128/C64-SPECIAL

128 C64 SPECIAL

Nr. 3

DM 14,80/ÖS 124/SFR 14,80

Basic-Schlüsselwörter

Einführung in die Maschinensprache

Programmiersprachen

Tips & Tricks

Super-Listings

EDITORIAL

128/ C64 Intern

HALLO LIEBE LESER!

Wieder einmal liegt ein 128/C64 SPECIAL-Heft vor Ihnen. Natürlich randvoll gefüllt mit Informationen, Tips und Tricks und jeder Menge Listings.

CPU UND WAS DAHINTERSTECKT

Schwerpunktthema in diesem Heft stellt die Maschinensprache mit allen ihren Problemen und Hintergründen dar. Unser Mitarbeiter Alfons Mittelmeyer hat zum besseren Verständnis dieses Themas einen CPU-Trainer verfaßt. Er enthält eine ausführliche Beschreibung und ein Programmlisting, mit dem sich alle Befehle sofort nachvollziehen lassen. Auch haben wir an unsere C64-Freunde gedacht. Speziell für die „kleinen Brüder" enthält dieses Heft auch eine für sie passende Version des CPU-Trainers. Gleichzeitig bringen wir zu diesem Thema einen Bericht über das Rechnen mit Binärzahlen.
Diese Einführung in die Maschinensprache nimmt zwar einen breiten Raum in der vorliegenden Ausgabe der 128/C64 SPECIAL ein. Aber was bringt es, den Befehlssatz scheibchenweise, alle zwei Monate etwas, zu servieren? Wer sich ein Auto kaufen und damit fahren will, kauft ja auch nicht zuerst die Reifen, dann das Lenkrad und die Gangschaltung und irgendwann einmal den Rest. Er wird daran nicht viel Freude haben, denn fahren kann er damit nicht. Wenn er alles beisammen hat, sind die Teile, die er zuerst erworben hat, vielleicht schon wieder verrostet, in unserem Falle vergessen. Es besteht ein Unterschied zwischen dem Lesen von Büchern über das Autofahren und dem Autofahren selbst.
Dies trifft auch auf den Umgang mit den CPU-Befehlen zu. Durch das begleitende Programm können Sie die CPU-Befehle und deren Wirkungen direkt verfolgen.
Probieren Sie also ruhig alle Befehle aus und lassen sich langsam in die Welt der Maschinensprache einführen. Die 128er-Besitzer haben es zwar durch den eingebauten Monitor etwas einfacher, doch dürften auch die C64-Besitzer auf ihre Kosten kommen.

SPIELE UND UTILITIES SATT

Wie schon gesagt, ist auch dieses Mal das Heft wieder voll mit Listings unterschiedlicher Art. Speziell für alle C64-User haben wir einige interessante Programme abgedruckt. Für alle, denen die Standardbefehle und Fehlermeldungen schon lange auf den Wecker gingen, gibt es jetzt in diesem Heft die Möglichkeit, sich ganz individuell einen neuen Befehlssatz und neue Fehlermeldungen zu kreieren. Auch dürfte der Zeicheneditor, mit dem Sie sämtliche Buchstaben, Grafikzeichen usw., neu gestalten können, großen Zuspruch finden.
Welcher C64-Benutzer hat sich nicht schon gewünscht, bequem aus dem Directory zu laden, ohne vorher das berüchtigte LOAD "S",8 zu tippen? Mit dem Programm „Easyload" geht der Wunsch nun in Erfüllung. Die Spiele-Freaks kommen natürlich auch nicht zu kurz. Das Spiel „Gelbe Kugeln" läßt langweilige Abende schnell vergessen.

Wem dies zu nüchtern erscheint, der kann sich bei „La Paloma" mit einem Gewehr bewaffnen und auf Tontauben schießen. Etwas ruhiger dagegen wird es beim Kartenspiel „Patience", bei dem Sie sich den Kopf zerbrechen werden, wie denn die verflixten Karten in die richtige Reihenfolge zu bringen sind.

Für alle Anwender haben wir „Text.128" im Heft, ein kleines Textverarbeitungsprogramm, das sich sehr leicht über die Funktionstasten steuern läßt. Einen Gag bietet das Programm „Kopfstand". Stellen Sie den kompletten Zeichensatz Ihres 128ers auf den Kopf und entdecken Sie, welche oft lustigen Konstellationen dabei herauskommen können. Wir hatten sehr großen Spaß dabei, herauszufinden, wer am schnellsten die auf dem Kopf stehenden Zeichen und Texte entziffern konnte. Außerdem dürfte es für einen geübten Programmierer nicht schwer sein, das Programm in Selbstgeschriebenes einzubinden.

Welche Methode erlaubt es besser, Ihren Computer genauer kennenzulernen, als selbst herumzutüfteln? Bewaffnet mit einem Handbuch und einer Fachzeitschrift hat schon so mancher Top-Programmierer angefangen. Wir wissen auch, daß das viele tun. Woher hätten wir sonst die vielen Programmeinsendungen, die wir ständig in der 128/C64 SPECIAL veröffentlichen?
Wenn Sie Ihr eingesandtes Programm nach dem Testen trotz allem wieder zurückbekommen haben, dann hat das nichts mit Ihren Programmierkün-

SELBST-PROGRAMMIEREN MACHT SPASS

sten zu tun, sondern lediglich damit, daß uns schon mehrere gleichartige Software von anderen Lesern vorliegt, so daß an eine Veröffentlichung in den folgenden Monaten nicht zu denken ist. Wir geben Ihnen damit die Möglichkeit, Ihr bestimmt gutes Programm woanders anzubieten. Also bitte nicht böse sein! Vielmehr hören Sie nicht auf, uns Ihre Programme zu schikken, vielleicht haben Sie eine gute Idee, die es noch nicht gab und es wert ist, veröffentlicht zu werden.
Am Ende dieses Vorwortes wollen wir Sie nochmals auffordern uns zu schreiben, wenn Ihnen irgend etwas am Heft nicht gefällt (Sie können natürlich auch schreiben, wenn Ihnen unsere 128/C64 SPECIAL gefällt). Außerdem stehen Ihnen unser Hotline-Telefon jeden Mittwoch von 15.00 bis 19.00 Uhr und unsere Mailbox zur Verfügung.
Viel Spaß beim Lesen, Tüfteln, Tippen und Testen wünschen Ihnen

Andreas Greil
und das gesamte
Redaktions-Team

INHALT

SERIE & SERVICE

TIPS & TRICKS

GRUNDLAGEN

LISTINGS C 128

LISTINGS C 64

LOAD & RUN

C – DIE SPRACHE DER ZUKUNFT

Multi-Talent

Kaum eine andere Sprache fand in der professionellen Software-Entwicklung so viele Anhänger wie die Programmiersprache C. Programme wie MS-DOS 3.0, WORDSTAR2000 oder GEM sowie Teile des Amiga-Betriebssystems sind in C programmiert worden.

Mit PROFI-C von Data Becker steht nun auch den C64/C128-Besitzern die wohl zur Zeit portabelste Hochsprache mit maschinennaher Programmierung zur Verfügung. Grund genug, uns mit der Sprache zu befassen.

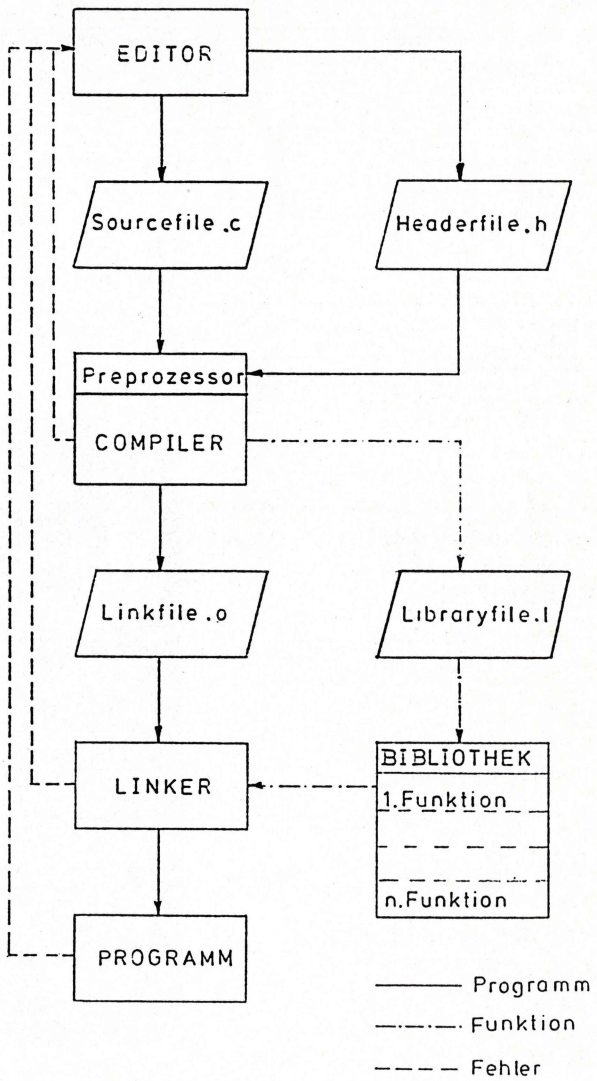
DIE „SCHÖPFUNGSGESCHICHTE":

Die Wiege von C stand im Bell-Laboratorium in Murray Hill, USA. Ihre geistigen Väter, Dennis Ritchie und Brian Kernighan, entwickelten sie 1972 aus der Sprache B, einer 1970 von Ken Thompson überarbeiteten Version von BCPL (Basic Combined Programing Language), die heute noch zur Entwicklung von Compilern und Betriebssystemen verwendet wird.
1974 war das entscheidende Jahr für C, denn das Betriebssystem Unix wurde zu 80 Prozent in C geschrieben. Damit war der Beweis erbracht, daß sich auch mit einer Hochsprache ein leistungsfähiges Betriebssystem entwickeln läßt. Unix ist immerhin Multiuser- und Multitasking-fähig).
Der erste damit ausgerüstete Rechner war ein PDP-11. Von nun an waren C und UNIX nicht mehr zu trennen.
Bis 1977 wurde die Sprache mehrmals überarbeitet und mit dem Erscheinen des Buches „The C-Programing Language" von Dennis Ritchie und Brian Kernighan (in C-Kreisen die „Bibel" genannt) war ein gewisser Standard vorgegeben, der auch heute noch bei der Entwicklung von Compilern und Bibliotheken maßgebend ist.
In den frühen 80er Jahren löste sich C von seiner UNIX-Umgebung (DEC-Rechner). Dies war endgültig der Durchbruch auf breiter Front. Mittlerweile gibt es C-Compiler auf fast allen Microcomputern und den gängigsten Betriebssystemen, zum Beispiel Acht-Bit-CP/M, CP/M 86, MS-DOS.

DIE GEGENWART

Die Sprache C ist eine sehr „kleine", nicht problembezogene, maschinennahe Compilersprache.
Es gibt im Sprachsatz von C keine I/O-Funktionen wie PRINT und INPUT im BASIC. Diese Funktionen sind in einer Standard-Bibliothek STDIO (STandarD Input/Output) abgelegt. Zu dieser Bibliothek gibt es ein Headerfile (stdio.h). In ihm sind die externen Funktionen aufgeführt. Dieses Headerfile wird vom Preprozessor in den Quelltext eingebunden und stellt das Bindeglied zwischen der Sprache C und dem System dar. Zur Verdeutlichung siehe Schema „Aufbau und Ablauf eines C-Systems".

Aufbau u. Ablauf eines C-Systems



Zur Grundausstattung einer solchen Bibliothek gehören die Funktionen:
printf() – formatierte Ausgabe Bildschirm
fprintf() – formatierte Ausgabe Datei
sprintf() – formatierte Ausgabe String
scanf(), fscanf(), sscanf(), als formatierte Eingabe sowie
putc(), getc(), getchar(), putchar(), gets(), fgets(), puts(), fputs(), fopen(), fclose(), um nur einige zu erwähnen. Sie gehören nicht unmittelbar zur Sprache C, sondern sind in C geschrieben.
Das Headerfile und die Standard-Bibliothek werden in systemangepaßter Form mit dem Compiler zusammen ausgeliefert und können nach Belieben erweitert werden. Durch diesen Kunstgriff ist der Quelltext somit vom C64 bis zum SUPER CRAY ohne nennenswerte Anpassung übertragbar.

Durch dieses Konzept ist es auch möglich, kleine einheitliche Compiler zu verwenden – im Gegensatz zu Pascal, wo man sich immer mit sogenannten verbesserten Compilern herumschlagen muß. Dies ist auch der Grund, warum MODULA 2, aus demselben Hause wie Pascal, dieselben Wege wie C einschlägt.

DER PREPROZESSOR

Der Preprozessor ist eine Besonderheit von C. Er ist die dem Compiler vorgesetzte Instanz und verändert den Quellcode, bevor er vom Compiler bearbeitet wird. Durch spezielle Befehle, die immer ein "#" vorangestellt haben, wird er aufgerufen. Einer der häufigsten ist der #include"stdio.h". Hier wird durch den Preprozessor das Headerfile der Standardbibliothek eingesetzt. Es gibt noch weitere Befehle wie etwa die Möglichkeit, mit #define Markos zu definieren oder mit #undef wieder aufzuheben.
Eine weitere Besonderheit ist die „bedingte Compilierung". Hier können zum Beispiel verschiedene Versionen mit einem Quellcode erzeugt werden. Die Auswahl der zu compilierenden Quellcode-Bereiche erfolgt mit den Auswahlbefehlen: #ifKONSTANTE. Wenn KONSTANTE ungleich Null ist, wird dieser Bereich übersetzt. Er muß dann mit #endif beendet werden.
Außerdem gibt es noch die Befehle #ifdef und #ifndef, die sich auch auf Makros beziehen.
Der Preprozessor ist eine große Unterstützung bei der Arbeit und ein weiteres Plus für C.

DER SPRACHSATZ

Sehen wir uns den Sprachsatz von C genauer an. Es handelt sich in erster Linie um Datentypen und Kontrollstrukturen sowie eine reiche Auswahl an Operatoren.

1. Kontrollstrukturen:

Kontrollstrukturen definieren die Reihenfolge der Befehlsabläufe.
if else-Verzweigungen

```
if (Ausdruck
   Anweisung 1
else
   Anweisung 2
```

C erlaubt hier eine Sonderform, nämlich die bedingte Bewertung:

```
max=(zahl1 kleiner zahl2)?zahl1:zahl2;
```

switch:
Mit der „switch"-Anweisung kann man, im Gegensatz zu IF, unter mehreren Alternativen auswählen.

```
case Ausdruck:
        Anweisung 2;
     break;
case Ausdruck2:
        Anweisung2;
     break;
case Ausdruckn:
        Anweisungn;
     break;
default:
        Anweisungd;
```

Es wird der bei „switch" angegebene Ausdruck ausgewertet und mit den „case"-Ausdrücken verglichen. Kommt es zu einer Übereinstimmung, so wird die nachfolgende Anweisung bis zur „break"-Marke abgarbeitet. Sollte keine Übereinstimmung erfolgen, so wird die Anweisung nach „default" ausgeführt.

for-Schleife:
Bei der „for"-Anweisung wird die Bedingung zum Schleifenanfang abgefragt.

```
for (Ausdruck1;Ausdruck2;Ausdruck3)
   Anweisungen
```

Ausdruck eins initialisiert die Schleifenvariable. Ausdruck zwei liefert das Abbruchkriterium. Der letzte Ausdruck ist der Schleifenzähler.

while:
Die „while"-Schleife ist der „for"-Schleife sehr ähnlich. Auch hier wird die Schleifenbedingung am Anfang abgefragt.

```
while(Ausdruck)
   Schleifenanweisung;
nächste Anweisung;
```

do...while-Schleife:
Der Unterschied zur „for"- und „while"-Schleife ist, daß die Schleifenbedingung erst am Ende abgefragt wird.

```
do
Schleifenanweisung;
while(Ausdruck);
nächste Anweisung;
```

continue:
Mit „continue" kann man eine Schleife vorzeitig abbrechen und mit dem nächsten Durchlauf beginnen.

break:
Im Gegensatz zu „continue" kann man mit „break" eine Schleife vorzeitig verlassen, ohne mit dem nächsten Durchlauf zu beginnen.

return:
Ähnlich dem RETURN im BASIC beendet „return" einen Funktionsaufruf. Der Rücksprung kann mit oder ohne Wertrückgabe erfolgen.

goto:
Als letzte Anweisung gibt es noch das verschmähte GOTO. Es sollte so selten wie möglich verwendet werden.

2. Datentypen:

In C existieren vier elementare Datentypen: char, int, float und double. Sie lassen sich noch mit den Attributen short, long und unsigned kombinieren. Dadurch ist die Größe des Speicherplatzes variabel.

char, unsigned char	–	Ein Zeichen aus einem Zeichensatz; ein Byte.
int, short, int, short	–	Eine Ganzzahl zwischen –32768 und +32767; zwei Bytes.
unsigned int, unsigned	–	Eine Ganzzahl zwischen Null und 65536; zwei Bytes.
long int, long	–	Eine Ganzzahl zwischen –214783648 und +2147483647; vier Bytes.
unsigned long int, unsigned long	–	Eine Ganzzahl zwischen Null und 4294967295; vier Bytes.

float	–	Gleitkommazahl zwischen ±9.09E-77 und ±6.78E+74 auf sechs bis sieben Stellen genau; vier Bytes.
long float, double	–	Gleitkommazahl zwischen ±9.09E-77 und ±6.78E+74 auf elf Stellen genau; sechs Bytes.

typedef:
Diese Anweisung ermöglicht die Definierung von „neuen Datentypen". Ein schönes Beispiel haben wir in der Headerdatei "stdio.h".
In der Zeile sechs wird der Datentyp FILE definiert. Dadurch kann ihm die Funktion iob(NFILE) zugewiesen werden. Im Grunde ist das nichts anderes, als dem Kind einen neuen Namen zu geben.
void:
In neueren Versionen gibt es noch den Datentyp „void" als Typ ohne Resultat. Dies kann man mit den Prozeduren in Pascal vergleichen.

3. Speicherklassen:

In C lassen sich Objekte in vier Speicherklassen vereinbaren:

auto:
Objekte der Speicherklasse „auto" sind blockorientiert und werden nach dem Verlassen des Blocks wieder gelöscht.

static:
Objekte der Speicherklasse „static" sind ebenfalls blockorientiert, behalten aber ihren Wert nach Verlassen des Blocks.

extern:
Objekte der Speicherklasse „extern" sind programmorientiert und behalten ihren Wert auch bei getrennt übersetzten Funktionen.

register:
Objekte der Speicherklasse „register" können direkt in die Prozessorregister gehalten werden, ansonsten verhalten sie sich wie „auto".

4. Strukturen:

Wie in Pascal die Records-Strukturen, so lassen sich auch in C Strukturen vereinbaren. Hierzu stehen die Schlüsselwörter „struct" und „union" zur Verfügung. Im weiteren existiert in neueren Implementierungen das Schlüsselwort „enum". Es ist dem Type bei Pascal vergleichbar.

C – DIE MASCHINENSPRACHE

Die nahe Maschinenprogrammierung wird bei C über das direkte Zugreifen mit Zeigern auf Adressen erreicht. Überhaupt haben die Zeiger (Pointer und Adressen) eine herausragende Bedeutung in C. Im Gegensatz zu Pascal, wo man erst spät oder gar nicht mit ihnen in Berührung kommt, sind sie in C ein unumgänglicher Bestandteil der Sprache. Sie sind transparent, dadurch gibt es kein „call by reference" (Übergabe des Arguments selber), sondern nur „call by value" (Übergabe des Argumentwerts).
Die Möglichkeit der Zeigernutzung geht weit über andere Sprachen hinaus. In C gibt es Zeiger auf alles, wie etwa auf Funktionen, Objekte und unbenannte Speicherbereiche von beliebiger Größe. Es läßt sich auch ein Datentyp-Zeiger auf einen Zeiger, der wiederum auf einen Vektor von Zeigern von Funktionen zeigt, deklarieren.
Hinzu kommen noch die Möglichkeiten der BIT- und ADRESSE-Manipulationen, zum Beispiel UND, ODER, EXKLUSIV ODER und SHIFTEN (verschieben). Im weiteren lassen sich häufig benutzte Variablen in Prozessor-Registern halten, was den Zugriff darauf beschleunigt. Dieses trifft natürlich nicht für den 6502/10 Prozessor zu, da er nur zwei Register besitzt.
Ein derart umfangreiches Zeiger- und Bitverarbeitungskonzept ist zur Realisierung von systemnaher Programmierung unabdingbar.
C-Programme sind blockstrukturiert und bestehen aus mindestens einem Hauptprogramm 'main()', welches eine Reihe von Funktionen aufruft.

C – SICHER UND LEICHT?

Jetzt kommt der Punkt, an dem der C-Kritiker ins Lachen kommt: Programmieren in C ist so sicher wie ein Nitroglyzeringlas in der Hand. Zwar werden vom Compiler Syntaxfehler erkannt, dies sind aber nur die harmlosen Fehler.
In C kann man zum Beispiel das 50. Element eines Feldes mit 40 Elementen nicht nur lesen, sondern auch beschreiben:
Den Buchstaben a mit PI multiplizieren und mit dem Ergebnis eine Funktion aufrufen.
Mit C ist das alles möglich, die Liste ließe sich beliebig fortsetzen.
C besitzt keine Fehlerüberwachung und Plausibilitätsüberprüfung wie Pascal. Mag einem dies auf den ersten Blick unsinnig erscheinen, darf man nicht vergessen, daß eine gute Codeüberwachung die Effizienz eines Compilers stark beeinträchtigt. Auch unterbindet sie nicht nur manch Unnützes, sondern auch Nützliches.
C ist in erster Linie eine Sprache der Profis, da das Debugging der Programme (95 Prozent aller Fehler führen zum Absturz des Systems) äußerst schwer ist. Der vom Assembler her berühmte Griff zur RESET-Taste ist auch hier das Erkennungszeichen des C-Programmierers.

C UND DIE ZUKUNFT

C ist bis heute ohne Normung ausgekommen. Zwar gibt es einen Normvorschlag (ANSI X3J11), der auch über den von Richie und Kernighan gesetzten Standard hinausgeht, dieser ist aber in vielen Implementierungen schon vorhanden. Hierbei geht es um Zuordnung von Strukturen als Ganzes, die Einführung des Datentyps „void" als Funktion ohne Rückwert und die Erweiterung des Preprozessors um die Möglichkeit, Strings zusammenzuziehen.
Während sich der Normungs-Ausschuß noch mit der Normung von C beschäftigt, steht schon ein neuer Stern am Himmel des Bell-Laboratoriums, C++.
C++ bedeutet c=c+1, ein wesentliches Erkennungszeichen von C. Entwickelt wurde C++ 1983 von Bjarne Stroustrup. Es soll aufwärtskompatibel sein, das heißt, alte C-Quelltexte können übernommen werden. Wenn überhaupt, wird es wohl noch lange Jahre dauern, bis es auf breiter Basis durchbricht.

Gerhard Szymanski □

Kennen Sie Poke 241?

Jeder, der sich schon ein bißchen mit dem Handbuch zum 128PC beschäftigt hat, weiß, daß durch die Anweisung COLOR 5, Farbzahl 1 – 16 jeweils die gewünschte Farbe eingestellt werden kann, die die Zeichen im Textmodus auf dem Bildschirm bekommen sollen. Das ist im 40-Zeichen-Modus nicht anders als im 80-Zeichen-Bildschirm.

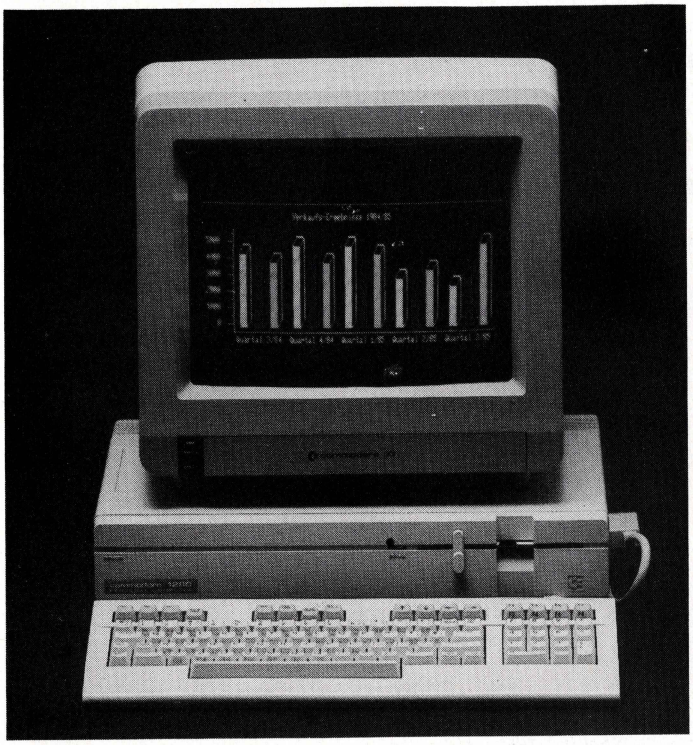
Möchten Sie beispielsweise, daß irgendein Textstring ab sofort in weiß auf dem Bildschirm erscheint, so genügt davor die Anweisung COLOR 5,1. (Die einzelnen Farbcodes lesen Sie im Handbuch zum C128 nach.) Es sind Werte von 1 bis 16 möglich, jede höhere Zahl erzeugt die Fehlermeldung „ILLEGAL QUANTITY". Allerdings gibt's im C128 noch eine Speicherstelle, in die Sie die Farbcodezahl mit dem POKE-Befehl eintragen können: 241 (SF1). Im Handbuch

Farbgebung der Zeichen ohne COLOR-Befehl

unter Anhang H-15 steht diese Adresse als „COLOR", Attribut des nächsten auszugebenden Zeichens, deklariert. Was ja auch den Kern der Sache trifft.
Und doch ist uns zur Farbgebung mit dem Befehl COLOR 5 ein gravierender Unterschied aufgefallen: Bei der Anweisung POKE 241, Farbcode-Zahl, müssen die erlaubten Werte jemals um 1 niedriger sein, also von 0 bis 15 reichen. Damit erreichen Sie zumindest in der 40-Zeichen-Bilddarstellung denselben Effekt wie mit COLOR 5. Die Adresse 241 im C128 entspricht hier in allen Belangen der Speicherstelle 646 im C64.

Andere Farbcodes beim VDC-Chip

Anders im 80-Zeichen-Modus, für den nicht der VIC-II-Chip (Video Interface Controller) zuständig ist, sondern der VDC-Baustein. Der unterstreicht nämlich seine Sonderstellung, den er beim C128 einnimmt, damit in eindrucksvoller Weise. Reagiert er noch bei der „normalen" Farbgebung mit



Gewisse Speicherstellen des C128, mit den richtigen Werten beschrieben, können Erstaunliches vollbringen

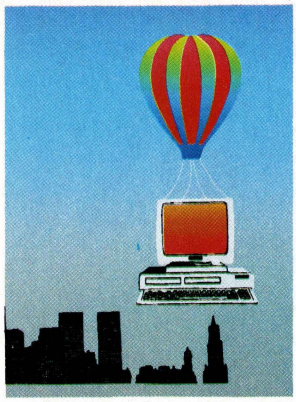
COLOR 5, so wie wir es gewohnt sind (die Farben werden in der Reihenfolge gezeigt, wie sie auf Ihrer Tastatur unterhalb der oberen Zahlentasten bezeichnet sind), so ändert sich das mit POKE 241, Farbzahl 0 bis 15, doch erheblich: Die gewohnte Reihenfolge stimmt nicht mehr. Auf „Schwarz" zum Beispiel folgt nicht „Weiß", sondern „Dunkelgrau" (POKE 241,1). Die weiße Farbe erreichen Sie jetzt mit POKE 241,15. Alle Veränderungen sehen Sie am besten, wenn Sie das kleine Listing abgetippt haben und es starten. Dabei fällt auf, daß die geraden Farbwerte alle eine gewisse Grundfarbe bilden. Die folgenden ungeraden Farbzahlen bilden im Prinzip dieselbe, nur um einige Grade heller. Auf „Normal-Blau" folgt zum Beispiel Hellblau, auf Rot eben Hellrot. Spätestens jetzt wird uns der Unterschied zur üblichen Farbcode-Verteilung klar.

Vier Editor-Funktionen mit einem POKE-Befehl

Bisher haben wir nur das untere „Nibble" (1 Nibble = 4 Bit) dieser außergewöhnlichen Speicherstelle erfaßt, also einen Wert von 0 bis 15 hineingePOKEd. Die oberen vier Bit waren damit nicht eingeschaltet, also auf „Null" gesetzt. Uns hat natürlich interessiert, was passiert, wenn diese Bit auch gesetzt sind.
Der Gesamtwert des oberen Nibbles beträgt 240, also muß dieser Wert zur Farbcode-Zahl 0 bis 15 addiert werden.
Und hier beweist sich die Adresse 241 als Multitalent: Ein POKE 241, 241 bringt zwar ebenfalls die dunkelgraue Zeichenfarbe auf den Bildschirm, zusätzlich werden aber noch folgende Funktionen des Bildschirm-Editors beim C128 aufgerufen:

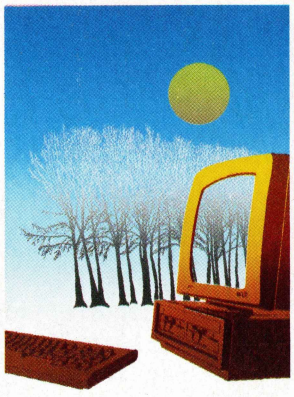
1. Einschalten des Invers-Modus
   =PRINT CHR$(18)
2. Zeichensatzumschaltung von Großschrift-/Blockgrafik in den Klein-/Großschrift-Modus
   = PRINT CHR$(14)
3. Der Blink-Modus, der nur beim 80-Zeichen-Bildschirm funktioniert, wird eingeschaltet.
   = PRINT CHR$(15)

Auch diese Funktionen verdeutlicht das kleine Programm, das Sie hier abgedruckt finden. Wir sind sicher, daß Sie bei der eigenen Programmiertätigkeit im 80-Zeichen-Modus diese vielseitige Adresse 241 noch oft verwenden werden. *hb*□



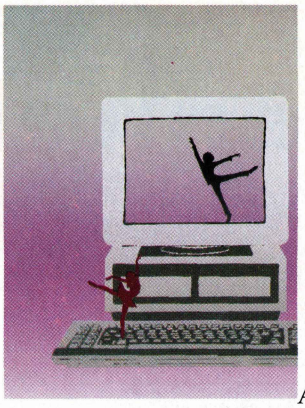
AT 588 31

AT 588 32

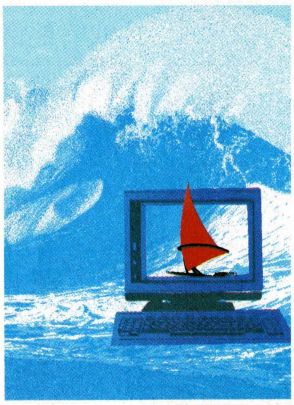


AT 588 33

AT 588 34

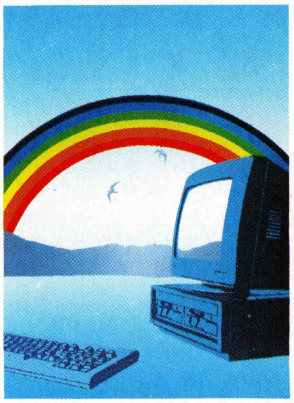


AT 588 35

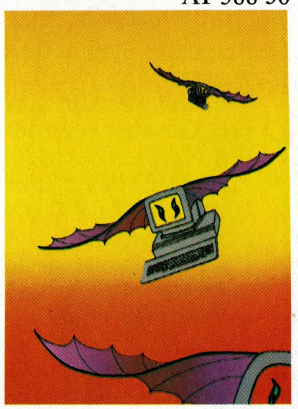


AT 588 29

Edition Computer-Kunst

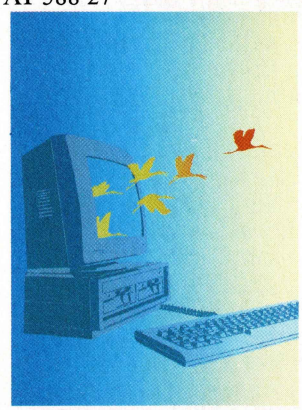


AT 588 30

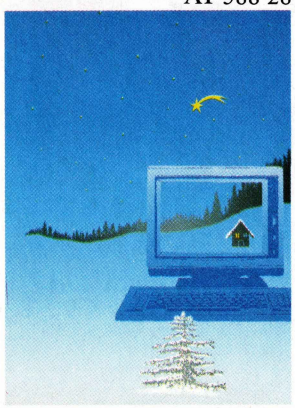


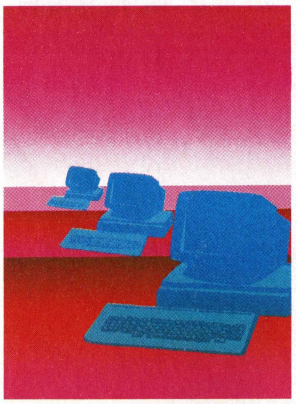
Computer sind nicht nur Gebrauchsgegenstände. Sie werden zunehmend auch von Künstlern als Motiv entdeckt. Einige besonders gut gelungene Arbeiten haben wir für Sie, unsere Leser, reservieren können. Alle Motive sind strikt auf eine Auflage von 99 Exemplaren limitiert und von der Künstlerin, Sybille Areco, handsigniert und numeriert. Die Exponate werden nach Bestelleingang im 24-Farbendruck von Hand gefertigt, die Vorlage nach dem 99. Druck vernichtet. Unser Angebot: Jedes Motiv nur DM 85,-, zwei Motive DM 150,-, drei DM 210,- und vier Motive nur DM 250,-. Jedes Bild ist 30 x 40 Zentimeter groß und kommt im Passpartout in stabiler Verpackung (im Preis enthalten).
Lieferzeit nach Bestelleingang: ca. drei Wochen.

AT 588 27

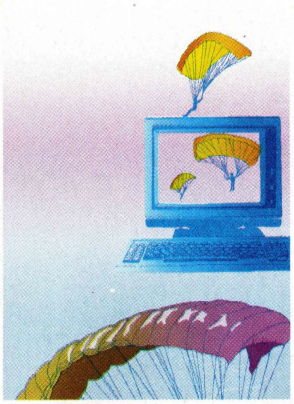


AT 588 28





AT 588 25



AT 588 26

Bestellcoupon

Hiermit bestelle ich in Kenntnis ihrer Verkaufsbedingungen folgende Exponate:

Nr.: _____

_____

Ich zahle: (Zutreffendes bitte ankreuzen!)
per beigefügtem Scheck ☐ Schein ☐ Gegen Bankabbuchung am Versandtag ☐

Meine Bank (mit Ortsname) _____

Meine Kontonummer _____

Meine Bankleitzahl _____ **(steht auf jedem Bankauszug)**

Nachname _____ **Vorname** _____

PLZ/Ort _____ **Str./Nr.** _____

Verkaufsbedingungen: Lieferung nur gegen Vorkasse oder Bankabbuchung. Wichtig: Scheckeinreichung und Bankabbuchung erfolgen erst nach dem Versand. Keine Nachnahme möglich. Auf Wunsch Rechnung mit ausgewiesener Mehrwertsteuer.

Unterschrift _____
Bitte ausschneiden und einsenden an

AKTUELL-VERLAG
Heßstraße 90
8000 München 40

Rechnen in anderen Zahlensystemen

Rechnen mit Binärzahlen

Inzwischen weiß es jeder, Computer rechnen nicht mit den uns gewohnten Zahlen von 0 bis 9 (Dezimalsystem), sondern mit nur zwei Zeichen oder Zuständen: der 0 und der 1. Im Grunde ist der Computer dumm, auch das ist bekannt. Er muß beim Rechnen gewissermaßen „an den Fingern abzuzählen", um zu Ergebnissen zu kommen. Finger heißt im Lateinischen *digitus*. Die zwei möglichen Zustände (1 oder 0) und die Art des Rechnens (Abzählen) verlieh der „kleinsten Einheit" für den Computer den Namen Bit (= **Bi**nary Digit).

1. Stellenwerte

Für das Rechnen im Binärsystem gelten andere Regeln als im geläufigeren Dezimalsystem, dennoch gibt es Gemeinsamkeiten: Jede Ziffer bei mehrstelligen Zahlen im Dezimalsystem hat gegenüber der rechts neben ihr stehenden einen zehnfach höheren Stellenwert. Die „8" in der Zahl „48" repräsentiert 8 Einheiten (Einer). Die „4" links daneben hat den Wert von 4 Zehnern, also 40 (40 + 8 = 48).

Das Binärsystem arbeitet prinzipiell nach dem gleichen Schema, jedoch beträgt der Stellenwert jeder Binärstelle immer das zweifache des Stellenwertes der benachbarten rechten Stelle. Zur Verdeutlichung hier eine Zahlenreihe in dezimaler und binärer Form:

1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
48	110000

Am Beispiel läßt sich auch die Umrechnungsmethode von Binärzahlen in Dezimalzahlen nachvollziehen. Um eine Binärzahl zu schreiben, werden an die Stellen, welche addiert die Dezimalzahl ergeben, Einsen geschrieben:

32	16	8	4	2	1					
1	1	0	0	0	0	=	32	+ 16	=	48
1	0	0	0	0	0				=	32
		1	0	0	0				=	8
		1	0	0	1	=	8	+ 1	=	9
				1	0				=	2

Hier sind schon einige Vor- und Nachteile der beiden Systeme zu erkennen: während das Dezimalsystem zur Darstellung einer Zahl weniger Stellen benötigt, braucht es andererseits viel mehr verschiedene Zeichen (zehn gegenüber zwei).

