COMMODORE DISC/64/128

COMMODORE DISC C64/ C128

Nr. 18 DM 19,80 ÖS 168,– SFR19,80
Unverbindliche Preisempfehlung

C64:

Maschinensprache pauken:
Assembler-Kurs 2
Ganz nach Belieben:
Startadresse ändern
Geschick und Grips:
Kontakte
Feinde im Weltall:
Space Enemy

Acht Commodore Programme auf Disc im Heft

166 KByte ohne Abtippen!

128 PC:

Ideal für Tippgemeinschaften:
Lotto-Kosten
Astronomisch:
Super-Orion
Würfelspiel-Automat:
Le Cube
Alle Jahre wieder:
Heizkosten-abrechnung

Alle Programme auf Disc im Heft!

INHALT COMMODOR

CD Intern

Was sich schon in den letzten Monaten deutlich abzeichnete, ist nun zur Gewißheit geworden: Die Zahl der Freunde der COMMODORE DISC steigt ständig.
Grund für uns, auch die Redaktion zu verstärken, zumal die Programmeinsendungen auch stetig zunehmen. Das gilt zumindest für den C64 (er läuft und läuft und läuft . . .), nicht so ganz zufrieden sind wir über den Eingang von Leserprogrammen für den C128. Woran liegt's? Nun, zum einen ist er natürlich nicht so verbreitet wie der 64er, zum anderen wird er vor allem von sogenannten „reinen Anwendern" genutzt, die halt am liebsten nur eine Diskette reinschieben und dann die Tasten drücken, die ihnen die Software vorschreibt. Ein bißchen schade, denn gerade mit den Möglichkeiten des C128 läßt sich im Vergleich zum C64 einiges mehr anfangen.
Es würde uns freuen, wenn wir jetzt den Ehrgeiz einiger 128er-Hobby-Programmierer angestachelt hätten. Und ganz umsonst ist es ja schließlich auch nicht, siehe Annonce „Schreiben Sie gerne?" im Heft . . .
Um bei den Möglichkeiten des C128 zu bleiben: Nehmen wir als Beispiel doch mal das Betriebssystem CP/M Plus, das jeder C128-Käufer gratis mitgeliefert bekommt, auf Diskette als Software und dem eingebauten Mikroprozessor Z80 als Hardware. Als großer „Fortschritt" bei Erscheinen des C128 angekündigt, scheint es sich leider doch ein wenig zu einem „Mauerblümchen" zu entwickeln. Zuwenig „Aufklärungsarbeit" wurde bisher dafür geleistet, nicht zuletzt auch von Commodore selbst. Denken Sie nur an das Handbuch zu CP/M Plus.
Dem wollen wir ein wenig abhelfen, lesen Sie bitte den entsprechenden Artikel in dieser Ausgabe der COMMODORE DISC. Bedenken Sie bitte eins: MS-DOS, das Betriebssystem der großen PCs (Personalcomputer) ist im Prinzip aus CP/M entstanden und wie dieses aufgebaut. Für jemanden, der mit CP/M umzugehen weiß, ist ein späterer Umstieg auf MS-DOS, privat oder beruflich bedingt, fast nur noch ein Kinderspiel.

HAUPTTHEMA: MASCHINENSPRACHE

Was haben wir heute sonst noch zu bieten? Ein roter Faden zieht sich durch das ganze Magazin zur Diskette: Programmieren in Maschinensprache (Assembler).
Da der Befehlssatz des C64 völlig identisch mit dem des C128 ist, sind die betreffenden Berichte (hundertmal schneller als BASIC, Aufstellung der Assembler-Codes) für den C128-Anwender ebenso interessant wie für den C64-Freak. Vor allem können Sie damit das erlernte Wissen des „Assembler-Kurses" auf Diskette, von dem heute Teil 2 erscheint, ausbauen und vertiefen. Es wird übrigens nicht das letzte Mal sein, daß Sie in der COMMODORE DISC Wissenswertes über Maschinensprache erfahren. Und auch Sie werden feststellen: So schwer ist diese ureigene Sprache des Mikroprozessors Ihrer Commodore-Computer doch gar nicht.
Auch die Leser, die ihren Computer gerne zum Spielen benutzen, kommen nicht zu kurz:

ACHT PROGRAMME AUF DER DISC

Kontakte ist ein Denkspiel für den C64, bei dem allerhand Grips erforderlich ist, bei Space Enemy können Sie sich mit Ihrem Spielpartner einen heißen, aber völlig ungefährlichen Kampf auf dem Computer-Bildschirm liefern. Glücksritter werden sicher viel Spaß mit Le Cube haben, dem Würfelspiel-Automaten. Apropos Glück: Eine sehr komfortable Lotto-Verwaltung auf der DISC eignet sich vor allem für Tippgemeinschaften, wie sie in vielen Firmen und im Freundeskreis üblich sind. Natürlich können damit auch Tippzahlen ermittelt werden, aber das Programm kann noch mehr. Es berechnet die gesamten Kosten für die verschiedenen Spielsysteme und nach Wunsch den Anteil des einzelnen Spielers, das gilt natürlich ebenso bei einem Gewinn. Wer für sich alleine sein Glück versuchen möchte, kann das Programm selbstverständlich auch verwenden.
Amateur-Astronomen unter unseren Lesern werden jubeln: Super Orion bietet als Lernprogramm (fast) alles Wissenswerte, das mit dem Weltall und unserem Sonnensystem zusammenhängt. Gute Grafikdarstellungen unterstreichen die Ausführungen des Programmautors auf dem 40-Zeichen-Bildschirm.
Wer aber den 80-Zeichen-Monitor für Grafik nutzen möchte, wird vom BASIC 7.0 des C128 in keiner Weise verwöhnt. Ohne PEEKS und POKES (=Maschinensprache) geht da leider gar nichts, es sei denn, Sie beschäftigen sich einmal näher mit eigens dafür geschaffenen Grafik-„Tools" wie etwa Hi Screen CAD, einem Mal- und Zeichenprogramm für den C128, dessen Testbericht Sie in dieser COMMODORE DISC finden.

DER VIRUS GEHT UM

„Kränkelt" Ihr C64? Laufen plötzlich bislang einwandfreie Programme nicht mehr zu Ihrer Zufriedenheit? Brauchen sie beim Laden doppelt so lange als bisher? Dann hat er sich mit ziemlicher Sicherheit einen Virus eingefangen — ein Glück, wenn's nur der von der Usergruppe „Bayerische Hacker-Post" ist! Er gilt als harmlos, kann aber doch ärgerliche Folgen haben. Wie Sie ihn lokalisieren und ihm den Garaus machen, davon weiß der entsprechende Artikel zu berichten.
Wir hoffen, auch mit dieser Ausgabe der COMMODORE DISC sowohl mit den Programmen auf Diskette als auch mit den Themen im Begleitmagazin wieder eine interessante Mischung gefunden zu haben. War für Sie nichts dabei, so schreiben Sie uns bitte, wie Ihrer Meinung nach die Programme auf DISC oder der Inhalt des Magazins gestaltet sein sollte. Denn schließlich und endlich soll die COMMODORE DISC „Ihr" Computermagazin mit und auf Diskette sein. Sie dürfen sicher sein, daß wir jedes wirklich gute Programm oder bahnbrechende Anregung in der redaktionellen Gestaltung berücksichtigen werden.

Bis zum nächsten Mal.
Ihr
COMMODORE-DISC-Team

E DISC Nr. 18 INHALT

SERIE & SERVICE

Die Diskette in diesem Heft ist weder list- noch kopiergeschützt. Aus verständlichen Gründen können wir daher bei Programmfehlern lediglich Umtauschrecht einräumen. Das Rückgaberecht gegen Kaufpreiserstattung ist ausgeschlossen! Sollte also eines der Programme auf Ihrer Diskette nicht laufen, senden Sie die Diskette an den Verlag zurück, Sie erhalten selbstverständlich eine korrigierte Fassung. Anschrift: Siehe Impressum.


```
0 "COMMODORE DISC18" 18 2A
1     "BOOT.64",8,1:        PRG
9     "DISCLADER64/128"     PRG
0     "--------------------" USR
18    "ASSEMB.KURS2",8:     PRG
34    "MC-ROUTINEN 2"       PRG
6     "STARTADRESSE",8:     PRG
62    "KONTAKTE",8:         PRG
48    "SPACE ENEMY",8:      PRG
0     "--------------------" USR
9     "LOTTO-KOSTEN":       PRG
6     "AUSGABEN    LADEN"   PRG
24    "AUSWERT.   NORMAL"   PRG
79    "AUSWERT.   SYSTEM"   PRG
24    "KOSTEN     NORMAL"   PRG
29    "KOSTEN     SYSTEM"   PRG
15    "SPIEL     '77'    "  PRG
32    "ZAHLENERMITTLUNG"    PRG
135   "SUPER-ORION":        PRG
75    "LE CUBE":            PRG
49    "HEIZKOSTEN":         PRG
0     "--------------------" USR
0     "(C) COPYRIGHT BY"    USR
0     "CA - VERLAG GMBH"    USR
0     "--------------------" USR
9 BLOCKS FREE.
```

☎ TELEFONSERVICE ☎

Alle Experten der COMMODORE-DISC stehen unseren Lesern jeden Mittwoch zwischen 15.00 und 19.00 Uhr zur Beantwortung aller Fragen zur Verfügung unter der Telefonnummer 089/129 80 13. Ebenso der Abo- und Kassettenservice. Einfach anrufen 089/129 80 14!!

AUF DISC IM HEFT

ASSEMBLER-KURS, 2. FOLGE

Nicht kompliziert, sondern implizit

Im 2. Teil unseres Assemblerkurses auf Diskette für den C64 (aber auch für den C128-Anwender, zumindest, was den Befehlsaufbau angeht) stellen wir Ihnen heute gleich fünfzehn neue, recht einfache Anweisungen vor.

Der überwiegende Teil der Assemblerbefehle, um die es heute geht, behandelt die sogenannte implizite Adressierung von Maschinencode, die durchweg nur aus einem Byte besteht.
Was bedeutet „implizit"?
Damit lernen Sie die einfachste aller Adressierungsmöglichkeiten des Mikroprozessors kennen. Durch den Befehl selbst wird nämlich bereits bestimmt, welche Register oder Speicheradressen verändert werden, es wird also keine zusätzliche Adresse mehr angegeben, wie es zum Beispiel bei der Anweisung "LDA $D020" (lade den Inhalt von Speicherstelle $D020 in das Register „Akkumulator") der Fall ist. Befehle mit impliziter Adressierung bestehen immer nur aus einem einzigen Befehlscode-Byte.
Doch vorher zur Ergänzung des letzten Kursteiles (Nr. 1 aus COMMODORE DISC 17) noch die restlichen Anweisungen, die einen Byteinhalt in ein Register laden und den dann in einer anzugebenden Speicherstelle ablegen.

LDX
LoaD X-Register
Diese Anweisung lädt den nachfolgenden, direkten Wert (oder den Inhalt der folgenden Speicherstelle) in das X-Register. LDX entspricht in seiner Form haargenau dem bereits bekannten LDA.
Beispiele: LDX #$0A, LDX $1000.

LDY
LoaD Y-Register
Hierbei verhält es sich analog zur Beschreibung der Anweisung LDX, nur bezieht sich hier jede Operation auf das Y-Register.

STX / STY
STore X(Y)-Register
Diese beiden Befehle erinnern stark an den bereits bekannten STA-Befehl. Bei STX (oder STY) wird der Inhalt des entsprechenden Registers in die angegebene Speicherstelle geschrieben.
Beispiel: STX $4000

Sicherlich ist Ihnen die verwandte Wesensart, die enge Beziehung zwischen dem Akku und den Indexregistern aufgefallen. Im Gegensatz zum Akku werden sie aber vornehmlich als Schleifenzähler verwendet. Für einen schnellen Datenaustausch zwischen diesen Registern sind die folgenden Anweisungen verantwortlich.

TAX / TAY
Transfer Akku into X(Y)
Der Inhalt des Akkus wird in einem der genannten Register abgelegt, der Wert aber nicht verändert. Unmittelbar nach der Ausführung dieses Befehles befindet sich im Akku und im angesprochenen Index-Register (X oder Y) derselbe Wert.

TXA / TYA
Transfer X(Y) into Akku
Bei diesen Anweisungen funktioniert es genau umgekehrt, nun wird der Inhalt des X- bzw. Y-Registers in den Akkumulator geschrieben, auch hier werden die Werte der abgebenden Register nicht geändert.
Nun folgen Anweisungen, die fast immer während einer Zählschleife verwendet werden:

DEC *DECrement Memory*
DEX *DECrement X-Register*
DEY *DECrement Y-Register*
Während DEX und DEY das jeweilige Indexregister um „1" erniedrigen, wirkt der DEC-Befehl auf die nach ihm folgende Speicherstelle, der Inhalt derselben wird ebenfalls um „1" erniedrigt. Der Akku-Inhalt ist davon nicht betroffen.

INC *INCrement Memory*
INX *INCrement X-Register*
INY *INCrement Y-Register*
Diese Befehle bewirken das Gegenteil der vorhin genannten, die jeweiligen Register werden um den Wert „1" erhöht, bei INC der Inhalt der nachfolgenden Speicherstelle.
Was passiert aber, wenn dann zum Beispiel im X-Register der Wert $FF(255) steht und der nachfolgende Befehl wieder ein INX ist, also weiter „erhöht" werden soll? Sie wissen ja, daß kein Byte irgendeines Acht-Bit-Computers einen größeren Wert als $FF(255) annehmen kann. Ganz einfach, der Computer hilft sich selber, er setzt den Registerwert kurzerhand wieder auf „00" (und nicht auf $100(256)). Haben Sie in Ihrer Schleife DEX oder DEY verwendet, sieht es ähnlich aus. Hat der Wert so eines Registers den absoluten Wert „00" erreicht, beginnt der Computer wieder bei $FF.

Zum Schluß: **RTS**
ReTurn from Subroutine
Im wahrsten Sinne des Wortes, dieser Befehl steht gewöhnlich am Ende eines Assembler-Programms (oder in einem Unterprogramm darin) und soll künftig unser BRK (Break) ersetzen, das Sie bei den Beispielen im letzten Kursteil nur immer wieder im verwendeten MONITOR-Programm landen ließ.
Stößt ein Maschinenprogramm auf ein RTS, verzweigt es immer auf den Befehl, der dem vorher getätigten Aufruf folgt.
Auch das BASIC Ihrer Commodore-Computer ist ein Maschinenprogramm. Wenn Sie nun eine Maschinenroutine mit dem BASIC-Befehl „SYS Adresse" oder der Assembler-Anweisung JSR aufrufen, kehrt das Programm nach so einem RTS wieder ins BASIC zurück. Logisch, oder?
Für heute möchten wir es bei Teil 2 des Assembler-Kurses bewenden lassen, falls Ihnen der Kopf aber noch nicht raucht, können Sie sich ruhig die gesamte Aufstellung der Maschinensprache-Anweisungen (Opcodes), ebenfalls in diesem Heft, zu Gemüte führen und dort nachlesen, was da über die Ihnen zwischenzeitlich bekannten Befehle steht. Motto: Ein Autor weiß viel, zwei wissen mehr!

Ralf Trabhardt/bu □

SUPER ORION

Weißt Du, wieviel Sternlein stehen...?

In klaren Sommer-, noch besser Winternächten zeigen sie sich in fast unüberschaubarer Menge dem staunenden Betrachter: Sternbilder und Galaxien. Für die meisten von uns sind's halt nur leuchtende Punkte am Himmel, die ja ganz schön aussehen. Nachdem Sie sich aber dieses Programm für den C128 im 40-Zeichen-Modus betrachtet haben, werden Sie eine ganze Menge mehr darüber wissen.

Super Orion darf getrost zu den guten Lehr- und Lernprogrammen gezählt werden. Der Autor, allem Anschein nach nicht nur Computer-, sondern auch Astronomie-Freak, hat sich den doch recht beachtlichen Speicherplatz und die grafischen Fähigkeiten des C128 zunutze gemacht und ein Programm geschrieben, das dem interessierten Laien auf gut verständliche Weise viele „Geheimnisse" des Weltalls näherbringt. Eine Menge hochauflösender Grafikbilder, allesamt mit den dafür zuständigen Befehlen des C128 erzeugt, ergänzt die Textausführungen.

PROGRAMM-BESCHREIBUNG

Nach dem Start erscheint das Hauptmenü mit folgenden Punkten, die durch Druck auf die entsprechende Zahlentaste aufgerufen werden müssen:

0 – Helligkeit der Sterne
1 – Farben der Sterne
2 – Die Spektralklassen
3 – Die Himmelskoordinaten
4 – Die Sternbilder
5 – Die hellsten Sterne
6 – Unser Planetensystem
7 – Die Galaxien
8 – Besondere Sternkonstellationen
9 – Programm-Ende

OPTIMALE BENUTZERFÜHRUNG

Das Programm ist ideal benutzergeführt, mehr als ein Tastendruck ist oft nicht erforderlich, um im Programm fortzufahren oder wieder zurück zum Menü zu kommen, so daß auch Einsteiger und Anfänger keine großangelegte Erklärung jedes einzelnen Menüpunktes brauchen. Wir möchten daher nur einige interessante erwähnen:

1 – Farben der Sterne

Hier erfahren Sie beispielsweise, daß die Farbe aller sichtbare Sterne mit deren Temperatur zusammenhängt.

3 – Himmelskoordinaten

Die Längen- und Breitengrade auf unserer Erde sind für jeden ein Begriff, doch auch der Himmel ist so eingeteilt. Wie genau, sagt Ihnen dieser Programmpunkt.

4 – Sternbilder

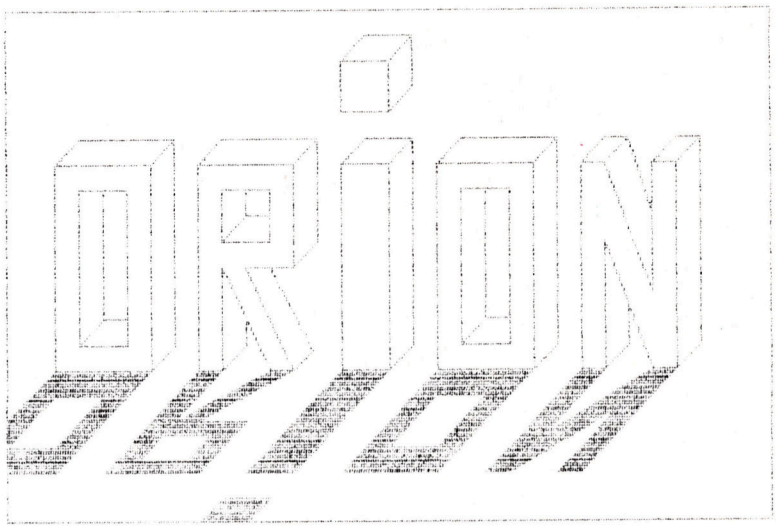
Den „Großen und Kleinen Wagen" kennt vermutlich jeder, dann hackt's aber schon bei den meisten. Nach diesem Menüpunkt können Sie ohne weiteres 24 Sternbilder am Himmel erkennen und identifizieren. Jedes Sternbild wird grafisch dargestellt, die imaginäre Verbindung der einzelnen Fixsterne untereinander angezeigt, und die Namen der wichtigsten Hauptsterne des jeweiligen Sternbildes finden Sie eingetragen.

5 – Die hellsten Sterne

Dieser Menüpunkt bringt eine Aufstellung der hellsten Sterne, deren wissenschaftliche lateinische Bezeichnung inklusive Abkürzung und den Größenangaben.

6 – Unser Planetensystem

Maßstabgerecht im Größen- und Entfernungsverhältnis zur Sonne werden in zwei Grafikbildern die Planeten unseres Sonnensystems dargestellt. Aus Gründen der „Maßstabstreue" mußte der Autor hier zwei Bilder entwickeln, für die nahen, die inneren Planeten (zu denen auch unsere Erde gehört), und die weiter entfernten (Jupiter, Saturn, Pluto).



7 – Die Galaxien

Das sind Ansammlungen von Millionen von Fixsternen innerhalb des Weltalls, auch die „Milchstraße", in der sich unser Sonnensystem befindet, ist so ein „Sternenhaufen". Wußten Sie, daß es drei verschiedene Typen von Galaxien gibt?

8 – Besondere Sternenkonstellationen

Hier hat der Autor in akribischer Kleinarbeit die vier jahreszeitlich bedingten Sternenhimmel grafisch dargestellt, wie sie sich dem Beschauer in unseren Breitengraden im Frühjahr, Sommer, Herbst und Winter präsentieren, wobei er natürlich auch an entsprechenden Erläuterungen nicht gespart hat.
Wir sind überzeugt, daß dieses Dokumentationsprogramm jeden interessierten Leser ansprechen wird. Betrachten Sie es als Ergänzung zum ebenfalls recht gut gelungenen Lernprogramm „Das Weltall" von Dirk Arnold (erschienen auf COMMODORE DISC, Ausgabe 8).

Rolf Lange/her □

LOTTO UND KOSTEN

Haupttreffer

Jeden Samstag und Mittwoch zieht es Millionen Bundesbürger in seinen Bann: Lotto. Zur Abgabe gezielter Tips und zur Berechnung des Aufwands soll Ihnen dieses Programm für den C128 im 40-Zeichen-Modus dienen.

Zunächst ein Wort an alle, die an „Computer-Wunder" glauben: Einen Hauptgewinn kann dieses Programm natürlich auch nicht garantieren, denn der Zufall ist (noch) nicht berechenbar.

Ob Sie mit unserem Lotto-Programm einen Haupttreffer landen können, steht in den Sternen. Auf jeden Fall hilft Ihnen das Programm bei der Berechnung der Einsätze und beim Überprüfen der Gewinn-Zahlen.

Trotzdem ist es eine zuverlässige Hilfe zur Ermittlung von beliebig vielen Tippzahlen und errechnet zugleich auf bequeme Weise die anteiligen Kosten und Gewinnverteilung (Gratulation!) für Tippgemeinschaften oder auch nur für den Einzelspieler.

