der erste übergebene Parameter statt dessen eingesetzt werden soll. (Die Numerierung der Parameter beginnt computertypisch bei Null.)

Durch diese Definition ist dem Assembler „bekannt", was bei einem Ausdruck wie PRINT TEXT zu tun ist, nämlich so tun, als ob dort LDA
< TEXT und so weiter stände.

. . . und Fehler vermeiden

Makros sind übrigens kein Ersatz für Unterprogramme! Bei jedem Makro-Aufruf wird der vollständige Code, der der Definition entspricht, erzeugt.

Ein weniger triviales Beispiel: abhängig von der Größe einer Integer-Variablen soll im Programm verzweigt werden. In BASIC wird dies durch
IF A% > = 500 THEN GOSUB . . .
erledigt. In 6502-Assembler stellt sich das Problem, Low- und High-Byte der Integer-Variablen getrennt abfragen zu müssen. Zudem muß der Vergleichswert in eines der Prozessor-Register geladen werden, um überhaupt vergleichen zu können – natürlich ebenfalls getrennt nach Low- und High-Byte. Dies sähe, bei der Annahme, die Variable A% entspräche einer durch das Label A dargestellten Adresse, etwa folgendermaßen aus[1]:

```
LDA A+1
CMP # > 500
BCC NICHT
BNE JUMP
LDA A
CMP # < 500
BCC NICHT
:JUMP JSR FERTIG
:NICHT . . .
```

Da solche Vergleiche relativ häufig sind, bedeutet dies bei längeren Programmen einiges an Tipparbeit und, das ist vielleicht noch wichtiger, eine immer wiederkehrende Fehlerquelle.

Wie das entsprechende Makro beschaffen sein muß, zeigt Listing 1. Es müssen zwei Variablen verarbeitet werden: die symbolische Adresse der zu prüfenden Variablen und die des Unterprogrammes, beispielsweise A und FERTIG. Aufgerufen wird dieses Makro dann durch

```
BGE A,FERTIG
```

Die Makro-Kennung MF bedeutet, daß es sich um ein Makro mit einer sogenannten Vorwärtsreferenz handelt, hier nämlich NO und JP. Der NOP-Befehl nach dem NO kann entfallen, wenn dem Makro-Aufruf noch mindesten ein Befehl folgt. Naheliegend ist natürlich, statt eines festen Vergleichs mit 500 auch diesen Wert übergeben zu können; wie das geht, zeigt Listing 2.

Weitere Beispiele finden Sie in den Listing-Auszügen, als Anregung für die Lösung eigener Programmierprobleme.

Differenzierte Bedingungen . . .

Bei Assemblerprogrammen, die nicht direkt auf spezifische Port-Bausteine und andere Rechner-Innereien zugreifen, spräche eigentlich nichts dagegen, diese sowohl für den C64 als auch den VC20 (den viele C64-Besitzer ja noch zu Hause stehen haben sollen) zu entwickeln. Unterschiedlich wären natürlich die Systemaufrufe für Tastatureingabe, Textausgabe und so weiter.

```
;incre: makro zum
;inkrementieren
;eines pointers
:incre mf 1
inc @0
bne nh
inc @0+1
:nh
/


;decre: makro zum
;dekrementieren
;eines pointers
:decre m 1
sec
lda @0
sbc #01
sta @0
lda @0+1
sbc #00
sta @0+1
/
```

Einmal definiert, immer zur Verfügung: Dekrementieren und Inkrementiern eines Zwei-Byte-Zeigers.

```
;if then else macro
;parameter muessen sein:
;@0 : adresse (variable)
;@1 : 1. unterprogramm
;@2 : dummy fuer else
;@3 : 2. unterprogramm
;als 'wahr' wird ein
;wert <> 0 angenommen
;beispiel:
;falls rmode,lies,else,schreib
;
:falls mf 4
lda @0
beq fl1
jsr @1
jmp fl2
:fl1 jsr @3
:fl2
/
```

Zum Expermentieren und Weiterentwickeln: IF-THEN-ELSE in Assembler.

Für solche Fälle ermöglicht die „Bedingte Assemblierung", alle Programmversionen in einem Source-Text unterzubringen. Die Syntax der entsprechenden Assembler-Anweisung lautet sinngemäß

```
IF bedingung wahr
assembliere dies
ELSE
assembliere etwas anderes
ENDE
```

Das ELSE ist optional, die Konstruktion verkürzt sich dann auf

```
IF bedingung wahr
assembliere
ENDE
```

Damit ist folgendes gemeint: ist die „Bedingung" wahr (ungleich Null), dann soll der anschließend folgende Text assembliert werden; ist die Bedingung falsch, soll der Teil nach dem ELSE beziehungsweise (in der verkürzten Form) nichts assembliert werden.

Der wenig geschwätzige INPUT-ASS erwartet das ELSE nicht ausgeschrieben, sondern nur EL, das Endekennzeichen ist ein EI. Assoziationen zu Hühnern oder Ostern sind völlig verfehlt, EI steht für Endlf. Beispiel:

```
IF C64
:STROUT = $AB1E
EL
:STROUT = $CB1E
EI
```

Ist die Variable C64 ungleich Null, wird STROUT der Wert $AB1E zugewiesen, sonst $CB1E. Diese Variable C64 muß natürlich vorher gesetzt werden, durch

```
:C64 = 0
```

oder

```
:C64 = 1
```

. . . für verschiedene Rechner

Statt nur einer Anweisung kann natürlich zwischen dem IF und dem EI beliebig viel Text stehen, etwa ganze Label-Tabellen oder spezielle Programmteile. Die IF-Konstruktionen können auch verschachtelt werden.

```
;macro WHILE
;syntax:
;while,expression,operation
;weiteres siehe FALLS
:while mf 2
:lpwhile lda @0
beq notwhile
jsr @1
jmp lpwhile
:notwhile
/
```

Sogar eine dem C64-BASIC unbekannte Struktur wie WHILE-DO kann durch Makros nachgebildet werden.

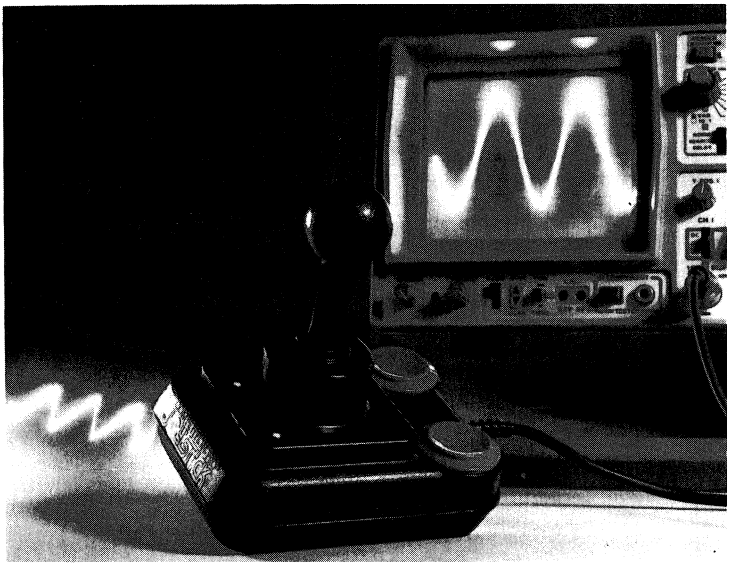
Durch die Möglichkeit der bedingten Assemblierung lassen sich Konstruktionen programmieren, die den Assembler – gewollt oder ungewollt – überlisten. So wird durch folgenden Programmausschnitt erreicht, daß das Source-File PROGRAM2.S nur im ersten Assembler-Pass, aber nicht mehr im zweiten Pass „included" (gleich: von Diskette eingelesen) wird:

```
:FLAG := 1

IF FLAG
IN 82PROGRAM2.S
FLAG := 0
EI
```

Die Kombination von bedingter Assemblierung und Makro-Programmierung gehört zu den Leckerbissen der Assemblerprogrammierung. Sie sollte allerdings nur nach reiflicher Überlegung angewandt werden.JS

[1] Die Assembler-Direktiven wie „<" und „>" gelten, wie auch die in den Listings verwendete Syntax, für den in Ausgabe 6/86 veröffentlichten Makro-Assembler INPUT-ASS. „<" steht für das Low-, „>" für das High-Byte eines nachfolgenden Wertes.



Graph von Zahl

Grafische Darstellungen mit INPUT-Graph

Bei Verfahren wie wissenschaftlichen Messungen, statistischen Auswertungen oder kaufmännischen Berechnungen fallen häufig größere Mengen an Daten an. Dies gilt vor allem für Samples (digitalisierte Klängen), bei denen man in der unüberschaubaren Datenflut die Klangcharakteristik kaum wiedererkennen kann. Für solche Fälle wünscht man sich ein Programm, das solches Zahlenmaterial grafisch auf dem Bildschirm darstellt und diese Grafik auch zu Papier bringt. Denn erst aus der grafischen Anschauung werden die Aussagen, die sich hinter dem Zahlenmaterial verbergen, erkennbar. INPUT-Graph wurde direkt für solche Anwendungen zugeschnitten und bietet eine Vielzahl von besonderen Leistungsmerkmalen:

- flexibles Datenformat
- grafische Bearbeitung (INPUT-CAD kompatibel)
- mathematische Analyse

Wo Daten erfaßt werden, sollte es eigentlich um Aussagen gehen. Doch meist stiften Zahlenwüsten eher Verwirrung und sorgen für Abschreckung. Erst wo sich Werte in Kurven darstellen, werden Tendenzen und Charakteristika deutlich. Mit INPUT-Graph können Sie Datenmaterial grafisch darstellen, analysieren und verschiedenen Operationen unterwerfen. Letztlich halten Sie Vorzeigbares schwarz auf weiß in der Hand.

INPUT-Graph kann innerhalb des INPUT 64-Betriebssystems benutzt werden. Der Bereich ab 49 152 ($C000) bleibt frei. Haben Sie sich das Programm mit CTRL-S auf eigenen Datenträger abgespeichert, können Sie Hilfsprogramme, wie zum Beispiel SuperDisk, die in diesem RAM-Bereich liegen, mit INPUT-Graph zusammen einsetzen.

Da INPUT-Graph kein spezielles Datenformat für das Zahlenmaterial verlangt, haben Sie die Möglichkeit, Daten von anderen Programmen zu übernehmen, selbst zu erzeugen oder auch von Hand einzugeben. Sie können sequentielle Dateien, Datenbereiche oder auch Bytes aus Speicherbereichen (bei digitalisierten Klängen beispielsweise) vom Datenträger einlesen. INPUT-Graph kann maximal 1024 Werte bearbeiten. Mehr Speicherplatz steht dem Programm nicht zur Verfügung. Diese Daten lassen sich dann direkt grafisch darstellen, wobei INPUT-Graph selbständig die Skalierung der Koordinaten übernimmt und Sie über Maximal- und Minimalwerte informiert. Doch was es dort zu sehen gibt, ist oft noch nicht zufriedenstellend. Der Autor von INPUT-Graph hat aufgrund eigener Erfahrungen nahezu alle Möglichkeiten vorgesehen, das Datenmaterial zu überarbeiten und zu analysieren. Sie können sowohl bei der angezeigten Grafik als auch am Datenmaterial direkt Änderungen vornehmen. Ein umfangreicher Formelapparat hilft Ihnen bei der Analyse und Synthese von Kurvenverläufen.

Die Einstellung . . .

So stehen Ihnen von einer umfassenden Analyse bis hin zur gezielten Manipulation der Daten alle Wege offen. Die Ergebnisse gibt INPUT-Graph auf dem Bildschirm, als Hardcopy auf dem Drucker oder als Bitmap auf Datenträger aus. Diese Bitmap können Sie mit anderen Grafik- oder Malprogrammen weiterbearbeiten. Die eigentliche Leistungsfähigkeit der Grafikausgabe liegt jedoch in der Schnittstelle zu INPUT-CAD (Ausgabe 11/86 – 2/87), wodurch die Grafiken auf allen daran angepaßten Druckern in einer Auflösung von bis zu 1024*1024 Punkten ausgegeben werden können. Zudem können Sie so auf den gesamten leistungsfähige Editor von INPUT-CAD zur professionellen Aufbereitung der Grafiken zurückgreifen, was auch das Erstellen von

repräsentativen Vorlagen ermöglicht, beispielsweise durch nachträgliche Beschriftung und/oder grafische Gestaltung.