Nach diesem Prinzip lassen sich beliebig große Zahlen binär darstellen. Damit sind sie vom Computer „lesbar".

Das allein genügt natürlich noch nicht, es muß auch mit diesen Zahlen gerechnet werden. Zunächst etwas Theorie hierzu; In dezimalen, beinären oder anderen Zahlensystemen können arithmetische und logische Operationen durchgeführt werden. Bei diesen Operationen treten Überträge auf, zum Beispiel bei der Addition von 3 + 7 = 10 im Dezimalsystem. Die Null wird in die Einerstelle gesetzt und die Eins in die Zehnerstelle übertragen.

Im Binärsystem wird 1 + 1 = 10 geschrieben (siehe oben). Auf diesem fundamentalen Grundsatz beruht die Leistungsfähigkeit der digitalen Datenverarbeitung.

Auch die Multiplikation schrumpft dabei auf die Tatsache zusammen, daß 1 * 1 = 1 ist. Damit ist der Rechengenauigkeit theoretisch keine Grenze gesetzt, denn es ist gleichgültig, ob 10 oder 100 mal eine Binärstelle 0 oder 1 wird; lediglich die Anzahl der verfügbaren Stllen begrenzt die Berechnung.

Subtraktionen können auf die Addition zurückgeführt werden und die Division wiederum auf die Subtraktion (dazu später mehr). Letztlich lassen sich damit alle Grundrechenarten auf die Addition zurückführen, wenn man die Bildung von Überträgen berücksichtigt, was sich mit Hilfe der Komplementbildung realisieren läßt.

2. Komplementbildung

Während die Addition auch im Rechner leicht verwirklicht werden kann, muß für die Subtraktion das Verfahren der Komplementierung angewendet werden. Unter Zuhilfenahme des Komplements einer Zahl lassen sich alle Vorgänge wieder durch Addition darstellen.

Das Komplement ist eine Zahl, die eine gegebene Zahl zu einer bestimmten Einheit ergänzt. Im Dezimalsystem wird in das Zehner- und Neunerkomplement, und im Binärsystem in das Einer- und Zweierkomplement unterschieden.

Das Zehnerkomplement bedeutet die Ergänzung jeder Ziffer einer Dezimalstelle auf 10. Die Komplementärzahl zu 1 ist 9, zu 2 ist 8 usw.

Das Neunerkomplement ist die Ergänzung jeder Ziffer auf 9. Also 1 und 8, 2 und 7 usw.

Hierzu ein Beispiel:
Von der Zahl 468 soll die Zahl 147 subtrahiert werden. Die Komplementärzahl zu 147 ist im Neunerkomplement 852.

```
1 4 7
+ + +
8 5 2
= = =
9 9 9
```

Wird das Komplement zur Zahl 468 addiert, so erhält man 1320. Abzüglich des Wertes 147 + 852 = 999 ergibt sich das Resultat mit 321.

Subtraktion:

```
  468
- 147
-----
  321
```

Komplementäre Addition:

```
  468
+ 852
------
 1320
- 999
------
  321
```

Im Binärsystem wird mit dem Zweierkomplement gearbeitet. Dabei muß jede Binärziffer auf 10000 (dez.16) ergänzt werden.

Doch zunächst wird einmal erklärt, wie Binärzahlen überhaupt addiert werden können.

3. Addition binärer Zahlen

Folgende Grundregeln gelten bei der Addition binärer Zahlen:

```
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 (+Überlauf 1)
```

Die letzte Zeile erzeugt allerdings einen „Überlauf". Die 0 wird gesetzt und der Überlauf beträgt 1, der zur nächsten Stelle übertragen wird.

Hier einige Beispiele.

Dezimal	Binär
2	10
+ 2	+ 10
4	100
10	1010
+ 4	+100
14	1110
14	1110
+ 2	+ 10
16	10000

Bekanntlich arbeitet ein Computer mit Byte, welche ihrerseits aus acht Bit bestehen, die in „gepackter" Form vorliegen. Bei der natürlichen binären Darstellung laufen die Bit der einzelnen Byte ineinander über und in der Binär-Dezimal-Darstellung (das ist die Verschlüsselung von Dezimalziffern beispielsweise für den Bildschirm) wird jede Ziffer durch vier Bit, also anderthalb Byte (oder Nibble) dargestellt. Als Beispiel wird im folgenden die Zahl 121 einmal in natürlicher binärer Form dargestellt und einmal in Binär-Dezimal-Form.

Natürliche binäre Form:

1111001
= 64 + 32 + 16 + 8 + 1 = 121

Binär-Dezimal-Form:

0001 00010 00001
1 2 1

Damit dürfte der Unterschied zwischen natürlicher binärer Form und Binär-Dezimal-Form deutlich geworden sein. Nun noch ein Beispiel einer Addition in Binär-Dezimal-Form:

Es werden die Zahlen 49 und 37 addiert (= 86).

```
  0100  1001   = dez.49
+ 0011  0111   = dez.37
-------------
  0111  1110   = Zwischenergebnis
+          1   Übertrag
-------------
  0111  1100   = Zwischenergebnis
+          1   Übertrag
-------------
  0111  1000   = Zwischenergebnis
+          1   Übertrag
-------------
  0111 1 0000  Dieses Ergebnis liegt
               außerhalb des Vier-Bit-
               Codes, daher Subtrak-
+ 0001 -1010  tion einer dez.10 und
               Übertrag
  1000  0110   = dez.86
```

4. Subtraktion binärer Zahlen

Die Subtraktion wird, wie schon erwähnt, in eine Addition umgewandelt. Die abzuziehende Zahl erhält dazu führende Nullen vorgestellt. Daraufhin werden alle Nullen in Einsen und umgekehrt überführt, die Zahl wird invertiert. Das Komplement einer Zahl ist also zu addieren. Die im Zwischenergebnis bei der erläuterten Addition am weitesten links stehende Eins wird gestrichen und an letzter Stelle addiert.

Beispiel:

```
10000 = dez.16
 1010 = dez.10
01010 = dez.10 mit
        führenden
        Nullen
10101 = Inversion
```

Addition:

```
  10000
+ 10101
 100101
+     1
  00110 = dez.16 – 10
          = 6
```

Und noch ein Beispiel:

68 – 23 = 45

```
1000100 = dez.68
  10111 = dez.23
0010111 = dez.23 mit
          Vornullen
1101000 = Inversion
```

Addition:

```
  1000100
+ 1101000
 10101100
+       1
 0101101 = dez.45
```

5. Division binärer Zahlen

Wie schon erwähnt, wird die Division binärer Zahlen auf eine Zwischensubtraktion und diese auf eine Addition des Komplementes zurückgeführt. Es gibt hier nur die Möglichkeit: Einmal oder keinmal ist der Divisor im Dividenten enthalten. Als Beispiel dient folgende Division:

66 : 22 = 3

```
1000010 = dez. 66
  10110 = dez. 22
0010110 = Nullen-
          auffüllung
1101001 = Inversion
```

Division:

```
  1000010 : 10110 = 11
+ 1101001
 10101011
+       1
  0101100
+ 1101001
 10010101
+       1
  10110
```

bin.11 = dez.3

6. Multiplikation binärer Zahlen

Die Gesetze der binären Multiplikation entsprechen denen der binären UND-Verknüpfung aus der Boole'schen Algebra. Hierbei gilt:

logische UND-Verknüpfung

0 UND 0 = 0
0 UND 1 = 0
1 UND 0 = 0
1 UND 1 = 1

binäre Multiplikation

0 * 0 = 0
0 * 1 = 0
1 * 0 = 0
1 * 1 = 1

An dieser Stelle sind die Wahrheitstabellen der Boole'schen Algebra zu erwähnen, um die logischen UND-Verknüpfungen besser verstehen zu können. Verständlich wird diese Zuordnung aber auch dann, wenn man sich ein Kabel vorstellt, in das zwei Schalter hintereinander eingebaut sind. Bezeichnet man die Schalterstellungen mit AUS = 0 und EIN = 1, so ist ersichtlich, daß nur dann ein Strom fließen kann, wenn beide Schalter auf EIN = 1 stehen. Ist auch nur einer der beiden Schalter auf AUS = 0, so ist der Stromkreis unterbrochen, es fließt also kein Strom (=0). Hier ist schon die Verbindung vom Binärzahlensystem zur elektronischen Rechenanlage zu erkennen. Beispiel einer Multiplikation durch fortlaufende Addition:

```
      13 * 5

   1101 * 101
     1101
     0000
+  1101
  1000001 = dez.65
```

Wie beim manuellen Rechnen mit Papier und Bleistift wird jede Produktzeile gegenüber der vorhergenenden um eine Stelle verschoben. Zu Beginn mit der Wertigkeit 1 des Multiplikators, dann 2, 4, 8 usw. von rechts nach links.

Im folgenden Schema ist die o.a. Multiplikation Schritt für Schritt unter Zuhilfenahme von vier Registern aufgezeigt. Ungefähr so läuft der Vorgang im Rechner ab. Dabei gibt es zwei Abbruchbedingungen für die Maschine: Wenn in Register A ein Überlauf auftritt, oder wenn B = 0 ist. Letzteres ist hier am Schluß der Fall. Die Multiplikation ist somit durchgeführt.

Ablauf einer Multiplikation mit Registern in einem Rechner:

Register A Multiplikant	Register B Multiplikator	Register C Test ob 1	Register D Ergebnis
0000001101	0000000101	1 (addiere A auf D)	0000000000 0000001101
schiebe links 0000011010	schiebe rechts 0000000010	0	
schiebe links 0000110100	schiebe rechts 0000000001	1 (addiere A auf D)	0001000001
schiebe links 0001101000	schiebe rechts 0000000000	0	

Die Register bestehen aus Speichergliedern. Jedes Speicherglied kann ein Bit darstellen. Zur Darstellung von zehn Bit sind also zehn Speicherglieder notwendig, die zu einem Register zusammengefaßt wurden. Die Register A und B sind dabei Schieberegister.

Die Stelle des Multiplikators mit der Wertigkeit 1 wird geprüft, ob sie 0

Basis^n	n für Basis 2	8	16	Basis^{-n}
1	0	0	0	1 0
2	1			0 5
4	2			0 25
8	3	1		0 125
16	4		1	0 062 5
32	5			0 031 25
64	6	2		0 015 625
128	7			0 007 812 5
256	8		2	0 003 906 25
512	9	3		0 001 953 125
1 024	10			0 000 976 562 5
2 048	11			0 000 488 281 25
4 096	12	4	3	0 000 244 140 625
8 192	13			0 000 122 070 312 5
16 384	14			0 000 061 035 156 25
32 768	15	5		0 000 030 517 578 125
65 536	16		4	0 000 015 258 [illegible] 5
131 072	17			0 000 007 629 [illegible] 25
262 144	18	6		0 000 003 814 [illegible] 625
524 288	19			0 000 001 907 [illegible] 812 5
1 048 576	20		5	0 000 000 953 674 [illegible] 406 25
2 097 152	21	7		0 000 000 476 837 158 203 125
4 194 304	22			0 000 000 238 418 579 101 562 5
8 388 608	23			0 000 000 119 [illegible] 259 550 781 25
16 777 216	24	8	6	0 000 000 059 [illegible] 644 775 390 625
33 554 432	25			0 000 000 029 802 322 387 695 312 5
67 108 864	26			0 000 000 014 901 161 193 847 656 25
134 217 728	27	9		0 000 000 007 450 580 596 923 828 125
268 435 456	28		7	0 000 000 003 725 290 298 461 914 062 5
536 870 912	29			0 000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	10		0 000 000 000 931 322 574 615 478 515 625
2 147 483 648	31			0 000 000 000 465 661 287 307 739 257 812 5

oder 1 ist. Hier im Beispiel ist sie 1 und die Addierschaltung veranlaßt die Addition von Register A auf Register B. Bei den folgenden Schritten wird immer zuerst der Inhalt von Register A um eine Stelle nach links, der von B eine Stelle nach rechts geschoben, wie oben zu sehen ist. Wenn die rechte Position der Zahl in Register B nun 1 ist, wird die Addition von A auf D veranlaßt. Bei 0 wird sofort die nächste Verschiebung durchgeführt. Das Endergebnis steht am Schluß in Register D.

Damit wurden die vier Grundrechenarten im binären Zahlensystem erläutert. In der Regel braucht sich der Anwender nicht um diese „maschinen-nahen" Operationen zu kümmern. Es sei auch erwähnt, daß es noch andere Zahlensysteme neben Dezimal- und Binärsystemen gibt, mit denen der Computer arbeitet, beispielsweise Hexadezimal- oder Oktalzahlen.

Dipl. Ing. (FH)
Oliver Rosenbaum■

Das Hexadezimalsystem

Wer sich mit Assemblerprogrammierung beschäftigen will, kommt nicht darum herum, sich auch der Hexadezimalzahlen anzunehmen. Hexadezimale Zahlen-Darstellungen erlauben, große Zahlen mit relativ wenigen Stellen darzustellen. Allerdings werden dabei mehr Zeichen benötigt.

Das Hexadezimalsystem wird auch Sedezimalsystem genannt von lat. sedecium = sechzehn. Hexadezimal kommt vom Griechischen hexa (sechs) und Lateinischen decem (zehn).

Während im Binärsystem nur zwei verschiedene Zeichen verwendet werden (0 und 1) und im Dezimalsystem bekanntlich zehn, benötigt das Hexadezimalsystem 16 Zeichen, was auch den Namen dieses Zahlen-Systems erklärt.

Schauen wir uns einmal die verschiedenen Zahlensysteme im Vergleich an:

Binär	Dezimal	Hexadezimal
0	0	0
1	1	1
10	2	2
11	3	3
100	4	4
101	5	5
110	6	6
111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F
10000	16	10

Das Hexadezimalsystem verwendet also die Ziffern von 0 bis 9 und zusätzlich die Buchstaben von A bis F. Es ist immer wichtig, zu wissen, in welchem Zahlensystem man sich befindet, denn beispielsweise ist (bin) 10 000 = (dez) 16 = (hex) 10.

Man gibt also immer das System mit an, und sei es nur durch einen Index (2,10 oder 16). Wir verwenden hier in Klammern gesetzte Abkürzungen.

Ein weiteres Beispiel einer Zahl in den drei Systemen:

(dez) 934 = (bin)
1110101111 = (hex) 3AF

Man rechnet die Binärzahl nach dezimal:

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

512 + 256 + 128 + 32 +
8 + 4 + 2 + 1 = 943

Binär nach hexadezimal läßt sich leicht nach der oben gezeigten Tabelle realisieren: Jede Dezimalzahl wird im Computer durch vier Bit dargestellt. Wird die Binärzahl aus dem Beispiel, von rechts beginnend, in Vierergruppen unterteilt und die entsprechenden hexadezimalen Werte aus der Tabelle abgelesen, ergibt sich:

```
11  1010  1111
 3     A     F
```

also ist
(bin) 1110101111 = (hex)
3AF

Genauso einfach lassen sich hexadezimale Zahlen binär darstellen. Es ist nur darauf zu achten, daß kleinere Binärzahlen mit Vornullen aufgefüllt werden.

Beispiel:
(hex) 333 = (bin)?

```
 3     3     3
0011  0011  0011
```

(hex) 333 = (bin)
1100110011

Die Umrechnung von binär nach hexadezimal (und umgekehrt) läßt sich also am einfachsten anhand von Tabellen durchführen. Dies resultiert aus der Tatsache, daß die Basis 2 des Binärsystems sich problemlos in die Basis 16 des Hexadezimalsystem umrechnen läßt.

Anders verhält es sich dagegen mit dem Dezimal- und dem Hexadezimalsystem (Basis 10 und 16).

Die Umrechnung einer Dezimalzahl in eine hexadezimale Zahl erfolgt schrittweise. Die Dezimalzahl wird immer wieder durch 16 geteilt, wobei der Rest der ersten Division die niedrigwertigste Ziffer der Hexadezimalzahl ergibt. Der Quotient wird dann wiederum durch 16 geteilt, der Rest links neben den vorhandenen Rest geschrieben, was die zweitniedrigste Stelle ergibt usw.

Beispiel: Die Zahl (dez) 246 soll hexadezimal geschrieben werden:

```
246 : 16 = 15
       Ergebnis (hex) F 6
16
086
 80
  6   ---(=hex) ---6

15 : 16 = 0
 0
15     ---(= hex)---F
```

Bei der Umwandlung von Hexadezimalzahlen in Dezimalzahlen bilden die Zeichen von 0 bis F den Multiplikanden. Der Stellenwert der Hexadezimalzahl (von links nach rechts) gibt den Exponenten an, mit dem die Basis 16 potenziert werden muß. Jedes Zeichen wird mit der entsprechenden Potenz von 16 multipliziert. Die Summe der einzelnen Produkte ergibt den Dezimalwert der Zahl.

Beispiel: Die Zahl (hex) 36F soll in eine Dezimalzahl umgerechnet werden:

```
              2     1     0
36F = 3 * 16 + 6 * 16 + 15 * 16

36F = 3*256 + 6*16 + 15*1 = (dez) 879
```

Die Umrechnung einer gebrochenen Dezimalzahl in eine Hexadezimalzahl erfolgt durch Multiplizieren mit 16. Die Ziffer links des Dezimalpunkts wird als erste hexadezimale Ziffer aufgeschrieben. Die Stellen nach dem Dezimalpunkt des Produkts werden daraufhin erneut mit 16 multipliziert. Die links vom Dezimalpunkt erscheinende Ziffer ergibt die zweite Ziffer der hexadezimalen Zahl usw. Sobald ein Produkt eine ganze Zahl ergibt, ist die Umrechnung beendet.

Beispiel: (dez) 0,43 = (hex)?

```
0,43 * 16 Ergebnis (hex)   0, 6 E 1 4 7
  4 3
  5 28
  6,88 ----------------6

  0,88 * 16
    8 8
    5 28
   14,08 ---------------E

    0,08 * 16
      0 8
      48
     1,28 ---------------1

      0,28 * 16
        2 8
        1 68
        4,48 ---------------4

        0,48 * 16
          4 8
          2 88
          7,68 ---------------7
```

Mit einer Genauigkeit von hier fünf Stellen entspricht der dezimalen Zahl 0,43 die Hexadezimalzahl 0,6E147.

Günstigerweise werden der ganzzahlige Anteil einer Dezimalzahl und der gebrochene getrennt umgeformt, also zunächst die Stellen vor dem Komma, dann erst die nach dem Komma.

Bei Zusammenfassung der beiden oben gezeigten Beispiele ergäbe also

```
(dez) 246,43
(dez) 246     =   (hex) F6
(dez) 0,43    =   0,6E147

(dez) 246,43  =   F6,6E147
```

Das Hexadezimalsystem ist im Prinzip nicht schwieriger zu handhaben, als das Dezimalsystem, – wäre man letzteres nicht gewohnt und hätte man nicht lediglich zehn Finger zum „Nachrechnen". Hilfreich zum Verständnis aller hier angesprochenen Zahlensysteme ist deren gleiches Prinzip:

- es werden 2, 10 oder 16 Zeichen bestimmter Wertigkeiten verwendet;
- in allen drei Systemen zeigt die Ziffer „0", daß die Wertigkeit dieser Stelle nicht zum Gesamtwert zu addieren ist;
- die Wertigkeit einer Stelle berechnet sich immer als Produkt aus dort notierter Ziffer und der Basis, potenziert mit der Platznummer (von rechts mit 0 beginnend).

```
              3  2  1  0
(bin)    ... 2  2  2  2

              3  2  1  0
(dez)    ... 10 10 10 10

              3  2  1  0
(hex)    ... 16 16 16 16
```

- Basis und Anzahl der verschiedenen Ziffern eines Systems sind immer gleich innerhalb eines Systems.

Umständlich ist ein Umrechnen immer, aber wenn man sich ein wenig mit dem Hexadezimalsystem beschäftigt und vielleicht auch in Assembler programmieren will, werden Zahlen wie FF oder FE vertraut. *Dipl. Ing. (FH) Oliver Rosenbaum*■

VIDEORECORDER UND 128er GEHT DAS?

Seit einiger Zeit versuche ich erfolglos, ein vernünftiges Bild von meinem Commodore 128 auf einen Videorecorder zu bekommen.
Problematisch ist das allein schon durch die unterschiedlichen Ein- beziehungsweise Ausgänge an den Geräten. Auch Selbstgelötetes, nach Tips eines Bekannten, half bisher nicht weiter.
Die Auskunft eines Verkäufers, der C128 besitze keinen Fernsehbild-Generator, hat mich weiter entmutigt. Andererseits gibt es für diesen Computer Video-Animationsprogramme, die wohl nur einen Sinn haben, wenn es möglich ist, das laufende Programm auf einen Recorder zu bekommen.
Kurz gesagt: Ich bitte um einen aufklärenden Tip oder Trick, wie ich das mir bisher Unmögliche vielleicht doch noch erreichen kann.
Michael Liebig
6053 Obertshausen

Natürlich können Sie Ihren 128er an einen Videorecorder anschließen. Die Aussage des Verkäufers, der 128er hätte keinen Fernsehbild-Generator, müssen wir, bayrisch ausgedrückt, als „Schmarrn" zurückweisen.
Die einfachste Art, ein Bild auf dem Videorecorder zu bekommen, ist der Anschluß über das mitgelieferte Antennenkabel. Sollten Sie dieses nicht mehr besitzen, so nehmen Sie ein 75-Ohm-Kabel, das auf der einen Seite einen koaxial Cinch-Stekker und auf der anderen einen TV-Normstecker besitzt. Stecken Sie nun den TV-Normstecker in die dafür vorgesehene Buchse Ihres Videorecorders (Antenneneingang). Dazu müssen Sie Ihre Hausantenne ausstecken. Suchen Sie sich eine nicht belegte Programmtaste und stellen auf ihr den Kanal 36 UHF ein. Ihr Fernsehgerät bleibt natürlich angeschlossen. Sollte Ihr Videorecorder das Programm nicht automatisch „durchschleifen", so müssen Sie die Aufnahmetaste drücken, ohne daß Sie das Band starten. Es kann zu kleinen Problemen bei der Einstellung des Kanals kommen, da der Kanal 36 oft etwas abweicht. Drehen Sie deshalb solange an der Kanaleinstellung, bis das Computereinschaltbild auf Ihrem Fernsehbildschirm erscheint. Allerdings können Sie auf diese Art nur Bilder im 40-Zeichen-Modus empfangen. Stellen Sie also Ihren Computer darauf ein.
Eine andere Möglichkeit bietet der RGB-Ausgang an Ihrem 128er. Dabei kommt es auf die Anschlüsse Ihres Videorecorders an. Die meisten Recorder haben einen speziellen Videoeingang, an dem Sie alle Peripheriegeräte anschließen können, wie zum Beispiel Videokameras und Computer. Dazu vergleichen Sie bitte die Anschlüsse des Recorders und des Computers in dem jeweiligen Handbuch.

NLQ-AUSDRUCK MIT STAR LC-10C

Zum in Ihrem Heft COMMODORE WELT 128 SPECIAL, Ausgabe 5/87 dargestellten Schreibmaschinenprogramm auf den Seiten 88 bis 93 habe ich eine Frage:
Wie kann man sich das Geschriebene auf den Drucker Star LC-10C mit NLQ ausdrucken lassen (Zeile 1340 bis 1440)?
Ulrich Lohse
5000 Köln 91

Leider haben wir zu diesem Zeitpunkt keinen Star LC-10C zur Verfügung, so daß wir das von Ihnen angesprochene Problem nicht genau testen konnten. Aber normalerweise müßte sich Ihr Drucker hardwaremäßig umstellen lassen. Das heißt, mit den Dipschaltern an Ihrem Drucker sollte sich eigentlich das Erwünschte erreichen lassen. Wollen Sie aber den NLQ-Ausdruck softwaremäßig ansteuern, so schauen Sie bitte in Ihrem Handbuch nach. Bei den meisten NLQ-Druckern lautet der Befehl zum Umschalten CHR$(27)"x1".

SUPERBASE UND BUCHHALTUNG, ABER WO?

Ich bin seit zirka einem Jahr Besitzer eines Commodore 128 und habe kürzlich ein Exemplar Ihrer Zeitschrift gekauft.
Nun habe ich eine Frage: Was haben Sie schon über Superbase gebracht?
Ich möchte zwei kleine Buchhaltungen für mich aufbauen. Gibt es da Beispiele? Haben Sie etwas veröffentlicht? Ist das erhältlich? Wie ist der Preis?
Werner Rüsch
CH-2552 Orpund

Wir können Ihnen helfen. Ja, wir hatten einen kleinen Bericht über Superbase in der normalen Ausgabe von COMMODORE WELT 2/88, Seite 15.
Auch können wir Ihnen zwei Buchhaltungsprogramme anbieten. Das erste nennt sich Calcu und ist ein kleines einfaches Programm und erschien in der 128/C64 SPECIAL Nummer zwei auf Seite 90. Etwas professioneller ist das Programm Buchhaltung. Es wurde in der Zeitschrift C64/128 Softwarejahrbuch 1988 „Das Beste aus CW" auf der Seite 87 veröffentlicht. Sollten diese Hefte nicht mehr bei Ihrem Zeitschriftenhändler erhältlich sein, so können Sie selbstverständlich diese Ausgaben direkt bei uns bestellen.

MONOPOLY UND ANDERE GESCHICHTEN

Ich habe mit einigen Programmen aus verschiedenen 128er-CW-Sonderheften erhebliche Schwierigkeiten. Diese betreffen die Programme:
– Monopoly aus dem Sonderheft Nr. 3. Bei diesem Programm lautet die Zeile 3470 's$-rr$+chr$(32)+oo$:printchr$(27)+"m" ', die in dieser Weise natürlich einen Syntax-Fehler liefert.
– Programmdatei aus dem Sonderheft Nr. 7.
Bei diesem Programm ist es mir nicht möglich, Daten von Diskette zu laden. Die Floppy (1570) läuft an, hört aber nicht wieder auf zu laden. Tippfehler sind aufgrund des Checksummers auszuschließen. Es wäre im übrigen sehr wünschenswert, wenn Sie Korrekturen, Druckfehler und ähnliches aus den 128er-Sonderheften auch in den nachfolgenden 128er-Sonderheften veröffentlichen würden und nicht nur in den anderen Zeitschriften der CW. Denn viele 128er-User kaufen sich nur die 128er-Sonderhefte, da in den anderen Zeitschriften der CW viel zu wenig Material für den 128er zu finden ist.
Achim Kramer
2000 Hamburg

Tja, wieder einmal hat unser Listingsdrucker gestreikt. Die Zeile in dem Monopoly-Listing muß natürlich folgendermaßen lauten:
s$=rr$+chr$(32)+oo$
:printchr$(27)+"m"
Bei dem Programm „Programmdatei" trat der Ladefehler nur auf, wenn Sie bei der Eingabe von Daten einen oder mehrere Datenfelder mit Return abschlossen, ohne Daten eingegeben zu haben. Das Problem dabei ist, wie bei allen Commodore-Homecomputern, daß bei einer sequentiellen Datenverarbeitung die Leerräume nicht ohne besonderen Abspeicherbefehl erkannt werden. Somit „weiß" der Computer beim Wiederladen der Daten nicht, wenn das File zu Ende ist. Fügen Sie also in dem Programm folgende Zeile ein:
2985 ifa$(h,y)=" "then
print#1," ";

DRUCK- UND PROGRAMMFEHLER

In Ihren beiden Zeitschriften COMMODORE WELT/ SPECIAL Nr. 2/88 und Nr. 6/88 scheinen meiner Meinung nach Tippfehler zu sein!
Bei Ihrem BASIC 7.0 LINKER (Heft 2/88 Seite 29 bis 33) müßte ein Druckfehler sein. Es erscheinen die Anführungsstriche nicht. Warum???
In Ihrem Heft 6/88 haben Sie bei dem Programm „Tellurium", Seite 78, Zeile 1390 bis 1420 Zeichen ausgelassen!
Die Checksumme konnte nicht erreicht werden. Das Programm unterbricht an dieser Stelle.

Rolf-Peter Bily
7835 Teningen 1

Zuerst einmal zu dem Programm BASIC 7.0 LINKER. Nach einem Testdurchlauf mit Ihrem Programmbeispiel hatten auch wir die gleichen Probleme. Es scheint wirklich ein Programmfehler vorzuliegen. Wir haben nun folgende Zeilen geändert:

```
1100 if a = 34 then f =
        xor(f,255)
1101 if a = 32 and f = 0
        then 1090:elseb$
        =b$+chr$(a)
```

Danach lief das Programm einwandfrei.
Bei Ihrem zweiten Problem, betreffend das Programm „Tellurium", müssen wir leider bedauern, daß in den von Ihnen beschriebenen Zeilen wirklich Zeichen fehlen.
Wir haben uns das Programm noch einmal angeschaut und siehe da, in den Leerräumen der Zeilen 1390 bis 1420 müßte eigentlich ein griechisches Pi stehen.
Sie bekommen dieses Zeichen durch gleichzeitiges Drücken der Commodore- und der Pfeil-nach-oben-Taste. Zur Kontrolle nochmals folgende Zeilen:

```
1390 a=sin(i*π/180)
1400 b=cos(i*π/180)
1410 c=sin(i*π/180)
1420 d=cos(i*π/180)
```

NUR BEDINGT LAUFFÄHIG: 128er-PROGRAMM AUF DEM C64

Ich bin seit einiger Zeit stolzer Besitzer eines C64 mit Floppy 1541. Nun habe ich vor einigen Tagen ein 128/C64 SPECIAL-Heft aus Ihrem Verlag erworben. Das Heft gefiel mir sehr gut, besonders die Programme für den C64. Aber leider versuchte ich vergeblich Programme, die für den C128 bestimmt sind, in meinen Rechner zu laden und sie zu starten. Nun meine Frage:
Gibt es eine Möglichkeit, jene Programme auch auf dem C64 zu benutzen?
Martin Hauser
8000 München

Leider müssen wir Sie enttäuschen. Die Programme für den 128er sind nur bedingt auf dem C64 lauffähig. Im Prinzip können Sie zwar jedes reine BASIC-Programm eines C128 in den C64 laden und starten, so lange es nicht größer als der freie BASIC-Speicher ist (38 KByte). Aber auch nur dann, wenn in diesem Programm nur Befehle des BASIC 2.0 verwendet wurden, also keine Anweisung des BASIC 7.0 (zum Beispiel SCNCLR, GRAPHIC, BOX usw,). Die versteht der C64 nicht und meldet sich mit einem „Syntax Error". Außerdem gibt es noch das Problem der BASIC-Zeilenlänge, die beim C128 160 Zeichen, beim C64 aber nur 80 Zeichen betragen darf.
Schon da könnten zum Beispiel beim Listen Fehler auftreten, weil sich der viel unkomfortablere Bildschirmeditor des C64 nicht mehr auskennt.
Falls Sie ein Maschinensprache-Programm des C128 im C64 laufen lassen möchten, müssen umfangreiche Speicherbereichs- und Adressenänderungen vorgenommen werden.

NACHTRAG ZU MUSIK TUTOR COMMODORE WELT 128 SPECIAL NR. 6/88

Bei dem im Heft Nr. 6/88 abgedruckten Programm „Musik Tutor" wurden im Unterprogramm MTTP III auf Seite 70, folgende Zeilen vergessen.

```
60060 zf$=chr$(176):
      zg$=chr$(177)
60070 zh$=chr$(178):
      zj$=chr$(180)
60080 zm$=chr$(183):
      ym$=chr$(219)
60090 rem********
      Zeichenfolge *
60100 for q = 1 to 40
60110 qd$ = qd$ + c4$
60120 next q
60130 return
```

Weiterhin bekamen wir einige Anfragen, was denn das tiefgesetzte Minuszeichen vor dem Wort „note" im Unterprogramm MITTP V auf Seite 74, Zeile 560 zu bedeuten habe.
Leider hat uns der Listingdrucker einen Streich gespielt. Das tiefgesetzte Minuszeichen ist nichts weiter, als ein PFEIL NACH LINKS. Bitte korrigieren Sie dieses in Ihrem Programm. □

Impressum 64/128 Special

C64/128 SPECIAL erscheint zweimonatlich in der CA-Verlags GmbH, einer Gesellschaft der Aktuell-Gruppe

REDAKTION UND STÄNDIGE MITARBEITER:
Peter Basch, Rosemarie Huber, Lothar Miedel, Michael Reppisch, Rudolf Schmid-Fabian, Torsten Seibt, Hermann Wellesen, Bernd Welte

ANZEIGENVERWALTUNG:
ADV-Mediendienste, Aindlingerstraße 17–19, 8900 Augsburg 1
Tel.: (0821)7904-227
Telekopierer: (0821)7904-243
Telex: adv 533502
Teletex: 821887

VERANTWORTLICH FÜR DEN ANZEIGENINHALT:
Brigitte Kostić

ANSCHRIFT FÜR ALLE VERANTWORTLICHEN:
Postfach 1161, 8044 Unterschleißheim,
Tel.: 089/1298011
Telex: 5214428 cav-d
Es gilt Preisliste Nr. 9 vom 1.6.1988
Media-Unterlagen bitte anfordern.

VERANTWORTLICH FÜR DEN INHALT:
Andreas Greil

GESCHÄFTSFÜHRER:
Werner E. Seibt

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

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

TIPS ZU EASY-SCRIPT

Texte rechtsbündig ausdrucken

Mit EASY-SCRIPT kann man auf einfache Weise beliebige Texte erstellen. Besonders schön werden diese mit der Formatanweisung „jul" (justification ein). Diese bewirkt, daß alle Zeilen gleich lang werden. Fehlende Buchstaben werden durch Leerzeichen zwischen den Worten aufgefüllt. Damit dabei nicht zu große Lücken entstehen, muß am Zeilenende richtig getrennt werden. Hierzu hat EASY-SCRIPT die Möglichkeit, sogenannte „soft hyphen" (weiche Trennungszeichen) einzufügen. Dies geschieht mit <F1><–>. Steht ein solches Zeichen im Wort, so wird an dieser Stelle getrennt, wenn sie am Zeilenende zu stehen kommt. Der Nachteil liegt darin, daß nur ein solches Trennungszeichen eingesetzt werden kann. Mehrere Trennvorschläge schaden zwar nichts, da sie normalerweise nicht ausgedruckt werden, aber es wird prinzipiell das erste verwendet, auch wenn ein anderes eine günstigere Trennung ergäbe. Soll ein wichtiger Text im Blocksatz (rechtsbündig) gedruckt werden, ist es unumgänglich, die Trennungen von Hand einzufügen. Dazu muß die Zeilenlänge im Eingangsmenü auf die gewünschte Länge eingestellt werden und nach jeder Silbentrennung der Abschnitt neu formatiert werden. Dies ist sehr aufwendig. Daher haben wir Ihnen ein Hilfsprogramm geschrieben („SILBENTRENNUNG"), mit dem die optimale Trennung auf einfache Weise möglich ist. Das Programm ist auf Diskettenbetrieb ausgelegt, kann aber leicht mit Hilfe des Handbuchs umgeschrieben werden. Der mit EASY-SCRIPT (oder einem anderen Programm, das mit ASCII-Zeichen arbeitet) erstellte Text wird in den Speicher geladen und Zeile für Zeile abgearbeitet. Immer wenn ein Leerzeichen oder Trennungszeichen (auch „soft hyphen") gefunden wird, wird überprüft, ob die erforderliche Zeilenlänge erreicht ist. Bei einer zu langen Zeile wird diese am Schirm angezeigt und mit Hilfe der Cursortasten (links und rechts) das Trennungszeichen an die gewünschte Position angebracht. Mit RETURN wird die Zeile abgeschlossen und das Programm fügt automatisch den RETURN-Code 13 in den Text ein.