MENÜGESTEUERT

Das Programm besteht aus folgenden Teilen auf der DISC:

- Lotto-Kosten (=Hauptmenü)
- Ausgaben laden
- Auswertung Normalscheine
- Auswertung Systemscheine
- Kosten Normalscheine
- Kosten Systemscheine
- Spiel 77
- Zahlenermittlung

„Lottokosten" ist das Hauptprogramm, aus dem alle anderen Teile nach Wahl der entsprechenden Taste nachgeladen werden. Allerdings ist jedes Teilprogramm für sich alleine lauffähig und kann auch einzeln geladen und gestartet werden.
Noch eines sollten Sie beachten: Bei den beiden Teilprogrammen zur Kostenermittlung (für Normal- und Systemscheine) wird jeweils ein sequentielles File auf der Diskette angelegt („Normal" oder „System"), was aber logischerweise nur auf einer nicht schreibgeschützten Disk geschehen kann. Wenn Sie also vorhaben, mit diesem Programm zu arbeiten, um sich beispielsweise die Arbeit als „Lotto-Manager" einer Tippgemeinschaft von Arbeitskollegen zu erleichtern, sollten Sie alle die genannten Files auf eine Arbeitsdisk kopieren. Wenn Sie kein Kopierprogramm haben, geht's ohne weiteres auch mit LOAD und SAVE, es handelt sich durchwegs um BASIC-Programme.

DAS HAUPTMENÜ

Nach Start des Programms erscheint ein Menü mit folgenden Auswahlpunkten, die wir im Anschluß erläutern werden:

1... Kosten (Normalscheine)
2... Kosten (Systemscheine)
3... Auswertung (Normalscheine)
4... Auswertung (Systemscheine)
5... Zahlenermittlung
6... Spiel 77
7... Ende
8... Ausgaben

Diese Menüpunkte werden nach Druck auf die entsprechende Zahlentaste aktiviert, das heißt nachgeladen und automatisch gestartet. Die Abfragen zu den nötigen Eingaben sind gut erläutert und eigentlich für je-

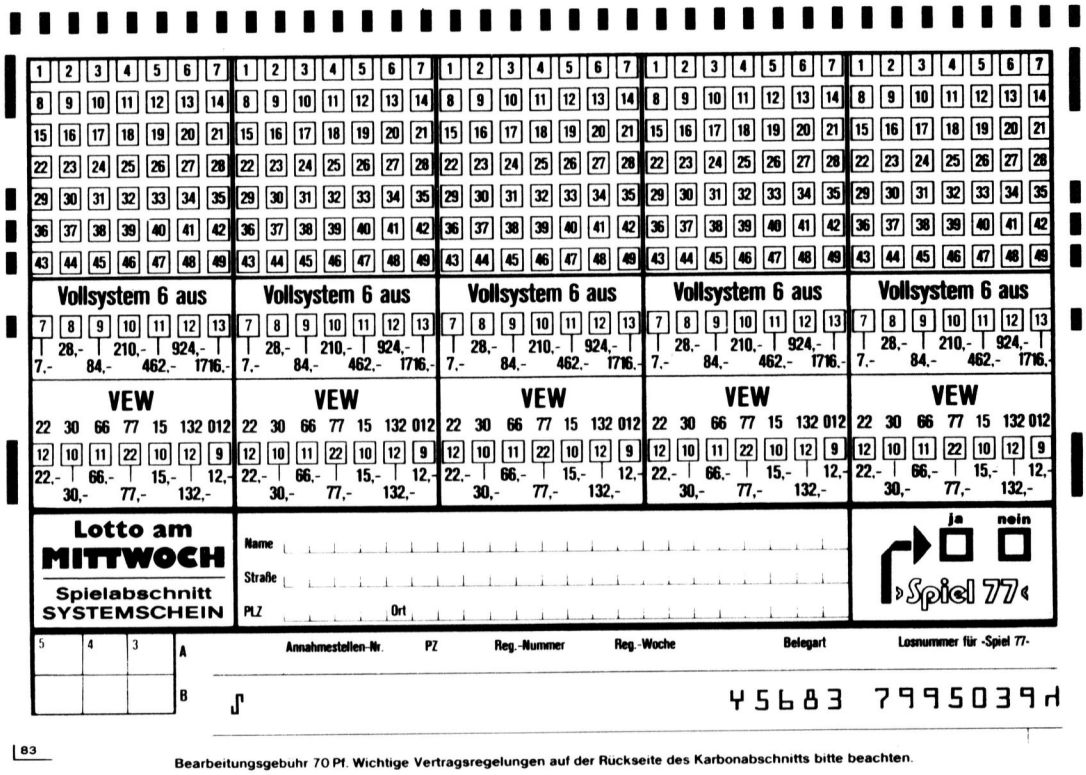
den verständlich, trotzdem möchten wir sie noch einmal kurz streifen.

Menüpunkt 1 und 2 = Kosten
Bevor Sie beim Lotto etwas gewinnen können, kostet es zunächst einmal Geld: Den Einsatz und die Gebühren. Wer nur ein Feld auf dem Schein ausfüllt, kann das sicher noch im Kopf ausrechnen, komplizierter wird's schon bei der Abgabe mehrerer Tips oder gar beim Spielen unterschiedlicher Systeme.
Folgen Sie bei diesen Menüpunkten lediglich den Fragen des Programms und machen dazu Ihre Eingaben. In Sekundenschnelle ermittelt das Programm die gesamten Kosten inklusive Gebühren, diese Aufstellung kann auch mit einem der üblichen Drucker (serieller Anschluß) schwarz auf weiß ausgedruckt werden.

Menüpunkt 3 und 4 = Auswertung
Gerade derjenige, der mehrere Felder eines Tippscheines ausgefüllt oder die seiner Tippartner überprüfen soll, weiß, wie mühselig ein Zahlenvergleich „per Hand" sein kann. Auch das übernimmt das Programm für Sie, Sie müssen lediglich zuerst die richtigen, also die gezogenen Lottozahlen eingeben und anschließend die, die Sie selbst getippt haben. Die Zahlenreihen werden verglichen, Gewinne werden in der entsprechenden Klasse angezeigt. Hier müssen Sie allerdings darauf achten, daß bei der Eingabe von einstelligen Zahlen (also von 1 bis 9) immer eine „0", oder die SPACE-Taste voranzustellen ist. Die andere Möglichkeit ist, nach der Eingabe einer einstelligen Zahl (ohne „Null" oder SPACE) dann RETURN zu drücken, damit das Programm fortfährt. Zweistellige Zahlen werden jedoch sofort übernommen, ohne RETURN oder SPACE. Im Punkt 4 werden Systemscheine ausgewertet, mit Angabe der Gewinne in den verschiedenen Gewinnklassen. Es können jedoch nur VOLL-Systemscheine berücksichtigt werden, da Tippreihen nach dem VEW-System (verkürzte Auswahl) nach einem anderen Gewinnschema berechnet werden. Diese diversen Schablonen auch noch ins Programm aufzunehmen, hätte dies dann doch zu umfangreich gemacht.
Ebenso wie bei den Kosten können Sie auch hier die ermittelten Zahlentabellen auf dem Drucker ausgeben lassen.

FÜR BEQUEME: TIPPZAHLEN AUS DEM COMPUTER

Der wohl wichtigste Punkt dürfte Nummer 5 sein. Hier können Sie je nach vorheriger Wahl 6 bis 26 Zahlen für mögliche Spielreihen ermitteln lassen, egal, ob Sie mit dem VOLL- oder VEW-System spielen. Sollten Sie als „bayrischer" Lotto-Tipper hier Unterschiede zu den Bezeichnungen auf Ihrem Lottoschein finden, so spielt das keine Rolle, das Kind hat nur einen anderen Namen, der Effekt ist derselbe.



Für den Normalbürger steckt ein Systemschein voller Rätsel. Lotto spielen mit System wird mit dem Programm Lotto-Kosten zum Kinderspiel.

Menüpunkt 6 = Spiel 77
Viele Lottofreunde versuchen noch zusätzlich mit der auf dem Wettschein bereits eingetragenen Zahl des „Spiel 77" ihr Gewinnkonto aufzubessern. Die Überprüfung und Auswertung dieser immerhin siebenstelligen Zahl, bei der die Ziffern von rechts nach links für Gewinne maßgebend sind, übernimmt dieser Menüpunkt.

Menüpunkt 7 = Ende
Damit beenden Sie dieses Lotto-Verwaltungsprogramm. Achtung: Es wird ein RESET durchgeführt, das Programm muß zum erneuten Start vorher wieder geladen werden.

Menüpunkt 8 = Ausgaben
In diesem Teilprogramm werden die Ausgaben und Kosten aller während des Programmverlaufs abgegebenen Tips, je nach System registriert, auf dem Bildschirm angezeigt und diese aktuelle Kostensituation auf Diskette abgespeichert. Dieses so entstandene File wird dann bei der nächsten „Lotto-Runde" nachgeladen und im Programm übernommen, so daß Sie immer auf dem neuesten Stand sind. Aber bitte immer daran denken: Abspeichern geht nur auf einer nicht schreibgeschützten Diskette, die eine Kerbe rechts vom Etikett besitzt.
Und zum Schluß noch ein Hinweis: In den Überschriften der einzelnen Teilprogramme ist immer vom „Mittwochslotto" die Rede, was aber in keinem Fall heißen soll, daß sich dieses Programm nicht ebenso für Spieler oder Tippgemeinschaften eignet, die lieber am Samstag (nach Rudi Carell oder Ohnsorg-Theater) ihr Glück versuchen möchten . . .

Laszlo Hauser/bu □

Kontakte

Kontakte ist ein Spiel, an dem sich bis zu acht Personen beteiligen können. Es handelt sich hier um ein Brettspiel, bei dem quadratische Karten innerhalb des Spielfeldes so aneinandergelegt werden, daß sich, ähnlich wie beim Domino, Kanten mit gleichem Muster berühren.

Das Spielfeld faßt 150 Karten, die in 34 verschiedenen Mustern vorliegen. Ziel ist es, das Spielfeld nach Möglichkeit völlig auszufüllen. Doch wird dies nicht immer gelingen, denn Fehler beim Auslegen rächen sich, indem einzelne Plätze nicht mehr belegbar sind, weil keine dazu passende Karte existiert.
Nach dem Start wird zuerst versucht, vom Diskettenlaufwerk eine Highscoretable zu laden. Bei Erfolg wird die Ehrentafel angezeigt. Im anderen Falle haben Sie noch Gelegenheit, die Diskette mit der Datei einzulegen. Haben Sie keine Tabelle oder überhaupt keine Floppy zur Verfügung, geben Sie ‚N' ein, ansonsten ‚J'. Im letzten Falle wird nun nochmal versucht zu laden. Mit RETURN verlassen Sie die Anzeige der Ehrentafel. Nun geben Sie die Anzahl der Mitspieler ein und im nächsten Bild für jeden Mitspieler die Ziffern 1 oder 2 für Joystick beziehungsweise 3 für Eingabe über Tastatur.
Beide Eingaben quittieren Sie mit ‚J', wenn alles in Ordnung ist, oder mit ‚N', wenn Sie sich vertan haben, worauf Sie neu eingeben können.
Nun kann das Spiel beginnen. Das Spielfeld erscheint und in dessen Mitte befindet sich die erste Karte. An diese muß zuerst angelegt werden. Außerdem sind in den Ecken noch vier rote Karten, an diese kann aber nur angelegt werden, wenn gleichzeitig auch an eine grüne Karte angelegt wird.
An der rechten Seite wird ein Feld aufgebaut mit den Nummern aller Mitspieler sowie den erzielten Punkten. Diese erscheinen in blauer Farbe, nur ein Feld ist grün. Der Spieler mit dieser Nummer ist an der Reihe, seine Karten zu setzen.
Dazu befinden sich in der obersten Reihe acht grüne Karten, von denen eine gelb blinkt. Der Spieler kann in jeder Runde vier Karten ins Spielfeld setzen. Dazu muß er erst eine Karte auswählen. Das kann er mit Joystick oder Cursor rechts–links. Welchen Joystick er dazu benutzt oder ob es per Tastatur geschieht, wurde ja am Anfang für jeden Spieler festgelegt. Zur Kontrolle wird dies noch einmal rechts oben angezeigt.
Hat man eine Karte ausgewählt, sollte noch überprüft werden, ob sie auch richtig liegt. Mit einem Druck auf den Feuerknopf oder die Spacetaste kann die Karte in die richtige Lage gedreht werden. Danach wird sie mit Joy oder Cursor abwärts in das Spielfeld geholt. Dort kann sie nun frei hin- und herbewegt werden, bis sie an der richtigen Seite an einer bereits liegenden grünen Karte anliegt. Die beiden sich berührenden Karten müssen nun mit einer oder, falls vorhanden, zwei Linien, Kontakt bekommen. Das hat dem Spiel auch den Namen gegeben.
Es wird bei weiterem Spielen vorkommen, daß eine neu hinzugelegte Karte nicht nur mit einer, sondern mit mehreren anderen Karten Kontakt erhält. Immer aber müssen die aneinanderstoßenden Seiten, soweit sie Linien enthalten, auch auf der Gegenkarte die entsprechenden Linien finden. Es ist nicht möglich, zwei Karten aneinanderzulegen, deren Muster nicht zueinander passen. Es können sich jedoch Kanten berühren, die gar keine Linien haben. Bedingung ist aber, daß mindestens eine Linie zu einer grünen Karte in Kontakt tritt.
Mit Druck auf Feuer oder Space wird die Karte abgelegt.
So können nacheinander vier Karten von oben ins Feld gebracht und angelegt werden. Bei jedem Anlegen werden dem Spieler für jeden geschlossenen Kontakt soviele Punkte gutgeschrieben, wie oben rechts angezeigt sind. Dabei werden die beiden Zahlen miteinander multipliziert. Für die erste Karte gibt es 2 x 1 Punkte pro Kontakt, für die zweite Karte 6 x 1 und so fort. Die erste Zahl beginnt in jeder Runde mit 3 und zählt bis zur vierten Karte bis zu zwölf hoch. Die zweite Zahl ist in der Regel eine Eins. Doch das ändert sich, sobald es zu Kontakten mit den roten Karten kommt. In diesem Falle werden zur zweiten Zahl pro rotem Kontakt zehn addiert. Durch mehrfaches Anlegen an rote Karten kann diese Zahl mit etwas Glück leicht bis hundert und höher getrieben werden. Das bedeutet dann aber auch pro Kontakt 300 und mehr Punkte. Aus dem Gesagten geht hervor, daß es günstig ist, nicht sofort jeden einzelnen roten Kontakt zu schließen, sondern erst mehrere vorzubereiten, um dann möglichst viele kurz hintereinander zu schließen. Die zweite Zahl wird nach jedem Anlegen einer Karte um eins vermindert, bis wieder nur noch mit eins multipliziert wird.
Nun wird es immer mal vorkommen, daß eine Karte ins Feld geholt wurde, die doch nicht richtig gedreht war, so daß sie nicht angelegt werden kann. Immer dann, wenn versucht wird, eine Karte auf einen Platz zu legen, an den sie nicht paßt, oder auf dem schon eine andere liegt, so wird diese Karte gedreht, wie dies auch oberhalb des Spielfeldes geschieht. Hier aber kostet das Punkte, und zwar in der ersten Runde sieben und in jeder weiteren zusätzlich sieben Punkte. In der zehnten Runde werden für so einen Fehler also schon 70 Punkte abgezogen. Ein Grund also, so etwas zu vermeiden. Eine Karte, die sich gar nicht anlegen läßt, kann man an den oberen Rand des Feldes führen, dort den Joystick nach oben und gleichzeitig Feuer drücken. Die Karte wird wieder oben abgelegt. Allerdings auch hier Punktabzug. Außerdem zählt diese Karte, als ob sie angelegt wurde, es können also nur drei Karten benutzt werden. Sollten Sie gegen

Ende des Spieles unter den zur Auswahl stehenden Karten keine finden, die Sie anlegen wollen oder können, so können Sie vom Auswahlfeld mit Joy hoch und Feuer das Spiel an den nächsten Spieler übergeben. Dasselbe geschieht automatisch, wenn vier Karten angelegt wurden. Bei Bedienung über Tastatur geben Sie statt Joy hoch und Feuer „ ↑ " (Pfeil hoch) ein.
Alle Spieler erhalten zu Anfang das gleiche Spielfeld und in jeder Runde die gleichen Karten zur Auswahl wie der erste Spieler. So haben alle dieselbe Chance und es liegt an jedem selbst, wie geschickt er die gebotenen Möglichkeiten nutzt.
Zu erwähnen wäre noch, daß es für die letzten 15 anzulegenden Karten noch einen Zusatzbonus gibt, der sich von 100 bis 1500 steigert. Dieser Bonus wird aber geringer, je öfter der Spieler auf Anlegen einer Karte verzichten mußte.
Wenn ein Spieler sein Spielfeld bis zum letzten Platz ausgelegt hat, färbt sich sein Punktekonto rot. Er hat damit das Spielende erreicht, die anderen können noch weiter spielen. Es kann aber vorkommen, daß ein Spieler vorzeitig Schluß machen will, weil er sieht, daß er die letzten Plätze nicht mehr belegen kann. In diesem Falle bewirkt F8 das Ende für diesen Spieler. Das Spiel ist aber erst zu Ende, wenn alle Spieler ihr Ziel erreicht oder F8 gedrückt haben.
Am Ende des Spieles bleibt das letzte Bild noch erhalten, bis es durch Eingabe eines beliebigen Zeichens gelöscht wird. Darauf werden die Spieler nach ihrem Namen gefragt. Hier können bis zu vier Zeichen eingegeben werden. Diese dienen zum Eintrag in die Ehrentafel, die nach Eingabe des letzten Namens angezeigt wird. Nach Return wird, falls die Floppy ready ist, diese Tabelle weggespeichert. Sollte sich noch eine alte Tabelle dieses Spieles darauf befinden, wird diese ohne Warnung gelöscht.

G. Kramer □

STARTADRESSEN ÄNDERN

Zwei Bytes, auf die's ankommt!

Manchmal möchte man aus der Haut fahren: Ein Programm wurde geladen – und nach dem Startbefehl tut sich gar nichts.

„Absolut" gespeicherte Programme müssen auch wieder so geladen werden: Beim C64 mit dem Zusatz ",8,1", beim 128er mit der Anweisung "BLOAD". Warum? Weil nur dann der Computer auf die zu jedem(!) Programm abgespeicherte Anfangsadresse zugreift, die in Form von zwei Bytes (High- und Low-Byte) beim Speichern ebenfalls auf der Disk eingetragen wird. Nur so werden beispielsweise Maschinensprache-Programme auch garantiert wieder an den Speicherplatz geladen, in dem sie erstellt wurden und in dem sie auch ausschließlich lauffähig sind, nur so ist gewährleistet, daß ein hochauflösendes Grafikbild wieder an seinen Platz kommt (in der Regel ab Speicherstelle $2000 (8192), das gilt beim C128 genauso).
Würden Sie diese genannten, besonderen Ladeanweisungen nämlich einfach weglassen und so ein File mit "LOAD" (oder "DLOAD") zu laden versuchen, würde dies automatisch immer an den Anfang des BASIC-Speichers geschehen, der liegt beim C64 ab $0801 (2049), beim C128 bei $1C01 (7169). Was bei reinen BASIC-Programmen durchaus wünschenswert ist, kann bei Maschinensprache-, Grafik- oder Sprite-Daten, die alle absolut gespeichert wurden, nur in die Hose gehen – oft stürzt der Computer sogar ab.

SINNVOLLE ÄNDERUNGEN EINER START-ADRESSE

Oft kann es aber sehr nützlich sein, so eine Startadresse zu ändern. Beispiele dafür gäbe es genug, zunächst etwas, das vor allem C128-Besitzern immer wieder passiert: Ein BASIC-Programm, das im Programmverlauf auch die hochauflösende Grafik verwendet, liegt nach dem Abspeichern nicht am „normalen" BASIC-Anfang 7169, sondern bei $4000 (16384). Die Erklärung ist recht einfach. Das Programm wurde vor dem Abspeichern zum Testen nochmals gestartet, für gut befunden und in zufriedener Stimmung mit DSAVE auf Diskette gespeichert – ohne vorher den mit dem GRAPHIC-Befehl verschobenen BASIC-Anfang (nach 16384) mit der Anweisung GRAPHIC CLR wieder richtig zu stellen (zurückverschieben nach 7169).
Oder: Ein Assembler-Freak würde gerne ein im C64 erstelltes Maschinenprogramm auch in den C128 laden, mit dem dort eingebauten Maschinensprache-Monitor weiterarbeiten und es an den C128 anpassen (bei einfachen Maschinenprogrammen geht das ohne weiteres). Doch das Vertrackte an der Sache ist, daß das C64-Assemblerprogramm im Speicherbereich $C000 (49152) erstellt und abgespeichert wurde, was aber beim C128 durch das ROM des Bildschirm-Editors belegt ist. Hier wäre der freie Bereich ab $1300 (4864) oder jeder andere freie BASIC-Speicherbereich unter $4000 wünschenswert.

KINDERLEICHT ZU BEDIENEN

Dieses kleine Utility-Programm für den C64 ändert nun jede Startadresse eines entsprechenden Files (ausgenommen relative und sequentielle, aber bei denen ist es auch nicht erforderlich) in die von Ihnen gewünschte um. Dabei ist es dem Programm völlig egal, ob es sich auf der Diskette, die Sie gerade bearbeiten möchten, um C64- oder 128er-Files handelt. Es wird nämlich auf die Programme oder die abgespeicherten Daten gar nicht zugegriffen, sondern lediglich auf die dazugehörigen Startadressen, und zwar in Form der Disketten-Direktzugriffsbefehle „B–P" (setzt den Pufferzeiger) und „U2" (diese Anweisung schreibt einen neuen Speicherinhalt in einen bestimmten Block auf Diskette zurück, ohne den Wert des Bytes in der Null-Position zu ändern).

ANWEISUNGEN ZUM PROGRAMM

Nach dem Programmstart müssen Sie den File-Namen eingeben, dessen Anfangsadresse Sie wissen und gegebenenfalls ändern möchten.
Bitte vergewissern Sie sich vorher über die gewünschten Filenamen, dieses Utility besitzt keine DIRECTORY-Funktion. Danach wird die entsprechende Startadresse ausgelesen und als Dezimalzahl angezeigt. Falls Sie diese nun ändern möchten, geben Sie bitte diese Zahl ebenfalls dezimal ein.

