DAS ELEKTRONISCHE MAGAZIN 5/88

INPUT 64

Infos · News · Programme · Unterhaltung · Tips

Unverbindliche Preisemptehlung

Für C64 und C128

Makro-Assembler

40-/80-Zeichen-Modus wahlweise

Für alle Grafikdrucker

HardCopySystem

Druckt Text, Grafik
und Sprites

Für Risikofreudige

Roulette

Spieltisch-Simulation

64er Tips
Französische Grammatik
Neues Rätsel
ID-Werkstatt
Sample Play

Über 140 KByte Software
Ohne Abtippen

Das Magazin für C64- und C128-Anwender

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 Serienausführungen 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 64er-BesitzerInnen!

„Wat dem einen sin Uhl, is dem annern sin Nachtigall" heißt es in einem bei uns in Norddeutschland beliebten plattdeutschen Sprichwort. Das hochdeutsche Äquivalent „Jedem recht getan ist eine Kunst, die niemand kann" dürfte bekannter sein und bedeutet etwa dasselbe.

Was das soll? In unserem ständigen Bemühen, unseren Lesern zwar nicht gerade die Wünsche von den Augen abzulesen (Wie sollten wir auch?), aber auf briefliche und telefonisch geäußerte Ansinnen möglichst prompt zu reagieren (Tja! Wir wieder!), sind wir bitter aufgelaufen. „Bringt auch mal was für den 128er!" Gewünscht, getan – seit Anfang dieses Jahres gibt es die Rubrik **128er Tools**. Nach dem ersten Erscheinen dieser Rubrik hagelt es Proteste von den „Nur-64ern" (das „Nur" ist nicht abwertend gemeint), die sich benachteiligt fühlen.

Grundtenor der Kritik: Ihr liefert ein Programm, mit dem wir überhaupt nichts anfangen können. Spontane Antwort: Recht haben sie. Obendrein stellen sie nach wie vor die Mehrheit unserer Leser; und das wird, legt man die auf der CeBIT von Commodore veröffentlichten Zahlen zugrunde, auch so bleiben (Näheres auf Seite 4 in dieser Ausgabe).

Erstes, vorsichtiges und etwas schwaches Gegenargument: Wir haben uns bemüht, den Umfang der „reinen" 64er Software nicht zu verringern. Trotzdem können wir aber gerade den Unmut von neuen Lesern, die unsere für den C128 umgeschriebenen C64-Programme noch nicht haben, verstehen. Wichtiger aber: es können ohnehin nicht alle Programme auf der Diskette für alle Leser gleichermaßen interessant sein. Ist für die einen Spiel XY das einzig Interessante im Magazin, finden die anderen Computerspiele sowieso öde, sind aber von der Anwendung ABC total begeistert. Dies relativiert meines Erachtens die Einwände der „Nur-64er".

Da aber, um noch einmal einen Griff in die Sprichwort-Kiste zu wagen, „das Bessere der Feind des Guten ist", gilt ab sofort folgendes: Wir bemühen uns, zusammen mit den 128er-Tools auch jeweils die 64er-Fassung der jeweiligen Programme zu veröffentlichen. Für diese Ausgabe heißt das: es gibt einen Makro-Assembler für den C128 und einen – überarbeiteten – Makro-Assembler für den C64. (Daß diese nicht unter der Rubrik „Tools" erscheinen, hängt natürlich damit zusammen, daß ein Makro-Assembler den mit dem Begriff „Tool" umrissenen Rahmen bei weitem sprengt.)

Dieses Veröffentlichungskonzept läßt sich natürlich nicht immer durchhalten (denken Sie beispielsweise an einen Disk-Monitor für die 1571 oder ein 80-Zeichen-Grafikpaket), aber wir versuchen wie immer unser Bestes und hoffen, es ist gut genug.

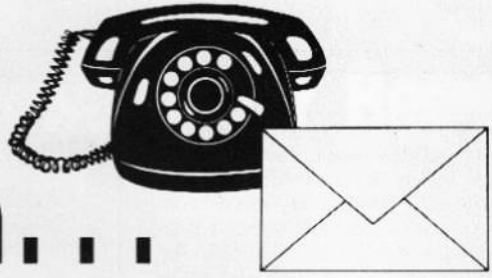
Jürgen Seeger

J. Seeger

5/88

INHALT

Leser fragen...



UniDat überformatiert

... für mich ist die Angelegenheit eigentlich schon gelaufen. UniDat (Ausgabe 12/87) wurde wegen Unfähigkeit feierlich zum Tode durch Formatieren verurteilt. (...) Man soll zwar Toten nichts Schlechtes nachsagen, aber es lebt wohl noch in unzähligen Kopien. Also:

1. Beim Beginn einer neuen Datei muß der erste Datensatz zweimal eingegeben werden. Die erste Eingabe verschwindet nämlich irgendwo auf Nimmerwiedersehen.

2. UniDat beschreibt gelöschte Records nicht wieder. Sie stehen leer mit CHR$(255) im File herum. Das bedeutet demnach, daß eine Datei, in der laufend Datensätze ausgewechselt werden, effektiv immer kleiner wird!

3. Die „Bandbreite" des Index von nur zwei Zeichen ist ein Witz, und zwar ein schlechter!

Dabei kann der Autor dieses Programms zweifellos mehr als ich. Die Eingaberoutine ist hervorragend gemacht! (...) Daß man Datensätze, die länger als 80 Zeichen sind und nicht mehr mit INPUT# einlesen kann, nicht unbedingt, wie es in allen Büchern steht, mit GET#-Schleife einlesen muß. Der Trick, den Satz mit CR's zu unterteilen und ihn dann mit mehreren INPUT# zu lesen, für diesen Tip danke ich vielmals!

H. Carstensen, Senden

Sie sollten vielleicht doch Ihre Original-INPUT 64-Diskette wieder hervorkramen und UniDat erneut auf Ihre Arbeitsdiskette kopieren, denn die von Ihnen aufgeführten Gründe sind nicht stichhaltig. Zu:

1. Hier muß wirklich ein Bedienungsfehler Ihrerseits vorliegen, wir haben uns die größte Mühe gegeben, aber den aufgeführten Fehler konnten wir nicht erzeugen.

2. Natürlich benutzt UniDat gelöschte Records wieder, aber intelligenterweise erst dann, wenn keine freien Eintragsmöglichkeiten mehr vorhanden sind.

3. Über die „Bandbreite" einer Index-Datei läßt sich trefflich streiten. Hier muß ein Kompromiß zwischen Speicherplatzbedarf im Rechner und erster Zugriffsgenauigkeit gefunden werden. Die von UniDat benutzte Breite von zwei Zeichen bedeutet ja nicht, daß Unterscheidungen im dritten oder fünften Zeichen nicht mehr möglich sind. Theoretisch können Sie natürlich auch die gesamte Feldbreite als Index-Datei führen, nur müssen Sie dann bei größeren Dateien einen weiteren C64 „danebenstellen". Für die meisten Anwendungen dürften die eingestellten zwei Zeichen ausreichen. (d. Red.)

Unidat angepaßt

Das Problem ist mein Drucker (ich habe einen CP-80 Typ II von C.T.I. mit Görlitz-Epson-Interface), der die hardwaremäßge Einstellung eines Auto-Line-Feed nicht zuläßt und zur Einstellung des Interface eine Sekundäradresse höher als 7 benötigt. Das Programm weigert sich aber standhaft, eine solche Adresse zu akzeptieren.

D. Krause, Bremen

Der CP-80-Drucker läßt unseres Wissens die Einstellung eines Auto-Line-Feed per DIP-Schalter zu. Die Interface-Einstellung kann durch ein Ladeprogramm erfolgen, das beispielsweise so aussehen könnte:

```
10 OPEN 4,4,Sekundäradresse
20 PRINT#4,Sende-Daten
30 CLOSE 4
40 LOAD"UniDat",8,0
```

Dieser kleine Trick funktioniert natürlich auch bei anderen Programmen. (d. Red.)

Noch einmal Ist BASIC

Vorweg ein großes Lob zu Ihrem Programm „Ist BASIC", das im Zusammenwirken mit „INPUT-BASIC" eine wirklich brauchbare BASIC-Erweiterung ist. „Ist BASIC" bedarf jedoch in folgenden Punkten noch der Korrektur. 1. Das mitgelieferte Ladeprogramm funktioniert trotz der im Heft 3/88 vorgeschlagenen Ergänzung mit "POKE 2267,96" nicht. 2. Die Befehle KEYLABEL, SHOW, WHEN Label THEN funktionieren nicht ordnungsgemäß. Es kommt zwar mit KEYLABEL ft, Label zur Belegung der Funktionstasten, was mit SHOW auch überprüft werden kann. Eine Abfrage der Funktionstasten erfolgt jedoch nicht. *G. Blank, Trier*

Vielen Dank für das Lob. Ist BASIC muß aber nicht, wie Sie vermuten, nachgebessert werden. Die von Ihnen angesprochenen Ladeprobleme sind – nach durchgeführter Änderung (siehe Ausgabe 3/88) – wohl nur noch auf Eingabefehler beim File-Namen zurückzuführen (zum Beispiel Nichtberücksichtigung der Groß- und Kleinschreibung); das Programm funktioniert jetzt jedenfalls tadellos. Ihre Schwierigkeiten mit den aufgeführten neuen Befehlen sind verständlich, da das Beispielprogramm

Dienstag ist Lesertag

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

im Beiheft schlichtweg falsch war. Das kleine Listing auf Seite 7, Spalte 2 muß richtig heißen:

```
10 KEYLABEL 1,"Hase"
20 WHEN "Hase" THEN GOTO 300
30 GOTO 20
```

Jetzt ist die Sache doch gleich viel logischer, oder? (d. Red.)

SANDI abspeichern

Wie kann ich denn bloß die Funktionen von SANDI anwählen? (mehrere telefonische Anfragen)

Erst mal gar nicht! SANDI, unser Sound-Sampler und -Sequenzer (Ausgabe 3/88), beansprucht den gesamten Speicher des C64. Innerhalb von INPUT 64 konnten wir Ihnen deshalb nur eine Vorführung anbieten. Erst wenn Sie SANDI mit CTRL-S auf einen eigenen Datenträger abgespeichert haben, können Sie alle Funktionen benutzen. Leider ist genau dieser Hinweis im Beiheft der Schere zum Opfer gefallen. Um das Programm kennenzulernen, sollten Sie auch die Beispiel-Sounds aus INPUT 64 mit CTRL-S auf eine eigene Diskette übernehmen.

Noch 'n paar Quellenhinweise: Die Platine zur Multifunktionskarte, auf der auch ein 8-Bit-Sampler aufgebaut werden kann, erhalten Sie beim Heise-Platinenservice, die Bauanleitung wurde in unserer Schwesterzeitschrift c't (Ausgabe 9/86) veröffentlicht. Bausatz und Fertiggerät können auch bei der Firma Soundlight in Hannover bezogen werden. (d. Red.)

INPUT jetzt mit 2400 Baud

Die CosmoNet-Mailbox ist jetzt auch mit 2400 Baud erreichbar. Bereits in der letzten Ausgabe haben wir Ihnen mitgeteilt, daß auch INPUT 64 in der Datenbank vertreten ist. Wenn Sie also mit der Redaktion direkt Kontakt aufnehmen wollen, hinterlegen Sie doch einfach eine Mitteilung am „Schwarzen Brett".

300 Baud: Tel.: 05 11 / 3 38 02 52
1200 Baud: Tel.: 05 11 / 3 38 02 55
2400 Baud: Tel.: 05 11 / 3 38 02 10 (!)
Datex-P-NUA: 45 5110 90835

Die Terminal-Einstellung folgt mit „8 N 1" dem Standard. (d. Red.)

C128-Analyser mit kritischen Zeitverhalten?

Ganz sicher ist der in 3/88 veröffentlichte „C128-Analyser" eine wertvolle Hilfe – wenn er läuft. Und bei mir tat er es in der Originalversion nicht. Im Step-Modus war die Reaktion auf die Taste '' gleich Null. Nach etlichen Stunden Sezierarbeit war die Schwachstelle gefunden:*

```
$27A1  A2 1D     LDX #$1D
...
$27A5  8E 04 DC  STX $DC04
```

Hier wird der Timer A auf Low gesetzt. Der Wert ist sehr kritisch, und das Zeitverhalten ist bei verschiedenen Computern scheinbar nicht immer gleich. Jedenfalls lief alles bestens nach Änderung des Wertes in $27A2 auf $1E. Da wahrscheinlich andere Freaks ähnliche Probleme haben dürften, wäre ein Hinweis in der kommenden Ausgabe angebracht. Beispiel:

```
RUN-Version
DLOAD"ANALYSER":POKE 8122,X
BSAVE"@ANALYSER"

BOOT-Version
BLOAD"ANALYSER":POKE 10146,X
BSAVE"@ANALYSER", P 9216 TO P 11041
```

Den Wert für 'X' muß man ausprobieren. Mit Sicherheit werden es Werte um 30 sein. B. Röske, Duisburg

Auf unseren C128ern läuft das Original zuverlässig. Wir können daher die Angaben nicht nachprüfen, geben die Information aber dennoch gerne weiter. (d. Red.)

Halle 7, Stand C 57

Auf unserem CeBIT-Stand haben wir viele interessante Gespräche führen können, die sicherlich Einfluß auf die künftige Redaktionsarbeit haben werden.

Wir wurden zum Beispiel von einem Besucher darauf hingewiesen, daß unser Assembler (Ausgabe 6/86) beim Assemblieren auf Diskette „großzügigerweise" ein Byte mehr abspeichert, als Objekt-Code berechnet wird. Da parallel zu dieser Mitteilung gerade der neue INPUT-ASS (erscheint in dieser Ausgabe) in der Entwicklung war, konnte dieser bisher unbekannte Bug auch gleich beseitigt werden.

Den „Hilferuf" von Herrn Drescher aus Fulda müssen wir an unsere Leser weiterleiten. Herr Drescher sucht für das Programm INPUT-Calc 64/128 (Ausgabe 8/87) eine Druckeranpassung für den Drucker Epson LX 800 mit Wiesemann-Interface – und zwar für den NLQ-Modus. An dieser Stelle unsere Bitte: Sollten Sie eine Druckeranpassung für INPUT-Calc (oder auch INPUT-CAD) erstellt haben, schicken Sie uns diese bitte, damit wir sie durch eine Veröffentlichung allen Anwendern zukommen lassen können. (d. Red.)

News

Spitzenreiter C64

Der C64 ist nach wie vor der meistverkaufte Homecomputer in der Bundesrepublik. Wie Commodore-Pressesprecher Gerold Hahn im Pressegespräch mit INPUT 64 auf der CeBIT mitteilte, wurden von Juli '87 bis Februar '88 insgesamt 245 000 Rechner im Homecomputer-Bereich ausgeliefert. Mit deutlichen Abstand vorn liege der C64 mit 130 000 Stück, der Amiga 500 folge mit 80 000, der C128D mit 35 000 Exemplaren. Damit bestreite Commodore 76 % des HC-Marktes.

Zu den Akten gelegt seien die Pläne, einen C64D mit integriertem Diskettenlaufwerk zu produzieren (wir berichteten darüber). Grund: zu hohe Herstellungskosten und keine Einigkeit über die Frage „3 1/2-Zoll- oder 5 1/4-Zoll-Floppy?". Der C128 wird nur noch als C128D ausgeliefert, die Floppy 1571 als Einzelgerät soll langfristig vom Markt genommen werden. Zu den Fußball-Europameisterschaften im Juni werde noch einmal der „alte" C64 im runden Gehäuse, zusammen mit einem Spiele-Modul, in den Handel gebracht. JS

Ein-Hand-Führer

Die „DATA BECKER Führer" mit dem Untertitel „Alles auf einem Blick" gibt es jetzt auch für den C64 und C128. Auf jeweils gut 200 Seiten werden Befehle, Fehlermeldungen, Hardware-Spezifikationen und die nötigen Interna in Form eines kleinen Nachschlagewerkes dargestellt.

Beide Bände behandeln gleichzeitig die zugehörigen Diskettenstationen – 1541 beziehungsweise 1571 –, der C64-Band geht bei einigen Themen (BASIC-Interpreter) deutlich mehr in die Tiefe; schließlich muß man hier nicht auch noch mit CP/M ein völlig eigenes Betriebssystem mit abhandeln. Mit dieser Reihe im handlichen Format von 20*12 cm will der Düsseldorfer Verlag „die überflüssigen Bücherberge" vom „Schreibtisch verbannen".

Die DATA BECKER FÜHRER:

Dieter Vüllers	Heinz Wrobel
Commodore 64	Commodore 128
Düsseldorf 1987	Düsseldorf 1987
ISBN 3-89011-413-X	ISBN 3-89011-414-8
19,80 DM	19,80 DM

Grafik greifbar gemacht

HCS: Hardcopy-System und Grafik-Monitor

Ein Bild so zu Papier zu bringen, wie es auf dem Bildschirm erscheint, ist beim C64 angesichts der Vielfalt der Grafik-Modi eh schon keine ganz leichte Aufgabe. Wenn dazu noch die Möglichkeit kommt, den Speicher nach Bildern und Texten in jedem beliebigen Ablageformat zu durchsuchen, ist die hochtrabend klingende Bezeichnung „Hardcopy-System" schon beinahe nobles Understatement.

Das Hardcopy-System ist eine unabhängige Programmerweiterung für IMC-BASIC. Das ganze System enthält mehrere Programme, die teilweise in IMC-BASIC geschrieben sind, meistens aber auch unabhängig von IMC-BASIC verwendet werden können. Innerhalb von INPUT 64 können Sie die für Ihren Drucker notwendigen Parameter einstellen und die einzelnen Files auf eine Arbeitsdiskette abspeichern. Von dort können diese dann einzeln geladen und benutzt werden. Das Hardcopy-Paket enthält die folgenden Programme:

Tabelle 1:

ab Adresse	7-Nadel-Drucker	8-Nadel-Drucker	IMC-BASIC
$9000/36864	HC 7 $9000 (1)	HC 8 $9000 (2)	V2 ($E0)
$C000/49152	HC 7 $C000 (3)	HC 8 $C000 (4)	V1 ($A0)

Tabelle 1: So liegen die verschiedenen Versionen des Hardcopy-Systems im Speicher.

- Hardcopy-Routine für 7-Nadel-Drucker
- Hardcopy-Routine für 8-Nadel-Drucker
- Druckeranpassung (für beide Routinen)
- „I/O-Rettungs"-Routine
- Kopierroutine
- Diverse BASIC-Programme zum Darstellen von Multicolor-Grafiken

Das Kernstück des HCS sind natürlich die Multicolor-Hardcopy-Routinen. Sie sollen, im Gegensatz zu den in IMC-BASIC eingebauten Routinen, ein originalgetreues Abbild des Bildschirms und der Farben liefern. Dazu werden die 16 Farben in Graustufen zerlegt und ausgedruckt.