Natürlich können die Daten auch auf Datenträger ausgegeben werden, um sie entweder in andere Programme zu übernehmen oder spezielle Fälle zu dokumentieren. Selbst die oben als unübersichtlich bezeichnete Wertetabelle kann auf dem Drukker ausgegeben werden, damit die Palette vollständig ist.

. . . zu der Sache . . .

Wahrscheinlich haben Sie INPUT-Graph gestartet und Verschiedenes ausprobiert. Erst als es Ihnen trotz größter Anstrengungen nicht gelungen ist, ein Bild auf den Monitor zu bekommen, haben Sie zur Anleitung gegriffen. INPUT-Graph ist ein sehr mächtiges Grafikwerkzeug, das in seiner Bedienung entsprechend komplex ist. Bis Ihnen die Anwendung flott von der Hand geht, brauchen Sie etwas Einarbeitungszeit, die sich aber lohnt. Als Kompromiß aus Arbeitsgeschwindigkeit und Bedienerfreundlichkeit ist das Programm mit einer Menüoberfläche ausgestattet, die aber natürlich nicht alle nötigen Erklärungen bieten kann. Teilweise brauchen die Menüpunkte diverse Voreinstellungen, ohne die die gewünschte Operation nicht ausgeführt wird. Da das Programm mit Speicherplatzproblemen zu kämpfen hat, gibt es in solchen Fällen keine Fehlermeldungen, es passiert nur nichts. Diese Methode der Voreinstellungen beschleunigt den Umgang mit dem Programm erheblich, wenn man sich eingearbeitet hat.

Beim Programmstart erscheint auf dem Bildschirm eine Hand, die den CURSOR-Tasten oder einem Joystick in Port 2 folgt. Mit dem „Zeigefinger" wird der gewünschte Fall ausgewählt, RETURN oder Feuerknopf führt die Aktion aus. Dabei kommt es nur auf die ausgewählte Zeile an, die Spaltenposition spielt keine Rolle. Die übrige Tastatur wird nur benutzt, um Zahlen oder Texte (bei File-Namen beispielweise) einzugeben.

Fast überall, wo zwischen den Menüpunkten noch eine Zeile frei ist, ist durch Anwählen derselben ein Taschenrechner zu erreichen, welcher beliebige Rechnungen durchführen kann. Die Eingabe muß der gewohnten BASIC-Syntax entsprechen.

Wollen Sie den weiteren Erläuterungen folgen, starten Sie am besten INPUT-Graph, das Sie vorher mit CTRL-S auf eigenen Datenträger übernommen haben. Sie begreifen INPUT-Graph am schnellsten, wenn Sie damit umgehen.

. . . mit den Feldern

INPUT-Graph kann maximal 1024 Werte auf einmal bearbeiten oder darstellen. Diese 1024 Werte sind in einem BASIC-Realfeld untergebracht. Die Zahlen, die dargestellt werden sollen, werden nacheinander in den Feldelementen abgelegt: f(n)=z. Der Name des Feldes ist dabei „f" , das „n" in Klammern die Nummer des Feldelementes, „z" die jeweilige Zahl.

Ein Bereich dieses Feldes kann durch Angabe des ersten und letzten Feldelementes festgelegt werden. Beispiel: f(0), f(128). Ein solcher zusammenhängender Bereich ist wieder ein Feld. Die beiden Eckelemente sollen im folgenden „Feldgrenzen" heißen.

Wenn Sie kein Programmierer sind und sich unter einem Feld nichts vorstellen können, denken Sie einfach an 1024 nebeneinander aufgestellte Schachteln, die von 0 bis 1023 durchnummeriert sind. In jede dieser Schachteln paßt genau eine Zahl. Diese 1024 Schachteln bekommen nun den Namen „Feld", und die einzelne Schachtel heißt „Feldelement".

Auch unser Beispiel stellt einen solchen Zusammenhang her. Es entstehen so die Wertepaare (Montag, 50 DM), (Dienstag, 55 DM), (Mittwoch, 57 DM), (Donnerstag, 54 DM) und (Freitag, 49 DM). Waagerecht, im x-Feld, stehen also die 5 Wochentage, senkrecht, im y-Feld, die Preise.

Bevor eine Grafik auf den Bildschirm kommt, muß noch geklärt werden, wie die Darstellung erfolgen soll.

In Mathematik und Technik hat sich zur Darstellung von Funktionen und funktionalen Zusammenhängen das zweidimensionale Koordinatensystem durchgesetzt. Allgemein hat eine Koordinate die Form (x,y). Eine Grafik entsteht dadurch, daß die x-Werte waagerecht und die y-Werte senkrecht aufgetragen und durch Linien verbunden werden. Das Teilfeld, das die y-Werte enthält, heißt im folgenden y-Feld. Die x-Werte können einfach aufwärts gezählt oder einem zweiten Teilfeld, dem x-Feld, entnommen werden.

Damit alle Arten von Daten verarbeitet werden können, nimmt INPUT-Graph nur Zahlenwerte an. Für das Beispiel bedeutet dieses, daß die Wochentage von 0 bis 4 durchnummeriert werden und die Einheit DM bei den Preisen einfach wegfällt. Das Ergebnis sehen Sie in Bild 1.

Welt und Bild

Wenn Sie in die 64er Tips in dieser Ausgabe geschaut haben, wissen Sie, daß der Grafikbildschirm des C64 eine Auflösung von 320 Punkten in x-Richtung und 200 Punkten in y-Richtung hat. Da sich die Kurve über den ganzen Bereich erstrecken soll, muß hier etwas gerechnet werden. Die originalen x- und y-Werte heißen „Weltkoordinaten", die umgerechneten heißen „Bildkoordinaten" und die Umrechnung selbst ist die „Welt-Bild-Transformation". Die Grenzen dieser Koordinaten bilden einen Rahmen, so daß man auch von einem Welt- oder Bildfenster sprechen kann. Beide Fen-

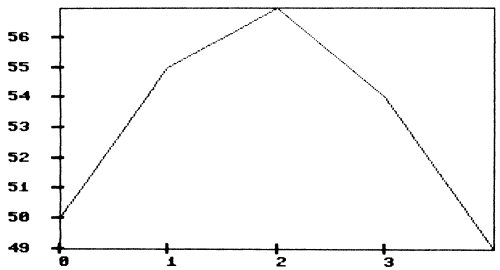


Bild 1:
Von Punkt zu Punkt

ster sind in INPUT-Graph innerhalb der sinnvollen Grenzen frei wählbar.

Nachdem alle Fachbegriffe geklärt sind, können die einzelnen Programmteile der Reihe nach erläutert werden. Wollen Sie zuerst ein Bild auf dem Monitor sehen, gehen Sie nach dem Beispiel im Kastentext vor.

Der Menüpunkt **Feldgrenzen festlegen** ist an mehreren Stellen des Programms zu erreichen, da er häufig gebraucht wird, denn hier wird das y-Feld festgelegt. Liegt zum Beispiel der y-Anfang bei Element 500, das Ende bei 600, so wird (bis auf drei Ausnahmen) nur dieser Bereich von allen Operationen verwendet, sei es bei Berechnungen oder bei der grafischen Darstellung. Nimmt y-Anfang einen anderen Wert an, bleibt die Feldlänge (hier 100 Elemente) erhalten, das heißt, das y-Ende verschiebt sich mit. So lassen sich schnell Datensätze austauschen. Werden mehr oder weniger Werte bearbeitet, muß y-Ende einen neuen Wert annehmen. Der y-Anfang verschiebt sich in diesem Fall nicht.

Für die x-Werte gibt es, wie gesagt, zwei Möglichkeiten: Sie können vom Startwert an mit der Schrittweite wie in einer Schleife gezählt werden oder es kann ein x-Feld definiert werden, das automatisch immer die Länge des y-Feldes hat. Damit sind nicht nur Funktionen sondern auch Relationen sowie Ortskurven darstellbar.

Wie in jedem Untermenü führt QUIT wieder in die nächsthöhere Ebene.

Eingebungen

Der Punkt **Zahlenwerte editieren** bietet die Möglichkeit, die Zahlenwerte per Tastatur einzugeben oder zu ändern. Je nach obiger Einstellung werden x- und y-Werte abwechselnd oder nur die y-Werte eingegeben. Wenn Sie für das Beispiel y-Anfang auf Null und y-Ende auf Fünf gestellt haben, dann können Sie hier nacheinander die fünf Werte eintippen. F5 druckt jederzeit eine Wertetabelle aus, wenn ein Drukker angeschlossen ist, F3 zaubert den Taschenrechner herbei und F1 beendet die ganze Aktion. Mit F7 können Sie innerhalb des Feldes bestimmte Feldelmente anspringen. Sie brauchen nur die Nummer des Feldelementes anzugeben.

Der erste Punkt im Hauptmenü lautet vielversprechend **Grafik**. Hierüber können Sie die Werte eines ausgewählten y-Feldes der Reihe nach grafisch dargestellen. Sie müssen dazu unter **Feldgrenzen setzen** „x-Daten zählen" eingestellt haben. Entscheiden Sie sich für „x-Daten aus Feld", werden die Punkte der Kurve aus den Wertepaaren (x/y) gebildet. Damit lassen sich Ortskurven oder Lissajous-Figuren erzeugen. Im Grafik-Menü kann auch die Skalierungen der x-/y-Achsen der Grafik sowie deren Länge festgelegt werden. Ist die Weltfensterbreite ein ganzzahliges Vielfaches dieser Anzahl, so sind die Skalierungsstufen ebenfalls ganzzahlig. Werden beide Zahlen auf Null gesetzt, wird nur der Graph gezeichnet, was ganz nützlich ist, wenn mehrere Kurven überlagert werden sollen.

Es stehen vier Ausgabemodi zur Verfügung:

Bildschirm, Linien

Die Koordinatenpunkte werden der Reihe nach durch Linien verbunden, das ergibt einen durchgezogenen Graphen.

INPUT CAD

Ausgabe einer Gesamtdatei für das von 11/86 bis 2/87 veröffentlichte Konstruktionsprogramm. Da in INPUT-Graph der Koordinatenursprung links unten liegt, müssen Sie in INPUT CAD links unten (0,1023) den Beginn der Grafik suchen.

INPUT CAD, 90 Grad gedreht

Die Grafik liegt hier nicht mehr aufrecht in INPUT CAD, sondern um 90 Grad nach rechts gedreht. Der Ursprung liegt jetzt tatsächlich in (0,0) und die x- und y-Achse sind sozusagen vertauscht. Das hat den Sinn, auch auf Druckern wie etwa dem MPS801, die nur 480 Punkte in der horizontal, jedoch beliebig viele Punkte vertikal

Der Weg zum Bild

- INPUT-Graph starten
- **Feldgrenzen festlegen** anwählen
- y-Ende auf 5 setzen
- mit **QUIT** zurück zum Hauptmenü
- **Zahlenwerte editieren** anwählen
- nacheinander die Zahlen 50, 55, 57, 54 und 49 eingeben
- mit F1 zurück ins Hauptmenü
- **Grafik** anwählen
- **Bildschirm, Linien** anwählen
- mit F3 die Weltfenstergrenzen setzen
- RETURN wechselt in die Spalte Bildfenster
- mit F3 die Bildfenstergrenzen setzen
- auf den Grafikbildschirm mit F1 umschalten
- mit RETURN oder FEUER die Funktion **neu Zeichnen** anwählen

Beispiel: Zahlen ins Bild

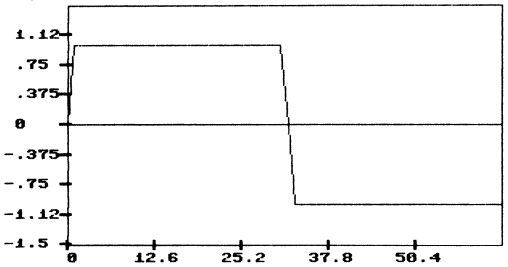


Bild 2: Eine Periode . . . .