Das Programm zerlegt jetzt Ihre Eingabe selbständig in High- und Low-Byte und trägt dann diese Information auf der Diskette ein, die bisher gültigen beiden Bytes werden überschrieben. Diese Änderung hat solange Bestand, bis sie erneut geändert wird. Aufgrund Ihrer Arbeit am Computer werden Sie sicher am besten selbst herausfinden, wann es für Sie vorteilhaft ist, die Startadresse eines Files zu ändern – dieses kleine Programm steht dann für diese Aufgabe bereit.

Markus Nix/tc □

Heizkostenabrechnung

Wie jedes Jahr werden Ihnen auch diesmal die Heizkosten von Ihrem Vermieter oder der beauftragten Hausverwaltung präsentiert. Und immer wieder taucht die Frage auf: Ist sie exakt in Ordnung?

Für die meisten Mieter ist so eine vom Computer erstellte Abrechnung ein Buch mit sieben Siegeln, wird als gegeben hingenommen und ohne Kommentar bezahlt.

Das muß nicht so sein. Dies zeigt dieses Programm für den 128PC im 80-Zeichen-Modus. Es überprüft unbestechlich die Angaben des Rechnungsstellers auf Richtigkeit.

BEZAHLEN IST GUT – KONTROLLE BESSER

Allerdings – eins kann das Programm nicht: Die internen Angaben des Vermieters überprüfen, dafür muß der Mieter selbst sorgen, indem er Einsicht in die Abrechnungsunterlagen des Vermieters nimmt. (Die Möglichkeit dazu ist übrigens gesetzlich verankert.)
Haben Sie sich von deren Richtigkeit überzeugt, errechnet das Programm exakt die der Abrechnung zugrunde liegenden Angaben.
Geben Sie die aus dem Abrechnungsformular ersichtlichen Daten ein, berücksichtigen Sie dabei auch eventuell geleistete Vorauszahlungen, so wird eine in Frage kommende Gutschrift oder Nachzahlung ermittelt.

AUTOR BIETET ANPASSUNG AN

Außerdem ist im Programm eine kleine Information eingebaut, die auf die Rechtsgrundlage sowie die für die Abrechnung geltende Formel hinweist.
Der Autor erklärt sich beispielgebend für alle anderen bereit, das Programm an besondere Abrechnungsmodalitäten (andere prozentuale Verteilung usw.) des Benutzers anzupassen. Schicken Sie dazu lediglich eine Kopie der Originalabrechnung mit Leerdisk und Freiumschlag an:
Günter Kl.-übbing, Eschenweg 8, 4290 Bocholt.

bu □

SPACE ENEMY

Feinde im All

Ob sich zwei Freunde eine „heiße" Schneeballschlacht liefern oder per Joystick auf dem Computerbildschirm mit „Geschossen" traktieren, der Effekt bleibt immer derselbe: Es tut nicht weh, wenn mal einer trifft.

Dieses „Ballerspiel im Weltraum" dreht sich um zwei feindliche Raumschiffe, von denen jedes fünf Leben besitzt und die bestrebt sein sollten, dem Gegner plazierte Treffer zu verpassen. Gedacht ist es allerdings für zwei Spieler (und zwei Joysticks – Port 1 und Port 2), einer allein wird logischerweise nicht den Spaß daran haben, der vom Programmautor vorgesehen ist.

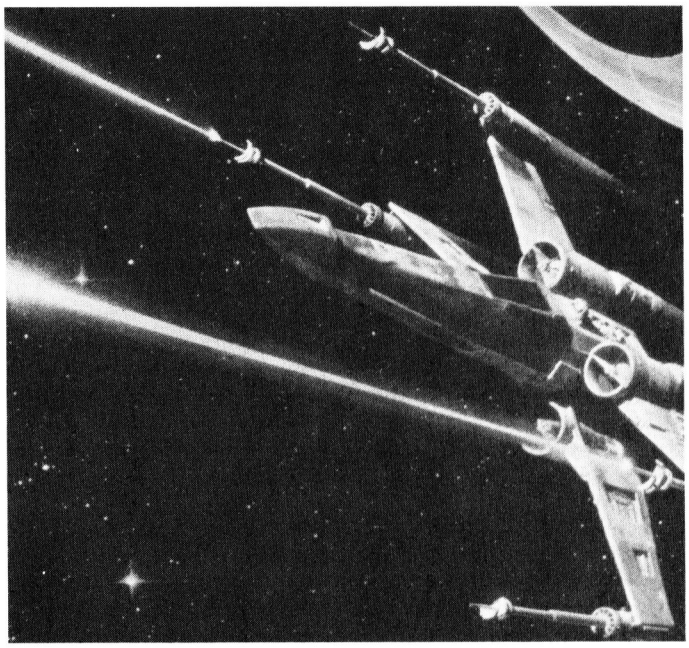
SPIEL-HANDLUNG

Zwei feindliche Raumschiffe stehen sich gegenüber, zunächst noch in sicherem Schutz an der linken und rechten unteren Ecke. Beide Spieler müssen nun bestrebt sein, das gegnerische Raumschiff zu treffen (gefeuert wird mit Druck auf den Feuerknopf), damit die Anzahl seiner Leben möglichst rasch abnimmt. Glau-

DER KÜRZESTE WEG IST NICHT IMMER DER SICHERSTE

ben Sie uns, das ist gar nicht so einfach. Nicht nur, daß der Gegner natürlich auch nicht auf den Kopf gefallen ist (das hoffen wir wenigstens) und kräftig zurückballern wird, da gibt's auch noch die Widrigkeiten eines unwegsamen Geländes, das Sie mit Ihrem Raumschiff durchfliegen müssen. Jede Kollision mit den Felsbrocken oder einer anderen Geländedarstellung auf dem Bildschirm bewirkt dasselbe wie ein gegnerischer Treffer: Sie verlieren ein Raumschiff.

Auf dem Bildschirmausschnitt ist zwar unschwer zu erkennen, daß ein relativ kurzer Weg zum anderen, feindlichen Raumschiff führt, aber auch da ist Vorsicht geboten. Der schmale Durchlaß wird stetig von einem „tödlichen" Energiestrahl bestrichen, so daß Sie den passenden Moment für die Durchfahrt abwarten müssen – und da kann's schon sein, daß der Strahl schneller ist als Sie oder daß der Gegner nur diesen günstigen Augenblick, an dem Sie sich mehr darauf konzentrieren, dem Energiestrahl auszuweichen, abwartet und Ihnen einen Volltreffer versetzt.



Aber auch ein sogenanntes „Ballerspiel" kann durch solche Fußangeln recht unterhaltsam und spannend sein.

NEUER START = TASTE „N"

Tja, die letzte berechtigte Frage bleibt dann noch: Was passiert eigentlich, wenn ein (oder beide) Gegner alle Raumschiffe (=Leben) verloren hat? Dann ist dieser Spieldurchgang zuende, doch mit Druck auf die Taste „N" kann ein neuer „Weltraum-Kampf" gestartet werden. *Laurenz Förnsinn/bu* □

LE CUBE

Die Würfel sind gefallen!

Schon die alten Römer haben's gern betrieben: das Würfelspiel. Tja, wenn die damals schon den C128 gekannt hätten . . .

Dieser Würfelspiel-Automat für den 128PC im 40-Zeichen-Modus zeichnet sich vor allem dadurch aus, daß hier nicht erst durch Sonderspiele ein paar Mark dazugewonnen werden können, sondern mit der „Risiko-leiter" gleich bis zu 90 Mark riskiert und auch gewonnen werden können. Zudem hat der Autor an attraktiven Farb- und Sound-Effekten nicht gespart, so daß der Spielspaß praktisch schon vorprogrammiert ist.

SPIEL-ANLEITUNG

Nach dem Start müssen Sie etwa 20 Sekunden warten, bis es richtig los geht. Dann können Sie mit Druck auf die Taste „S" sofort mit dem Spiel beginnen. Beim allerersten Mal sollten Sie sich aber doch vorher die Anleitung zu diesem Würfelspiel ansehen, die Sie mit der Taste „A" anwählen können. Hier ist sie nochmals zum Nachlesen:
Es stehen sechs Würfel zur Verfügung, das Würfeln besorgt der Computer. Haben Sie nach der ersten Ziehung bereits drei oder mehr Richtige (also mindestens drei Würfel mit derselben Augenanzahl), so kommen Sie automatisch in die „Risiko-Leiter" am linken Bildschirmrand. Sie können nun den gezeigten Gewinnbetrag (die Zahlen blinken) mit der Taste „N" annehmen (das machen ganz Vorsichtige, die auf Nummer Sicher gehen möchten) oder mit der Taste „S" erhöhen, ab 30 Pfennig bis 90 Mark. Jedoch Achtung, es hängt immer von der momentanen Laune Ihres C128 ab, wie hoch Sie gehen können, ganz nach Gutdünken wirft er Sie dann plötzlich aus der „Risiko-Leiter" raus, und mit einem Gewinn ist es Essig. Aber das ist eben der Grund, warum die „Risiko-Leiter" so heißt.

KEIN WÜRFEL RICHTIG?

Natürlich kann's das auch geben, aber Sie bekommen eine zweite Würfel-Chance. Sie müssen dazu dem C128 mitteilen, welche gezogenen Würfelwerte er wieder löschen und neu auswerfen soll. Mindestens vier Würfelwerte müssen gelöscht werden, niedrigere Zahlen nimmt der Computer nicht an. Diese Zahlen werden in chronologischer Reihenfolge eingegeben, also etwa „1346" oder „2356". Welcher dargestellte Würfel welche Nummer besitzt, ist auf dem Spielfeld auf dem Bildschirm sehr gut gekennzeichnet.
Ihr Startkapital beziehungsweise Guthaben zu Spielbeginn beträgt zehn Mark, wovon jedoch beim ersten Würfeldurchgang bereits 30 Pfennige Einsatz abgezogen und somit nur noch 9,70 Mark im entsprechenden Feld auf dem Bildschirm angezeigt werden. Aufgrund folgenden Gewinnplanes können Sie Ihr Startkapital vermehren:
3 Richtige: –,30 bis 90 Mark (durch d. „Risiko-Leiter")
4 Richtige: 2,40 Mark
5 Richtige: 20 Mark
6 Richtige: 90 Mark
Außerdem brauchen Sie zum Bedienen dieses Programms noch folgende Tasten:
Taste „S": Risko – Stop – Risiko;
Taste „A": Auszahlen;
Taste „N": Betrag annehmen.
Das Spiel kann alleine oder mit mehreren Personen „gezockt" werden, der Autor und wir hoffen, daß Sie genauso viel Spaß daran haben werden wie er und seine Freunde. *Siegfried Janka/her* □

UND ES SORTIERT DOCH

Im Programm „Disc Box" (COMMODORE DISC 10) dürfte wohl ein Fehlerteufel im Spiel sein. Im Unterprogramm „Sortieren" tut sich gar nichts, woran mag das wohl liegen?
Peter Scholtysik
Duisburg

Vielleicht liegt's gar nicht am Druckfehlerteufel, sondern nur an der Programmbedienung. Vor dem Sortieren müssen Sie unbedingt mit der Taste F7 den Punkt „Daten pflegen" aufrufen. Wenn jetzt das dazugehörende Menü erscheint, wählen Sie mit der F3-Taste die Funktion „Kennzeichen" an und beantworten die Fragen, die das Programm Ihnen stellt (siehe Programmbeschreibung Seite 14). Nach dieser Zuordnung funktioniert auch das Sortieren.

VERSCHOBENER AUSDRUCK

Ich habe schon einige Ausgaben der COMMODORE DISC, von denen ich die verschiedensten Programme sehr gut einsetzen konnte. Ganz begeistert war ich von dem Programm in COMMODORE DISC 13, „Hires-Lores-Print", weil das Programm speziell für einen Panasonic-Drucker entwickelt wurde. Beim Ausprobieren mußte ich feststellen, daß beim Ausdruck der Grafik der Drucker immer einen zu großen Zeilensprung macht. Wie kann ich erreichen, daß die Grafik zusammenhängend ausgedruckt wird?
Horst Schröder
Hamburg

Das Problem dürfte rasch behoben sein, vorausgesetzt, Ihr Panasonic 1080 besitzt dieselbe DIP-Schalterordnung wie ein Panasonic 1091, für den das Programm entwickelt wurde. Der DIP-Schalter 3 ist dort dafür verantwortlich, ob nach jeder Druckzeile ein Zeilenvorschub (CHR$(10)) ausgeführt wird oder nicht. Er sollte auf der Position „OFF" stehen. Falls das nicht hilft, dann versuchen Sie es mit diesem kleinen Programm, das Sie vor dem Ausdruck, aber nach dem Laden und Starten von „Hires-Lores-Print" laden und starten sollten:

```
10 OPEN 4,4
20 PRINT#4,CHR$(27)
   +"A"+(Zahl)
30 PRINT#4,CHR$(27)
   +"2":
40 PRINT#4:CLOSE4
```

Der Wert von „Zahl" in Zeile 20 ist auf „12" eingestellt, hier müssen Sie kleinere Werte probieren, bis der Ausdruck stimmt. Die Stellung der DIP-Schalter des Panasonic 1091 hat beim exakten Ausdruck der Grafik folgende Stellung (0=off, 1=on):

```
DIP 1 2 3 4 5 6 7 8
    1 1 0 0 1 0 1 1
```

Vergleichen Sie bitte die entsprechenden Passagen mit Ihrem Druckerhandbuch.

UNMASKIERTE DATEI

Ich habe eine Frage zum Programm "UNI Datei II" (COMMODORE DISC 14). Hier ist ein Erstellen der Maske nicht möglich, es ging nicht über das erste Datenfeld hinaus. Ich habe versucht, das Programm nach allen Möglichkeiten zu laden.
Eckhard Böttcher,
Flensburg

Eine Maske zu erstellen, ist nicht so schwierig. Dies geschieht nach Wahl der Taste „E" aus dem ersten Menü.
Nehmen wir an, Sie wollen sich eine Adreßdatei anlegen:
Frage des Computers:
1. Datenfeld,
Ihre Eingabe: zum Beispiel Name,20 und die RETURN-Taste drücken usw. Damit geben Sie dem ersten Datenfeld die Bezeichnung „Name", grenzen das mit einem Komma ab (bitte unbedingt eingeben) und tippen dann die maximale Länge dieses Datenfeldes ein, in diesem Beispiel eben „20". Dann erscheint die Frage nach dem nächsten Datenfeld, bei der Sie genauso verfahren.

UNGEDULDIG

Ich habe Probleme mit dem Programm „Vokabel-Trainer" aus COMMODORE DISC 11. Als erstes habe ich mir eine Arbeitsdisk angelegt. Dabei läuft diese Disk beziehungsweise die Floppy beim Anlegen der Daten immer durch. Ich muß abschalten. Nun lege ich meine Original-COMMODORE DISC ein und starte das Programm. Nach Anzeige der Bedienungsanweisung tausche ich die DISC gegen das Duplikat. Danach kann ich das erste Wort anlegen. Als zweites kann ich keine 417 eingegebenen Wörter abfragen. Bei „360" steigt das Laufwerk aus und die grüne Lampe blinkt. Alle meine 417 eingegebenen Wörter sind aber auf der Arbeitsdisk. Ich kann also ab Wort 361 nicht mehr abfragen. Weitere Wörter kann ich aber noch eingeben, und die werden auch auf Diskette geschrieben. Ich muß noch erwähnen, daß genügend Platz auf der Arbeitsdiskette vorhanden ist.
Detlef Timmer
Gock

Sie haben leider nicht genug Geduld gehabt, zu warten, bis das Programm gleich zu Beginn die vorbereitetenden Files (REL und SEQ) auf die Arbeitsdiskette geschrieben hat, sondern – wie Sie selbst schildern – den Computer ausgeschaltet. Bitte unterbrechen Sie die Floppy nicht in ihrer Arbeit, es dauert etwa fünf Minuten, dann meldet sich das Programm mit dem Arbeitsmenü ganz automatisch wieder. Ihre Floppy läuft also nicht durch, wie Sie meinen, sie arbeitet ganz bewußt. Ein korrektes DIRECTORY nach dem Anlegen der Files muß so aussehen:

```
46  Vokabel-Prg.128
    PRG
1   Voc.Stat.        USR
1   Voc.Menge        USR
1   Voc.Daten 1      SEQ
1   Voc.Daten 2      SEQ
406 Voc.Daten 3      REL
```

ZU WENIG DEUTSCHE PROGRAMME

Zum Leserbrief von Rolf-Dieter Vick, „Commodore versteht nur Englisch" (COMMODORE DISC 12), möchte ich auch ein paar Anmerkungen machen.
Es wäre schön, wenn es mehr Software in deutscher Sprache gäbe. Aber auch bei Programmen, wo es in Deutsch leicht ginge, wird einem leider auch alles in Englisch um die Ohren gehauen. Und das wird sich wohl nur ändern lassen, wenn keiner mehr diese Software kaufen würde. Aber um seinen Computer weiter zu betreiben, ist man, wenn man nicht selber programmiert, weiterhin darauf angewiesen. Und nun zum Problem des Herrn Vick mit der unverständlichen Kommunikation mit dem Computer, die für mich noch schwieriger war, denn ich kann überhaupt kein Englisch. Zum Zitat von der Firma Commodore ist zu bemerken, daß sie anscheinend gar nicht weiß, was sie ihren Kunden verkauft. Sonst hätte sie Herrn Vick darüber aufklären können, daß seine Computeranlage in dieser Zusammenstellung in vier Sprachen (Englisch, Französisch, Deutsch und Italienisch) kommunizieren kann, eben mit der Test/Demo-Disk und DOS-Shell. Mit diesem Programm läßt sich auch ohne Probleme die COMMODORE DISC duplizieren. Vorteil: Das Original wird geschont. Zu Ihrer Zeitschrift mit Diskette

ist nur zu sagen, macht weiter so, ich besitze alle Ausgaben seit Nummer 3, als ich sie das erstemal entdeckte. Man kann sie getrost weiter empfehlen. Dieser Brief wurde übrigens mit dem Textverarbeitungsprogramm „Scriptcall" aus COMMODORE DISC 12 im Drei- bis Vier-Finger-System geschrieben. Da mein Monitor (1702) nicht achtzig Zeichen bringt, wird der Fernseher mit Scart-Eingang für die Bildschirmeingabe „mißbraucht". Ausgedruckt wird das Ganze gleich mit einem Seikosha SP-180VC. Zu diesem Programm ist noch zu bemerken, daß das Programm sehr gut und leicht zu bedienen ist.

Manfred Hinze
Bergheim

COMPUTER SPRECHEN ENGLISCH

Dem Leserbrief des Herrn Vick kann ich nur beipflichten. Speziell Commodore hat das bestimmte Vorstellungen, was, sprachlich gesehen, anderen Völkern zusteht. Es gibt keinen logischen Grund, warum der Computer unbedingt auf eine ganz bestimmte, vom Hersteller definierte und verwendete Sprache abgestimmt ist, da er sowieso nur Zahlen versteht. Und es ist ja nur dem menschlichen Unvermögen anzulasten (mit Zahlen und mathematischen Symbolen umzugehen), daß es sogenannte „höhere" Computersprachen gibt. Denn je menschlicher die Befehlssyntax, desto langsamer wird der Computer. Kein Wunder, daß bei Kindern und Jugendlichen der Computer so hoch im Kurs steht, da diese in der Regel auf sprachliche Unterschiede noch nicht so festgelegt sind. Außerdem steht die spielerische Komponente weit im Vordergrund, während im Business-Bereich eher profitorientierte Interessen den Ausschlag geben, ohne Rücksicht auf den einzelnen Benutzer. Daß Englisch die Computersprache ist, liegt klar auf der Hand. Wäre Deutschland das Welt-Computerzentrum, so wäre Deutsch die Computersprache. Also hängt es im Grund genommen von den sprachlichen Fähigkeiten des Anwenders ab, ob es zu Anpassungsschwierigkeiten kommt? In erster Linie liegt's doch wohl an den deutschen Programmherstellern, Softwarehäusern und Distributoren, ob der Kunde auch ein deutschsprachiges Programm bekommt. Es ist ja nicht nur Bequemlichkeit, die für die „Muttersprache" spricht, sondern es sind auch praktische Überlegungen, um Fehler zu vermeiden. Zum Beispiel: „Header" wird mit „Validate" verwechselt oder „Scratch" mit „Load". Man lacht darüber! Aber seien wir ehrlich: Jeder hat mal angefangen. Fazit: Der Computer ist nur so gut wie sein Hersteller, ein Programm so intelligent wie sein Autor . . . und der Anwender ist das letzte Glied dieser Kette.

Winfried Jäger
Schweinfurt

Beide Leser haben recht und doch auch wieder nicht. Wir sollten uns damit abfinden, daß Geschichtsschreibung des Computers in USA zwar nicht begonnen, sich aber doch dort zu der derzeitigen „Hochblüte" entwickelt hat.
Und diese Software-Entwickler denken natürlich nicht im Traum daran, in Deutsch zu programmieren. Gerade deshalb möchten wir doch eine Lanze für deutsche Software-Häuser brechen, die ausgezeichnete Anwenderprogramme wie Textverarbeitung, Dateiverwaltung usw. ausschließlich mit deutschen, für jeden verständlichen Anweisungen anbieten.

DER BHP-VIRUS

Dreht euch nicht um, der Virus geht um

BHP ist kein medizinischer Fachausdruck, sondern lediglich die Abkürzung für „Bayerische Hackerpost", eine Gruppe von Computer-Freaks aus Deutschlands heimlicher Hauptstadt.