PROGRAMM-ERKLÄRUNG

Zeile 100: Der Speicher wird begrenzt, um einen geschützten Textspeicher zu schaffen.
Zeile 160–180: Die Zeilenparameter und der Filename werden abgefragt. Die Variable DM gibt an, wieviele Zeichen noch bis zum Zeilenende fehlen dürfen, damit das Programm automatisch ein RETURN setzt.
Zeile 200–250: Einlesen des ASCII-Textes in den Speicher. Die PRINT-Anweisung in 230 kann weggelassen werden (schneller).
Zeile 280: Das alte Textfile wird als Sicherheitskopie umbenannt. Hier kann der RENAME-Befehl nicht verwendet werden, da dieser keine Variablen verarbeitet.
Zeile 300: SEQ-File zum Schreiben öffnen.
Zeile 330–370: Überprüfung, ob ein Trennungszeichen gefunden wurde.
Zeile 380–390: Prüfung auf Zeilenende. Ist die maximale Zeilenlänge erreicht und das letzte Trennungszeichen zu weit

Silbentrennung

```
10 rem silbentrennung ==========64 <je>
20 rem (p) commodore welt team     <ho>
30 rem ========================== <ng>
40 rem (c) by  r. schmid-fabian    <jp>
50 rem              heidelberg     <kp>
60 rem                             <ah>
70 rem basic v2.0                  <nc>
80 rem c64/c128     floppy         <ml>
90 rem ========================== <jg>
100 poke55,0:poke56,39:clr         <bi>
110 l$=" ":b$="-":rem trennzeichen <le>
120 sh$=chr$(192):rem soft-trennze
ichen                              <li>
130 cr$=chr$(013):cl$=chr$(147)    <bm>
140 li$=chr$(157):re$=chr$(29)     <el>
150 ro$=chr$(146):rv$=chr$(18)     <pe>
160 input"max. Zeilenlaenge";ml    <df>
170 input"max. Abweichung";dm      <ae>
180 input"file-Name";p$            <nn>
190 rem ****** einlesen ********** <la>
200 open2,8,2,p$+",s,r"            <md>
210 fori=12000to60000              <ca>
220 get#2,a$:pokei,asc(a$):printa$
;                                  <kl>
230 if st=0 then next              <md>
240 close2:hlt=i                   <hf>
250 rem **** rename v. p$  ******* <gl>
260 printchr$(147)"Jetzt wird "+ch
r$(34)+p$+chr$(34)+" umbenannt (Ta
ste)":getkeya$                     <ja>
270 open1,8,15,"r:"+left$(p$+"
       ",12)+".bak+"+"="+p$:close
1                                  <hj>
280 rem ****** bearbeiten ******** <am>
290 open2,8,2,p$+",s,w"            <bb>
300 for i=12000 to hlt             <mp>
310 a$=chr$(peek(i))               <jj>
320 if a$=sh$ then tp=zl:en$=b$:go
to370                              <oj>
330 if a$=cr$ then printrv$w$:prin
t#2,w$:zl=0:w$="":goto430          <pf>
340 w$=w$+a$:zl=zl+1               <kc>
350 if a$=l$ then tp=zl:en$=""     <mf>
360 if a$=b$ then tp=zl:en$=b$     <og>
370 if zl<ml then 430              <bj>
380 if(ml-tp)>dm then gosub 470    <pg>
390 pr$=left$(w$,tp)+en$:printrv$p
r$                                 <ij>
400 print#2,pr$                    <ab>
410 w$=right$(w$,ml-tp)            <ch>
420 tp=0:zl=len(w$)                <de>
430 next                           <ob>
440 close2:end                     <ff>
450 rem ****** trennung ********** <kl>
460 rem ****** von hand ********** <bm>
```

```
470 d=0:x$="":forv=i+1toi+6:x$=x$+
chr$(peek(v)):nextv                    <ek>
480 if asc(x$)=32 then i=i+1:tp=le
n(w$):en$="":return                    <fd>
490 x$=rv$+x$                          <di>
500 pr$=right$(w$,ml):z$=left$(pr$
,ml-d):t$=right$(pr$,d)                <aa>
510 en$=b$:p=peek(i-d):if p=32 the
n en$=l$                               <hd>
520 printchr$(147)+z$+rv$+en$+ro$+
t$+x$                                  <oc>
530 getkeyq$                           <pi>
540 if q$=li$ then d=d+1               <cp>
550 if q$=re$ then d=d-1:if d<0 th
en d=0                                 <cj>
560 if q$=cr$ then tp=len(w$)-d:en
$="-":printcl$:return                  <da>
570 goto500                            <gn>
580 rem ========================== <oa>
590 rem  p r o g r a m m e n d e   <kl>
600 rem ========================== <km>
```

seq to viza-seq

```
10 rem ========================== <jf>
20 rem seq to viza-seq ========c64 <fd>
30 rem ========================== <ng>
100 poke55,0:poke56,40:clr             <po>
110 input"programmname (seq)";p$       <bp>
120 open2,8,2,p$+",s,r":i=10500        <kd>
130 get#2,a$:a=asc(a$)                 <bg>
140 if a=34 then a=98:rem hochkomm
a                                      <nl>
150 pokei,a:printa$;:i=i+1:if st=0
 then 130                              <bh>
160 close2:hlt=i                       <if>
170 print:input"programmname neu";
p$                                     <ek>
180 open2,8,2,p$+",s,w"                <ab>
190 fori=10500tohlt:a$=chr$(peek(i
)):print#2,a$;:next:print#2:close2 <jd>
200 rem ========================== <ne>
210 rem  p r o g r a m m e n d e   <jl>
220 rem ========================== <hm>
```

entfernt, wird in das Unterprogramm ab Zeile 380 verzweigt.
Zeile 480: Die nächsten fünf Zeichen werden aus dem Speicher geholt, damit man beim Trennen die Fortsetzung des Wortes sehen kann.
Zeile 490: Ist das Folgezeichen ein Leerzeichen, so wird die Zeile selbständig abgeschlossen.
Zeile 500–550: Die überlange Zeile wird angezeigt. Der Wort- oder Satzteil, der über die maximale Zeilenlänge hinausgeht, erscheint revers.

TIPS ZU VIZA-WRITE

Wie die meisten Textverarbeitungsprogramme hat auch VIZA-WRITE die Möglichkeit, sequentielle Files einzuladen. Dies ist sehr praktisch, wenn man Programmlistings verarbeiten will. Zum Beispiel ist es sehr nützlich, wenn beim Schreiben einer Programmanleitung diese in den Text eingebaut werden kann. Zuerst muß dazu das Listing in ein sequentielles File umgewandelt werden:

```
OPEN,8,8,"name,S,W"
      :CMD8:LIST
PRINT#8:CLOSE8
```

Um ein sequentielles File einzuladen, muß man einen kleinen Trick anwenden, da ein direktes Laden von SEQ-Files mit „WRONG FILE TYP" kommentiert wird. Zuerst wird ein beliebiges Textfile eröffnet. Dann kann das SEQ-File mit dem Merge-Befehl dazugeladen werden (Commodore-Taste/Shift M). Statt der Seitenzahlen (Start beziehungsweise Stop) gibt man einfach „s" ein. Jetzt gibt es nur noch ein Problem: Bei der Umwandlung von ASCII-Zeichen aus dem SEQ-File in VIZA-WRITE-Code (Bildschirmcode) werden Anführungszeichen (CHR$ (34)) nicht umgewandelt. Dies hat zur Folge, daß alle Anführungszeichen im Programmtext verschwinden. Daher haben wir ein kleines Hilfsprogramm geschrieben (SEQ TO VIZA-SEQ), das lediglich alle CHR$-Codes 34 in 98 umwandelt. Danach kann das Listing,wie oben beschrieben, problemlos eingelesen werden. □

Nachtrag zum letzten 128 Special

In der letzten Ausgabe der Commodore Welt 128/C64 SPECIAL wurde auf den Seiten 14 und 15 Turbo Pascal für den C128 unter dem CP/M-Betriebssystem vorgestellt. In dieser Ausgabe folgen nun die Pascal-Programme. □

```
Program Programm1;
Begin
   Clrscr;
   Writeln ('Hallo')
End.
Program Code;
 Var name : String[10];
Begin
   Clrscr;
   Gotoxy (10,10);
   Writeln ('Bitte Name eingeben');
   Readln (name);
   Gotoxy (10,20);
   Writeln ('Dein Name ist ',name)
End.
Program Programm3;
   Var alf,pause : Integer;
Begin
   Clrscr;
   Writeln;
   Write ('Das 3.Programm');
   Writeln;
   Write ('Von 1 nach 100 und zurueck');
   For pause:=1 to 30000 do;
   Writeln;
    For alf:=1 to 100 do
    Writeln (' Wert = ',alf);
    For alf:=100 downto 1 do
    Writeln ('Wert = ',alf)
End.
```

2500 PROGRAMMIERSPRACHEN ZUR AUSWAHL

Wie in Babylon

Der vielschichtige, schnell wachsende Markt der Programmiersprachen und ihrer Dialekte ist auch für Insider kaum noch überschaubar. Das komplexe Thema ist nicht nur für den Profi-Programmierer, sondern auch für jene geeignet, die sich einen Überblick dieser babylonischen Sprachenverwirrung verschaffen wollen. Wir beenden unsere Serie mit neunzehn weiteren Programmiersprachen — von OCCAM bis V.I.P.

49. OCCAM

Algorithmische Sprache. OCCAM entstand als Ergänzung zu dem Transputer, einem Microprozessor, der für Computer mit mehreren parallel arbeitenden CPUs (Zentraleinheiten) bestimmt ist.
Im Gegensatz zu Pascal ist die Möglichkeit, eine Aufgabe in parallellaufende Prozesse zu untergliedern, in OCCAM bereits integriert. Pascal mußte hierzu erst erweitert werden (Concurrent Pascal).
OCCAM ist auch für neuere Generationen von Transputern geeignet. Während bei den Transputern der ersten Entwicklungsphasen (mit herkömmlichen Prozessoren bestückt, ein Teil der Rechenleistung der Prozessoren zur Organisation und Kommunikation zwischen diesen gebraucht wurde, kommen bei den Weiterentwicklungen spezielle Prozessoren zum Einsatz, die hierfür besser geeignet sind.
Hundert herkömmliche CPUs konnten sinngemäß niemals die hundertfache Leistung einer einzelnen erbringen, trotz der Programmiersprache OCCAM, die beide Systemkonzepte gleichermaßen unterstützt. Die volle Leistungsfähigkeit kann OCCAM bei den neuen Systemen unter Beweis stellen.
Haupteinsatzgebiet von OCCAM (und auch das der Transputer) sind technisch-wissenschaftliche Applikationen wie die Konstruktion von Filterschaltungen oder schnelle Kurvenberechnung (Fast Fourier Transformation). Auch Echtzeitgrafiken, welche naturgemäß sehr rechenintensiv sind, können hiermit realisiert werden.

50. Pascal

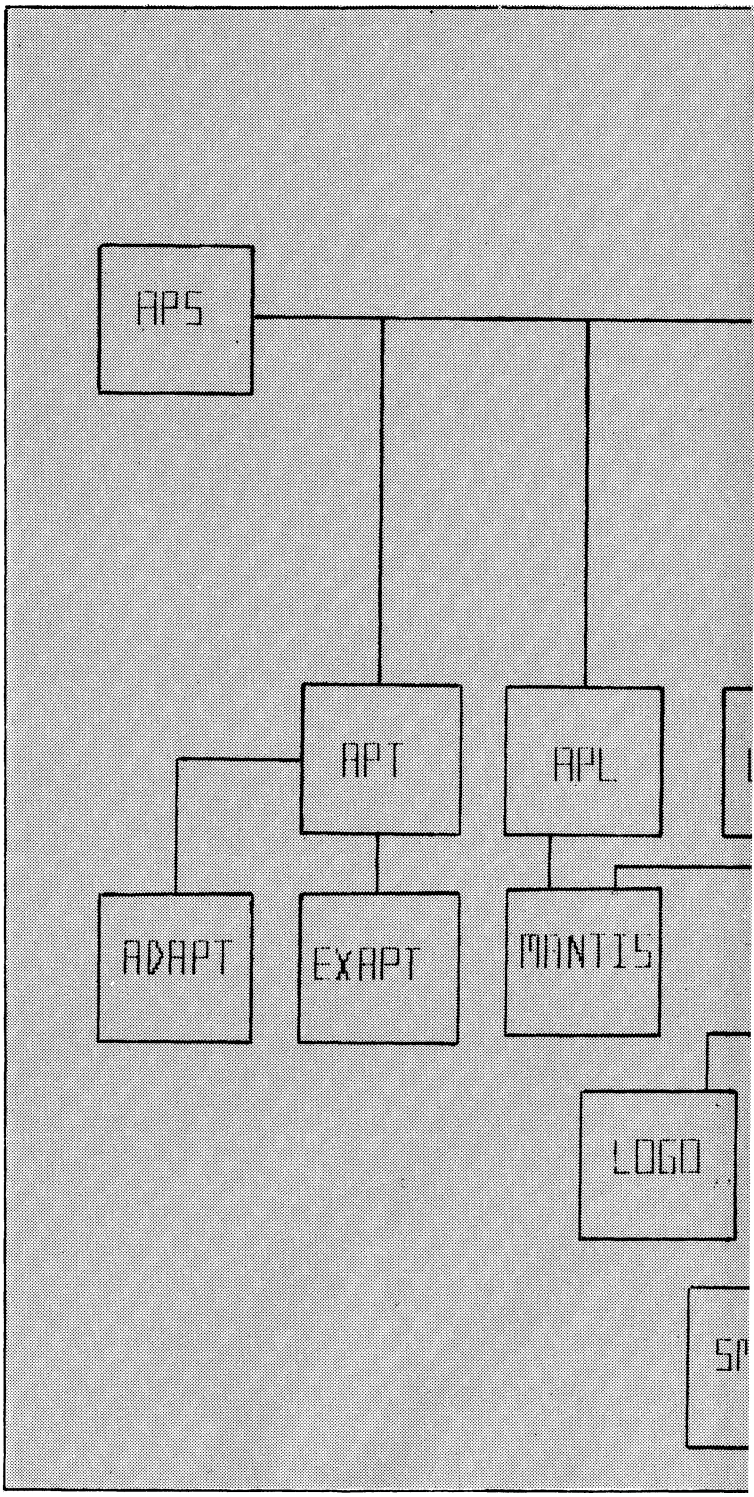
Benannt nach dem Wissenschaftler Blaise Pascal (1623 – 1663).
Pascal ist eine starke logisch strukturierte Programmiersprache, die hervorragende Hilfestellungen beim Programmieren umfangreicherer Probleme gibt.
Auf dem Markt ist Pascal seit etwa 1969.
Als Nachteil wäre zu nennen, daß Pascal für das Erstellen kleinerer Programme zu aufwendig ist.
Pascal ist eine der wenigen Sprachen, die Mengen (aus der Mengelehre) sinnvoll direkt verarbeiten kann, ohne den Programmaufwand wesentlich zu erhöhen.
Es gibt zwischenzeitlich mehrere Versionen von Pascal:

- Pascal (Jensen/Wirth-Standard),
- Pascal/Z,
- Pascal/MT+,
- UCSD-Pascal *(siehe unten)*,
- TINY Pascal,
- Concurrent Pascal (für Parallelverarbeitung),
- Pascal/M*,
- Pascal-XT (Extended Pascal),
- Pascal-SC (Pascal for Scientific Computation),
- TURBO-Pascal
- und andere.

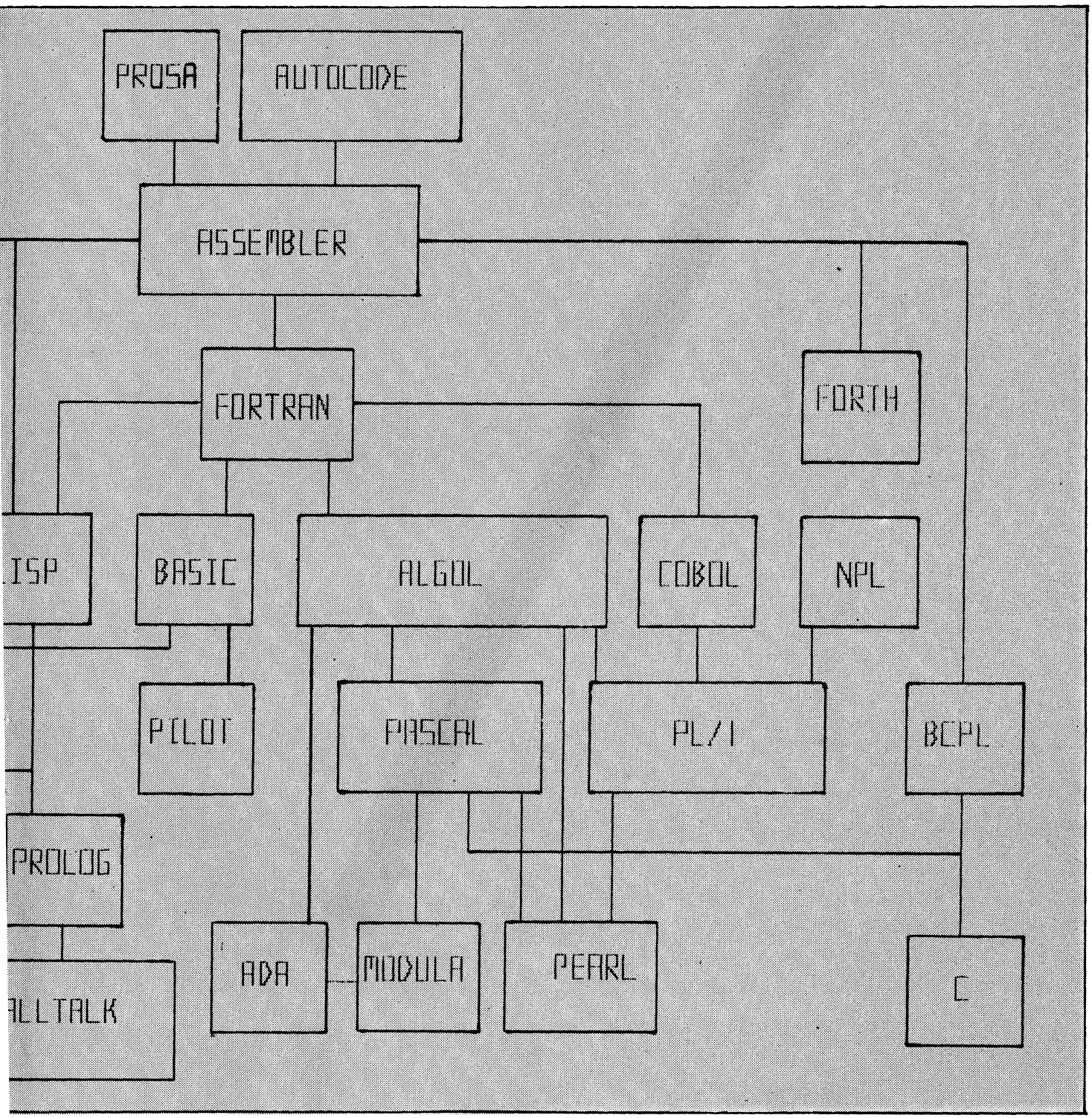
Letztere Version dürfte wegen ihrer ausgezeichneten Verarbeitungsgeschwindigkeit die Verbreitung von Pascal wesentlich beeinflußt haben.
(Programmbeispiel siehe Tabelle 15.)

UCSD-Pascal System

Das UCSD-Pascal wurde an der University of California in San Diego entwickelt, daher auch sein



Name. Mittlerweile gibt es UCSD-Pascal für fast alle Rechnertypen, so daß die Software in Pascal sehr portabel ist.
Der Standard des UCSD-Pascal wird hier einmal anhand seines Betriebssystems dargestellt, da es sich um mehr als einen einfachen Interpreter oder Compiler handelt.
UCSD-Pascal besteht aus mehreren Files (auf Diskette). Die Aufteilung in mehrere Files ist wegen des Umfangs einzelner Leistungsabschnitte notwendig. Bei Bedarf wird auf diese Files zugegriffen, daher müssen sie im Falle des Aufrufs durch den Anwender verfügbar sein; die Diskette muß sich im Laufwerk befinden. So ist beispielsweise das File mit dem Namen SYSTEM.APPLE notwendig, um das System zu booten (hier im Beispiel auf einem Apple-Computer).
Alle Systemfiles beginnen mit dem Namen SYSTEM:
SYSTEM.PASCAL
SYSTEM.CHARSET
SYSTEM.APPLE
SYSTEM.EDITOR
SYSTEM.COMPILER
SYSTEM.LIBRARY
und so weiter.
Die Namen lassen oft schon ahnen, welche Files sie bezeichnen. Dem Anwender ist es nicht möglich, Files anzulegen, die ebenfalls mit SYSTEM beginnen, damit sind die systemeigenen Files deutlich von den Anwenderfiles abgegrenzt.



Die Struktur des UCDS-Systems ist klar gegliedert in Systemeinheiten wie Editor, Compiler, Interpreter, Linker, Assembler und Filer. Die Systemkomponenten müssen also je nach Bedarf in den Rechner geladen werden. Zum Programmieren muß zunächst ein File erstellt werden, in welches mit Hilfe des Editors das Programm geschrieben werden kann. Zum Übersetzen muß dann der Compiler geladen werden.
Die Arbeit mit dem System findet also auf verschiedenen Ebenen statt. Das UCSD-System kennt drei verschiedene Filetypen zur Diskettenorganisation:
Text-Files,
Data-Filex,
Code-Files.
Der entsprechende Filetype muß bei der Namensgebung des Files mitangegeben werden, beispielsweise NAME.TEXT oder INHALT.DATA. Textfiles sind solche Dateien, die mit dem Editor beschrieben werden können (etwa Programme in Pascal). Dieses File kann abgedruckt (Programmlisting) oder compiliert werden. Der Compiler bearbeitet den Inhalt des angegebenen Textfiles und speichert das Ergebnis der Übersetzung in einem Codefile ab. Dieses enthält also den ausführbaren Code. Datenfiles enthalten, wie ihr Name schon sagt, Daten, welche in einem Programm abgespeichert oder von diesem gelesen werden können.
Das Workfile ist das Systemfile (Text oder Code), auf das immer automatisch zugegriffen wird. Der Editor oder der Compiler öffnet – sofern nichts anderes angegeben wird – immer ein Input-File und ein Output-File. Befindet sich kein Workfile auf der Diskette und wurde auch kein anderes File angegeben, so erscheint eine Fehlermeldung.

Tabelle 15:

```
Programmbeispiel zu PASCAL
Listing
-------------------------------------------------------------

PROGRAMM SCHEIBEN (INPUT, OUTPUT);
VAR I: INTEGER
   LINKS, MITTE, RECHTS: CHAR;

PROCEDURE VERLAENGERE (I:INTEGER; LINKS, RECHTS, MITTE: CHAR);
BEGIN
  IF I <> 0 THEN
  BEGIN
    VERLAENGERE (I-1, LINKS, MITTE, RECHTS);
    WRITELN ('VON', LINKS, 'NACH', RECHTS);
    VERLAENGERE (I-1, MITTE, RECHTS, LINKS);
  END;
END;

BEGIN                          (* HAUPTPROGRAMM *)
  I:= 3                        (* = ANZAHL DER SCHEIBEN *)
  LINKS:= 'A';
  MITTE:= 'B';
  RECHTS:= 'C';
  VERLAENGERE (I, LINKS, RECHTS, MITTE);
END.
```

Tabelle 16:

```
Beispiel zu PEARL Echtzeitsprache
Listing
-------------------------------------------------------------
AFTER 5 SEC ALL 10 SEC          Der Rechenprozeß RELAIS wird nach 5
DURING 100 MIN ACTIVATE         Sekunden, während 100 Minuten, im 7
RELAIS PRIORITZ 1               Sekundenzyklus, mit Priorität 1 belegt.
-------------------------------------------------------------
```

Im Directory-Beispiel sind zwei solcher Workfiles:

SYSTEM.WRK.TEXT
und
SYSTEM.WRK.CODE

Das SYSTEM.WRK.TEXT-File dient dem Editor zugleich als Ein- und Ausgabefile, denn der Editor bearbeitet nur Texte, während er Compiler Texte in Codes umformt. Daher benutzt dieser zwar das gleiche Eingabefile, braucht aber zur Ausgabe ein Code-File: SYSTEM.WRK.CODE.

Eine Diskette darf immer nur zwei Workfiles enthalten (Text und Code).
Nach dem Booten (Laden) des USCD-Systems befindet sich der Rechner in der Kommandoebene, von der aus Direktbefehle gegeben und Dienstprogramme gestartet werden können. Der Rechner erwartet einen Befehl:

COMMAND: Edit, Run, File, Comp, Link, Xecute, Assem, Debug.

Weitere Befehlsmöglichkeiten erfährt man beim Drücken auf die Taste „?". Es erscheint:

COMMAND: User restart, Initialize, Halt, Swap, Make exec.

Diese Befehle sind sogenannte Direktbefehle, also Teil des Betriebssystems, während die Befehle der oberen Zeile Aufrufe für Dienstprogramme sind. Diese müssen erst von der Diskette nachgeladen werden.
Anders als in BASIC bewirkt Run hier nicht lediglich einen Programmstart. Run ist selbst ein Dienstprogramm, welches den Compiler aufruft und startet. Dieser compiliert daraufhin das Workfile. Der Linker wird bei Bedarf automatisch dazugeladen. Das so übersetzte und gebundene Programm wird anschließend noch gestartet.
Run wird also nur für noch nicht compilierte Programme benötigt. Der Befehl, welcher dem BASIC-Run entspricht, wäre hier der „eXecute"-Befehl.
Run erleichtert das Austesten von Programmen, da bei der Fehlersuche nicht jedesmal der Compiler von Hand aufgerufen werden muß.
Neben dieser Möglichkeit kann der Pascal-Compiler auch direkt aufgerufen werden.
Der Compier übersetzt den Inhalt des Workfiles (Text) in direkt ausführbaren Code und speichert diesen in dem Workfile SYSTEM.WRK.CODE ab. Ist eines der beiden Work-

files auf der Diskette nicht vorhanden, oder sollen andere Files benutzt werden, so kann dies optional angegeben werden. Erst compilierte Programme können mit „eXecute" gestartet werden.
Der Pascal-Linker ermöglicht die Einbindung von Assembler-Routinen in ein Pascal-Programm. Diese Routine muß im Pascalprogramm als „external" deklariert sein. External veranlaßt das System, vor Ausführung des Programmes die einzubindende Routine aufzusuchen und zu linken. Der Compiler meldet dies am Bildschirm. Beim Run-Befehl wird der Linker automatisch aufgerufen.
UCSD-Pascal erlaubt auch eine Art Batch-Betrieb: Ein Exec-File, welches mit „M" erzeugt wird, kann Systembefehle enthalten, die bei Start des Exec-Files ausgeführt werden. Dabei ist es gleichgültig, ob es sich um Direktbefehle oder Dienstprogramme handelt. Wie bei manchen anderen Betriebssystemen kann somit ein Batchfile erstellt werden, ein File, das Stapelverarbeitung zuläßt (Batch = Stapel).
Der Name rührt noch aus der Lochkartenzeit. Man hatte die Möglichkeit, Lochkarten mit verschiedenen Direktbefehlen aufeinanderzustapeln und diese gebündelt dem Computer einzugeben. Die Bearbeitung der Befehle erfolgt in der vorgegebenen Reihenfolge. Sprünge oder Verzweigungen sind nicht möglich.
Das UCSD-Pascal-System hat in seiner Struktur große Ähnlichkeit zu verbreiteten Betriebssystemen größerer Rechner. Ein markanterer Vorteil ist aber die Standardisierung für verschiedene Rechner und die Verbindung von Sprache und System zu einer Einheit. Erst dadurch ist es möglich, echte portable Software zu entwickeln. Zwar gibt es inzwischen viele Compiler für Pascal, aber das UCSD-System konnte sich behaupten.

51. Pearl

Process and Experiment Automation Realtime Language.
Wie der Name dieser Programmiersprache schon sagt, ermöglicht sie das Realtime Processing, also die Echtzeit-Verarbeitung: Das Programm läuft in der gleichen Zeit wie ein Prozeß. Der Rechner arbeitet also parallel zu einem technischen Ablauf. Die Daten, die aus diesem technischen Ablauf (Prozeß) gewonnen

ECHTZEIT-VERARBEITUNG

werden, werden simultan verarbeitet. Aufgrund dieser Daten kann der Rechner in den Prozeß in Form einer Korrektur des Prozeß-Ablaufs eingreifen, oder für die Fortführung des Prozesses wichtige Entscheidungen treffen (Prozeßdatenverarbeitung).
Bei der Konzeption von Pearl wurde besonderen Wert darauf gelegt, daß Echtzeit-Sprachelemente in den Sprachkern von Pearl aufgenommen wurden – anders als bei den meisten übrigen höheren Programmiersprachen –, die über solche spezifischen expliziten Sprachmittel nicht verfügen. Meist werden dort beispielsweise in Unterprogrammaufrufen Funktionen des Betriebssystems organisiert.
(Programmbeispiel siehe Tabelle 16.)
Die Zeichenketten-Verarbeitung in Pearl ist unzureichend, obwohl diese Sprache fast ausschließlich auf digitalen Großrechnern zum Einsatz kommt.
Hingegen sind die Strukturen für die Ein- und Ausgabe, sowie die Sprachmittel für Algorithmik ähnlich denen anderer höherer Programmiersprachen. Einflüsse von Algol 68 und Pascal sind augenfällig. Die Syntax ist weitgehendst an PL/1 angepaßt.
Versuche der Implementierung von Pearl auf kleineren Anlagen scheiterten lange an dem für damalige Verhältnisse großen Speicherplatzbedarf.
Inzwischen stehen auch bei kleinen Computern Speicherkapazitäten zur Verfügung, die eine Installation von Pearl zuließen. Jedoch nur für wenige Kleinrechner oder Microcomputer gibt es Implementierungen (zum Beispiel RTOS-Pearl).
Eines der Hauptmerkmale dieser „Prozeßrechnersprache" ist die Trennung in resistente und nichtresistente Prozeduren.
Pearl wurde gemeinsam von deutschen Wissenschaftlern, dem VDI/VDE, industriellen Anwendern, deutschen Prozeßrechnerherstellern und Softwareproduzenten entwickelt und wurde erstmals 1980 in der DIN 66.253, Teil 2: „Informationsverarbeitung; Programmiersprache Pearl, Full Pearl", festgelegt, allerdings in englischer Sprache mit der Absicht, den internationalen Anwenderkreis zu berücksichtigen.
(Programmbeispiel siehe Tabelle 17.)
Ähnlich anderer höherer Programmiersprachen (etwa Cobol) wird ein Pearl-Programm in verschiedene Divisions unterteilt. Diese Einheiten sind unabhängig voneinander compilierbar. Die getrennt übersetzten Programmteile (Module) werden mit sogenannten Global-Größen miteinander verbunden.
Ein fertiges Modul besteht im allgemeinen aus zwei Divisions: der System-Division, welche die Systemumgebung beschreibt und der Problem-Division, der Problemlösung einer bestimmten Aufgabe.
Im System-Teil eines Pearl-Programms werden die peripheren Geräte und deren Verbindungen beschrieben; insbesondere werden Eingangs- und Ausgangskanäle definiert. Es kann hier weiterhin festgelegt werden, ob die Übertragung auf diesen Kanälen (also von oder zu den peripheren Geräten) digital oder analog erfolgen soll. An dieser Stelle werden die Vorteile der Programmaufteilung in Problem- und System-Division am deutlichsten: Sollen bei bestehendem System einzelne Teile der Peripherie gegen verbesserte Gerätetypen ausgetauscht oder ergänzt werden, müssen lediglich einzelne Teile der System-Division geändert werden. Nämlich nur die dieses Gerät beschreibenden Programmzeilen, nicht jedoch alle Befehle im Programm, welche das entsprechende Gerät ansprechen, wie es in manchen anderen Programmiersprachen der Fall sein kann.

SYSTEM- UND PROBLEM-DIVISION

Bei der Übertragung des Pearl-Programms auf andere Rechnertypen oder Anlagenkonfigurationen kann die System-Division komplett ausgetauscht werden, die Problem-Division kann dabei unverändert bleiben.
Hierdurch ist der System-Teil von Pearl durchaus vergleichbar mit einer Job-Control-Language üblicher Compiler.
In der Problem-Division wird das eigentliche Problem formuliert, das mit Hilfe des Programms bewältigt werden soll. Der Problem-Teil besteht aus verschiedenen Tasks, die eine ähnliche Struktur haben können wie Haupt- und Unterprogramme anderer höherer Programmiersprachen. Diese Tasks aktivieren sich gegenseitig in vorgegebener Abfolge oder gleichzeitig, je nach Anforderung. Die erste Task eines Pearl-Programms wird von außen aktiviert (Programmstart).

Pearl erlaubt die Verwendung von sechs verschiedenen Datentypen:

FIXED
ganze Zahlen,
FLOAT
Gleitkommazahlen,
CHARACTER
Zeichenketten,
BIT
Bitmuster (Kette),
CLOCK
Zeitpunkte,
DURATION
Zeitdauer.

Im Programm verwendete Variablen und Konstanten müssen als einer der oben aufgeführten Datentypen vereinbart werden. Ähnlich wie in Pascal lassen sich in Pearl Felder und Zuordnungen von Variablen bilden. Felder können wieder Felder enthalten und getrennt angesprochen werden. Hierdurch ist es in Pearl möglich, die Daten hierarchisch zu strukturieren, Gruppen innerhalb von Feldern verschiedenen Prioritätsstufen zuzuordnen.
Pearl-Prozeduren steuern die Übergabe von Parametern und Konstanten. Die Übergabe eines Parameters kann etwa von dessen Wert oder seiner Adresse (Identität) abhängig gemacht werden. Man unterscheidet:

- Initial-Prozedur: Der aktuelle Parameter wird einfach kopiert;
- Identical-Prozedur: Im Gültigkeitsbereich des Prozedurblocks (Modul oder Task) wird ein neuer Name für den Parameter vereinbart;
- Returns sind Funktionsprozeduren, die eine Größe zurückliefern;
- Inline bewirkt eine Direkteinfügung des entsprechenden Prozedur-Parameters.

In Pearl sind Sondervereinbarungen möglich. Dem Benutzer wird gestattet, sowohl neue, abstrakte Datentypen (neben den oben genannten) als auch neue Operationen zu definieren. Besonders interessant und effektiv ist diese Möglichkeit der Sonderfunktionsgestaltung im Zusammenhang mit den oben genannten Feldern.
Die E/A-Struktur von Pearl besteht aus einem Netzwerk von „Datenstationen" und „Kanalumsetzern".

Tabelle 17:

Programmbeispiel zu PEARL
Listing

```
MODULE;                                    /* EXAMPLE PRESSURE CONTROL */

      SYSTEM;
            MULTIPLEXOR                    CPU*5;
            DIGITALOUT(0)                  MULTIPLEXOR*0;
            ANALOGIN(0)                    MULTIPLEXOR*9;
            VALVE          :               DIGITALOUT(0)*0,2;
            LAMP           :               DIGITALOUT(0)*3,1;
            DISCHARGE      :               DIGITALOUT(0)*4,1;
            PRESSURESENSOR1:               ANALOGIN(0)*1;
            PRESSURESENSOR2:               ANALOGIN(0)*3;
            READY:                         INTERRUPTINPUT*0;
            ALARM:                         INTERRUPTINPUT*2;

      PROBLEM;
                        SPECIFY VALVE DATION OUT BASIC DIM() TFU,
                               (LAMP, DISCHARGE) DATION OUT BASIC,
                               (PRESSURESENSOR*1, PRESSURESENSOR*2)
                               DATION IN BASIC DIM() TFU,
                        DECLARE (PRESS1, PRESS2, MEANPRESS) FLOAT,
                               OPENVALVE BIT(2)  INITIAL('10'B),
                               CLOSEVALVE BIT(2) INITIAL('10'B),
                               (TURNON, TURNOFF)
                               BIT INITIAL ('1'B, '0'B);

            START:     TASK GLOBAL;     /* ACTIVATED FROM OUTSIDE */

                        EVERY 1 SEC ACTIVATE PRESSURECONTROL;
                        WHEN ALARM ACTIVATE ALARMING;
                        END;                        /* OF TASK STRAT */
            PRESSURECONTROL: TASK PRIORITY 5;
                        TAKE PRESS 1 FROM PRESURESENSOR1;
                        TAKE PRESS 2 FROM PRESURESENSOR2;
                        MEANPRESS = (PRESS1 + PRESS2)/2;
                        IF MEANPRESS = .30
                            THEN SEND OPENVALVE TO VALVE;
                            ELSE IF MEANPRESS = .20 THEN SEND
                            CLOSEVALVE TO VALVE;
                              FIN;
                        FIN;
                        END;              /* OF TASK PRESURECONTROL */

            ALARMING:   TASK PRIORITY 3;
                        SEND TURNON TO LAMP;
                        SEND TURNON TO DISCHARGE;
                        WHILE MEANPRESS = .20
                              REPEAT
                               AFTER 2 MIN RESUME;
                        END;                              /* LOOP */
                        SEND TURNOFF TO LAMP;
                        SEND TURNOFF TO DISCHARGE;
                        END;                  /* OF TASK ALARMING */
```

Datenstationen verkörpern allgemein alle Peripheriegeräte oder deren Kanäle. Jede Datenstation besteht aus maximal vier Kanälen: dem Datenkanal, dem Steuerkanal, dem Unterbrecherkanal und dem Signalkanal.
Die einzelnen Kanäle müssen im Programm spezifiziert werden.

Kanalumsetzer schaffen die Möglichkeit der Anpassung verschiedenartiger Peripheriegeräte auf einheitliche Formate zur Kommunikation mit der Zentraleinheit oder untereinander. Kanalumsetzer arbeiten wie Interfaces. Sie setzen die Daten innerhalb eines Kanals in einer prozedurartigen Routine um.
Zur Pearl-Syntax gehört eine Reihe von Sprachelementen, die es ermöglichen, beispielsweise parallel ablaufende Prozesse zu steuern und sowohl termin- und zeitgerechte Abläufe zu aktivieren.
Auf die vielschichtigen Befehlsstrukturen kann hier nicht eingegangen werden, aber im obigen Beispiel wird ziemlich deutlich, welche Realtime-Möglichkeiten Pearl besitzt.
Pearl ist für Anwender geschaffen worden. Gegenüber Systemimplementierungssprachen wie Concurrent Pascal, Modula oder Ada ist Pearl leichter erlernbar und bietet zudem große Vorteile.
Die Sprachelemente zur Formulierung algorithmischer Zusammenhänge und Abläufe entsprechen dem Standard moderner Programmiersprachen (so etwa Pascal, PL/1).
Pearl hat durch die Aufteilung in Divisions ein hohes Maß an Portabilität und Dokumentationswert. Strukturiertes Programmieren wird unterstützt. Pearl ist in vielen Anwendungsbereichen eine echte Alternative zur Assembler-Programmierung.

52. PILOT

Programmed Inquiry Learning Or Teaching. PILOT ist eine Dialogsprache (siehe auch BASIC) und wurde von Prof. John Starkweather entwickelt. PILOT hat nie den Bekanntheitsgrad von BASIC erreichen können. Gründe hierfür sind unter BASIC (im ersten Teil unserer Serie) nachzulesen.