Gut Ding braucht Weile

Die Routine benötigt etwa 15 Minuten Zeit für eine Hardcopy. Dies liegt zum einen an der Farbzerlegung, zum anderen an den vielen weiteren Abfragen, die während des Druckvorgangs vorgenommen werden. Neben dem Multicolor-Bildschirm druckt das Programm auch genauso alle anderen Bildschirmdarstellungen (HiRes-Bitmap, Text-Modus, Multicolor-Text-Modus, Extended Colour-Modus), ebenso alle Sprites in allen Größen, Überlappungen und Prioritäten.
Selbstverständlich berücksichtigt das Programm auch neue Zeichensätze, geänderte Bildschirmadressen, andere VIC-Adressierungsbereiche und und und . . .

Die Hardcopy-Routinen

Es liegen insgesamt vier Versionen vor (siehe Tabelle 1).
In der letzten Spalte der Tabelle steht, mit welcher der beiden IMC-BASIC-Versionen die jeweilige Hardcopy-Routine zusammenarbeitet. Werden die beiden ersten Hardcopy-Programme mit IMC-BASIC (oder auch mit dem normalen BASIC) verwendet, muß der Bereich ab $9000 vor den Strings geschützt werden. Dies geschieht mit den Befehlen POKE 56,155:CLR. Die Hardcopy-Routine nimmt ihre Arbeit nach SYS 36864 (oder SYS 49152, im folgenden Text verallgemeinernd mit SYS STARTADRESSE bezeichnet) auf und erstellt vom aktuellen Bildschirm eine wunderschöne Kopie.
Der Aufruf SYS STARTADRESSE+3 bindet eine kleine Routine in den System-Interrupt ein, die regelmäßig die Tastenkombination CTRL-← abfragt. Dadurch kann während eines laufenden Programms durch Tastendruck der Ausdruck eingeleitet werden.

Sieht alles, versteht alles

SYS STARTADRESSE+6 startet einen ganz kleinen Mikro-Mini-Grafik-Monitor. Mit ihm kann man den Speicher nach Bildern und Texten durchsuchen. Die erste Benutzung dieses Programms ist etwas irritierend: es gibt nämlich nach dem Start keine Meldung aus oder zeigt sonst irgendwie an, daß es arbeitet. Erst wenn die unten näher beschriebenen Tasten gedrückt werden, wird seine Arbeit augenfällig.

Bekanntlich kann der Video-Chip (VIC) „von Haus aus" nur auf 16 KByte (1/4 des C64-Speichers) zugreifen. Da dies die Möglichkeiten des VICs erheblich einschränken würde, eilt ihm der CIA (Complex Interface Adapter) zu Hilfe. Er verhilft ihm zu zwei weiteren Adreßleitungen, damit der VIC (fast) im ganzen Speicher schalten und walten kann.
Diese einzelnen 16K-Bereiche können im Monitor mit den Tasten '1' bis '4' angesprochen werden.

Taste	Bereich
'1'	$0000–$3FFF
'2'	$4000–$7FFF
'3'	$8000–$BFFF
'4'	$C000–$FFFF

Die einzelnen Grafik-Modi des VIC werden mit den Tasten
't' für den Text-Modus,
'h' für den Bitmap-Modus (Einzelpunktgrafik) und
'e' für den meist vernachlässigten Extended Colour-Modus angesprochen.
Der Multicolor-Modus wird mit 'm', der Normal-Modus mit 'n' aufgerufen.
Des weiteren sind noch einige Tasten nötig, um innerhalb des 16K-Bereichs die einzelnen Startadressen für Bildschirme und Zeichensätze zu verschieben.

Listing 1: So werden unter IMC-BASIC alle relevanten Bild-Informationen gerettet.

```
60000 rem speichern eines bildes        / Version 1 ($a000)
60010 rem
60020 tb = 35840 [V2: 52224]            :rem textbildschirm
60030 n$="name"
60040 mcon                              :rem wichtig !
60050 poke tb+1000,peek(53281)          :rem hintergrundfarbe retten
60060 gsave "gr."+n$,8
60070 msave tb,tb+1001,"tb."+n$,8       :rem textbildschirm speichern
60080 msave 55296,56320,"fr."+n$,8      :rem farbram speichern
60090 mcoff
```

```
60200 rem laden eines bildes          / Version 1 ($a000)
60210 rem
60220 tb = 35840 [V2: 52224]          :rem textbildschirm
60230 n$="name"
60240 mcon                            :rem wichtig!
60250 gload "gr."+n$,8
60260 mload tb,"tb."+n$,8             :rem textbildschirm laden
60270 poke 53281,peek(tb+1000)        :rem hintergrundfarbe
60280 mload 55296,"fr."+n$,8          :rem farbram laden
60290 mcoff
```

Listing 2: Auf die richtige Plazierung kommt es an — beim Laden müssen alle Informationen an die richtige Adresse.

'+' erhöht, '-' erniedrigt die Startadresse des Textbildschirms um 1024. Da der VIC nur auf den ihm vorgesetzten 16K-Speicher zugreifen kann, wiederholen sich die Textbildschirme nach dem sechzehnten Drükken auf '+' oder '-'.

'CRSR down' erhöht, 'CRSR up' erniedrigt die Startadresse des Zeichensatzes um 2048. Diesmal wiederholen sich die Zeichensätze schon nach dem achten Mal. Die Zeichensätze können auch direkt über SHIFT-1 bis SHIFT-8 angesprochen werden.

Mit dem '*' kann man zwischen den zwei Grafikseiten (es passen ja nur zwei in den 16K-Bereich) umschalten.

Die letzten drei Tasten fallen unter die vielsagende Kategorie „Sonstiges".

'f' erhöht die Hintergrund- und Rahmenfarbe um eins.

'q' wie „Quit" ermöglicht das Verlassen des Monitors.

SHIFT + 'h' startet den Ausdruck des aktuellen Bildschirms.

Aber bitte mit Farbe!

'l' lädt eine Datei namens „IO" von der Diskette. „IO" wird vom „IO-Rettungs"-Programm erstellt, diese Datei enthält neben allen VIC-Registern auch das Farb-RAM, so daß zum Beispiel ein Bild mit Sprites auch nach einem Reset, bei dem bekanntlich derlei Informationen verlorengehen, rekonstruiert werden kann. Das entsprechende Programm zum Abspeichern des Farb-RAMs ist ebenfalls ein Teil des HCS. Angenommen, Sie wollen ein schönes Bild aus einem Programm ausdrucken, so laden Sie zunächst dieses kleine „IO-Rettungs"-Programm und starten es. Anschließend kann das Programm mit dem Bild geladen und das Bild dargestellt werden (das Bild muß auf dem Bildschirm erscheinen, da sonst die zugehörigen Daten nicht abgespeichert werden können).

Die „IO"-Datei wird abgespeichert, wenn die CTRL-, die SHIFT- und die C=-Taste gleichzeitig gedrückt werden oder durch Betätigen von RUN-STOP/RESTORE (NMI) oder, falls auch dies nicht möglich ist, indem ein Reset ausgelöst wird. (Führen alle drei Möglichkeiten nicht zum Erfolg, so hat das nachgeladene Programm leider alle Vektoren überschrieben, und das Bild läßt sich nicht ausdrucken.) Der IO-Retter speichert den Inhalt des VIC und des Farb-RAMs auf Diskette ab.

Anschließend kann die Hardcopy-Routine (vorzugsweise Version 1 oder 2) geladen

Listing 3: Mit IMC-BASIC und einem kleinen Maschinenprogramm kommt man auch den vertracktesten Formaten auf die Spur.

```
1 poke 55,0 : poke 56,64 : clr        :rem basic-ende auf $4000/16384
...
60000 rem laden eines zusammengesetzten bildes / Version 1 ($a000)
60010 rem
60020 gr = 40960 [V2: 57344]          :rem grafikbildschirm
60030 tb = 35840 [V2: 52224]          :rem textbildschirm
60040 fr = 55296                      :rem farbram
60050 sa = 16384                      :rem bereich, an den geladen
                                      wird
60060 n$="name des Files"
60070 mcon                            :rem wichtig!
60080 mload sa,n$,8
60090 sys 900,sa,sa+8000,gr           :rem grafikbildschirm
60100 sa=sa+8000
60110 sys 900,sa,sa+1000,tb           :rem textbildschirm
60120 sa=sa+1000
60130 sys 900,sa,sa+1000,fr           :rem farbram
60140 sa=sa+1000
60150 poke 53281,peek(sa)             :rem hintergrundfarbe
60160 get a$ : if a$="" then 60160
60170 mcoff
```

und der Grafik-Monitor aufgerufen werden. Nachdem die Datei mit 'l' geladen wurde, muß nur noch der 16K-Bereich ausgewählt werden – der Reihe nach die Tasten '1' bis '4' ausprobieren! Irgendwo taucht das Bild dann schon auf. SHIFT + 'h' druckt das Bild jetzt aus.

Fremdgehen ist möglich

Bilder, deren Daten sich bereits auf einer Diskette befinden (etwa Bilder aus Malprogrammen), lassen sich mit IMC-BASIC problemlos weiterverarbeiten. Wir haben dazu ein paar kleine Programmbeispiele für diese Spracherweiterung entwickelt (s. Kästen).

In Zeile 60020 von Listing 1 ist in eckigen Klammern die Startadresse des Textbildschirms für die Version 2 ($E000) angegeben. Ändert man die Zeile in 'TB = 52224', ist das Programm genauso für die Version 2 gültig. Das Programm erzeugt folgende Programme auf der Diskette:

"GR.TEST" Grafikdaten
"TB.TEST" Textbildschirm + Hintergrundfarbe
"FR.TEST" Farb-RAM

In ähnlicher Weise können auch Bilder aus Malprogrammen geladen werden, die ebenfalls dreigeteilt auf der Diskette vorliegen (siehe Listung 2).

Schwieriger ist es bei Programmen, die ein einziges File erzeugen. Die einzelnen Datenbereiche sind dann im Speicher an eine bestimmte Stelle verschoben und gemeinsam abgespeichert worden. Um die Daten wieder zurückzuverschieben, enthält das HCS auch ein kleines Kopierprogramm. Es muß unter IMC-BASIC absolut geladen werden und kann dann mit

SYS 900,startadr,endadr+1,zieladr

aufgerufen werden. Listing 3 zeigt die Einbindung in BASIC.

Das Beispiel kann nur als Anregung für eigene Programme dienen. Da fast jedes Malprogramm seine Daten anders abspeichert, wäre es ein großer Zufall, wenn dieses Programm auf Anhieb funktionieren würde. (In der April-Ausgabe unseres Magazins wurden im Zusammenhang mit dem Programm MIKROPROJEKTOR 64 diverse Formate dokumentiert.) Falls es nicht klappt, hier ein paar Möglichkeiten, wie man das Programm abändern könnte:

60100 sa=sa+8192 Manche Programme gehen etwas verschwenderisch mit dem Speicherplatz um.
60120 sa=sa+1024 siehe oben
60140 sa=sa+1024 siehe oben
60150 poke 53281,peek(?) Die Hintergrundfarbe kann eigentlich überall liegen.

Außerdem könnten die einzelnen Bereiche (Bitmap, Textbildschirm und Farb-RAM) beliebig vertauscht sein.

Druckeranpassung

Die Druckeranpassung wird vor dem Abspeichern innerhalb von INPUT 64 aufgerufen, indem man bei der gewünschten Hardcopy-Routine RETURN drückt. Hier kann man der Reihe nach eingeben, welche Steuercodes an den Drucker gesendet werden sollen. Die Eingaben werden ohne Kennzeichen dezimal, mit führendem '$' hexadezimal und mit führendem '"' als ASCII-Zeichen aufgefaßt.

Funktionstaste F7 springt zum nächsten Code, mit F5 geht's zurück. Dies ist jedoch nur dann erlaubt, wenn die letzte Eingabe korrekt war. '260' ist beispielsweise nicht erlaubt, da die Zahl größer als '255' ist und nicht mehr durch ein Byte darstellbar ist. Am unteren Bildschirmrand sind zwei fertige Druckeranpassungen (STAR NL-10 und MPS801) mit RETURN abrufbar. Die Druckeranpassung wird mit RUN/STOP beendet.

Die 7-Nadel-Routinen sind speziell für die weit verbreiteten Drucker MPS801/MPS803 von Commodore entwickelt worden. Da diese keine Grafik in doppelter Dichte drucken können, wird das Bild um 90 Grad gedreht. Für einen Ausdruck stehen so 400*640 Punkte zur Verfügung. In der Anpassung für diese Drucker haben die Werte für 1/216 Inch und 27/216 Inch Zeilenvorschub keine Bedeutung mehr.

Dunkel, hell und bunt

Das Programm, das innerhalb von INPUT 64 läuft, erlaubt unter anderem den Aufruf des Grafik-Monitors. Dieser Aufruf erstellt zuvor eine Farbbalkengrafik auf dem Bildschirm. Besitzer eines Farbmonitors oder Farbfernsehers werden sich sicher wundern, was die scheinbar zufällig zusammengewürfelten Farben darstellen sollen. Wird der Farbkontrast ganz zurückgedreht, erkennt man, daß die Farben der Helligkeit nach geordnet sind. Die gleiche Farbenfolge benutzt auch das Hardcopy-Programm, um die Farben in Graustufen zu zerlegen. Beginnend mit der dunkelsten Farbe ergibt sich folgende Reihenfolge der Farben und zugehörigen POKE-Werte:

Schwarz/0, Braun/9, Blau/6, Dunkelgrau/11, Rot/2, Violett/4, Orange/8, Mittelgrau/12, Hellblau/14, Grün/5, Hellrot/10, Hellgrau/15, Zyanblau/3, Gelb/7, Hellgrün/13, Weiß/1

Ein paar Zahlen . . .

Das Programm sendet grundsätzlich 960 Grafik-Bytes pro Zeile an den Drucker. Bei 8-Nadel-Druckern muß diese Information beim Einschalten des Grafik-Modus übergeben werden. Jede Zeile wird dreimal gedruckt. Jede Farbe wird aufgelöst in eine 3*6-Punktmatrix. Das sind 3*6=18 Bits. Der Bildschirm des C64 ist 320 Pixel breit und 200 Pixel hoch. Damit ergeben sich 320*200=64 000 Bits pro Bildschirm. Es müssen also 64 000*18 = 1 152 000 Bits, 144 000 Bytes oder 141 KByte an den Drucker gesendet werden. Bei einem 7-Nadel-Drucker werden 400*640=256 000 Punkte gesetzt, pro Nadel also im Extremfall 36 800 Anschläge.

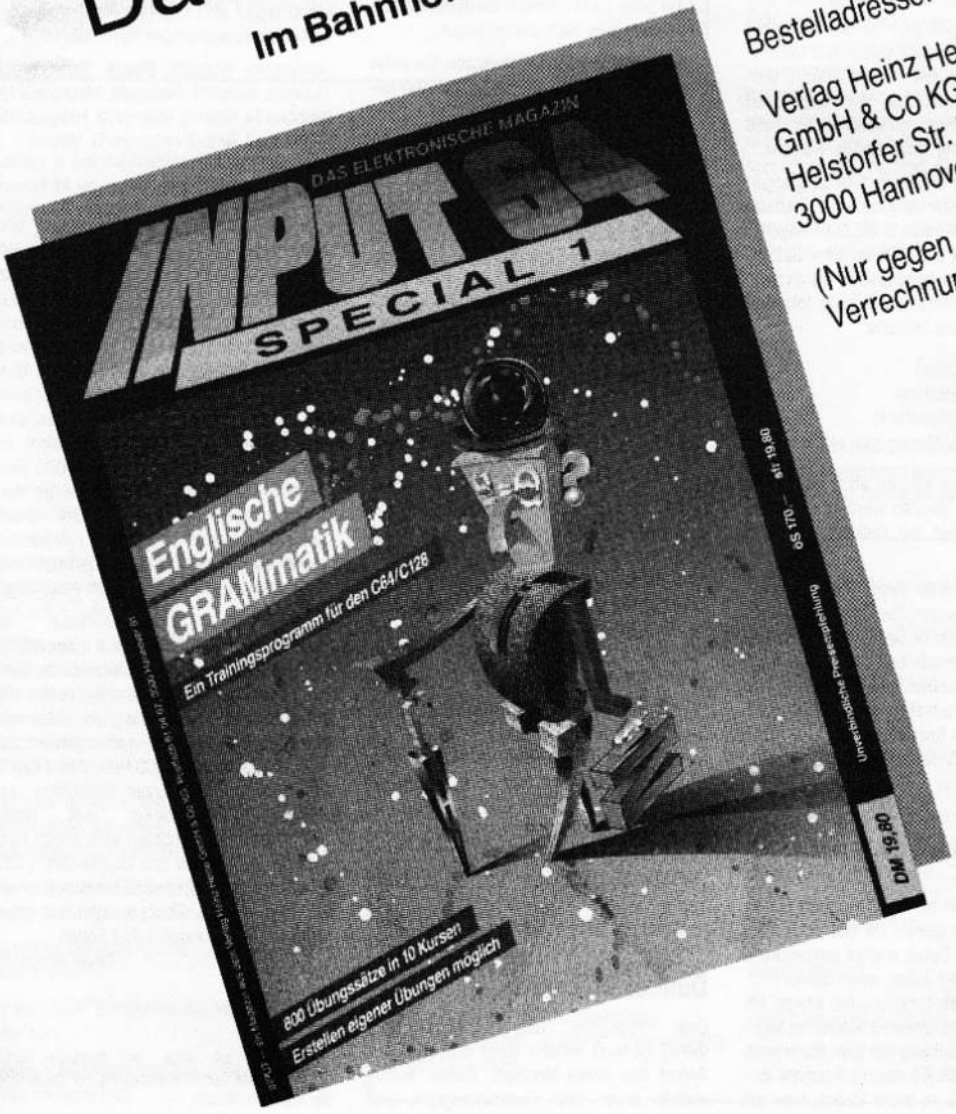
In meinem Druckerhandbuch ist angegeben, daß der Druckkopf 100 000 000 Zeichen drucken würde. Geht man davon aus, daß es sich hierbei um EDV-Schriftzeichen handelt, so haben die einzelnen Nadeln eine Lebensdauer von vielleicht etwa 200 000 000 Anschlägen. Da bei einem schwarzen Bildschirm und einem 8-Nadel-Drucker jede Nadel 144 000mal angeschlagen wird, ergibt sich, daß bis zum Exitus des Druckkopfs 1 388 schwarze Seiten gedruckt werden können. Für den 7-Nadel-Drucker sind es unter gleichen Bedingungen 5 434 Seiten.

Oliver Kraus/JS

'IMC-BASIC ist eine in Ausgabe 9/87 veröffentlichte Spracherweiterung zur Bedienung der Multicolor-Grafik.

Das Lernprogramm.

Im Bahnhofsbuchhandel und direkt beim Verlag.