Fast täglich erreichen uns neue Schreckensmeldungen über Viren, Bakterien, Bandwürmer und „Trojanische Pferde" in Computersystemen. Mancher wird die Achseln zucken und fragen: „Was habe ich denn mit meinem kleinen C64 damit zu tun?"
Als der Verfasser dieses Berichtes eines schönen Abends neue Public-Domain-Disketten in seine Diskettensammlung aufnahm und an nichts Böses dachte, erschrak er fürchterlich über eine Meldung auf dem Bildschirm seines Monitors:
„Hallo, Dickerchen, dies ist ein echter Virus!"
Ein VIRUS in seinem C64? Nun, das konnte doch nur ein schlechter Witz sein. Ein sofort durchgeführter LIST-Befehl bestätigte die Vermutung, denn statt dem BASIC-Programm stand da nur:

```
1986 SYS PEEK(43)+
     PEEK(44)*256+
     48:VIRUS
```

Tja, was tun (sprach Zeus), RESET durchführen, Kaffee kochen. Maschinensprache-MONITOR laden und starten und suchen. Nach einer (oder zwei) Kannen Kaffee fand er den Virus im schwer zugänglichen Speicherbereich ab $D000 (53248).
Was hatte der Virus angestellt?
Ein Blick ins Diskettenin-haltsverzeichnis zeigt eigentlich keine nennenswerte Veränderung, doch bei genauer Betrachtung fällt einem die Differenz zwischen belegten und freien Blöcken auf.
Das sah dann so aus:

```
"virus          " vv   0
"promon 64"      prg  33
"virus"          prg   9
"test-prog"      prg   1
blocks free.          613
```

Es wurden also nur 613 statt 621 Blöcke als frei angegeben. Damit belegt der Virus wahrscheinlich acht Blöcke auf der Diskette.
Also, Disk-MONITOR laden, starten und Spur 18, Sektor 01 untersuchen (zweiter Block des DIRECTORY, Filenamen). Folgendes sah er auf dem Bildschirm:

```
:CFA0 00 00 82 14 0F
      54 45 53
:CFA8 54 2D 50 52 4F
      47 A0 A0
:CFB0 A0 A0 A0 A0 A0
      00 00 00
:CFB8 00 00 00 00 00
      00 01 00
```

Der Fileeintrag im DIRECTORY beginnt mit dem dritten Byte, das ist der Filetyp (82=PRG). In den nächsten beiden Bytes ist die Spur und der erste Block des Files angegeben (Spur $14, Block $0F). Die folgenden 16 Bytes enthalten den Filenamen (in diesem Fall: TEST-PROG). Die Länge des Files wird in Byte 30 und 31 festgehalten, hier handelt es sich um einen Block.
Also auch hier keine Veränderung.
Das große Erwachen kam beim Listen der Spur 20 (=$14), Sektor 15 (=$0F). Anstelle des erwarteten Files 'TEST-PROG' stand hier der Virus!

```
:CF00 15 07 01 08 1F
      08 C2 07
:CF08 9E C2 28 34 33
      29 AA C2
:CF10 28 34 34 29 AC
      32 35 36
:CF18 AA 34 38 3A 56
      49 52 55
```

Die ersten beiden Bytes geben die Adresse des folgenden Blocks an (Spur $15, Sektor $07). Der Wert 01 08 steht für die Adresse des BASIC-Anfangs (2049) und ab dem 5. Byte beginnt der Virus mit dieser bereits bekannten, ominösen BASIC-Zeile:
1986 SYS PEEK(43)+usw.
Der Virus belegt also die Blöcke $14, 0F und $15, 07 – $15, 01.
Auf der Spur 21, Sektor 0 steht nun das gesuchte File:

```
:CF00 00 12 01 08 0E
      08 0A 00
:CF08 99 22 48 41 4C
      4C 4F 22
:CF10 00 00 00 BD C6
      A3 D0 F7
```

Das zweite Byte gibt die Länge des Files auf diesem Sektor an, also 12.
Im Klartext ein überaus geistreiches BASIC-Programm:

```
10 PRINT"Hallo".
```

Bevor wir aber zur Entfernung des Virus kommen, wollen wir uns mit dem Aufbau und der Arbeitsweise vertraut machen.

WAS SIND VIREN?

Biologisch gesehen ändert ein Virus das Erbgut einer gesunden Zelle so ab, daß sie selbst Viren produziert (Reproduktion). Dadurch soll ihr Erhalt und die schnelle Verbreitung gewährleistet sein.
Computer-Viren verhalten sich ganz genauso. Sie bestehen aus einem sehr kurzen Programm (etwa 200 bis 300 Befehle), das sich in zwei Teile splittet. Der erste Teil hat den Auftrag, sich zu vermehren, sich also in oder an (noch) nicht infizierten Programmen zu kopieren, damit wird der Erhalt gesichert.
Der zweite Teil sorgt für die Manipulation der Programme.
Dies kann sowohl von gutartiger als auch von bösartiger Natur sein. Die Vernichtung aller Datenbestände, die jeden User in die Verzweiflung treiben kann, ist ebenso möglich wie die gern gesehene Komprimierung der Programme zur Verringerung des Speicherplatzes.

UNTERSCHIEDLICHE VIREN

In der Praxis gibt's eine Vielzahl von Viren. Man kann sie grob nach Art des „Befalls" in „überschreibende" und „nichtüberschreibende" einteilen.

Überschreibende Viren

Sie löschen einen Teil des nicht infizierten Programmes und kopieren sich an diese Stelle, dadurch fällt dieser Virus nicht durch erweiterten Speicherbedarf auf.
Allerdings ist ein so infiziertes Programm nicht mehr lauffähig (die überschriebenen Daten fehlen) und dient nur zur erneuten Initialisierung des Virus.

Nichtüberschreibende Viren

Im Gegensatz zu der vorher beschriebenen „Bakterie" setzt diese Art des Virus vor oder hinter ein Programm. Die infizierte Version wird wieder abgespeichert, gleichzeitig wird das Diskettenin-haltsverzeichnis bereinigt, so daß die Daten und die Programmlänge wieder mit der Original-Version übereinstimmen.
Diese Art zerstört die befallenen Programme nicht und ist nur an den verlängerten Lade- und Speicherzeiten zu erkennen.
Und genau in diese – im Prinzip harmlose – Grup-

pe gehört der BHP-Virus. Er initialisiert sich im RAM-Bereich unter dem Ein-/Ausgabe-ROM ab $D000 (53248). Da er relativ schwierig zu handhaben ist, wird er von den meisten Programmierern gemieden. Selbstverständlich besitzt der Virus auch eine CBM80-Kennung, die ihn RESET-beständig macht. Nur durch das Ausschalten des C64 ist eine wirksame Vertreibung gewährleistet.
Findet er ein noch nicht infiziertes Programm, verschiebt er es und kopiert sich davor.
Beim Listen wird dann eben nur die erwähnte Zeile mit dem SYS-Befehl angezeigt. Mit dem start durch RUN rufen Sie praktisch den Virus erneut auf, er initialisiert sich neu und startet das Programm. Speichern Sie so ein „verseuchtes" Programm, speichern Sie auch den Virus mit ab.
Wie schon erwähnt, wird die Blockzahl dabei so manipuliert, daß die Originalblockzahl (also die vor dem Befall) angegeben wird. Der Virus fällt dadurch niemandem auf. Erst wenn sich einer die Mühe macht und die belegten und freien Blöcke vergleicht, wird er sehen, daß insgesamt acht Blökke mehr belegt sind. Die Tarnung ist fast perfekt, wenn da nicht noch die verlängerte Ladezeit wäre. Gerade bei kurzen Programmen fällt das sehr auf.
Einen weiteren Gag hat der BHP-Virus noch parat: Er kopiert sich schon beim Laden eines Programmes auf Diskette. Bereits bei diesem Vorgang schreibt er sich vor das Programm und speichert sich selbst wieder ab, die Diskette ist damit infiziert.
Da können Sie nur von Glück sprechen, daß der BHP-Virus nicht bösartig ist, denn dann wäre es bereits zu spät. So aber ist er eigentlich nur lästig und frißt halt Speicherplatz auf Ihrer Diskette. Das sollte den Entwicklern dieser „Spielerei" doch lobend angerechnet werden, denn wir sind überzeugt, daß sie auch anders gekonnt hätten, wenn . . .

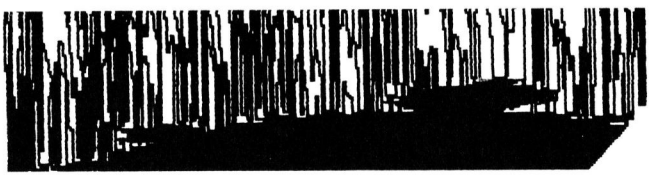
SO WERDEN SIE IHN WIEDER LOS

Zum Abschluß dieses „Krankenhaus-Reports" noch eine simple Methode, den Virus wieder loszuwerden. Dazu benötigen Sie einen Disketten-MONITOR. Suchen Sie die Spur und den Sektor des eigentlichen Programms, bei unserem Beispiel war es Spur 21, Sektor 0. Dieser Wert wird im DIRECTORY im 4. und 5. Byte übernommen und dann auf Diskette zurückgeschrieben:

```
:CFA0 00 00 82 15 00
      54 45 53
:CFA8 54 2d 50 52 4F
      47 A0 A0
:CFB0 A0 A0 A0 A0 A0
      00 00 00
:CFB8 00 00 00 00 00
      00 01 00
(= TEST-PROG)
```

Damit haben Sie den Virus abgehängt, das Programm kann wieder normal benutzt werden.
Ein Tip, den Sie beherzigen sollten:
Neue Disketten eine gewisse Zeit unter „Quarantäne" halten, bis Sie sicher sein können, daß sie virenfrei sind.
Merke: Der nächste Virus kann tödlich für Datensammlungen auf Diskette sein.

Gerhard Szymanski/bu □



IMPRESSUM

COMMODORE DISC

C-DISC erscheint monatlich in der CA-Verlags GmbH, einer Gesellschaft in der Aktuell-Gruppe, Heßstr. 90, 8000 München 40.
Tel.: 089/1298011
Telex: 5214428 cav-d

VERANTWORTLICH FÜR DEN INHALT:
Harald Beiler

REDAKTION UND STÄNDIGE MITARBEITER:
Peter Basch, Harald Beiler, Renate Huber, Lothar Miedel, Alfons Mittelmeier, Michael Reppisch, Rudolf Schmid-Fabian, Torsten Seibt, Hermann Wellesen, Bernd Welte

GESCHÄFTSFÜHRER:
Werner E. Seibt

ANZEIGENVERWALTUNG:
ADV-Mediendienste, Postfach 101124, 8900 Augsburg 1
Telefon: 0821/7904-227
Telex: 533502 adv
Teletex: 821887
Telefax: 0821/7904-243

Es gilt Preisliste Nr. 9 vom 1.4.1988.
Media-Unterlagen bitte anfordern.

VERANTWORTLICH FÜR DEN ANZEIGEN-INHALT:
Brigitte Kostic

ANSCHRIFT FÜR ALLE VERANTWORTLICHEN:
Postfach 1107, 8044 Unterschleißheim
Telex: 5214428 cav-d

VERTRIEB:
Verlagsunion Wiesbaden

Printed in Germany by ADV, 8900 Augsburg 1, Aindlingerstraße 17–19

Hinweis gem. §8,3 Bayr. Pressegesetz: Gesellschafter der CAV GmbH sind zu 26 v.H. Henny Rose Seibt, Kauffrau, und zu 74 v.H. Werner E. Seibt, Journalist, beide Elisabethstraße 1, 8044 Unterschleißheim.

So laden Sie Ihre Commodore Disc

Bitte lesen Sie als Computeranfänger diese Ladeanweisungen genau durch, dann gibt's beim Laden der COMMODORE DISC keine Probleme!

Als zweites Programm auf Ihrer DISC finden Sie das Programm *Disclader64/128,* das sowohl im C64 (oder im entsprechenden Modus im C128) als auch mit dem 40- und 80-Zeichen-Bildschirm des 128PC läuft.
Es ist ein Programm in BASIC 2.0, da nur so eine derartige Kompatibilität für alle drei genannten Systeme zu erreichen war.
Folgende Punkte sollten aber noch immer beachtet werden:
1) Schalten Sie Computer und Floppy ein.
2) Legen Sie die COMMODORE DISC in den Laufwerksschacht und verriegeln Sie diesen.
Jetzt richtet sich Ihr weiteres Vorgehen danach, ob Sie C64- oder 128-PC-Programme laden möchten.
Als C64-Benutzer geben Sie bitte ein:

LOAD":*",8,1

Damit lädt der C64 zuerst das Maschinenfile *Boot.64,* das sich an erster Stelle auf Ihrer Diskette befindet, das seinerseits wieder den *Disclader 64/128* nachlädt und auch sofort startet.
Besitzen Sie einen C128 und befinden sich auch in diesem Modus, genügt es normalerweise, wenn Sie kurz den RESET-Taster (seitlich rechts an Ihrem Gerät) betätigen oder den Befehl BOOT eingeben und dann die RETURN-Taste drücken.
Beides erzeugt denselben Effekt, auf Ihrem Bildschirm erscheint die Meldung

BOOTING...DISCLADER 64/128

Auch hier wird dieses Ladeprogramm nach dem „Booten" sofort gestartet.
3) Nachdem das Anfangsbild auf dem Bildschirm erschienen ist, rufen Sie jetzt bitte nach Druck auf die Leertaste das Inhaltsverzeichnis, das *DIRECTORY,* der gerade aktuellen COMMODORE DISC auf.
4) Die Liste der Programme, die sich auf der DISC befinden, werden nun aufgelistet, vor den Programmnamen erscheint ein kleiner Pfeil, den Sie mit den Cursortasten auf und ab bewegen können.
Als C64-Benutzer sollten Sie nur die mit diesem Kennzeichen beginnenden Programmnamen auswählen, wenn Sie sich im 128er-Modus befinden, dann nur solche. Das Laden von Programmen jeweils in den nicht dafür vorgesehenen Computertyp ergibt in den meisten Fällen nur Unsinn.
Bei den C128-Programmen ist zusätzlich noch in Klammern angegeben, ob Sie für den 40- oder 80-Zeichen-Modus gedacht sind, also (40) oder (80).
5) Haben Sie sich für ein entsprechendes Programm für den zutreffenden Computermodus entschieden, positionieren Sie den Pfeil vor dessen Namen und drücken einfach die RETURN-Taste.
6) Das gewünschte Programm wird jetzt automatisch geladen und sofort gestartet.
Wie bereits erwähnt, ist dieser Disclader ein BASIC-Programm, das aber von Ihrem ausgewählten Programm gelöscht wird. Wollen Sie ein anderes Programm auf die gleiche Art, also unter Benutzung des *Disc-laders,* in Ihren Computer holen, so müssen Sie halt den *Disclader* wieder auf die vorher beschriebene Art laden.

LADEN „VON HAND"

A. Im C64-Modus

Für alle, denen dies immer noch zu umständlich ist, gibt es natürlich noch die „normale" Art des Ladens: aus dem DIRECTORY.
a) C64-Benutzer:
Punkt 1) und 2) (einschalten und Disk einlegen) sollten klar sein, so daß es weitergeht mit:
3) Geben Sie bitte folgenden Befehl ein:

LOAD"$",8

und drücken Sie die RETURN-Taste. Durch die Bezeichnung "$" wird das Disketteninhaltsverzeichnis der COMMODORE DISC (oder jeder anderen Diskette) geladen.
4) Nachdem sich der Computer wieder mit READY meldet, tippen Sie bitte LIST ein und drücken wieder die RETURN-Taste. Auf dem Bildschirm wird jetzt das genaue Inhaltsverzeichnis Ihrer COMMODORE DISC aufgelistet.
5) Entscheiden Sie sich für das Programm, das Sie laden möchten.
6) Gehen Sie mit dem Cursor nach oben, bis dieser in der Zeile vor dem gewählten Programmnamen steht (an die Stelle, an der die Zahl der belegten Blocks eingetragen ist).
7) Geben Sie an dieser Stelle den Befehl LOAD ein. Bitte achten Sie darauf, daß das erste „Gänsefüßchen" vor dem Programmnamen nicht überschrieben wird! (Die Zahl, die die vom Programm belegten Blöcke angibt, dürfen Sie ruhig überschreiben.)
8) Drücken Sie jetzt bitte nur die RETURN-Taste, das Laufwerk beginnt nun, Ihr gewünschtes Programm zu laden.
Achtung: Der *Disclader 64/128* besitzt keinen Zusatz wie ',8:' oder nur den Doppelpunkt alleine, da er von allen drei Programm-Modi (C64, C128 im 40- und im 80-Zeichen-Bildschirm) benutzt wird, hier müssen Sie als einzige Ausnahme den Ladezusatz selbst eintippen.
Im C64-Modus:
LOAD"DISCLADER64/128",8:
Im 128er-Modus:
RUN"DISCLADER":
9) Wenn sich nach dem

Ladevorgang, bei dem SEARCHING FOR... und LOADING vom Computer als Arbeitsmeldung auf dem Bildschirm ausgegeben werden, der blinkende Cursor mit READY wieder meldet, so tippen Sie dort ein:
RUN: (den Doppelpunkt nicht vergessen und die RETURN-Taste drücken).
Das geladene Programm wird so gestartet und steht zu Ihrer Verfügung.

Ein kleines Beispiel: Sie möchten das Programm „Eisenbahn" von Diskette laden. Die exakte Ladeanweisung auf dem Bildschirm müßte dann so aussehen:

LOAD"EISENBAHN",8:

B. Im C128-Modus

1) Hier müssen ebenso der Computer und das Laufwerk eingeschaltet, die COMMODORE DISC eingelegt und der Laufwerkschacht verriegelt werden.
2) Drücken Sie jetzt bitte die Funktionstaste 'F3', (oben rechts, über der numerischen Tastatur).

Diese Taste ist schon beim Einschalten des 128PC mit dem Befehl zum Aufruf des Disketteninhaltsverzeichnisses belegt (DIRECTORY) und wird auch sofort ausgeführt.
3) Stoppen können Sie die Ausgabe auf dem Bildschirm jederzeit mit der NO-SCROLL-Taste (oben Mitte), den Ladevorgang des DIRECTORY brechen Sie mit der STOP-Taste ab.
4) Wählen Sie auch hier ein Programm aus und fahren Sie mit Hilfe der entsprechenden Tasten den Cursor vor den gewählten Programmnamen.
5) Tippen Sie hier ein: RUN (auch hier kann die Blockzahl ruhig überschrieben werden).
6) Nach Drücken von <RETURN> läuft die Floppy an, das Programm wird geladen und im Gegensatz zum C64 sofort gestartet, es erübrigt sich also, hier nochmals 'RUN' einzugeben.
Auch hier ein Beispiel, wie eine korrekte Ladeanweisung aus dem DIRECTORY beim C128 aussieht:

RUN"CHECK-LISTE":

NOCH EIN PAAR TIPS

● Als C64-Benutzer können Sie (allerdings immer nur das erste File, das sich auf Ihrer Diskette befindet, und dazu muß es auch unbedingt ein Programm = PRG-File sein) mit Druck auf zwei Tasten in den C64 laden: SHIFT/RUNSTOP. Es startet dann sofort.
● Wenn's nicht das erste File auf der Disk ist, wird die Ladeanweisung etwas umfangreicher: Siehe soeben behandelte Ladeanweisungen!
● Files, die hinter dem Namen nicht die Bezeichnung 'PRG eingetragen haben, etwa SEQ, REL, DEL und USR lassen sich mit diesen besprochenen Ladeanweisungen weder laden noch starten. Es handelt sich hier um reine Datenansammlungen von Bytes, die nur vom dafür gedachten Hauptprogramm (egal, ob in BASIC oder Maschinensprache) wieder in den Computerspeicher geladen und dort verarbeitet werden können.
Lesen Sie dazu bitte die entsprechenden Seiten in Ihrem Handbuch zur Floppy nach.
● Nicht alle PRG-Files sind BASIC-Programme, die mit RUN gestartet werden können. Sehr oft handelt es sich um Maschinensprache-Routinen, die an einen ganz anderen Speicherplatz geladen werden, als ihn ein BASIC-Programm benötigt, zum Start würden sie die Eingabe eines 'SYS'-Befehls brauchen. Oder es handelt sich auch hier um eine reine Byte-Sammlung wie etwa der Inhalt eines Sprite-Speichers oder eines hochauflösenden Grafikbildes. Diese Files werden ebenfalls in der Regel von einem Haupt-Steuerprogramm zur Verarbeitung benötigt und bei Bedarf während des Programmablaufs nachgeladen.
Ein Blick genügt: Alle PRG-Files auf der COMMODORE DISC, die entweder mit dem Zusatz ',8: beziehungsweise ',8,1:' (bei den C128-Programmen nur mit Doppelpunkt) gekennzeichnet wurden, sind lad- und ausführbare PRG-Files, alle anderen nur Daten, die von einem Hauptprogramm nachgeladen werden müssen.
● Die Zahl vor den Filenamen im DIRECTORY ist die Anzahl der belegten Blocks. Möchten Sie (in etwa) wissen, wieviele Bytes so ein Programm im Speicher Ihres Computers belegen würde, so multiplizieren Sie diese Zahl mit ,256'.
● Die Meldung am Ende des DIRECTORY 'XX BLOCKS FREE' gibt Ihnen darüber Auskunft, wieviel Platz (in Blöcken gerechnet) sich noch auf der aktuellen Diskette befindet. Wollen Sie die genaue Byteanzahl wissen, so müssen Sie diese Zahl ebenfalls mit ,256' malnehmen.
● Die COMMODORE DISC ist (normalerweise) dadurch schreibgeschützt, daß sich auf der rechten oberen Seite neben dem Etikett keine Schreibkerbe befindet. Das hat den Vorteil, daß Sie diese Diskette nie versehentlich löschen oder neu formatieren können, aber auch den Nachteil, daß keine Daten mehr darauf gespeichert werden können. (Von DISC lesen oder laden geht immer.)
Viele Programme der verschiedensten Ausgaben der COMMODORE DISC verlangen aber, daß Daten abgelegt werden können, denken Sie nur an Anwender-, Dateiverwaltungs- und Textprogramme. Ja, auch verschiedene Spiele brauchen einen Schreibzugriff auf die Diskette, und sei es nur, die neuen, aktuellen High-Score nach Beendigung des Spiels auf Diskette zurückzuschreiben.
Da gibt's nur eins: das entsprechende Programm auf eine andere Diskette *mit* Schreibkerbe zu speichern beziehungsweise diese Files einzeln mit einem entsprechenden Kopierprogramm rüber zu kopieren oder sich gleich mit einem Kopier-Tool für ganze Disketten ein „Backup" Ihrer COMMODORE DISC auf eine schreibfähige Disk zu ziehen. Oder Sie benutzen einen Diskettenlocher und bringen damit die entsprechende Schreibkerbe an. Wir empfehlen auf alle Fälle die Kopiermethode.