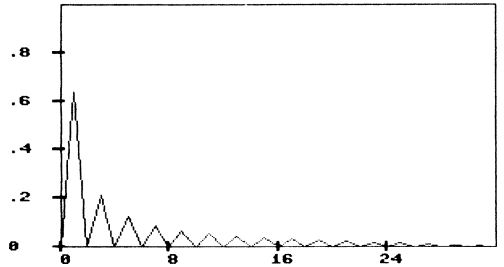


Bild 3: . . . und das Spektrum

drucken können, die Längsauflösung von 1024 Punkten ausnutzen zu können.

Da bei der Berechnung der Datei der Grafikspeicher benutzt wird, ist eine eventuell dort vorhandene Grafik verloren. Zur Ausgabe wird automatisch das Speichermenü aufgerufen, wobei der Punkt **INPUT CAD - DATEI** schon ausgewählt ist. Trotzdem können Sie natürlich erst mal einen Namen eingeben oder andere Punkte aktivieren. Sofortiges Verlassen des Menüs mit **QUIT** bewirkt, daß die Datei nicht abgespeichert und gelöscht wird.

Beim Abspeichern wird der Name mit Leerzeichen und „.g" ergänzt, weil INPUT CAD das so verlangt.

Bildschirm, Punkte

Hier werden einfach nur die Koordinatenpunkte auf den Bildschirm gebracht. Zu Analysezwecken ist das manchmal ganz nützlich.

Ansichten

Nach Anwählen eines dieser Modi bleibt die Hand stehen, statt dessen blinkt der Cursor im unteren Bildschirmfenster. Wer sich vergriffen hat, findet mit der F5-Taste den Rückweg. Wer aber bleibt, muß hier das Weltfenster und das Bildfenster einstellen.Die F3-Taste erleichtert diesen Vorgang, da INPUT-Graph nun die maximalen Werte selbst einsetzt. Das gilt jeweils für die Spalte, in der die Taste gedrückt wird. Das Bildfenster wird für INPUT CAD dabei auf Maximallänge und -höhe von 1023 eingestellt. Der Grafik-Bildschirm ist auf eine Auflösung 200×320 festgelegt. Kleinere Werte als die so vorgegebenen bewirken, daß die Grafiken nur ein Teil des Bildschirms beziehungsweise der Zeichenfläche bei INPUT - CAD einnehmen. So lassen sich mehrere Bilder zum Vergleich nebeneinander plazieren.

Die F1-Taste löst den gewünschten Zeichenvorgang aus, wenn keine unzulässigen Werte eingestellt worden sind, etwa mehr als 320 x-Punkte auf dem Grafikbildschirm oder auch linker Bildschirmrand gleich rechter Bildschirmrand.

Im Falle der Ausgabe an INPUT CAD folgt nun die Berechnung und dann das Speichermenü wie oben beschrieben.

Auf dem Grafikbildschirm taucht zuerst einmal ein Menü auf, das unter anderem die beiden Funktionen **Neu Zeichnen** und **Überlagern** zu bieten hat. Normalerweise wird man die erste verwenden, das heißt, der Bildschirm wird vorher gelöscht. Sollen sich dagegen zwei Kurven in einem Koordinatensystem zum Vergleich präsentieren, müssen sie sich natürlich überlagern. Nach dem Zeichnen stehen auch die Punkte **Ausschnitt** und **Dateneingabe** zur Verfügung. Beim Betrachten eines Ausschnitts ist dieser mittels der „Hand" durch den linken oberen sowie den rechten unteren Punkt zu definieren. Damit wird das Weltfenster neu festgelegt, was sich beim Rückwechseln ins vorherige Grafikmenü überprüfen läßt. Die Funktion **Neu Zeichnen** wird anschließend automatisch angesprungen.

Dateneingabe ermöglich die direkte grafische Manipulation der Daten, was beim Konstruieren von Kurvenzügen oder beim Bearbeiten von Samples nützlich ist. Die Aufforderung **wählen** verlangt, daß die x-Koordinate des zu ändernden Datums mit dem „Zeigefinger" bestimmt wird. Die Höhe spielt dabei keine Rolle, kann aber trotzdem verändert werden, um gezielt Vergleichspunkte anfahren zu können. Anschließend

Für Experten

Wer sich schon einmal mit der Fourier-Analyse beschäftigt hat, weiß, daß die Transformierte eines Rechtecksignals eine SIN-Funktion der Form sin(x)/x ist. Das läßt sich auch mit INPUT-Graph nachvollziehen:

Zuerst sollte das ganze Feld über **Löschen von Feldteilen** mit Nullen gefüllt werden. Das y-Feld wird in den Bereich von 0 bis 64 gelegt, x-Daten auf Zählen geschaltet. Sie erzeugen ein Rechtecksignal, indem Sie unter Punkt 9 im Menü **Rechnen** die Funktion (x>0)*(x<32) +(x>32) eintragen und über Punkt 9 die Werte berechnen. Wenn Sie dieses grafisch darstellen, erhalten Sie Bild 2.

Stellen Sie nun in **x-Daten aus Feld** und X-Anfang auf 64. Jetzt können Sie die **Fourier-Analyse** anwählen. Sie erhalten Werte für den Imaginär- und den Realteil im x-beziehungsweise y-Feld. Zur übersichtlichen Darstellung empfiehlt es sich, die **Feldgrenzen** des y-Feldes auf 0 bis 128 einzustellen und x-Daten auf Zählen umzuschalten. Im Bereich von 64 bis 128 finden Sie die harmonische Reihe der Sinus-Komponenten.

Da die Fourier-Funktion einer Rechteckschwingung keine Cosinus-Komponenten enthält, entsteht im y-Feld von 0 bis 63 aufgrund der Rechengenauigkeit nur ein Rauschen, das heißt, Werte nahe bei Null. Rechnen Sie danach den **Betrag des Spektrums** aus, erhalten Sie nur noch die positiven Beträge der Spektrallinien. Für technische Interpretationen ist nur der Bereich von 0 bis 32 relevant.

Stellen Sie wieder um auf x-Daten zählen. Für die **Werte x-Achse** und **Werte y-Achse** wählen Sie am besten Vier und Fünf. Im **Weltfenster** sollten die Werte 0, 32, 0, 1 für links/rechts und unten/oben gelten. Das Ergebnis entspricht Bild 3. Da der Immaginärteil Null war, ist das Spektrum spiegelsymmetrisch, die Werte oberhalb von 31 sind eine Spiegelung der Werte von 0 bis 32.

Der etwas kompliziert anmutende Vorgang mit den vielen Umschaltungen dient nicht dazu, Sie zu ärgern, sondern folgt aus der Tatsache, daß diese Transformation nur mit komplexen Zahlen ablaufen kann. Will man reelle Signale bearbeiten, muß Ihnen künstlich ein Immaginärteil Null angehängt werden. Sollten Sie nur mit komplexen Zahlen arbeiten, haben Sie diese Probleme nicht.

Wenn es Ihnen Spaß macht, dann experimentieren Sie etwas mit dem Rechtecksignal, indem Sie die Zahlen in der Funktion verändern oder die Formel variieren. Sie können auch über den Punkt **Betrag des Spektrum** die Beträge des Spektrums mit den Ergebnissen der Fourier-Analyse vergleichen. Wählen Sie bei geeigneten Werten eine grafische Darstellung aus x-Feld, erhalten Sie die sehr reizvollen Lissajous-Figuren.

verkündet ein weiteres Fenster: **setzen**, wobei die Hand auf den nächstgelegenen x-Wert gesprungen ist und nur noch in y-Richtung bewegt werden kann.

Ist der neue Wert gewählt und versuchen Sie, die „Hand" in eine beliebige Richtung zu bewegen, können Sie gleich den nächsten Wert manipulieren. RETURN oder Feuerknopf dagegen setzt wiederum die Funktion **Neu Zeichen** in Gang, und das Ergebnis läßt sich bewundern. **Dateneingabe** funktioniert nur mit der Voreinstellung **x-Daten zählen**. Ist ein x-Feld definiert, ist es kaum noch möglich, von einer Bildschirmposition auf das entsprechende Wertepaar zu schließen.

Die im Hauptmenü von INPUT-Graph erreichbare Funktion **Hardcopy** ist die bekannte Input 64-Hardcopy für MPS801 und Kompatible, die den Grafikbildschirm normal und in 4facher Größe ausdrucken kann.

Die nächste Funktion des Hauptmenüs, die Ihnen ein reichhaltiges Menü bietet, heißt **Rechnen** und ist für jede mathematische Manipulation der Daten zuständig.

Ersteinmal gibt es die Möglichkeit, das y-Feld mit einer Konstante zu bearbeiten: jedes Element des Feldes wird um die Konstante erhöht, erniedrigt, mit der Konstante multipliziert oder durch selbige dividiert. Für Techniker bedeutet das: Betrachtet man ein Signal, so könnte man von Beaufschlagung mit einem Gleichanteil oder einer Verstärkung des Signals sprechen.

Des weitern stehen hier der dekadische Logarithmus sowie seine Umkehrfunktion 10^x zur Verfügung, die bei der logarithmischen Darstellung von Daten nützlich sind. Wenn Sie gegen die mathematischen Regeln verstoßen und die Logarithmusfunktion auf negative Zahlen anwenden, führt das selbstverständlich nicht zum Absturz, aber auch nicht zu einem sinnvollen Ergebnis.

Die Betragsfunktion entspricht einer Gleichrichtung, die bekanntlich nur positive Daten zurückläßt, was eine gute Ausgangsbasis für die Logarithmusfunktion ist.

Die Wurzelfunktion, wie sie zur Betragsbildung komplexer Daten benötigte wird, ist komplett neu implementiert worden und daher schneller als auf dem C64 üblich. Dort wird diese Funktion als $x^{0.5}$ realisiert, was den Aufruf von Exponential- und Logarithmusfunktion erforderlich macht. INPUT-Graph dagegen benutzt eine iterative Näherung.

BASIC-Unterstützung

Diese Variablen können im Taschenrechner verwendet oder teilweise in den Eingabefeldern eingegeben werden.

pi – Die Kreiszahl PI
ta – Taschenrechner-Ergebnis

Mit dieser Variablen können Ergebnisse des Taschenrechners in andere Eingabefelder übernommen werden.

x1 – Anfang x-Feld
x2 – Ende x-Feld
y1 – Anfang y-Feld
y2 – Ende y-Feld
le – Feldlänge
sw – Schrittweite
xa – Startwert x
ko – Konstante
f( ) – Das Wertefeld
a( ) – Das Approximations-Feld
mit a(0) – erster Parameter
a(1) – zweiter Parameter
a(2) – Summe aller x-Werte
a(3) – Summe aller y-Werte
a(4) – Summe x*x-Werte
a(5) – Summe y*y-Werte
a(6) – Summe x*y-Werte
a(7) – B (Korrelationskoeffizient)

Die Werte stehen jeweils nach einer Approximation-Rechnung zur Verfügung.

Tabelle: Nützliche BASIC-Variablen:

Auf den Punkt

Der Menüpunkt mit der Nummer 9, **eigene Funktion berechnen**, bietet größtmögliche Vielfalt. In der Zeile darunter muß jedoch zunächst eine Funktion in bekannter BASIC-Syntax eingegeben werden. Die laufende Funktionsvariable heißt, wie könnte es anders sein, „x". Die Variable x kann vor der Auswertung der Funktion nach den oben erklärten Regeln mit einem Wert gefüllt werden. Die Ergebnisse werden im y-Feld abgelegt. So lassen sich sehr schnell Funktionstherme in diskrete Daten verwandeln. Im einfachsten Fall hat man so einen flinken Funktions-Plotter zur Verfügung. Weil aber das Datenfeld ein BASIC-Feld ist (siehe Tabelle 1), können Sie auch eine Funktion über das Feld selbst beschreiben. Damit öffnet sich das ganze Spektrum der Signalfilterung. In Verbindung mit der Fourier-Transformation kann ein Signal im Zeit- und im Frequenzbereich gefiltert werden.