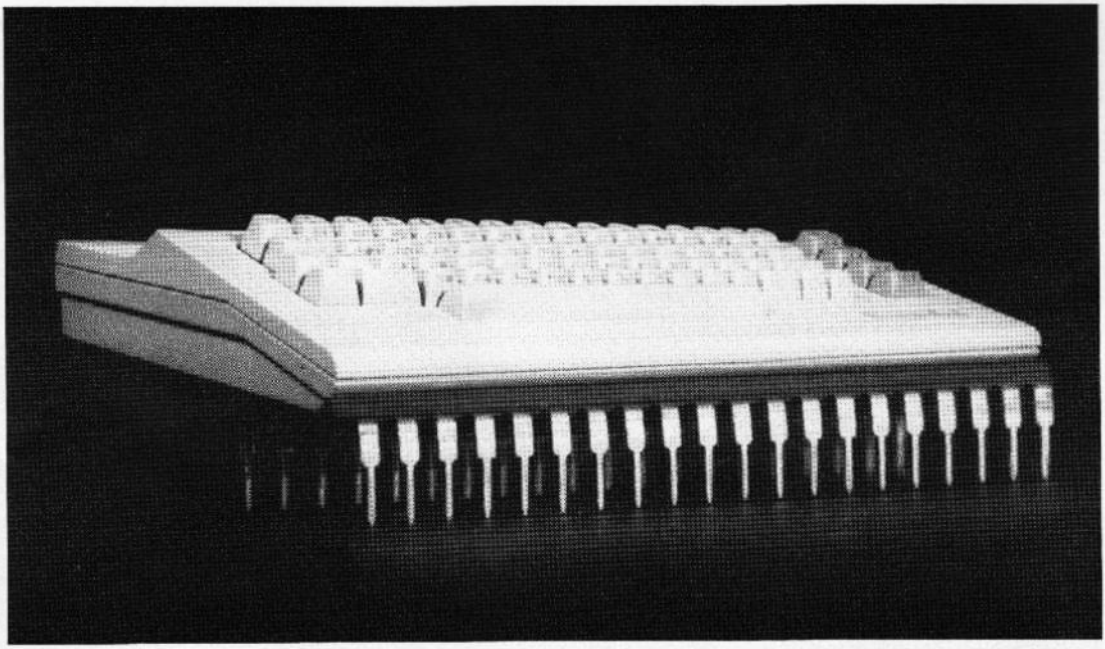


Bestelladresse:

Verlag Heinz Heise
GmbH & Co KG
Helstorfer Str. 7
3000 Hannover 61

(Nur gegen
Verrechnungsscheck)

HEISE



Ganz schnell wird's maschinell

Makro-Assembler/Editor für C64 und C128

INPUT-ASS-64 wird von der eigenen Diskette (Sie müssen ihn natürlich aus dem Magazin heraus zunächst abspeichern) wie ein BASIC-Programm mit LOAD"INPUT-ASS-64",8,0 geladen und mit RUN gestartet; die 128er-Version entsprechend mit DLOAD"INPUT-ASS-128" und RUN. Die C128-Version erkennt selbständig, ob der 40- oder 80-Zeichen-Bildschirm bedient werden will. Zwischen den beiden Bildschirmen kann ohne Textverlust hin- und hergeschaltet werden, aber nur außerhalb des Assemblers (also: CTRL-K/Q, ESC und "X", RUN).

Die Bezeichnung Assembler/Editor deutet schon darauf hin, daß es sich genaugenommen um zwei Programme handelt: den Texteditor zum Erstellen, Laden, Speichern und Ändern der Programmtexte (der natürlich auch für beliebige andere Texte benutzt werden kann) und den Assembler, der den Source-Code in Maschinenbefehle übersetzt (siehe auch „Wie das geht . . ."). Vorweg noch eine Bemerkung zur Schreibweise: alle Mnemonics, Pseudo-OPs und Fehlermeldungen (LDA, PHC, AM und so weiter) sind in diesem Text der Übersichtlichkeit halber in VERSALIEN geschrieben. Der Assembler erwartet aber grundsätzlich Kleinschrift.

Als INPUT-ASS ist dieser Assembler treuen Lesern unseres Magazins bereits seit Juni 1986 bekannt. Wir haben dieses für Maschinensprache-Programmierer unentbehrliche Werkzeug für den 128er angepaßt — mit 40/80-Zeichen-Modus wahlweise und voller Nutzung des größeren Speichers. Und damit die „Nur-64er" nicht zu kurz kommen, finden Sie auf Diskette auch einen INPUT-ASS-64, in den fast zweijährige Redaktions- und Lesererfahrung sowie „Debugging" einflossen.

Der Editor — beinahe eine Textverarbeitung

Der Editor dürfte denjenigen, die schon mit WordStar oder Turbo-Pascal gearbeitet haben, „irgendwie bekannt" vorkommen. Die Befehle sind tatsächlich weitgehend WordStar-kompatibel. Am einfachsten und

Was das ist, wie das geht und wozu das alles gut ist

INPUT-ASS ist ein 2-Pass-Makro-Assembler mit integriertem Editor und optionaler Speicher- oder Peripherie-Orientierung, der sowohl symbolische als auch berechnete Ausdrücke verarbeitet.

Klingt gut, nicht? Wenn Sie gerade Ihr hundertunderstes Assemblerprogramm schreiben, sowieso wenig Zeit haben und Begriffe wie Vorwärts-Referenz und Phase-Error im Schlaf erklären können, lesen Sie bitte nur die Kurzübersicht. (Wahrscheinlich lesen Sie dann nicht mal die, weil Sie zu den Leuten gehören, die Anleitungen erst nach dem fünften System-Crash zwischen den Comics hervorholen, oder?) Für alle anderen also:

Das Prinzip

INPUT-ASS verwandelt die für den Menschen verständlichen Befehle der Assemblersprache in für den Prozessor verständlichen 6502-Maschinencode. Aus dem Befehl JMP $FCE2 (Sprung zur Adresse 64738) wird die Byte-Folge $4C,$E2,$FC. Klartextbefehle wie JMP, LDA und so weiter werden übrigens „Mnemonics"[1] genannt. Nun sind – um gleich auf das Stichwort „Symbolverarbeitung" zu kommen – Adressen, Bytes und Bits genau das, was ein Computer gern verarbeitet. Da der Mensch aber den Dingen lieber einen Namen gibt, verarbeitet jeder vernünftige Assembler Symbole. Ein Befehl wie JMP RESET klingt doch wesentlich deutlicher als das tumbe JMP $FCE2.

Von „berechneten Ausdrücken" war oben die Rede. Das heißt einfach, daß der Assembler für Sie das tut, was er sowieso besser kann als Sie, nämlich mehr oder weniger komplizierte Rechenoperation durchführen. Wenn beispielsweise in die zehnte Reihe und fünfte Spalte des Bildschirms ein Zeichen geschrieben werden soll, kann man als Adresse angeben: 10*40+5+VIDEO. Natürlich muß dem Assembler vorher „mitgeteilt" werden, welchen Wert VIDEO hat, etwa durch die Zuweisung VIDEO=$400.

Wie erledigt ein Assembler nun seine eigentliche Arbeit, die Umwandlung von Programmtext in Maschinencode? Stellen Sie sich einmal vor, Sie seien der Assembler. Ihnen wird Zeile für Zeile Source-Text (nicht neudeutsch: Quelltext, besseres Wort für Programmtext) vorgehalten, den Sie in den entsprechenden Maschinencode verwandeln sollen. Das ist in solchen Fällen nicht besonders schwierig, wo es nur um die Zuordnung von Mnemonics wie LDA zum Code $A9 geht. Das kann man in einer „Kartei" (sprich Tabelle) zuordnen, und Zahlen werden ohnehin nur in ihre entsprechende Bit-Kombination umgewandelt. Labels (Sprungmarken, Einsprungspunkte für Unterprogramme und so weiter) sind dann kein Problem, wenn Sie zum Zeitpunkt des Aufrufs bekannt sind. Beispiel:

```
ORG $C000
:WAIT DEX
BNE WAIT
```

Die Anfangsadresse ist durch die ORG-Anweisung bekannt. Deswegen können Sie das Label WAIT eindeutig der Adresse $C000 zuordnen. Interessant wird es bei dem indirekten Sprung BNE WAIT. Sie – immer noch in der Rolle des Assemblers – müssen jetzt feststellen, an welcher Adresse WAIT liegt. Dies ist, werden Sie sagen, kein Problem, denn dem Symbol WAIT wurde bereits vor (!) dem Befehl BNE WAIT die Adresse $C000 zugeordnet. Richtig, Sie mußten sich diese Zuordnung nur merken. Darum jetzt der schwierige Fall:

```
ORG $C000
:BEGIN LDX #100
JSR WAIT
...
...
:WAIT DEX
BNE WAIT
RTS
```

Beim Interpretieren des Programmtextes stoßen Sie auf den Befehl JSR WAIT – und haben nicht den Schimmer einer Ahnung, an welcher Adresse sich WAIT befinden könnte, denn es taucht ja wesentlich später im Text auf. (Und heißt deswegen „Vorwärtsreferenz"!) Was tun? Die gängige Möglichkeit ist die: sich erstens die Stelle merken, an der das noch unbekannte Label auftauchte. Zweitens, so zu tun, als ob nichts gewesen wäre, und den Programmzähler um die richtige Länge erhöhen. Denn es ist bekannt, daß ein JSR-Befehl immer drei Bytes lang ist. Beim weiteren Durchsuchen des Assemblertextes stoßen Sie dann irgendwann auf das Label WAIT und merken sich natürlich dessen Adresse. (Dieses Merken, also die Zuordnung von Labels beziehungsweise Symbolen, nennt man übrigens fachgerecht „Generieren einer Symboltabelle").

schnellsten können Sie die Befehle kennenlernen, indem Sie alle ausprobieren.

Bildschirmaufteilung: Die oberste Zeile des Bildschirms ist die Statuszeile. Ein inverses Zeichen am linken Rand zeigt an, daß ein aus zwei Zeichen bestehender Befehl eingegeben wird. Die Hex-Zahl daneben gibt die Größe des noch freien Speichers an. Der Rest der Statuszeile ist vorgesehen für Eingaben, Statusmeldungen der Floppy und Fehlermeldungen des Assemblers. Der Text wird in den restlichen 24 Zeilen des Bildschirms dargestellt. Die Zeilen sind 80 Zeichen lang, durch horizontales Scrolling sind jeweils 40 von 80 Zeichen der Zeile sichtbar. Der Cursor wird invers angezeigt.

Texteingabe: Die maximale Zeilenlänge beträgt 79 Zeichen, danach sind keine Eingaben mehr möglich. Die Zeilen werden durch RETURN abgeschlossen. Bei der Eingabe gibt es zwei Möglichkeiten (durch ein Flag in der linken oberen Ecke angezeigt): Im Insert-Modus (I) werden die eingegebenen Zeichen automatisch eingefügt, im Overwrite-Modus (O) die unter dem Cursor liegenden Zeichen überschrieben.

Alle Befehle, die Peripherie-Operationen betreffen (Laden, Speichern, Drucken), sind blockorientiert. Das heißt, daß immer der

„Aber", werden Sie einwenden, „wann trage ich denn die echte Adresse von WAIT nach dem JSR-Befehl ein?" Sie ahnen es sicherlich schon: Sie müssen ein zweites Mal durch. Wenn Sie dann auf den Befehl JSR WAIT stoßen, sehen Sie in der Symboltabelle nach, ob es schon eine Zuordnung zu WAIT gibt. Wenn ja, wird die entsprechende Adresse eingesetzt. Wenn nicht, hauen Sie – nach wie vor in der Rolle des Assemblers – dem schlampigen Programmierer eine Fehlermeldung um die Ohren.

Das erklärt wohl auch den Begriff „2-Pass-Assembler": damit ist nichts anderes als das eben beschriebene zweimalige Bearbeiten des Programmtextes gemeint, so daß erst im zweiten Durchgang (Pass) Maschinencode erzeugt wird.

Die oben erwähnte „optionale Speicher- oder Peripherie-Orientierung" hat zu tun mit dem eingebauten Editor. Der Editor ist eigentlich ein eigenes Programm, nämlich eine zeilenorientierte Textverarbeitung. Ihr aus den oben erwähnten „Mnemonics", Zuweisungen und (hoffentlich!) Kommentaren bestehender Source-Code (wörtlich übersetzt: Quell-Code, also der vom Assembler zu verarbeitende Text) wird mit diesem Editor erstellt. Näheres zum Editor siehe unten, hier soll es nur um die Beziehung zwischen Editor und Assembler gehen.

Makros und Include-Files

Im Prinzip erwartet der Assembler den Text im Editor. Aber eben nur im Prinzip. Es gibt nämlich zwei Spezialfälle:

Erstens: Makros. Makros sind, theoretisch ausgedrückt, „Text-Ersatz-Anweisungen".

Ihr praktischer Nutzen besteht darin, ständig wiederkehrende Befehlsfolgen nicht immer neu eintippen zu müssen. Dazu wird einmal ein Makro definiert; diese Definition besteht aus

- dem Namen des Makros
- einem Befehl, der angibt: „Jetzt wird ein Makro definiert" (beim INPUT-ASS 'm' oder 'mf')
- gegebenenfalls der Anzahl der Parameter, die diesem Makro übergeben werden sollen
- der diesem Makro zuzuordnenden Befehlsfolge
- und, natürlich, einer Endemarkierung.

Findet der Assembler nach dieser Makro-Definition im Text den Namen dieses Makros, so „tut er so", als ob der vollständige Maschinencode erzeugt. Makros sind also keinesfalls ein Ersatz für Unterprogramme, die ja nur einmal im Code vorhanden sind und beliebig oft aufgerufen werden können.

Zweitens: Include-Files. Durch Include-Files können häufig benutzte Unterprogramme oder Systemdefinitionen bequem eingebaut werden. Außerdem ist dadurch die Länge von Source-Texten prinzipiell unbegrenzt. Include-Files sind Source-Texte, die während des Assemblierens von einem Datenträger Zeile für Zeile eingelesen werden. Der entsprechende Maschinencode wird im erzeugten Programm dort eingefügt, wo das Include-File aufgerufen wurde.

Damit hätten wir den einen Fall von „optionaler Speicher- oder Peripherie-Orientierung" abgehandelt, nämlich die Frage: Woher kriegt der Assembler seinen Text? Da drängt sich natürlich eine andere Frage geradezu auf: Wohin soll der Assembler mit dem erzeugten Code? Beim INPUT-ASS gibt es drei Möglichkeiten.

Erstens: Der Code wird direkt in den Speicher geschrieben. Das ist praktisch zum Austesten, geht aber nur, wenn der betreffende Speicherbereich nicht belegt ist. In den Assembler oder in den Textspeicher zu assemblieren, führt natürlich zu bösen Fehlern.

Zweitens: Der Code wird auf Diskette geschrieben. Geht im Prinzip immer.

Drittens: Es wird kein Code erzeugt. Dies kann zum Test auf syntaktische Fehler im Source-Code sinnvoll sein oder dann, wenn sich Drucker und Floppy nicht vertragen.

Das soll an Theorie erst einmal reichen. Alles Konkrete finden Sie in der Bedienungsanleitung und – wie immer – in der Praxis. Für alle, die Maschinensprache erst noch lernen wollen, sei hier noch einmal auf INPUT 64-Special 2/Maschinensprache hingewiesen, das unter anderem eine sechsteilige interaktive Assemblerschule enthält. JS

Dieser Zungenbrecher hat etwas mit „mnemonisch" zu tun, was in etwa „gedächtnisunterstützend" bedeutet.

aktuell markierte, invers dargestellte Block gedruckt oder abgespeichert wird. Dies kann natürlich auch der gesamte Text sein. Diese Blockorientierung erlaubt ein beliebiges Zusammenbauen und Auseinandernehmen des Programmtextes. Haben Sie zum Beispiel eine Eingaberoutine geschrieben, die als Unterprogramm für weitere Anwendungen geeignet ist – den entsprechenden Teil als Block markieren und abspeichern. Dieser Programmteil kann dann in andere Source-Texte wieder eingefügt werden. Oder Sie wollen nur bestimmte Teile ausdrucken – als Block markieren und über CTRL-K/CTRL-W (siehe unten) auf Geräteadresse 4 schreiben. INPUT-ASS unterstützt so die Möglichkeit, sich eine Programmbibliothek zusammenzustellen – Sie müssen dann nur noch entsprechend überlegt programmieren ...

Alles unter CTRLIe

Sämtliche Befehle werden über die Eingabe von CTRL und gleichzeitig irgendeines anderen Zeichens erreicht. CTRL meint dabei die CTRL-Taste, CTRL-V bedeutet beispielsweise, die CTRL-Taste gedrückt zu halten und gleichzeitig (!) ein kleines 'V' einzugeben. Mit Cursor-Zeile beziehungsweise Cursor-Position ist immer die Zeile/Position ge-

meint, in der sich der Cursor gerade befindet.
Bei allen Befehlen, die aus zwei Zeichen bestehen, kann das zweite Zeichen mit oder ohne CTRL eingegeben werden. So bedeutet CTRL-K/W: zuerst gleichzeitig die CTRL-Taste und 'K' drücken, anschließend 'W' oder CTRL und 'W'.

CTRL-V Insert-Taste geht auch; schaltet zwischen Insert- und Overwrite-Modus um.

CTRL-P erlaubt die Eingabe von Kontrollzeichen in den Text, die sonst als Befehle interpretiert würden. Kontrollzeichen werden invers dargestellt. Beispiel:

CTRL-P und anschließend CTRL-C gibt CTRL-C in den Text ein.

Cursor-Bewegungen

Die Tasten zur Bewegung des Cursors sind als Kreuz angeordnet (natürlich können auch weiterhin die Cursor-Tasten benutzt werden!):

```
    W     E     R
  A   S   D   F
    Z     X     C
```

CTRL-S	Cursor links
CTRL-D	Cursor rechts
CTRL-A	Wort links
CTRL-F	Wort rechts
CTRL-E	Cursor rauf
CTRL-X	Cursor runter
CTRL-W	Scroll up
CTRL-Z	Scroll down
CTRL-R	Seite rauf
CTRL-C	Seite runter
CTRL-I	Tabulator
CTRL-Q/S	zum Zeilenanfang
CTRL-Q/D	zum Zeilenende
CTRL-Q/R	zum Textanfang
CTRL-Q/C	zum Textende
CTRL-Q/B	zum Blockanfang
CTRL-Q/K	Blockende

Der Begriff „Block" wird unten noch einmal aufgenommen. Der Tabulator ist nur ein „Pseudo-Tabulator": er bewegt den Cursor in die Spalte des nächsten Wortanfangs der vorigen Zeile. Dies funktioniert natürlich nicht, wenn die Zeile über dem Cursor eine Leerzeile ist. Klingt komplizierter, als es ist – ausprobieren!

Einfügen und Löschen

DEL Zeichen links vom Cursor löschen. Falls der Cursor am linken Rand steht, wird die Cursor-Zeile mit der vorigen Zeile verknüpft.

CTRL-G Zeichen unter dem Cursor löschen.

CTRL-Q/Y Von der Cursor-Position bis zum Zeilenende löschen.

CTRL-Y Cursor-Zeile löschen. Vorsicht: Das Auto-Repeat kann hier verheerende Auswirkungen haben!

CTRL-Q/L macht alle Änderungen in der Cursor-Zeile rückgängig, sofern die Zeile noch nicht verlassen wurde. Vorsicht: CTRL-Y wird dadurch nicht rückgängig gemacht!

Block-Befehle

Ein Block wird invers angezeigt, die Cursor-Zeile aber immer normal. Es gibt immer nur einen aktuellen Block.

CTRL-K/A Gesamten Text als Block markieren. Empfehlenswert vor dem Abspeichern!