KOPIERSCHUTZ – NEIN, DANKE!

Wir haben es bisher nicht getan und werden es auch künftig nicht tun: einen Kopierschutz auf der COMMODORE DISC installieren. Nicht, weil wir es nicht könnten, sondern weil wir keinen Sinn darin sehen.
Und schließlich ist die Tatsache, daß sich Raubkopierer strafbar machen, nicht erst seit heute bekannt.
Jeder soll die Möglichkeit haben, sich so viele Sicherheitskopien seiner Disketten zu eigenen, privaten Zwecken zu machen, wie er für richtig hält.
Und nun viel Spaß mit der neuen COMMODORE DISC! □

CP/M PLUS

Das dritte Betriebssystem

Mit großen Vorschußlorbeeren ausgestattet, macht doch kaum ein C128-Benutzer Gebrauch davon: CP/M, gesteuert vom Mikroprozessor Z 80.

Stellen wir zunächst die Kardinalsfrage: Wozu dient überhaupt ein Betriebssystem? Schließlich und endlich braucht einer den C128 nur einzuschalten, und schon meldet sich der Cursor mit READY und wartet auf die Eingaben. Und genau das ist der Kernpunkt. Ein Computer, egal, wie er heißen mag, besteht nicht nur aus Tastatur und Zentraleinheit (CPU), sondern aus mehreren Prozessoren wie etwa den Ein-/Ausgabe-Bausteinen, Videochips, Joystickanschluß, Diskettenstation, Userport usw.
Ein Betriebssystem kümmert sich nun darum, daß der Anwender oder Programmierer recht einfach auf diese genannte „Umgebung" (Peripherie) des Computers zugreifen kann. In der Regel ist es bei Homecomputern (wie dem C128) so, daß jeder Rechner sein eigenes, meist zum Betrieb mit BASIC zugeschnittenes Betriebssystem mitbringt, das bereits fest eingebaut ist. (Drum ist es nach dem Einschalten auch präsent.)
Allerdings gibt's schon bei den diversen Homecomputer-Typen ein und derselben Hersteller-Firma erhebliche Unterschiede bei den BASIC-Dialekten, noch schlimmer wird es, wenn Sie beispielsweise ein auf dem C64 geschriebenes Maschinenprogramm im C128 betreiben möchten. Nicht, daß die Assemblercodes anders wären, nein, es sind genau dieselben – aber die Adressen der System-Routinen stimmen bis auf ganz wenige Ausnahmen nicht überein. Fazit: Ohne großmächtige Änderungen läßt sich kaum ein BASIC-Programm von einem C128 auf den C64 übertragen, und ein Assemblerprogramm auch nur dann, wenn Sie, grob gesagt, jede zweite Adresse ihrer Bestimmung entsprechend anpassen.

UNIVERSELL SOLLTE ES SEIN

CP/M dagegen wurde zu dem Zweck entwickelt, eine gemeinsame Programmierumgebung für die unterschiedlichsten Maschinen nicht nur desselben, sondern sogar verschiedener Hersteller zu ermöglichen. Daß dieser Versuch erfolgreich war, beweist die Tatsache, daß etwa 250 unterschiedliche Computertypen mit CP/M arbeiten können. Eine Standardisierung also.
Einige Grundkenntnisse dürfen wir voraussetzen: wie man erfährt, welche Dateien sich auf der Diskette befinden (DIR), wie kopiert wird (PIP) oder wie Dateien umbenannt werden. Auch

SUBMIT – DER STAPLER

wenn das Handbuch zur CP/M-Version 3.0 (wird beim Kauf eines C128 gratis mitgeliefert) für manche als „Witzblatt" empfunden wird, das alles steht da doch drin.
Drum möchten wir uns mit diesen allgemeinen Dingen auch nicht aufhalten, sondern Ihnen vielmehr Tips geben, wie Sie mit CP/M effektiv arbeiten können.
Zum Beispiel existiert da ein Dienstprogramm namens SUBMIT, das Stapelverarbeitung unter CP/M zuläßt. Wenn Sie sich aber unter Computer-„Kollegen" umhören, erfreut sich dieses Programm keineswegs der Beliebtheit, die seinem Leistungsumfang angemessen wäre.
Wie Sie wissen, werden CP/M-Befehle in der Kommando-Zeile (Command Line) editiert, also eingegeben und durch Druck auf die RETURN-Taste (sie wird ENTER genannt) zur Ausführung gebracht. Erst wenn so eine Ausführung zuende ist, können Sie den nächsten Befehl eintippen.
SUBMIT bietet nun die Möglichkeit, Kommandos hübsch der Reihe nach aus einer Textdatei zu lesen und abzuarbeiten.
Wenn's auch noch so schön klingt, beim ersten Ausprobieren wird's zum echten Ärgernis: Die Möglichkeit, eine Textdatei zu erstellen, ist sehr schlecht beschrieben. Es wird auf den Editor ED verwiesen. Die Arbeit mit diesem Programm verlangt in etwa die Sentimentalität eines Museumsdirektors, der seine Paßbilder mit minutenlangem Stillsitzen und Abbrennen von Magnesium erstellen will (statt ein Blitzlicht zu benutzen).
Auch für aufgeschlossene Computer-Freaks ist ED ein echtes Rätselprogramm, auf das wir hier nicht näher eingehen wollen; vorzuziehen ist die Erstellung von Texten mit PIP. Natürlich ist auch die Arbeit damit umständlich, aber wesentlich leichter zu erlernen als ED. Außerdem nehmen wir doch an, daß jeder, der ernsthaft mit CP/M arbeiten will, sich über kurz oder lang einen Editor oder ein Textverarbeitungsprogramm anschafft, um etwas umfangreichere Texte zu erstellen.
PIP ist eigentlich ein Dienstprogramm zum Kopieren von Dateien unter CP/M.
Neben den tatsächlichen Dateien auf Diskette kennt CP/M aber auch sogenannte „logische" Dateien wie Tastatur, Drukker oder die V24-Schnittstelle.
Manche dieser „Dateien" eignen sich nur zum Lesen, wie's beispielsweise bei der Tastatur der Fall ist.
Demnach ist unser Wunsch,

PIP ALS TEXTEDITOR

per Tastendruck eine Textdatei auf Diskette zu erzeugen, nichts anderes als ein Kopieren von der Tastatur auf eine Diskettendatei mit einem von uns gewählten Dateinamen.
Rufen wir also PIP auf: A> PIP, das Programm meldet sich mit dem bekannten Sternchen „*" und wartet auf eine Anweisung. Nehmen wir mal an, unser erster Versuch soll TEST.TXT heißen, die logische Datei, die wir lesen wollen, ist die Tastatur, im CP/M-Sprachgebrauch „CON" genannt (von „Console"). Die Anweisung an PIP sieht dann so aus:

```
*A:TEXT.TXT=CON:

(RETURN/ENTER-
Taste drücken)
```

(Den Stern am Zeilenanfang nicht nochmals eintippen, das stammt vom Aufruf von PIP!)
Der Cursor rückt in die nächste Zeile. Nun könnten wir eintippen:

```
"DIES IST EIN
 TESTTEXT."
```

Drücken wir jetzt ENTER, wandert der Cursor zum Zeilenbeginn. So war das aber nicht gedacht, schließlich wollten wir ja mehrere Zeilen erstellen.
Das Rätsel ist jedoch schnell gelöst: RETURN oder ENTER hat den Tastencode 13 (wie in BASIC); physikalisch bewirkt unter CP/M dieses Zeichen nur einen Wagenrücklauf (CR = Carriage

Return), nicht aber noch zusätzlich einen Zeilenvorschub (LF = Line Feed). (In BASIC ist auch der dabei.)
Diesen „Line Feed", Code 10, können wir durch CTRL-J erzwingen; nun steht der Cursor da, wo wir ihn haben möchten: Am Anfang der nächsten Zeile.
Weitere wichtige Steuerzeichen mit Hilfe der CONTROL-Taste finden Sie in einer Tabelle zu diesem Artikel, vor allem, wenn sich einer vertippt hat, muß er ja mit dem Cursor auch innerhalb der Zeilen wandern können (wie beim komfortablen Bildschirmeditor des C128 bei BASIC-Betrieb).
Nachdem wir so einen kleinen Text eingegeben haben, sollte uns PIP den natürlich auch auf Diskette schreiben. Seien Sie bitte aber nicht zu übereifrig: Wenn Sie mit CTRL-C den Editiermodus verlassen haben, werden Sie vergeblich nach der Datei TEXT.TXT suchen. PIP glaubt nun, Sie hätten einen Fehler gemacht und das Programm dadurch mit CTRL-C abgebrochen. Dann einen (falsch) editierten Text abzuspeichern, ist CP/M doch zu dumm.

ZUM ABSCHLUSS CTRL-Z DRÜCKEN

Da die Console (CON:) als Textdatei angesprochen wird, sucht PIP stets nach dem Zeichen EOF (End of File = File-Ende), das unter CP/M den Code 26 besitzt. Erst wenn dieser Code erkannt wird, schließt CP/M die Datei CON: und Zeichen für Zeichen einschließlich des EOF wird als neue Datei TEXT.TXT auf die Diskette geschrieben. Dieses EOF erreichen Sie mit CTRL-Z.
Also, daran denken: Einen mit PIP erstellten Text beliebiger Länge nur mit CTRL-Z abschließen und PIP verlassen. Falls Sie nicht glauben, daß es geklappt hat, tippen Sie zur Kontrolle ein

TYPE TEXT.TXT
(ENTER-Taste)

Sie können erkennen, daß PIP auch die CTRL-Codes mit in die Datei übernimmt; TYPE ist so etwas wie ein LIST-Befehl; bei der Bildschirmausgabe werden diese Codes jedoch wieder neu interpretiert, so daß Sie den Text so sehen, wie Sie ihn haben wollten.

RAN AN CP/M!

Mit diesen ersten „fundierten" Kenntnissen möchten wir uns nun endlich daran machen, die erste SUBMIT-Befehlsdatei zu erstellen. Um bei Aufruf von SUBMIT nicht stets diese sechs Buchstaben eintippen zu müssen – wer arbeitet schon gerne unnötig – legen wir uns eine Kopie vom File SUBMIT.COM namens DO.COM an. Jetzt stellt sich der Aufruf als „DO" dar, das sind immerhin vier Buchstaben weniger, bei 1000 Aufrufen sind das 4000 Tastendrucke weniger, bei 10000 dann . . . (Scherz beiseite).

Das Kopieren erledigt wieder:
PIP (ENTER)
*DO.COM=SUBMIT.
COM (ENTER)
*
Schreiben wir uns doch mal eine Befehlsdatei, die den freien Diskettenplatz und die auf Disk vorhandenen Dateien ausgibt:
*INFO.SUB=CON:
(ENTER)
(Achtung: Niemals den Filenamen-Zusatz „SUB" vergessen, sonst kommt SUBMIT/DO nicht damit zurecht!)
Der Eingabetext für die zu bastelnde Datei sieht dann so aus:

SHOW.A:
DIR.A:

Nach CTRL-Z können wir PIP verlassen und mit DO INFO (den Zusatz „SUB" brauchen Sie jetzt wiederum nicht mehr anzugeben, denn nun liegt er ja bereits vor und wird automatisch von CP/M angehängt) wird unser erstes Programm, ein sogenanntes „BATCH"-Programm ausgeführt. Sollte CP/M Fehlermeldungen ausgeben, so ist ein solcher leicht zu finden: Bevor SUBMIT einen Befehl aus der Datei ausführt, wird die entsprechende Zeile auf dem Bildschirm ausgegeben, Tippfehler lassen sich dadurch leicht erkennen. Maschinenintern wird bei der Abarbeitung dieser Befehlsdatei eine Zwischendatei mit dem Zusatz .$$$ erzeugt; erst aus dieser werden die eigentlichen Befehle ausgelesen und ausgeführt.
Warum so umständlich? Das liegt an einem weiteren Vorteil von SUBMIT. Es besteht die Möglich-

BATCH-DATEIEN

keit, automatisch Platzhalter zu ersetzen. Platzhalter sind dazu da, Wörter, die beim Aufruf von SUBMIT/DO in der Kommandozeile hinter dem Namen der Befehlsdatei stehen, in dieselbe einzufügen – und dazu braucht CP/M eben die Zwischendatei, um die „echte" Batch-Datei nicht zu zerstören. Stellen, an denen Ersetzungen vorgenommen werden sollen, werden mit dem Dollarzeichen und einer Zahl angegeben: Die Zahl bedeutet hierbei die Nummer, an deren Stelle in der Parameterliste das betreffende Argument erwartet wird. Klingt kompliziert, stimmt's? Ist es aber gar nicht.
Nehmen wir einmal an, wir haben DO mit folgender Zeile aufgerufen:

DO.INFO*.COM*.DTA

DO ruft SUBMIT auf. (Wir haben es ja unter diesem neuen Namen kopiert.) INFO ist die Datei, in der die Befehle stehen, *.COM ist Argument 1, *.DTA ist Argument 2. Sollte in der Datei INFO.SUB nun irgendwo "$1" stehen, ersetzt DO *jede* Stelle, an der $1 steht, durch das vorhandene Argument "*.COM". Entsprechend werden alle $2-Platzhalter durch "*.DTA" ersetzt. Ist kein Argument angegeben, werden die Platzhalter

WICHTIGE EDITIER- UND STEUERZEICHEN BEI CP/M

Tastenbefehl	Funktion
CTRL-A	Cursor ein Zeichen nach links
CTRL-C	Programmabbruch
CTRL-E	Zeilenvorschub ohne ENTER-Taste (Line Feed)
CTRL-F	Cursor ein Zeichen nach rechts
CTRL-G	Löscht das Zeichen, auf dem der Cursor steht
CTRL-H	Löscht das Zeichen links neben dem Cursor
CTRL-M	Enter-Funktion
CTRL-W	Gesamte letzte Zeile wird wiederholt
CTRL-Z	EOF, Dateiendekennzeichen (End of File

durch Leerzeichen ersetzt. Das nämliche gilt für den Fall, daß weniger Argumente angegeben werden, als Platzhalter in der Befehlsdatei angefordert wurden: auch hier werden einfach Leerzeichen eingesetzt. Um die Funktion auszutesten, schreiben wir uns am besten eine neue „Batch"-Datei namens INFO2.SUB:

```
SHOW.A:
DIR.A:$1
TYPE $2
```

Wie sieht es nun aus, wenn sie abgearbeitet wird?
Aufruf:
DO INFO2*.TXT
INFO2.SUB
Zwischendatei:
SHOW.A:
DIR.A:*.TXT
TYPE INFO2.SUB
Resultat:
Diskettenplatz auf A: wird angezeigt, alle Dateien mit Zusatz .TXT werden angezeigt und der Inhalt von INFO2.SUB auf dem Bildschirm ausgedruckt. Der Aufruf DO INFO2 verhält sich dagegen genau wie das Programm DO INFO: Da DIR ohne Parameter bleibt, werden alle Dateien angezeigt, und TYPE ohne Argument macht überhaupt nichts. Die mögliche Anzahl der Argumente/Platzhalter ist mit insgesamt 10 übrigens völlig ausreichend, bedenken Sie dabei jedoch, daß der Computer bei „Null" zu zählen beginnt: S0 bis S9.

CP/M UND PASSWORT

Oft wurde die Frage gestellt, welchen praktischen Sinn die von CP/M angebotenen Möglichkeiten der Definition von Paßwörtern, geschützten Dateien, Benutzerbereichen usw. haben sollen. Ganz ehrlich: Beim 128 PC überhaupt keinen! Alle diese Utilities wurden entwickelt, als CP/M als ein für die damalige Zeit sehr leistungsfähiges Betriebssystem überwiegend im geschäftlichen Bereich eingesetzt wurde. Ein einziger Computer stand da für viele Benutzer zum Beispiel in einer Firma zur Verfügung, die Massenspeicher waren Winchester-Laufwerke oder Festplatten mit 20 und mehr Megabyte Kapazität. Um diese große Datenmenge zu verwalten, war es erforderlich, den zur Verfügung stehenden Platz für mehrere Benutzer aufzuteilen, damit jeder so arbeiten konnte, als habe er seine eigene Festplatte mit vielleicht nur einem oder zwei Megabyte. Da sollte natürlich der Kollege aus einer anderen Abteilung oder eines fremden Ressorts tunlichst seine Nase aus den Dateien lassen.
Für den Privatbereich zu Hause und den Betrieb einer oder zwei Diskettenstationen brauchen keine Paßwörter oder Benutzerbereiche festgelegt werden; schneller ist eine Diskette gewechselt und unter Verschluß gebracht als ein Paßwort gesetzt.

RMAC UND LINK

CP/M Plus bietet auch Programmierwerkzeug an: Da gibt es den Macro-Assembler RMAC und den dazugehörigen Linker LINK. Wer als Programmierer mit Compilersprachen arbeitet (Pascal, C) wird feststellen, daß auch einem Compiler nicht immer alles möglich ist. Manch einer möchte sich eine Assembler-Routine auch selbst schreiben. Dann freut sich der um so mehr, ein entsprechendes Werkzeug zur Hand zu haben, das dazu noch so weit verbreitet ist, daß die meisten Compiler die Option besitzen, mit RMAC geschriebene Routinen direkt mit einbinden zu können.
Die Bedienung ist wirklich einfach. Tippen Sie doch das kleine Beispielprogramm ab, es ist in der 8080-Assemblersprache geschrieben; Spezialisten werden wissen, daß der im 128 PC eingebaute Z80-Prozessor kompatibel zum INTEL 8080 ist, aber noch ein paar Befehle mehr versteht. Dieses kleine Programm muß jetzt in Maschinensprache, den „Object-Code", umgesetzt werden. Dazu rufen wir den Assembler RMAC auf. RMAC erzeugt noch keine richtige Maschinensprache, sondern eine Datei, in der eventuelle externe Adressen, Makros usw. freigelassen werden. So eine Datei trägt den Zusatz ".REL" für „relativierbar". Daneben wird ein Übersetzungsprotokoll mit dem erzeugten Code erstellt; in der Regel wird so ein Protokoll ausgedruckt, daher enthält es den Zusatz ".PRN" (Printer).
Die Namen für die Dateien können wir explizit vorschreiben:
RMAC FILE1, FILE2=
FILE3 erzeugt FILE1.
REL und FILE2.PRN
aus der Datei mit Assemblercode FILE3.ASM, wobei .ASM als Zusatz vorgeschrieben ist. Übersetzen wir also unser Programm:
TEST.ASM
RMAC TEST,PFILE
=TEST
Wenn Sie sich nicht vertippt haben, wird ordnungsgemäß übersetzt und kein Fehler gemeldet. Falls Sie keine PRN-Datei brauchen, dann lassen Sie einfach diesen Parameter weg.
RMAC REFILE,=TEST
Nun kommt noch das „Binden" (Linken). Da wir nur eine REL-Datei linken müssen, genügt:
LINK TEST
Mit DIR TEST.* vergewissern wir uns, daß ein Programm TEST.COM auf Diskette existiert; mit TEST wird es aufgerufen. Und schon haben Sie Ihr erstes Assembler-Programm unter CP/M geschrieben.
Wer tiefer in CP/M einsteigen möchte: Eine empfehlenswerte Anschaffung, da auch in deutscher Übersetzung vorliegend, ist Rodney Zak's „Einführung in CP/M" (Sybex-Verlag, Düsseldorf).
Last not least kommt schließlich noch das Programm LIB.COM ins Spiel, mit dessen Hilfe Sie bequem auch große Mengen bereits übersetzter Assembler-Programme verwalten können (LIB kommt von „Library"). Auch hier möchten wir Sie auf weiterführende Literatur verweisen (CP/M User's Guide usw.).

CP/M UND HOCHSPRACHEN

Programmieren in Assembler ist zwar die hohe Kunst, doch existiert für CP/M eine derartige Fülle von Compilern, daß selbst für den eingefleischten „Puristen" und Geschwindigkeitsfanatiker 95% der Software von einer Hochsprache zu realisieren ist.
Auch sollten Sie beachten, daß Programme, die in Assembler erstellt wurden, beim Um- oder Aufstieg beispielsweise zum Amiga nur noch fürs Archiv taugen; Pascal-Programme dagegen sind meist innerhalb weniger Stunden übertragen, bei strikter Einhaltung der Konventionen sogar innerhalb weniger Minuten.

BASIC

Die wichtigsten CP/M-Vertreter dieser Sprache sind CBASIC und MBASIC; letzteres ist recht preiswert (ca. 200 Mark) und bietet neben einem Compiler auch noch einen Interpreter sowie einen Makro-Assembler, M80, den Linker L80 und ein Bibliotheks-Utility-Programm namens LIB80. Vorteil: Sie können mit Z80-Maschinensprache arbeiten, außerdem ist MBASIC ein überaus „kompatibles" BASIC, das auch unter MS-DOS (das gebräuchlichste Betriebssystem der großen PC-Rechner) sowie

mit Einschränkungen auch dem AmigaBASIC gleicht.

PASCAL
Unbestrittene Nummer eins ist hier Turbo Pascal; komfortabel in der Programmentwicklung, billig zu erstehen und auch sehr portabilitätsfreundlich, also auf andere Computer übertragbar, da es den Pascal-Standard weitestgehend einhält. Preis: Bei knapp 200 Mark.
Nicht zu verachten ist aber auch Nevada Pascal, insbesondere, wenn Sie viele Assembler einbinden möchten. In der „Public-Domain"-Version (= frei verfügbare Software für jeden, SIG/M Nr. 82 als „JRT-Pascal") kostet es mit Assembler, Linker, Debugger (=„Fehler-Entwanzer") und vielen Hilfsprogrammen ab zehn Mark und entspricht ebenfalls dem Standard. Bei dem Preis ist allerdings kein Handbuch und kein Editor dabei . . .

FORTRAN/COBOL
Diese Sprachen unter CP/M sind wohl in der Hauptsache für denjenigen interessant, der sie ohnehin schon beherrscht. Von Nevada gibt es beide zu einem Preis von etwa 100 Mark.