Ein weiterer Bereich ist **Feldverknüpfungen**. Hier können ganze Felder addiert, subtrahiert, multipliziert und dividiert werden. Besonders wer sich an der digitalen Klangsynthese versucht, wird diese Funktionen benötigen. Natürlich müssen vor der Anwendung y-Feld und x-Feld definiert sein, sonst passiert nichts.

Zwei Dienstfunktionen haben wir übersprungen, nämlich **Umkopieren** und **Löschen**. Diese fallen etwas aus dem Rahmen, denn sie halten sich nicht an die Feldgrenzen, sondern erfragen interaktiv, was sie tun sollen. **Umkopieren** verlangt die Angabe des ersten Elementes des Verschiebungsbereichs und dessen Länge sowie das erste Element des Zielbereichs. Beide Bereiche dürfen sich teilweise überlappen. So kann man sich auch eine Kopie eines Datensatzes anlegen, bevor man ihn manipuliert, um einen Vergleich zu haben.

Löschen braucht nur die ersten beiden Angaben und füllt die Feldelemente mit dem Wert Null.

Zugang zu theoretisch sehr aufwendigen Methoden bietet der Punkt **Transformation /Approximation**. Der Umgang mit diesen Verfahren setzt Kenntnis des methodischen Hintergrunds voraus. Da Sie aber jederzeit Zwischenergebnisse abspeichern können, steht dem experimentellen Zugang nichts im Wege. In der nächsten Ausgabe werden wir Ihnen Hintergrund-Information und einige Beispiele anbieten, womit wir den Rahmen dieser Ausgabe mit Sicherheit sprengen würden. Wer sich näher für die besonders interessanten Fourier-Verfahren und deren Anwendung interessiert, sei auf den Grundlagenartikel in c't 8/86 verwiesen. Im Kastentext „Für Experten" finden Sie die notwendigen Hinweise zur Anwendung dieser Funktionen.

Zugriffe

Aus Speicherplatzgründen wird für **Laden und Speichern** jeweils das gleiche Menü mit entsprechender Überschrift verwendet. Es ist jedoch nicht möglich, eine INPUT CAD Datei zu laden – was sollte INPUT-Graph auch damit anfangen?

Sie können die Daten oder Bilder auf vier verschiedene Arten und Weisen laden und speichern:¡

1. **Datenfeld**: Dies ist die kompakteste Methode zum Ablegen der Realzahlen. Das BASIC-Feld wird direkt auf Datenträger geschrieben und kann auch von dort wieder direkt geladen werden.

2. **Sequentielle Datei**: Hier kann eine Datei gelesen oder geschrieben werden, die zum Beispiel mit dem Editor von INPUT-ASS bearbeitete wurde. Es ist das gleiche Format, mit dem auch normale BASIC-Programme mit PRINT# und INPUT# die Zahlen ablegen. Ebenso kann eine solche Datei mit dem Editor von INPUT-ASS geladen, bearbeitet und gespeichert werden.

3. **Speicherbereich**: Die universelle Speicherfunktion, die den angegebenen Speicherbereich laden oder speichern kann. Damit kann auch das Grafikbild ganz oder teilweise gesichert werden.

Um Daten aus Speicherbereichen (zum Beispiel Samples) bearbeiten zu können, werden diese zunächst in einen freien Speicherbereich geladen und dann mit der Funktion **Bytes** ins y-Feld geholt. Genauso läßt sich ein bearbeiteter Sample auch wieder speichern (Siehe dazu Speicherbelegungsplan). Ebenso kann INPUT-Graph mit dieser Funktion verändert werden, indem Teile des Programms überschrieben werden. Das kann allerdings nur von Sachkundigen erfolgreich genutzt werden,da INPUT-Graph dabei leicht zerstört werden kann.

4. **Bytes**: Ab der angegebenen Adresse werden Bytes in das y-Feld geladen (bei Laden), oder der Inhalt des y-Feldes wird ab der Adresse ins RAM gespeichert (unter Speichern), sofern sich die Werte zwischen Null und 255 bewegen.

Damit die Zugriffe auf Datenträger richtig funktionieren, müssen oben noch Gerätenummer und der File-Name angegeben werden (Gerätenummern: 1 = Kassette, 7 = SuperTape, wenn es geladen und initialisiert, und 8 = Diskette). Ist eine Floppy vorhanden, können Disketten-Befehle gesendet und das Directory aufgerufen werden. Nach jeder Disketten-Operation wird der Fehlerkanal abgefragt angezeigt, wenn ein Fehler aufgetreten ist.

Selbstverständlich läuft INPUT-Graph auch mit dem Beschleuniger SuperDisk aus INPUT 64, Ausgabe 1/87, wenn dieser im Bereich oberhalb von 49 151 installiert wird.

0400	Hardcopy-Puffer
0800	HIRESSPEED + Print At/Inline
1075	Peripherie 3 (überdeckt FILL-Befehl)
13D3	HIRESSPEED + Directory-Routine von $1668-$16D5
1A01	Window
1FE0	JoyTast(CRSR)
2090	Hardcopy
2570	FFT (Fast-Fourier-Transformation)
2A00	Rechenroutinen –
	Maschinen-Code
2D30	Draw (die Grafik)
3A90	Funkt (Approximation + diverses)
3EA0	Sprite (Hand)
3F51	BASIC-Anfang
87C0	BASIC-Ende
	Spriteblock 31: hier liegt die Hand
8800	Farbspeicher HiRes
8C00	Bildschirmspeicher
9000	Zwischenspeicher Fenster
A000	Grafik-RAM
C000	nicht benutzt

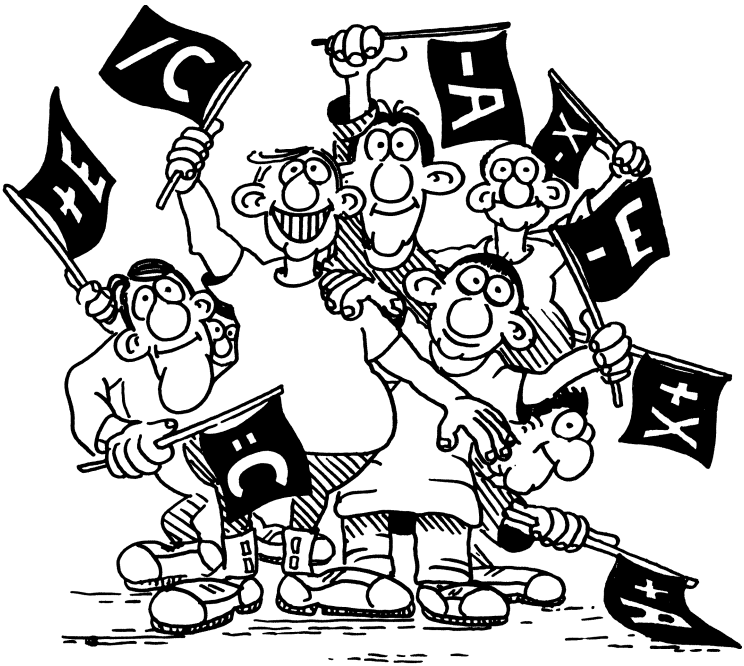
Tabelle 2: Speicherbelegung

Wissen macht's

Je besser Sie den C64 kennen, desto besser können Sie die Leistungsfähigkeit von INPUT-Graph ausnutzen. Hier zahlt sich beim Computer-Hobby gewonnenes Wissen aus. Als Nur-Anwender haben Sie es etwas schwerer, weil Sie wahrscheinlich mit Themen wie Speicheraufteilung , Datenformat oder Grafikauflösung weniger vertraut sind. Unsere Rubrik 64er Tips bietet Ihnen einige Unterstützung.

Natürlich können Sie mit INPUT-Graph einfach herumexperimentieren. Wenn Sie von der Funktion „Speichern" reichlich Gebrauch machen, lassen sich „verunstaltete" Datenfelder jederzeit wiederherstellen. Sie können sich aber auch der Beispiele bedienen, um einige typische Anwendungsfälle kennenzulernen.

Es ist nicht Sinn der Sache, alle Funktionen dieses umfangreichen Programms zu beherrschen und zu verwenden. Die Vielfalt soll nicht abschrecken, sondern Anregung geben, sich mit den verschiednen Anwendungen aus dem Bereich Datenmanipulation auseinanderzusetzen. Wir haben versucht mit diesem Programm Ihnen alle Wege offen zu halten. Franz Dreismann/rh



ICI zum zweiten

Befehlserweiterung und Dokumentation

Doch zuerst für alle, die ICI noch nicht kennen, ein kurzer Rückblick auf die Ausgabe 7/87. Mit ICI – INPUT Command Interpreter – können Sie, ähnlich wie unter CP/M, Datei- und File-Kommandos bis hin zum Batch-Processing einfach als Befehl über die Tastatur eingeben. Ihre Eingaben (beziehungsweise bei Batch-Dateien Eingabefolgen) werden unmittelbar interpretiert und ausgeführt. ICI ist als erweiterbares Programm konzipiert worden und stellt neben dieser Eigenschaft dem geübten Assembler-Programmierer eine Bibliothek von Routinen zur Verfügung.

Innerhalb von INPUT 64 können Sie zwei Programmteile auf Ihren Datenträger überspielen; zum einen die eigentliche Befehlserweiterung und zum anderen ein dokumentiertes Assembler-Source-Listing eben dieser Erweiterung. Auf die Source werden

Den eigentlichen Kommando-Interpreter ICI haben wir Ihnen in der Ausgabe 7/87 vorgestellt. Wir hatten Ihnen damals versprochen, eine ausführliche Dokumentation und eine beispielhafte Befehlserweiterung nachzuliefern. Wir wollen diesem Versprechen hiermit nachkommen und bieten Ihnen zusätzlich noch einen ausführlich dokumentierten Source-Code als Zugabe.

Erweiterung einbinden

wir später eingehen und wollen zunächst für die reinen Anwender sowohl die Einbindung als auch die Anwendung des neuen Befehls beschreiben. Wir gehen dabei davon aus, daß Sie die Erweiterung (sinnvollerweise unter dem Namen „move") abgespeichert und ICI aus der Ausgabe 7/87 in den Rechner geladen haben.

Sie dürfen den ICI-Pufferbereich nicht verändert haben, da die Erweiterung nur in dem voreingestellten Bereich lauffähig ist. Wenn Sie nun den ICI-Befehl **dir** eingeben, sollte sich das File „move" auf der eingelegten Diskette befinden. Dauerhaft (bis zum Ausschalten des Rechners) einbinden können Sie den neuen Befehl mit der Befehlssequenz **ici move** . Das Laufwerk läuft kurz an, und es erscheint unmittelbar das Promptzeichen. Wenn dem so ist, haben Sie ab sofort einen neuen Befehl zur Verfügung.

MOVE als Befehl

Bisher konnten Sie zwar mit dem ICI-Befehl **poke $$$$** unmittelbar in den Speicher schreiben, eine Speicherstelle auszulesen (vergleichbar mit der BASIC-Funktion PEEK) war aber nicht implementiert. Da PEEK(X) als Funktion eine Variable benötigt, ICI aber keine Variablen zur Verfügung stellt, haben wir uns für eine Mischform aus PEEK und POKE entschieden, eben einem Verschiebe-Befehl.

Allgemeine Syntax: **move $$$$ $$$$**

Der MOVE-Befehl liest den Inhalt aus einer Speicherstelle (erste Adresse) und schreibt diesen in die Speicherstelle, auf die der Vektor (zweite Adresse) zeigt. Ein Vektor ist eine Zwei-Byte-Adresse, die im low/high Byte-Format abgelegt ist. Ein klassisches Beispiel sind die sogenannten BASIC-Zeiger in der Zero-Page (zum Beispiel zeigt 2B/2C (hex) auf den BASIC-Anfang).