CTRL-K/B Cursor-Zeile wird als Blockanfang markiert.

CTRL-K/K Cursor-Zeile wird als Blockende markiert, das heißt als erste Zeile, die nicht zum Block gehört.

Falls die Markierungen zu einem Widerspruch führen (Blockende vor Blockanfang), wird der Blockanfang auf den Textanfang beziehungsweise das Blockende auf das Textende gelegt.

CTRL-K/C kopiert den markierten Block an die Cursor-Position, als Block ist danach die Kopie markiert; der zuvor als Block markierte Text bleibt unverändert.

CTRL-K/V verschiebt den Block an die Cursor-Position; der zuvor als Block markierte Text wird gelöscht. Sehr große Blöcke müssen in Portionen verschoben werden, da CTRL-K/V zuerst den Block kopiert und dann das Original löscht.

CTRL-K/Y löscht den Block **endgültig**.

Eingaben in der Statuszeile

Wenn die Eingabe eines File-Namens oder eines Suchwortes erwartet wird, erscheint eine Frage (zum Beispiel „LOAD:") und der Cursor. In der Statuszeile kann nur mit DEL editiert werden. Die Eingabe wird mit RETURN abgeschlossen und mit STOP abgebrochen.

File-Namen werden in einem speziellen Format eingegeben:

LOAD:82NAME

dabei ist 8 die Gerätenummer und 2 die Sekundäradresse. Diese Zahlenangaben sind hexadezimal einzugeben, Gerätenummer 8 und Sekundäradresse 15 ist also 8F. Der Name wird ohne Anführungsstriche eingegeben. Beispiele:

40: Printer (kein File-Name !)
10NAME: File „NAME" von Kassette lesen
11NAME: File „NAME" auf Kassette schreiben
82NAME: File „NAME" von Diskette lesen
83NAME: File „NAME" auf Diskette schreiben

Mit CTRL-D kann nach der Eingabe diese als Default (festgelegte Vorgabe) gespeichert werden. Dieser Default erscheint dann jedesmal, wenn diese Eingabe gemacht werden soll.

File-Befehle

CTRL-K/E Disk-Befehl senden. Beispiel:

DISK:8FS:TEXT

löscht das File „TEXT" auf der Diskette (8F für 8,15 !).

CTRL-K/F Directory einlesen. Die Ausgabe kann mit der Leertaste angehalten und mit der C=-Taste wieder fortgesetzt werden. Am Schluß muß eine Taste gedrückt werden, um wieder in den Editor zu kommen.

CTRL-K/R File einlesen. Beispiel:

LOAD:82ASM.S

liest das File „ASM.S" von der Diskette ein. Der Text wird an der Cursor-Position eingefügt und als Block markiert.

CTRL-K/S Adreßbereich abspeichern (nur C64!). Gefragt wird nach Anfangs- (FROM:) und Endadresse+1 (TO:) – vierstellige Hex-Zahlen gefordert – und dem File-Namen.

CTRL-K/W Block als File schreiben. Beispiel:

SAVE:82ASM.S

speichert den markierten Block unter dem

Namen „ASM.S" auf Diskette. Wenn es verdächtig schnell ging, haben Sie wahrscheinlich vergessen, den Block zu markieren, und die Warnung NO BLOCK übersehen . . .

CTRL-K/* schreibt den Editor mitsamt Text und Defaults auf die Diskette. Dies geht nur mit Diskette, da ein Daten-File geschrieben wird. Beim INPUT-ASS-128 wird durch diesen Befehl **nicht** der Source-Text mit abgespeichert, sondern nur der Assembler mit den Defaults. Mit diesem Befehl kann ohne Kopierprogramm eine angepaßte Version vom INPUT-ASS auf die Arbeitsdiskette geschrieben werden.

Nach den Disketten-Operationen wird in der Statuszeile der Disk-Status angezeigt.

File-Format

Der Editor speichert die Texte als SEQ-Files ohne jegliche Zusätze, das heißt, die Zeilen werden durch CR (Carriage Return, entspricht CHR$(13)) begrenzt. CHR$(0) ist verboten, da der Editor das als Endmarke verwendet. Zeilen sollten – CR inbegriffen – nicht länger als 80 Zeichen sein. Texte von Assemblern, die den BASIC-Editor verwenden (herzliches Beileid . . .), können so lesbar gemacht werden:

```
LOAD „SOURCE",8
OPEN 1,8,2,„DEST,S,W"
CMD 1:LIST
PRINT #1:CLOSE 1
```

Ein vorheriges POKE 22,35 erzeugt übrigens ein Listing ohne Zeilennummern. Außerdem befinden sich sowohl auf der im Verlag erhältlichen Source-Diskette (siehe Anzeige in diesem Heft) als auch auf INPUT 64-Special 2 verschiedene Source-Code-Wandler.

Suchen und Ersetzen

Die Suchwörter beziehungsweise Ersatzwörter werden in der Statuszeile eingegeben.

CTRL-J Label-Definition suchen, zum Beispiel setzt

```
JUMP:.LABEL
```

den Cursor auf die Label-Definition :LABEL. Dieser Befehl durchsucht den ganzen Text,also nicht nur vorwärts wie CTRL-Q/F oder CTRL-Q/A.

CTRL-Q/F Beliebige Zeichenfolge finden. Dabei wird ab Cursor-Position der Text bis zum Ende durchsucht. Der Cursor bleibt jeweils am Ende des gefundenen Wortes stehen. Sollen auch Zeichenfolgen vor der aktuellen Cursor-Position gefunden werden, empfiehlt sich ein vorheriges CTRL-Q/R.

CTRL-Q/A Suchen und Ersetzen. Beispiel:

```
FIND:.ASC
CHNG:B
```

ersetzt '.ASC' durch 'B'. Bei jeder Fundstelle fragt der Computer:

REPLACE (Y/N/*)?

Die möglichen Antworten bedeuten:

'Y' = ersetzen
'N' = nicht ersetzen
'*' = automatisch ersetzen
STOP = Abbruch

Durch '*' werden also sämtliche weiteren Suchworte durch das Ersatzwort ersetzt, ohne nochmalige Abfrage.

CTRL-L wiederholt die letzte Operation, sucht also die nächste Fundstelle oder fährt mit dem Suchen und Ersetzen fort.

Back to BASIC

CTRL-K/Q Editor verlassen. Mit RUN kann man wieder in den Editor zurückkehren, ohne den Text zu verlieren.

Der Assembler: die Maschine für Maschinensprache

Sämtliche Assemblerbefehle werden wie die Editor-Kommandos über CTRL und eine weitere Taste eingegeben. Eventuell erforderliche Eingaben erwartet das Programm in der Statuszeile.

CTRL-K/X Text assemblieren. Dabei wird abgefragt:

CODE Dadurch wird festgelegt, wohin der erzeugte Maschinencode geschrieben werden soll; ob in den Speicher, auf Diskette oder nirgendwohin. Beispiele:

CODE:82DEST.O: Objektcode unter dem Namen „DEST.O" auf Diskette schreiben.

Speicherbelegung INPUT-ASS-64

$0801 –	$2622	Assembler/Editor
$2623 –	$B6FF	Textspeicher, aktuelle Endadresse in Statuszeile
$B700 –	$BFFF	diverse Puffer, Anfang Symboltabelle
$C000 –	$CFFF	nicht benutzt
$D000 –	$DFFF	I/O-Bereich
$E000 –	$FFFF	Fortsetzung Symboltabelle

Nach einem Reset kann INPUT-ASS-64 durch SYS 2088 wieder aufgerufen werden, in der Regel ohne Textverlust.

Speicherbelegung INPUT-ASS-128

Bank 0:

$0800 –	$09FF	div. Assembler-/Editor-Routinen
$0A00 –	$0AFF	System-Variablen C128
$0B00 –	$10FF	div. Puffer/I
$1100 –	$12FF	System-Variablen C128
$1300 –	$1BFF	nicht genutzt
$1C01 –	$3EFF	Assembler/Editor
$3F00 –	$FEFF	div. Puffer/II, Symboltabelle
$FF00 –	$FFFF	MMU/System-Variablen

Bank 1:

$1001 -	$FEFF	Textspeicher, aktuelles Ende in Statuszeile

Beim Start des Programms wird der Adressbereich von $0000 bis $0FFF als Common Area definiert. Nach einem Reset kann INPUT-ASS-128 durch SYS 7208 wieder aufgerufen werden, in der Regel ohne Textverlust.

CODE:RETURN: keine Angabe erzeugt keinen Code.

CODE:*: Objektcode wird in den Speicher an die Adresse des ersten ORG-Befehls geschrieben. Vorsicht, nicht in den Textspeicher oder den Assembler schreiben! (siehe unten: Speicherbelegung)

LIST Ausgabegerät des Listings. Beispiele:
LIST:47: Formatiertes Listing auf den Drukker
LIST:00: dito auf Bildschirm, erfordert SYMB:30
LIST:RETURN: kein Listing ausgeben.

SYMB Ausgabegerät der Symboltabelle. Beispiele:
SYMB:83SYM: Symboltabelle unter dem Namen „SYM" auf Diskette schreiben.

SYMB:30: Symboltabelle auf dem Bildschirm ausgeben. Anhalten/Ende wie CTRL-K/F.
SYMB:RETURN: nicht ausgeben.

CTRL-K/D Probeassemblierung ohne Ausgaben; sinnvoll für Syntax-Checks.

Labels

Labels dürfen beliebig lang sein und aus Buchstaben und Zahlen bestehen. Das erste Zeichen muß ein Buchstabe sein. Vor jeder Label-Definition muß ein Doppelpunkt stehen. Label-Definitionen dürfen auch allein in einer Zeile stehen. Mnemonics und Pseudo-OPs können nicht als Label definiert werden. Zwischen Label und folgendem Text in derselben Zeile muß ein Leerzeichen stehen. Beispiele:

```
:LABEL1 LDA #00
:LABEL2 ;Kommentar
:LABEL3
```

Mnemonics

Die Befehle werden entsprechend der normalen 6502-Notation erwartet; Ausnahmen:

- Adressierungsart Akkumulator: „ASL" statt „ASL A"! Der Quelltext muß kleingeschrieben sein.
- Adressierungsart Indirekt-X: „(AD),X" statt (AD,X)

Der Quelltext muß klein geschrieben sein.

Assemblerbefehle (Pseudo-OPs)

Diese Befehle werden in den Programmtext geschrieben und dienen als Anweisungen für den Assembler.

b Byte-Befehl. Die Zahlen beziehungsweise der Text werden als Tabelle assembliert. ASCII-Zeichen werden durch einen vorangestellten Apostroph gekennzeichnet, fortlaufende Texte können durch Anführungszeichen eingeschlossen werden. Die einzelnen Byte-Anweisungen werden durch Kommata getrennt. Entspricht .BYT, .DC.B, .ASC bei anderen Assemblern. Beispiele:

```
B 'A,'B,'C,12,10,13,0
B "INPUT 64",13,0
:SCHWARZ = 0
B SCHWARZ
```

W Word-Befehl. Assembliert 16-Bit-Worte. Die Worte werden in der 6502-üblichen Reihenfolge gespeichert, erst Low-, dann High-Byte.

Beispiel 1:

```
W $FFEA
```

erzeugt die Byte-Kombination

```
SEA, $FF
```

Beispiel 2:

```
ORG $C000
:START LDA #00
:END RTS
. . .
:TABEL W START, END
```

erzeugt an der durch Label TABEL gekennzeichneten Adresse die Byte-Kombination „$00, $C0, $02, $C0".

S NUM assembliert Null-Bytes der Anzahl „NUM" in den Text. Zum Freihalten von Datenbereichen geeignet. Nachteil: diese Null-Bytes müssen dann jeweils auch von Diskette/Kassette geladen werden. Beispiel:

```
ORG $C000
:INIT JMP BEGIN
:BUFF S $100
:BEGIN LDA #00
```

erzeugt einen Buffer ab Adresse $C003 ($C000 plus Befehlslänge des JMP-Befehls), der $100 Bytes lang ist. Label BEGIN entspricht damit der Adresse $C103.

ORG ADRESsetzt den Programmzähler des Assemblers auf die Adresse „ADRES". Die ORG-Anweisung muß zu Anfang des Programms stehen. Wird dies vergessen, kommt es zu „unerklärlichen" Branch-Errors. Der Programmzähler kann mehrmals neu gesetzt werden, der Code wird auf jeden Fall an die Adresse des ersten Programmzählers geschrieben! Alle weiteren ORG-Definitionen beeinflussen nur den logischen, nicht den physikalischen Stand des Programmzählers. Um auch den realen Stand des Programmzählers zu beeinflussen (natürlich nur vorwärts), wird der S-Befehl benutzt:

```
S $3000-*
```

setzt den Programmzähler logisch und physikalisch auf $3000.

= Zuweisung (Equate). Beispiel:

```
:PRINTC=$FFD2
```

weist dem Label PRINTC den Wert $SFFD2 zu. Entspricht EQU bei anderen Assemblern. Das Label wird nur im ersten Durchlauf gesetzt. Eine erneute Zuweisung erzeugt einen „DD-Error".

:= Mehrfachzuweisung. Beispiel:

```
:PASS := 1
```

erlaubt die (eventuell mehrfache) Neu-Definition eines Labels, das Beispiel setzt das Label PASS auf 1. Falls es dieses Label schon gibt, wird es geändert.

PAG bewirkt einen Seitenvorschub im Listing.

LIS N setzt den Listing-Mode.

lis 0: nicht ausgeben
lis 4: ausgeben, Makros nicht expandieren
lis 5: ausgeben, Makros voll ausgeben
lis 6: ausgeben, nicht erfüllten If-Teil ausgeben
lis 7: alles ausgeben

Der Listing-Mode kann innerhalb eines Programmtextes beliebig oft gewechselt werden.

TIT TEXT

TEXT wird als Titel am Anfang jeder Seite ausgedruckt. Der Titel kann mehrmals im Programm umdefiniert werden. In den Titel können auch Steuerzeichen für den Drukker eingestreut werden.

Ausdrücke

Der Assembler verarbeitet folgende Zahlentypen:

dezimal ist Vorgabe (Default)
hexadezimal Vorzeichen '$'. Beispiel: $FFFF
binär Vorzeichen '%' Beispiel: %01010111
ASCII Vorzeichen '". Beispiel: "a
Programmzähler Zeichen *.
Beispiel: :LB = *

Die Zahlensysteme können beliebig vermischt werden. Beispiel:

:TABEL = $1000+12

weist TABEL den Wert $100B zu.

Rechenoperationen

Der Assembler arbeitet mit vorzeichenlosen 16-Bit-Zahlen.

+ - * / : Grundrechenarten
& : logisches AND
! : logisches OR
£ : logisches EOR
= < > : Vergleiche. $FFFF=wahr, 0=falsch

Die Ausdrücke werden von links nach rechts ausgewertet. Klammern sind nicht erlaubt. Außerdem gibt es noch die Operatoren HIGH und LOW, die den ganzen Ausdruck beeinflussen. Beispiel:

:VIDEO = 1024

LDY #>NAME+10 ;High-Byte
LDX #<NAME+10 ;Low-Byte

führt zu

LDY #$04
LDX #$10

Kommentare

; Kommentar-Einleitung. Kommentare gehen bis zum Zeilenende. ';←' gibt im Listing einen Strich.

Makros

Mit Makros kann man eigene Befehle konstruieren. Die Syntax ist

1. Name des Makros
2. Makro-Befehl 'm'
3. eventuelle Parameter
4. Befehle dieses Makros
5. Endekennzeichen '/'

Das durch '/' markierte Ende der Makro-Definition muß allein in der Zeile stehen. Beispiel:

:MESSAGE M 1 ;Makro MESSAGE mit 1 Parameter
LDY #>@0 ;@0=1. Parameter
LDA #<@0
JSR $AB1E
/

Dieses so definierte Makro kann aufgerufen werden mit

MESSAGE USERTXT

und entspricht dann folgendem expandierten Text:

LDY #>USERTXT
LDA #<USERTXT
JSR $AB1E

USERTXT muß natürlich vorher definiert sein.

Ein Makro kann bis zu 10 Parameter haben, beim Aufruf muß diese Anzahl eingehalten werden. Beim Aufruf können beliebige Ausdrücke übergeben werden. Man kann Makros mehrmals aufrufen und auch

INPUT-ASS-Befehle Kurzübersicht

Alle Befehle sind zu verstehen als CTRL und gleichzeitig die angeführten Tastenkombination(en). Bei Befehlen, die aus zwei Buchstaben bestehen, kann der zweite Buchstabe auch ohne CTRL eingegeben werden.

V Insert-Modus ein/aus
P Control-Zeichen-Eingabe

Cursor-Bewegung

W Scroll up
E Zeile hoch
R Seite rauf
A Wort links
S Zeichen links
D Zeichen rechts
F Wort rechts
Z Scroll down
X Zeile runter
C Seite runter
I Tabulatorsprung
QS Zeilenanfang
QD Zeilenende
QR Textanfang
QC Textende
QB Blockbeginn
QK Blockende

Einfügen und Löschen

G Zeichen unter Cursor löschen
QY Zeichen rechts ab Cursor löschen
Y Zeile löschen
QL UNDO

Block-Handling

KA Ganzen Text als Block markieren
KB Blockanfang markieren
KK Blockende markieren
KC Block kopieren
KV Block verschieben
KY Block löschen

Statuszeile

D Aktuelle Eingabe wird Default

File-/Peripherie-Befehle

KE Disk-Befehl senden
KF Directory
KR Block laden
KS Bereich speichern (nur C64!)
KW Block schreiben
KQ Editor verlassen

Suchen und Ersetzen

J Sprung zum Label
QF Zeichenfolge suchen
QA Zeichenfolge austauschen
L Letzten Befehl wiederholen

Assembler

KX Assembleraufruf
KD Probeassemblierung

verschachteln. Ein Makro kann erst aufgerufen werden, nachdem es definiert wurde. Probleme kann es mit Labels geben:

- Alle Labels sind global, nichtssagende Bezeichner wie LOOP provozieren Double-Definition-Errors. Bewährt haben sich Bezeichnungen, bei denen die ersten zwei bis drei Zeichen angeben, zu welcher Prozedur sie gehören.
- Innerhalb von Makros führen doppelte Definitionen nicht zu einer Fehlermeldung.
- Makros mit Vorwärts-Referenzen müssen statt durch 'm' durch 'mf' definiert werden:

```
:COM MF 2
LDX 0+1
CPX 1+1
BNE COM2
LDA 0
CMP 1
:COM2
/
```

Bedingte Assemblierung

Der Programmierer hat durch diese Assembler-Option die Möglichkeit, bestimmte Programmteile nur unter festgelegten Bedingungen assemblieren zu lassen.

```
:DEBUG = 1
IF DEBUG
JSR TEST
EL
JSR NORMAL
EI
```

Wenn die Bedingung erfüllt ist (<> 0), wird der folgende Code bis EL (else) assembliert, wenn nicht, wird der Code hinter EL beziehungsweise EI (endif) assembliert. Im Beispiel wird also JSR TEST assembliert, aber nicht JSR NORMAL. IF, EL und EI müssen am Anfang der Zeile stehen. Die Bedingungen können auch verschachtelt werden.