Im Gegensatz zu BASIC ist PILOT aber eine Dialogsprache ohne Kompromisse. Sie wird oft als *die* Dialogsprache schlechthin bezeichnet. Heutigen Versionen, wie dem Utah-PILOT (auch für IBM-PC kompatible Rechner unter MS-DOS erhältlich), liegt der PILOT-73-Standard zugrunde, – ständig erweitert und verbessert. PILOT ist für Anfänger geschrieben und leicht erlernbar und kann daher auch ideal für Lernzwekke eingesetzt werden. Als Dialogsprache ist sie im besonderen Maße für interaktive Anwendungen geeignet. Hierzu werden großartige Hilfestellungen in der Syntax geboten: spezielle Dateneingänge, programmierte Instruktionen, Testroutinen und anderes. Interaktive Anwendungen liegen dann vor, wenn im besonderen Maße Wert auf den Dialog zwischen Mensch und Maschine gelegt wird (was oft problembedingt sein kann). PILOT hat ein recht einfaches Format und eine leicht verständliche Syntax.

Tabelle 18:

Programmbeispiel zu PL/1:
Listing

```
 1:     BLOCK1: PROCEDURE OPTIONS(MAIN);
 2:             DECLARE A DECIMAL FIXED(5,2),
 3:                     B BINARY FLOAT(24),
 4:                     C CHARACTER(20), VARYING;
 5:             A = 2.34;
 6:             B = 4.56E+4;
 7:             C = 'BEISPIEL';
 8:             PUT SKIP(2) LIST('BLOCK 1, ZEILE 9:');
 9:             PUT SKIP LIST('A =' ,A,' B =' ,B,' C = ' ,C);
10:
11:         BLOCK2: BEGIN;
12:                 DECLARE A BINERY FLOAT(18) EXTERNAL,
13:                         X BINERY FIXED(15),
14:                 A = 2.875E+3;
15:                 B = 5.625E-3;
16:                 X = 275;
17:                 PUT SKIP(2) LIST('BLOCK 2, ZEILE 18:');
18:                 PUT SKIP LIST('A =' ,A,' B =' ,B,' C =' ,C,' X =' ,X,);
19:                 END BLOCK2;
20:
21:             PUT SKIP(2) LIST('BLOCK 1, ZEILE 22:');
22:             PUT SKIP LIST('A =' ,A,' B =' ,B,' C =' ,C);
23:
24:         BLOCK 3: BEGIN;
25:                 DECLARE A BINARY FLOAT(!0) EXTERNAL,
26:                         C CHRACTER(!=) VARYING,
27:                         X BINARY FIXED(!2);
28:                 X = 386;
29:                 C = ' BEISPIEL 1';
30:                 PUT SKIP(2) LIST('BLOCK 3, ZEILE 31:');
31:                 PUT SKIP LIST('A =' ,A,' B =' ,B,' C =' ,C' X =' ,X,);
32:                 END BLOCK3;
33:
34:             PUT SKIP(2) LIST('BLOCK 1, ZEILE 35:');
35:             PUT SKIP LIST('A =' ,A,' B =' ,B,' C =' ,C);
36:
37:             END BLOCK1;
```

53. PLANKALKÜL

Eine der ersten universellen algorithmischen Sprachen überhaupt. Entwikkelt wurde sie schon 1945 von Dr. Ing. e.h. Konrad Zuse, dem Erbauer der ersten programmgesteuerten Rechenanlage. PLANKALKÜL war wegweisend für die Entwicklung aller symbolischen Programmiersprachen.

54. PL/1

Programming Language Nr. 1

PL/1 könnte man als Synthese aus Fortran, Algol und Cobol bezeichnen, was ihren Anwendungsbereich sowohl in den technisch-mathematischen als auch in den kaufmännischen Bereich legt (Mischsprache). Hier wurde versucht, die aus der Erfahrung mit diesen Sprachen erkannten Mängel zu vermeiden und die Vorteile in einer gemeinsamen Sprache zu vereinen. Von Algol wurde die strenge Einteilung eines Blocks in Vereinbarungs- und Anweisungsteil, die Blockstruktur sowie die formatfreie Ein- und Ausgabe, die Ablaufstrukturen, Records, Pointer und das Prozedure-Konzept mit rekursivem Aufruf übernommen. Cobol lieferte die Dezimalarithmetik, Picture-Formatierung, Textverarbeitung, Datenstrukturierung und eine umfangreiche Dateiverwaltung. Aus Fortran kommen die impliziten Laufanweisungen, numerische Standardfunktionen, die Functions-Subroutine und das Prinzip der Common-Bereiche (siehe hierzu auch unter Fortran). Der Nachteil: PL/1 ist in vollem Umfang nur auf größeren Rechenanlagen anwendbar, da der benötigte Speicherplatz wesentlich größer ist als der bei anderen Programmiersprachen. Auch die Syntax ist umfangreicher, was das Erlernen von PL/1 erschwert. Vorteile: PL/1 ist flexibler in den Anwendungen und problemloser bei der Verarbeitung großer Datenmengen innerhalb der Rechenanlage. PL/1 vereint die Vorteile verschiedener anderer Programmiersprachen in sich: auf der einen Seite gute Texthandhabung und Dialogfähigkeit, auf der anderen Seite die mathematischen Möglichkeiten der genannten Programmiersprachen. Der modulare Aufbau von PL/1 gestattet es dem Anfänger, zunächst einen kleinen Teil der Syntax zu erlernen und sich dann schrittweise den gesamten Sprachumfang anzueignen. Vieles wird vom PL/1-Compiler übernommen, wie zum Beispiel die Speicherplatzorganisation, Datenumwandlungen, Formatierung der Ein- und Ausgabe und anderes. Man spricht hierbei vom Default-Konzept. Der PL/1-Sprachumfang ist so umfassend, daß fast alle Anwendungsgebiete abgedeckt werden können. So muß nicht bei wechselnden Anforderungen auch die Programmiersprache gewechselt werden. PL/1 unterstützt Mehrbenutzer-Betriebssysteme durch mehrstufige Passwortabfrage und Dateischutz bis zum Sperren. Integriert sind außerdem

FLEXIBEL UND UMFASSEND

umfangreiche Möglichkeiten zur Behandlung von Ausnahme- und Fehlersituationen sowie Overlaytechniken.

PL/1 wurde zum ersten Mal 1965 vorgestellt. Seitdem wurde sie, wie viele andere Programmiersprachen auch, weiterentwickelt. Da dies aber nicht durch einen neutralen Ausschuß geschah (ANSI), gibt es heute mehrere PL/1-Dialekte. Erst 1979 wurde PL/1 in einer Norm vereinheitlicht:

- ISO 6160: „Programming Language PL/1" (international);
- DIN 66.255: „Informationsverarbeitung; Programmiersprache PL/1", Ausgabe 1980.

Jedoch in englischer Sprache, da man eine Abweichung bei der Übersetzung ins Deutsche befürchtete, was unter allen Umständen vermieden werden sollte, denn die Deutsche Industrie-Norm sollte fachlich der internationalen (ISO) Norm entsprechen.

PL/1 gehört heute zu den am meisten verbreiteten Programmiersprachen und ist auch für die Simultanverarbeitung geeignet. Simultanverarbeitung liegt

dann vor, wenn ein Programm gleichzeitig mit mehreren peripheren Geräten (Drucker, Dateien...) arbeitet. Hierbei entsteht weniger Wartezeit, da die Peripherie einer Anlage sehr viel langsamer ist als die Zentraleinheit. Steuereinheiten in den Kanälen ermöglichen den gleichzeitigen Einsatz mehrerer Geräte.
Ein weiterer Vorteil von PL/1 liegt in den vorhandenen Sprachbestandteilen zur Gliederung von Programmen.
In PL/1 werden Unterprogramme als Prozeduren geschrieben, die wiederum von anderen Prozeduren aufgerufen werden können. Komplizierte Aufgabenstellungen können so in logisch unterteilte Abschnitte (Module)

MODULTECHNIK

untergliedert werden. Dies verbesserte die Effezienz eines Programms erheblich, vor allem in Hinblick auf die Transparenz des Lösungswegs. Korrekturen und Änderungen (die Programmpflege) werden dadurch erheblich erleichtert.
Alle Prozeduren, die nach dem Start des Haupt-Programms bis zur Rückgabe der Ablaufsteuerung an das Betriebssystem aktiviert werden, bilden das Gesamtprogramm.
Dieses kann aus internen und externen Prozeduren bestehen. Das Hauptprogramm ist dabei immer eine externe Prozedur, die durch den Zusatz Options(Main) gekennzeichnet wird, alle anderen Prozeduren sind Unterprogramme.
Interne Prozeduren sind Bestandteile anderer Prozeduren (übergeordnete Prozedur) und werden mit dieser gemeinsam übersetzt und gebunden. Sie können nur mit der übergeordneten Prozedur verwendet werden.
Im Gegensatz hierzu werden externe Prozeduren separat übersetzt und bilden eigenständige Module. Diese können in Bibliotheken abgelegt und bei Bedarf an Hauptprogramme angebunden werden.
PL/1 gibt es in der Zwischenzeit auch für viele Klein- und Micro-Computersysteme unter CP/M oder MS-DOS.
Bekannt wurde noch PL/9900, eine Abwandlung von PL/1, entwickelt von Texas Instruments für deren Rechnerkonzept 9900.
Inzwischen hat aber fast jeder Hardware-Hersteller eigene Implementationen von PL/1 für seine Maschinen im Angebot, sofern die Rechner für PL/1 geeignet sind.
Eine interessante Neuentwicklung stellte auch PL/M dar, eine spezielle, abgemagerte Version für Micro-Computer.
(Programmbeispiel siehe Tabelle 18.)

55. Prolog

PROgramming in **LOG**ic.
Prolog ist eine deklarative Programmiersprache, wie sie zur Lösung von Problemen aus dem Bereich der künstlichen Intelligenz (KI) eingesetzt werden.
Deklarative Sprachen beschreiben die vom Programm zu lösende Aufgabe, der Lösungsweg (Algorithmus) muß nicht definiert werden. Daher spricht man auch von logischer oder symbolischer Programmierung.
Die in Prolog bevorzugt verwendeten Datenstrukturen sind einfache und verkettete Bäume. Der Begriff des Datentyp im Sinne konventioneller Programmiersprachen existiert in der logischen Programmierung nicht.
Verschiedene Prolog-Compiler verlangen jedoch eine Deklarierung von Variablen für die Compilierung des Quellcodes.
Statt Datentypen werden in Prolog Domain-Typen verwendet:

integer	**ganze Zahlen**
real	**reelle Zahlen**
char	**Zeichen**
string	**Zeichenkette**
symbol	**Symbol**
file	**Datei-Name.**

Integerzahlen liegen zwischen −32.768 und +32.767.
Reelle Zahlen sind durch einen Dezimalpunkt gekennzeichnet, auch wenn keine Nachkommastellen auftreten.
Größere Zahlen werden in exponentieller Schreibweise dargestellt. Die Char-Variablen können einzelne Zeichen enthalten. Die Zeichen werden mit Hochkommas geschrieben. Zeichenketten (Strings) sind beliebige Zeichenkombinationen, auch sie werden durch Hochkommas abgegrenzt.
Symbol-Variablen beginnen mit Kleinbuchstaben oder sind Zeichenfolgen, die von Hochkommas eingeschlossen sind. Symbol und String werden in Prolog unterschiedlich verarbeitet.
File-Variablen ermöglichen die Definition symbolischer Dateinamen, welche als Argumente in Prädikaten verwendet werden können.
Jedes Programm darf aber nur einen Domaintyp enthalten, der zur Gruppe „File" gehört.
In der Deklarierung können mehrere Objekte des gleichen Domain-Typs zusammengefaßt werden. In Prolog ist es möglich, für die Definition einer Domain diese selbst zu verwenden. Dadurch entsteht ein rekursives Objekt.
Standard-Domaintypem müssen nicht deklariert werden.
Ein Prolog-Programm ist als Sammlung von Fakten und Regeln auch eine Datenbank. Durch Einfügungen und Löschungen während der Ausführung ist sie dynamisch.
Standard-Prolog erlaubt den Austausch von Fakten und Regeln. Manche Prolog-Dialekte wie Turbo-Prolog lassen nur den Austausch von Fakten zu.
Die Anordnung von Fakten und Regeln, sowie der Datensätze ist in Prolog sehr wichtig für die Ausführung des Programms.
Je nach Anordnung kann ein Programm auf verschiedene Weisen interpretiert werden und es kann zu völlig unterschiedlichen Ergebnissen kommen.
An dieser Stelle sollte vielleicht einmal etwas Grundsätzliches zu den sogenannten KI-Sprachen gesagt werden.
Wie schon erwähnt, handelt es sich hier um deklarative Programmiersprachen. Im Gegensatz zu den herkömmlichen Sprachen wird nur ein Problem, nicht aber der komplette Lösungsweg beschrieben.
In algorithmische Sprachen wie Fortran, Cobol, BASIC, Pascal und andere müssen Lösungswege für alle möglichen Fälle bei der Programmierung definiert werden, was sehr aufwendig sein kann und damit auch als Qualitätsmerkmal eines Programms gilt.
Logische oder deklarative Programmiersprachen erlauben aber die Festlegung von Fakten (Facts) und Regeln (Rules), die für die eigentliche Lösung des Problems notwendig sind. Also nicht der Programmie-

KÜNSTLICHE INTELLIGENZ

rer löst das gestellte Problem, sondern das Programm. Die Hauptarbeit des Programmierers besteht darin, die „Facts" und „Rules" korrekt und vollständig zu formulieren.
Bei Prolog ist das oben Gesagte leider nur zum Teil verwirklicht worden. Auch Prolog kann nicht ganz ohne explizite Anweisungen (Algorithmen) auskommen. Bei den konventionellen Programmiersprachen ist dieser Algorithmus das Programm selbst. Bei Prolog ist der benötigte Algorithmus inhärenter Bestandteil des Interpreters oder des Compilers. Aus diesem Grunde wurde schon mehrfach versucht, Prolog den deklarativen Charakter abzuerkennen. Sicher ist Prolog keine reine deklarative Programmiersprache, sondern eher eine Mischform,

wie viele höhere Programmiersprachen.
Um einen Eindruck zu gewinnen, wie ein Prolog-Programm aussieht, in Tabelle 19 ein Ausschnitt, geschrieben in Turbo-Prolog.
Es gibt viele Prolog-Implementierungen, von denen sich die meisten an den De-Facto-Standard halten.
Prolog-Programme werden mit einem gewöhnlichen Editor erstellt. Manche Implementationen erlauben die Syntax-Überprüfung schon während des Editierens.
Zudem werden sogenannte Software-Interfaces angeboten, die Schnittstellen zu anderen Programmiersprachen darstellen (etwa zu C).
Grundsätzlich zu unterscheiden ist zwischen Prolog-Interpreter und Compilern. Vor- und Nachteile dieser Übersetzungsmethoden wurden schon hinreichend besprochen. Sie gelten für alle in Frage kommenden Programmiersprachen.

Tabelle 19:

Programmbeispiel zu Turbo-PROLOG
Listing

```
domains
        position = integer*
        nummer = integer
        zugliste = nummer*

Predicates
        regel ( nummer, position, position)
        wende (integer, integer)
        endposition (position)

        postcheck (position)
        gültig (position)

        löse (position, integer, zugliste)
        run

clauses
        regel(1,(x1,x2,x3,x4,x5,x6,x7,x8,x9),(y1,y2,y3,y4,y5,y6,y7,y8,y9,)
        if wende(x1,y1), wende(x2,y2), wende(x4,y4), wende(x5,y5),
        x3 = y3, x6 = y6, x7 = y7, x8 = y8, x9 = y9
        regel(2,(x1,x2,x3,x4,x5,x6,x7,x8,x9),(y1,y2,y3,y4,y5,y6,y7,y8,y9,)
        if wende(x1,y1), wende(x2,y2), wende(x3,y3),
        x4 = y4, x5 = y5, x6 = y6, x7 = y7, x8 = y8, x9 = y9
        regel(3,(x1,x2,x3,x4,x5,x6,x7,x8,x9),(y1,y2,y3,y4,y5,y6,y7,y8,y9,)
        if wende(x2,y2), wende(x3,y3), wende(x5,y5), wende(x6,y6),
        x1 = y1, x4 = y4, x7 = y7, x8 = y8, x9 = y9
        .
        .
        .
        wende (0,1).
        wende (1,0).

        endposition ((1,1,1,1,0,1,1,1,1)).

        poscheck (AP) if gültig (AP,9).
        poscheck (-) if
            beep, nl,
            write ("Eingabefehler."), nl,
            !,
            fail.

        gültig ((),0).
        gültig ((X,T),N) if
          X <> 0,
          X <= 1,
          N1 = N - 1
          gültig (T,N1)

        löse (A,-,Y) if
          endposition (A)
          !,
          Z = ().
          .
          .
          .
goal
        run.
```

56. Prosa
PROgrammiersprache mit Symbolischen Adressen.
Prosa ist wie Autocode ein Vorläufer der heute gebräuchlichen Assembler-Sprachen. Wie diese ist Prosa maschinenorientiert.
Prosa findet heute kaum noch Anwendung. Gebräuchlich war Prosa zwischen 1966 und 1965.

57. QBE
Query **B**y **E**xample.
Grafische Datenbank-Abfragesprache, von IBM Ende der 70er Jahre im San Jose Research Laboratory entwickelt.
QBE ist non-prozedural (siehe auch SQL).
Im Gegensatz zu SQL ist QBE allerdings nur für große Rechenanlagen tauglich.

58. QL
Query **L**anguage.
Abfragesprache (DL-Sprache).
Query Language hat eine sehr einfache Syntax und Struktur und ist daher leicht erlernbar.
Sie wird hauptsächlich zur Handhabung (Abfragung) von Datenbanksystemen benutzt (siehe auch SQL).
DL-Sprachen sind meist nicht-prozedural. Der Anwender teilt dem System mit, wie das zu lösende Problem aussieht und nicht, wie es zu lösen ist.

59. RPG
Report-**P**rogramm-**G**enerator, auch **LPG** (**L**isten-**P**rogramm-**G**enerator).
RPG und das heute gebräuchlichere RPG II, eine Weiterentwicklung, sind eigentlich keine echten höheren Programmiersprachen im eigentlichen Sinne, sondern, wie der Name schon sagt, Programm-Generatoren, also Systeme, die von Dateien eingelesene Daten-

AUSWERTUNG VON DATENSÄTZEN

sätze nach vorherbestimmten, gleichartigen Umformungen bearbeiten und meistens in Form einer Liste (daher LPG) ausgeben. RPG findet deshalb hauptsächlich in der kommerziellen Datenverarbeitung Anwendung, wo solche Probleme überwiegend auftreten.
Man spricht bei RPG von einer parametrischen Programmierung, da alle Anweisungen und Angaben in Form von Parametern beschrieben werden. Mit Hilfe dieser Parameter wird das Programm dann erstellt.
Erst die Entwicklung von RPG II konnte eine vollständige, gute und umfassende Bearbeitung aller, im kommerziellen Bereich auftretenden Probleme bewältigen. Dennoch kann RPG nicht zu den problemorientierten Programmiersprachen gezählt werden, obwohl einige Softwarehäuser dies gerne sähen. RPG enthält keine frei kombinierbaren Befehle. Und dies ist nun einmal eines der Kriterien für eine problemorientierte Programmiersprache.
RPG wurde ursprünglich von IBM entwickelt und von verschiedenen anderen Softwareherstellern weiterentwickelt. RPG II

wird meist in die Standard-Sprachumgebung des Rechners eingegliedert. Programme in RPG II können Module in anderen Sprachen aufrufen sowie diverse Laufzeit-Bibliotheksprogramme. Andererseits werden die Ergebnisse der RPG II-Programme (Listen) so ausgegeben, daß sie ohne weiteres auch von anderen Programmen in anderen Sprachen weiterverarbeitet werden können.

60. RPNL

Reverse Polish Notation Language.
RPNL ist eine höhere Programmiersprache, die sowohl mit Pascal als auch mit Forth große Ähnlichkeiten aufweist. Der erste Unterschied zu anderen Programmiersprachen ist die Verarbeitung der „Umgekehrten Polnischen Notation" (UPN) bei den Operationen (siehe hierzu auch EOL).
RPNL unterstützt die strukturierte Programmierung und läßt hohe Ausführungsgeschwindigkeiten zu. RPNL ist compilerorientiert, obwohl das vom Compiler erzeugte Objektprogramm noch keinen direkten Maschinencode darstellt. Man spricht hierbei von einem „Zwischencode". Dieser Zwischencode besteht aus einer Zusammenstellung von Bibliotheksprogramm-Aufrufen, die allerdings sehr maschinennah sind, was sich positiv auf die Ausführungsgeschwindigkeit auswirkt.
In RPNL werden grundsätzlich nur zwei Datentypen unterschieden:
■ *Numerische Daten:*
INTEGER-TYP
(bis zur Größe 2 hoch 16 –1),
BOOLEAN-TYP
(True, False),
ARRAY-TYP
(Zusammenfassung verschiedener oder gleicher Typen).
■ *Alphanumerische Daten:*
STRING
(maximal 255 Zeichen lang).

Alle in einem RPNL-Programm verwendeten Variablen oder Konstanten müssen am Programmanfang definiert werden. Folgende Namen dürfen nicht als Variablen oder Konstanten benannt werden, da sie bereits vergeben sind:

■ **I, J, K**
als Laufindizes in Schleifen,
■ **TRUE, FALSE**
als logische Werte für „wahr" oder „falsch".

Tabelle 20:

Wahrheitstabelle der Boolschen Operatoren (Prädikate)

Operand 1	Prädikat	Operand 2	Ergebnis
TRUE	NOT	-	FALSE
FALSE	NOT	-	TRUE
TRUE	AND	TRUE	TRUE
TRUE	AND	FALSE	FALSE
FALSE	AND	TRUE	FALSE
FALSE	AND	FALSE	FALSE
TRUE	OR	TRUE	TRUE
TRUE	OR	FALSE	TRUE
FALSE	OR	TRUE	TRUE
FALSE	OR	FALSE	FALSE

Der Programmaufbau eines RPNL-Programmes ist grundsätzlich in drei Teile gegliedert:

1. Programmklammer mit Namen
Die erste Zeile eines RPNL-Programmes enthält immer die Anweisung:
PROGRAMM Programmname
.
In der letzten .
Zeile steht .
immer END

2. Deklarationsliste
Beispielsweise VAR für Variablenname,
CONST für Konstantenname,
Anzahl ARRAY Feldname.

Hier wird einerseits der Variablen/Konstanten- und Feldname deklariert, andererseits Speicherplatz zugewiesen und belegt: normalerweise für jede Größe 16 Bit, oder für die Größe des Feldes (Anzahl) * 16 Bit.

3. Anweisungsliste
Die Anweisungsliste enthält die Algorithmen, welche den Lösungsweg des Problems beschreiben.
Man unterscheidet bei RPNL drei verschiedene Anweisungstypen:
a) einfache Anweisungen,
b) E/A-Anweisungen für die Ein- und Ausgabe,
c) Steueranweisungen für Datenfluß und Ausführungssteuerung.
Einfache Anweisungen können aus einstelligen oder zweistelligen Operatoren bestehen. Einstellige Operatoren bearbeiten einen Operanden, zweistellige Operatoren benötigen zwei Operanden.

Einstellige Operatoren:
INC bildet das Inkrement des angegebenen Operanden (=Erhöhung um 1),
DEC das Dekrement wird gebildet (Verringerumg um 1),
NOT logische Verneinung (Negation),
DUP Duplizierung eines Wertes (Kopie),
? Abruf eines Speicherplatzes, einer Variablen oder Konstanten.

Zweistellige Operatoren:
+ Addition,
– Subtraktion,
DIV Division (nur ganzzahlig), der Rest ist verloren,
***** Multiplikation,
MOD Division zweier Operanden, welche nur den errechneten Rest anzeigt (Modulo Division),
SWAP Vertauschung von zwei Operanden,
AND Logisches UND (Konjunktion),
OR Logisches ODER (Disjunktion).

Zur „Wahrheitstabelle der Boole'schen Operatoren" siehe *Tabelle 20.*

Außerdem stehen noch die logischen Vergleichsoperatoren gleich/größer/kleiner und deren Kombination zur Verfügung. Sie liefern für Integer-Operanden logische Wahrheitswerte als Ergebnis.

Ein- und Ausgabeanweisungen und RPNL-Prozeduren:
CR Carriage Return (Zeilenvorschub)
WRITE Ausgabe von Text (in Hochkommas),
PRINT Ausgabe einer Integerzahl,
PRINT(S) Ausgabe eines Strings,
READ liest eine Integerzahl ein,
READ(S) liest einen String ein,
STRING weist einem Text Speicherplatz zu.

Bedingungen und Schleifen
IF-THEN-ELSE
Vor dem Schlüsselwort IF steht ein Ausdruck, der die Bedingung darstellt (ein logischer Wert), da-

hinter die Anweisungsliste, die nur dann ausgeführt wird, wenn die Bedingung erfüllt ist (TRUE). Die Anweisungsliste hinter ELSE wird genau dann ausgeführt, wenn die Bedingung vor dem IF nicht erfüllt ist (FALSE).

REPEAT UNTIL LOOP
Hinter REPEAT folgt die Anweisungsliste, deren Anweisungen wiederholt werden sollen. Abgeschlossen wird die Schleife durch das Schlüsselwort LOOP. An einer geeigneten Stelle zwischen REPEAT und LOOP steht UNTIL, dem direkt ein Ausdruck (logischer Wert) vorangeht.
Abhängig von dem aktuellen Wert des Ausdruckes erfolgt die Fortsetzung des Programms mit der ersten Anweisung hinter UNTIL (bei FALSE) oder hinter LOOP (bei TRUE) und damit wird aus der Schleife herausgesprungen.

FOR LOOP
Vor dem Schlüsselwort FOR müssen zwei Operanden angegeben werden, die den Endwert und den Startwert der Laufvariablen festlegen. Zwischen FOR und LOOP steht die Anweisungsliste. Es können drei Schleifen ineinander verschachtelt werden. Die Variable I ist immer für die innere Schleife reserviert, J für die mittlere und K für die äußere. Bei nur einer Schleife oder einer Verschachtelung von zwei Schleifen gilt das oben Gesagte analog.
(Programmbeispiel siehe Tabelle 21.)

61. SIMULA

SIMULA ist mehr ein System als eine Sprache. Mit Hilfe des SIMULA-Systems lassen sich beispielsweise rechnerinterne Abläufe anhand von Modellen untersuchen. So wurden umfangreiche Untersuchungen von Betriebssystemen mit Hilfe von SIMULA durchgeführt.
SIMULA gibt es seit etwa 1965.
Der Vorteil gegenüber anderen Programmiersprachen liegt in der Struktur des Systems, welches sich besonders zu Simulationszwecken eignet (daher der Name SIMULA).
Es gibt verschiedene SIMULA-Systems, auf die aber wegen ihrer Spezialisierung hier nicht eingegangen werden kann.

Tabelle 21:

Programmbeispiel zu RPNL
Listing

```
PROGRAMM BEISPIEL
  5 ARRAY LISTE
  VAR ZAHL
  4  0  FOR
     REPEAT
     WRITE "EINGABE" CR
        ZAHL READ
        ZAHL ? I 2 * LISTE + ! =
     ZAHL ? 100 (= UNTIL
        ZAHL PRINT
        WRITE "ZU GROSS" CR
        LOOP
     LOOP
END
```

62. SIL 3

System Implementation Language 3.
Maschinenunabhängige System-Implementierungssprache (SIL).
Sie dient zur Software-Herstellung und zeichnet sich in erster Linie durch ihre Codeeffizienz aus.
Die Programmierung von SIL ist recht kompliziert, nicht zuletzt durch die fehlende Problemnähe und die speziellen Anforderungen an der Anwendersoftware.

63. Smalltalk 80

Smalltalk wurde Anfang der siebziger Jahre entworfen, 1980 fertiggestellt (daher Smalltalk 80) und 1981 zum ersten Mal der Öffentlichkeit vorgestellt. Im „Xerox Paolo Alto Research Center" entwickelt, ist Smalltalk zugleich Programmiersprache und Rechensystem. Großen Wert bei der Ausstattung dieses Systems wurde auf grafische Interaktionsmechanismen wie Windows gelegt. Menü- und Maustechniken werden ebenfalls unterstützt. In Smalltalk konnte also schon das realisiert werden, was später von Betriebssystemen (grafischen Benutzeroberflächen) wie GEM oder anderen übernommen wurde.
Das Ausgabemedium für Smalltalk ist immer ein Bitmap-Display, da jede Ausgabe grundsätzlich grafisch erfolgt. Dies geschieht mit Hilfe sogenannter „Forms". Cursor, Bildfläche und auch Schriftarten werden alle als Forms gehandhabt. Smalltalk enthält zwei Editoren zur Veränderung der Forms.
Auch echte Grafik ist in Smalltalk möglich: Liniengrafik und Splines, Kreise sowie andere geometrische Figuren.
Zur Simulation gibt es in Smalltalk das sogenannte Klassenkonzept, welches von der Programmiersprache SIMULA übernommen wurde. Die Klassen dienen den Wahrscheinlichkeitsverteilungen und ereignisgetriebenen Simulationen.
Smalltalk ist, wie schon erwähnt, nicht nur eine Sprache, sondern gleichzeitig ein System. Das System ist seinerseits in Smalltalk geschrieben und basiert auf einer virtuellen Maschine. Das System ist dem Benutzer zugänglich und ermöglicht so Eingriffe.
Für alle Anwenderprogramme wird ein einheitlicher Mechanismus für die Realisierung der interaktiven Schnittstelle zur Verfügung gestellt – ein ähnliches Vorgehen wie bei den Betriebssystemen mit grafischer Benutzeroberfläche. Dieser Mechanismus besteht aus den Bestandteilen Model, View und Controller. Das Modell wird von den Daten und Informationen gebildet. Die View beschreibt die Darstellungen und der Controller übernimmt die Steuerung, also er beschreibt, welche Auswirkungen Ereignisse (etwa Benutzereingaben) auf das System haben.

SPRACHE UND SYSTEM

Smalltalk ist objektorientiert. Jede Komponente von diesem System ist ein Objekt. Die Programme in Smalltalk beschreiben lediglich die Kommunikation dieser Objekte untereinander und natürlich die Objekte selbst.
Die Programmierung in Smalltalk besteht daher im Wesentlichen aus der Neubildung von Objekten und dem „Senden von Botschaften" zwischen diesen.
Die Objekte in Smalltalk 80 sind sehr abstrakt und haben kaum noch etwas mit den technischen Eigenschaften der Hardware des Rechners zu tun oder den klassischen Objekten in der Software, also Zahlen oder Zeichen. Smalltalk gibt es für ver-

schiedene Rechnersysteme, in der Regel sind dies aber keine Microcomputer. Lediglich für den Macintosh von Apple gibt es eine Implementation.

64. SQL

Structured Query Language.
Strukturierte Abfrage-Sprache, vorwiegend für Datenbanken entwickelt von IBM im San Jose Research Laboratory.
SQL ist seit Ende der 70er Jahre auf dem Markt und wurde seitdem in verschiedensten Datenbanksystemen installiert. Bekanntestes Beispiel ist die relationale Datenbank ORACLE (1979).
SQL ist non-prozedural, also muß nicht der Lösungsweg vom Programmierer beschrieben werden, sondern nur das Problem selbst, das Programm sucht nach Lösungen.
Datenstrukturen können in Form von Tabellen dynamisch definiert und manipuliert werden. Diese Möglichkeiten stehen dem Benutzer über ein spezielles Interface mit dem bezeichnenden Namen UFI = User Friendly Interface zur Verfügung, welches die Datenmanipulation im Dialogbetrieb zuläßt.
SQL besitzt erstaunliche Abfragemöglichkeiten, die jedoch je nach gewünschter Abfrage Zeit kosten können.
SQL bietet hierfür die Möglichkeit, Indizes zur Retrieval-Beschleunigung zu definieren. Die Indexdefinition kann auch dazu benutzt werden, um eine von SQL nicht abgedeckte, jedoch gewünschte Leistung sicherzustellen.
Neben SQL hat sich – ebenfalls von IBM entwickelt – eine weitere Abfrage-Sprache etabliert: QBE (Query By Example) – mehr ein grafisches Abfragesystem. QBE, auch als Ergänzung zu SQL, ist nur auf mittleren und großen Rechenanlagen einsatzfähig.
Für Mini- und Microcomputer (so VAX von DEC und IBM, PC, XT, AT) gibt es SQL ebenfalls in Verbindung mit der Datenbank ORACLE.

65. SYMAP

Eigenentwicklung der DDR, aufgebaut aus verschiedenen Elementen von Fortran und Exapt.

66. V.I.P.

Visuelle Interaktive Programmiersprache.
Programme in V.I.P. werden über Flußdiagramme eingegeben. V.I.P. ist auf den Apple-Macintosh zugeschnitten und erlaubt Zugriffe auf die Macintosh-Toolbox. V.I.P. verfügt über einen visuellen Debugger und einen sehr schnellen Interpreter. Es gibt auch Transferprogramme, die den V.I.P.-Code in LS-Pascal übersetzen.

67. PL/PC

Programming Language for Personal Computers.
PL/PC ist eine relativ neue Universal-Programmiersprache, speziell geschaffen zum Einsatz auf PCs unter MS-DOS.
Wie BASIC ist PL/PC interaktiv, hat aber diesem gegenüber den Vorteil, strukturiert zu sein (wie MODULA-2).
Auch von APL sind Elemente bei der Schaffung dieser neuen Programmiersprache übernommen worden; so zum Beispiel die mächtigen Datenstrukturen und Manipulationsfunktionen.
Zum Sprachumfang gehören Editoren für Datenstrukturen, die wie ein Spreadsheet aufgebaut sind, und Text zur Deklaration von Subroutinen, Compiler für selbst definierte Subroutinen, Interpreter, Debugger, Grafikunterstützung für alle bekannten Bildschirme einschließlich Turtle-Grafik.
Für rechenintensive Anwendungen existiert eine Spezialversion, welche die Coprozessoren 8087 und 80287 unterstützt.
Variable können ihren Typ (Boolean, Byte, Integer, Long [32 Bit], Real und Complex, String, Charakter) dynamisch ändern.
Die meisten mathematischen Funktionen sind eingebaut.
PL/PC ist leicht lernbar und daher auch für Anfänger geeignet, hat aber gegenüber BASIC gewisse Vorteile.
Am Ende dieses Beitrages können wir nur hoffen, etwas Klarheit in dieses Sprachengewirr gebracht zu haben.

Ende der Serie

Oliver Rosenbaum □

BÖRSE

VERKAUFE Kniffel f. C64 (40 Zeichen) f. 10,– DM; H. Güntner, Geyerswörthstraße 13, 8600 Bamberg

VERKAUFE C64 m. Drukker, Akustikkoppler, 2 Joysticks, Floppy, 10 LEE-Disks u. Interfacediskette f. Akustikkoppler f. 1000,– DM; Oliver Peter, Tel. 0911/733851

VERKAUFE meine deutsche Flight-2-Anleitung! Public-Domain-Software f. C128 günstig, u.a. der 1. Flight-Simulator f. 128er. Info gegen Rückporto bei Uwe Schwesig, Dorfstr. 9a, 2406 Stockelsdorf, Tel. 0451/493306

HALLO Freaks! Suche Tauschpartner f. C64/C128-Software; habe gute Games; nur Disk. Listen an Elmar Lohmann, Südring 53a, 4834 Harsewinkel

VERKAUFE C64, Datasette, 2 x Floppy 1541 u. C116 günstig. Klaus Spoden, Tel. 0228/442490

ABGEBRANNTER 12jähriger Schüler sucht dringend billige, funktionstüchtige u. gut erhaltene Floppy 1541, Drucker u. Datasette 1531. Thorsten Bieck, Loher-Weg 12, 2240 Heide

VERKAUFE Phonemark-Kass.-Gerät f. C64/128 neuw. 30,– DM; GEOS V 1.2 m. Handb. 30,– DM; Giga-CAD Plus-Buch +2 Disk. 30,– DM; M+T 64er Programmservice-Disk. Sonderh. 15/18/87; Ausgabe 3/8/87 je 20,– DM; Tel. 08230/1581 bei Maier

VERKAUFE für C64 Cartridge „Magic-Formel V 1.2" neuw., aus finanz. Gründen 150,– DM; Tel. 0871/22659 ab 18 h tägl.

Public-Domain-Gratisliste für C64/C128 anfordern bei Fr. Neuper, 8473 Pfreimd, Postfach 72. Es lohnt sich!

WER kann mir helfen? Suche für meinen Seikosha-GP-700 VC Drucker-Routine, damit ich Programme von GEOS ausdrucken kann! Alle vorhandenen Routinen sprechen auf meinen Drucker leider nicht an! Hans Jürgens, Provinstr. 111, 1000 Berlin 51

SCHÜLER mit geringen Mitteln sucht Computer + Zubehör (C64, Floppy o.a. Typen) geschenkt od. sehr billig! Nehme auch Einzelteile od. defekte. C. Maihöfer, Leinzellerstr. 27, 7071 Brainkoten

SUCHE Commodore C128 u. CP/M Software. Listen an: Mellinger Jakob, Bodenackerstr. 23, CH-4657 Dulliken

C64/128 Lernprogr. Techn. Mathe+Schulanwend. +Grafik zu reellem Preis. Bruchr., Physik, Chemie, Vokabeltr., Geometrie, Test +Trainprg. Werkzeugmasch. Zanr. Festigk. Hydr. E-Techn. Katlg. 1 DM. Softvers. A. Ristau, Peetzweg 9, 3320 Salzgitter 1

SUCHE 128er-Freak, der mir hilft, Reaktionszeitmessung (genauer als 1/60 Sek.) zu programmieren. Wer kennt Software für 128er-Modus? Dirk Stelling, Peinerweg 26, 2080 Pinneberg, Tel. 04101/200483 bis 24 h

WEGEN Hobby-Aufgabe zu verk.: Commodore 128D kompl. m. Handbüchern u. ca. 300 Programmen (Spielanwender); Preis VB; Edmund Engelking, Baesweilerweg 24, 5132 Übach-Palenberg, Tel. 02451/41868

GEOS 128

Gelungen

Mit vielen Vorschußlorbeeren bedacht, ist in der Zwischenzeit die 128er-Version der bekannten Benutzeroberfläche für den C64 auf den Markt gekommen. Zu einem stolzen Preis von rund 120 Mark. Lesen Sie hier, was unser Leser Jürgen Heinisch dazu meint.