Wir wollen Ihnen an einem einfachen Beispiel die Wirkungsweise des MOVE-Befehls zeigen.

filename
poke 8000 0000
move 8000 002D

Mit dieser Befehlsfolge wird ein File mit dem Namen 'filename' geladen und an das Ende des Files ein Null-Byte geschrieben. Im einzelnen: Das erste Kommando braucht keine weiteren Anmerkungen. Mit dem zweiten Befehl legen Sie den Wert Null in den Speicher, in diesem Fall in die Adresse 8000 (hex). Der dritte Befehl nun schreibt eben dieses Null-Byte in die Adresse, die durch den Inhalt des Vektors 2D/2E(hex) bestimmt wird. Der Vektor 2D/2E(hex) zeigt nach dem Laden eines Files immer unmittelbar hinter das File-Ende.

Batch-File als Beispiel

Ein zweites Beispiel stellt die folgende Batch-Datei dar.

```
8>edit farbspiel.b
:poke 8001 20
:poke 8002 d0
:move 00a2 8001
:poke 8001 21
:move 00a2 8001
:batch farbspiel.b
:@
8>
```

Listing 1: Batch-File farbspiel.b

Auch hier wieder eine ausführliche Beschreibung. Die Adressen 8001 (hex) und 8002 (hex) werden als Vektor benutzt. Dieser Vektor zeigt nach den beiden POKE-Befehlen auf das Register im Video-Chip, das für die Rahmenfarbe verantwortlich ist; nämlich D020 (hex). Der MOVE-Befehl schreibt nun den augenblicklichen Wert aus der internen Uhr A2 (hex) in die Adresse D020 (hex). Danach wird das Low-Byte des Vektors auf 21 (hex) gesetzt, wodurch der Vektor nun auf die Hintergrundfarbe D021 (hex) zeigt. Der nächste MOVE-Befehl ändert somit die Hintegrundfarbe. Zum Schluß ruft das Batch-File sich selber wieder auf, und das Programm beginnt von vorne. (Abbruch bekanntlich mit RUN/STOP!)

ICI Systemroutinen

Name	Adresse	Übergabe	Rückgabe	Beeinflussung	Beschreibung
INKEY	$CDC1	A=Länge X=Zeile Y=Länge	String ab $2A7 (INKBUFF)	A,X,Y, $57 bis $5C	Stringeingabe am Ende wird ein Nullbyte gesetzt
STROUT	$CDC4	A=LByte Y=HByte	Bildschirm	A,Y, $22 und $23	Stringausgabe String muß mit Nullbyte enden
PRINT	$CDC7	A=ASCII	Bildschirm	A	Zeichenausgabe
DECODE	$CDCA	–	–	–	Interpretervektor
FOPEN	$CDCD	A=Länge ab $2CE (TMPBUF)	–	A,X,Y	File öffnen Name in TMPBUF
FCLOSE	$CDD0	–	–	A,X,Y	geöffnetes File schließen
STATUS	$CDD3	–	Bildschirm	A,X,Y	Fehlerkanal auslesen
DIVICE	$CDD6	$CDA9 (DNUM)	A	A,X,Y	Floppy testen A=0 ok. A=1 nicht ok.
ECHON	$CDD9	–	–	A,X,Y	Echo einschalten
ECHOF	$CDDC	–	–	A,X,Y	Echo ausschalten
DOS	$CDDF	$57=LByte $58=Hbyte	auf Floppy	A,X,Y	String ab Adresse als DOS-Befehl senden
BASC	$CDE2	A=BS-Code	A=A-Code	A	Bildschirm-Code in ASCII-Code
HEXOUT	$CDE5	A=Wert	Bildschirm	A	A-Inhalt als Hexwert ausgeben
GHEXN	$CDE8	A=ASCII	A=Hexwert	A	ASCII in Hex
GHEXW	$CDEB	$57 und $58	$59 und $5A	A,Y, $57 bis $5A	Hexzahl (ASCII) in 16-Bit-Wert
GHEXB	$CDEE	$57 und $58	A=Wert	A,Y, $57 und $58	Hexzahl (ASCII) in 8-Bit-Wert
FILT	$CDF1	A=ASCII	A=ASCII	A	nicht druckbare ASCII ausfiltern A=0 nicht druckbar
ERROR	$CDF4	ab $2A7 (INKBUFF)	Bildschirm	A,X,Y, $22 und $23	Inhalt INKBUFF mit Fragezeichen ausgeben
NXTCOM	$CDF7	–	–	A,Y, $57 und $58	Vektor $57 $58 wird incrementiert wenn Leerzeichen

Tabelle 1: Die Systemroutinen von ICI

Der MOVE-Befehl als Source-Listing

```
9000                    org $9000

                        ; erweiterung fuer ici - move kommando
                        ; (w) frank boerncke   (c) input64
0057        w1      =     $57                   arbeitsvariablen fuer move-routine
0058        w2      =     $58
0059        w3      =     $59
005a        w4      =     $5a
c17f        backici =      $c17f                rueckesprungadresse
                        ; konstante von ici
cdae        mcom    =      $cdae                noch ein kommando in der zeile?
                        ; routinen der ici-sprungleiste
cdc7        print   =      $cdc7                zeichen ausgeben
cdca        decode  =      $cdca                vektor in interpreterschleife
cdeb        ghexw   =      $cdeb                16-bit hexwert holen
cdf7        nxtcom  =      $cdf7                leerzeichen ueberlesen
                        ; routine durch vektor newcom in ici einbinden
                        ; init darf nur einmal aufgerufen werden
9000 adcbcd init    lda decode+1                das alte ziel von
9003 acccd          ldy decode+2                decode dient jetzt
9006 8d2890         sta end+1                   als rucksprungadresse
9009 8c2990         sty end+2                   fuer die eigene routine
900c a917           lda #<newcom                der vektor decode
900e 8dcbcd         sta decode+1                zeigt ab jetzt
9011 a990           lda #>newcom                auf die eigene
9013 8dcccd         sta decode+2                decodierroutinen
9016 60             rts                         zurueck zu basic
                    ; test, ob die naechste zeichenfolge = move ist
9017 a0ff   newcom  ldy #$ff                    zaehler initialisieren
9019 c8     search  iny                         zaehler erhoehen
901a b157           lda (w1),y                  naechstes zeichen holen
901c f011           beq execute                 bei stringende wird in die eigene
901e c920           cmp #32                     routine verzweigt. dieser fall wird
9020 f00d           beq execute                 durch space oder 0 angezeigt
9022 d92a90         cmp comtab,y                mit zeichen aus tabelle vergleichen
9025 f0f2           beq search                  wenn gleich dann weitervergleichen
9027 4cffff end     jmp $ffff                   kommando wurde nicht erkannt.
                    ; zurueck zur normalen decodierung
                    ; $ffff wird durch init gesetzt
902a 4d4f56 comtab b"move",0                    neuer kommandoname
                    ; zeiger und adressen setzen
902f 98     execute tya                         vektor w1/w2 so hochsetzen,
9030 18             clc                         dass er auf das zeichen
9031 6557           adc w1                      hinter dem kommando 'move'
9033 8557           sta w1                      in dem eingabepuffer zeigt.
9035 a900           lda #0                      dazu wird die kommandolaenge
9037 6558           adc w2                      einfach zum vektor addiert.
9039 8558           sta w2                      ( 16-bit addition )
903b a557           lda w1                      der zeiger (w1/w2) wird auf
903d 48             pha                         dem stack gesichert, weil er
903e a558           lda w2                      fuer die weitere abarbeitung
9040 48             pha                         der aktuellen zeile wichtig ist
9041 a90d           lda #13                     erzeugt einen zeilenvorschub auf
9043 20c7cd         jsr print                   dem bildschirm.
9046 a900           lda #0                      mcom = 0 zeigt an, dass nur noch
9048 8daecd         sta mcom                    parameter folgen aber kein kommando
                    ; ab hier die eigentliche routine
904b 20f7cd move    jsr nxtcom                  leerzeichen ueberlesen
904e 20ebcd         jsr ghexw                   2-byte hexzahl nach w3/w4 schreiben
9051 a000           ldy #0                      den wert auslesen, auf den der
9053 b159           lda (w3),y                  vektor (w3/w4) zeigt und ihn dann
9055 48             pha                         auf dem stack zwischenlagern.
9056 20f7cd         jsr nxtcom                  leerzeichen ueberlesen
9059 20ebcd         jsr ghexw                   2-byte hexzahl nach w3/w4 schreiben
905c a000           ldy #0                      den wert auslesen, auf den der
905e b159           lda (w3),y                  vektor (w3/w4) zeigt und ihn dann
9060 aa             tax                         adresse low-byte nach x-register
9061 c8             iny                         adresse+1
9062 b159           lda (w3),y                  adresse high-byte auslesen
9064 8659           stx w3                      adresse nach w3/w4
9066 855a           sta w4
9068 68             pla                         zwischengelagerten wert holen...
9069 88             dey                         ... und in die angegebene
906a 9159           sta (w3),y                  adresse hineinschreiben.
906c 4c7fc1         jmp backici                 ruecksprung zu ici
```

Listing 2: Assembler-Source des MOVE-Befehls

ICI Systemvariablen

Name	Adresse	Beschreibung
DNUM	$CDA9	Gerätenummer des aktuelles Laufwerk
ECHO	$CDAA	1=Echo eingaschaltet 0=Echo ausgeschaltet
FSTAT	$CDAB	1=File geöffnet 0=kein File geöffnet
COPYN	$CDAC	1=PIP mit 2 Floppys 0=PIP mit einer Floppy
SEK	$CDAD	2=BLOAD laden 1=Absolut laden 0=BASIC-Start laden
MCOM	$CDAE	1=in Zeile folgt weiteres Kommando 0=kein weiteres Kommando in der Zeile
BATCHM	$CDAF	1=Befehl kommt aus Batchpuffer 0=Befehl kommt aus Zeilenpuffer
CBATL CBATH	$CDB0 $CDB1	Vektor zeigt auf aktuelles Kommando im Batchpuffer
BATHL BATHH BATLL BATLH	$CDB2 $CDB3 $CDB4 $CDB5	LByte Ende Puffer- HByte Ende Bereich LByte Anfang für Batch- HByte Anfang Datei
ICIHL ICIHH ICILL ICILH	$CDB6 $CDB7 $CDB8 $CDB9	LByte Ende Puffer- HByte Ende Bereich LByte Anfang für ICI- HByte Anfang Puffer
ADRL ADRH	$CDBA $CDBB	Startadresse für GO-Kommando

Tabelle 2: Die Systemvariablen von ICI

ICI-Intern

Dieser Artikel enthält zwei umfangreiche Tabellen, die zum einen die ICI-Systemroutinen dokumentieren und zum anderen die Adressen der ICI-Variablen aufzeigen. Die verwendeten Label-Namen wurden vom Autor festgelegt und sollten auch in Ihren Programmen nicht umbenannt werden, um eine Kompatibilität auf der Source-Ebene zu gewährleisten.

Tabelle 1 enthält die Systemroutinen. Die erste und die zweite Spalte finden Sie im Deklarationsteil der Beispiel-Sources wieder. In der Spalte 'Übergabe' werden die Parameter beschrieben, die vor dem Ansprung der Routine mit entsprechenden Werten besetzt sein müssen. (A=Akkumulator, X=X-Register und Y=Y-Register des 6510 sowie eventuell Zero-Page-Adressen oder ICI-Systemvariablen) Der Inhalt der Spalte 'Rückgabe' zeigt Ihnen, was die ICI-Routine (nach dem Durchlauf) zurückgibt, und in der Spalte 'Beeinflussung' werden die Register und Adressen aufgeführt, die von der jeweiligen Systemroutine verändert werden. Sollten Sie zum Beispiel die hier aufgeführten Adressen mit eigenen Werten versehen haben, müssen Sie diese vor dem Ansprung der Routine retten.