Include-Files

Mit dem Include-Befehl können Source-Module von der Diskette assembliert werden. Beispiel:

```
IN 84INFILE
```

Der File-Name wird im selben Format wie im Editor angegeben und geht bis zum Ende der Zeile. Include-Befehle können nicht verschachtelt werden; in einem Include-File ist keine Makro-Definition möglich (aber natürlich der Aufruf eines vorher im Haupttext definierten Makros).

Phase Errors

Der Pseudo-Befehl PHC schaltet die Kontrolle auf Phase-Errors ein. Beispiel:

```
ORG $C000
PHC
LDA TEST
:TEST = 0
:END
```

Im ersten Durchlauf wird LDA als 3-Byte-Befehl assembliert, da die Variable TEST noch nicht bekannt ist. Im zweiten Pass macht der Assembler daraus einen 2-Byte-Befehl. Zero-Page-Variablen müssen also vor der Benutzung definiert werden! Mit Hilfe der bedingten Assemblierung kann man übrigens auch sehr schöne Phase-Errors zusammenbasteln . . .

Fehlermeldungen

Fehlermeldungen werden in der Statuszeile angezeigt, der Cursor wird im Text hinter den Fehler positioniert. Falls ein Fehler in einem Makro oder in einem Include-File auftritt, wird zuerst der Makro-Aufruf beziehungsweise die fehlerhafte Zeile im Include-File angezeigt, nach Drücken einer beliebigen Taste erscheint die Fehlermeldung, der Cursor ist jetzt auf die fehlerhafte Zeile im Makro oder auf den Aufruf des Include-Files gesetzt.

AM Illegale Adressierungsart (adress-mode error); Beispiel:

```
STA #8000
```

Die Adressierungsart „immediate" kann nicht Ziel eines Befehls sein.

BR Zu weiter Sprung (branch-error); Beispiele: bedingter Sprung weiter als 127 Bytes vorwärts oder -128 Bytes rückwärts; oder erste ORG-Anweisung fehlt.

DD Label doppelt definiert (double definition error); Beispiel:

```
:PRINT = $AB1E
ORG $C000
:PRINT LDX #00
```

Tu's nur einmal

Einem Symbol wurde erst (durch '=') ein Wert zugewiesen, und das gleiche Symbol wird später als Label im Programm verwandt.

MD: Falsche Makro-Definition (macro-definition-error); Beispiel:

Es wurde nicht als erstes der Name des Makros, sondern die Makro-Anweisung selbst (M) gegeben.

NO Kein Fehler

OP Illegaler Opcode (opcode-error); Beispiel:

Leerzeichen zwischen Befehl und Operand fehlt.

OV Zahl zu groß (overflow-error); Beispiel:

```
STA 80000
```

Der Operand ist größer als 65535.

PH Phase error; im zweiten Pass ergibt sich für eine Adreßberechnung ein anderer Wert als im ersten Pass – selten auftretender Fehler.

Syntax-Probleme

SN Syntax error; Beispiel:

'W' oder 'B' statt 'W' beziehungsweise 'B'.

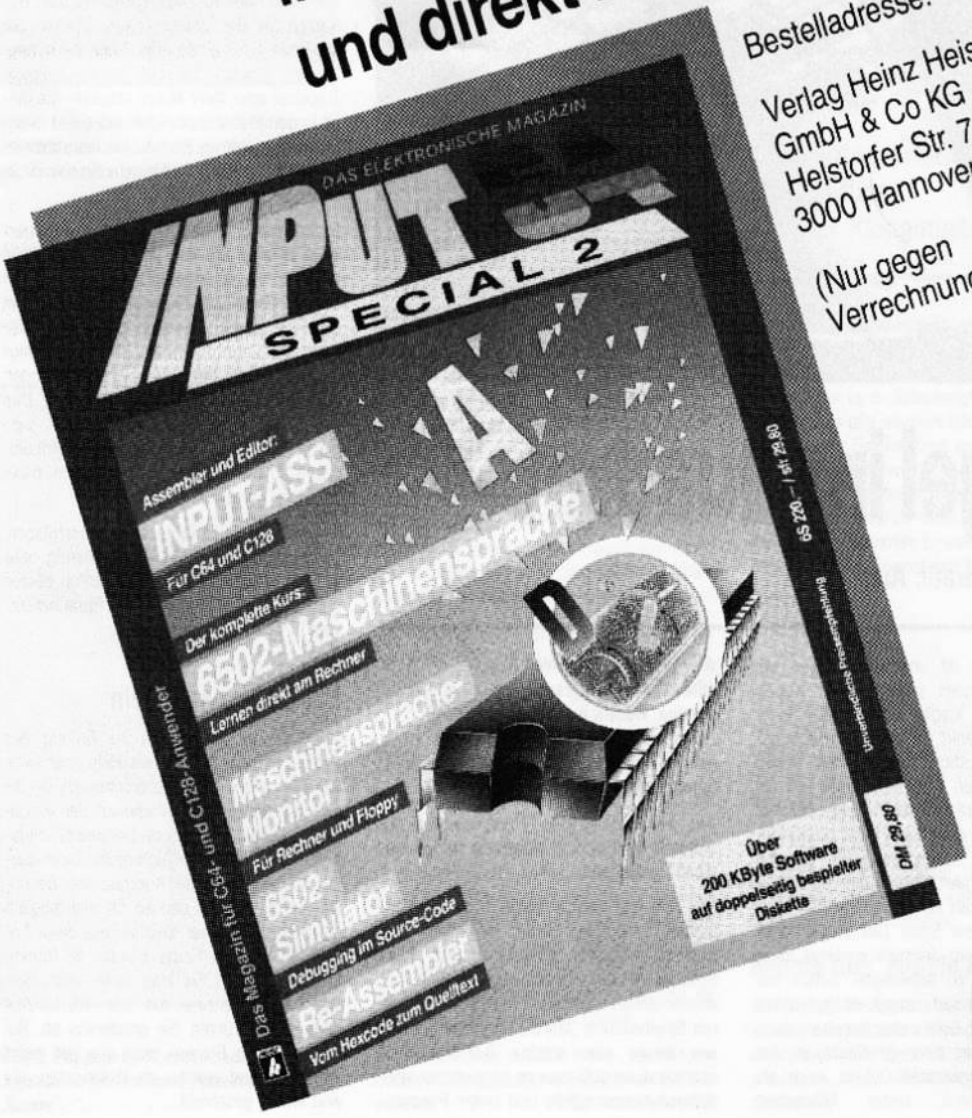
SV Symboltabelle zu groß (symbol-table-overflow); die Symboltabelle paßt nicht mehr in den Speicher. Lösung: Text in zwei Teilen assemblieren.

TM Falsches Symbol; Beispiel:

Der erste Parameter in einem Makro wird über '@1' statt über '@0' angesprochen.

UD Label nicht definiert (undefined label); es wurde ein Label angesprochen, das im Programm nicht existiert. P. Dornier/JS

Ab 6. Mai 1988
im Bahnhofsbuchhandel
und direkt beim Verlag

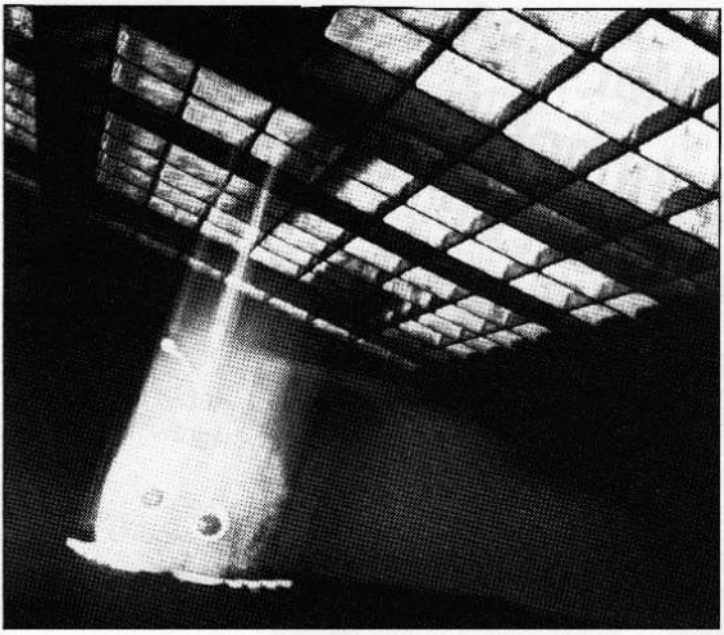


Bestelladresse:

Verlag Heinz Heise
GmbH & Co KG
Helstorfer Str. 7
3000 Hannover 61

(Nur gegen
Verrechnungsscheck)

HEISE



Kugel im Quadrat

Spiel: Quadratic Area

Was haben die folgenden Begriffe miteinander gemeinsam: Pacman, Cubert, Domino, Wedeln und Marble Madness? „Nicht viel", höre ich Sie sagen. Weit gefehlt! Aus der Essenz aller genannten Spiele ist Quadratic Area entstanden.

Pacman-ähnlich ist unsere Spielfigur: ein fröhlich grinsendes janusköpfiges Kugelmännchen, das klaglos eifrig übers Spielfeld huscht, gelenkt von einem beziehungsweise zwei Joysticks. Da sind wir schon beim Wedeln, den Skiläufern unter uns als elegante Abfahrtstechnik bekannt, einmal zick nach links, dann zack nach rechts geschwungen, so kommt man gut lenkend und die Schußfahrt unter Kontrolle behaltend den steilsten Berg hinunter. So auch hier bei unserem Spiel: Das Kugelmännchen ist, mit dem Joystick wedelnd, über die Diagonalen zu schwingen. Schön mit Gefühl, versteht sich, damit es gut übers Spielfeld kommt. Geht's aber daneben, dann erinnert das, was dann geschieht, an das bekannte Computerspiel Cubert, auch als Hexbert bekannt: unser Männchen stürzt ab und verliert eines von seinen wenigen Spielleben.

Das Spielfeld sieht aus wie nach einer gelungenen Domino-Partie, nachdem die Spielsteine schön ausgelegt sind. Alles Spielgeschehen zusammengenommen hat wieder Gemeinsamkeiten mit dem legendären Spielhallenhit „Marble Madness": Es hat wie dieses eine schöne 3-D-Grafik, ist spannend, so daß man es gern immer wieder von neuem spielt, und unser Pacmanchen kann auch als eine „Marble" mit Gesicht beschrieben werden. Die „Madness" ergreift alsbald den oder die beiden Spieler, denn man kann Quadratic Area auch zu zweit spielen.

Auf die Plätze . . .

Im Titelbild können Sie sich entscheiden, ob Sie allein oder zu zweit spielen wollen. Benutzen Sie die Tastatur, dann drücken Sie die Taste 1 für einen oder Taste 2 für zwei Spieler. Spielen Sie mit Ihrem Joystick, Trackball oder Ihrer Maus, stöpseln Sie diese in den Port 2 Ihres C64 und einen eventuellen zweiten in Port 1. Die Tastaturbelegung für das Spiel mit einem Spieler ist in Bild 1 dargestellt.

Durch mehrmaliges Drücken des Feuerknopfes oder der Leertaste wird das Spiel nach dem Laden gestartet. Sie verlassen den Vorspann und befinden sich mitten im Spielfeld. Beeilen Sie sich, denn sofort beginnt der Countdown. Nach dessen Ablauf ist das Leben Ihres Männchens beendet, und Sie haben nur noch zwei übrig. Das Leben des Männchens ist auch verwirkt,wenn Sie die Domino-Spielfeldbahn durch zu heftiges Steuern verlassen, dann stürzt Ihr Freund ab.

Sie brauchen sich übrigens nie festzulegen, ob Sie lieber die Joystick-Steuerung oder aber die Tastatur bevorzugen; denn beides geht immer. Finden Sie also in Ruhe heraus, was Ihnen besser gefällt.

Das Punktesammeln

Drücken Sie am besten zu Anfang des Spieles mal die Taste H wie Help oder Hilfe, dann wird eine Bildseite erscheinen, die Ihnen die Werte und Funktionen der einzelnen Spielsteine unseres Domino-Spielplanes erklärt. Damit Sie sich nicht zuviel merken müssen, hier die Kurzfassung: Berühren Sie so schnell und so oft wie möglich die Steine mit einer Würfel-Eins, dann haben Sie je nach Farbe 5 oder 10 Punkte erreicht; meiden Sie aber unter allen Umständen die Steine mit der Würfel-Vier, denn dann stürzen Sie gnadenlos ab. Bei allen anderen Steinen seien Sie halt recht vorsichtig und warten im Vorbeigehen ab, was weiter geschieht.

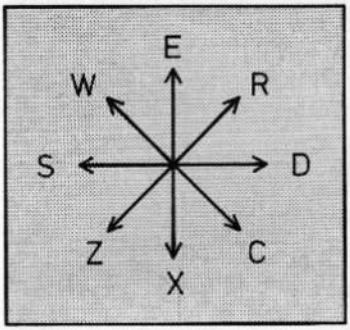


Bild 1: Die gelbe Kugel kann mit dem Joystick genauso wie mit der Tastatur in alle Richtungen gesteuert werden.

Wenn Sie, bevor der Countdown die Null erreicht hat, alle Würfel-Einser abgeräumt haben, dann geht es ab in den nächsten von 21 Levels. Das Nachprüfen, ob es wirklich so viele Levels gibt, wird sicherlich auch Ihnen schwerfallen, erfordert es zumindest sehr viel Training. Wenn Sie an einem Abend bis Level 2 kommen, dann ist das schon ein beachtlicher Erfolg.

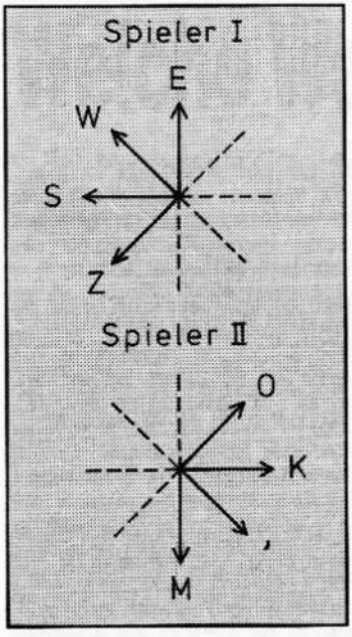


Bild 2: Für jede Seite einen „halben" Joystick — die Software verwaltet das „Ganze".

Wie schon bereits, kommen Sie vom Titelbild mit der Taste H auf die Bildschirmseite, die Ihnen die Werte und Funktionen der einzelnen Domino-Spielfeldquadrate erläutert. Von dort gelangen Sie gegebenenfalls durch mehrmaliges Drücken der Leertaste beziehungsweise des Feuerknopfes wieder ins Titelbild.

Die Funktionstaste F1 bringt Sie ganz nach vorne an den Spielanfang zurück, eben an den Vorspann, von dort ins Spielfeld durch Drücken der Leertaste oder des Feuerknopfes.

Das Spiel zu zweit

Besonders reizvoll an diesem Spiel ist die Möglichkeit, miteinander statt gegeneinander zu spielen. Gemeinsam einträchtig die Joysticks wedelnd, erringt man die Punkte auf einem gemeinsamen Punktezähler. Man wirtschaftet also in einen Topf. Anders als bei der Ein-Spieler-Version sind hier die Funktionen der Joysticks und der Tastatur halbiert, das heißt, daß immer dann, wenn der eine Spieler den Kugel-Pacman nicht mehr weiterbewegen kann, rasch sein Partner einspringen muß, um dem Marble-Männchen weiterzuhelfen (siehe Bild 2).

Wenn Sie zu zweit spielen möchten, stellen Sie diese Version im Eröffnungsmenü mit Ihrem Joystick (Port 2) durch Drücken nach rechts ein, das ist schon alles.

Rekord-Brecher

Sie werden froh sein, am Anfang mal so 200 oder gar 300 Punkte zu sammeln, immer schön langsam, 5 oder 10 Punkte. Oben rechts im Spielfeld aber steht etwas von einem Rekord, der bei 5000 Punkten liegt. Versuchen Sie mal, ihn zu brechen, allein oder gemeinsam. Meine Hochachtung, wenn Sie's schaffen. Stürzen Sie mit Ihren drei Männchen mal nicht ab, gibt es immerhin 1000 Punkte als Bonus.

Versuchen Sie's einfach mal — es werden bestimmt lange, spannende Abende für Sie werden.

W. Schmidt-Pabst/kfp

Vertauschte Bits

SampleKonverter: Umwandlung digitaler Klangmuster

Wenn digitalisierte Klänge, als sogenannte Samples auf Diskette abgelegt, von verschiedenen Anwendern genutzt werden sollen, kann es zu Schwierigkeiten kommen. Es gibt nämlich leider keinen Standard zur Speicherung von Samples für den C64. Um dennoch Fremdformate mit SANDI aus Ausgabe 3/88 bearbeiten zu können, liefert INPUT 64 in dieser Ausgabe den SampleKonverter. Er paßt andere Formate an das Format von SANDI an.

Ein Byte, bestehend aus 8 Bits, läßt sich in zwei 4-Bit-Anteile, sogenannte Nibbles, zerlegen: in High- und das Low-Nibble. Digitale Sounds, die über den Soundchip ausgegeben werden sollen, dürfen nur aus 4-Bit-Samples bestehen, da das Lautstärkeregister des Soundchips nur 16 ($=2^4$) Stufen verarbeiten kann. Solche Samples setzen sich demnach aus einer Folge von Nibbles oder Halbbytes zusammen.

Nibble hier, Nibble da.

Ein Sample-Programm kann bei der Aufnahme eines Sounds eine der vier Methoden verwenden, diese Nibbles im Speicher abzulegen, wie in Abbildung 1 schematisch dargestellt.

SANDI verwendet das Verfahren HL: Die Sample-Routine blendet zuerst das obere Nibble aus (AND 15) und überträgt dann das verbleibende untere Nibble an das Lautstärkeregister des SID. Anschließend werden die 8 Bits im ursprünglichen Byte viermal nach rechts verschoben und die oberen 4 Bits mit Null aufgefüllt, so daß das obere Nibble den Platz des unteren einnimmt. Danach kann das zweite Nibble an den SID übergeben werden. Auf diese Weise zerlegt man ein Byte in zwei Werte, die jeweils im Bereich zwischen 0 und 15 liegen.

Versuchen Sie mit SANDI ein Sample abzuspielen, das mit der Methoden LH erzeugt wurde, klingt dies natürlich völlig verfremdet. Abhilfe würde eine Funktion schaffen, die die Nibbles vertauscht.

Andere Sampler wiederum benutzen die Methoden LO oder OH, legen also nur ein Nibble pro Byte ab. Dies ergibt einen kleinen Geschwindigkeitsvorteil, bedeutet aber, daß der doppelte Speicherplatz benötigt wird. Hier müßte ein Werkzeug her, das zwei Bytes zu einem zusammenpackt, so daß die relevanten Nibbles im niederwertigen und höherwertigen Anteil des Bytes enthalten sind.

Diese Aufgaben erledigt das Programm SampleKonverter in Zukunft für Sie. Um es jederzeit nach Bedarf einsetzen zu können, sollten Sie es sich mit CTRL-S abspeichern. Sie können es aber auch innerhalb von INPUT 64 ausprobieren.