FORTH
Für alle Fans dieser Computersprache dürfte ein dickes Software-Paket ebenfalls aus der „Public Domain" (SIG/M Nr. 204, FORTH83) sein: Multitasking-Simulation (der Computer kann mehrere Aufgaben zur gleichen Zeit ausführen), rieisger Sprachumfang. Fast 600 KByte Daten, Infos, Programme, Dokumentationen – kurz, ein Paket, das alle Wünsche erfüllt. Es ist ebenso wie JRT-Pascal schon ab zehn Mark erhältlich.

„C"
Wenn Sie die COMMODORE DISC 17 aufmerksam gelesen haben, wissen Sie bereits um die grandiosen Vorzüge dieser Sprache. dBase III und UNIX, Word 2000, große Teile des Amiga-Betriebssystems und vieles andere ist in „C" programmiert. Vornehmlich mit zwei Versionen sollten Sie sich befreunden: BDS-C und C80.
Das Erstgenannte ist in den USA sehr weit verbreitet; eine stattliche Anzahl Programme, auch aus der „Public Domain", ist mit diesem Compiler erstellt worden. Leider ist es in Europa bislang kaum zu bekommen, außerdem besitzt es recht eigenwillige Funktionen.

BEISPIELPROGRAMM FÜR ASSEMBLER RMAC/LINK

```
CPM     EQU 05H; Schnittstelle
        ORG 0100H; 1. Adresse
        LXI D,TEXT; LD DE,TEXT
        MVI C,0;  LD C,9
        CALLCPM; Stringausgabe
        RET

TEXT:   DB 'Hier meldet sich'
        DB 'der C128 mit dem'
        DB 'ersten Assembler'
        DB 'programm.'
        DB 0AH,0DH; Linefeed/Carr.Ret.
        DB 'Programmierer:'
        DB '(Ihr Name!)'
        DB 0AH,0DH
        DB '$', Ende des Strings
        END
```

C80 ist nicht gerade billig, läßt aber fast alle Tricks zu, wegen denen „C" so beliebt ist.
Große Bibliothek und ausführliche Dokumentation machen den Preis (278 Mark) wieder wett; da der ANSI-Standard weitgehend erfüllt ist, können wir es Ihnen nur empfehlen.
Ferner gibt's noch ein SMALL C in der Public Domain; zum Kennenlernen ganz nett, aber im Sprachumfang so eingeschränkt, daß Sie größere Programmpakete nicht verwenden können. Doch es ist ideal zum „Reinschmecken", bevor Sie eine größere Ausgabe für ein komfortables „C"-Programm tätigen, um dann festzustellen, daß Ihnen diese Sprache doch nicht so behagt.
Übrigens: Die „professionelle" Version, die unter demselben Namen angeboten wird, bietet außer einem Handbuch und einem Editor auch nicht mehr. Nur kostet es viel mehr als die volkstümliche Public-Domain-Fassung.
Neben tausend Speziallösungen insbesondere im geschäftlichen und meßtechnischen Bereich sind vor allem drei Programme mit sehr allgemeinen Anwendungsmöglichkeiten zu nennen: dBase II, WordStar und Multiplan.
dBase II, unter CP/M und MS-DOS erhältlich, kostete noch vor einiger Zeit etwa 1000 Mark, da aber die Entwicklungskosten bei weitem schon gedeckt waren, konnte der Hersteller (Ashton Tate) überzeugt werden, daß eine Unzahl von 128PC-Besitzern in Europa auch Interesse an diesem Programm zeigen könnte, was natürlich nur eine Frage des Preises sei. Ergebnis: dBase gibt's inzwischen für etwa 200 Mark.
Dafür erhalten Sie ein ausgereiftes Datenbanksystem, das dem Benutzer jede Freiheit läßt, eine eigene Sprache bereitstellt, Programmierung erlaubt und vor allem mit sehr umfangreichen Erklärungen und Dokumentationen ausgeliefert wird, kurz: *die* Datenbank für CP/M-Computer.
Dem Textverarbeitungsprogramm WordStar kann getrost nachgesagt werden, daß es der Wegbereiter für alle nachfolgenden bildschirm-orientierten Editoren (sowas ist nämlich ein Textverarbeitungsprogramm) im weitesten Sinne war. WordStar glänzt weniger durch seine Geschwindigkeit oder seinen Bedienungskomfort, sondern vielmehr durch seine Verbreitung und seinen Bekanntheitsgrad. Voll ausnutzen kann das Programm, wer „Mailmerging" betreiben, Drukker anpassen, Formatieren, Formulare drukken und dergleichen mehr möchte. Zum Schreiben von einfachen Briefen, Programmen im Quellcode oder simplen Zeitungsartikeln, wie etwa den, den Sie gerade lesen, genügt vollauf der in Turbo-Pascal (Vertrieb: Firma Heimsoeth, München) eingebaute Editor, der in seinen Funktionen sowieso kompatibel zu WordStar ist.

MULTIPLAN – DAS „SPREADSHEET"
Multiplan (von Microsoft) hat gewisse Gemeinsamkeiten mit dBase: Es ist seit geraumer Zeit zu einem weitaus günstigeren Preis zu haben als ehedem; es ist die „große alte Dame" der Sparte; alles Vergleichbare ist mehr oder weniger „nachprogrammiert". Und: es ist absolut ausgereift.
Wir können hier unmöglich alle Funktionen beschreiben, das macht das 350seitige Handbuch dazu recht ordentlich, zudem gibt's bei den be-

kannten Fachbuchverlagen sehr gute ergänzende Bücher (Data Becker und Sybex, beide Düsseldorf, Markt und Technik, Haar).

CP/M UND INTEGRIERTE SOFTWARE

Die in den vergangenen Jahren vor allem für MS-DOS sehr oft verkauften Pakete „Eines für alle", Kalkulation, Textsystem, Datenbank, eventuell Modemsteuerung, automatische Verwaltung der Festplatte vom Programm aus und dergleichen sind zum Teil erst durch die rapide gesunkenen Preise für die RAM-Speicher möglich geworden. Solche Pakete arbeiten mit Programmgrößen von mehreren 100 KByte; für CP/M undenkbar, da es ja stets nur 64 KByte gleichzeitig ansprechen kann, das Los jedes Acht-Bit-Computers. Und genau für diese „Bit"-Zwerge, zu denen auch der C128 gehört, wurde ja CP/M entwickelt. Für die großen Personalcomputer, die schon 16 Bits adressieren können, wurde aus CP/M dann MS-DOS entwickelt. Doch brauchen wir CP/M-ler den Kopf nicht hängen lassen: Die drei Klassiker dBase, WordStar und Multiplan wurden mit der Zeit so bedeutsam, daß es sich keiner der drei verschiedenen und voneinander unabhängigen Hersteller mehr erlauben konnte, „Insellösungen" zu schaffen. Das Ergebnis: dBase-Dateien lassen sich mit WordStar editieren oder zum „Mailmerging" lesen. Multiplan-Tabellen können von den beiden anderen direkt verarbeitet werden, dafür lassen sich Recherchen aus dBase direkt mit Multiplan bearbeiten, die Ergebnisse in WordStar-Reports einlesen und ausdrucken usw. Verglichen mit den Preisen für MS-DOS-Lösungen wie etwa Framework, das im Prinzip auch nicht mehr kann – zumindest nicht in dem Sinne, daß der Privatanwender oder kleinere Betrieb diese Möglichkeiten auch ausnutzen könnte –, kommen Sie mit insgesamt etwa 600 Mark für alle drei Programme um gut 1000 Mark besser weg als der Besitzer eines „echten" MS-DOS-Personalcomputers.

GIBT'S EINE ZUKUNFT FÜR CP/M?

Angesichts der immer billiger werdenden 16-Bit-Rechner sowohl unter den PCs als auch der „68000er"-Familie (Atari ST, Amiga, Apple) muß sich der Besitzer eines 128 PC durchaus die Frage stellen: Was wird aus CP/M? Betrachten wir es mal nüchtern und ohne rosarote Brille: Zukunft hat das gute, alte 8-Bit-CP/M wohl keine mehr. Programmpakete wie dBase werden nicht mehr entwickelt werden: Die

UMSETZUNGEN FÜR DIE NEUE GENERATION

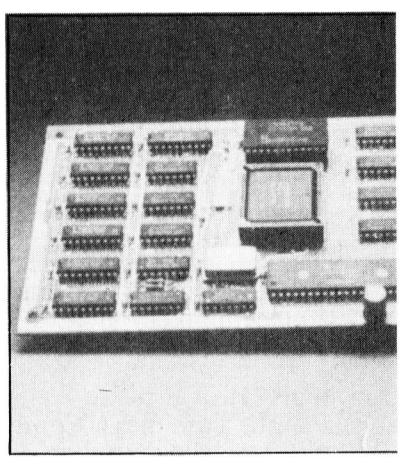
Programmierer dieser Firmen sind voll dabei, diese alten Programme 1:1 auf die Rechner der neuen Generation umzusetzen, auch wenn das den Eindruck eines Anachronismus vermittelt. Gute Ideen sind eben zeitlos. Wir als CP/M-Anwender können uns jedoch auf etwas berufen, das die bestaunten 32-Bit-Computer in absehbarer Zeit noch nicht haben werden: ein sehr ausgereiftes Standard-Betriebssystem, eine ruhmreiche Vergangenheit und eine mit einer Million CP/M-Geräten allein in Deutschland (natürlich sind das nicht nur Commodore-Rechner) nicht wegzudiskutierende Zahl von Anwendern, die, was die Rechnerleistung betrifft, für den Hausgebrauch vollkommen ausreichend gerüstet sind.
Die Hardware-Produzenten, das ist deren Geschäft und auch ihr gutes Recht, versuchen uns einzureden, wir bräuchten jedes Jahr einen neuen Computer.
Die Software-Häuser hingegen bemühen sich, uns einzureden, daß genau ihre Programme auf genau unserem Computer gut seien und schreiben folglich Programme für Rechner, die bereits in großer Stückzahl verkauft worden sind.
Dies ist die Gegenwart von CP/M: eine Gegenwart, die es uns leicht fallen läßt, ein „veraltetes", ein „kleines" Betriebssystem zu benutzen.

LM □

TEXTAUSGABE IN MASCHINENSPRACHE

Hundertmal schneller als BASIC

Maschinensprache oder Assembler ist die Ursprache Ihres C64 oder 128PC. Als Ergänzung zum Assemblerkurs auf der COMMODORE DISC möchten wir Ihnen heute zeigen, wie Sie jeden beliebigen Text mit Hilfe von Maschinensprache ausgeben können.

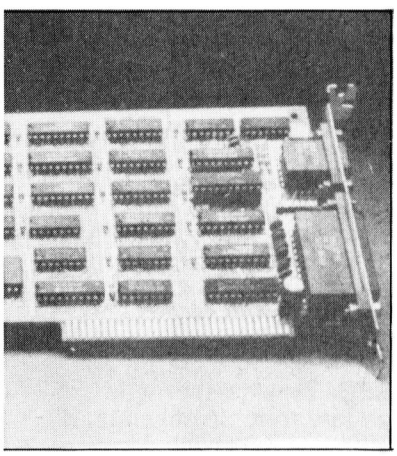


In unserem Assemblerkurs für den C64 von Ralf Trabhardt ist bereits oft genug darauf hingewiesen worden: Um Maschinensprache auf einem Computer zu programmieren, ist in jedem Fall ein Assembler oder zumindest ein Maschinensprache-MONITOR notwendig, so wie etwa der *Supermon* aus COMMODORE DISC 9. Selbstverständlich tut's aber auch jeder andere.
Sie haben auch als Anfänger sicher längst gemerkt, daß in diesem Fall der Begriff „Monitor" mit einem Bildschirmgerät nichts zu tun hat, außer daß natürlich auch mit so einem Programm ebenfalls Ihre Eingaben auf dem Sichtgerät angezeigt werden – so als würden Sie ganz „normal" in BASIC programmieren. Und das war schon das Stichwort: Ein Maschinensprache-Monitor ist nichts anderes als ein Programm für Ihren Computer, mit dem Sie Daten und Bytes bequem an die vorgesehenen Speicherstellen übermitteln können. Wenn Sie in BASIC programmieren, verfolgen Sie damit dasselbe Prinzip. Und doch besteht ein ganz erheblicher Unterschied: Da Ihr Computer im Endeffekt ja doch nur seine eigene „Maschinensprache" versteht, muß so ein BASIC-Programm von einer speziell dafür vorgesehenen Software übersetzt und in eine für den Computer verständliche Form gebracht werden: Dieses Programm wird „BASIC-Interpreter" genannt.
Bei den Commodore-Homecomputern ist er bereits im Betriebssystem enthalten und steht nach dem Einschalten des Computers zu Ihrer Verfügung. Nur dadurch ist es überhaupt möglich, den Com-

puter in einer „vermenschlichten“ Sprache anzusprechen.
Daß die Übersetzertätigkeit auf Kosten der Verarbeitungsgeschwindigkeit geht, ist sicher jedem klar. Mit einem Computer gibt's keine schnellere und zeitsparendere Kommunikation als in der Sprache seines Mikroprozessors.

EXPEDITION INS ASSEMBLER-LAND

Daher sollten auch Sie sich aufmachen, ins unbekannte Reich der Maschinensprache vorzudringen, schließlich werden Sie sicher nicht verneinen können, daß ein Aufenthalt in einem fremden Land gleich viel problemloser wird, wenn Sie die Landessprache beherrschen. Wenn wir nun behaupten, daß Maschinensprache



ebenso einfach wie BASIC ist, so möchten wir das gerne mit einigen Beispielen unter Beweis stellen. Das Problem liegt unseres Erachtens nämlich ganz woanders: Je mehr sich einer in BASIC vertieft und je besser er es beherrscht, desto schwieriger wird der Umdenkungsprozeß, wenn er sich plötzlich mit Maschinensprache beschäftigen soll. Das geht los bei der Zahlendarstellung (ausschließlich in Hexadezimal) bis hin zur Tatsache, daß ein „Object-Code“ (das fertige, lauffähige Maschinenprogramm) im Gegensatz zum BASIC-Programm überhaupt keine Zeilennummern braucht, sondern nur Bezug auf Speicherstellen innerhalb des Computers nimmt. Wie auf einer Perlenschnur gereiht, stehen die Speicherstellen im dafür vorgesehenen Bereich zur Abarbeitung zur Verfügung.
Weniger erfreulich ist daher die Tatsache, daß Sie einen fertigen Object-Code nachträglich nicht ohne tiefgreifende Manipulationen ändern können, wenn Sie etwa nur noch einige Anweisungen einfügen möchten. In BASIC wird halt einfach die neue Zeile dazugetippt und das Programm neu abgespeichert.

DAS "PRINT" IN MASCHINENSPRACHE

Doch genug der allgemeinen Betrachtungen, hier sind die versprochenen praktischen Beispiele. Heute möchten wir uns einmal mit der Ausgabe von Text in Maschinensprache beschäftigen. Der Standardbefehl in BASIC dafür ist ohne Zweifel die PRINT-Anweisung. Ab und zu wird auch ein direktes Beschreiben des Bildschirm-RAM (von Speicherstelle 1024 bis 2023) praktiziert, was aber aufgrund der ausschließlichen POKE-Befehle auch schon wieder mit Maschinensprache zu tun hat.
Das Mini-BASIC-Programm würde also so aussehen:

```
10 PRINTCHR$(147)
   :REM Bildschirm
   löschen
20 PRINT“COMMODORE
   DISC“
   :REM Text ausgeben
```

Bei der Version „Bildschirmspeicher beschreiben“ könnte das Programm so vor sich gehen:

```
10 PRINTCHR$(147)
20 FORI=0TO13
30 READD: POKE1024
   +I,D:NEXT
40 DATA3,15,13,13,15,
   4,14,18,5,32,4,9,19,3
```

In einer Schleife (Zeile 20) werden die als DATA in Zeile 40 abgelegten Werte der Bildschirmcodes (nicht der ASCII-Werte) eingelesen und die entsprechenden Bildschirmspeicherstellen eingePOKEd, ganz links oben beginnend (Speicherstelle 1024).

TEXTAUSGABE IN NULL KOMMA NICHTS

Was sich bei diesen kurzen Programmbeispielen in BASIC noch nicht so deutlich zeigt, macht sich jedoch bei größeren Textmengen, etwa für ein Menü einer Datenverwaltung oder ähnlichem grandios bemerkbar: In Maschinensprache geht der Ausgabevorgang ungleich schneller vor sich, praktisch in „Nullzeit“.
Die Software-Programmierer des C64 (und auch des C128) haben dafür bereits eine fest verankerte Betriebssystem-Routine vorgesehen und sie CHROUT genannt. Sie beginnt bei Speicherstelle 65490 ($FFD2) und hat nichts anderes zu tun, als einen ASCII-Wert, der sich im Register „Akkumulator“ befindet, an der aktuellen Cursorposition auf dem Bildschirm auszugeben.
Haben Sie Ihr Monitor-Programm für den C64 bereits geladen? (Die glücklichen C128-Besitzer müssen nur den Befehl "MONITOR“ eingeben oder die Tasten SHIFT/F7 drücken). Dann wollen wir anhand des vorher genannten BASIC-Beispieles dasselbe in Maschinensprache praktizieren. Als C64-Besitzer sollten Sie mit Ihrem Monitor den Speicherbereich ab $C000 (49152) benutzen, beim C128 eignet sich der Bereich ab $1300 (4864).
Zunächst müssen wir in einem beliebigen Speicherbereich den auszugebenden Text bereit halten, nehmen wir als Beispiel beim C64 den Bereich ab $C100, beim C128 ab $1400. Die einzelnen Bytes des Textes werden – wie beim PRINT-Befehl in BASIC – der Reihe nach in hexadezimaler Darstellung eingetippt, wobei wir noch die Zeile 10 (Bildschirm löschen) ebenfalls gleich zu Beginn als Textbyte eintragen können:
CHR$(147) = Hex. 93.
Verwenden Sie bitte die Anweisung für „Memory-Dump“, beim C128 mit dem vorangestellten Zeichen für die Spitzklammer rechts (>), bei den meisten C64-Monitoren mit dem Doppelpunkt (:).
Der Text, in dem Falle die 15 Zeichen, müssen am Ende unbedingt ein Nullbyte stehen haben ($00), das ist für den Computer das Kennzeichen, daß der Text zuende ist (siehe Listing 1 und Listing 2).
Nun kommt das wichtigste: das jeweilige Hauptprogramm, um den Text ausgeben zu können.
Beim C64 beginnen wir mit der Eingabe bei $C000, beim C128 bei $1300 (siehe Listing 3 und Listing 4).

MASCHINEN-PROGRAMME STARTEN MIT "SYS"

Verlassen Sie nun Ihren MONITOR (in der Regel mit der Eingabe von "X“) und starten Sie Ihr selbstgeschriebenes Maschinenprogramm. Beim C64 heißt der Aufruf SYS 49152, beim C128 SYS 4864.
Was machen eigentlich beide Programme?
Als erstes wird als Zähl-

variable das X-Register auf Null gesetzt.
LDX #$00
Jetzt folgt die Anweisung, den Inhalt der Speicherstelle $C100 beziehungsweise $1400 + Wert des X-Registers in den Akku zu laden. Da das X-Register zur Zeit noch den Wert „Null" besitzt, ergeben sich folgende Speicherstellen-Nummern:
C100 + 0 = C100
1400 + 0 = 1400
Gut, da wir wissen, daß hier auch der Speicherbereich unseres Textes beginnt und das erste auszugebende Zeichen ein CHR$(147) sein soll (Bildschirm löschen), ist das ganz in unserem Sinne.
Der nächste Befehl springt zu der besagten Routine FFD2 (JSR = GOSUB), gibt das Zeichen aus und kehrt wieder ins laufende Maschinenprogramm zurück. Da nun das nächste auszugebende Zeichen ein Byte weiter im Text steht, müssen wir auch unsere Schleifenzähler „X" um „eins" erhöhen, das besorgt die Anweisung INX. Allerdings möchten wir auch gerne wissen, ob das Nullbyte, das am Textende steht, bereits erreicht ist.
Dafür gibt es in der Assemblersprache den Vergleichsbefehl „Compare". Für einen Vergleich des Akkus lautet er CMP, für das X- und Y-Register CPX und CPY. Hier interessiert uns, ob im Akku dieses Nullbyte steht, demnach lautet die nächste Anweisung: Vergleiche Akku, ob „00" drin steht. Nach einem Vergleich sollte immer auch eine Reaktion erfolgen, in dem Fall eine Verzweigung zu einer bestimmten Speicherstelle. Das besorgt speziell hier der Verzweigebefehl BNE (verzweige, wenn Register ungleich dem Vergleichswert), was nichts anderes bedeutet, als daß im Akku halt noch keine „Null" steht und der Computer weitermachen soll, den nächsten Wert des Textes in den Akku zu laden, ihn auszugeben usw. Erst, wenn die Bedingung erfüllt ist, also „Null" im Akku steht, überspringt der Computer die BNE-Anweisung (sie trifft ja nicht mehr zu) und übernimmt den nächsten Befehl, in dem Falle RTS, das heißt, „kehre von der Unterroutine zurück ins Programm, von dem aus du aufgerufen worden bist". Das Aufrufprogramm ist in diesem Fall BASIC (denken Sie an den SYS-Befehl, mit dem wir dieses Programm gestartet haben), der Computer kehrt auch brav wieder dahin zurück. Stünde statt „RTS" ab dieser Stelle ein weiteres Maschinenprogramm, so würde der Computer eben dieses abarbeiten, da er die Textausgabe zu aller Zufriedenheit erledigt hat. Sie sehen also, es ist gar nicht so schwer, Text in Maschinensprache auszugeben.