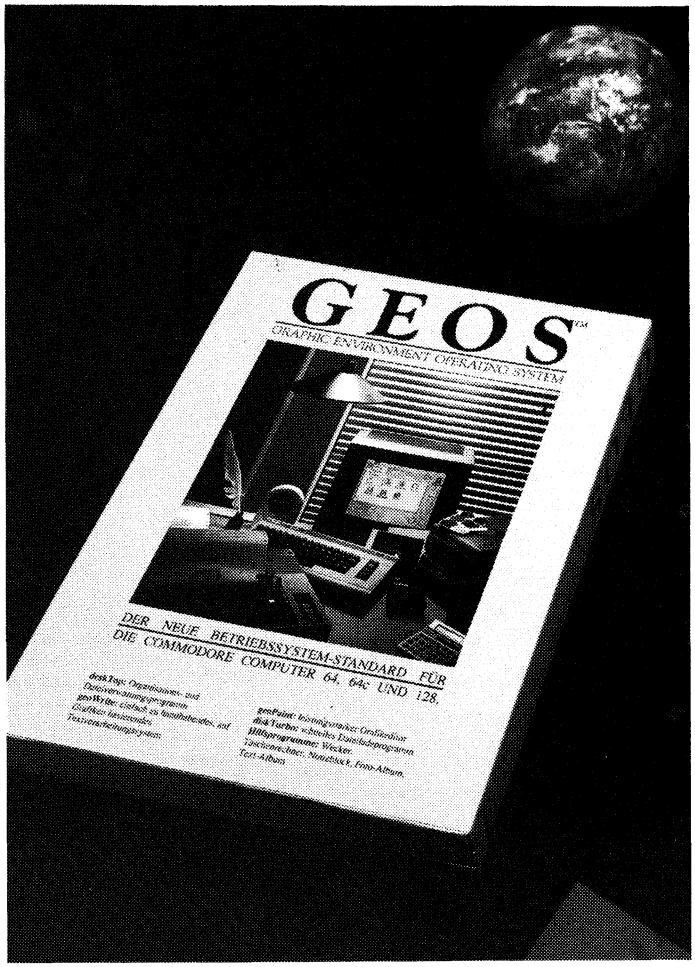
Geos 128 wird in einer festen Pappschachtel, Größe etwa DIN A5, ausgeliefert. Darin befinden sich zwei Disketten.
Beide sind beidseitig beschrieben. Eine ist jeweils die Sicherheitskopie der anderen, falls mal eine Version kaputtgeht, für die Sie übrigens ohne weiteres Ersatz anfordern können. Sie müssen nur diese nicht mehr funktionierende Original-Disk mit Ihrer Reklamation einsenden.
Die erste Diskettenseite enthält das Betriebssystem, auf der Rückseite sind der Druckertreiber und die Unterprogramme GeoPaint und GeoWrite abgespeichert. Diese Files können Sie nach dem Start von Geos 128 jederzeit kopieren, nicht aber die Vorderseite, also das Betriebssystem (da hat sich im Vergleich zum Urvater Geos 64 nichts geändert).
Wesentliche Verbesserungen zu Geos 64 sind in mehreren Details zu finden. So nutzt der installierte Maustreiber die entsprechende Maus (Commodore Maus 1531, NCE-Maus) wirklich im Maus-Modus, der viel präziser und schneller ist als die gewohnte Joystick-Simulation.

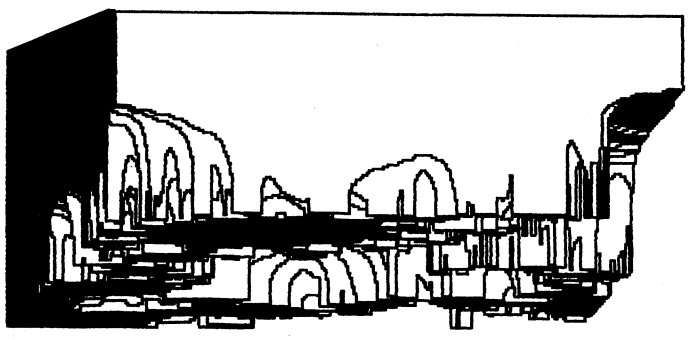
SCHWÄCHEN BEI DER 80-ZEICHEN-GRAFIK

Der 80-Zeichen-Bildschirm offenbart sich durch seine Übersichtlichkeit sehr anwenderfreundlich, allerdings können Sie durch Anklicken des entsprechenden Symboles auch auf 40-Zeichen-Darstellung umschalten (oder umgekehrt). Das empfiehlt sich vor allem bei GeoPaint, wenn Sie noch Grafikfiles von Geos 64 verwenden möchten.

Auch ein Nachteil von GeoPaint im 80-Zeichen-Bildschirm soll nicht verschwiegen werden: Kreise erscheinen als aufrecht stehende Ellipsen. Nicht so im 40-Zeichen-Modus, dort erscheint ein echter Kreis, ebenso auf dem Drucker. Das ist natürlich davon abhängig, ob er die Ausgabe 1:1 beherrscht.

Das wohl absolut beste – neben der Möglichkeit der Arbeit mit der Benutzeroberfläche – ist die Tatsache, daß Sie problemlos eine RAM-Erweiterung für den C128 (beispielsweise die 1750) verwenden können. Über ein beiliegendes Konfigurationsprogramm läßt sich die Speichererweiterung bestens als RAM-Floppy benützen.
Da mein Laufwerk „nur" eine Floppy 1570 ist, habe ich diese RAM-Floppy so eingestellt, daß ein imaginäres Laufwerk B als Floppy 1541 existiert und das Laufwerk A, meine 1570, zusätzlich von einer RAM-Disk 1541 schattiert wird.
Schattieren heißt: Einmal geladene Files einer Diskette werden in der schattierten RAM-Disk abgelegt und beim Aufruf nicht von der Diskettenstation, sondern aus der „schattierten" RAM-Floppy geladen. Das geht natürlich wahnsinnig schnell.





SPEICHERERWEITERUNG – KEIN PROBLEM

Die RAM-Disk im Laufwerk B nutzen Sie am besten, indem Sie die Arbeitsdiskette sofort dorthin kopieren. Bei einer vollen Disk mit etwa 165 KByte dauert das nur 60 Sekunden – eine Super-Arbeitsgeschwindigkeit. Nur sollten Sie daran denken, neu geschaffene Files und Daten am Ende einer Computer-Sitzung zu sichern.
Als „echtes" Laufwerk läßt sich jede gängige Commodore-Floppystation anschließen: 1541, 1570, 1571 und auch die 1581.

WANN GIBT'S ZUSATZPROGRAMME?

Bleibt abzuwarten, ob sich was auf dem Zubehörsektor für Geos 128 bewegt. Angeblich sollen die neuen Geos-64-Zusatzprogramme auch in einer 128er-Version erscheinen. Warten wir's ab.

Jürgen Heinisch/hb □

MASCHINENSPRACHE

Tausendfach schneller: Ihr Commodore kann mehr, als Sie ahnen

Mehr als eine bloße Übersicht über Befehlssatz, Adressierungsarten und Flagbeeinflussungen bietet diese Einführung. Im Vordergrund stehen Übung und Praxis. Dafür sorgt das Programm CPU-TRAINER, mit dem CPU-Befehle direkt eingegeben und auf ihre Wirkung beobachtet werden können.

Viele Probleme lassen sich in BASIC nur unzureichend lösen. Oft ist BASIC zu langsam oder total ungeeignet. Der direkte Weg, der Griff zur Maschinensprache, ist vielen noch versperrt, da die nötigen Informationen fehlen. Sollte dem einen oder anderen der Befehlssatz der CPU-8501 oder 6502 bereits bekannt sein, so hilft dieses herzlich wenig, wenn er das Betriebssystem des Rechners nicht kennt.
Ohne die Kenntnis wichtiger I/O-Adressen und -Routinen läßt sich nicht einmal ein einziges Zeichen von der Tastatur einlesen oder auf den Bildschirm bringen. Wir besprechen daher auch die wichtigsten Kernel-Routinen.
Sie lernen, wie ein Ein- oder Ausgabekanal definiert werden kann, wie Sie Zeichen auf den Drucker senden, wie Sie etwas auf Kassette oder Diskette ausgeben können, und wie Sie sie auch wieder von dort einlesen. Wenn die CPU auch keine Multiplikation kennt, mit einem bestimmten Algorithmus läßt sie sich verwirklichen. Mit dem eingebauten Maschinenmonitor können wir kleine Programme schreiben. Vom CPU-TRAINER aus können wir sie aufrufen und deren Ergebnisse kontrollieren.
Wie Sie sehen, ist der Rahmen unserer Einführung sehr weit gespannt. Wenn Sie diese gewissenhaft durcharbeiten, sind Sie bereits in der Lage, selbst kleine Maschinenprogramme zu verfassen.

I. ZU BEACHTEN BEIM C64

Da der C64 leider keinen eingebauten Maschinensprachemonitor besitzt, kann mit diesem Rechner nicht alles nachvollzogen werden.
Mit dem Programm CPU-Trainer, das sowohl für den C128 als auch für den C64 vorliegt, können CPU-Befehle direkt eingegeben werden. Mit dem Befehl BRK landet der C128 im Maschinenmonitor, der C64 dagegen nur im BASIC-Editor. Die mit dem Maschinenmonitor durchgeführten Operationen und einige kurze Programmbeispiele können mit dem C64 deshalb nicht nachvollzogen werden.
Da nur ein geringfügiger Teil dieses Artikels den Maschinenmonitor des C128 behandelt, entgeht dem C64-User nicht allzuviel. Die Beschreibung des Programmes CPU-Trainer bezieht sich auf die C128-Fassung. Um den direkten Vergleich der beiden Fassungen zu erleichtern, wurde auf eine Neunumerierung der C64-Fassung verzichtet.

II. DER CPU-TRAINER

Dieses Programm wird uns durch unseren ganzen Einführungskurs begleiten. Es ermöglicht die Direkteingabe von CPU-Befehlen.
Das Programm besitzt einen Maschinensprache- und einen BASIC-Teil. Der in den *DATA-Zeilen 130 bis 320* abgelegte Maschinenteil wird durch die *Zeilen 110 und 120* im Bereich von $2F00 bis $2F74 abgelegt. Er enthält am Anfang die Routine zur Übernahme der Registerinhalte in die CPU, zur Ausführung des CPU-Befehles und zum Zurückschreiben der CPU-Register in die dafür vorgesehenen Speicherstellen.
Die restlichen Routinen sorgen für die Sonderbehandlung der in *Zeile 1260* vermerkten Befehle. In *Zeile 330* werden die für die Registerinhalte reservierten Speicherstellen mit den Inhalten der CPU-Register oder mit dem Wert 127 initialisiert.
Ab *Zeile 370* beginnt der eigentliche BASIC-Teil. Es werden Variablen, die für die Bildschirmausgabe wichtig sind, die entsprechenden Inhalte zugewiesen. Interessant sind hier die*Zeilen 480 und 490,* durch die die Inhalte der Variablen A$(0) bis A$(6) in *Zeile 380 bis 440* umgewandelt werden.
Die auf jedem Drucker darstellbaren Zeichen 0 und a bis j werden durch die in den *DATA-Zeilen 450 und 460* vermerkten Codes ersetzt, die nachher auf dem Bildschirm erscheinen sollen.
Der Bildschirmaufbau – *Zeile 570 bis 630* – begnügt sich mit wenig Programmzeilen, da geschickt auf die Bildschirmvariablen und Unterprogramme zurückgegriffen wird.
Die Eingabe in den *Zeilen 680 bis 760* wurde nicht durch einen INPUT-Befehl realisiert.
Auffallend ist, daß auch kein GET- oder PRINT-Befehl im Programm vorkommt. Anstelle der Befehle, die auf ein definiertes Ein- oder Ausgabegerät zugreifen, oder dieses umdefinieren, verwendeten wir neben dem CHAR-Befehl die SYS-Aufrufe SYS61167 und SYS61313. SYS 61167 ersetzt den GET-Befehl. Das Zeichen wird unabhängig vom aktuellen Eingabegerät immer von der Tastatur eingelesen. Das abgefragte Zeichen wird von der Systemroutine im Akku abgelegt.
Nach dem SYS-Aufruf befindet es sich daher in Speicherstelle 6, aus der wir es mit einem PEEK hervorholen können. Statt PRINT verwendeten wir SYS61313. Es kann nur ein einziges Zeichen ausgegeben werden. Dessen ASCII-Wert ist vorher mit einem POKE in die Speicherstelle 6 zu bringen.
Der SYS-Aufruf lädt den Akku mit diesem Wert, die Systemroutine gibt das Zeichen unabhängig vom aktuellen Eingabegerät immer auf den Bildschirm aus. So wurde dafür Sorge getragen, daß auch bei umdefinierten Ein- und Ausgabegeräten unsere Bildschirmausgaben immer auf den Bildschirm gelangen und auch bei umdefiniertem Eingabegerät die Abfrage unserer Eingabe funktioniert.
Nach einer Eingabe definieren wir uns in den *Zeilen 830 und 840* ein leeres Fenster. Bildschirmausgaben, die wir durch ein Maschinenprogramm veranlassen, sollten die Darstellung der CPU-Register nicht beeinflussen. Beim Sprung in den Monitor mit BRK soll uns der ganze Bildschirm zur Verfügung stehen *(Zeile 820).*

Nach der Befehlsabarbeitung sorgt *Zeile 680* wieder für den neuen Bildschirmaufbau. In *Zeile 850* wird durch den GOSUB-Befehl die eigentliche Abarbeitung veranlaßt. In *Zeile 1180* findet eine Verzweigung statt, wenn es sich nicht um einen Ein-Byte-Befehl handelt. Die Unterroutine in *Zeile 1050* untersucht, ob ein existierender Befehl vorliegt und übergibt in diesem Falle einen weiter zu verarbeitenden Wert.
Dieses ist im Normalfall der Operationscode für die CPU oder bei der Sonderbehandlung in *Zeile 1240 bis 1260* die Adresse der anzuspringenden Routine. In *Zeile 1200* wird der Operationscode an die richtige Stelle in der Maschinenroutine eingefügt, und nachher die Routine mit SYS 12032 aufgerufen. Da für den Operationscode drei Byte reserviert sind, werden die verbleibenden zwei Stellen mit dem NOP-Code gefüllt, der die CPU zu keiner Operation veranlaßt.
In *Zeile 1380* bei den Zwei-Byte-Befehlen folgt dem Operationscode ein Datenparameter, daher ist hier nur noch ein NOP-Code erforderlich. In den *Zeilen 1550 bis 1570* bei den Drei-Byte-Codes folgen dem Operationscode zwei Adreß-Byte.

III. DIE ERSTEN SCHRITTE

Die Datenübergabe vom Speicher zu den Registern, von den Registern zum Speicher, den Datentransfer zwischen den Registern, und das Retten von Registerinhalten auf den Stapel bekommen wir mit einigen wenigen Befehlen in den Griff.
Fünf der sechs CPU-Register sind auf dem Bildschirm dargestellt. Die Inhalte können wir in binärer, hexadezimaler und dezimaler Form betrachten. Ein Fragezeichen signalisiert, daß das Programm offensichtlich auf eine Eingabe wartet. Wenn wir jetzt CPU-Befehle in mnemonischer Form eingeben, können wir beobachten, wie die CPU-Register auf unsere Befehle reagieren. Machen wir uns also mit der Materie vertraut und starten unsere ersten Versuche.

Laden und Speichern

Wir geben ein:

```
LDA #$01
LDY #$02
LDY #$03
```

Wie wir wohl vermutet haben, zeigen unsere ersten drei Register die Werte eins, zwei und drei. Die Adressierungsart, die durch #$ gekennzeichnet wird, und die den darauffolgenden Wert in das Register lädt, heißt *unmittelbare Adressierung.*
Es existieren auch Befehle, die für sich allein stehen können, ohne daß noch irgendwelche Daten oder sonstige Parameter zu folgen brauchen. Diese Adressierungsart heißt *implizite Adressierung.* Ein sehr interessanter Befehl ist der Break, der mit BRK abgekürzt wird. Geben Sie doch einmal ein:

```
BRK
```

Das Bild auf dem Bildschirm hat sich auf einmal total verändert. Wir befinden uns jetzt nicht mehr in unserem BASIC-Programm, sondern im Maschinen-Monitor, der in unseren Rechner von Haus aus eingebaut ist. Der BRK-Befehl löst einen Software-Interrupt aus, der beim C16/116/Plus4 und auch beim C128 zum Aufruf des Monitors

SPRUNG IN DEN MONITOR

führt. Daß nichts mehr von unserem früheren Bildschirminhalt zu sehen ist, dafür kann der Monitor nichts. Das CPU-Programm war so frei, für einen sauberen Bildschirm zu sorgen. Auch der Maschinen-Monitor zeigt uns die Registerinhalte, wenn auch nur in hexadezimaler Form allein.
Ein Unterschied fällt sofort ins Auge. Gleich an erster Stelle findet sich der Programmzähler, der mit PC für *Programm Counter* abgekürzt ist. Dieses Register ist nicht nur ein Byte, sondern zwei Byte breit. Schließlich muß es alle Adressen von $0000 bis $FFFF erreichen können. Zur Zeit zeigt es auf eine Stelle im Maschinenteil unseres Programmes, wo es anschließend weitergeht, wenn wir mit einem Go dafür sorgen. Wir geben ein:

```
G
```

Nach dem Druck der RETURN-Taste zeigt der Bildschirm wieder das vertraute Bild mit den CPU-Registern. Sehr kurz war der Blick in den Monitor. Vielleicht ist Ihnen bereits noch eine zweite kleine Abweichung aufgefallen. Prägen Sie sich doch einmal den Inhalt des Stapelzeigers ein und gehen dann wieder mit BRK in den Monitor.
Der Stackpointer SP weist einen ganz anderen Wert auf. Das CPU-Programm simuliert einen Stapelzeiger mit anderem Wert, damit beim Herumprobieren ein Programmabsturz nicht so leicht verursacht werden kann.
Ein sehr wichtiger Bestandteil des Rechners neben der CPU ist der Speicher. Mit dem Monitor können Sie sehr leicht Speicherbereiche betrachten. Wir geben ein:

```
M3000
```

Von der Adresse $3000 ausgehend, wird ein Speicherabschnitt in hexadezimaler Form dargestellt. Rechts finden wir invers die ASCII-Darstellung davon. Da ein Rechner auch Texte speichern soll, ist es interessant, zu sehen, wo ein Text abgelegt ist.
Da das CPU-Programm nicht bis $3000 reicht, steht uns ab dieser Adresse der Hauptspeicher des Rechners fast uneingeschränkt zur Verfügung. Wir sorgen zunächst einmal für einen sauberen Speicher mit

```
F3000,3FFF,0
```

Der Bereich von $3000 bis $3FFF ist nun vollständig mit Nullen gefüllt. Wir sehen uns wieder den ersten Abschnitt an $3000 mit

```
M3000
```

Wenn nach den folgenden Manipulationen andere Werte hier drinstehen sollten, kann keiner mehr sagen, dies wäre schon vorher der Fall gewesen. Wir können nun mit *G* getrost den Monitor wieder verlassen.
Mit LDA, LDX und LDY nur unmittelbar Werte in die Register zu schieben, ist uns zu wenig, denn die Daten, um die es geht, befinden sich in der Regel irgendwo im Hauptspeicher des Rechners. Daten können nur durch die CPU verarbeitet werden. Daher muß es auch Befehle geben, die die Daten vom Speicher in die CPU und wieder von der CPU in den Rechner befördern.
Beginnen wir mit dem Weg von der CPU in den Speicher. Uns stehen die Speicher- oder auch die Store-Befehle STA, STX und STY zur Verfügung. Diesen Speicherbefehlen muß eine Adreßangabe folgen, die angibt, wo im Hauptspeicher die Registerinhalte gespeichert werden sollen. Wir geben ein:

```
LDA #$54
STA $3000
LDA #$45
STA $3001
LDA #$53
STA $3002
LDA #$54
STA $3003
```

Diese Adressierungsart, bei der dem Operationscode die Speicheradresse

folgt, heißt *absolute Adressierung.* Gekennzeichnet wird sie durch das $-Zeichen, gefolgt von einer vierstelligen HEX-Zahl. Unsere CPU kennt auch noch weitere Adressierungsarten, bei denen nur ein einziges Byte in Form von zwei Hexziffern anzugeben ist, oder solche, bei denen das X- oder Y-Register mit zur Adreßbildung verwandt wird. Um das Programm nicht allzusehr aufzublasen, wurde darin auf weitere Adressierungsarten jedoch verzichtet.
Um den Erfolg unseres Speichertests zu begutachten, sollten Sie sich im Monitor nochmals den Bereich ab $3000 ansehen. Wie dieses vor sich geht, dürfte Ihnen inzwischen bereits bekannt sein. Wir finden die erwarteten Werte, die in ASCII-Darstellung das Wort TEST ergeben.
Nicht nur die Speicherbefehle, sondern auch die Ladebefehle lassen sich absolut adressieren. Spaßeshalber wechseln wir einmal das Register:

```
LDX $3000
STX $3008
LDX $3001
STX $3009
LDX $3002
STX $300A
LDX $3003
STX $300B
```

Wir haben damit die von $3000 bis $3003 abgelegten Werte nach $3008 bis $300B übertragen, wie uns die Überprüfung mit dem Monitor bestätigt.

Transferieren zwischen Registern

Datenübertragung von CPU-Register nach Speicher und umgekehrt kennen wir jetzt schon. Irgend etwas scheint aber noch zu fehlen. Wenn wir einen Wert verändern, so kommt es oft vor, daß wir gerne den ursprünglichen Zustand wiederherstellen würden. Im Akku könnte zum Beispiel ein Wert stehen, den wir für irgendwelche Zwecke benötigen. Vorher sollen aber noch ein paar Rechenoperationen erfolgen, die leider den Akkuinhalt verändern.
Eine Lösungsmöglichkeit ist mit unserem bisherigen Wissen bereits denkbar. So könnten wir den Akkuinhalt einfach an irgendeiner Adresse abspeichern und unseren Wert, sobald er wieder gebraucht wird, erneut einladen. Doch warum sollen Daten, die wir bereits in der CPU bereit haben, erst wieder in das RAM geschrieben und dann wieder hergeholt werden?
Speicherzugriffe kosten mehr Zeit als reine CPU-Operationen und außerdem mehr Platz, da immer noch eine Speicheradresse anzugeben ist. Es sollte doch auch Möglichkeiten geben, Daten direkt von einem Register in das andere zu transferieren. Diese Transferbefehle gibt es selbstverständlich. Um Daten vom Akku in die Intexregister X und Y zu schaufeln, gibt es die Befehle:

```
TAX
TAY
```

Den umgekehrten Weg ermöglichen:

```
TXA
TYA
```

Scheuen Sie sich nicht, dies ruhig auszuprobieren. Solange nur ein oder zwei Werte gerettet zu werden brauchen und die anderen Register nicht für andere Zwecke benötigt werden, ist dieses Verfahren ein brauchbarer Weg.
Oft werden aber Unterroutinen angesprungen, die alle Registerinhalte verändern. In diesen Unterroutinen kann wiederum der Bedarf bestehen, neue Registerinhalte zu retten. Als Ausweg scheint in diesem Falle nur das Abspeichern in dafür vorgesehenen Speicherplätzen gangbar zu sein. Jede Unterroutine bräuchte also wieder ihre besonderen Speicherstellen zum Retten der Register.
Nicht leicht ist es wohl, da noch den Überblick zu behalten. Für rekursive Programmierung, bei der ein Unterprogramm sich selbst mehrmals aufruft, bringt auch dieses keine Lösung. Die vorgesehenen Speicherstellen würden bei jedem rekursiven Aufruf neu überschrieben werden, wodurch die früheren Werte verlorengehen. Das Stapelregister sorgt hier für Abhilfe.

Der Prozessorstapel

Laden Sie einmal einen beliebigen Wert in den Akku. Um diesen zu retten, geben wir jetzt den Befehl ein:

```
PHA
```

Wenn Sie dabei auf das Stapelregister geachtet haben, haben Sie sicherlich eine Veränderung wahrgenommen. Wenn nicht, so beobachten wir es bei einem erneuten Versuch. Laden Sie bitte den Akku mit einem neuen Wert und retten Sie diesen wieder mit PHA.
Der Wert des Stapelzeigers wurde jedesmal um den Betrag Eins verringert. Wenn wir unsere geretteten Werte vom Stapel wieder abheben, so beobachten wir umgekehrt ein Erhöhen des Stapelzeigers. Vorher laden wir noch schnell den Akku mit einem anderen Wert, damit wir das Erscheinen der alten Werte sehen. Wir geben ein:

```
PLA
```

Es erscheint der zuletzt gerettete Wert. Nach einem erneuten PLA haben wir auch den zuerst geretteten Wert wieder. Der Stapelzeiger müßte nun wieder, wie am Anfang, den Wert 7F aufweisen. Um zu untersuchen, was geschehen ist, geben wir ein:

```
LDX $017F
LDY $017E
```

Wir sehen, vielleicht auch zu unserer Verblüffung, die ehemals geretteten Werte nun in den Indexregistern stehen.
Der Stapel funktioniert folgendermaßen: Für den Stapel ist ein RAM-Bereich von $0100 bis $01FF reserviert. Der Inhalt des Stapelzeigers gibt das Low-Byte der Adresse an, in die der Registerinhalt bei einem PUSH-Befehl geschrieben wird.
Der Stapelzeiger wird zusätzlich um den Wert Eins erniedrigt, so daß er auf die nächstniedrige Adresse weist. So können verschiedene Daten hintereinander gerettet werden, ohne daß sie sich gegenseitig ins Gehege kommen.
Beim PULL-Befehl wird der Stapelzeiger wieder um Eins erhöht, und der unter der Adresse mit High-Byte 01 und dem Stapelzeigerinhalt als Low-Byte in das entsprechende Register geladen. Nicht nur der Akku kann „gestapelt" werden, sondern auch das Statusregister. Die Befehle hierfür sind:

```
PHP
PLP
```

Wir können somit alle Registerinhalte auf den Stapel retten:

```
PHP
PHA
TXA
PHA
TYA
PHA
```

Abheben müssen wir den Stapel in umgekehrter Reihenfolge:

```
PLA
TAY
PLA
TAX
PLA
PLP
```

Die Stapelbefehle erlauben uns gar, mit dem Statusregisterinhalt genauso wie mit sonstigen Daten umzugehen. Wir bekommen die Werte in den Griff mit:

```
PHP
PLA
```

Das Zurückübertragen macht auch keine besonderen Schwierigkeiten:

```
PHA
PLP
```

Sogar das Stapelregister bekommen wir mit zwei Transferbefehlen völlig in den Griff:

```
TSX
TXS
```

Übersicht der besprochenen Befehle

Laden	LDA,	LDX,	LDY
Speichern ·	STA,	STX,	STY
Transferieren	TAX	TAY,	TSX
	TXA,	TYA,	TXS
Stapeln	PHA,	PHP	
	PLA,	PLP	

IV. SCHALTEN MIT DER CPU

Einzelne Bit lassen sich mit den Befehlen ORA, AND und EOR setzen.

Sollen einzelne Bit eines Byte auf einen bestimmten Wert gesetzt werden, ohne Beeinflussung der übrigen Bit, so bedarf es einiger Befehle, die eine solche Bit-Manipulation zulassen. Wir geben ein:

```
LDA #$00
LDA #$01
LDA #$02
LDA #$04
LDA #$08
LDA #$10
LDA #$20
LDA #$40
LDA #$80
```

Ein bestimmtes Bit läßt sich mit dem Ladebefehl leicht setzen. Der Nachteil ist, daß die bereits vorhandenen Bit leider auch verändert werden. Zum Verändern einzelner Bit eignet sich der Befehl EOR. Probieren Sie doch einmal:

```
LDA #$99
EOR #$01
EOR #$02
EOR #$04
EOR #$08
EOR #$10
EOR #$20
EOR #$40
EOR #$80
```

Wir haben ein Bit nach dem anderen umgedreht. Es lassen sich natürlich auch alle umdrehen mit:

```
EOR #$FF
```

Für uns könnte der EOR-Befehl bereits ausreichend sein, da wir ja sehen, ob ein bestimmtes Bit bereits den gewünschten Wert hat oder nicht. Wenn nein, wenden wir den EOR-Befehl an, wenn ja, dann eben nicht. Die CPU nachsehen zu lassen, ob ein Bit gesetzt ist oder nicht und gegebenenfalls mit dem EOR-Befehl zu reagieren, wäre wohl etwas aufwendig. Daher existieren die Befehle ORA und AND, mit denen unabhängig vom Bit-Zustand diesem in jedem Falle der Wert Eins oder Null zugewiesen werden kann:

```
LDA #$99
ORA #$01
ORA #$02
ORA #$04
ORA #$08
ORA #$10
ORA #$20
ORA #$40
ORA #$80
```

Ebenso wie beim EOR-Befehl lassen sich auch mehrere Bit setzen:

```
LDA #$81
ORA #$18
```

Die zwei mittleren Bit sind durch den ORA-Befehl hinzugekommen. Um zu wissen, welche Werte wir für den AND-Befehl brauchen, sollten wir die Werte, mit denen wir vorher geordert hatten, invertieren, und uns die Resultate merken:

```
LDA #$01
EOR #$FF
LDA #$02
EOR #$FF
LDA #$04
EOR #$FF
LDA #$08
EOR #$FF
LDA #$10
EOR #$FF
LDA #$20
EOR #$FF
LDA #$40
EOR #$FF
LDA #$80
EOR #$FF
```

Wir können nun den Akku wieder mit #$99 oder auch einem anderen Wert laden und das Nullsetzen der Bit beobachten.

```
LDA #$99
AND #$FE
AND #$FD
AND #$FB
AND #$F7
AND #$EF
AND #$DF
AND #$BF
AND #$7F
```

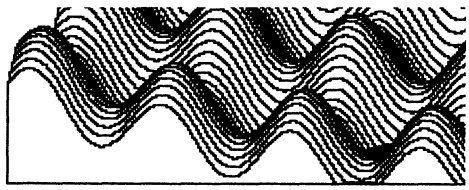
Es lassen sich wieder mehrere Bit auf einmal setzen. Wir laden hierzu das X-Register.

```
LDX #$7E
```

Wir können sehen, daß die äußeren beiden Bit den Wert Null haben. Beim Undieren mit diesem Wert würden folglich die äußeren Bit auf Null gesetzt.

```
LDA #$99
AND #$7E
```

Probieren Sie ruhig noch ein wenig mit den Befehlen EOR, ORA und AND herum, denn Übung macht ja bekanntlich den Meister. Sie können auch die absolute Adressierung verwenden, bei welcher der Inhalt einer Speicherzelle mit dem Akkuinhalt verknüpft wird.



V. DIE LOGIK DER CPU

CPU-Operationen beeinflussen bestimmte Flags des Statusregisters. Acht bedingte Sprungbefehle reagieren auf die Zustände von vier Statusflags.

Viele sprechen von der Intelligenz des Computers. Alles, was eine CPU kann, ist jedoch nur auf ganz gewisse Schalterstellungen hin zu verzweigen. Damit diese Verzweigungen richtig stattfinden, ist die Intelligenz des Programmierers gefordert. Die Verzweigungsbefehle sind:

```
BEQ
BNE
BPL
BMI
BCS
BCC
BVS
BVC
```

Diese Befehle können Sie mit unserem CPU-Programm nicht eingeben. Sie sind nicht für sich, sondern nur in einem Programm brauchbar. Wir können uns jedoch mit den Schaltern befassen. Ein Register wurde bisher noch kaum besprochen, das Statusregister. Es ist sozusagen das Auge des Computers. Nur auf das, was das Statusregister anzeigt, kann die CPU mit Verzweigungen reagieren. Set- und Clear-Befehle erlauben uns die gezielte Einflußnahme auf einzelne Statusflags.

SEC
CLC

Das Carry-Flag ist das einzige Flag, das wir mit Set und Clear sowohl an- und ausschalten können und auf welches außerdem noch Verzweigungsbefehle reagieren, die Befehle BCS und BCC. Daher wird dieses Flag oft gesetzt, um den erfolgreichen Ablauf einer Operation zu signalisieren.
Zurückgekehrt aus einem Unterprogramm, das zum Beispiel Daten auf ein externes Gerät ausgeben sollte, läßt sich mit dem Branch-Befehl auf das Carry-Flag reagieren, für den Fall, daß die Ausgabe nicht geklappt hätte, da vielleicht der Drucker gar nicht angeschaltet war.

SEI
CLI

Mit diesen Befehlen können wir das Interrupt-Flag setzen und löschen. Wie wir wissen, kommt es zu Zwecken der Bildschirmausgabe und der Tastaturabfrage immer wieder zu Unterbrechungen des laufenden Programms. Dies kann aus irgendwelchen Gründen unerwünscht sein. Ist gar die RAM-Bank eingeschaltet und der Prozessor bearbeitet hierin ein Programm, so würde der Interrupt einen Absturz bewirken, wird doch durch diesen wieder die ROM-Bank aktiviert. Anstelle des Rücksprungs in die unterbrochene Routine, landet der Programmzähler dann irgendwo im Betriebssystem oder im BASIC-Interpreter, so daß dann nur mehr der Himmel weiß, was anschließend stattfindet.

SED
CLD

Damit eingegebene Zahlen nicht mühsam erst in Hex-Zahlen umgerechnet werden müssen, wenn wir Rechenoperationen vornehmen wollen, können wir dem Rechner durch Setzen des Dezimal-Flags mitteilen, daß er seine Additionen und Subtraktionen nun im sogenannten BCD-System vornehmen soll.

CLV

Bei Rechenoperationen mit vorzeichenbehafteten Zahlen ist das Überlauf-Flag von Interesse. Es sollte daher auch wieder zurückgesetzt werden können.

Zusammenfassung der Set- und Clear-Befehle

```
SEC
CLC
SEI
CLI
SED
CLI
CLV
```

Das Zero-Flag

Wissen Sie, warum viele Programmierer das Ende eines Datensatzes mit einer Null kennzeichnen?
Laden Sie den Akku oder irgend ein anderes Register mit einer von Null verschiedenen Zahl, dann mit der Zahl null. Beim Beobachten des Statusregisters bemerken wir eine Veränderung von Bit eins, des zweiten Bit von rechts. Ist der geladene Wert Null, so zeigt das Zero-Flag eine Eins, im anderen Falle eine Null. Mit den Branchbefehlen BEQ und BNE kann daraufhin entsprechend verzweigt werden. Branch Equal ist gleichbedeutend mit Branch Zero und heißt soviel wie: „Verzweige, wenn Zero-Flag gleich eins".
Branch Not Equal verzweigt im umgekehrten Falle.