Byte für Byte . . .

Über die Menüauswahl in der linken Hälfte des Bildschirms können Sie die einzelnen Funktionen anwählen. Gesteuert wird mit Joystick oder den Cursor-Tasten. Die Auswahl erfolgt über den Feuerknopf oder RETURN.

Zuerst wählen Sie ein Sample-File über die Auswahlfunktionen **DISK FUNKTIONEN** und **DIR+LOAD** mit dem Cursor an und laden es mit einem Druck auf den Feuerknopf oder die RETURN-Taste. Alle übrigen Diskettenoperationen entsprechen denen von SANDI.

Nach dem Laden analysiert der SampleKonverter automatisch die Daten und gibt anschließend das Ergebnis aus. Das Format erscheint im Analysefenster rechts auf dem

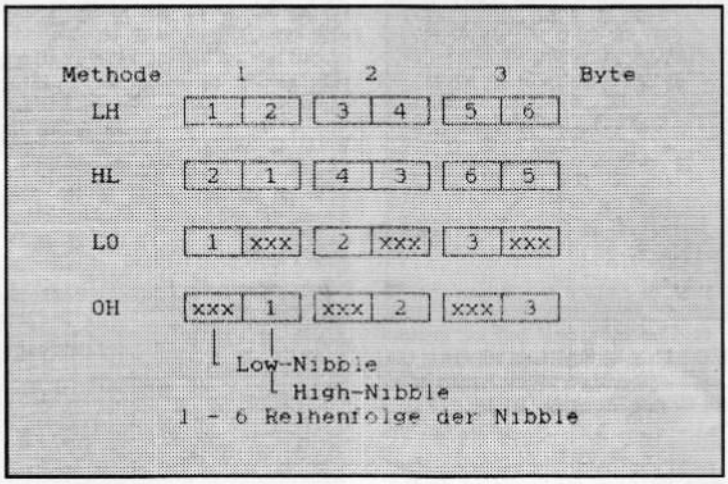


Abbildung 1:
In der Reihenfolge der Aufnahme

Bildschirm. Der SampleKonverter stellt automatisch fest, mit welchem der oben genannten Verfahren das Sample erzeugt wurde. Ein Punkt markiert die entsprechende Anordnung der Nibble. Diese Formate können dann ins SANDI-Format konvertiert werden. Diese Analyse findet nach jedem Laden oder Konvertieren statt.

Mit der Play-Funktion können Sie sich das aktuelle Sample zur Probe anhören. Möchten Sie die Ausgabegeschwindigkeit ändern, stellen Sie mit den Cursor-Tasten oder dem Joystick den Cursor auf die gewünschte Ziffer und können dann mit RETURN den Wert erhöhen oder mit SHIFT-RETURN erniedrigen. Alternativ steht Ihnen hier auch der Joystick zur Verfügung. Zur Änderung der Werte müssen Sie zusätzlich den Feuerknopf gedrückt halten.

. . . behandelt

Wählen Sie **KONVERTIEREN**, gelangen Sie in ein weiteres Menü. Hier finden Sie drei Möglichkeiten, einen Sound umzuwandeln:

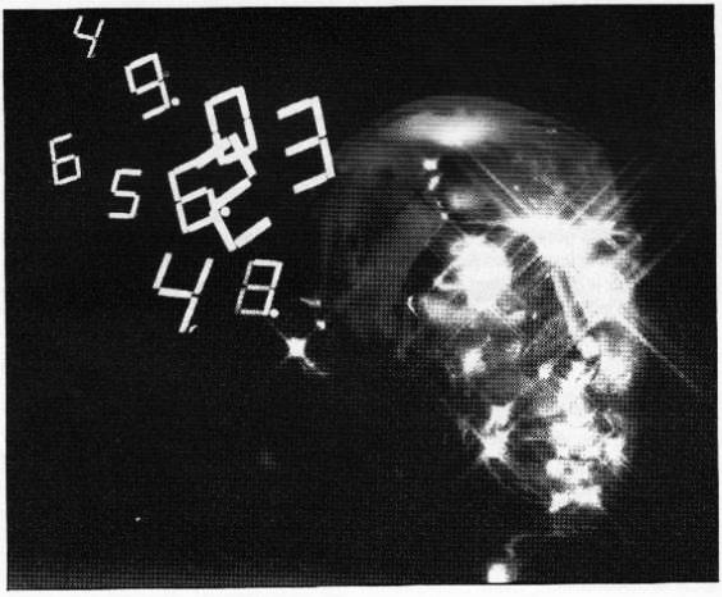
'**X X**': High- und Low-Nibbles sind nach dem Konvertieren vertauscht.

'**X 0**': Das konvertierte Sample setzt sich nur aus den High-Nibbles des ursprünglichen Samples zusammen. Low-Nibbles fallen weg, die Sound-Datei wird um die Hälfte kürzer.

'**0 X**': Umgekehrt wie oben; es bleiben nur die Low-Nibbles erhalten. Die High-Nibbles werden ignoriert, das Sound-File wird ebenfalls um die Hälfte kürzer.

Nach dem Konvertieren sollten Sie sich das Ergebnis noch einmal zur Probe anhören. Sind Sie zufrieden, speichern Sie den Sound mit der Diskettenfunktion **SAVE**. Hierbei wird automatisch die SANDI-Kennung "/4" an den von Ihnen angegebenen File-Namen angehängt. Haben Sie irrtümlich **SAVE** ausgewählt, gelangen Sie mit der Taste '←' wieder ins Menü zurück.

Klingt das Klangmuster nach dem Konvertieren immer noch unerträglich, so liegt dies wahrscheinlich am ungenauen Timing anderer Sampler-Programme, die keine Timer bei der Aufnahme und Wiedergabe verwenden, sondern über Programmschleifen einen Zeitabgleich vornehmen. Hier kann selbst ein Konverter nicht mehr helfen. Manfred Sonnenberg/RH



Weiche Wellen

SamplePlay: BASIC-Tool für Soundsamples

In der Märzausgabe von INPUT 64 haben wir das Programm SANDI veröffentlicht. Wer das Modul mit CTRL-S auf eignen Datenträger übernommen hatte und ebenso mit den Beispiel-Sounds verfuhr, die ebenfalls in dieser Ausgabe angeboten wurden, konnte sich bereits mit Aug' und Ohr an Samples erfreuen.

Besitzer der Multifunktionskarte aus c't 9/86 können über die 8-Bit-A/D- und 8-Bit-D/A-Wandler Klänge aufnehmen und wiedergeben. Kurzbeschreibung der Karte und Bezugsquellen finden Sie in INPUT 64, Ausgabe 9/86, im Rahmen des SAM-Projekts.

Damit Sie nun die Möglichkeit haben, Samples in eigene Programme einzubauen, können Sie sich im Modul SamplePlay ein geeignetes BASIC-Tool abspeichern. Für „Maschinisten" gibt es zusätzlich noch den Quellcode dieses Tools, passend zum INPUT-ASS, den Sie an anderer Stelle in dieser Ausgabe finden. Wählen Sie mit '1' oder '3' das Gewünschte und speichern es mit CTRL-S auf Diskette.

Wenn Ihr C64 wohlartikuliert mit Ihnen redet oder ein ganzes Orchester aus dem Lautsprecherchen Ihres Monitors oder Fernsehers erklingt, lauschen Sie aller Wahrscheinlichkeit nach digitalisierten Klängen, die irgendwo im Programm verborgen sind. Um ähnliche Effekte in Ihrem eigenen Programm verwirklichen zu können, bietet INPUT 64 Ihnen hier ein entsprechendes Tool.

Im Modul führen wir Ihnen unter '2' die Anwendung des BASIC-Tools vor. Hiermit können Sie innerhalb eines Programms einen beliebigen Speicherbereich über den SID im C64 zu Gehör bringen. Haben Sie in einem solchen Bereich vorher ein Sample abgelegt, können Sie dieses wiedergeben und so beliebige Klangmuster in Ihre Programme einbinden.

Feste Lagen

Wie bei INPUT-Tools üblich, wird beim ersten Start der BASIC-Anfang hochgesetzt, so daß Sie vor dem Abspeichern Ihrer Programme immer eingeben müssen:

POKE 44,8:POKE 43,1

Zum Aufruf des Tools schreiben Sie einfach eine Zeile mit der folgenden Sytax in Ihr Programm:

SYS 2084 , ANFANG , ENDE , ZT

ANFANG und ENDE geben dabei die Grenzen des abzuspielenden Speicherbereiches an. ZT ist ein Wert, der dem gleichnamigen Wert aus der SANDI Tabelle entspricht. Es sind Werte von 0-255 erlaubt (von schnell bis langsam, analog zu den in SANDI verwendeten Hexadezimalzahlen $00-$FF). Es ist sogar möglich, Klänge in den selten benutzten Bereichen unter dem BASIC-ROM ($A000-$BFFF) oder unter dem KERNAL-ROM ($E000-$FFFD) abzulegen. Das Tool schaltet bei Lesezugriffen in diesem Bereich automatisch auf RAM.

Mit Hilfe des Sourcecodes (unter '3') läßt sich das Tool auch in Assemblerprogramme einbinden. Die Parameterübergabe für den Aufruf ist im Quelltext dokumentiert.

```
:SAMPLE M 3
      LDA    #<@0
      STA    MELL
      LDA    #>@0
      STA    MELH
      LDA    #<@1
      STA    MEHL
      LDA    #>@1
      STA    MEHH
      LDA    #@2
      JSR    PLAY
      /
```

Listing 1: Ein Makro machts's lesbarer

```
 96 REM TD ABSPIELRATE
 97 REM AD ANFANGSADRESSE
 98 REM RD ROSENGRAD IN WINKEL
 99 REM 32 BESTIMMT DIE TONHOEHE
100 AD=49960:RD=PI/32:TD=200
110 FOR I=0 TO 2048 STEP 2
118   REM V0 IST LOWNIBBLE
119   REM V1 IST HIGHNIBBLE
120   V0=INT(SIN(I*RD)*8+8)
130
V1=INT((SIN((I+1)*RD)*8+8)/16)
140   POKE AD+I,V0 OR V1
150 NEXT I
160 AE=I-1
200 SYS 2084,AD,AE,TD
210 SYS 2084,AD,AE,TD/2
220 FOR J=0 TO 255
230   SYS 2084,AD,AE,J
240 NEXT J
300 END
```

Listing 2: So geht's in BASIC

Wem es gefällt, der kann sich auch ein INPUT-ASS-Makro für die Klangwiedergabe definieren wie im Listing 1 gezeigt. Damit können Sie im Assembler-Quelltext Klänge genau so komfortabel aufrufen wie mit dem BASIC-Tool. Es heißt dann ganz einfach:
SAMPLE ANFANG ENDE RATE
ANFANG und ENDE sind selbstverständlich Label, die auf Anfangs- und End-Adresse des Sample-Bereichs gesetzt sein müssen. Für RATE gilt das Gleiche wie oben gesagt: Werte von 0 bis 255; je größer der Wert, desto langsamer die Wiedergabegeschwindigkeit, wodurch sich natürlich auch die Tonhöhe des Klangmusters verändert.

Auf die weiche Tour

Das Tool läßt sich aber auch ohne Samples zur Klangerzeugung einsetzen. So können Sie mit einem kurzen BASIC-Programm einen reinen Sinuston generieren, eine Schwingungsform, die der SID im C64 nicht erzeugen kann. Anhand des Listings 2 läßt sich dies leicht nachvollziehen.

Natürlich ist dies ein recht langsames Verfahren, aber für experimentelle Zwecke ausreichend. Die eigentlichen „Zeitfresser" sind die Zeilen 100 – 160. Aber die müsssen ja in jedem Programm nur einmal durchlaufen werden. Danach läßt sich der erzeugte Sinuston beliebig oft abspielen. Mit SANDI oder einem Monitorprogramm können Sie die erzeugten Werte auch abspeichern und diese im Programm später nachladen. Bei dieser Variante können Sie auf den ersten Teil des BASIC-Beispiels anschließend verzichten. Maschinenprogrammierer könnten ein Sinus-Sample per Tabelle berechnen.
Auf die gleiche Weise lassen sich mit etwas mathematischem Geschick andere Schwingungsformen erzeugen, die dann in einem vernünftigen Speicherbereich abgelegt werden. Mit Hilfe des SamplePlay-Tools können Sie diesen Bereich dann zu Gehör bringen. Durch mehrfache Aufrufe des Tools, möglicherweise auch eingebettet in Schleifen, lassen sich Toneffekte und kleine Melodien erzeugen.

F.Börnke/RH

64'er Tips

Verwandlungskünste

Die Typen unter den BASIC-Variablen

Jeder, der schon einmal versucht hat, ein BASIC-Programm zu schreiben, hat sicher in der Bedienungsanleitung zum C64 geblättert und sich auf den Seiten 35 bis 37 über den Umgang mit BASIC-Variablen informiert. Das göttliche Vergnügen, selbst Namen vergeben zu können, erspart einem die Mühe, riesige Adreßverzeichnisse absuchen zu müssen, wenn man aus der Speicherwüste bestimmte Größen zurückholen will.

Sie sollten jedoch nie vergessen, daß Ihre schöpferische Freiheit, Namen zu vergeben,

Überschreiten Ihre Programmiervorhaben ein auf die Dauer langweiliges PRINT "HALLO!", werden Sie ohne Variablen nicht auskommen. Schließlich möchte man in der doch recht großen Speicherlandschaft des C64 alte Bekannte unter vertrautem Namen auch wiederfinden. Solche Begegnungen können aber leicht von der unheimlichen Art sein, wenn einem anstelle alter Bekannter ein fremder ?TYPE MISMATCH ERROR begegnet.

eingeschränkt ist. Variablennamen dürfen weder Bezeichungen von BASIC-Befehlen noch die Kürzel der Systemvariablen ST unf TI enthalten. Die Variable DIFFERENZ wird der BASIC-Interpreter nicht akzeptieren. Er stolpert über das 'IF' mit einem ?SYNTAX ERROR. ERNST UND ERNIE sind zwar für den Leser verschiedene Namen, der BASIC-Interpreter kümmert sich aber nur um die ersten beiden Zeichen, für ihn wird nur von ER gesprochen. Erstaunlicherweise interessiert er sich aber für das letzte Zeichen im Namen: ERNST, ERNIE$ und ERWIN% sind für ihn drei verschiedene Typen: ER, ER$ und ER%.

Ausgeprägte Charaktere . . .

Im Ihrem C64-Handbuch werden diese drei Charaktere als Integer-, Floating-Point- und String-Variable beschrieben. Wie Sie dort nachlesen können, unterscheidet der BASIC-Interpreter des C64 diese drei Gruppen einzig anhand des letzten Zeichens im Namen. Enden die Namen auf '%' oder '$', in-

terpretiert der C64 diese als Integer- oder String-Variablen.

String-Variablen können nur Zeichenketten aufnehmen. Ohne sie gäbe es keine Möglichkeit, innerhalb von BASIC-Programmen Texte zu verarbeiten. Der String W$="2.5" besteht nur aus den Zeichen '2', '.' und '5'. Rechnen können Sie damit nicht.

Integer-Variablen begnügen sich mit dem ganzzahligen Anteil einer Zahl. Kommaanteile werden einfach abgeschnitten. Sie bieten den Vorzug, wenn Sie als Felder benutzt werden, weniger Speicherplatz zu verbrauchen als Fließkomma-Variablen.

. . . mit Einschränkungen

Für jeden dieser Variablentypen gelten bestimmte Grenzen:

Fließkomma-Variablen können Werte zwischen 10^{-38} bis 10^{38} annehmen. Werden Ihre Zahlen zu groß, stehen Sie vor einem ?OVERFLOW ERROR (Deutsch: eine Bereichsüberschreitung). Unterschreiten Sie diese Grenzen, bleibt der Interpreter hartnäckig bei der Zahl Null.

Fließkomma-Variablen sind nie völlig exakt, vor allem wenn sie das Ergenbis einer Berechnung enthalten, da der C64 recht komplizierte Umwandlungen durchführen muß, bis er eine solche Zahl, die ja sehr groß werden kann, in den Speicher gepackt hat. Geben Sie auf Ihrem C64 „? 8.13" ein, erhalten Sie die verwirrende Antwort „ 8.13000001".

Sie müssen beim C64 immer auf Rechenungenauigkeiten gefaßt sein. Die 64er Tips in der Ausgabe 6/85 haben sich bereits mit diesem Thema beschäftigt. Für gehobene mathematische Ansprüche hat INPUT 64 in Ausgabe 3/87 das BCD-Tool mit einer Genauigkeit von 250 Stellen veröffentlicht.

Wahre Größe

Hinzu kommt, daß Sie immer nur elf Stellen der Zahl inklusive Vorzeichen und Dezimalpunkt zu sehen bekommen, also maximal neun Ziffern. Enthält eine Variable mehr als neun Ziffern vor oder zwei Ziffern hinter dem Dezimalpunkt, gibt BASIC die Zahlen mit einem Exponenten aus:

5E+9 = $5*10^9$ = 5 000*000*000
5E-3 = $5*10^{-3}$ = .005

Beachten Sie aber, daß bei mehr als 9 Ziffern mit kleinerem Vorkommaanteil merkwürdig gerundet wird:

2.999 999 993 = 2
2.999 999 994 = 3 (!)
1.999 999 994 = 1.99999999

Dies liegt an der Rechenungenauigkeit der Interpreter-Routinen, die für die Umwandlung in ein für den C64 verständliches Format zuständig sind.

Die Integer-Variablen haben nicht mit solchen Ungenauigkeiten zu kämpfen. Dafür ist der Wertebereich von -32 768 bis 32 767 erheblich kleiner. Weisen Sie einer Integer-Variablen eine Zahl mit Nachkommastellen zu, werden diese einfach abgeschnitten.

Eine String-Variable kann bis zu 255 Zeichen aufnehmen. In Ihrem Handbuch zum C64 finden Sie eine ASCII-Tabelle, die sämtliche Zeichen aufführt, die der C64 kennt.

Im Wandel

Immer wenn Sie innerhalb eines Programms einen Wert behalten oder im Laufe des Programms einen Wert ändern wollen, werden Sie sich der Variablen bedienen. Sie können sogar den Anwender während des Programmablaufs mit einer Eingabeanweisung auffordern, selbst Variablen einen Wert zuzuweisen. Außerdem können Sie Variablen miteinander verknüpfen und vergleichen und das Ergebnis solcher Operationen wiederum in einer anderen Variablen ablegen.

Wer übersichtliche Programme bevorzugt und unliebsame Variablenwerte vermeiden möchte, sollte zu Beginn seines Programms die benötigten Variablen aufführen und ihnen einen Startwert (Default-Wert) zuweisen. Unerwartete Werte können sonst zu Programmabbrüchen und Fehlermeldungen führen.

Beim C64 führt, anders als beim C128, jede Änderung am BASIC-Programm zum Verlust sämtlicher Variableninhalte. Verwenden Sie beispielsweise Variablen als Adressen in SYS- oder POKE-Befehlen, fühlt sich nach einer Programmänderung und bei einem wilden Einsprung über GOTO immer die Adresse Null angesprochen. Die Effekte sind in fast allen Fällen wenig wünschenswert. Nach einem CLR-Befehl innerhalb eines Programms sollte man sich versichern, ob nicht doch eine der Variableninhalte noch benötigt wird. 'RUN' löscht ebenfalls alle alten Variablen-Inhalte.