ES GEHT NOCH KÜRZER

Eine komfortable und weitaus kürzere Variante stellt uns der C64 noch zur Verfügung: durch Aufruf der Systemroutine STROUT, $AB1E (43806), was soviel bedeutet wie „eine Zeichenkette ausgeben". Auch hier kann, wie im vorigen Beispiel, der Text an beliebiger Stelle eines freien Speichers stehen, ebenso muß auch hier ein „Nullbyte" als Endekennzeichnung angegeben sein (siehe Listing 5).
Bitte beachten Sie, daß diese Routine nur für den C64 gedacht ist, also im C128 nicht lauffähig ist. Auch hier lautet der Aufruf: SYS 49152.
Die ganze Arbeit für dieses noch kürzere Maschinenprogramm besteht darin, in den Akku und das Y-Register den Beginn des Speicherbereichs für den Text einzutragen, also $C100, und zwar in Form von High- und Low-Byte. Gerade bei diesem Beispiel kommt der immense Vorteil zum Tragen, daß bei der Maschinenprogrammierung nur mit hexadezimalen Zahlen gearbeitet wird. Bilden Sie doch mal auf die Schnelle das höher- und niederwertige Byte von der Dezimalzahl „49408"! Gar nicht so einfach, wie? Bei der Hexzahl dafür (C100) geht's recht einfach: in der Mitte teilen. Rechts steht das Low-Byte „00" und wird in den Akku eingetragen, das High-Byte links „C1" kommt ins Y-Register, jetzt noch mit JSR die Systemroutine aufrufen, fertig.
Wenn Sie kein weiteres Maschinenprogramm mehr angehängt haben, sollten Sie auch hier diese Maschinenroutine mit RTS abschließen.

DIE AMERIKANISCHE VARIANTE

Im alten Buch für den C64, das aus den USA stammt, habe ich noch eine interessante Variante der Textausgabe in Maschinensprache gefunden, die es wert ist, näher betrachtet zu werden. Das Prinzip beruht darauf, dem Programmierer die Möglichkeit zu geben, auszugebenden Text nicht irgendwo an einem verfügbaren freien Speicherplatz einzutragen, sich die Zahl des Textanfangs merken und die genauen Parameter zum Beispiel an das X-, Y-Register und den Akku übergeben zu müssen (was auch fehlerträchtig sein kann). Vielmehr sollte mitten im entstehenden Assemblerprogramm Text geschrieben und auch ausgegeben werden können.
Von welchen Überlegungen ging der Autor aus? Dazu muß man wissen, daß der Stack (Stapelspeicher) des C64 und C128 bei jedem JSR oder GOSUB die Rückkehradresse in seinen Speicherstellen festhält, sonst wüßte der Computer schwerlich, wo er nach diesem Sprung weitermachen sollte. So eine auf dem Stapel abgelegte Zahl enthält die um „eins" verringerte Rücksprungadresse. Das macht sich diese Maschinenroutine zunutze, da der auszugebende Text unmittelbar hinter einem JSR-Befehl folgt, in diesem Fall ein JSR-Sprung zum Ausgabe-Programm dieses Textes, das irgendwo als Unterprogramm stehen kann und mit RTS abgeschlossen ist.
Wir haben diese Unterroutine nach $C000 verlegt (siehe Listing 6). Obwohl es bei beiden Computern lauffähig wäre, haben wir uns bei diesem Beispiel nur auf den C64 beschränkt, da der C128 eine viel komfortablere Lösung bietet, Text innerhalb eines Maschinenprogramms auszugeben, wobei ein ähnliches Prinzip verwendet wird. Ein entsprechendes Beispiel später. Sehen wir uns zunächst diese trickreiche Routine dieses amerikanischen Programmierers ein wenig näher an.
Mit der Anweisung PLA zieht er ein Byte (und zwar das letzte, das gerade gestapelt wurde), vom Stack, überträgt es in den Akku. Von dort gelangt es in eine Zwischenspeicherstelle, in diesem Fall $FB. Es wäre aber jede andere freie Speicherstelle in der Zeropage denkbar. Das nächste Byte wird vom Stapel geholt und in $FC eingetragen.
Zur Schleifenbildung wird ins X-Register eine „Null" geschrieben (diese Methode kennen wir bereits), danach der Inhalt von $FB um „eins" erhöht. Das besorgt die Anweisung INC. Sofern der Wert ungleich „Null" ist, wird zu der Speicherstelle verzweigt, die den Akku mit dem Inhalt der Speicherstelle $FB + Wert des Schleifenzählers „X" lädt. Auch davon haben Sie in den vorhergehenden Beispielen schon gehört. Andernfalls, also Wert von $FB = 0, soll noch der Inhalt von $FC erhöht werden, um

dann ebenfalls den Akku mit $FB + „X“ zu laden. Jetzt kommt eine der für dieses Programm wichtigen Anweisungen: Eine Verknüpfung des Wertes „127“ ($7F) durch AND (logisches UND) mit dem Akkumulatorinhalt (unser Text). Auch hier dient die Routine CHROUT $FFD2 zur Ausgabe der Zeichen auf dem Bildschirm. Sodann wird das X-Register wieder auf „Null“ gesetzt und erneut der aktuelle Inhalt der Speicherstelle $FB + „X“ geladen, in dem Fall also von $FB selbst, da „X“ ja momentan „Null“ ist.
Und jetzt kommt der Kernpunkt: Die erwähnte logische Verknüpfung zweier Byte-Werte trägt Früchte. Der nächste Verzweigebefehl BPL (verzweige, wenn Register größer oder gleich Vergleichswert ist und die Differenz nicht mehr als $79 umfaßt), kehrt nämlich nur dann zur Einleseschleife für den Text zurück, wenn die Verknüpfung der beiden Werte mit AND#$7F diese Voraussetzungen erfüllt. Das macht sie bei den üblichen Bytewerten für unseren Text allemal, nehmen wir ruhig einmal das „C“ (Hexwert 43, Dez. 67). Überprüfen wir die Methode in BASIC:
PRINT 67 AND 127 = 67
Das Ergebnis der Verknüpfung ergibt demnach „67“, also „größer oder gleich Vergleichswert“.

KEIN NULLBYTE NÖTIG

Nun ist Ihnen sicher schon aufgefallen, daß wir hier bei der Ablage der Bytes für den Text kein abschließendes Null-Byte gefordert haben, wohl aber das letzte Byte im Text, ebenfalls ein „C“, um den Wert „128“ erhöht haben (Hex. C3, Dez. 195). In der Darstellung auf dem Bildschirm macht sich das nicht bemerkbar, wohl aber bei der Boole'schen Verknüpfung und beim anschließenden Vergleich mit BPL:
PRINT 195 AND 127=67
Das Ergebnis ist zwar dasselbe wie im vorigen Beispiel, aber die Bedingung mit BPL ist nicht mehr erfüllt, da es ja nun kleiner ist als der Vergleichswert. Demnach wird der Verzweigebefehl übersprungen, der aktuelle Akku-Inhalt aus den Speicherstellen $FB und $FC wieder ausgelesen und zurück auf den Stapel gebracht. Alles hat wieder seine Richtigkeit. Ach ja, RTS nicht vergessen, denn es ist ein Unterprogramm!
Was folgt, ist der Aufruf dieses Unterprogramms mit JSR $C000, unmittelbar daran schließt sich der Text an, wobei – wie schon erwähnt – das letzte auszugebende Byte um den Wert „128“ (Hex. 80) erhöht sein muß. Unmittelbar nach diesem Text könnte im Maschinenprogramm weitergemacht werden, da wir aber das nicht vorhaben, steht hier auch ein RTS, diesmal zur Rückkehr ins BASIC. Noch eins: Diesmal wird das Programm nicht mit SYS 49152 aufgerufen, weil dort ja schon das Ausgabeunterprogramm liegt. Der SYS-Aufruf muß an der Speicherstelle erfolgen, die ihrerseits wieder dieses Unterprogramm initialisiert, also SYS 49186.

BEIM C128 IST ALLES SO EINFACH

Wir haben es bereits erwähnt: Ein ähnliches Prinzip wie das vorhin besprochene für den C64 verfolgt eine Betriebssystem-Routine des C128, ab Speicherstelle $FF7D (genannt PRIMM, die Abkürzung von „Print Immediate“). Damit ist eigentlich schon alles gesagt: Der C128 soll die Bytes, die er hinter dem JSR-Aufruf findet solange auf dem Bildschirm ausgeben, bis er ein „Nullbyte“ findet. Das wird nämlich hierbei wiederum als Endekennzeichnung benötigt. Nach der Textausgabe setzt das Programm mit dem Befehl fort, der auf die „00“ folgt, in dem Falle eben wieder unser inzwischen recht beliebtes RTS. Listing 7 soll es Ihnen verdeutlichen, starten müssen Sie es mit SYS 4864.
Die wohl mit Abstand schnellste Textausgabe, auch in Maschinenspra-

Listing 1
So sollte unser Text im C64 gespeichert sein:

```
:C100 93 12 43 4F 4D 4D 4F 44
:C108 4F 52 45 20 44 49 53 43
:C110 00 00 00 00 00 00 00 00
```

Listing 2
Beim C128 sieht es so aus:

```
>1400 93 12 43 4F 4D 4D 4F 44
>1408 4F 52 45 20 44 49 53 43
>1410 00 00 00 00 00 00 00 00
```

Listing 3
Textausgabe beim C64 unter Verwendung von $ FFD2 (BSOUT/CHROUT):

```
C000 A2 00     LDX #$00
C002 BD 00 C1  LDA $C100,X
C005 20 D2 FF  JSR $FFD2
C008 E8        INX
C009 C9 00     CMP #$00
C00B D0 F5     BNE $C002
C00D 60        RTS
(Start mit: SYS 49152)
```

Listing 4
Textausgabe beim 128 PC mit der Routine $ FFD2 (BSOUT/CHROUT):

```
1300 A2 00     LDX #$00
1302 BD 00 14  LDA $1400,X
1305 20 D2 FF  JSR $FFD2
1308 E8        INX
1309 C9 00     CMP #$00
130B D0 F5     BNE $1302
130D 60        RTS
(Start mit: SYS 4864)
```

Listing 5
Textausgabe durch die Routine $ AB1E (STROUT, nur C64):

```
C000 A9 00     LDA #$00
C002 A0 C1     LDY #$C1
C004 20 1E AB  JSR $AB1E
C007 60        RTS
```

che, geschieht durch das Beschreiben der gewünschten Stellen des Bildschirmspeichers. Allerdings sind die Zeitunterschiede mit bloßem Auge nicht zu erkennen, aber unser Bericht wäre unvollständig, hätten wir diese Methode nicht auch noch behandelt. (Siehe Listing 8 und 9.)

ANDERE BYTE-WERTE, ABER AM SCHNELLSTEN

Bei diesem Maschinenprogramm fällt auf, daß wir uns keine Betriebssystemroutine (wie $FFD2 oder $AB1E) zu Hilfe holen müssen, der Programmierer kann sich frei entfalten, zumindest was die Position der Textausgabe anbelangt: oben, unten, in der Mitte usw. Bei den bisher genannten Beispielen war alles bei der Ausgabe von der aktuellen Cursorposition abhängig, die sich nach der Ausgabe von CHR$(147) = Hex.93 immer in der linken oberen Bildschirmecke befand. Natürlich kann die Cursorposition durch Beschreiben der richtigen Register und dem Aufruf einer Betriebssystemroutine Ihren Wünschen entsprechend angepaßt werden, aber das ist einen eigenen Artikel wert.

Zunächst wird auch bei der „Bildschirm-Routine" ein Schleifenzähler „X" bereitgestellt. Dann wird auf bekannte Manier auch hier der Akku mit den Bytewerten des Textspeichers geladen, aber – die Werte darin haben sich geändert! Dazu müssen Sie jetzt in Ihrem Textspeicher nicht mehr den ASCII-, sondern den Bildschirmcode eintragen. Auch hier bietet Ihnen Ihr Handbuch zum C64/128 eine wunderschöne Tabelle zum Nachschlagen.

Kurz gesagt: Die Bildschirmcodes sind um den Wert „64" niedriger als die ASCII-Codes, das stimmt aber nur bis zur Codezahl „32" (=SPACE), die folgenden Sonderzeichen stimmen dann wieder überein.

Diese Bildschirmcodes werden nun als Inhalt des Akku in die entsprechenden Bildschirmspeicherstellen eingetragen, ab wo, das bestimmen Sie bei der STA $, X-Anweisung. Wir haben uns hier auf die achte Zeile von oben und die erste Spalte festgelegt, Sie können das jederzeit ändern. Der Schleifenzähler wird um „eins" erhöht (INX), danach überprüft, ob bereits der letzte Buchstaben des vorgesehenen Textes eingelesen und ausgelesen wurde: CPX #$0E(=dez. 14). „14" deswegen, weil das exakt die Anzahl der auszugebenden Zeichen ist. Ist diese Bedingung nicht erfüllt, wird in der Einleseschleife weitergemacht, andernfalls die Verzweigungsprüfung übersprungen und nach dem nächsten Maschinensprache-Befehl gesehen.

Und was, glauben Sie, findet der Computer in unserem Beispiel?

Richtig, RTS.

Auf RTS, pardon: Wiedersehen bis zum nächsten Ausflug in die Assemblerwelt.

Butcher □

Listing 6
Textausgabe beim C64 unter Verwendung des Stapelspeichers:

```
C000 68       PLA
C001 85 FB    STA $FB
C003 68       PLA
C004 85 FC    STA $FC
C006 A2 00    LDX #$00
C008 E6 FB    INC $FB
C00A D0 22    BNE $C00E
C00C E6 FC    INC $FC
C00E A1 FB    LDA ($FB,X)
C010 29 7F    AND #$7F
C012 20 D2 FF JSR $FFD2
C015 A2 00    LDX #$00
C017 A1 FB    LDA ($FB,X)
C019 10 ED    BPL $C008
C01B A5 FC    LDA $FC
C01D 48       PHA
C01E A5 FB    LDA $FB
C020 48       PHA
C021 60       RTS
C022 20 00 C0 JSR $C000
: C025 12 43 4F 4D 4D 4F 4D 4F
: C02D 44 4F 52 45 20 44 49 53
: C035 C3 60 00 00 00 00 00 00
C036 60       RTS
```

Listing 7
Textausgabe beim C128 unter Aufruf der Systemroutine $ FF7D (PRIMM):

```
1300 20 7D FF JSR $FF7D
>1303 93 12 43 4F 4D 4D 4F 44
>130B 4F 52 45 20 44 49 53 43
>1313 00 60 00 00 00 00 00 00
1314 60       RTS
```

Listing 8
Direktes Beschreiben des Bildschirmspeichers (für den C64):

```
C000 A2 00    LDX #$00
C002 BD 00 C1 LDA $C100,X
C005 9D 18 05 STA $0518,X
C008 E8       INX
C009 E0 0E    CPX #$0E
C00B D0 F5    BNE $C002
C00D 60       RTS
:C100 03 0F 0D 0D 0F 04 0F 12
:C108 05 20 04 09 13 03 00 00
```

Listing 9
So wird beim C128 der Bildschirmspeicher zur Textausgabe genutzt:

```
1300 A2 00    LDX #$00
1302 BD 00 14 LDA $1400,X
1305 9D 18 05 STA $0518,X
1308 E8       INX
1309 E0 0E    CPX #$0E
130B D0 F5    BNE $1302
130D 60       RTS
>1400 03 0F 0D 0D 0F 04 0F 12
>1408 05 20 04 09 13 03 00 00
```

AUF EINEN BLICK

Assembler-Codes der CPU 6510 (C64) und 8502 (C 128)

Programmieren in Maschinensprache ist eine feine Sache, ohne genaue Kenntnis der Befehle und deren Schreibweise wird aber nicht viel daraus. Hier nun eine Übersicht dieser Anweisungen mit Hexadezimal-Codezahl und Beispielen.

Ausdruck	Codezahl	Funktion	Beispiel
A D C		Addition zweier Werte mit Übertrag (=Carry-Flag)	
	69	Addieren eines echten, unmittelbar angegebenen Wertes	ADC #$00
	65	Addieren des Speicherinhalts einer Adresse in der Zeropage	ADC $FB
	75	Addieren des Speicherinhalts einer durch das X-Register indizierten Speicherstelle in der Zeropage	ADC $FB,X
	6D	Addieren des Speicherinhalts einer 2-Byte-Adresse (absolut)	ADC $C00B
	7D	Addition des Speicherinhalts einer X-indizierten Adresse	ADC $C00B,X
	79	Addition des Speicherinhalts einer Y-indizierten Adresse	ADC $C00B,Y
	61	Addition des Speicherinhalts mit einer vorindizierten (X) Adresse in der Zeropage	ADC ($FB,X)
	71	Addition des Speicherinhalts mit einer nachindizierten (Y) Adresse in der Zeropage	ADC ($FB),Y
A N D		Logische UND-Verknüpfung des Speichers mit dem Akkumulator	
	29	Akku UND angegebener Wert	AND #$40
	25	Akku UND Speicherinhalt einer Zeropage-Adresse	AND $FB
	35	Akku UND Zeropage-Adresse + Wert der X-Schleife	AND $FB,X
	2D	Akku UND 2-Byteadresse	AND $C00B
	3D	Akku UND 2-Byteadresse + Wert der X-Schleife	AND $C00B,X
	39	Akku UND 2-Byteadresse + Wert der Y-Schleife	AND $C00B,Y
	21	Akku UND vorindizierte Zeropage-Adresse	AND ($FB,X)
	31	Akku UND nachindizierte Zeropage-Adresse	AND ($FB),Y
A S L		Bits im Akku oder einer Speicherstelle um eine Stelle nach links schieben. Entspricht einer binären Multiplikation ("hoch 2")	
	0A	Akku	ASL
	06	Zeropage-Adresse	ASL $FB
	16	Zeropage-Adresse + X-Wert	ASL $FB,X
	0E	2-Byteadresse	ASL $C00B
	1E	2-Byteadresse + X-Wert	ASL $C00B,X
B C C	90	Verzweige, wenn Register kleiner als Vergleichswert (Carry-Flag = 0). Die Adresse hinter dieser Anweisung gibt an, wohin das Maschinenprogramm springen soll.	BCC $C100
B C S	B0	Verzweige, wenn Register gleich oder größer dem zu vergleichenden Wert ist. (Carry-Flag = 1). Auch hier ist die Adresse anzugeben, wohin das Programm dann springen soll.	BCS $C150
B E Q	F0	Verzweige, wenn der Inhalt des Registers gleich dem Vergleichswert ist. (Zero-Flag = 1)	BEQ $C00C
B I T		Akku mit Bits im Speicher überprüfen. Die letzten beiden Bits (6 und 7) werden in das Negativ- und Overflow-Flag verbracht, danach wird der Inhalt des Akkumulators mit dem der angegebenen Speicherstelle UND-verknüpft und bei negativem Ergebnis (0) das Zero-Flag gesetzt.	
	24	Bit-Test mit Zeropage	BIT $FB
	2C	mit 2-Byteadresse	BIT $C00B
B M I	30	Verzweige, wenn der Register-Inhalt kleiner ist als der verglichene Wert. (Negativ-Flag = 1)	BMI $C00D
B N E	D0	Verzweige, wenn der Inhalt des Registers mit dem Vergleichswert nicht übereinstimmt. (Zero-Flag = 0)	BNE $C030
B P L	10	Verzweige, wenn das Register gleich oder größer mit dem Vergleichswert ist. (Negativ-Flag = 0)	BPL $C001
B R K	00	Unterbrechungsanweisung für ein Maschinenprogramm. Das Break- und Interrupt-Flag wird gesetzt. Eigentlich nur zum Austesten eines Programmteils notwendig, um zu verhindern, daß das Maschinenprogramm in die nächste Routine weiterspringt.	BRK
B V C	50	Verzweige, wenn das Overflow-Flag nicht gesetzt ist. (Overflow-Flag = 0) (z.B. Abfrage nach einer BIT-Anweisung)	BVC $C00B
B V S	70	Verzweige, wenn das Overflow-Flag gesetzt ist (=1).	BVS $C004
C L C	18	Das Carry-Flag wird gelöscht (=0). Sollte vor jeder Addition mit ADC angewandt werden.	CLC
C L D	D8	Eine vorher mit der Anweisung SED eingeschaltete BCD-Berechnungsart (Dezimalrechnen) wird wieder auf die übliche Binärarithmetik des Computers umgestellt.	CLD
C L I	58	Läßt den Interrupt wieder zu, der vorher mit SEI abgeschaltet wurde. Das Interruptflag wird dadurch wieder auf "Null" gesetzt.	CLI
C L V	B8	Das bereits bekannte Overflow-Flag löscht dieser Befehl (=0).	CLV
C M P		Ein angegebener Wert wird mit dem Inhalt im Akkumulator verglichen.	
	C9	Akku mit dem direkten Wert	CMP #$20
	C5	Akku mit Inhalt einer Zeropage-Adresse	CMP $FB
	D5	Akku und Inhalt einer Zeropage-Adresse + X-Wert	CMP $FB,X
	CD	Akku und 2-Byte-Adresse	CMP $C010
	DD	Akku, 2-Byte-Adresse und X-Wert	CMP $C010,X
	D9	Akku, 2-Byte-Adresse und Y-Wert	CMP $C010,Y
	C1	Akku und X-indizierte Zeropage-adresse	CMP ($FB,X)
	D1	Akku und Y-indizierte Zeropage-adresse	CMP ($FB),Y
C P X		Inhalt des X-Registers mit Wert vergleichen.	
	E0	X-Reg. und direkter Wert	CPX #$00
	E4	X-Reg. und Zeropage-Adresse	CPX $FB
	EC	X-Reg. und 2-Byte-Adresse	CPX $C001
C P Y		Dies ist der Vergleich des Y-Registers.	
	C0	Y-Reg. und direkter Wert	CPY #$00
	C4	Y-Reg. und Zeropage-Adresse	CPY $FC
	CC	Y-Reg. und 2-Byte-Adresse	CPY $C010
D E C		Inhalt der angegebenen Speicherstelle um "1" dezimieren, vermindern.	