Für eigene Erweiterungen ist DECODE von herausragender Bedeutung. DECODE ist eigentlich keine Routine, sondern ein Vektor. Wenn Sie einen Blick auf das Listing 2 werfen, erkennen Sie in dem Programmteil 'init', daß genau dieser Vektor „verbogen" wird. Bei der Eingabe versucht ICI zuerst die Eingabe als Befehl zu interpretieren. Erst wenn dieses erfolglos ist, wird die Eingabe als File-Name verwendet. Nach der einmaligen Ausführung des Programmteils 'init' wird nun bei jeder Eingabe auch die Buchstabenkombination „move" als möglicher Befehl mit untersucht und, wenn dieses erfolgreich war, in den Programmteil 'execute' verzweigt.

In der Tabelle 2 finden Sie die Namen und die Adressen der Variablen von ICI, unter anderem auch die Zeiger auf die diversen mit ICI-Befehlen änderbaren Adressen und Puffer. Auch diese Variablen sind im Deklarationsteil der Beispiel-Source enthalten.

Der neue MOVE-Befehl ist ein residenter Befehl und in den Adreßbereich 9000 (hex) assembliert. Sie könnten zum Beispiel unmittelbar nach dem Einbinden dieser Erweiterung den ICI-Puffer auf 9100 (hex) legen und hätten dann den ICI-Befehl **ici filename** wieder frei für nicht residente Befehle.

Mit diesen Informationen sollten erfahrene Assembler-Programmierer in der Lage sein, sowohl Erweiterungen selbst zu programmieren als auch die ICI-Systemroutinen quasi als Bibliothek zu nutzen.

Source als Zugabe

Aus INPUT 64 können Sie – neben der eigentlichen Befehlserweiterung – auch den kompletten Source-Code abspeichern. Das File liegt Ihnen dann im INPUT-ASS-Format vor und kann unmittelbar von unserem Assembler (Ausgabe 6/86) gelesen werden. Im Defionitionsteil sind alle ICI-Routinen und ICI-Variablen als Labels deklariert, so daß Sie diese Source auch für eigene Erweiterungen gut einsetzen können. Das Listing in diesem Artikel ist allerdings aus Platzgründen im Deklarationsteil auf das Wesentliche reduziert worden.

Sofern Sie eigene ICI-Erweiterungen schreiben wollen, haben Sie jetzt alle notwendigen Informationen zur Verfügung. Sollte die eine oder andere Erweiterung die Redaktion erreichen, könnte es durchaus noch einen dritten Teil von ICI geben.

Frank Börncke/WM

Einer gegen Alle

Code wieder geknackt

Bei 100 000 INPUT-64-Lesern stellen zwar 63 Einsender eine verschwindende Minderheit dar, nur: bei unseren Spielregeln hätte auch eine richtige Lösung gereicht.

Auch die Dechiffrierungs-Aufgabe aus der Ausgabe 6/87 ist gelöst worden. Wir erhielten diesmal sogar über 60 richtige Einsendungen.

Das Lösungsprogramm

Sie können sich innerhalb von INPUT 64 die Demonstration des Lösungsweges ansehen. An dieser Stelle nur einige Anmerkungen zur Anwendung des Chiffrierungs-Programms, das Sie aus INPUT 64 abspeichern können. Wenn Sie dieses Programm laden und mit 'RUN' starten, werden Sie aufgefordert, den zu cheffrierenden Texteinzugeben. Nach einer kurzen Rechenzeit erscheinen auf dem Bildschirm mehrere Programmzeilen, in denen die kodierten Werte stehen. Diese Zeilen übernehmen Sie einfach mit der RETURN-Taste fest in das Programm. Wenn Sie danach das Programm mit 'RUN 300' starten, werden die neuen Zeilen abgearbeitet, und als Ergebnisdiverser Berechnungen erscheint Ihre Text-eingabe wieder. Da das Programm gut dokumentiert ist, werden Sie den kleinen Trick mit der Zeileneingabe sicherlich schnell erkennen.

Erste Code-Folge

13290, 22116, 25251, 92230, 62270,
61220, 35201, 81240, 82169, 52281,
33220, 13250, 32241, 91250, 92129,
83189, 94109, 91199, 94189, 23176,
44221, 55137, 33200, 04290, 84312,
21136, 93169, 95129, 61362, 24176,
54251, 85210, 63230, 64230, 23146,
32196, 31271, 33210, 82250, 55342,
04343, 52109, 41281, 61210, 94169,

Die Chiffrierung bedeutet:
„Diese Code-Folge dient nur als Hilfestellung!"

Tabelle 1: der erste bekannte Code

Zweite Code-Folge

33200, 65240, 13230, 94179, 94109,
53221, 22291, 02303, 81199, 93290,
64280, 92179, 54129, 33280, 23146,
82220, 85352, 53271, 51312, 65312,
83210, 24116, 92159, 12260, 21166,
21176, 83159, 62230, 94149, 94199,
43261, 91280, 82240, 62210, 34261,
01240, 34251, 55137, 31251, 61220,
95109, 32186, 44231, 31230, 25126,

Die Chiffrierung bedeutet auch:
„Diese Code-Folge dient nur als Hilfestellung!"

Tabelle 2: der zweite bekannte Code

Dritte Code-Folge

23280, 65201, 83210, 33200, 22241,
44291, 24116, 61139, 24166, 32280,
81260, 94179, 95119, 05290, 22201,
73149, 22196, 32211, 95189, 25146,
95159, 75260, 45221, 14230, 24126,
73251, 33290, 64302, 21280, 52129,
94189, 81290, 93129, 46201, 82260,
06210, 91179, 84250, 53271, 35230,
61210, 63230, 72179, 81324, 52139,
21250, 33166, 21146, 66220, 32250,
24240, 25271,

Dieser Text wird gesucht

Tabelle 3: der erste unbekannte Code

Vierte Code-Folge

65149, 95159, 72220, 23126, 23260,
24290, 61220, 81280, 16210, 95109,
83210, 92159, 33270, 61271, 92139,
91199, 22201, 72119, 53199, 44281,
44221, 25126, 73189, 82260, 83250,
04200, 61250, 62290, 21116, 25186,
24230, 56129, 81230, 24251, 32280,
41261, 31240, 83344, 56201, 33231,
24211, 24260, 25136, 62342, 35260,
92179, 35196, 94179, 74241, 35210,
03200, 25176,

Dieser Text ist identisch mit der dritten Code-Folge

Tabelle 4: der zweite unbekannte Code

Die Spielregeln

Mit dieser Ausgabe stellen wir Ihnen bereits die dritte Dechiffrierungs-Aufgabe. Für alle neu hinzugekommenen INPUT-Leser hier noch einmal die Spielregeln. Wir geben jedesmal eine chiffrierte Code-Folge vor, und erwarten von Ihnen nicht mehr und nicht weniger als den Klartext, der sich hinter dieser Code-Folge versteckt. Um Ihnen die Aufgabe zu erleichtern, erhalten Sie mehrere Code-Folgen (natürlich mit dem gleichen Algorithmus kodiert), wovon wir Ihnen für zwei auch gleich den Text mitliefern.

Natürlich können Sie auch was gewinnen. Wenn Ihr Chiffrierungs-Programm (Sie können also auch ein Programm einschicken) innerhalb von vier Wochen nicht geknackt wird, bekommen Sie ein INPUT-64-Jahresabonnement. Sie spielen also alleine gegen 100 000 INPUT-Leser. Sollten – was bisher jedesmal der Fall war – in dieser Frist richtige Lösungen bei uns eintreffen, verlosen wir unter den Einsendern fünf Bücher, während der Programmentwickler dann leer aus geht.

Aus den vielen richtigen Einsendungen wurden per Los die folgenden fünf Gewinner ermittelt: R. Böhm, 8500 Nürnberg; H. Graf, A-8020 Graz; T. Kriete, 4970 Bad Oeynhausen; J. Kurz, 6830 Schwetzingen; L. Schragmann, 5000 Köln. Die Gewinner erhalten entsprechend den Spielregeln einen Buchpreis. Herzlichen Glückwunsch!

Neue Aufgabenstellung

Unter den uns vorliegenden Chiffrierungs-Programmen haben wir das von M. Schmidt aus Gailingen ausgewählt. Die Ergebnisse dieser Chiffrierung finden Sie in den vier Tabellen und in einem Programm, das Sie aus INPUT 64 abspeichern können. Wenn Sie sich beteiligen möchten, beachten Sie bitte, daß uns Ihre Einsendung bis Freitag den 2.10.87 erreicht haben muß. WM

KI
Die Computer-anwendung von morgen.

COMPUTER-BUCH

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

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

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

Broschur, ca. 220 Seiten
DM 44,80
ISBN 3-88229-126-5

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

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

Im Buch-, Fachhandel oder beim Verlag erhältlich. KI/12

HEISE

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

Auf den Punkt gebracht

64er Tips: Grafik gewußt wie

Thema der letzten Ausgabe der 64er Tips waren die Grundlagen der Speicherverwaltung und die Organisation einer Bitmap. Dieses Mal soll dem unbefriedigenden Zustand des leeren Bildschirms abgeholfen werden, wobei etwas Rechenarbeit nicht zu umgehen ist. Wenn der Algorithmus aber fertig ist, dann rechnet nur noch der C64 und der Programmierer kann sich ganz der Kreativität widmen.

Wer künstlerisch oder technisch mit dem Bleistift eine Zeichnung aufs Papier bringt, tut das in der Regel mit Strichen, sprich Linien. Ganz unverständlich scheint da das Bedürfnis des Computers, seine Grafiken aus einzelnen Punkten zusammenzusetzen. Da aber Punkte nebeneinander aufgereiht auch eine Linie ergeben, scheint doch noch eine Verständigung möglich.

Koordinationsprobleme

Grundlage jedes Grafikbildes auf dem C64 ist der Bildpunkt, auch Pixel genannt. Im hochauflösenden Modus besteht der Bildschirm aus 64 000 Einzelpunkten, 320 in einer Zeile und das 200 Mal übereinander, die alle einzeln ein oder ausgeschaltet sein können. Je nachdem, welche Pixel angesprochen sind, entsteht ein anderes Bild auf dem Monitor. Als erstes muß also ein Algorithmus (eine Rechenvorschrift) entwickelt werden, die das Setzen und Löschen von Pixeln erlaubt. Alles weitere kann darauf aufbauen.

Um anzugeben, welches Pixel gemeint ist, könnte man einfach alle von oben nach unten durchnummerieren, und jedes Pixel anhand seiner Nummer ansprechen. Das ist für den Menschen aber nicht sehr anschaulich, oder können Sie sich auf Anhieb vorstellen, wo das Pixel Nr. 25 387 in etwa auf dem Bildschirm zu finden ist? Sicher nicht. Weiter kann man sagen: „das 107te Pixel in der 79ten Zeile", und das hört sich dann schon besser an.

Allgemein heißt das also, daß ein Pixel in der X-ten Spalte und in der Y-ten Zeile zu finden ist. Mehr mathematisch heißt das im folgenden: „Das Pixel hat die Koordinaten (107,79) oder allgemeiner (x,y)". Ein Pixel wird also durch seine beiden Koordinaten x und y bestimmt, wobei der Bildschirm als Koordinatensystem mit x von 0–319 und y von 0–199 zu sehen ist. Der Punkt (0,0) liegt dabei wie aus dem Mathematikunterricht vertraut links unten. Wenn Sie bereits mit Grafik auf dem C64 zu tun hatten, wissen Sie vielleicht, daß die meisten BASIC-Erweiterungen diesen Punkt nach links oben legen, was aber außer der Ersparnis einer Subtraktion beim Algorithmus keine Vorteile bringt, sondern nur den Programmierer ärgert, der sich die y-Achse immer auf dem Kopf vorstellen muß.