In Funktion

Führen Sie mit numerische Variablen (Integer- und Floating-Point-Variablen) Rechenoperationen aus, müssen Sie sich sicher sein, daß keine undefinierten Ergebnisse herauskommen können. Dies geschieht zum Beispiel beim Teilen durch Null. Verwenden Sie mathematische Funktionen, wie etwa LOG und SQR, dürfen Sie im Argument (in den Klammern) keine negativen Werte verwenden.

LOG (X) – X>0
SQR (X) – X>=0

String-Variablen enthalten ohne Zuweisung kein Zeichen, haben also die Länge Null. Demzufolge führt die Funktion ASC(X$) unweigerlich zu einem Fehler. Ebenso gibt es für String-Funktionen keine nullte Position im String. Setzen Sie in der MID$-Funktion für den mittleren Parameter '0' ein, gibt es ebenfalls eine Fehlermeldung.

Auch wenn das '+'-Zeichen bei einer String-Verknüpfung dem Additionszeichen gleicht, können Sie mit String-Variablen keine Rechenoperationen durchführen. Zwischen String-Variablen sind jedoch die Zeichen '=', '>', '<' zulässig und sinnvoll. Bei Vergleichen ist jedoch nur der ASCII-Code des ersten Zeichens ausschlaggebend, nicht die Länge des Strings:

A$="RAINER"
B$="ZUFALL"
PRINT A$<B$
-1 (wahr)

Die einzelnen Variablentypen lassen sich aber ineinander umwandeln. Die Umwandlung von Integer-Variablen in Fließkomma-Variablen ist unproblematisch. Hier reicht das Gleichheitszeichen: A=A%. Im umgekehrten Fall müssen Sie sicher sein, daß der Wert der Fließkomma-Variablen nicht die Grenzen der Integer-Variablen überschreitet. Dabei wird nicht gerundet. Durch einfache Addition von .5 können Sie jedoch nachhelfen. Zum Beispiel: A%=A+.5 oder A%=A/3+.5.

```
990 rem "a$ = eingegebenes Zeichen
992 rem "a = ASCII-Wert
994 rem "a% ist fuer Gross-Klein-
995 rem "Buchstaben gleich!
996 rem "i = Laenge der Eingabe
998 rem "nn$ = Name
1000 nn$="":i=0
1010 print"(swlc)(clr)Ihren Namen? "
1015 print"(<f8> Korrektur <f1> ok)
1020 print"(home)(down)(down)"nn$
1030 get a$:if a$="" then 1030
1035 rem "ASCII erspart String-
1037 rem "Operationen
1040 a=asc(a$):a%=a and 127
1050 if (a%<65 or a%>90) and a<>32 then 1100
1060 if i>39 then 1030
1070 nn$=nn$+a$:i=i+1:goto1120
1095 rem "Die f1-Taste beendet
1097 rem "die Eingabe
1100 if a=133 then 1200
1105 rem "'f8' loescht die Eingaben
1110 if a=140 then 1000
1120 goto 1020
1200 print"(down)Sie heissen also:"
1210 print nn$
1220 print"(down)Immerhin"i"Zeichen lang!"
```

Listing 1:
Sinnvolle Namen?

Rollenwechsel

String-Variablen und numerische Variablen lassen sich nur mit Hilfe der speziellen Funktionen VAL(X$) und STR$(X) direkt ineinander überführen. Wird aus einer Zahl ein String erzeugt, enthält der String bei positiven Zahlen immer noch ein Leerzeichen für die Vorzeichenstelle, also: LEN(STR$(0))=2.

Ein zweiter Weg führt über die Funktionen ASC und CHR$. Bei A=ASC("A") enthält die numerische Variable A den ASCII-Kode des Zeichens 'A', nämlich 65. Natürlich kann in der Klammer auch eine String-Variable stehen. Das Ergebnis ist dann der ASCII-Kode des ersten Zeichen im String. Die CHR$-Funktion ist die geeignete Umkehrfunktion: X$=CHR$(65). Der String X$ enthält jetzt das Zeichen 'A'. Sie können auch eine Zahlenvariable als Argument verwenden: X$=CHR$(X). Dabei darf das Argument nur im Bereich zwischen Null und 255 liegen. Andere Werte erzeugen die Fehlermeldung: ?ILLEGAL QUANTITY ERROR. Nachkommaanteile werden abgeschnitten. Der Weg über die ASC-Funktion erspart in einigen Fällen Zeit und Speicherplatz raubenden String-Operationen, wie Sie in den Beispielen in Listing 4 und im Programmbeispiel aus den 64er Tips nachverfolgen können.

Neben der direkten und der errechneten Variablenzuweisung, können Sie auch den READ-Befehl benutzen, um Variabeln mit Werten zu versorgen. Die Wertetabellen stehen dann fest im Programm in DATA-Anweisungen. Für Strings gilt eine Einschränkung: die Zeichen ':' und ',' können Sie nur innerhalb von Anführungszeichen in den DATA-Anweisungen verwenden. Anderenfalls interpretiert der BASIC-Editor sie als Steuer-Zeichen, was dann zu einem „?OUT OF DATA ERROR IN ..." führt. Der Doppelpunkt gilt außerhalb von Anführungsstrichen als Trennzeichen von Befehlen, das Komma als Trennzeichen der Werte einer DATA-Anweisung. Beabsichtigen Sie einen String mit diesen Zeichen zu belegen, erreichen Sie dies nur mit folgendern Syntax:

DATA "LISTE: MUSTER1, MUSTER2"

Die Anführungsstriche selbst, das Zeichen "", lassen sich nur über +CHR$(34) an Strings anbinden.

T$=T$+CHR$(34)

Die meisten Programme stellen eine Verbindung zwischen dem Benutzer und dem Programmablauf her. Eingabebefehle wie INPUT und GET sorgen dafür, daß der Benutzer Variablenwerte von außen setzen kann.

Fortsetzung auf Seite 27

Listing 2: Auf den Punkt gebracht

```
900 rem "PRINT USING in BASIC
901 rem "Werte < .01
902 rem "Werte > 999999999
903 rem "erzeugen e-Darstellung
904 rem "x = Wert
905 rem "l = Anzahl der Stellen
907 rem "vor dem Dezimalpunkt
1000 input"anzahl";ll
1010 i=0:ll=ll-1
1020 dim x(ll),l(ll)
1030 input "zahl";x
1040 l=len(str$(int(x)))
1045 rem "Zahlen ohne Vorkommaanteil
1050 ax=abs(x)
1060 if ax<1 and ax>0 then l=1
1065 rem "hier wird die Formatierung
1066 rem "errechnet
1070 l=10-l
1080 x(i)=x:l(i)=l
1090 i=i+1:ifi<=llthen 1030
2000 print"(clr)     Wertetabelle"
2010 for i=0toll
2020 x$=str$(x(i))
2030 printspc(l(i));x$;
2040 if x(i)-int(x(i))=0 then print".0";
2050 print
2060 next i
```

Grundlagen

Adressenverwaltung und CAD-Datei

Wir stellen Ihnen diesmal eine kleine Adressenverwaltung vor, die an der einen oder anderen Stelle durchaus noch zu erweitern ist, bereits aber alle grundlegenden Routinen beinhaltet. Außerdem erhalten Sie mit der CAD-Datei die notwendige Grundlage für Zeichnungen im Bereich des Maschinenbaus.

Adressen verwalten

Wenn Sie das Programm (außerhalb von INPUT 64) gestartet haben, gelangen Sie in das Hauptmenü, von wo aus Sie – über die Eingabe der Ziffern '1' bis '8' – alle Funktionen des Programms aufrufen können.

Über die Ziffer '8' können Sie das Programm verlassen und gelangen in den Direktmodus, von wo aus Sie das Programm einfach mit 'LIST' auflisten können. Das Programm enthält keine Maschinenroutinen, so daß Sie beim Abspeichern eines veränderten Programms nichts beachten müssen.

Oft stecken in den unscheinbaren kleinen Programmen genau die Routinen, die gerade benötigt werden; und wenn dieses Programm dann noch gut kommentiert, strukturiert programmiert und außerdem lauffähig ist, hat man schon halb gewonnen. Wer weiß, welche Arbeit in der Erstellung von Dateien stecken kann, wird sicherlich auch die CAD-Datei zu schätzen wissen.

Durch eine nachträgliche Erweiterung um die Inkey- und die Directory-Routine (Ausgabe: 11/86 und 3/88) könnte das Programm sicherlich aufgebessert werden.

Zeichnungen erstellen

In Verbindung mit der Datei "maschinenbau g" können Sie mit dem CAD-Programm (Ausgabe: 11/86 bis 2/87) Zeichnungen im Bereich Maschinenbau erstellen. Es sind Objekte definiert, die zur Kenntlichmachung von Toleranzen und Oberflächenbeschaffenheit benötigt werden. Die Objekte entsprechen der DIN ISO 1101 und ISO 1302. Außerdem enthält die Datei einige Objekte für die Bemaßung von Zeichnungen.

Sie sollten aber den vorgegebenen Namen beim Speichern nicht ändern, da bekanntlich das CAD-Programm auf die Dateikennzeichnung (in diesem Fall 'g') angewiesen ist.

WM

Werkstatt intern

Für alle neu hinzugekommenen INPUT-Anwender einige Worte über die Philosophie und die Besonderheiten der ID-Werkstatt.

Wir bekommen gelegentlich Programme auf den Schreibtisch, die entweder von der Idee, der Art der Realisierung oder einer beispielhaften Programmierung eigentlich für eine Veröffentlichung geeignet sind, beim näheren Hinsehen aber noch Unzulänglichkeiten aufweisen. (Die Adreßdatei beispielsweise realisiert die Eingaben über den INPUT-Befehl, und die Diskettenzugriffe sind auch nicht „narrensicher". Aber dank einer guten Dokumentation kann dieses kleine Programm von jedem Programmierer weiterentwickelt werden.)

Einen zweiten Schwerpunkt der ID-Werkstatt stellen Dateien für bereits veröffentlichte Programme dar (so in dieser Ausgabe eine weitere CAD-Datei). Diese Dateien sind selbstverständlich ohne die entsprechenden „Mutterprogramme" nicht sinnvoll einzusetzen. Hieraus ergibt sich, daß die einzelnen Programm(-teile) innerhalb von INPUT 64 nicht lauffähig sind. Sie können lediglich das gewünschte Programm in einem kleinen Menü auswählen und über die bekannte Sequenz CTRL-S zum Abspeichern gelangen. Auch hier gibt es eine Besonderheit, die Sie beachten sollten. Da Dateien oftmals bezüglich der Namensgebung Restriktionen (Kennungen, Steuerzeichen im File-Namen) unterliegen, erhalten Sie einen Namensvorschlag.

Zwei Bemerkungen zum Schluß. Autoren, die innerhalb der ID-Werkstatt ein Programm oder eine Datei veröffentlichen, erhalten als Honorar ein INPUT 64-Jahresabonnement. Da Programme in dieser Rubrik Werkstattcharakter haben, kann eine Betreuung von uns nicht übernommen werden.

Das Bundesliga-Problem

Neues Rätsel

Die neue Rätselaufgabe heißt: Erstellen Sie ein möglichst kurzes Programm in BASIC, das es erlaubt, die Spielpaarungen von 4 bis 18 Mannschaften zu erstellen. Dabei soll keine Paarung doppelt auftreten.

Zur näheren Erklärung nehmen wir das Beispiel der 1. Fußball-Bundesliga. Dort spielen in einer Saison alle Mannschaften (18) nach einem bestimmten Schema gegeneinander. Die Saison wird in zwei Hälften aufgeteilt, in die Hin- und in die Rückrunde. Die erste Vereinfachung für uns soll sein, daß wir nur die Hinrunde berücksichtigen.

Anpfiff

Um eine weitere Vereinfachung zu vereinbaren, numerieren wir die Mannschaften von 1 bis 18 durch. Das hat den Vorteil, daß man den einzelnen Mannschaften keine Namen zu geben braucht. Das Hantieren mit numerischen Variablen ist auch einfacher als mit String-Variablen. Ein Problem kommt allerdings noch hinzu. In der Fußball-Bundesliga hat man es mit genau 18 Mannschaften zu tun, nicht mehr und nicht weniger. Das von Ihnen zu erstellende Programm soll aber allgemeingültig sein, das heißt, es soll die Paarungen von 4 Mannschaften genauso berechnen können wie von 50.

Um etwas mit den Zahlen zu hantieren, nehmen wir als Beispiel für sechs Mannschaften sechs Zahlen (1, 2, 3, 4, 5, 6) und fassen sie zu Pärchen zusammen (1-2, 3-4, 5-6). Das wäre schon der erste Spieltag. Es spielt also Mannschaft 1 gegen Mannschaft 2, Mannschaft 3 gegen Mannschaft 4 und Mannschaft 5 gegen Mannschaft 6. Als nächstes suchen wir die Paarungen für den zweiten Spieltag (1-3, 4-5, 2-6), für den dritten (1-4, 2-5, 3-6), für den vierten (1-5, 2-3, 4-6) und für den fünften (1-6, 2-4, 3-5). Die Paarungen sind übrigens von „Hand" berechnet. Mehr Paarungen sind,

Im nächsten Monat (Juni) steht die Fußball-Europameisterschaft ins Haus, und danach wieder eine neue Fußball-Bundesligasaison. Ich stand letztens vor dem Problem, bei einem Bundesliga-Verwaltungsprogramm die einzelnen Mannschaftsnamen und Spielpaarungen eingeben zu müssen.

ohne daß eine doppelt auftritt, nicht möglich.

Jeder gegen jeden

Wie man sieht, ergeben sich aus sechs Zahlen fünf Spieltage mit je drei Paarungen. Setzt man für eine beliebige Anzahl von Mannschaften beziehungsweise Zahlen den Buchstaben 'n' und für die Anzahl der Spieltage den Buchstaben 's' ein, ergibt sich die Formel: s = n - 1. Kommen wir zu einer weiteren Vereinfachung. Wir lassen eine ungerade Anzahl einfach unter den Tisch fallen und berücksichtigen nur eine gerade Anzahl von Mannschaften (4, 6, 8, 10, 12, 14, 16, 18).

Die Spielregeln

Das war alles, was Sie zum Erstellen des Programms wissen müssen. Selbst um die Ausgabe der Paarungen brauchen Sie sich nicht zu kümmern. Dazu stellen wir Ihnen ein Rahmenprogramm zur Verfügung. Es kann wie üblich mit CTRL-S aus dem Magazin auf Ihren eigenen Datenträger abgespeichert werden. Dieses Rahmenprogramm enthält außerdem schon ein vordefiniertes dreidimensionales Feld (Array), in das Sie die einzelnen Paarungen eintragen müssen.

Wir haben diesem Feld den Namen F gegeben (F(T,P,M)). Sie brauchen jetzt nur noch zu wissen, wie in das Feld was einzutragen ist. Dabei gehen wir wie folgt vor: der erste Index des Feldes, ein T, gibt den Spieltag an, der zweite Index (P) die Paarung des jeweiligen Spieltages und der dritte (M) die Mannschaft der jeweiligen Paarung. Greifen wir noch mal die oben genannten Paarungen des ersten Spieltages als Beispiel auf (1-2, 3-4, 5-6). Wir wollen die erste Mannschaft der ersten Paarung des ersten Spieltages eintragen.

Dazu setzen wir für das T (Spieltag) eine 1, für das P (Paarung) eine 1 und für das M (Mannschaft) eine 1 ein (F(1,1,1) und tragen in das so definierte Feld eine 1 ein (Mannschaft 1). Um die Mannschaft 2 einzugeben, gehen wir genauso vor, wir definieren unser Feld (F(1,1,2)) und tragen hier eine 2 ein. Jetzt kommt die zweite Paarung. Für das P wird jetzt eine 2 eingesetzt und für das M wieder eine 1 (F(1,2,1), hier können wir eine 3 (Mannschaft 3) eintragen und so weiter. Die weiteren Einträge werden in der gleichen Weise ausgeführt.

Ein Schiedsrichter muß sein

Bei der Bewertung der Programme gehen wir von der Länge aus, das heißt, es wird jeder Befehl gezählt. Dabei gehört zu einem Befehl alles, was man zwischen zwei Befehlstrennungen, sprich Doppelpunkte, schreiben kann. Sind zwei oder mehrere Programme gleich lang, das heißt, haben sie die gleiche Anzahl von Befehlen, entscheidet die Geschwindigkeit (das schnellere Programm gewinnt). Sind auch dann noch mehrere Programme gleich gut, entscheidet das Los. Der Rechtsweg ist dabei wie immer ausgeschlossen.

Schreiben Sie bitte auf Ihr Listing die Anzahl der Befehle und die Zeit, die es zur Lösung von 18 Mannschaften braucht. Außerdem bitte Ihre Adresse auf dem Listing angeben. Programme, bei denen keine Angaben gemacht sind, können nicht berücksichtigt werden. Ihre Lösung (bitte nur als Listing, keine Kassette oder Diskette) schikken Sie bitte bis zum 1. Juni 1988 (es gilt das Datum des Poststempels) ein.
(Anschrift s. Impressum)

kfp

Listing 3: Verständliche Genauigkeit

```
990 rem "nk = Anzahl der Stellen
992 rem "nach dem Dezimalpunkt
1000 nk=0:z=0
1010 input "(clr)Nachkommastellen: ";nk
1020 input "Zahl ";z
1030 nk=10↑nk
1035 rem "Es werden nur nk
1037 rem "Nachkommastellen
1039 rem "angezeigt
1040 if abs(z*nk)<32767then1060
1050 print"Zu gross!!":goto 1080
1060 z%=z*nk+.5
1070 print"Ergibt: ";z%/nk
1075 rem "Warten auf Taste
1080 wait198,1:poke198,0
1090 ifz>0then1000
```

Fortsetzung von Seite 24

Aus dem Ärmel

Der INPUT-Befehl ist in den meisten Fällen jedoch völlig unbrauchbar. Macht der Benutzer irgendeine unsinnige Eingabe, gibt anstelle von Zahlen Buchstaben ein oder benutzt Editor-Tasten wie HOME oder CLR oder eine der Farbsteuertasten, kommt es entweder zu Fehlermeldungen und Programmabstürzen oder der liebevoll entworfene Bildschirm ist verunstaltet. In Verbindung mit String-Variablen verträgt der INPUT-Befehl eigentlich weder Komma noch Doppelpunkt. Es sei denn, der Benutzer tippt als erstes einen Anführungsstrich ein. Das Zeichen " selbst kann überhaupt nicht eingegeben werden.

Deshalb benutzen erfahrene Programmierer lieber den GET-Befehl, der im Gegensatz zum INPUT-Befehl nur ein Zeichen aufnimmt. Damit besteht die Chance, zu kontrollieren, ob der Benutzer für das Programm sinnvolle Zeichen eingegeben hat. Sie können so auch die Funktionstasten abfragen oder die Funktion anderer Tasten ändern. Im Listing 1 finden Sie ein einfaches Beispiel.