	C6	Zeropage-Adresse direkt	DEC $FB
	D6	Zeropage-Adresse + Wert von "X"	DEC $FB,X
	CE	2-Byte-Adresse direkt	DEC $C002
	DE	2-Byte-Adresse + Wert von "X"	DEC $C002,X
D E X	CA	Den Inhalt des X-Registers um "1" vermindern.	DEX
D E Y	88	Der Inhalt des Y-Registers wird um "1" reduziert.	DEY
E O R		Der Akku-Inhalt wird einem angegebenen Wert "Exklusiv-ODER" verknüpft.	
	49	Akku mit direktem Wert	EOR #$80
	45	Akku und Zeropage-Adresse	EOR $FB
	55	Akku, Zeropage und X-Wert	EOR $FB,X
	4D	Akku, 2-Byteadresse	EOR $C050
	5D	Akku, 2-Byteadresse, X-Wert	EOR $C050,X
	59	Akku, 2-Byteadresse, Y-wert	EOR $C050,Y
	41	Akku und vorindizierte Zeropage	EOR ($FB,X)
	51	Akku und nachindizierte Zeropage	EOR ($FB),Y
I N C		Das Gegenstück zu DEC. Der Inhalt der angegebenen Speicherstelle wird um "1" erhöht (inkrementiert).	
	E6	Zeropage-Adresse erhöhen	INC $FB
	F6	Zeropage + X-Wert	INC $FB,X
	EE	2-Byte-Adresse erhöhen	INC $C002
	FE	2-Byte-Adresse + X-Wert	INC $C002,X
I N X	E8	Erhöht den Inhalt des X-Reg. um "1".	INX
I N Y	C8	Erhöht den Inhalt des Y-Reg. um "1"	INY
J M P		Hier handelt es sich um einen Sprung "ohne wenn und aber". Entspricht in BASIC dem "GOTO".	
	4C	springe zur angegebenen Adresse	JMP $C100
	6C	springe zur ersten Adresse von zweien, in denen als Low- und Highbyte die "echte" Einsprungadresse steht.	JMP ($9150)
J S R	20	Entspricht dem BASIC-Befehl "GOSUB". Ein an dieser Stelle stehender Maschinencode wird abgearbeitet, bis der Computer auf die Anweisung "RTS" stößt. Dann wird zurückgesprungen und mit den Anweisungen weitergemacht, die dem JSR-Aufruf folgen.	JSR $FFD2
L D A		Der Akku wird mit einem bestimmten Speicherinhalt geladen.	
	A9	mit direktem Wert	LDA #$00
	A5	mit Inhalt einer Zeropageadresse	LDA $FB
	B5	mit Inhalt der Zeropageadresse + Wert von "X"	LDA $FB,X
	AD	mit Inhalt einer 2-Byteadresse	LDA $C000
	BD	mit Inhalt der 2-Byteadresse + X-Wert	LDA $C000,X
	B9	mit Inhalt der 2-Byteadresse + Y-Wert	LDA $C000,Y
	A1	mit Inhalt einer vorindizierten Zeropageadresse	LDA ($FB,X)
	B1	mit Inhalt einer nachindizierten Zeropageadresse	LDA ($7A),Y
L D X		Lädt das X-Register mit einem Speicherinhalt.	
	A2	direkter Wert	LDX #$00
	A6	mit Zeropage	LDX $FB
	B6	mit Zeropageadr. + Y-Reg.Wert	LDX $FB,Y
	AE	mit Inhalt einer 2-Byte-Adresse	LDX $C00B
	BE	2-Byteadresse + Y-Reg. Wert	LDX $C00B,Y
L D Y		Das Y-Register wird mit einem Wert beschrieben.	
	A0	direkt	LDY #$00
	A4	Zeropage	LDY $FB
	B4	Zeropage + X-Wert	LDY $FB,X
	AC	2-Byteadresse	LDY $C00B
	BC	2-Byteadresse + X-Wert	LDY $C00B,X
L S R		Akku oder Inhalt einer anderen Speicherstelle um ein Bit nach rechts verschieben. Damit wird eine Division durch "zwei" erreicht.	
	4A	Operation im Akku	LSR
	46	Zeropage	LSR $FB
	56	Zeropage + X-Wert	LSR $FB,X
	4E	2-Byteadresse	LSR $C000
	5E	2-Byteadresse + X-Wert	LSR $C000,X
N O P	EA	"No Operation" = hier gibts für den Mikroprozessor nichts zu tun.	NOP
		Alle NOP-Befehle werden übersprungen.	
O R A		Wieder ein Verknüpfungsbefehl aus der "Boole'schen Wahrheitstabelle". Der Akku und ein anderer Wert werden "geODERt".	
	09	direkter Wert	ORA #$10
	05	Zeropage	ORA $FB
	15	Zeropage + X-Wert	ORA $FB,X
	0D	2-Byte-Adresse	ORA $C000
	1D	2-Byte-Adresse + X-Wert	ORA $C000,X
	19	2-Byte-Adresse + Y-Wert	ORA $C000,Y
	01	Zeropage vorindiziert	ORA ($FB,X)
	11	Zeropage nachindiziert	ORA ($FB),Y
P H A	48	Der Akkuinhalt wird auf dem Stapel (Stack) abgelegt, der Stackpointer reduziert sich um "1".	PHA
P H P	08	Der Inhalt des Statusregisters des Mikroprozessors wird auf den Stack gebracht. Der Inhalt sämtlicher Flags (Zero, Carry usw.) ist somit für eine spätere Verwendung zwischengespeichert.	PHP
P L A	68	Das Gegenteil von "PHA". Der Inhalt des Akku wird jetzt vom Stack "abgezogen". Der Stackpointer wird um "1" erhöht.	PLA
P L P	28	Das Statusregister wird wieder vom Stack geholt.	PLP
R O L		Der Akku bzw. ein Speicherinhalt wird in Verbindung mit dem Carry-Flag um ein Bit nach links "rotiert". Hier kann ein ASL um mehr als acht Bit realisiert werden.	
	2A	Akku selbst	ROL
	26	Zeropage-Adresse	ROL $FB
	36	Zeropage + X-Wert	ROL $FB,X
	2E	2-Byteadresse	ROL $C010
	3E	2-Byteadresse + X-Wert	ROL $C010,X
R O R		Der Befehl für die andere Richtung: Rotation um jeweils ein Bit nach rechts.	
	6A	Akku	ROR
	66	Zeropage	ROR $FB
	76	Zeropage + X	ROR $FB,X
	6E	2-Byteadresse	ROR $C000
	7E	2-Byteadresse + X	ROR $C000,X
R T I	40	Rückkehr von einer Unterbrechungsroutine (in dem Fall dem "RTS" vorzuziehen).	RTI
R T S	60	Rückkehr aus einem beliebigen Unterprogramm. Entspricht der Anweisung "RETURN" in BASIC.	RTS
S B C		Bestimmten Wert vom Inhalt des Akkku subtrahieren, also abziehen. Ebenso wie bei ADC wieder hier das Carry-Flag beeinflußt.	
	E9	direkte Wertangabe	SBC #$05
	E5	Zeropage	SBC $FB
	F5	Zeropage + X-Reg.Wert	SBC $FB,X
	ED	2-Byteadresse	SBC $C000
	FD	2-Byteadresse + X-Wert	SBC $C000,X
	F9	2-Byteadresse + Y-Wert	SBC $C000,Y
	E1	Zeropage vorindiziert	SBC ($FB,X)
	F1	Zeropage nachindiziert	SBC ($7A),Y
S E C	38	Damit wird das Carry-Flag, das Bit für einen übertrag bei einer Rechenoperation des Computers gesetzt. (=1)	SEC
S E D	F8	Umschalten auf BCD-Arithmetik. Der Akku hat jetzt in seinen möglichen acht Bit zwei Dezimalzahlen stehen, von denen jede jeweils vier Bit beansprucht und die nicht größer als "9" werden können, ansonsten muß das Carry-Flag wieder gesetzt werden.	SED
S E I	78	Damit werden Interrupts des Computers verhindert. Das ist vor allem dann zu empfehlen, wenn Sie die Speicherstellen für den "normalen" Interrupt ändern möchten ($0314 und $0315 beim C64 und C128), um sie auf ein eigenes Maschinenprogramm zu richten, das vor jedem Interrupt des Computers ablaufen soll. Tun	SEI

Sie das nicht, wird Ihr Computer unweigerlich abstürzen, weil das Eintragen der neuen Adreßbytes mit Sicherheit länger dauert als 1/60-Sekunde. Nach so einer Routine muß mit CLI die "normale" Unterbrechung wieder eingeschaltet werden.

S T A		Speichere den angegebenen Wert im Akkumulator.	
	85	Zeropage	STA $FB
	95	Zeropage + X	STA $FB,X
	8D	2-Byteadresse	STA $D020
	9D	2-Byteadresse + X	STA $C000,X
	99	2-Byteadresse + Y	STA $C000,Y
	81	Zeropage vorindiziert	STA ($FC,X)
	91	Zeropage nachindiziert	STA ($FC),Y
S T X		übertrage den angegebenen Wert ins X-Register.	
	86	Zeropage	STX $FB
	96	Zeropage + Y-Wert	STX $FB,Y
	8E	2-Byteadresse	STX $D020
S T Y		Der angegebene Wert wird im Y-Register gespeichert.	
	84	Zeropage	STY $FB
	94	Zeropage + X	STY $FB,X
	8C	2-Byteadresse	STY $D021
T A X	AA	Aktuellen Inhalt des Akkus ins X-Register übertragen. Der Wert im Akku bleibt unverändert, sodaß er sich dann in beiden Registern befindet.	TAX
T A Y	A8	Akku-Inhalt ins Y-Register kopieren.	TAY
T S X	BA	Inhalt des Stackpointers (Stapelzeiger) im X-Register speichern.	TSX
T X A	8A	Aktuellen X-Registerinhalt in den Akku verbringen.	TXA
T X S	9A	Das X-Register wird in den Speicher des Stackpointers übertragen.	TXS
T Y A	98	Der Registerinhalt von "Y" wird in den Akku kopiert.	TYA

Anmerkung:

Wenn Sie in der folgenden Tabelle des öfteren Erläuterungen finden wie "Zeropage + X-Wert" oder "2-Byteadresse + Y-Wert", so bedeutet das, daß nicht die rechts außen angegebene Speicheradresse gemeint ist, sondern Sie diese Zahl nur als Basis betrachten müssen. In einem „echten" Maschinenprogramm wird dann immer die Speicherstelle manipuliert, die das Ergebnis aus so einer Basisadresse + Wert, zum Beispiel des X-Registers bildet.
Beispiel: Das X-Register hat im Programmverlauf den Wert "6" angenommen. Wenn Sie jetzt die Anweisung programmiert haben: "LDA $C000,X", so betrifft das nicht die Adresse "C000", sondern "C006". Für das Y-Register gilt exakt dasselbe.
Dieser Hinweis ist vor allen Dingen für die Einsteiger in Maschinensprache unter unseren Lesern gedacht und soll von vornherein Mißverständnisse ausschalten.

BÜCHERKISTE – FÜR SIE GELESEN

Bei den Computer-Fachbuchautoren gibt es zwei grobe Richtungen: Die einen verstehen eine Menge von Computern, beim Lesen ihrer Bücher ringt man noch im Schlaf damit. Die anderen schreiben so lokker, daß man sich fragen muß, ob sie jemals den Rechner zur Hand hatten, über den sie schreiben. Es gibt aber auch Mischformen dieser Typen: Die einen, die weder vom Schreiben noch vom Computer etwas verstehen und jene die beides verstehen. Zur letzten Kategorie gehört Remy Eyssen.

Nun zum Buch:
Wurde Ihre dreijährige Tochter kürzlich zur Musterung vorgeladen, oder sollte sich Ihr Hund schon mal als Schöffe zur Verfügung halten? Wenn Sie solche oder ähnliche Erlebnis hatten, war immer der Computer daran schuld.

Dieses Buch ist ein etwas zynischer Ratgeber, der mit frechem Humor durch den Dschungel der modernen Datenelektronik führt. Die Themen reichen von ersten Tests „Bin ich computersüchtig?" über düstere Rachepläne „10 hundsgemeine Tricks, meinen Computer zu ärgern", bis zum bitteren Ende „So wird man ihn wieder los".
Der Text ist leicht verständlich geschrieben, Fachausdrücke der Mikroelektronik wie zum Beispiel „CPU = Christliche Programmierer Union" oder „Festplatte = Großes Buffet", werden in einem Computerlexikon am Ende aufgeführt.
Neben den Texten greifen einige Karikaturen aktuelle Themen auf.
Ich will nicht zuviel über das Buch erzählen, man muß es gelesen haben. Empfehlenswert ist unbedingt, es von Anfang an zu lesen, um so richtig vom speziellen Humor Remy Eyssens angesteckt zu werden.
Das Buch selbst ist auf einem Kaypro4-Computer mit WordStar geschrieben worden. Hiermit wird wohl die Zielgruppe deutlich: Alle Computer-Interessierten, die den Humor noch nicht verloren haben.

Gerhard Szymanski □

Titel: Ich hasse Computer!
Autor: Remy Eyssen
Verlag: Knaur-Taschenbuch 1985
Seiten: 95
Preis: 6.80 DM

Programmieren in C
Brain W. Kernigham u. Dennis M. Ritchie
Carl Hanser Verlag, München 1983
262 Seiten
48 Mark

Diese von Prof. Dr. A.T. Schreiner und Dr. Ernst Janich ins Deutsche übersetzte Ausgabe des amerikanischen Standardwerks von Kerningham & Ritchie heißt in C-Kreisen ehrfurchtsvoll „die Bibel" oder einfach nur „K&R", die Anfangsbuchstaben der geistigen Väter.
Das Buch weicht von der üblichen Form der Mindestsprachdarstellung ab. Normalerweise wird mit Hilfe einer Norm der Mindestumfang einer Sprache dargestellt, jedoch nicht bei C. Es existiert nämlich ein Sprachstandard, der in diesem Buch definiert wird. Aus diesem Grund gibt es (fast) keinen Anhänger von C, der dieses Buch noch nicht sein eigen nennt.
Nach einer kurzen Einführung in die Entstehungsgeschichte von C und ihrer „Philosophie" wird auf den folgenden 145 Seiten mittels einiger kurzer Beispiele der Sprachschatz vorgestellt. Da in C die I/O-Funktionen (Ein-/Ausgabe) in Standardbibliotheken abgelegt sind, befaßt sich ein kurzes Kapitel damit, etwa mit "printf()". Es ist völlig ausreichend für Sprachumsteiger wie etwa von Pascal auf „C". Einsteigern und Anfängern sei an dieser Stelle doch noch zusätzliche Lernliteratur zu C empfohlen.
Ein anderes Kapitel behandelt die Schnittstelle zum UNIX-Betriebssystem und ihre Funktionen. Abschließend besitzt das Buch einen Anhang mit einer Kurzbeschreibung der Sprache C unter Berücksichtigung verschiedener Systeme, wie zum Beispiel DEC PDP-11, DEC VAX, IBM 360/370, SIEMENS PC und MOTOROLA 68000. Ein Hoch auf die Portabilität!

FAZIT

Auch wenn die deutsche Ausgabe etwas vom Bibel-Image verliert, handelt es sich doch um ein Buch, in dem „C" sehr klar definiert wird.
G. Szymanski/her □

Bibliothek der C-Routinen
Kris Jamsa
MCGraw-Hill Book Company, Hamburg,1986
ISBN: 3-89028-78-1
48 Mark

Wie schon der Titel verrät, handelt es sich hier nicht um ein neues Lehrbuch für die Computersprache C. Das Buch liefert eine Sammlung leistungsfähiger Tools, um anspruchsvolle Programme in der Sprache C zu erstellen.
Zwar gibt das erste von elf Kapiteln einen kurzen Überblick der C-Syntax, ist aber in erster Linie als Nachschlagewerk gedacht. Sehr gut ist hier die Einführung in die Benutzung der Syntaxgraphen zum besseren Verständnis der Befehlsabläufe.
Kapitel zwei befaßt sich mit Konstanten und Makros, die in drei Deklarationstabellen zusammengefaßt wurden. Sie dienen zur Angleichung unterschiedlicher Compiler und werden in den späteren Kapiteln mit verwendet. Die weiteren Kapitel enthalten Routinen aus den Bereichen:
- Bearbeitung von Zeichenketten,
- Zeiger,
- Eingabe/Ausgabe,
- Bearbeiten von Feldern,
- Rekursion, zum Beispiel Fakultät und Türme von Hanoi,
- Sortieren, zum Beispiel Buble, Shell und Quick Sort,
- Trigometrische Funktionen,
- Zeichenkonvertierung.

Eine Besonderheit stellen die beiden letzten Kapitel dar. In Kapitel zehn „Bearbeitung von Dateien" werden unter anderem eine Großzahl von UNIX-Funktionen und Hilfsprogrammen nachgebildet (zum Beispiel ECHO, HEAD, TAIL, DIFF, GREP, MORE).
Das letzte Kapitel befaßt sich mit der UNIX-Funktion PIPE. Diese Funktion ermöglicht, die Ausgabe eines Programms direkt auf die Eingabe des zweiten Programms umzulenken.
Die guten Funktionsbeschreibungen sind dem UNIX-Programmer's Manual nachempfunden. Zu jeder Funktion werden außer Quelltext noch Aufruf, Beispiele, verwendete Makros, Funktionen und Variablen beschrieben. Zusätzlich wird in einem Pseudocode die Problemlösung veranschaulicht.

```
if(sinus Null)
    return(nicht definiert)
else
    Berechne . . .
```

Insgesamt sind hier 125 gut strukturierte Bibliotheks-Funktionen und Makros aufgeführt, die das Programmieren auch für erfahrene C-Programmierer erleichtern.
Bei dem Preis-Leistungsverhältnis darf das Buch in keiner C-Bibliothek fehlen. *Gerhard Szymanski* □

The C++ Programming Language
Bjarne Stroustrup
Addision-Wesley Publishing 1986
328 Seiten
ca. 65 Mark

Während die Sprache C gerade anfängt, sich immer mehr auf den Mikrocomputern zu verbreiten, kommt aus den Bell Labaratories ein möglicher Nachfolger dieser Sprache. C++ ist eine Anspielung auf die Sprache C und bedeutet C=C+1. Damit soll die Aufwärtskompilität der Sprache C++ ausgedrückt werden. C++ enthält die Sprache C und fügt noch einiges hinzu.
Der Autor ist zugleich der Entwickler von C++, somit erhält man mit diesem Werk Informationen zu dieser Sprache aus erster Hand.
Zur Zeit gibt es Implementierungen auf AT&T-Rechnern wie zum Beispiel DEC VAX, IBM 370 und Motorola 68000.
Das Buch hält sich stark an dem Standardwerk „The C Programming Language" von K&R und wendet sich an die schon C-erfahrenen Praktiker, denn C-Kenntnisse werden im gesamten Buch vorausgesetzt. Es werden in erster Linie die Unterschiede zu C beschrieben. Dies sind unter anderem: Typeüberprüfung, benutzerdefinierte Datentypen und Classes (Operatoren).
Das Buch erklärt im ersten Abschnitt die C++-Philosophie. Die nachfolgenden Kapitel geben einen kurzen Überblick der Sprache C++, der in sieben weiteren Kapiteln vertieft wird. Auf den letzten 70 Seiten befindet sich das Reference Manual im K&R-Stil.

FAZIT

Sollte sich die Sprache C++ auf breiter Ebene durchsetzen, hätte dieses Buch gute Chancen auf den Titel „Die C++-Bibel". Zu empfehlen ist dieses Buch jenen, die einerseits mit C vertraut sind, andererseits über gute Englischkenntnisse verfügen.
Gerhard Szymanski □

Das C-Buch
Unger/Herold
TI-WI
574 Seiten
79 Mark

C ohne Vorkenntnisse ist hier die Devise. Die Autoren Unger und Herold nutzten ihre mehrjährige Erfahrung als Schulungs-

leiter, um ein umfangreiches Lehrbuch zum Selbststudium sowie als Begleitbuch für Kurse zu schreiben.
In 16 Kapiteln von unterschiedlicher Länge und unterschiedlichem Gewicht werden die Schlüsselkonzepte der Sprache C dargestellt.
An Hand kleiner funktionsbezogener Programme werden in den ersten neun Kapiteln die Schlüsselwörter der Sprache C erläutert. Dank der schon erwähnten Unterrichtserfahrung der Autoren kennen sie die Mißverständnisse und Fehlerquellen, welche gerade den Anfängern das Erlernen der Sprache C erschwert. Im Buch geben sie daher, in hin und wieder eingestreuten Kästchen mit Merksätzen und Regeln, ausreichend Hinweise zur Abwendung der gängigsten Fehler.
Die restlichen Kapitel befassen sich mit Datentypumwandlung, Funktionen und Programmstrukturen, Zeiger und Vektoren, Dateien.
Je nach Anforderung sind die über 100 einfachen Programmierbeispiele mit Struktogrammen (NASSI-Schneidermann) dokumentiert. Etwas unglücklich ist die Verwendung von unterschiedlichen Schrifttypen zum Hervorheben einiger Wörter und Textstellen geraten. Die Schrift ändert sich bis zu siebenmal pro Seite (von fett bis Matrix-Drucker). Erfreulich dagegen ist die Berücksichtigung verschiedener Betriebssysteme und Compiler bei den Programmbeschreibungen. Es werden Hinweise zu den C-Compilern von Lattice, Digital Research, Microsoft und Intel gegeben und in einem besonderen Kapitel die Betriebssysteme UNIX, MS-DOS, ISIS und CP/M 86 behandelt. Hierzu werden auszugsweise die STDIO.H-Dateien der Betriebssysteme aufgeführt und miteinander verglichen. So dürfte bei Anpassung auf andere Umgebung keine nennenswerten Probleme entstehen.
Die wichtigsten Funktionen der Standard-Bibliothek werden in Kapitel 15 kurz dokumentiert, auch hier wird auf Differenzen hingewiesen.
Das letzte Kapitel kündigt die Behandlung der Schlüsselwörter ENUM und ENTRY an. Hier wird nur kurz auf das nicht zum C-Standard gehörende Schlüsselwort ENUM eingegangen. Zu ENTRY ist nur die lapidare Bemerkung zu lesen: „Reserviert für spätere Verwendung."
Ganz im Sinne von K&R, denn auch sie haben noch keine Definierung für ENTRY.

FAZIT

Abgesehen von dem abschreckend hohen Preis handelt es sich hier um ein erstklassiges C-Lehrbuch für den leichten Einstieg in C.

Gerhard Szymanski □

COMPUTERN LEICHT GEMACHT

Das PC-Magazin

NEU

Jetzt an ausgewählten Kiosken und im Bahnhofs-Buchhandel