Punktierter Algorithmus

Zuerst interessiert die Speicheradresse des Bytes, in dem der Punkt mit den Koordinaten (x,y) liegt. Da ein Byte aus acht Bits besteht, muß diese Adresse für jeweils acht Punkte gleich sein. Da die Bytes waagrecht auf dem Monitor abgebildet werden, trifft diese Gleichheit nur die x-Koordinate. Alles, was nicht durch acht teilbar ist, beeinflußt nur die Bitposition innerhalb des Bytes und kann mit 8*INT(X/8) abgetrennt werden.

Da der Aufbau der Bitmap große Ähnlichkeit mit dem Textmodus hat, ist der Einfluß der y-Koordinate etwas komplizierter. Liegt diese zwischen 0 und 7, kann sie durch den Ausdruck (Y AND 7) direkt addiert werden, ist sie größer, muß jeweils eine Textzeile 'übersprungen' werden, das heißt es wird der Ausdruck 40*(Y AND 248) addiert. Da die Bitmap nicht ab Adresse Null im Speicher liegt, kommt dieser Wert noch hinzu. Für eine Bitmap unter dem BASIC-ROM ist der Startwert 40 960. Damit der Punkt (0,0) links unten liegt, muß nun noch die y-Achse umgedreht, das heißt Y mit 199-Y vertauscht werden.

Damit steht die Adresse des Bytes in der Bitmap fest. Bei der Bestimmung der Bitposition leistet die zwei-hoch-Funktion gute Dienste, denn jedes Bit in einem Byte entspricht ja als Dualzahl betrachtet einer Zweierpotenz. Hier ist genau der Teil der x-Koordinate von Interesse, der eben abgeschnitten wurde. Er errechnet sich einfach mit (X AND 7). Mit (7-(X AND 7)) erhält man den richtigen Exponent für die Potenzfunktion, die ein Byte liefert, in dem das gewünschte Bit gesetzt und alle anderen gelöscht sind.

Schlußendlich darf dieses Byte nicht einfach an die berechnete Adresse geschrieben werden, weil dabei bereits gesetzte Punkte verloren gehen könnten. Eine ODER-Verknüpfung des Bytes in der Bitmap und des berechneten Bytes bringt hier das gewünschte Ergebnis.

Listing 1 zeigt Ihnen ein BASIC-Unterprogramm, das problemlos für die Bitmaps 1–4 ($2000-$8000) arbeitet. Für echte Anwendungen ist es viel zu langsam, aber es zeigt gut das eben beschriebene Prinzip.

Breite Farben

Derselbe Algorithmus läßt sich auch zum Setzen von Multicolor Punkten benutzen.

```
1000 A=8*INT(X/8)+(199-Y AND 7)
     + 40*(199-Y AND 248)+8192
1010 POKE A,PEEK(A) OR
     2↑(7-(X AND 7))
1020 RETURN
```

Listing 1: Dieses Unterprogramm setzt einen Punkt in der Grafik ab $2000 (8 192)

Beschreibung des Grafiktools 2. Teil

Das Tool, das Sie dieses Mal mit CTRL-S aus dem Magazin heraus sichern können, funktioniert nur zusammen mit dem ersten Teil, der in der letzten Ausgabe enthalten war. Das Maschinenprogramm ist wiederum relokatibel und kann in der gleichen Weise behandelt werden, wie in der letzten Ausgabe beschrieben. Der zweite Teil unterstützt das Setzen eines Punktes und das Ziehen einer Linie im Hires- und im Multicolor-Modus. Dabei gibt es zwei Abweichungen gegenüber bekannten BASIC-Erweiterungen:

1) Der Punkt (0,0) liegt links unten und nicht etwa links oben.
2) Im Multicolor Modus sind nach wie vor x-Koordinaten von 0–319 möglich. Dafür entsprechen jeweils eine gerade und die folgende ungerade Koordinate demselben Bildschirmpunkt.

Line: zeichnet eine Linie
SYS ladeadresse+73,x1,y1,x2,y2,fa
x1,y1,x2,y2=Anfangs- und Endpunkt der Linie. x zwischen 0 und 319
y zwischen 0 und 199
fa: Zeichenfarbe,
0 oder 1 für hochauflösend,
0 – 3 für Multicolor

Plot: setzt einen Punkt
SYS ladeadresse+76,x1,y1,fa
x1,y1,fa siehe oben

Dort wird die Farbe von zwei nebeneinander liegenden Bits bestimmt. Der Punkt erscheint nun doppelt so breit wie zuvor. Da beide Bits immer in einem Byte liegen, braucht die Berechnung der Byte-Adresse nur einmal stattfinden. Unter der Bedingung, daß nur gerade x-Koordinaten zugelassen sind, ist die Formel in Listing 1, Zeile 1000, ohne jede Veränderung brauchbar. Viele BASIC-Erweiterungen arbeiten hier allerdings mit x-Werten von 0–159, was dann zwar jeder Koordinate einen Punkt zuweist, von der Vorstellung her aber irreführend ist. Eine Strecke in x-Richtung mit 40 Punkten ist genauso lang wie eine Strecke in Y-Richtung mit 80 Punkten auf dem Bildschirm. Demgegenüber steht die Lösung, weiter den vollen Bereich von 0–319 zuzulassen und ungerade Koordinaten gleich der nächstniederen geraden Koordinate zu setzen, besser dar. Assemblerprogrammierer, die selbst an einer Grafik Erweiterung schreiben wollen, sollten über diesen Punkt einmal nachdenken.

Die Tatsache, daß beim Setzen eines Multicolor Punktes auch gezielt Bits gelöscht werden müssen, macht ein entsprechendes BASIC-Programm allerdings sehr unhandlich. Deshalb soll hier nicht weiter darauf eingegangen werden.

Immer geradeaus

Ist die Routine zum Setzen eines Pixels auf einem Rechner einmal erledigt, so gibt es für die weiteren Entwicklungen eine große Zahl von Algorithmen, auf die der Programmierer zurückgreifen kann. Die nächste interessante geometrische Figur nach dem Punkt ist sicher die kürzeste Verbindung zwischen zwei Punkten, die gerade Linie. Im Magazin stellen wir Ihnen die einfachste Form des Bresenham-Linien-Algorithmus vor, der auf einer einfachen Idee beruht. Eine Linie ist gegeben durch ihre Steigung und ihren Anfangspunkt. Die x-Koordinate wird schrittweise um Eins erhöht, der Steigungszähler jeweils um das entsprechende Stück. Nur wenn dieser den Wert eins überschreitet, wird er um Eins vermindert und die y-Koordinate um Eins erhöht. Dadurch werden die Punkte möglichst nahe an der originalen Geraden gesetzt. Das funktioniert leider nicht mehr, wenn die Steigung größer Eins ist, das heißt wenn die y-Koordinate schneller wächst als die x-Koordinate. In diesem Fall müssen x und y die Rollen tauschen, es wird dann y ständig um Eins erhöht und x nur bei Bedarf.

```
800 DX=ABS(X1-X2):DY=ABS(Y1-Y2)
810 IF X1>=X2 THEN 850
820 X=X1:Y=Y1:Z=1:if Y1>Y2thenZ=-1
830 GOTO 860
840 X=X2:Y=Y2:Z=1:if Y2>Y1 THEN Z=-1
850 I2=DY:if DX>DY THEN I2=DX
860 DZ=INT(I2/2)
870 GOSUB 1000
880 FOR I1=1 TO I2
890 IF DZ<DX THEN DZ=DZ+DY:X=X+1
900 IF DZ>=DX THEN DZ=DZ-DX:Y=Y+Z
910 GOSUB 1000
920 NEXT:RETURN
```

Listing 2: Dieses Unterprogramm ruft obiges auf und kann eine Linie von (x1,y1) nach (x2,y2) ziehen.

Ein BASIC-Unterprogramm für den auf alle Richtungen erweiterten Bresenham-Algorithmus finden Sie in Listing 2. Dort wird mit GOSUB 1000 das Unterprogramm zum Setzen eines Pixels aufgerufen. Natürlich können Sie hier auch den PLOT-Befehl einer Grafikerweiterung einsetzen, was sehr zur Beschleunigung beiträgt.

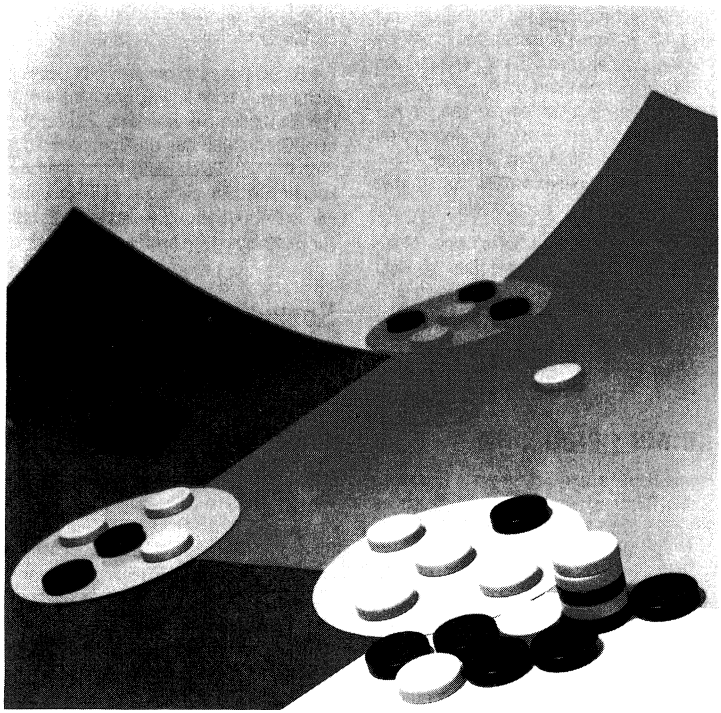
Speziell und schnell

Obwohl der Bresenham-Algorithmus schon gut für die Assemblerprogrammierung geeignet ist, wird er auf dem C64 meist nicht verwendet. Die schnellen Grafik Erweiterungen benutzen einen speziellen Algorithmus, der davon ausgeht, daß es zu aufwendig ist, die Byteadresse jedes Punktes neu zu berechnen. Anhand des Aufbaus der Bitmap erkennt man, daß beim Zeichnen einer Linie das nächste Pixel immer in unmittelbarer Nähe des letzten, oft sogar im selben Byte, zu finden ist. Das Grafikpaket HIRESSPEED, zuletzt in INPUT 64, Ausgabe 4/85, veröffentlicht, bietet ein Beispiel dafür.

```
10 REM Hier muß auf
20 REM Grafik umgeschaltet werden
30 REM Hier Grafik löschen
40 FOR I=0 TO 319 STEP 5
50 X1=0:Y1=0:X2=I:Y2=199:GOSUB 800
60 X1=319:Y1=199:X2=I:Y2=0:GOSUB 800
70 X1=160:Y1=199:X2=I:Y2=0:GOSUB 800
80 NEXT I
```

Listing 3: Demonstration zum Moriee-Effekt.

Wenn Sie nach all den Berechnungen genug davon haben, dann sollten Sie sich etwas Entspannung gönnen und mit dem Moiree-Effekt experimentieren. Ein Beispiel dafür finden Sie im Magazin. Dort ist kein ausgefuchstes Programm am Werk sondern ein paar einfache BASIC-Schleifen, die in einem gleichmäßigen Muster die Linien auf den Bildschirm bringen. Beginnen Sie einfach mit Listing 3 und versuchen Sie nach und nach andere Formen zu erzeugen. Der Aufruf der Linienroutine mit GOSUB 800 sollte auch hier wenn möglich durch den LINE-Befehl einer Grafik Erweiterung ersetzt werden. Franz Dreismann/kfp



Bunt zum Basar

Spiel: Farb-Basar

Farb-Basar ist ein Spiel, an dem sich bis zu 6 Personen beteiligen können. Jeder Spieler erhält durch Würfeln oder Tauschen farbige Punkte. Stimmen je fünf dieser Punkte mit den Punkten eines CHIPS überein, kann dieser gekauft werden. Der unterschiedliche Wert dieser CHIPS wird in entsprechende Bonuspunkte umgerechnet, allerdings gehen in diese Berechnung noch andere Gesichtspunkte ein. Dazu später mehr.