Der GET-Befehl kann auch numerische Variablen verarbeiten. Dies macht aber kaum Sinn, denn es kann immer nur eine Ziffer eingegeben werden. Völlig exotisch ist die Möglichkeit, mehrere Variablen, durch Komma getrennt beim GET-Befehl angeben zu können, von denen aber nur die erste bei Eingabe von Tastatur ein Zeichen empfängt. Besonderheiten, die erst beim GET#-Befehl vernünftige Wirkung zeigen.

Darstellungskünste

Bei der Ausgabe von Variablen tauchen meist dann Schwierigkeiten auf, wenn Sie Zahlen oder Zeichenketten formatiert ausgeben wollen. Die SPC- und TAB-Funktion sind da nur eine kleine Hilfe. Bei kaufmännischen Berechnungen, wo es um Mark und Pfennig geht, möchte der Anwender gerne Zahlenkolonnen sehen, bei denen die Dezimalpunkte untereinanderstehen. Andere BASIC-Dialekte kennen dafür den PRINT USING-Befehl. Im Listing 2 finden Sie eine mögliche Lösung.

FLießkomma-Variablen schleppen oft Nachkommaanteile mit, die dem Betrachter in vielen Fällen nicht mehr allzuviel sagen. Mit der Anweisung in Listing 3 können Sie die Ausgabe auf die gewünschte Genauigkeit reduzieren.

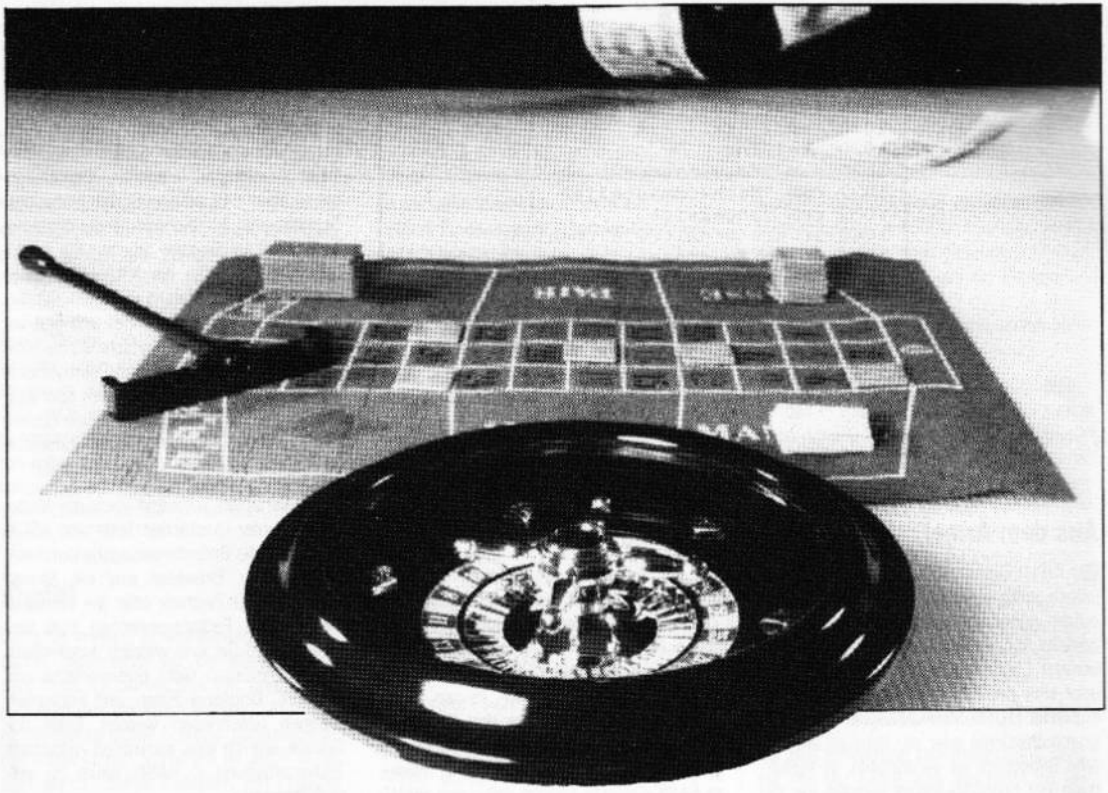
In anderen Fällen möchten man von vornherein nur auf wenige Stellen hinter dem Dezimalpunkt genau rechnen. Kilometerangaben etwa sind mit drei Stellen hinter dem Dezimalpunkt hinreichend genau beschrieben. Nach der Eingabe arbeiten Sie mit Integer-Variablen, die die Kilometerangaben mal 1000, also in Metern enthalten. Mit diesen Variablen wird gerechnet und erst vor der Ausgabe wird die Variable durch 1000 geteilt und an eine Floating-Point-Variable übergeben, die Sie dann anzeigen oder ausdrucken lassen. Listing 4 zeigt eine Lösung des Problems. Arbeiten Sie mit Zahlenfeldern, sparen Sie so sogar Speicherplatz!

String-Variablen enthalten nach der Verarbeitung unter Umständen Texte und ASCII-Werte, die die Bildschirmausgabe durcheinanderbringen. Entweder sind die Strings länger als 40 Zeichen, oder sie enthalten Steuerkodes, Farbsteuerzeichen zum Beispiel. Wollen Sie eine wirklich ansprechendes Bild, müssen recht umfangreiche und in BASIC langsame Filter- und Formatierroutinen geschrieben werden. Einer der Gründe, warum eine annähernd zumutbare Textverarbeitung in BASIC kaum zu verwirklichen ist.

Das BASIC-Beispiel, das Sie sich am Ende der 64er Tips mit CTRL-S abspeichern können, gibt Ihnen einige Anregungen und genug Möglichkeit zum Ausprobieren. RH

```
1000 print"(clr)":
1010 t=0:v=0
1020 input "(down)Tankinhalt":t
1030 input "Verbauch auf 100km";v
1035 rem "Bei 0 geht's raus
1040 if v=0then1100
1045 rem "rk% = Kilometeranteil
1050 rk%=(t/v*100)
1055 rem "rm% = Meteranteil
1060 rm%=((t/v*100)-rk%)*1000+.5
1065 rem "Die Zahl ensteht wieder
1070 r=rk%+rm%/1000
1080 print"Reichweite:"r"km ("t/v*100")"
1085 rem mal wieder warten
1090 wait198,1:poke198,0:goto1010
1100 end
```

Listing 4: ablesbare Kilometerangaben



Casino Commodore

Spiel: Roulette

„Roulette" ist eine Spieltisch-Simulation für einen bis vier Spieler. Benutzen Sie zur Steuerung einen Joystick in Port 2 oder die Cursor-Tasten und RETURN.

Es geht – wie in den real existierenden Spielkasinos – darum, durch Auswahl geeigneter Zahlen oder Zahlenkombinationen eine Voraussage über das Zielfeld der Kugel zu treffen. Das Prinzip: je gezielter die Wette, desto höher der Gewinn. So kann man beim Setzen auf eine einzelne Zahl das 35fache des Einsatzes gewinnen, wettet man dagegen auf „Pair", also auf die Hälfte aller Zahlen, gibt's nur den 1fachen Einsatz. Die genauen Spielpläne und die

Ehe man für eine „todsichere" Gewinnstrategie am Spieltisch Haus und Hof einsetzt, kann man sich in dieser grafisch überzeugenden Casino-Simulation mit dem C64 in finanziell risikolosen Trockenübungen mit dem Metier vertraut machen.

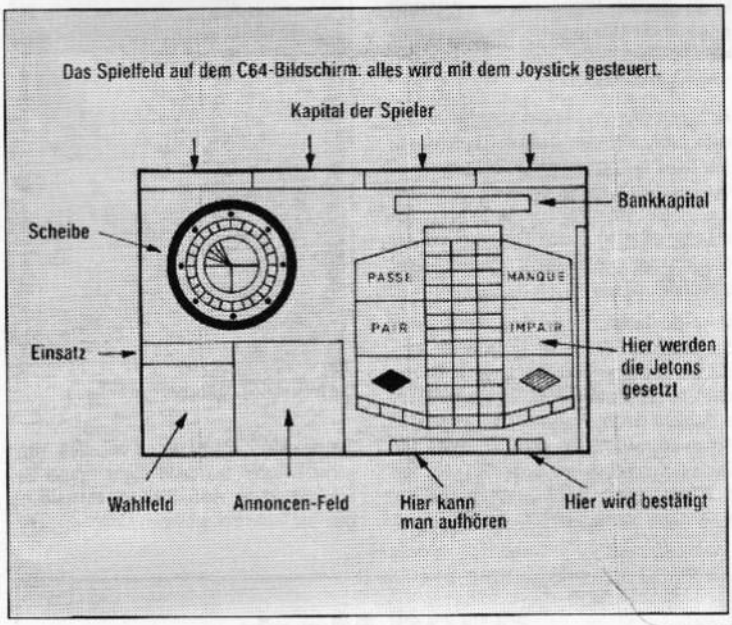
Aufteilung der Scheibe entnehmen Sie bitte den Kästen und Zeichnungen auf diesen Seiten.

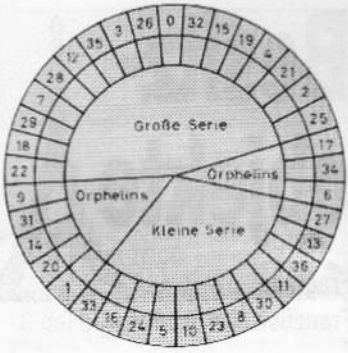
Wie wird gespielt? Nach Auswahl der Zahl der Spieler erscheint das Spielfeld (siehe Zeichnung); danach kann jeder Spieler sein Anfangskapital bestimmen, und zwar mit Joystick/Cursor-hoch/runter – +/-50 DM, -rechts/links – +/-500 DM. Ist dies für alle Spieler geschehen, wandert der Pfeil auf das Wahlfeld, wo die Setzmöglichkeiten angeklickt werden können. Diese Ansagen-Annoncen und eventuelle „Sub-Annoncen" (wie „Dutzende" oder „Kolonnen" nach der Ansage „Zwölf") werden dann jeweils invertiert angezeigt. Der Pfeil gibt „Nicht setzen" vor – so kann man die Wahl rückgängig machen –, mit Joystick-hoch gelangt man auf den Spieltisch, um die einzelnen Nummern auszuwählen.

Wurde die Wahl durch Klicken auf OK bestätigt, kann wie bei der Festlegung des Anfangskapitals der Einsatz bestimmt werden, allerdings in 5er- und 50er-Schritten. Mit 0,– DM Einsatz kann man auch hier seine Wahl annullieren.

Ist der Einsatz gültig, wandert der Pfeil wieder auf das Wahlfeld, damit der **gleiche** Spieler noch mehr setzen kann. Will er das nicht, kommt er mit Joystick-rechts auf das Feld „Nicht setzen", wo er mit RETURN oder dem Feuerknopf seine Eingabe bestätigen muß. Dann kommt der nächste Spieler dran.

Haben alle Spieler gesetzt, gilt „Rien ne va plus", und die Kugel rollt. Sobald sie zum Stillstand kommt, werden Gewinnstand und





So sind die Zahlen und Felder auf der Scheibe angeordnet.

Gewinn-Modi (Dutzend, Kolonne, Farbe, Serie, Pair/Impair) im Annoncen-Feld angezeigt. Ein Druck auf den Feuerknopf bringt Gewinne und Verluste auf den Schirm, und das Ganze beginnt von vorn.

Zum Abbruch des Spiels müssen alle Spieler **ohne zu setzen** auf das Feld „Nicht setzen" gehen. Es wird dann eine Spielbilanz mit Gewinnen und Verlusten angezeigt. Dort kann man mit den Funktionstasten auswählen, ob man aufhören, weitermachen oder noch mal von vorn anfangen möchte. Übrigens: Aus technischen Gründen können Sie dieses Spiel nicht auf eine eigene Diskette überspielen.

F. Leuband/JS

Besitz-objekte

Fragram: Französische Grammatik/Teil 3

Klare Besitzverhältnisse und persönliche Hinweise sollten Sie auch in einer Fremdsprache korrekt formulieren können. Sie ersparen sich so manch unangenehme Verwirrung. Mit Fragram können Sie diesmal selbst ausprobieren, wie sicher Sie in solchen Situationen mit der französischen Sprache umgehen können.

In dieser Folge können Sie mit dem französischen Grammatik-Trainer den richtigen Gebrauch von Personal- und Possessivpronomen üben – die Verwendung von persönlichen und besitzanzeigenden Fürwörtern.

Pronom personnel objet:
– le, la, l', les

Pronom possessif:
– mon, ma, mes
– son, sa, ses

Im Laufe der Übungen werden Sie einigen bekannten Persönlichkeiten begegnen und, wie in der letzten Folge auch, ein paar kleinere Rätsel lösen können, wenn Sie auf den Textzusammenhang der Übungssätze achten.

Fragram vermittelt kein neues Wissen über die französische Grammatik, sondern dient der Einübung und Wiederholung von Grammatikregeln. Sie sollten deshalb einige Grundkenntnisse der französichen Sprache besitzen. Erscheint Ihnen das eine oder andere Wort unbekannt zu sein, können Sie die Bedeutung meist aus dem Sinnzusammenhang herausfinden.

Stets zu Diensten

Im Hauptmenü können Sie mittels der angegebenen Tasten 1 – 6, f1, f8 und RETURN Ihre Wahl treffen. Nach einer Bildschirmseite mit einigen Beispielen zum Eingewöhnen in die gewählte Übung geht es auf Tastendruck in die Übungssätze. Fragram erwartet von Ihnen immer erst einmal eine Eintragung in die Textlücken.

Richtige Lösungen werden vom Flic mit einem bewundernden Ausruf (Lautstärkeregler ruhig mal aufdrehen!) belohnt. Nach jeder Eintragung stehen die Tasten der unteren Bildschirmleiste zur Verfügung. Bei manchen Sätzen dürfen Sie auch mal raten, Irrtümer werden dabei nicht als Fehler gerechnet.

Über die Tastatur Ihres Rechners können Sie sowohl die deutschen Umlaute als auch die französischen Akzente und Sonderzeichen eingeben. In dieser Folge erscheinen außerdem die im Französischen benutzten Anführungsstriche bei wörtlicher Rede. Die Cursor- und Editor-Tasten wie DEL/INS und CLR helfen Ihnen, falls Sie sich bei der Eingabe eines Wortes vertippt haben. Lassen Sie den Flic nicht zu oft rot anlaufen. Vielleicht erhalten Sie dann sogar einmal die Erfolgsmeldung, wenn Sie nach allen Übungen fast fehlerfrei geblieben sind. RH

Die Tastaturbelegung:

französisch		*deutsch*	
Taste	Symbol	Taste	Symbol
←	»	£	ß
↑	ˇ	:	ä
		:	ö
		@	ü
		<	;
		>	:
mit SHIFT			
+	ë	:	Ä
–	ë	:	Ö
£	ë	@	Ü
mit C=			
+	«		
–	a		
u	ü		
£	ö		
c	ç		
j	ï		

Die internationale Tastatur zu Fragram

Am 27. Mai 1988 an Ihrem Kiosk:
INPUT 64, Ausgabe 6/88

Wir bringen unter anderem:

EM '88

Rechtzeitig zum diesjährigen Fußball-Großereignis – der Fußball-Europameisterschaft – bieten wir Ihnen ein Programm an, mit dem Sie die Ergebnisse verwalten, jederzeit Tabellen berechnen und die Torschützenbestenliste führen können. Daß die Fernsehübertragungstermine abrufbar für Sie im Programm enthalten sind, ist nur eine der vielen weiteren Feinheiten. Nur die Vorhersage, wer nun wirklich Europameister wird, war programmtechnisch einfach nicht zu lösen.

Sourcecode-Debugger

Fehlersuche ist immer eine undankbare Aufgabe, und wir behaupten auch nicht, daß sie mit unserem Debugger zum reinsten Vergnügen wird, aber sie wird mit erträglichem Aufwand machbar. Wenn Sie unseren interaktiven Maschinensprachelehrgang kennen, werden Ihnen die meisten Funktionen nicht unbekannt sein. Auf vielfachen Wunsch haben wir den Debugger aus diesem Lehrgang isoliert und als Programm aufbereitet.

Mathe mit Nico

Nach langer Pause und vielen fordernden Briefen ist der kleine Drachen wieder aufgetaucht. Nico beschäftigt sich und Sie mit „proportionalen Zuordnungen" und bietet parallel drei Lösungstechniken an. Die Aufgaben können grafisch experimentell, praktisch erfahrbar und mathematisch rechnerisch bewältigt werden. Daß dieser Lernstoff im siebten Schuljahr behandelt wird, ist vielleicht für die Eltern interessant; daß Nico – obwohl ernst gemeint – nicht viel mit Schule gemein hat, sei ein Hinweis für die Jugendlichen.

c't Magazin für Computertechnik

Ausgabe 6/88 – ab 13. Mai am Kiosk

Grundlagen: Simulation der Umwelt mittels finiter Elemente * Test: Archimedes, ein neuer, schneller und preiswerter 32-Bit-Computer * Software-Know-how: Supercomputer und Parallelrechner-Konzepte * IBM-Tastaturen: Was zwischen Rechner und Tastatur passiert * Der „Protected Mode" des 80286-Prozessors * u.v.a.m.

elrad – Magazin für Elektronik

Ausgabe 6/88 – ab 20. Mai am Kiosk

Bauanleitung MP-Technik: IEC-Schnittstellenumsetzer für E.M.M.A. * Report: Radarwarner fürs Auto * Bauanleitung NF-Technik: Digitale Schallverzögerung * Bauanleitung Audio: IR-Übertragung für Kopfhörer * Report: Kann man Lautsprecherchassis reparieren lassen? * Bauanleitung gegen Sonne: Automatische Markisensteuerung * 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,–

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

Vorname/Zuname

Beruf/Funktion

Straße/Nr.

PLZ Wohnort

Datum/Unterschrift

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

Unterschrift: _____

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

INPUT64-Abonnement Abruf-Coupon

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

Name des Kontoinhabers

Bankleitzahl Konto-Nr.

Geldinstitut Ort des Geldinstitus

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

IMPRESSUM:

INPUT 64 Das elektronische Magazin

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

Technische Anfragen:
nur dienstags von 9.00 — 16.30 Uhr

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

Herausgeber: Christian Heise

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

Redaktionsassistent: Wolfgang Otto

Ständige Mitarbeiter:
Irene Heiner, Peter Sager, Hajo Schulz, Eckart Steffens, Frank Börncke

Vertrieb: Anita Kreutzer

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

Herstellung: Heiner Niens

Lithografie: Reprotechnik Hannover

Druck: SONOPRESS GmbH, Gütersloh

Diskettenherstellung:
SONOPRESS GmbH, Gütersloh

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

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

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

Jahresabonnement: Diskette DM 210,–

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

Verantwortlich:
Christian Persson
Helstorfer Straße 7 · 3000 Hannover 61

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

ISSN 0177-3771

Titelidee: **INPUT 64**
Titelillustration: Michael Thiele, 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.