Das Vorzeichenflag

Datensätze in Standard-ASCII, wobei nur Werte bis #$7F vorkommen, lassen sich sogar noch kürzer abspeichern. Ein extra Byte für die Endemarkierung kann entfallen. Es braucht nur Bit sieben des letzten Zeichens auf Eins gesetzt werden. Das Vorzeichenflag zeigt dieses an.

```
LDA #$01
LDA #$81
```

Bit sieben, das ganz links wiedergegeben wird, ist das Vorzeichenflag. Es heißt so, da beim Rechnen mit negativen Zahlen definiert wird, daß Zahlen von #$00 bis #$7F positiv seien und Zahlen von #$80 bis #$FF negativ. Bit sieben des Statusregisters gibt also Bit sieben eines geladenen oder auf Grund einer sonstigen Operation erhaltenen Wert eines Registers oder Speicherinhaltes wieder.
Bit sieben läßt sich ganz einfach mit dem Vorzeichenflag abfragen. Der Prozessor sollte aber auch die restlichen Bit aus einem Register oder aus einer Speicherzelle abfragen können. Einen Befehl, mit dem wir dieses leicht bewerkstelligen können, kennen wir bereits.

```
LDA #$01
AND $0001
LDA #$02
AND $0001
LDA #$04
AND $0001
LDA #$08
AND $0001
LDA #$10
AND $0001
LDA #$20
AND $0001
LDA #$40
AND $0001
LDA #$80
AND $0001
```

Wir setzen im Akku alle Bit auf Null, mit Ausnahme desjenigen, das wir abfragen wollen. Durch den AND-Befehl werden alle Bit, die uns nicht interessieren, zu Null. An der Stelle, die wir mit einer Eins maskiert hatten, gelangt das gewünschte Bit aus der Speicherstelle in den Akku. Wenn dieses Null war, so ist der gesamte Akku Null und das Zero-Flag ist gesetzt. Im anderen Falle wissen wir, daß uns eine Eins vorliegt.
Der *AND*-Befehl ändert auch den Akku-Inhalt. Manchmal wäre es wünschenswert, wenn dieser uns erhalten bliebe. Außerdem kann mit dem AND-Befehl nur ein einziges Bit über das Zero-Flag abgefragt werden. Ein paar Möglichkeiten mehr bietet der Bit-Befehl. Um Bit sieben abzufragen, braucht nicht extra der Wert eines Registers durch Hereinladen zerstört werden. Zusätzlich zu Bit sieben im Vorzeichenflag wird Bit sechs im Überlauf-Flag erfaßt. Die restlichen Bit können auf dieselbe Art, wie wir es bereits vom AND-Befehl her kennen, abgefragt werden.
Die Besonderheit des *BIT*-Befehls ist, daß zwar auch eine Undierung stattfindet, jedoch das Ergebnis nur im Setzen des Zero-Flags sichtbar wird, und so der Akkuinhalt unverändert bleibt.

```
LDA #$01
BIT $0001
LDA #$02
```

```
BIT  $0001
LDA #$04
BIT  $0001
LDA #$08
BIT  $0001
LDA #$10
BIT  $0001
LDA #$20
BIT  $0001
```

Wir haben die Abfragen jetzt nur bis Bit fünf ausgeführt, da das Statusregister uns sowieso immer die Werte von Bit sechs und sieben wiedergab. Da das Überlauf-Flag auf Eins steht, gibt uns dieses die Gelegenheit, den Befehl *CLV* auszuprobieren. Es funktioniert wie gewünscht.
Wir können den Wert #$00 abfragen, wir können abfragen, ob ein Wert größer als #$7F ist, sogar vor einzelnen Bit brauchen wir nicht zu kapitulieren. Dennoch fehlt noch irgend etwas. Was tun wir denn, wenn Datensätze nicht mit einer Null abgeschlossen sind und auch nicht mit einem Wert größer als #$7F, sondern schlicht mit einem Carriage-Return oder einem sonstigen Wert?
Wir müßten demnach vergleichen können, ob wir einen bestimmten Wert vor uns haben oder nicht. Dafür gibt es den *Compare*-Befehl, der zwei Werte miteinander vergleicht. Vergleichen können wir die Registerinhalte unmittelbar mit einem bestimmten Wert oder aber mit dem Inhalt einer beliebigen Speicherzelle. Die Compare-Befehle sind:

```
CMP
CPX
CPY
```

CMP vergleicht mit dem Akkuinhalt, die beiden anderen Befehle sind wohl nicht schwer zu erraten.

```
LDA #$04
CMP #$03
```

Das Zero-Flag zeigt eine Null.

```
LDA #$04
CMP #$04
```

Das Zero-Flag zeigt eine Eins zum Zeichen, daß die beiden verglichenen Werte übereinstimmen. Einen Vergleich können wir uns so vorstellen, daß vom Wert im Register der verglichene Wert subtrahiert wird, ohne daß der Wert im Register verändert wird. Wie beim Bit-Befehl erscheint das Resultat nur im Statusregister.

Die Namensgebung der Befehle BEQ und BNE ist auf diesen Vergleich zurückzuführen. Stimmen die beiden Wert überein, so ist die Differenz der Werte Null, das Zero-Flag somit gesetzt und BEQ verzweigt entsprechend. *Branch Equal* heißt somit: „Verzweige, wenn Gleichheit besteht."
Von BASIC her kennen wir:

```
IF A=B
IF A<>B
IF A<B
IF A>B
```

Die ersten beiden Fälle sind in Maschinensprache bereits abgehandelt. Die letzten beiden fehlen uns noch. Wir beobachten bei den folgenden Vergleichen einmal das Carry-Flag, ob hier ein Unterschied wahrnehmbar ist.

```
LDA #$04
CMP #$03
CMP #$05
CMP #$04
```

Ist der Registerinhalt größer oder gleich dem zu vergleichenden Wert, so wird das Carry gesetzt, im anderen Falle gelöscht.
Mit den Branch-Befehlen *BCS*, Branch Carry SET, und *BCC*, Branch Carry Clear, können wir nach Wunsch verzweigen. Beim *CMP*-Befehl wird auch noch das Vorzeichenflag beeinflußt. Eine große Bedeutung kommt diesem Umstand wohl nicht zu. Wir können aber leicht noch einmal nachprüfen, ob Bit sieben in einem Register gesetzt ist, wenn zwischendurch auch schon andere Ladeoperationen erfolgt sein sollten.

```
LDA #$90
LDX #$01
CMP #$00
```

Wir besitzen jetzt ein fundiertes Wissen über die Programmlogik unseres Rechners. Ob wir unser Gerät Rechner nennen dürfen, werden uns die Rechenbefehle zeigen.

VI. DIE RECHENOPERATIONEN DER CPU

Addition, Subtraktion, Rotations-, und Schiebe-, Ikrementier- und Dekrementierbefehle sind Operationen, mit denen der Prozessor aufwartet. Multiplikationen und höhere Rechenoperationen lassen sich mit diesen Befehlen realisieren.

Ihr Taschenrechner wartet mit Grundrechenarten, Wurzeln, Potenzen und vielen mathematischen Funktionen auf. BASIC besitzt ebenso eine Menge von Rechenoperationen. Der Rechner muß also eine Menge können. Mancher wird wohl sehr verblüfft sein, wie wenig die CPU tatsächlich nur kann. Sie wartet nicht mit vielstelligen Dezimalzahlen und Exponenten, sondern nur mit Byte-weiser Addition und Subtraktion auf.
Multiplikation und Division sind für sie bereits Fremdwörter. Solche Wörter wie Sinus oder Cosinus getrauen wir uns gar nicht mehr in den Mund zu nehmen. Wenn auch der arithmetische Befehlssatz nicht sehr umfangreich ist, so läßt sich doch mit diesem alles programmieren, was das Herz begehrt.

Addition und Subtraktion

Wir haben die Wahl zwischen binärem oder BCD-System. Vor dem Addieren ist das Carry-Flag auf Null zu setzen, vor dem Subtrahieren auf Eins.

```
CLC
LDA #$04
ADC #$03
```

Wie gewünscht, ist das Ergebnis im Akku sichtbar. Wir nehmen nun Zahlen, die geringfügig größer sind.

```
CLC
LDA #$08
ADC #$07
```

Die dezimale Darstellung zeigt den Wert 15, wie es sein soll. Die hexadezimale Darstellung wartet mit #$0F auf, was zwar noch vorstellbar ist, jedoch für uns Zehn-Finger-Menschen schon etwas unanschaulich wirkt. Doch es kommt noch schlimmer.

```
CLC
LDA #$28
ADC #$49
```

Weder die Dezimale Darstellung noch die Hex-Darstellung zeigen an, daß 28 und 49 die Summe 77 ergeben müßten. Wir und die CPU benützen eben zwei unterschiedliche Zahlensysteme. Wenn wir die Zahlen #$28 und #$49 in die Index-Register laden, sehen wir, daß #$28 eben nicht auch dezimal 28 bedeutet. Die Summe der Werte in den Indexregistern ergibt jedoch genau den Wert, den wir im Akku vorfinden. Wir brauchen nur die dezimal dargestellten Werte miteinander vergleichen. Keine Sorge, wir brau-

chen uns deshalb nicht umgewöhnen, von nun an in Hex-Zahlen zu rechnen.

HEX, DEZIMAL UND BCD

Die Schwierigkeit ist nur, daß der Rechner und wir andere Zahlensysteme verwenden. Der Programmanwender merkt davon nichts.
Der Programmierer hat dafür Sorge zu tragen, daß die dezimal eingegebenen Zahlen in Hex-Zahlen umgewandelt werden. Die CPU kann nun auf ihre Weise mit diesen Zahlen rechnen. Das Ergebnis muß wieder in das Dezimalsystem zurückübersetzt werden. Wenn der Rechner allerdings nicht einmal multiplizieren kann, so ist dieses wohl gerade kein leichtes Unterfangen. Glücklicherweise kommt uns das Statusregister mit dem Dezimalflag einen Schritt entgegen, so daß wir die Rechenleistungen gut verfolgen können.

```
SED
CLC
LDA #$08
ADC #$07
LDA #$28
ADC #$49
```

Bei der Subtraktion setzen wir das Carry-Flag.

```
SEC
LDA #$77
SBC #$49
```

Das Ergebnis entspricht auch unseren Erwartungen. Wenn der Rechner nur mehr Stellen zulassen würde! Mit dem Carry-Flag hat es noch eine besondere Bewandtnis.

```
CLC
LDA #$60
ADC #$50
```

Wir beobachten, daß das Carry-Flag nun gesetzt ist. Da die Summe aus 50 und 60 größer als 99 ist, ist ein Übertrag entstanden, der bei weiteren Additionen dazuaddiert wird.

```
TAX
LDA #$01
ADC #$02
```

Eins und zwei sind plötzlich vier, da der Übertrag aus der vorangegangenen Rechnung Berücksichtigung fand.
Betrachten wir die beiden Rechnungen im Zusammenhang, so haben wir eigentlich die Summe aus 160 und 250 gebildet. Akku und X-Register enthalten 0410. Wenn wir mehrmals hintereinander die einzelnen Stellen einer vielstelligen Zahl unter Berücksichtigung der Überträge addieren, so sind uns von der Stellenzahl her keine Grenzen gesetzt. Wir brauchen nur einen bestimmten Speicherbereich unserer Rechnenoperation vorsehen. Die jeweilige Summe aus den beiden Summanden schreiben wir wieder in den Speicher.

```
BRK
M3000
F3000,3FFF,0
M3000
```

Nach der Adreßangabe >3000 finden wir die Hexzahlen 00 00 00. Diese ändern wir in eine beliebige sechsstellige BCD-Zahl. Mit der Zeile weiter unten wollen wir genauso verfahren.

```
G
CLC
LDA $3002
ADC $300A
STA $3012
LDA $3001
ADC $3009
STA $3011
LDA $3000
ADC $3008
STA $3010
BRK
M3000
```

Wir können nun das Ergebnis betrachten. Wenn nicht vorne bei den Hunderttausendern noch ein Überlauf geschehen ist, sollte unser Ergebnis stimmen.
Für mehrstellige Subtraktionen dient ebenso wieder das Carry-Flag. Normalerweise müßte es in diesem Falle Borrow-Flag heißen. Für unsere CPU gilt, daß ein gesetztes Carry ein gelöschtes Borrow bedeutet und umgekehrt. Andere CPUs können auch eine andere Regelung besitzen. Wir versuchen uns nun an der mehrstelligen Subtraktion. Zahlen brauchen wir keine hierzu noch eingeben. Wir löschen lediglich eine Zeile im Monitor.

```
F3000,3007,0
M3000
```

Der erste Summand aus unserer Addition ist verschwunden. Von der früheren Summe ziehen wir den zweiten Summanden ab und sollten wieder die erste Zeile, wie gehabt, bekommen.

```
G
SEC
LDA$3012
SBC $300A
STA $3002
LDA$3011
SBC $3009
STA $3001
LDA$3010
SBC $3008
STA $3000
BRK
M3000
```

Für unsere Zwecke reicht das Rechnen mit positiven Zahlen vollständig. Die CPU kennt schließlich nur positive Adressen von $0000 bis $FFFF. Wenn einmal etwas errechnet werden muß, wie die Länge eines Speicherbereiches aus der Anfangsadresse und der Endadresse, so ist maximal eine zweistellige Rechnung vonnöten. Für diejenigen, die sich einmal vornehmen sollten, eine Tabellenkalkulation oder ein Buchrechnungsprogramm in Maschinensprache zu programmieren, sind auch das Vorzeichenflag und das Überlauf-Flag von Bedeutung. Wir definieren uns nun, daß Zahlen von #$00 bis #$7F positiv sein sollen und Zahlen von #$80 bis #$FF negativ, und schenken den Status-Bit sechs und sieben unsere Aufmerksamkeit. Zuerst verlassen wir aber erst einmal den Monitor und setzen das Dezimalflag zurück.

```
G
CLD
```

Wir können jetzt mit dem Rechnen beginnen.

```
CLC
LDA #$40
ADC #$50
```

Das Ergebnis #$90 ist als negative Zahl definiert, was das Vorzeichen-Flag auch angibt. Nur kann die Summe aus zwei positiven Zahlen wohl kaum negativ sein. Daher ist das Überlauf-Flag gesetzt. Es signalisiert, daß das Vorzeichen-Flag den Sachverhalt nicht richtig wiedergibt.

```
CLC
LDA #$40
ADC #$20
```

Beide Flags sind null. Die Zahl ist positiv. Das Vorzeichen-Flag zeigt es richtig an.

```
CLC
LDA #$40
ADC #$FF
```

Die Hexzahl #$FF entspricht einer

negativen Eins. Das Ergebnis ist daher positiv und richtig.

```
SEC
LDA #$10
SBC #$20
```

Das Ergebnis ist negativ, das Vorzeichen-Flag zeigt dieses an.

```
SEC
LDA #$90
SBC #$20
```

Von einer negativen Zahl wird eine positive abgezogen. Das Ergebnis muß daher negativ sein. Daß das Vorzeichen-Flag den Sachverhalt nicht richtig wiedergibt, wird vom Überlauf-Flag signalisiert.

```
SEC
LDA #$05
SBC #$90
```

Von einer positiven Zahl wird eine negative subtrahiert. Das Ergebnis muß daher positiv sein. Das Vorzeichen-Flag gibt es richtig wieder.

```
SEC
LDA #$30
SBC #$90
```

Das Ergebnis muß aus den vorher genannten Gründen wieder positiv sein. Das Überlauf-Flag widerspricht dem Vorzeichen-Flag.

```
CLC
LDA #$90
ADC #$90
```

Zu einer negativen Zahl wird eine negative Zahl addiert. Das Ergebnis muß negativ sein. Das Überlauf-Flag sagt aus, daß das Vorzeichen-Flag nicht recht hat.
Wir sprachen davon, daß die CPU nicht multiplizieren könne. Ganz richtig ist dieses jedoch nicht. Wir, die im Dezimalsystem rechnen, benötigen das Einmaleins bis zum Neunereinmaleins. Das Multiplizieren mit zehn geschieht durch einfaches Linksverschieben. Im Binärsystem, das nur die Zahlen Null oder Eins kennt, brauchen wir nur das Einereinmaleins und das Verschieben, das einer Multiplikation mit zwei gleichkommt. Mit eins multiplizieren ist keine Kunst, da dieses sich von selbst erübrigt. Für das Verschieben stehen uns diverse Befehle zur Verfügung.

```
LDA #$01
ASL
```

Der Akkumulatorinhalt wird durch den *ASL-Befehl,* Arithmetic Shift Left, um eins nach links geschoben. Wiederholen Sie den ASL-Befehl, bis die Eins aus dem Akku verschwindet. Ist dieses der Fall, erscheint ein gesetztes Bit im Carry-Flag. Nach einem weiteren Befehl ist auch dieses verschwunden. Sie brauchen den ASL-Befehl nicht jedesmal neu einzugeben. Unser Programm hat eine Wiederholautomatik. Wenn Sie nur die Return-Taste drücken, wird der zuletzt eingegebene Befehl immer wieder ausgeführt.

```
LDA #$01
ROL
```

Fast das gleiche wie der ASL-Befehl scheint der *ROL-Befehl,* **RO**tate **L**eft, zu bewirken, daß dieses jedoch nicht so ist, sehen wir, wenn die Eins aus dem Akku verschwunden und im Carry-Flag gelandet ist. Beim nächsten ROL-Befehl geht sie von dort wieder in den Akku über.

```
LDA #$80
LSR
```

Logical Shift Right bewirkt dasselbe wie ASL, nur in umgekehrter Richtung.

```
LDA #$02
ROR
ROR
ROR
```

Rotate Right funktioniert genau wie ROL in umgekehrter Richtung. Diese Befehle greifen nicht nur auf den Akku zu, sondern es lassen sich auch beliebige Speicherstellen dadurch verändern. Auf ASL und LSR könnte zur Not auch verzichtet werden, da durch Löschen des Carry-Flags dafür gesorgt werden kann, daß die Rotate-Befehle stets eine Null in den Akku oder die Speicherstelle rotieren.
Das Hereinnehmen des Carry-Flags durch die Rotate-Befehle ist äußerst nützlich. Denken wir hierzu nur einmal an eine Grafik-Hardcopy. Durch acht mal acht Bit wird das Bit-Muster eines Zeichens bestimmt. Der Bildschirm hat es zu acht Reihen mit jeweils einem Byte, der Drucker jedoch braucht es in Spaltenform. Nachdem die Matrix aus dem Bildschirmbereich herauskopiert wurde – schließlich wollen wir den Bildschirminhalt nicht zerstören –, können wir sie mit ROL-Befehlen für den Drucker zurechtschneidern. Aus jedem der acht Bildschirm-Byte rotieren wir je ein Bit heraus und rotieren es mit einem zweiten ROL-Befehl in den Akku. Nachdem der Akkuinhalt ausgegeben ist, holen wir uns nach demselben Verfahren die zweite Spalte heraus, bis alle acht Spalten ausgegeben sind.
Zum Multiplizieren laden wir jeweils eine Zahl, deren Produkt nicht größer als 255 oder #$FF betragen soll, in die Indexregister. Das Produkt soll nach der Multiplikation im Akku stehen, die Faktoren noch in den Indexregistern.

```
LDX #$06
LDY #$09
```

Es gilt nun ein geeignetes Verfahren, die Multiplikation, zu finden. Wir laden hierzu den Akku mit Null und legen die beiden Werte in zwei Speicherstellen ab, so daß wir sowohl schieben als auch addieren können.

```
LDA #$00
STX $3000
STY $3001
```

Eine Zahl können wir dadurch mit sechs multiplizieren, daß wir sie in den Akku laden. Dadurch haben wir sie einmal erfaßt. Indem wir sie nach links verschieben, haben wir sie bereits mit zwei multipliziert. Wenn wir sie noch einmal dazuaddieren, ist sie schon mit drei malgenommen. Ein abschließendes Linksverschieben bringt das gewünschte Ergebnis.
Für beliebige Zahlen, die bis zu acht Bit lang sein dürfen, benötigen wir noch ein paar weitere Befehle. Acht Bit sind zu untersuchen. Daher muß unser Rechner bis acht zählen können. Zum Zählen dienen Befehle, mit denen vorwärts und rückwärts gezählt werden kann.

```
LDY #$08
DEY
```

Bei jedem DEY wird das Y-Register um Eins erniedrigt, bis beim Erreichen von Null das Zero-Flag diesen Zustand anzeigt. Beim weiteren Decrementieren spricht das Vorzeichen-Flag an.

```
INY
```

Mit INY kann das Y-Register hochgezählt werden. Die restlichen Befehle sollen nun nicht gleich ausprobiert werden, da wir den Inhalt des X-Registers weder decrementieren noch incrementieren wollen, da wir diesen, bevor wir nicht multipliziert haben, gerne unverändert hätten.
Neben den Indexregistern können auch Speicherstellen angesprochen werden. Es folgt eine Auflistung der Increment- und Decrement-Befehle.

```
INX
DEX
```

```
INY
DEY
INC
DEC
```

Wir laden also das Y-Register mit dem Wert #$08 und unser Algorithmus kann beginnen. Da hierbei auch Verzweigungen stattfinden, sei er in einzelnen Schritten verfaßt.

1. ASL
2. ASL $3000
3. Wenn Carry = 0, dann Sprung nach 6.
4. CLC
5. ADC $3001
6. DEY
7. Wenn Zero = 0, dann Sprung nach 1.
8. LDY $3001

Etwas mühsam ist es, solch einen Algorithmus mit der Hand durchzuprobieren. Zum Programmieren fehlt uns fast nichts mehr, denn unser Befehlssatz ist bereits nahezu vollständig. Uns fehlt nur noch der Befehl, um ein Unterprogramm anzuspringen, und der Befehl für den Rücksprung. Der Befehl für den Rücksprung heißt *RTS*. Wir können ihn in unserem CPU-Programm nicht aufrufen. Genauso wie die Branch-Befehle verwenden wir ihn erst in einem richtigen Programm.
Der *JSR*-Befehl, der ein Unterprogramm anspringt, ist auch schon in unserem CPU-Programm verfügbar. Wir dürfen ihn allerdings erst verwenden, wenn wir ein anzuspringendes Maschinenprogramm vor uns haben. Besonders beachtet werden muß bei diesen Befehlen, daß der JSR, Jump SubRoutine, die Rücksprungadresse auf dem Stapel ablegt. Nur so kann *RTS*, ReTurn from Subroutine, wieder an die richtige Stelle im Hauptprogramm zurückkehren. Dies bedeutet, daß ein auf den Stapel gepushter Wert nicht nach einem Sprung in eine Unterroutine so ohne weiteres abgehoben werden kann. Anstatt des geretteten Akkuinhalts würde ein Pull-Befehl ein Byte der Rücksprungadresse abheben.
Der RTS hätte als Rücksprungadresse eine Kombination von gerettetem Akkuinhalt als High-Byte und High-Byte-Rücksprungadresse als Low-Byte vor sich und würde ganz schön ins Nirwana stürzen. Außer den zwei soeben genannten Befehlen gibt es noch einen unbedingten Sprung namens *JMP*, der nicht nur für Sprünge in einem Bereich von $7F Adressen vorwärts und $80 Adressen rückwärts eingesetzt werden kann, sondern den Programmzähler auf jede beliebige Speicheradresse einstellen kann. Stapelbeeinflussungen finden hierbei nicht statt.
Um vollständig zu sein, wollen wir auch einen ganz unscheinbaren Befehl nicht unerwähnt lassen. Sie können ihn ruhig gleich ausprobieren. Er heißt *NOP*, was No OPeration bedeutet. Dieser Befehl bewirkt nichts. Er wird gerne als Platzhalter verwendet, so wie wir es auch im Maschinenteil unseres CPU-Programmes taten. An eine bestimmte Stelle des Programms setzen wir je nach auszuführendem Befehl entweder ein, zwei oder drei Byte lange Befehle ein. Damit die CPU bei ein oder zwei Byte langen Befehlen nicht über den zurückgebliebenen Rest der drei Byte langen Befehle stolpert, füllten wir die verbleibenden ein oder zwei Adressen mit NOP-Codes auf. Wir schreiben jetzt unser erstes Maschinenprogramm. Da wir den Algorithmus für die Multiplikation bereits notiert hatten, fällt es uns bestimmt nicht schwer, diesen mit Hilfe des Monitors einzugeben.

```
BRK                 BCC $3010
A3002 LDA #$00      CLC
STX $3000           ADC $3001
STY $3001           DEY
LDY #$08            BNE $300C
ASL                 LDY $3001
ASL $3000           RTS
```

Der Sprungbefehl bei Adresse $3010 stimmt noch nicht ganz. Wir haben, da wir die Sprungadresse beim Niederschreiben noch nicht gekannt haben, einen provisorischen Wert eingesetzt. Da jetzt die Adresse des DEY-Befehles bekannt ist, ersetzen wir BCC $3010 durch BCC $3016.
Den Monitorbefehl zum Betrachten eines Speicherbereichs im Hexformat kennen wir bereits. Zum Betrachten eines Maschinenprogramms wäre es sehr schön, wenn dieses sich auch in Assembler-Mnemonics anschauen ließe. Auch dafür gibt es einen Monitorbefehl. Er heißt *Disassemble* und wird mit dem Buchstaben **D**, gefolgt von einer Adresse eingegeben. Sie dürfen ruhig einmal den Bildschirm löschen. Danach sehen wir uns das Maschinenprogramm wieder an.

```
D3002
```

Ganz wird das Multiplikationsprogramm durch diesen Monitorbefehl nicht angezeigt. Um den Rest zu disassemblieren, ist nur noch ein D ohne nachfolgende Adresse einzugeben.

```
D
```

Disassembliert sollte das Programm sich nun darstellen, wie es nachfolgend zu sehen ist.

```
.03002 a9 00    lda #$00
.03004 8e 00 30 stx $3000
.03007 8c 01 30 sty $3001
.0300a a0 08    ldy #$08
.0300c 0a       asl
.0300d 0e 00 30 asl $3000
.03010 90 04    bcc $3016
.03012 18       clc
.03013 6d 01 30 adc $3001
.03016 88       dey
.03017 d0 f3    bne $300c
.03019 ac 01 30 ldy $3001
.0301c 60       rts
```

Rechts neben dem Punkt erscheint die jeweilige Speicheradresse. Die nachfolgenden ein, zwei oder drei Byte sind der Operationscode und ein oder zwei Parameter. Darauf folgen die disassemblierten Mnemonics. Interessant ist, daß bei der absoluten Adressierung dem Operationscode erst das Low-Byte der Speicheradresse und dann erst das High-Byte folgt. Den Operationscodes für die bedingten Sprünge folgt nicht eine absolute Sprungadresse, sondern nur ein einziges Byte, das angibt, um wieviel Adressen der Programmzähler verstellt werden soll. Im Verzweigungsfalle landet der Programmzähler beim BCC in Adresse $3010 nicht bei $3012 sondern, vier Adressen weiter, bei $3016. Zahlen ab $80 sind negativ, wobei $ff der Dezimalzahl −1 entspricht und $80 der Dezimalzahl −128.
Der Sprungbefehl in Adresse S3017 landet daher bei Adresse $300 c. Relative Sprünge bringen einen entscheidenden Vorteil. Die Programme sind verschiebbar. Wir kopieren unser Maschinenprogramm.

```
T3002,301C,3100
D3100
```

Die disassemblierten Sprungadressen sind andere, als wir ursprünglich erfaßt hatten. Der Disassembler zeigt jetzt automatisch die richtigen neuen Sprungadressen an, da ja nur die Sprungweite im Maschinenprogramm steht.

```
G
```

Jetzt haben wir die Gelegenheit, den JSR-Befehl und unser Maschinenprogramm auszuprobieren.

```
LDX #$05
LDY #$07
JSR $3002
```

Das Multiplizieren klappt ja direkt hervorragend. Mit dem Befehlssatz des Prozessors sind wir am Ende angelangt. Zu besprechen sind lediglich noch die verschiedenen Adressierungsarten.

VII. ADRESSIERUNGSARTEN DER CPU

Neben impliziter, unmittelbarer und absoluter Adressierung bietet die CPU unseres Rechners eine Vielzahl weiterer Adressierungen. Dreizehn sind es insgesamt. Mit indirekter und indizierter Adressierung können wir ganze Speicherbereiche durchwühlen. Eine Befehlstabelle gibt uns den Überblick.

In der Beschreibung der Adressierungsarten sind auch Befehlslänge und Befehlsdauer angegeben. Die Befehlsdauer mißt sich in Takzyklen. Die CPU 8502 kann sowohl mit einem, als auch mit zwei Megahertz getaktet werden. Ein Megahertz, das sind eine Million Takte in der Sekunde. Ein Takt dauert so eine Millionstel Sekunde, anders ausgedrückt, eine Mikrosekunde. Ein Befehl, der in zwei Taktzyklen abgearbeitet wird, braucht somit zwei Mikrosekunden. 500000 solcher Befehle könnten also in der Sekunde durchgeführt werden, eine ganz erkleckliche Anzahl. Kein Wunder also, daß in Maschinensprache geschriebene Programme als ungeheuer schnell gelten.
Manche Adressierungsarten arbeiten schneller als andere. Manche nehmen weniger Programmspeicher in Anspruch, weil die Befehlslänge eventuell kürzer ist. Andere Adressierungsarten fixieren uns nicht auf bestimmte Adressen, sondern erlauben uns, Routinen zu schreiben, die auf jeden beliebigen Speicherbereich zugreifen können. In der Befehlstabelle sehen wir, welche Adressierungen für welche Operationen zur Verfügung stehen, und können so das für unseren Zweck geeignetste auswählen.

A. Unmittelbare Adressierung

Beispiel: LDY #$05
Befehlslänge: Zwei Byte
Taktzyklen: Zwei

Dem Operationscode folgt unmittelbar das Daten-Byte, das mit dem Akkumulatorinhalt verknüpft wird.

B. Absolute Adressierung

Beispiel: STA $3000
Befehlslänge: Drei Byte
Taktzyklen: Drei bis sechs

Dem Operationscode folgt eine Zwei-Byte-Adresse. Hierbei wird erst das Low- und dann das High-Byte im Speicher abgelegt. Der JMP-Befehl benötigt drei Taktzyklen, der JSR sechs. Befehle, die eine Änderung einer Speicherzelle bewirken, benötigen sechs Taktzyklen. Die Ausnahme bildet der Store-Befehl mit vier Taktzyklen. Lade- und sonstige Befehle, die den Inhalt einer Speicherzelle mit einem CPU-Register verknüpfen, dauern ebenfalls vier Taktzyklen.

C. Zero-Page-Adressierung

Beispiel: LDA $01
Befehlslänge: Zwei Byte
Taktzyklen: Drei bis fünf

Um Adressen von $00 bis $FF anzusprechen, kann auf ein High-Adress-Byte verzichtet werden. Dem Operationscode folgt nur ein einziges Adress-Byte. Kürzer ist nicht nur die Länge des Befehls, sondern auch die Abarbeitungszeit. Ein Taktzyklus wird gespart.

D. Akku-Adressierung

Beispiel: ASL
Befehlslänge: Ein Byte
Taktzyklen: Zwei

Da keine Werte übergeben werden und keine Speicherzelle zu adressieren ist, wird nur der Operationscode benötigt.

E. Implizite Adressierung

Beispiel: SEC
Befehlslänge: Ein Byte
Taktzyklen: Zwei bis sechs

Die Adressierung ist bereits durch den Operationscode bestimmt. Daten oder Adreßparameter erübrigen sich daher. Befehle, die nicht auf den Speicher zugreifen, dauern zwei Taktzyklen. Push-Befehle vier, die Rücksprünge RTS und RTI benötigen gar sechs Zyklen. Mit RTI wird von einem Interrupt wieder zurückgesprungen. Da ein Interrupt nicht nur die Rücksprungadresse, sondern auch das Statusflag auf den Stapel rettet, wird dieses durch RTI wieder restauriert.

F. Indiziert-indirekte Adressierung

Beispiel: LDA ($FA,X)
Befehlslänge: Zwei Byte
Taktzyklen: Sechs

Dem Operationscode folgt ein Adreß-Byte. Indirekt adressiert heißt, daß nicht die durch den Adreßparameter bestimmte Zelle adressiert wird, sondern, daß in der spezifizierten und der nachfolgenden Speicherstelle sich erst die Adresse der Zelle findet, auf die die Operation zugreift. Indiziert heißt, daß zur Adreßbildung zusätzlich ein Index-Register einbezogen wird. Bei der indiziert-indirekten Adressierung der CPU 6502 ist dieses das X-Register.
Der Inhalt des X-Registers wird in unserem Beispiel zur Adresse $FA addiert. Wir erhalten so die Adresse der Speicherstelle. In dieser und der nachfolgenden finden wir endlich die Adresse unserer Speicherstelle, die es anzusprechen gilt. Bei der Summierung der X-Register und der Zero-Page-Adresse bleibt ein Übertrag unberücksichtigt, so daß für indirekte Adressierungen auch wirklich nur Speicherstellen von $00 bis $FF in Frage kommen.
Die meisten Zero-Page-Adressen sind bereits für Operationen des Betriebssystems vergeben. Zur freien Verfügung steht dem C64-C128-User der Bereich von $FA bis $FE, in dem er nach Belieben schalten und walten kann. Um die Sache noch etwas zu verdeutlichen, nehmen wir einmal an, das X-Register hätte den Wert #$02, in der Speicherstelle $FC stünde der Wert #$00 und in $FD der Wert #$30. Die adressierte Speicherstelle wäre in diesem Falle $3000.

G. Indirekt-indizierte Adressierung

Beispiel: LDA $(FA),Y
Befehlslänge: Zwei Byte
Taktzyklen: Fünf bis sechs

Der *Store*-Befehl benötigt sechs Zyklen, die anderen Befehle nur fünf. Diese Adressierungsart wird öfter benötigt als die indiziert-indirekte. Zur Adresse in den zwei Speicherstellen in der Zero-Page wird der Inhalt des Y-Registers addiert. Die Summe davon ist die Adresse unserer anzusprechenden Speicherzelle. Hätte das Y-Register den Wert #$81, die Speicherstelle $FA den Wert #$80 und $FB den Wert #$30, so würde durch die Operation die Speicherstelle $3101 angesprochen werden. Überträge bei der Summierung der Adress-Low-Byte werden berücksichtigt. Bei allen indizierten Adressierungsarten ist, wenn solche Pageüberschreitungen auftreten, ein Taktzyklus hinzuzuaddieren.

H. Zero-Page-indizierte Adressierung mit Indexregister X

Beispiel: LDA $FA,X
Befehlslänge: Zwei Byte
Taktzyklen: Vier bis sechs

Die Abarbeitungszeit der Befehle entspricht derjenigen der absoluten Adressierung. Zur Adresse wird der Inhalt des X-Registers addiert. Es können nur Speicherstellen von $00 bis $FF angesprochen werden. Ein Übertrag bei der Addition der Adressen wird unterschlagen.

I. Direkt indizierte Adressierung mit Indexregister X

Beispiel: LDA $3000,X
Befehlslänge: Drei Byte
Taktzyklen: Vier bis sieben

Der *STA*-Befehl benötigt fünf Taktzyklen, sonstige Befehle, die den Inhalt einer Speicherstelle verändern, gar sieben. Alle übrigen Operationen geben sich mit vier Zyklen zufrieden. Wie bei der direkten Adressierung besteht die Basisadresse aus zwei Byte.
Anders als bei der Zero-Page-indizierten Adressierung wird bei der Bildung der Operandenadresse ein aus der Summierung der Low-Byte resultierender Übertrag berücksichtigt. Hätte X den Wert #$81, so würde mit LDA $3080,X die Speicherstelle $3101 angesprochen.

J. Direkt indizierte Adressierung mit Indexregister Y

Beispiel: LDA $3000,Y
Befehlslänge: Drei Byte
Taktzyklen: Vier bis fünf

Zur Basisadresse wird das Y-Register addiert. Der STA-Befehl braucht fünf Taktzyklen, die anderen Befehle nur vier.

K. Relative Adressierung

Beispiel: BCC $3016
Befehlslänge: Zwei Byte
Taktzyklen: Zwei bis drei

Dem Operanden folgt ein Byte, das die Sprungweite signalisiert. Zahlen von #$80 bis #$FF gelten als negativ. Sprünge können so nur in einem Bereich von Null bis 127 Adressen nach vorne und −1 bis 128 Adressen nach rückwärts erfolgen. Ist die Verzweigungsbedingung nicht erfüllt, so benötigt der Befehl zwei Taktzaklen, im Verzweigungsfalle drei. Bei Page-Überschreitung ist ein weiterer Taktzyklus hinzuzuaddieren.

L. Indirekte Adressierung

Beispiel: JMP ($0326)
Befehlslänge: Drei Byte
Taktzyklen: Fünf

Eine indirekte Adressierung, die nicht auf Zero-Page-Adressen beschränkt ist, kennt nur der *JMP*-Befehl. In unserem Falle erfolgt der Sprung nach der Adresse, die in den Speicherstellen $0326 und $0327 vermerkt ist. Einige wichtige Betriebssystemroutinen unserer Commodore-Rechner werden indirekt über sogenannte Sprungvektoren aufgerufen, die im RAM lokalisiert sind. Durch das Ersetzen der Adresse in den Speicherstellen $0326 und $0327, ist es ein Leichtes, eine neue Ausgaberoutine zu schreiben, die zum Beispiel für einen Epson-Drucker den Commodore-ASCII in Standard-ASCII umwandelt. Die Adresse eines direkten Sprunges im ROM hätten wir nicht manipulieren können, da aus dem ROM ja bekanntlich nur gelesen werden kann.

M. Zero-Page-indizierte Adressierung mit Indexregister Y

Beispiel: LDX $FA,Y
Befehlslänge: Zwei Byte
Taktzyklen: Vier

Indexregister ist Y. Ansonsten gilt das bereits über diese Adressierungsart mit Indexregister X Gesagte.
Die *Übersichtstabelle* gibt Auskunft, welche Adressierungsarten für welche Operationen bereitstehen. Sollte der Assembler einmal einen Befehl nicht annehmen wollen, wie LDA $FA,Y, so belehrt uns die Tabelle sogleich, daß es eine Zero-Page-indizierte Adressierung mit Y-Register nur für LDX und STX gibt. Ein paar Spalten weiter links sagt sie uns, daß die direkt indizierte Adressierung

	A	B	C	D	E	F	G	H	I	J	K	L	M		
ADC	*	*	*			*	*	*	*	*				NV	ZC
AND	*	*	*			*	*	*	*	*				N	Z
ASL		*	*	*				*	*					N	ZC
BCC											*				
BCS											*				
BEQ											*				
BIT		*	*											NV	Z
BMI											*				
BNE											*				
BPL											*				
BRK					*										
BVC											*				
BVS											*				
CLC					*										C
CLD					*									D	
CLI					*									I	
CLV					*									V	
CMP	*	*	*			*	*	*	*	*				N	ZC
CPX	*	*	*											N	ZC
CPY	*	*	*											N	ZC
DEC		*	*					*	*					N	Z
DEX					*									N	Z
DEY					*									N	Z
EOR	*	*	*			*	*	*	*	*				N	Z
INC		*	*					*	*					N	Z
INX					*									N	Z
INY					*									N	Z
JMP		*										*			
JSR		*													
LDA	*	*	*			*	*	*	*	*				N	Z
LDX	*	*	*							*			*	N	Z
LDY	*	*	*					*	*					N	Z
LSR		*	*	*				*	*					N	ZC
NOP					*										
ORA	*	*	*			*	*	*	*	*				N	Z
PHA					*										
PHP					*										
PLA					*									N	Z
PLP					*									NV	DIZC
ROL		*	*	*				*	*					N	ZC
ROR		*	*	*				*	*					N	ZC
RTI					*									NV	DIZC
RTS					*										
SBC	*	*	*			*	*	*	*	*				NV	ZC
SEC					*										C
SED					*									D	
SEI					*									I	
STA		*	*			*	*	*	*	*					
STX		*	*										*		
STY		*	*						*						
TAX					*									N	Z
TAY					*									N	Z
TSX					*									N	Z
TXA					*									N	Z
TXS					*										
TYA					*									N	Z

jedoch möglich ist, und wir mit LDA $00FA,Y zum Ziel kommen.