Gewinner ist der Spieler, der bis zum Schluß die meisten Bonuspunkte gesammelt hat. Außerdem geht die Anzahl der gekauften CHIPS positiv und die übrig gebliebenen Farbpunkte negativ in die Bewertung ein.

Bei diesem Kombinatorik-Spiel geht es nicht um Geld, sondern um einen sicheren Blick für Farben und Farbkombinationen. Denn diese müssen um- und eingetauscht werden.

Schwarz, Rot, Gelb . . .

Kommen wir zum Spielverlauf. Hat man keine oder nicht genügend Farbpunkte, was ja zu Beginn des Spiels zutrifft, muß gewürfelt werden. Dabei wird der Farbpunkt, dessen Farbe vom Würfel angezeigt wird, im jeweiligen Spielerfeld eingetragen. Erscheint auf dem Würfel die Farbe 'Grau', erhält man einen Joker. Das heißt, man kann sich von den zur Verfügung stehenden Farben eine aussuchen.

Wird mindestens ein Farbpunkt im Spielerfeld angezeigt, kann auch getauscht werden, sobald die Tabelle 'TAUSCHEN' diese Möglichkeit zuläßt. Man bewegt den Pfeil auf die Farbkombination, die man abgeben will und erhält dafür die gegenüberliegende. Der Pfeil läuft, wenn die unterste Zeile erreicht ist, nach rechts oben um.

. . . oder Weiss und Blau

Die zu kaufenden CHIPS befinden sich etwa in der Mitte der unteren Bildschirmhälfte. Sie sind in vier Stapel zu je zwölf Stück angeordnet, wobei immer die zu oberst liegenden CHIPS zu sehen und zu kaufen sind. Besitzt man die gleichen Farbpunkte, mit denen ein CHIP gekennzeichnet ist, kann man diesen kaufen. Den jeweils eingestellten CHIP erkennt man an der veränderten Form. Hat man sich für den Kauf entschieden, werden die entsprechenden Farbpunkte abgezogen und ein Bonus gutgeschrieben. Je weniger Farbpunkte man nach einem Kauf in seinem Feld übrig hat, desto höher ist der Bonus, den man dafür bekommt.

Das Spiel kann wahlweise mit dem Joystick oder der Tastatur bedient werden. Erscheint auf dem Bildschirm ein blinkender Cursor, so ist die Eingabe nur über die Tastatur möglich. Mit den Cursor-Tasten oder dem Steuerknüppel läßt sich die jeweils mögliche Option einstellen. Durch Drücken der RETURN-Taste oder des Feuerknopfes wird der vorher eingestellte Befehl ausgeführt.

Übrigens: Mehr als acht Punkte der gleichen Farbe können nicht angezeigt werden. Sie gehen aber nicht verloren. Ein vorzeitiger Spielabbruch ist jederzeit mit der Taste 'E' möglich.

Schnell überblickt

Betrifft: Programmdokumentation

Der Autor von Farb-Basar (Theodor Carstensen) hat zu diesem Programm eine ausführliche Dokumentation mitgeliefert. Sie soll, weil wirklich gut gemacht, Anreiz und Vorbild für alle unsere Autoren sein. Denn für eine bessere Dokumentation könnte es auch mehr Geld geben. In den folgenden Bildern wird sie auszugsweise vorgestellt. Außer der Programmbeschreibung und der Variablen-Liste existieren noch eine 'SYS- und Peek-Adressen-Liste', sowie ein gut aufbereitetes Programm-Listing. Falls man nachträglich Änderungen am Programm vornimmt, kann man anhand der Variablen-Liste sofort sehen, welche Variablen schon belegt sind und welche nicht. Sehr hilfreich ist die Programmbeschreibung. Auch hier sieht man sofort, was in den einzelnen Programmbereichen vom Programm abgearbeitet wird. kfp

Variablen-Liste
zu "Farb-Basar 3.0"

Variable	Bedeutung
.b.....	*Schalter Bonus (ein/aus)
.c(x)....	.Stapel-Nr. (x=Chipanzahl)
cs(x)....	.Chipanzahl (x=Spieler-Nr.)
.w$.....	.allgemeiner String
.x$.....	.allgemeiner String
.y$.....	.allgemeiner String

POKE- und SYS-Adressen
zu "Farb-Basar 3.0"

Zeile	poke	peek	sys	Begründung
90	788,52	.....	.....	RUN/STOP Tasten ausschalten
1010	53280,0	.....	.....	Bildschirmrand schwarz
1010	53281,0	.....	.....	Hintergrund schwarz
11010	53281,0	.....	.....	Hintergrund schwarz
11780	53281,5	.....	.....	Hintergrund grün
11810	198,0	.....	.....	Tastaturpuffer löschen

Programmbeschreibung
zu "Farb-Basar 3.0"

Zeilen	Routinenname	Beschreibung
10 - 720	Variablenzuweisung	RUN/STOP-Taste ausschalten Variablenzuweisung
1000 - 1160	Vorbereitung	Rahmen- und Hintergrundfarbe einstellen / Großschrift einstellen und blockieren / Titelbild zei-
		Spielernummer sp wird auf 0 gestellt / es folgt die Routine "Start"
2700 - 2820	Start	aktueller Spielername und Punktestand werden revers mittelgrau angezeigt / Variable sp wird um 1 erhöht / aktueller Spielername und Punktestand wird revers rot angezeigt / Sprung in die Routine

Sanft gelöscht

Trickblende für Bildschirmwechsel

Das Hilfsprogramm entstand aus dem Bedürfnis, Bildschirmwechsel „mal anders" zu gestalten. Viele INPUT 64-Leser kennen das Verfahren bereits aus so mancher 64er Tips-Serie.

Sie erhalten die Maschinen-Programme für diese Animation in dieser Ausgabe als BASIC-Tool. Sie können sich das Tool mitsamt Demo-Programm über CTRL-S auf eigenen Datenträger abspeichern. Wollen Sie Ihr eigenes Programm entwickeln, löschen oder überschreiben Sie einfach das BASIC-Programm. Denken Sie bitte immer daran, vor dem Abspeichern neuer Raffinesse POKE 44,8 und POKE 43,1 im Direktmodus einzugeben. Ganz nebenbei haben wir Print At noch mit in das Tool gepackt.

In der Regel führt ein nüchternes Clear-Home beim C64 zum harten Löschen des Bildschirms. Schnelle Bildschirmwechsel bewirken eine unruhige Darstellung und sind nicht gerade gnädig mit den Augen. Da gibt es bei Film und Video schon feinere Übergänge zu sehen. Mit dem „Trickblender" können Sie jetzt auch bei Programmen Bildwechsel abwechslungsreicher gestalten.

Print At dabei

SYS 2078, zl, sp, Text

zl	Zeilennummer	von 0 bis 24
sp	Spaltennummer	von 0 bis 39

Text: BASIC-String (-Variable)

Gezielt angezeigt

Trickblende per SYS

SYS 2075, t, f, a

t Tempo : 0–255 (schnell/langsam)
f Farbe : 0–15 (C64-Standard)
a Arten:
1 Balken senkrecht von links
2 Balken senkrecht von rechts
3 Balken waagerecht von oben
4 Balken waagerecht von unten
5 Rahmen öffnend
6 Rahmen schließend

Trick-Parameter und -Aufrufe

Wie Sie im Modul „Trickblende" beoachten können, bietet dieses Tool drei verschiedene Formen für die „Blende": Balken waagerecht und senkrecht laufend und ein sich öffnendes oder schließendes Rechteck. Außerdem können Sie das Tempo der Überblendung und die Farbe des Balkens selbst bestimmen. Adressen, Bedeutung und Funktion der Parameter können Sie den Tabelle entnehmen. rh

Lernen im Dialog

Englische GRAMmatik

Die Möglichkeit, korrektes Englisch zu sprechen, hat im Prinzip jeder. Ob er auch die notwendigen Fähigkeiten besitzt, ist eine ganz andere Sache. Um diese feinen, aber entscheidenden Unterschiede dreht sich dieser Teile der Grammatikserie.

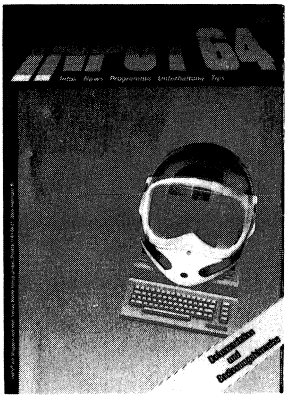
Mit diesem interaktiven Grammatiktrainer können Sie diesmal folgende Konstruktionen üben:

- Modal verbs, expressing ability
- Modal verbs, expressing permission
- Modal verbs, expressing possibility
- Modal verbs, expressing obligation
- Shall or will?
- Should or would?
- Mixed forms of modal verbs

Die Bedienung des Lernprogramms ist denkbar einfach: Die richtigen Eingaben müssen jeweils in die Textlücken der Beispielsätze eingegeben werden (mit RETURN abschließen); nach jeder Eingabe können Sie entweder eine der auf dem Bildschirm gezeigten Möglichkeiten wählen oder mit einer beliebigen anderen Taste mit der nächsten Frage fortfahren.

Die Bezeichnung „Lernprogramm" ist übrigens etwas irreführend. Es werden nämlich nicht die grammatischen Regeln oder gar deren Hintergrund vermittelt, sondern es geht darum, vorhandene oder im Lauf der Zeit verschüttete Kenntnisse einzuüben beziehungsweise aufzufrischen. js

Am 5. Oktober an Ihrem Kiosk: INPUT 64, Ausgabe 10/87



Wir bringen unter anderem:

Speed-Compiler

Ein langgehegter Leserwunsch geht in Erfüllung: ein BASIC-Compiler, der nicht nur Ihre Programme beschleunigt (bis zum Faktor 10!), sondern auch die Einbindung unserer Spracherweiterung INPUT-BASIC ermöglicht. Außerdem wird eine eigene Integer-Arithmetik zur Verfügung gestellt, ist die String-Verwaltung frei von Garbage Collection, können Programmteile nachgeladen werden und so weiter und so fort . . .

Chamäleon

Mustererkennnung ist eine der Fähigkeiten, die der Mensch deutlich besser beherrscht als sein elektronischer Konkurrent. Trotzdem ist dieses Spiel, das übrigens auch zu zweit gespielt werden kann, nicht so einfach, wie der erste Eindruck vermuten läßt.

HexBinDez-Help

Wenn Sie auf Anhieb sagen können, wie die Dezimalzahl 33333 hexadezimal dargestellt wird oder Sie die Binärdarstellung 10100111 total durchsichtig finden, ist dieses Tool für Sie überflüssig. Alle anderen können künftig beliebige Hin- und Herrechnungen vom Rechner erledigen lassen, der diese Programmiererhilfe als „Backgroundtool" im Direkt- und im Programmmodus im Hintergrund bereithält.

c't — Magazin für Computertechnik

Ausgabe 10/87 — ab 18. September am Kiosk

Projekte: Transputer-Board: 32-Bit-Grundstein für den Super-Computer zum Selbstbau * PAK-68: Unterstützung von 68020 und 68881-Arithmetikprozessor im AmigaDOS * Software-Know-how: Compiler-Benchmarks * Faktisch falsch: Norton-Faktor im Detail * Effizientes Suchen und Sortieren mit AVL-Bäumen * u.v.a.m

elrad — Magazin für Elektronik

Ausgabe 10/87 — ab 28. September am Kiosk

Zykluslader — Ladeautomat für NiCd-Zellen * Bauanleitung Röhrentechnik: 250-W-Endstufe * Grundlagen: Schrittmotoren-Know-how * Und läuft und läuft und läuft . . . CMOS-Bausteine unter Energie-Gesichtspunkten * 12 s in einem Chip: Voice-Sampler * Wochenendprojekt: Fern-Fernbedienung * „Leidet" unter Rauscharmut: Mikrofon-Vorverstärker * 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.

INPUT 64 - 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
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

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
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.