Beeinflussung der Statusflags

Zur Erstellung der Programmlogik ist es notwendig, zu wissen, welcher Befehl welches Statusflag beeinflußt. In der rechten Spalte unserer Übersichtstabelle findet sich diese Information. Auf die Darstellung von Bit vier und fünf des Statusregisters haben wir verzichtet, da Bit fünf keine Funktion hat und Bit vier, das Break-Bit, nur bei einem einzigen Befehl gesetzt wird, dem BRK-Befehl.
Dieses Flag wird im Betriebssystem unseres Rechners nur ein einziges Mal beim Interrupt abgefragt. Ist dieses Bit dort gesetzt, wird über den Sprungvektor $0316 der Maschinenmonitor aufgerufen. Wollen wir den Sprung in den Monitor durch eine andere Routine ersetzen, so genügt eine Umstellung des Break-Vektors. Um das Break-Bit brauchen wir uns also nicht zu kümmern, wenn wir nicht selbst ein vollkommen neues Betriebssystem schreiben wollen.
Wir ersehen aus der Tabelle, daß die Statusflags bei Sprüngen, ob bedingten, absoluten oder Unterroutinen-Aufrufen, nicht beeinflußt werden. Ebenfalls keine Änderungen treten beim Schreiben von Registerinhalten in Speicherstellen auf, ob dieses nun mit Store-Befehlen, wie STA, STX, STY oder mit den Stapelbefehlen PHA und PHP geschieht.
Für den Registertransfer TSX, BRK, dem Sprung in den Break-Interrupt, und NOP, das gar nichts macht, gilt dasselbe.
Die spezifischen Set- und Clear-Befehle für die einzelnen Statusflags ändern nur das jeweilige Bit, auf das sie sich beziehen. PLP und RTI stellen das ehemals gerettete Statusflag wieder her. Die sonstigen Befehle beeinflussen das Vorzeichen-Bit N und das Zero-Flag Z. Bei Shift-, Rotate-Befehlen, Addition, Subtraktion und Vergleichen wird zusätzlich das Carry-Flag beeinflußt. Addition, Subtraktion und der Bit-Befehl wirken sich auch auf das Überlauf-Flag V aus.

VIII. EIN- UND AUSGABEROUTINEN

Über eine Standard-Sprungtabelle, das Kernel, kann jedes Ein- und Ausgabegerät angesprochen werden. Das Kernel sorgt dafür, daß Maschinenprogramme, die sonst keine systemspezifischen Adressen verwenden, auf jedem Acht-Bit-Commodore-Rechner lauffähig sind.

Was nutzt ein Rechner, der eine CPU besitzt, über ROM und RAM verfügt, jedoch keine Möglichkeiten bietet, Daten ein- oder auszugeben? Ein Rechner, der nicht die Möglichkeit bietet, mit seiner Umgebung zu kommunizieren, ist zu überhaupt nichts nütze. Wir benötigen daher noch diverse Ein- und Ausgabeoperationen. Für die Kommunikation mit der Außenwelt besitzt der Rechner verschiedene I/O-Bausteine, die über bestimmte Adressen angesprochen werden können.
Da für das Ansprechen solcher Bausteine ziemliche Kenntnisse über dieselben und über den Aufbau des bestimmten Rechners erforderlich wären, haben die Systemprogrammierer bereits die erforderlichen Routinen programmiert.
Ein solches Betriebssystem muß vorhanden sein, sonst könnten wir gar kein Programm in den Rechner bringen. Nicht einmal ein einziger Tastendruck würde vom Rechner verarbeitet werden können.
Der Teil des Betriebssystems, der für Ein- und Ausgaben verantwortlich ist, heißt *BIOS*, was **B**asic **I**nput **O**utput **S**ystem bedeutet. Das Wort Basic hat nichts mit der Programmiersprache BASIC zu tun, sondern sagt aus, daß es hier um einfache Ein- und Ausgaberoutinen geht, die die Grundlage für die Kommunikation mit der Peripherie bilden, und auf die alle komplexeren Ein- und Ausgaberoutinen wie beispielsweise der PRINT-Befehl oder der INPUT-Befehl zurückgreifen.
Um auch bei unterschiedlich organisierten Rechnern dafür zu sorgen, daß diese in gleicher Weise zu handhaben sind, wird der Aufruf dieser Funktionen standardisiert. Beim VC20, C64, C16/116/Plus4 und dem C128 gibt es einen Speicherbereich im ROM, der an genau denselben Stellen eine Sprungliste, das Kernel, für diese wichtigen Systemaufrufe enthält. Um ein Zeichen aus dem Tastaturpuffer zu lesen, brauchen wir deshalb nicht zu wissen, wo dieser in unserem Rechner ist, und wie er verwaltet wird. Es existiert schließlich schon die Routine, die dies für uns erledigt. Wo diese Routine in unserem Rechner genau liegt, braucht uns ebenfalls nicht zu interessieren. Alles, was wir wissen müssen, ist, daß bei Ansprung der Adresse $FFE4 der Rechner in die gewünschte Routine verzweigt.

Eingaberoutinen

Zwei fundamentale Routinen besitzt unser Rechner, mit denen wir Daten von einem externen Gerät lesen können:

GETIN ($FFE4)

Mit dieser Routine können wir Daten von einem Eingabegerät einlesen. Die BASIC-Befehle GET und GET# bauen darauf auf. Wurde zuvor kein anderes Gerät als Eingabegerät definiert, so wird ein Zeichen aus dem tastenpuffer an den Akku übergeben. Wurde keine Taste gedrückt, so erhält der Akku den Wert #$00 zugewiesen.

JSR $FFE4

Im Akku sollte sich nun der Wert #$00 befinden. Wir geben den Befehl jetzt erneut ein. Da eine Wiederholung vorliegt, brauchen wir hierzu nur die Return-Taste zu drücken. Gleich nach dem Druck dieser Taste, sollten wir blitzschnell noch eine andere Taste drücken. Bis das in BASIC verfaßte CPU-Programm in das Maschinenprogramm verzweigt, sollte der ASCII-Wert der gedrückten Taste bereits im Tastenpuffer abgelegt sein. Der Akkumulator zeigt im Anschluß diesen Wert. GETIN wartet nicht auf einen Tastendruck, wie dieses der BASIC-Befehl GETKEY tut. Da je nach erhaltenem Wert auch das Statusflag beeinflußt wird, können wir das Warten auf einen Tastendruck leicht selbst programmieren.

```
BRK
A301D JSR $FFE4
BEQ $301D
RTS
G
JSR $301D
```

BASIN ($FFCF)

Auch BASIN liest sowohl von der Tastatur, wie auch von sonstigen Eingabegeräten. Ist nicht die Tastatur das Eingabegerät, so besteht zwischen GETIN und BASIN kein Unterschied. Beim Lesen von der Tastatur ist der Unterschied nicht zu übersehen.

JSR $FFCF

Deutlich wird hier, daß BASIN die Grundlage des INPUT-Befehls bildet. Es erscheint ein blinkender Cursor und wir können soviel Zeichen eingeben, wie wir gerne möchten. Erst beim Druck der Return-Taste wird die BASIN-Routine beendet und wir bekommen das an erster Stelle eingegebene Zeichen in den Akku. Wie bekommen wir aber nun die restlichen Zeichen? Da das CPU-Programm hierauf keine Antwort zu

geben vermag, so sei Ihnen dieses jetzt verraten. Bei jedem folgenden Aufruf von BASIN wird das nächste Zeichen in den Akku geladen, bis schließlich das Ende der Eingabe erreicht ist, was der Akku uns mit #$0D für das Returnzeichen anzeigt. Eingegebene Leerstellen am Ende werden ignoriert. Das CPU-Programm stört diese Eigenheit von BASIN. Deshalb verfassen wir ein kleines Maschinenprogramm, mit dem wir Eingaben nach Wahl im Hauptspeicher unseres Rechners positionieren können. Wie dieses geht, sollten Sie nun schon wissen. Hier ist das disassemblierte Monitorlisting.

```
.03023 84 fa       sty  $fa
.03025 86 fb       stx  $fb
.03027 a0 00       ldy  #$00
.03029 20 cf ff    jsr  $ffcf
.0302c 91 fa       sta  ($fa),y
.0302e e6 fa       inc  $fa
.03030 d0 02       bne  $3034
.03032 e6 fb       inc  $fb
.03034 c9 0d       cmp  #$0d
.03036 d0 f1       bne  $3029
.03038 a4 fa       ldy  $fa
.0303a a6 fb       ldx  $fb
.0303c 60          rts
```

Um das Programm verschiebbar zu gestalten, benutzten wir die indirekte Adressierung. Um einen größeren Bereich als 256 Byte ansprechen zu können, und nicht am Ende extra den Wert des Y-Registers zum Inhalt der Speicherstelle $FA addieren zu müssen, verzichteten wir auf eine Indizierung, indem wir das Y-Register kurzerhand auf Null setzten. Schnell genug ist das Programm wohl schon auch so.
Vor dem ersten Aufruf dieses Programmes ist dem X-Register das High-Byte der Speicheradresse und dem Y-Register das Low-Byte mitzuteilen. Die Eingabe wird beim Aufruf ab dieser Adresse gespeichert. Sowohl die Speicherzellen $FA und $FB, als auch die Indexregister weisen hinterher auf die den abgespeicherten Daten folgende Speicherstelle. Durch mehrmaligen Aufruf dieses Maschinenprogrammes können so kontinuierlich weitere Abspeicherungen vorgenommen werden. Sollten bei zwischendurch erfolgenden Operationen die Indexregisterinhalte verlorengehen, so braucht die Eingabe das nächste Mal anstatt mit JSR $3023 nur mit JSR $3027 angesprungen zu werden. Wir probieren in unserem CPU-Programm ein paar Eingaben.

```
LDX #$31
LDY #$00
JSR $3023
```

Durch Druck der Returntaste können wir weitere Eingaben vornehmen.

```
BRK
M3100
G
```

Im Monitor können wir uns vom Funktionieren der Routine überzeugen.

Die Ausgaberoutine BSOUT ($FFD2)

BSOUT bildet die Grundlage für alle Ausgaben. Der im Akku befindliche Wert wird an das Ausgabegerät gesandt.
Sofern nichts anderes definiert wurde, ist dieses der Bildschirm.

```
LDA #$41
JSR $FFD2
```

Auf dem Bildschirm sollte der Buchstabe A erscheinen. Bei einem längeren auszugebenden Text ist jedesmal ein Zeichen in den Akku zu laden, und es dann mit BSOUT auszugeben, ist die schnellste, aber natürlich nicht die richtige Methode. Genauso, wie bei der Eingabe, verfassen wir daher eine Routine, die Daten ausgibt, welche in irgendeinem Speicherbereich liegen.

```
.0303d 84 fc       sty  $fc
.0303f 86 fd       stx  $fd
.03041 a0 00       ldy  #$00
.03043 b1 fc       lda  ($fc),y
.03045 20 d2 ff    jsr  $ffd2
.03048 e6 fc       inc  $fc
.0304a d0 02       bne  $304e
.0304c e6 fd       inc  $fd
.0304e c9 0d       cmp  #$0d
.03050 d0 f1       bne  $3043
.03052 a6 fd       ldx  $fd
.03054 a4 fc       ldy  $fc
.03056 60          rts
```

Oft erfolgen Ein- und Ausgaben abwechselnd. Damit keine gegenseitige Verstellung der Zeiger auftritt, haben wir die Adressen $FC und $FD als Ausgabezeiger verwendet. Nun wollen wir die vorher eingegebenen Daten auf den Bildschirm ausgeben.

```
LDX #$31
LDY #$00
JSR $303C
JSR $303C
JSR $303C
```

Ansteuerung externer Geräte

OPEN, CLOSE, und CMD gibt es auch für Maschinensprache. Zur Parameterübergabe an den OPEN-Befehl bedarf es aber erst einiger vorbereitetender Routinen.

SETLFS ($FFBA)

Mit SETLFS werden die logische Dateinummer, die Geräte- und die Sekundäradresse erfaßt. Wir laden die Dateinummer in den Akku, die Geräteadresse in das X-Register und die Sekundäradresse in das Y-Register.

Beispiel: OPEN1,4,7

```
LDA #$01
LDX #$04
LDY #$07
JSR $FFBA
```

Soll keine Sekundäradresse angegeben werden, so ist Y mit #$FF zu laden.

SETNAM ($FFBD)

Um bei Kassetten- oder Diskettenoperation auch einen Dateinamen angeben zu können, existiert die SETNAM-Routine. Hierbei wird dem Akku die Länge des Dateinamens, dem X-Register das Low-Byte der Adresse, wo der Dateiname sich befindet, und dem Y-Register das Adreß-High-Byte mitgeteilt. Beim Drucker ist kein Dateiname erforderlich. In diesem Falle geben wir die Länge des Dateinamens mit Null an. Um die Adresse brauchen wir uns dann nicht mehr zu kümmern.

```
LDA #$00
JSR $FFBD
```

OPEN ($FFC0)

Nachdem die Parameter durch SETLFS und SETNAM festgelegt sind, können diese durch OPEN in der File-Tabelle des Rechners vermerkt werden.

```
JSR $FFC0
```

Bei Floppy oder Datasette werden diese Geräte zusätzlich angesteuert, um entweder den entsprechenden File zu suchen oder anzulegen.

CHKOUT ($FFC9)

Dieser Befehl entspricht dem CMD-Befehl, der uns bereits vom BASIC her bekannt ist. Nachdem ein entsprechender File mit OPEN eröffnet wurde, können wir das Ausgabegerät umdefinieren. Hierzu ist das X-Register mit der Kanalnummer zu laden und CHKOUT aufzurufen.

```
LDX #$01
JSR $FFC9
```

Daten, die jetzt mit BSOUT ausgegeben werden, erscheinen nun nicht mehr auf dem Bildschirm. Spätestens beim ersten Zeichen nach einem Return mit Code #$0D sollte der Drucker mit dem Druck beginnen.

```
LDA #$41        LDA #$42
JSR $FFD2       JSR $FFD2
LDA #$0D        LDA #$0D
JSR $FFD2       JSR $FFD2
```

Der Buchstabe A wird jetzt in jedem Fall auf dem Drucker ausgegeben. Wenn der Buchstabe B noch nicht erscheint, so tut er dies bestimmt, wenn der Druckerkanal wieder geschlossen wird.

CLRCHN ($FFCC)

Um wieder die Tastatur als Eingabegerät und den Bildschirm als Ausgabegerät zu definieren, ist keine Parameterübergabe, sondern nur der Aufruf von CLRCHN nötig.

```
JSR $FFCC
LDA #$41
JSR $FFD2
```

Das Zeichen A erscheint wieder auf dem Bildschirm. Mit CHKOUT kann jederzeit wieder auf den Drucker umgeschaltet werden.

CLOSE ($FFC3)

Wurde ein Kanal mit CLRCHN geschlossen, so können wir, sofern wir ihn nicht mehr benötigen sollten, ganz aus unserer File-Tabelle beseitigen. Hierzu ist der Akkumulator mit der Kanalnummer zu laden.

```
LDA #$01
JSR $FFC3
```

Mit CHKOUT auf den Drucker umschalten zu wollen, ist jetzt zwecklos. Denn der Rechner kennt die Geräteparameter nicht mehr. Erst nach erneutem Anlegen mit SETLFS, SETNAM und OPEN bekommen wir unseren Drucker wieder in den Griff. Der CLOSE-Befehl ist besonders wichtig bei Kassetten- und Diskoperationen, damit Floppy- und Datasettendatei ordentlich geschlossen werden.

CLALL ($FFE7)

Ziemlich radikal ist diese Routine. Wie CLRCHN schließt sie den jeweiligen Gerätekanal. Zusätzlich beseitigt sie aber alle Einträge aus der Filetabelle. Wenn ein Gerät noch ordentlich mit CLOSE abgeschlossen werden sollte, gibt es Probleme, da die Geräteparameter und der Dateiname nicht mehr greifbar sind.

CHKIN ($FFC6)

Wie CHKOUT einen Ausgabekanal öffnet, so tut dieses CHKIN mit einem Eingabekanal. Mit GETIN oder BASIN können die Daten dann vom entsprechenden Gerät eingelesen werden. Das Öffnen eines Eingabekanals hat keinen Einfluß auf den Ausgabekanal und umgekehrt. Daten können so von einem externen Gerät eingelesen und mit BSOUT gleich wieder auf das Ausgabegerät ausgegeben werden. Die Parameterübergabe erfolgt bei CHKIN ebenfalls über das X-Register.

READST ($FFB7)

Von BASIC her ist uns die Statusvariable bereits bekannt. Wenn READST aufgerufen wird, bekommen wir die Statusvariable in den Akku. Daraus können wir ersehen, ob das Ende einer Datei erreicht ist, oder sonstige Fehler vorliegen.

Ein- und Ausgabe-Experimente

Wenn unsere beiden Maschinenprogramme für Ein- und Ausgabe noch im Speicher vorliegen,so können wir ein wenig experimentieren.

Datei für Drucker anlegen

```
LDA #$ 04
TAX
LDY #$07
JSR $FFBA
LDA #$00
JSR $FFBD
JSR $FFC0
```

Daten erfassen

```
LDA #$31
LDY #$00
JSR $3023
Eingabe nach Wahl
JSR $3023
Eingabe nach Wahl
JSR $3023
Eingabe nach Wahl
```

Daten auf Drucker ausgeben

```
LDX #$04
JSR $FFC9
LDX #$31
LDY #$00
JSR $303D
JSR $303D
JSR $303D
JSR $FFCC
```

Dateiname erfassen

```
LDX #$38
LDY #$00
JSR $3023
TESTDATEI,S,W
```

Diskettendatei eröffnen

```
LDA #$08
TAX
TAY
JSR $FFBA
LDA #$0D
LDX #$00
LDY #$38
JSR $FFBD
JSR $FFC0
```

Auf Diskette speichern

```
LDX #$ 08
JSR FFC9
LDX #$31
LDY #$00
JSR $303D
JSR $303D
JSR $303D
JSR $FFCC
LDA #$08
JSR $FFC3
```

Da die Daten nun gespeichert sind, können wir in den Monitor gehen und sie löschen.

```
BRK
F3100,3FFF,0
M3100
G
```

Wir führen diese Operation durch, damit niemand sagen kann, wir würden jetzt die Daten aus dem Hauptspeicher lesen.

Dateiname eingeben

```
LDX #$38
LDY #$00
JSR $3023
TESTDATE,S,R
```

Diskettendatei eröffnen

```
LDA #$08
TAX
TAY
JSR $FFBA
LDA #$0D
LDX #$00
LDY #$38
JSR $FFBD
JSR $FFC0
```

Daten einlesen und ausgeben

```
LDX #$08
JSR $FFC6
LDX #$31
LDY #$00
STX $00FD
STY $00FC
JSR $3023
JSR $3041
```

```
JSR  $3023
JSR  $3041
JSR  $3023
JSR  $3041
JSR  $FFCC
LDA  #$08
JSR  $FFC3
JSR  $FFE7
```

Ein Blick in den Monitor belehrt uns zusätzlich, daß unsere Daten nun auch wieder im Hauptspeicher vorhanden sind.

```
BRK
M3100
G
```

Damit sind wir schon fast am Ende unserer Einführung angelangt. Das Speichern und Laden von Maschinenprogrammen und das Wandeln in DATA-Zeilen sollen noch kurz abgehandelt werden.

Laden und Speichern von Maschinenprogrammen

Das Speichern geschieht am besten vom Monitor aus. Wir speichern den Bereich ab, in dem das Maschinenprogramm steht. Zuerst ist der Buchstabe S für Save einzugeben. Diesem folgt zwischen Anführungszeichen der Filename. Durch Komma getrennt geben wir die Gerätenummer ein. Nach einem weiteren Komma folgt die Anfangsadresse und zuallerletzt, ebenfalls durch Komma getrennt, die Endadresse. Die Endadresse muß um eine Speicherstelle höher angegeben werden. Der folgende Befehl würde so zum Beispiel den Bereich von $3000 bis $30FF auf Diskette sichern.

```
S"TESTFILE",8,3000,3100
```

Beim Laden erübrigen sich die Adreßangaben.

```
L"TESTFILE",8
```

Bei der Kassette ist anstelle einer Acht die Eins als Gerätenummer zu verwenden.

Wandeln in DATA-Zeilen

Für kurze Maschinenroutinen, die in ein BASIC-Programm eingebunden werden sollen, empfiehlt sich die Umwandlung in DATA-Zeilen. Die DATA werden vom BASIC-Programm wieder in die ursprünglichen Speicherstellen gePOKEd. Ein kurzes Programm *Datawandler* besorgt die Umwandlung. Nach dem Start ist die Zeilennummer einzugeben, ab der die DATA abgelegt werden sollen. Danach will das Programm Anfang und Ende des umzuwandelnden Speicherbereiches in hexadezimaler Form wissen. Die erzeugten Zeilen werden nur auf dem Bildschirm ausgegeben. Erst wenn Sie mit dem Cursor auf die zu sehenden Zeilen fahren und sie mit Return aufnehmen, stehen diese im Programm zur Verfügung. Mit dem DELETE-Befehl brauchen Sie nur noch die Programmzeilen des Datawandlers zu löschen. *AM* □

DATAWANDLER — 1 —

```
1 rem datawandler============c128 <nm>
2 rem (p) commodore welt team     <lm>
3 rem ========================== <cd>
4 rem (c) by alfons mittelmeyer   <ce>
5 rem                             <jj>
6 rem                             <jl>
7 rem basic v7.0                  <od>
8 rem c128                        <hf>
9 rem ========================== <nd>
10 input"zeilennummer";zn         <cg>
11 input "anfang,ende";x$,y$      <pf>
12 x=dec(x$):y=dec(y$)            <mf>
13 x$=str$(x):y$=str$(y)          <hc>
14 x$=right$(x$,len(x$)-1)        <ca>
15 y$=right$(y$,len(y$)-1)        <ng>
16 gosub29:print" fori="x$"to"y$:z
n=zn+10                           <jc>
17 gosub29:print" reada:pokei,a:ne
xt"                               <ac>
18 zn=zn+10                       <bf>
19 l=int((y-x+1)/6)+1:k=(y-x+1)-(l
-1)*6                             <dm>
20 ifk=0thenl=l-1:k=6             <hh>
21 forj=1tol:n=6:ifj=lthenn=k     <ip>
22 gosub29:print" data ";         <lh>
23 for i=0 to n-1:a=peek(x+i)     <kj>
24 a$=str$(a):a$=right$(a$,len(a$)
-1)                               <ef>
25 a$=right$("00"+a$,3)           <cc>
26 print a$",";:next:printchr$(20) <li>
27 x=x+6:zn=zn+10:next            <ec>
28 end                            <li>
29 printright$(str$(zn),len(str$(z
n))-1);                           <mi>
30 return                         <mk>
31 rem ========================== <kh>
32 rem  p r o g r a m m e n d e   <mb>
33 rem ========================== <md>
```

```
10 rem cpu-trainer==========c128 <ca>
20 rem (p) commodore welt team   <ho>
30 rem ========================= <ng>
40 rem (c) by alfons mittelmeyer <cg>
50 rem                           <pd>
60 rem                           <ah>
70 rem basic v7.0                <og>
80 rem c128                      <gm>
90 rem ========================= <jg>
100 rem maschinencode            <dh>
110 fori=12032to12148            <kc>
120 reada:pokei,a:next           <ag>
130 data 173,120,047,072,173,117 <kh>
140 data 047,174,118,047,172,119 <bg>
150 data 047,040,076,050,003,141 <dn>
160 data 117,047,142,118,047,140 <jm>
170 data 119,047,008,104,141,120 <ph>
180 data 047,088,216,096,173,117 <oe>
190 data 047,174,121,047,157,000 <gd>
200 data 001,206,121,047,096,238 <ej>
210 data 121,047,174,121,047,173 <md>
220 data 120,047,072,040,189,000 <hj>
230 data 001,141,117,047,008,104 <cp>
240 data 141,120,047,088,216,096 <ho>
250 data 173,120,047,024,144,215 <im>
260 data 238,121,047,174,121,047 <jk>
270 data 189,000,001,141,120,047 <ig>
280 data 096,173,120,047,072,040 <da>
290 data 173,121,047,141,118,047 <jj>
300 data 008,104,141,120,047,088 <hb>
310 data 024,096,173,118,047,141 <bm>
320 data 121,047,096             <gf>
330 sys12049:poke12153,127       <fa>
340 rem ------------------------ <ip>
350 rem  variablen               <oo>
360 rem ------------------------ <gg>
370 b$=chr$(32):b4$=b$+b$+b$+b$  <cg>
380 a$(0)="aiiiiiiiib"           <cd>
390 a$(1)="j00000000j"           <ff>
400 a$(2)="diiiiieiif"           <kh>
410 a$(3)="000000j00j"           <dl>
420 a$(4)="00000agiif"           <eh>
430 a$(5)="00000j000j"           <en>
440 a$(6)="00000diiic"           <bc>
450 data 032,176,174,189,173     <hf>
460 data 178,179,177,171,192,221 <fl>
470 g$="":fori=0to10:reada:g$=g$+c
hr$(a):next                      <il>
480 forj=0to6:fori=1to10         <oe>
490 mid$(a$(j),i,1)=mid$(g$,((asc(
mid$(a$(j),i,1))and15))+1,1):next:
nextj                            <pp>
500 b$(0)="binaer":b$(1)="hex"   <cp>
510 b$(2)="dezimal"              <gd>
520 c$="akku"+b4$+"x-register y-re
gister"                          <le>
530 d$="nv--dizc"+b4$+"stapel"   <fh>
540 rem ------------------------ <eg>
550 rem  bildschirmaufbau        <mp>
560 rem ------------------------ <eh>
570 char,0,0,"":scnclr           <jo>
580 gosub910:y=3:gosub900        <ll>
590 char,10,1,c$:x=7:y=2:gosub890 <kk>
600 x=18:gosub890:x=29:gosub890  <mn>
610 y=12:gosub900                <ea>
620 char,8,10,d$:x=7:y=11:gosub890 <of>
630 x=18:gosub890:char,0,18,"":gos
ub910                            <op>
640 x$="":gosub950               <ml>
650 rem ------------------------ <fi>
660 rem  eingabe                 <mc>
670 rem ------------------------ <mk>
680 poke6,19:sys61313:poke6,19:sys
61313:ifx$="brk"then570          <ae>
690 char,0,19,chr$(27)+"q? "+chr$(
164)                             <ji>
700 y$=" ":x=1                   <id>
710 sys61167:a=peek(6):if(a<32anda
<>13anda<>20)ora>95then710       <oe>
720 ifa=13then800                <fj>
730 ifa<>20theny$=y$+chr$(a):x=x+1
:goto760                         <hk>
740 ifx=1then700                 <an>
750 x=x-1:y$=left$(y$,x)         <ja>
760 poke6,157:sys61313:poke6,a:sys
61313:poke244,0:poke6,164:sys61313
:goto710                         <gg>
770 rem ------------------------ <ob>
780 rem  eingabe bearbeiten      <mg>
790 rem ------------------------ <io>
800 ify$<>" "thenx$=right$(y$,x-1) <kh>
810 poke6,13:sys61313            <ji>
820 ifx$="brk"thenscnclr         <hp>
830 poke6,27:sys61313:poke6,84:sys
61313                            <he>
840 poke6,147:sys61313           <pi>
850 gosub1180:goto680            <gj>
860 rem ------------------------ <ie>
870 rem  routinensammlung        <pe>
880 rem ------------------------ <ad>
890 fori=0to6:char,x,y+i,a$(i):nex
t:return                         <pc>
900 fori=0to2:char,0,y+i+i,b$(i):n
ext:return                       <lb>
910 poke6,61:fori=1to40:sys61313:n
ext:return                       <fo>
920 data 12149,7,2,12150,18,2    <pn>
930 data 12151,29,2,12152,7,11   <of>
940 data 12153,18,11             <om>
950 ifx$="brk"thenreturn         <fj>
960 restore920:forz=1to5:reada   <bd>
970 readx:ready:a=peek(a)        <nf>
980 poke6,19:sys61313:sys61313:cha
r,x+1,y+1,"":s=a:n=128           <pd>
990 fori=1to8:a$="0"             <ma>
1000 ifs>=nthens=s-n:a$="1"      <nd>
```

```
1010 char,x+i,y+1,a$:n=n/2:next     <bl>
1020 char,x+7,y+3,right$(hex$(a),2
)                                   <ep>
1030 char,x+6,y+5,right$(" "+str$(
a),3)                               <dl>
1040 nextz:return                   <mb>
1050 x=-1:fori=1tom:reada$:reada:i
fa$=y$theni=50                      <em>
1060 next:return                    <en>
1070 char,0,19,"fehler"+chr$(27)"q
":poke208,0:wait208,1:poke208,0:re
turn                                <bk>
1080 rem ------------------------   <fh>
1090 rem  ein-byte-befehle          <bm>
1100 rem ------------------------   <om>
1110 data asl,10,brk,0,clc,24       <ab>
1120 data cld,216,cli,88,clv,184    <kg>
1130 data dex,202,dey,136,inx,232   <lf>
1140 data iny,200,lsr,74,nop,234    <nn>
1150 data rol,42,ror,106,sec,56     <if>
1160 data sed,248,sei,120,tax,170   <pg>
1170 data tay,168,txa,138,tya,152   <cg>
1180 iflen(x$)>4then1340            <pg>
1190 y$=x$:restore1110:m=21:gosub1
050                                 <gc>
1200 ifi=51thenpoke12046,a:poke120
47,234:poke12048,234:sys12032:goto
950                                 <gg>
1210 rem ------------------------   <pj>
1220 rem  sonderbehandlung          <mg>
1230 rem ------------------------   <ib>
1240 m=6:gosub1050:ifi<>51then1070  <dh>
1250 sysa:goto950                   <ei>
1260 data pha,12066,pla,12079,php,
12104,plp,12110,tsx,12123,txs,1214
2                                   <ai>
1270 rem ------------------------   <le>
1280 rem  zwei-byte-befehle         <lk>
1290 rem ------------------------   <ii>
1300 data adc,105,and,41,cmp,201    <go>
1310 data cpx,224,cpy,192,eor,73    <lp>
1320 data lda,169,ldx,162,ldy,160   <kc>
1330 data ora,9,sbc,233             <gf>
1340 ifleft$(right$(x$,4),2)<>"#$"
then1510                            <on>
1350 restore1300:y$=left$(x$,3)     <ig>
1360 m=11:gosub1050:ifi<>51then107
0                                   <mc>
1370 x=dec(right$(x$,2))            <jg>
1380 poke12046,a:poke12048,234      <po>
1390 poke12047,x:sys12032:goto950   <ja>
1400 rem ------------------------   <ea>
1410 rem  drei-byte-befehle         <gp>
1420 rem ------------------------   <jc>
1430 data adc,109,and,45,asl,14     <eh>
1440 data bit,44,cmp,205,dec,206    <nc>
1450 data eor,77,inc,238,jsr,32     <fh>
1460 data lda,173,ldx,174,ldy,172   <he>
1470 data lsr,78,ora,13,rol,46      <in>
1480 data ror,110,sbc,237,sta,141   <hg>
1490 data stx,142,sty,140,cpx,224   <ll>
1500 data cpy,192                   <em>
1510 ifleft$(right$(x$,5),1)<>"$"t
hen1070                             <ba>
1520 restore1430:y$=left$(x$,3)     <if>
1530 m=22:gosub1050:ifi<>51then107
0                                   <cg>
1540 y$=right$(x$,4)                <oh>
1550 poke12048,dec(left$(y$,2))     <ff>
1560 poke12047,dec(right$(y$,2))    <np>
1570 poke12046,a                    <fl>
1580 sys12032:goto950               <pc>
1590 rem ========================   <mi>
1600 rem  p r o g r a m m e n d e   <of>
1610 rem ========================   <pd>
```



```
10 rem cpu-trainer==========c64     <dn>
20 rem (p) commodore welt team      <ho>
30 rem ==========================  <ng>
40 rem (c) by alfons mittelmeyer    <cg>
50 rem                              <pd>
60 rem                              <ah>
70 rem basic v2.0                   <nc>
80 rem c64                          <cg>
90 rem ==========================  <jg>
100 poke55,0:poke56,47              <ac>
110 gosub60000:fori=12032to12162    <he>
115 reada:pokei,a:next              <mo>
120 data 173,120,047,072,173,117    <ck>
125 data 047,174,118,047,172,119    <cd>
130 data 047,040,076,050,003,141    <ac>
135 data 117,047,142,118,047,140    <lp>
140 data 119,047,008,104,141,120    <hi>
145 data 047,088,216,096,173,117    <kj>
150 data 047,174,121,047,157,000    <gk>
155 data 001,206,121,047,096,238    <nm>
160 data 121,047,174,121,047,173    <pa>
165 data 120,047,072,040,189,000    <aa>
170 data 001,141,117,047,008,104    <mc>
175 data 141,120,047,088,216,096    <if>
180 data 173,120,047,024,144,215    <ok>
185 data 238,121,047,174,121,047    <ie>
190 data 189,000,001,141,120,047    <pj>
195 data 096,173,120,047,072,040    <hf>
200 data 173,121,047,141,118,047    <jm>
205 data 008,104,141,120,047,088    <ek>
210 data 024,096,173,118,047,141    <ab>
215 data 121,047,096,000,000,000    <aa>
220 data 000,000,165,122,133,065    <ic>
225 data 165,123,133,066,096        <cb>
330 sys12049:poke12153,127          <fa>
340 rem ------------------------    <ip>
```

```
350 rem  variablen                    <oo>
360 rem ------------------------      <gg>
370 b$=chr$(32):b4$=b$+b$+b$+b$       <cg>
380 a$(0)="aiiiiiiiib"                <cd>
390 a$(1)="j00000000j"                <ff>
400 a$(2)="diiiiieiif"                <kh>
410 a$(3)="000000j00j"                <dl>
420 a$(4)="00000agiif"                <eh>
430 a$(5)="00000j000j"                <en>
440 a$(6)="00000diiic"                <bc>
450 data 032,176,174,189,173          <hf>
460 data 178,179,177,171,192,221      <fl>
470 g$="":fori=0to10:reada:g$=g$+c
hr$(a):next                           <il>
480 forj=0to6:z$="":fori=1to10        <hf>
490 z$=z$+mid$(g$,((asc(mid$(a$(j)
,i,1))and15))+1,1):next:a$(j)=z$:n
extj                                  <ki>
500 b$(0)="binaer":b$(1)="hex"        <cp>
510 b$(2)="dezimal"                   <gd>
520 c$="akku"+b4$+"x-register y-re
gister"                               <le>
530 d$="nv--dizc"+b4$+"stapel"        <fh>
540 rem ------------------------      <eg>
550 rem  bildschirmaufbau             <mp>
560 rem ------------------------      <eh>
570 sys58692                          <fb>
580 gosub910:y=3:gosub900             <ll>
590 o=10:p=1:z$=c$:gosub1581:x=7:y
=2:gosub890                           <eg>
600 x=18:gosub890:x=29:gosub890       <mn>
610 y=12:gosub900                     <ea>
620 o=8:p=10:z$=d$:gosub1581:x=7:y
=11:gosub890                          <fj>
630 x=18:gosub890:o=0:p=18:z$="":g
osub1581:gosub910                     <ki>
640 x$="":gosub911                    <me>
650 rem ------------------------      <fi>
660 rem  eingabe                      <mc>
670 rem ------------------------      <mk>
680 ifx$="brk"then570                 <ig>
681 o=0:p=19:z$=bl$:gosub1581         <jh>
690 gosub1581:z$="? "+chr$(164):go
sub1581                               <bo>
700 y$=" ":x=1                        <id>
710 sys61762:a=peek(780):if(a<32an
da<>13anda<>20)ora>95then710          <ni>
720 ifa=13then800                     <fj>
730 ifa<>20theny$=y$+chr$(a):x=x+1
:goto760                              <hk>
740 ifx=1then700                      <an>
750 x=x-1:y$=left$(y$,x)              <ja>
760 poke780,157:sys61906:poke780,a
:sys61906                             <bl>
765 poke212,0:poke780,164:sys61906
:goto710                              <aj>
770 rem ------------------------      <ob>
780 rem  eingabe bearbeiten           <mg>
790 rem ------------------------      <io>
800 ify$<>" "thenx$=right$(y$,x-1)    <kh>
810 poke780,13:sys61906               <on>
820 ifx$="brk"thensys58692            <df>
850 gosub1101:goto680                 <il>
860 rem ------------------------      <ie>
870 rem  routinensammlung             <pe>
880 rem ------------------------      <ad>
890 fori=0to6:o=x:p=y+i:z$=a$(i):g
osub1581:next:return                  <pp>
900 fori=0to2:o=0:p=y+i+i:z$=b$(i)
:gosub1581:next:return                <hm>
910 poke780,61:fori=1to40:sys61906
:next:return                          <ek>
911 sys12154                          <ho>
920 data 12149,7,2,12150,18,2         <pn>
930 data 12151,29,2,12152,7,11        <of>
940 data 12153,18,11                  <om>
950 ifx$="brk"thenreturn              <fj>
960 forz=1to5:reada                   <oa>
970 readx:ready:a=peek(a)             <nf>
980 o=x+1:p=y+1:z$="":gosub1581:s=
a:n=128                               <gc>
990 fori=1to8:a$="0"                  <ma>
1000 ifs>=nthens=s-n:a$="1"           <nd>
1010 o=x+i:p=y+1:z$=a$:gosub1581:n
=n/2:next                             <mg>
1020 o=x+7:p=y+3:gosub1600:gosub15
81                                    <ce>
1030 o=x+6:p=y+5:z$=right$(" "+str
$(a),3):gosub1581                     <fl>
1040 nextz:return                     <mb>
1050 x=-1:fori=1tom:reada$:reada:i
fa$=y$theni=50                        <em>
1060 next:return                      <en>
1070 o=0:p=19:z$="fehler":gosub158
1:poke198,0:wait198,1:poke198,0:re
turn                                  <io>
1080 rem ------------------------     <fh>
1090 rem  ein-byte-befehle            <bm>
1100 rem ------------------------     <om>
1101 sys12154                         <ap>
1110 data asl,10,brk,0,clc,24         <ab>
1120 data cld,216,cli,88,clv,184      <kg>
1130 data dex,202,dey,136,inx,232     <lf>
1140 data iny,200,lsr,74,nop,234      <nn>
1150 data rol,42,ror,106,sec,56       <if>
1160 data sed,248,sei,120,tax,170     <pg>
1170 data tay,168,txa,138,tya,152     <cg>
1180 iflen(x$)>4then1291              <pn>
1190 y$=x$:m=21:gosub1050             <om>
1200 ifi=51thenpoke12046,a:poke120
47,234:poke12048,234:sys12032:goto
911                                   <fp>
1210 rem ------------------------     <pj>
1220 rem  sonderbehandlung            <mg>
1230 rem ------------------------     <ib>
1240 m=6:gosub1050:ifi<>51then1070    <dh>
```

```
1250 sysa:goto911                          <eb>
1260 data pha,12066,pla,12079,php,
12104,plp,12110,tsx,12123,txs,1214
2                                          <ai>
1270 rem ------------------------          <le>
1280 rem  zwei-byte-befehle                <lk>
1290 rem ------------------------          <ii>
1291 sys12154                             <np>
1300 data adc,105,and,41,cmp,201           <go>
1310 data cpx,224,cpy,192,eor,73           <lp>
1320 data lda,169,ldx,162,ldy,160          <kc>
1330 data ora,9,sbc,233                    <gf>
1340 ifleft$(right$(x$,4),2)<>"#$"
then1421                                   <om>
1350 y$=left$(x$,3)                        <dk>
1360 m=11:gosub1050:ifi<>51then107
0                                          <mc>
1370 q$=right$(x$,2):gosub1587:x=q         <ni>
1380 poke12046,a:poke12048,234             <po>
1390 poke12047,x:sys12032:goto911          <ij>
1400 rem ------------------------          <ea>
1410 rem  drei-byte-befehle                <gp>
1420 rem ------------------------          <jc>
1421 sys12154                             <ho>
1430 data adc,109,and,45,asl,14            <eh>
1440 data bit,44,cmp,205,dec,206           <nc>
1450 data eor,77,inc,238,jsr,32            <fh>
1460 data lda,173,ldx,174,ldy,172          <he>
1470 data lsr,78,ora,13,rol,46             <in>
1480 data ror,110,sbc,237,sta,141          <hg>
1490 data stx,142,sty,140,cpx,224          <ll>
1500 data cpy,192                          <em>
1510 ifleft$(right$(x$,5),1)<>"$"t
hen1070                                    <ba>
1520 y$=left$(x$,3)                        <ip>
1530 m=22:gosub1050:ifi<>51then107
0                                          <cg>
1540 y$=right$(x$,4)                       <oh>
1550 q$=left$(y$,2):gosub1587:poke
12048,q                                    <jf>
1560 q$=right$(y$,2):gosub1587:pok
e12047,q                                   <bp>
1570 poke12046,a                           <fl>
1580 sys12032:goto911                      <ol>
1581 poke783,peek(783)and254               <df>
1582 poke781,p:poke782,o:sys65520          <np>
1583 ifz$=""thenreturn                     <gh>
1584 forq=1tolen(z$):p$=mid$(z$,q,
1)                                         <lf>
1585 poke780,asc(p$):sys61906:next         <bl>
1586 return                                <po>
1587 q=asc(left$(q$,1))-48                 <pb>
1588 q=q+(q>9)*7                           <og>
1589 r=asc(right$(q$,1))-48                <ce>
1590 r=r+(r>9)*7:q=16*q+r:return            <bh>
1600 q=int(a/16):r=a-16*q                  <di>
1610 q=q-(q>9)*7:r=r-(r>9)*7               <kj>
1620 z$=chr$(q+48)+chr$(r+48):retu
rn                                         <mo>
1630 bl$=chr$(32):fori=1to4                <jj>
1640 bl$=bl$+bl$:next:return               <oo>
1650 rem ========================          <bi>
1660 rem  p r o g r a m m e n d e          <jf>
1670 rem ========================          <jd>
```

KURZ, KÜRZER, AM KÜRZESTEN

BASIC-Eingabe mit Abkürzungen der BASIC-Schlüsselwörter

139 Abkürzungen müssen Sie beim 128PC lernen, wenn Sie zum Editieren von BASIC-Programmen die BASIC-Kürzel optimal einsetzen wollen. Natürlich sind einige dieser Codes zum Überfluß auch noch in den verschiedenen BASIC-Versionen 7.0/3.5/2.0 mit unterschiedlichen Kürzeln belegt.

Wenn die Computer-Hersteller es schon untereinander nicht schaffen, eine wie auch immer geartete Normung zu vereinbaren – die wirtschaftlichen Gründe dafür sind bekannt – sollte man eigentlich davon ausgehen können, daß dies wenigstens im Geräteangebot eines Herstellers möglich wäre. Weit gefehlt, denn auch in sogenannten unwichtigen Details, wie Abkürzungen von BASIC-Schlüsselwörtern beim Editieren, sind einige gravierende Unterschiede festzustellen.

BASIC-VERSIONEN MIT TOKENS-MÄNGELN

Daß ein BASIC 7.0 des 128PC einen größeren Befehlsumfang als das schmalbrüstige BASIC 2.0 des C64, das angejahrte BASIC 4.0 der CBM-8000-Serie oder das schon bessere BASIC 3.5 der C16/Plus4-Reihe hat, ist ja landläufig bekannt. Daß aber für die Versionen 2.0/3.5/4.0/7.0 auch unterschiedliche Abkürzungen für die BASIC-Schlüsselwörter der Commodore-Rechner verwendet wurden, wissen nur die, die mehrere Computer mit unterschiedlichen BASIC-Versionen besitzen. Diese User haben wohl bei mancher Eingabe einige unliebsame Überraschungen erlebt, denn der entsprechende BASIC-Interpreter kann nur seine ihm verständliche BASIC-Version erkennen. Dies trifft natürlich auch zu, wenn höherwertige BASIC-Versionen in nieder- bzw. anderswertige geladen werden. Die wahrscheinliche Folge: Programmzerstörung.

UNTERSCHIEDLICHE BASIC-KÜRZEL

Zum Beispiel ist der BASIC-Befehl „BOX" in der 2.0-BASIC-Version des C64 nicht enthalten, in der 3.5-BASIC-Version der C16/Plus4-Rechner kann man den BASIC-Befehl „BOX" mit B SHIFT O abkürzen. Dieses Kürzel ist in der 7.0-BASIC-Version des 128PC nicht möglich, da hier eine Abkürzung für „BOX" nicht existiert. B SHIFT O steht vielmehr für den 7.0-Befehl „BOOT". Im 128PC-Modus sind für folgende BASIC-Schlüsselwörter keine Abkürzungen möglich:
BOX – CONT – COS – DEC – DIM – DO – DS$ – EL – ELSE – END – ER – ERR$ – EXIT –

Abkürzungen von BASIC-Schlüsselwörtern und deren TOKEN und VORTOKEN in Dezimal und Hexadezimal bei Commodore Computern CBM8032/VC20/C64/C16/C116/Plus4/128PC

BASIC EDIT	BASIC 7.0 128PC		BASIC 4.0 CBM8032		BASIC 3.5 C16/C116/P4		BASIC 2.0 VC20/64		TOKENS/DEZIMAL VT 7.0/4.0/3.5/2.0	TOKENS/HEXADEZIMAL VT 7.0/4.0/3.5/2.0
ABS	A	SHIFT B	A	SHIFT B	A	SHIFT B	A	SHIFT B	182/182/182/182	B6/ B6/ B6/ B6
AND	A	SHIFT N	A	SHIFT N	A	SHIFT N	A	SHIFT N	175/175/175/175	AF/ AF/ AF/ AF
APPEND	A	SHIFT P	A	SHIFT P		keine		keine	254 14 /212/---/---	FE 0E/ D4/ --/ --
ASC	A	SHIFT S	A	SHIFT S	A	SHIFT S	A	SHIFT S	198/198/198/198	C6/ C6/ C6/ C6
ATN	A	SHIFT T	A	SHIFT T	A	SHIFT T	A	SHIFT T	193/193/193/193	C1/ C1/ C1/ C1
AUTO	A	SHIFT U		keine	A	SHIFT U		keine	220/---/220/---	DC/ --/ DC/ --
BACKUP	BA	SHIFT C	B	SHIFT A	B	SHIFT A		keine	246/210/246/---	F6/ D2/ F6/ --
BANK	B	SHIFT A		keine		keine		keine	254 02 /---/---/---	FE 02/ --/ --/ --
BEGIN	B	SHIFT E		keine		keine		keine	254 24 /---/---/---	FE 18/ --/ --/ --
BEND	BE	SHIFT N		keine		keine		keine	254 25 /---/---/---	FE 19/ --/ --/ --
BLOAD	B	SHIFT L		keine		keine		keine	254 17 /---/---/---	FE 11/ --/ --/ --
BOOT	B	SHIFT O		keine		keine		keine	254 27 /---/---/---	FE 1B/ --/ --/ --
BOX		keine		keine	B	SHIFT O		keine	225/---/225/---	E1/ -- E1/ --
BSAVE	B	SHIFT S		keine		keine		keine	254 16 /---/---/---	FE 10/ --/ --/ --
BUMP	B	SHIFT U		keine		keine		keine	206 03 /---/---/---	CE 03/ --/ --/ --
CATALOG	C	SHIFT A	C	SHIFT A		keine		keine	254 12 /215/---/---	FE 0C/ D7/ --/ --
CHAR	CH	SHIFT A		keine	CH	SHIFT A		keine	224/---/224/---	E0/ --/ E0/ --
CHR$	C	SHIFT H	C	SHIFT H	C	SHIFT H	C	SHIFT H	199/199/199/199	C7/ C7/ C7/ C7
CIRCLE	C	SHIFT I		keine	C	SHIFT I		keine	226/---/226/---	E2/ --/ E2/ --
CLOSE	CL	SHIFT O	CL	SHIFT O	CL	SHIFT O	CL	SHIFT O	160/160/160/160	A0/ A0/ A0/ A0
CLR	C	SHIFT L	C	SHIFT L	C	SHIFT L	C	SHIFT L	156/156/156/156	9C/ 9C/ 9C/ 9C
CMD	C	SHIFT M	C	SHIFT M	C	SHIFT M	C	SHIFT M	157/157/157/157	9D/ 9D/ 9D/ 9D
COLLECT	COLL	SHIFT E	COL	SHIFT L	COL	SHIFT L		keine	243/209/243/---	F3/ D1/ F3/ --
COLLISION	CO	SHIFT L		keine		keine		keine	254 23 /---/---/---	FE 17/ --/ --/ --
COLOR	COL	SHIFT O		keine	CO	SHIFT L		keine	231/---/231/---	E7/ --/ E7/ --
CONCAT	C	SHIFT O	CON	SHIFT C		keine		keine	254 19 /204/---/---	FE 13/ CC/ --/ --
CONT		keine	C	SHIFT O	C	SHIFT O	C	SHIFT O	154/154/154/154	9A/ 9A/ 9A/ 9A
COPY	CO	SHIFT P	CO	SHIFT P		keine		keine	244/211/---/---	F4/ D3/ --/ --
COS		keine		keine		keine		keine	190/190/190/190	BE/ BE/ BE/ BE
DATA	D	SHIFT A	D	SHIFT A	D	SHIFT A	D	SHIFT A	131/131/131/131	83/ 83/ 83/ 83
DCLEAR	DCL	SHIFT E		keine		keine		keine	254 21 /---/---/---	FE 15/ --/ --/ --
DCLOSE	D	SHIFT C	D	SHIFT C		keine		keine	254 15 /206/---/---	FE 0F/ CE/ --/ --
DEC		keine		keine	D	SHIFT C		keine	209/---/209/---	D1/ --/ D1/ --
DEF	D	SHIFT E	D	SHIFT E	D	SHIFT E	D	SHIFT E	150/150/150/150	96/ 96/ 96/ 96
DELETE	DE	SHIFT L		keine	DE	SHIFT L		keine	247/---/247/---	F7/ --/ F7/ --
DIM		keine	D	SHIFT I	D	SHIFT I	D	SHIFT I	134/134/134/134	86/ 86/ 86/ 86
DIRECTORY	DI	SHIFT R	DI	SHIFT R	D	SHIFT R		keine	238/218/238/---	EE/ DA/ EE/ --
DLOAD	D	SHIFT L	D	SHIFT L	D	SHIFT L		keine	240/214/240/---	F0/ D6/ F0/ --
DO		keine		keine		keine		keine	235/---/235/---	EB/ --/ EB/ --
DOPEN	D	SHIFT O	D	SHIFT O		keine		keine	254 13 /205/---/---	FE 0D/ CD/ --/ --
DRAW	D	SHIFT R		keine	D	SHIFT R		keine	229/---/229/---	E5/ --/ E5/ --
DS		keine		keine		keine		keine	68 83	44 53
DS$		keine		keine		keine		keine	68 83 36	44 53 24
DSAVE	D	SHIFT S	D	SHIFT S	D	SHIFT S		keine	239/213/239/---	EF/ D5/ EF/ --
DVERIFY	D	SHIFT V		keine		keine		keine	254 20 /---/---/---	FE 14/ --/ --/ --

EL		keine		keine		keine		keine		69 76			45 4C	
ELSE		keine		keine		keine		keine		213/---/213/---			D5/ --/ D5/ --	
END		keine	E	SHIFT N	E	SHIFT N	E	SHIFT N		128/128/128/128			80/ 80/ 80/ 80	
ENVELOPE	E	SHIFT N		keine		keine		keine	254	10 /---/---/---		FE	0A/ --/ --/ --	
ER		keine		keine		keine		keine		69 82			45 52	
ERR$		keine		keine	E	SHIFT R		keine		211/---/211/---			D3/ --/ D3/ --	
EXIT		keine		keine		keine		keine		237/---/237/---			ED/ --/ ED/ --	
EXP	E	SHIFT X	E	SHIFT X	E	SHIFT X	E	SHIFT X		189/189/189/189			BD/ BD/ BD/ BD	
FAST	F	SHIFT A		keine		keine		keine	254	37 /---/---/---		FE	25/ --/ --/ --	
FETCH	F	SHIFT E		keine		keine		keine	254	33 /---/---/---		FE	21/ --/ --/ --	
FILTER	F	SHIFT I		keine		keine		keine	254	03 /---/---/---		FE	03/ --/ --/ --	
FN		keine		keine		keine		keine		165/165/165/165			A5/ A5/ A5/ A5	
FOR	F	SHIFT O	F	SHIFT O	F	SHIFT O	F	SHIFT O		129/129/129/129			81/ 81/ 81/ 81	
FRE	F	SHIFT R	F	SHIFT R	F	SHIFT F	F	SHIFT R		184/184/184/184			B8/ B8/ B8/ B8	
GET	G	SHIFT E	G	SHIFT E	G	SHIFT E	G	SHIFT E		161/161/161/161			A1/ A1/ A1/ A1	
GET#	G	SHIFT E#	G	SHIFT E#	G	SHIFT E#	G	SHIFT E#		161 35 /			A1 23	
GETKEY	GETK	SHIFT E		keine	GETK	SHIFT E		keine		161 249/-/161 249/-			A1 F9/-/A1 F9/-	
GO		keine		keine		keine		keine		203/203/203/203			CB/ CB/ CB/ CB	
GO64		keine		keine		keine		keine		203 54 52/-/-/-			CB 36 34/-/-/-	
GOSUB	GO	SHIFT S	GO	SHIFT S	GO	SHIFT S	GO	SHIFT S		141/141/141/141			8D/ 8D/ 8D/ 8D	
GOTO	G	SHIFT O	G	SHIFT O	G	SHIFT O	G	SHIFT O		137/137/137/137			89/ 89/ 89/ 89	
GRAPHIC	G	SHIFT R		keine	G	SHIFT R		keine		222/---/222/---			DE/ DE/ DE/ DE	
GSHAPE	G	SHIFT S		keine	G	SHIFT S		keine		227/---/227/---			E3/ --/ E3/ --	
HEADER	HE	SHIFT A	HE	SHIFT A	HE	SHIFT A		keine		241/208/241/---			F1/ D0/ F1/ --	
HELP	HE	SHIFT L		keine		keine		keine		234/---/234/---			EA/ --/ EA/ --	
HEX$	H	SHIFT E		keine	H	SHIFT E		keine		210/---/210/---			D2/ --/ D2/ --	
IF		keine		keine		keine		keine		139/139/139/139			8B/ 8B/ 8B/ 8B	
INPUT		keine		keine		keine		keine		133/133/133/133			85/ 85/ 85/ 85	
INPUT#	I	SHIFT N	I	SHIFT N	I	SHIFT N	I	SHIFT N		132/132/132/132			84/ 84/ 84/ 84	
INSTR	IN	SHIFT S		keine	IN	SHIFT S		keine		212/---/212/---			D4/ --/ D4/ --	
INT		keine		keine		keine		keine		181/181/181/181			B5/ B5/ B5/ B5	
JOY	J	SHIFT O		keine	J	SHIFT O		keine		207/---/207/---			CF/ --/ CF/ --	
KEY	K	SHIFT E		keine	K	SHIFT E		keine		249/---/249/---			F9/ --/ F9/ --	
LEFT$	LE	SHIFT F	LE	SHIFT F	LE	SHIFT F	LE	SHIFT F		200/200/200/200			C8/ C8/ C8/ C8	
LEN		keine		keine		keine		keine		195/195/195/195			C3/ C3/ C3/ C3	
LET	L	SHIFT E	L	SHIFT E	L	SHIFT E	L	SHIFT E		136/136/136/136			88/ 88/ 88/ 88	
LIST	L	SHIFT I	L	SHIFT I	L	SHIFT I	L	SHIFT I		155/155/155/155			98/ 98/ 98/ 98	
LOAD	L	SHIFT O	L	SHIFT O	L	SHIFT O	L	SHIFT O		147/147/147/147			93/ 93/ 93/ 93	
LOCATE	LO	SHIFT C		keine	LO	SHIFT C		keine		230/---/230/---			E6/ --/ E6/ --	
LOG		keine		keine		keine		keine		188/188/188/188			BC/ BC/ BC/ BC	
LOOP	LO	SHIFT O		keine	LO	SHIFT O		keine		236/---/236/---			EC/ --/ EC/ --	
MID$	M	SHIFT I	M	SHIFT I	M	SHIFT I	M	SHIFT I		202/202/202/202			CA/ CA/ CA/ CA	
MONITOR	MO	SHIFT N		keine	M	SHIFT O		keine		250/---/250/250			FA/ --/ FA/ --	
MOVESPR	M	SHIFT O		keine		keine		keine	254	06 /---/---/---		FE	06/ --/ --/ --	
NEW		keine		keine		keine		keine		162/162/162/162			A2/ A2/ A2/ A2	
NEXT	N	SHIFT E	N	SHIFT E	N	SHIFT E	N	SHIFT E		130/130/130/130			82/ 82/ 82/ 82	
NOT	N	SHIFT O	N	SHIFT O	N	SHIFT O	N	SHIFT O		168/168/168/168			A8/ A8/ A8/ A8	
OFF		keine		keine		keine		keine	254	36 /---/---/---		FE	24/ --/ --/ --	
ON		keine		keine		keine		keine		145/145/145/145			91/ 91/ 91/ 91	
OPEN	O	SHIFT P	O	SHIFT P	O	SHIFT P	O	SHIFT P		159/159/159/159			9F/ 9F/ 9F/ 9F	
OR		keine		keine		keine		keine		176 176/176/176			B0/ B0/ B0/ B0	
PAINT	P	SHIFT A		keine	P	SHIFT A		keine		223/---/223/---			DF/ --/ DF/ --	
PEEK	PE	SHIFT E	P	SHIFT E	P	SHIFT E	P	SHIFT E		194/194/194/194			C2/ C2/ C2/ C2	
PEN	P	SHIFT E		keine	·	keine		keine	206	04 /---/---/---		CE	04/ --/ --/ --	
PLAY	P	SHIFT L		keine		keine		keine	254	04 /---/---/---		FE	04/ --/ --/ --	
POINTER	PO	SHIFT I		keine		keine		keine	206	10 /---/---/---		CE	0A/ --/ --/ --	
POKE	PO	SHIFT K	P	SHIFT O	P	SHIFT O	P	SHIFT O		151/151/151/151			97/ 97/ 97/ 97	
POS(		keine		keine		keine		keine		185/185/185/185			B9/ B9/ B9/ B9	
POT	P	SHIFT O		keine		keine		keine	206	02 /---/---/---		CE	02/ --/ --/ --	
PRINT	?		?		?		?			153/153/153/153			99/ 99/ 99/ 99	
PRINT#	P	SHIFT R	P	SHIFT R	P	SHIFT R	P	SHIFT R		152/152/152/152			98/ 98/ 98/ 98	
PRINT USING	?US	SHIFT I		keine	?US	SHIFT I		keine		153 251/-/153 251/-			99 FB/-/99 FB/-	
PUDEF	P	SHIFT U		keine	P	SHIFT U		keine		221/---/221/---			DD/ --/ DD/ --	

FN – GO – GO64 – IF – INPUT – INT – LEN – LOG – NEW – OFF – ON – OR – POS(– QUIT – REM – SPC(– ST – TAN – TI – TI$ – TO

In der nachfolgenden Vergleichsliste sind die möglichen Abkürzungen aller vier BASIC-Versionen 2.0/3.5/4.0/7.0 enthalten, vorausgesetzt, das BASIC-Schlüsselwort ist in Ihrem BASIC-Interpreter enthalten beziehungsweise mit Ihrer Konfiguration überhaupt möglich. Nun viel Spaß mit dem Auswendiglernen der Kürzel, denn Sie sollten schon sattelfest bei Ihrer mühsamen Editierarbeit sein. Alleine BASIC 7.0 hat satte 139 Abkürzungen, bei Version 3.5 gibt es „nur" 98 und bei BASIC 2.0 sind es gar lächerliche 52 Abkürzungen.

DIE TOKEN-LISTE KANN ERWEITERT WERDEN

BASIC-Befehle werden, zur Einsparung von Speicherplatz und zur Beschleunigung der Programmausführung, in einen bestimmten Code umgewandelt, den Tokens. Das heißt, es erscheint im Speicher nicht das gesamte Wort, sondern ein Byte (TOKEN) – beziehungsweise zwei Byte (VORTOKEN/ TOKEN) beim 128PC – als Token-Codezahl. Dies spart natürlich Speicherplatz, denn das Wort "PRINT" in Anführungszeichen wird als HEX $50 $52 $49 $4E $54 / DEZ 80 82 73 78 84 (5 Byte), das Schlüsselbefehlswort PRINT aber als HEX $99/ DEZ 153 (1 Byte) abgespeichert. Der 7.0-BASIC-Befehlsumfang ist aber für die alte Tokenliste – vom 4.0 BASIC des CBM 8032 von HEX $80 bis DB (DEZ 128 bis 219); oder vom 2.0 BASIC des C64 die von HEX $80 bis $CB (DEZ 128 bis 203), oder vom 3.5 BASIC des C16/ Plus4 HEX $80 bis $FD (DEZ 128 bis 253) – viel zu klein, denn bei $FF

(255) ist die Fahnenstange zu Ende. Aber mit einem Trick (Vor-Token) kann die Tokenliste erweitert werden: $CE (206) & SFE (254) sind die Vor-Token-Adressen des Commodore 128PC, die sich in weitere 47 BASIC-Schlüsselbefehlswörter verzweigen.
An Hand eines Beispiels wollen wir die Eingabevorteile von Abkürzungen aufzeigen. Wenn Programmzeilen mit Kürzeln eingegeben werden, wandelt der BASIC-Interpreter, bei der Übernahme in seinen Arbeitsspeicher, die Programmzeilen wie folgt um – vorausgesetzt, die angewendeten Abkürzungen sind in seiner Tokenliste enthalten und keine SYNTAX-Eingabefehler wurden begangen.

Eingabe:

```
10?cH(147)"Commodore
   Welt"
20ifx>ytH?"GROESSER"
   :elseify>xtH?"
   "KLEINER":else?
   "GLEICH"
```

Ergebnis:

```
10 printchr$(147)
    "Commodore Welt"
20 ifx>ythenprint
    "GROESSER":else
    ify>xthenprint
    "KLEINER":else
    print"GLEICH"
```

Beim Editieren wurden 82 Zeichen eingegeben, die vom Interpreter dann in 106 Zeichen umgewandelt wurden. Also 24 Zeichen bei nur zwei Programmzeilen gespart. Wenn Sie jetzt an Programme mit mehr als zwei Programmzeilen denken und eine mögliche Einsparung von bis 25 Prozent sehen, verlieren auch 30-KByte-Listings ihren Schrecken, der von der Eingabe abhält. Der bequemere Weg ist natürlich, sich das Programm zu kaufen, aber der Lerneffekt bei der Eingabe und der anschließenden Fehlersuche, ist bei Selbsteditiertem wesentlich besser.

QUIT		keine		keine		keine		keine		254 30 /---/---/---	FE 1E/ --/ --/ --
RCLR(n)	R	SHIFT C		keine	R	SHIFT C		keine		205/---/205/---	CD/ --/ CD/ --
RDOT(n)	R	SHIFT D		keine	R	SHIFT D		keine		208/---/208/---	D0/ --/ D0/ --
READ	RE	SHIFT A	R	SHIFT E	R	SHIFT E	R	SHIFT E		135/135/135/135	87/ 87/ 87/ 87
RECORD	R	SHIFT E	RE	SHIFT C		keine		keine		254 18 /207/---/---	FE 12/ CF/ --/ --
REM		keine		keine		keine		keine		143/143/143/143	8F/ 8F/ 8F/ 8F
RENAME	RE	SHIFT N	RE	SHIFT N	RE	SHIFT N		keine		245/216/245/---	F5/ D8/ F5/ --
RENUMBER	REN	SHIFT U		keine	REN	SHIFT U		keine		248/---/248/---	F8/ --/ F8/ --
RESTORE	RE	SHIFT S	RE	SHIFT S	RE	SHIFT S	RE	SHIFT S		140/140/140/140	8C/ 8C/ 8C/ 8C
RESUME	RES	SHIFT U		keine	RES	SHIFT U		keine		214/---/214/---	D6/ --/ D6/ --
RETURN	RE	SHIFT T	RE	SHIFT T	RE	SHIFT T	RE	SHIFT T		142/142/142/142	8E/ 8E/ 8E/ 8E
RGR	R	SHIFT G		keine	R	SHIFT G		keine		204/---/204/---	CC/ --/ CC/ --
RIGHT$	R	SHIFT I	R	SHIFT I	R	SHIFT I	R	SHIFT I		201/201/201/201	C9/ C9/ C9/ C9
RLUM		keine		keine	R	SHIFT L		keine		---/---/206/---	--/ --/ CE/ --
RND	R	SHIFT N	R	SHIFT N	R	SHIFT N	R	SHIFT N		187/187/187/187	BB/ BB/ BB/ BB
RREG	R	SHIFT R		keine		keine		keine		254 09 /---/---/---	FE 09/ --/ --/ --
RSPCOLOR	RSP	SHIFT C		keine		keine		keine		206 07 /---/---/---	CE 07/ --/ --/ --
RSPPOS	R	SHIFT S		keine		keine		keine		206 05 /---/---/---	CE 05/ --/ --/ --
RSPRITE	RSP	SHIFT R		keine		keine		keine		206 06 ---/---/---	CE 06/ --/ --/ --
RUN	R	SHIFT U	R	SHIFT U	R	SHIFT U	R	SHIFT U		138/138/138/138	8A/ 8A/ 8A/ 8A
RWINDOW	R	SHIFT W		keine		keine		keine		206 09 /---/---/---	CE 06/ --/ --/ --
SAVE	S	SHIFT A	S	SHIFT A	S	SHIFT A	S	SHIFT A		148/148/148/148	94/ 94/ 94/ 94
SCALE	SC	SHIFT A		keine	SC	SHIFT A		keine		233/---/233/---	E9/ --/ E9/ --
SCNCLR	S	SHIFT C		keine	S	SHIFT C		keine		232/---/232/---	E8/ --/ E8/ --
SCRATCH	SC	SHIFT R	SC	SHIFT R	SC	SHIFT R		keine		242/217/242/---	F2/ D9/ F2/ --
SGN	S	SHIFT G	S	SHIFT G	S	SHIFT G	S	SHIFT G		180/180/180/180	B4/ B4/ B4/ B4
SIN	S	SHIFT I	S	SHIFT I	S	SHIFT I	S	SHIFT I		191/191/191/191	BF/ BF/ BF/ BF
SLEEP	S	SHIFT L		keine		keine		keine		254 11 /---/---/---	FE 0B/ --/ --/ --
SLOW	SL	SHIFT O		keine		keine		keine		254 38 /---/---/---	FE 26/ --/ --/ --
SOUND	S	SHIFT O		keine	S	SHIFT O		keine		218/---/218/---	DA/ --/ DA/ --
SPC(		keine	S	SHIFT P	S	SHIFT P	S	SHIFT P		166/166/166/166	A6/ A6/ A6/ A6
SPRCOLOR	SPR	SHIFT C		keine		keine		keine		254 08 /---/---/---	FE 08/ --/ --/ --
SPRDEF	SPR	SHIFT D		keine		keine		keine		254 29 /---/---/---	FE 1D/ --/ --/ --
SPRITE	S	SHIFT P		keine		keine		keine		254 07 /---/---/---	FE 07/ --/ --/ --
SPRSAV	SPR	SHIFT S		keine		keine		keine		254 22 /---/---/---	FE 16/ --/ --/ --
SQR	S	SHIFT Q	S	SHIFT Q	S	SHIFT Q	S	SHIFT Q		186/186/186/186	BA/ BA/ BA/ BA
SSHAPE	S	SHIFT S		keine	S	SHIFT S		keine		228/---/228/---	E4/ --/ E4/ --
ST		keine		keine		keine		keine		83 84	53 54
STASH	S	SHIFT T		keine		keine		keine		254 31 /---/---/---	FE 1F/ --/ --/ --
STEP	ST	SHIFT E	ST	SHIFT E	ST	SHIFT E	ST	SHIFT E		169/169/169/169	A9/ A9/ A9/ A9
STOP	ST	SHIFT O	ST	SHIFT O	ST	SHIFT O	ST	SHIFT O		144/144/144/144	90/ 90/ 90/ 90
STR$	ST	SHIFT R	ST	SHIFT R	ST	SHIFT R	ST	SHIFT R		196/196/196/196	C4/ C4/ C4/ C4
SWAP	S	SHIFT W		keine		keine		keine		254 35 /---/---/---	FE 23/ --/ --/ --
SYS	S	SHIFT Y	S	SHIFT Y	S	SHIFT Y	S	SHIFT Y		158/158/158/158	9E/ 9E/ 9E/ 9E
TAB(	T	SHIFT A	T	SHIFT A	T	SHIFT A	T	SHIFT A		163/163/163/163	A3/ A3/ A3/ A3
TAN		keine		keine		keine		keine		192/192/192/192	C0/ C0/ C0/ C0
TEMPO	T	SHIFT E		keine		keine		keine		254 05 /---/---/---	FE 05/ --/ --/ --
THEN	T	SHIFT H	T	SHIFT H	T	SHIFT H	T	SHIFT H		167/167/167/167	A7/ A7/ A7/ A7
TI		keine		keine		keine		keine		84 73	54 49
TI$		keine		keine		keine		keine		84 73 36	54 49 24
TRAP	T	SHIFT R		keine	T	SHIFT R		keine		214/---/214/---	D6/ --/ D6/ --
TROFF	TRO	SHIFT F		keine	TRO	SHIFT F		keine		217/---/217/---	D9/ --/ D9/ --
TRON	TR	SHIFT O		keine	TR	SHIFT O		keine		216/---/216/---	D8/ --/ D8/ --
TO		keine		keine		keine		keine		164/164/164/164	A4/ A4/ A4/ A4
UNTIL	U	SHIFT N		keine	U	SHIFT N		keine		252/---/252/---	FC/ --/ FC/ --
USING	?US	SHIFT I		keine	?US	SHIFT I		keine		251/---/251/---	FB/ --/ FB/ --
USR	U	SHIFT S	U	SHIFT S	U	SHIFT S	U	SHIFT S		183/183/183/183	B7/ B7/ B7/ B7
VAL	V	SHIFT A	V	SHIFT A	V	SHIFT A	V	SHIFT A		197/197/197/197	C5/ C5/ C5/ C5
VERIFY	V	SHIFT E	V	SHIFT E	V	SHIFT E	V	SHIFT E		149/149/149/149	95/ 95/ 95/ 95
VOL	V	SHIFT O		keine	V	SHIFT O		keine		219/---/219/---	DB/ --/ DB/ --
WAIT	W	SHIFT A	W	SHIFT A	W	SHIFT A	W	SHIFT A		146/146/146/146	92/ 92/ 92/ 92
WHILE	W	SHIFT H		keine	W	SHIFT H		keine		253/---/253/---	FD/ --/ FD/ --
WIDTH	WI	SHIFT D		keine		keine		keine		254 28 /---/---/---	FE 1C/ --/ --/ --
WINDOW	W	SHIFT I		keine		keine		keine		254 26 /---/---/---	FE 1A/ --/ --/ --
XOR	X	SHIFT O		keine		keine		keine		206 08 /---/---/---	CE 08/ --/ --/ --

Mit unserer Tabelle (Seite 49–51) können Sie nun schnellstmöglich BASIC-Befehle eingeben.

+	+	+	+	+	170/170/170/170	AA/ AA/ AA/ AA
-	-	-	-	-	171/171/171/171	AB/ AB/ AB/ AB
*	*	*	*	*	172/172/172/172	AC/ AC/ AC/ AC
/	/	/	/	/	173/173/173/173	AD/ AD/ AD/ AD
^	^	^	^	^	174/174/174/174	AE/ AE/ AE/ AE
>	>	>	>	>	177/177/177/177	B1/ B1/ B1/ B1
=	=	=	=	=	178/178/178/178	B2/ B2/ B2/ B2
<	<	<	<	<	179/179/179/179	B3/ B3/ B3/ B3

TABELLE 2

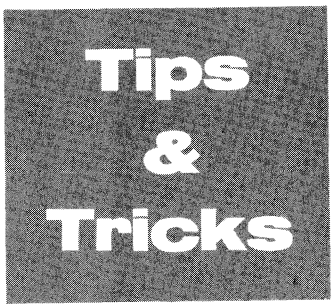
BASIC-Schlüsselwörtern und deren TOKEN und VORTOKEN in Dezimal und Hexadezimal
bei Commodore Computern CBM600/CBM700/CBM2000/CBM3000/CBM4000/CBM8032/VC20/C64/C16/C116/Plus4/128PC
S = C64 + SIMON'S BASIC - E1/E2/E3 = Computer + EXBASIC LEVEL II - EXP = C64 + Expander-Modul

DEZ	HEX	CBM 2.0 VC20 C64 CBM 4.0 CBM 600 CBM 700 C16/P4 128PC	CBM 4.0 CBM 600 CBM 700	CBM 600 CBM 700	C16/P4 128PC	128PC	CBM 2.0 VC-20 E1 C64 E2 C64S CBM 4.0E3	C64+Exp
01	01	-	-	-	-	-	OFF	-
02	02	-	-	-	-	-	RENUM	-
03	03	-	-	-	-	-	FIND	-
04	04	-	-	-	-	-	DEL	-
05	05	-	-	-	-	-	AUTO	-
06	06	-	-	-	-	-	DUMP	-
07	07	-	-	-	-	-	VPLOT	-
08	08	-	-	-	-	-	CALL	-
09	09	-	-	-	-	-	BORDER/REK-	12/3
10	0a	-	-	-	-	-	EXEC	-
11	0b	-	-	-	-	-	MERGE	-
12	0c	-	-	-	-	-	HPLOT	-
13	0d	-	-	-	-	-	DOKE	-
14	0e	-	-	-	-	-	SPACE	-
15	0f	-	-	-	-	-	INPUTLINE	-
16	10	-	-	-	-	-	SET	-
17	11	-	-	-	-	-	KEY/INSTLINE	12/3
18	12	-	-	-	-	-	CURSOR/DELLINE	12/3
19	13	-	-	-	-	-	ADSR/BEGINLINE	12/3
20	14	-	-	-	-	-	PAUSE/ENDLINE	12/3
21	15	-	-	-	-	-	PLAY/UP	- 12/3
22	16	-	-	-	-	-	VOLUME/DOWN	12/3
100-01	64-01	-	-	-	-	-	- HIRES	-
100-02	64-02	-	-	-	-	-	- PLOT	-
100-03	64-03	-	-	-	-	-	- LINE	-
100-04	64-04	-	-	-	-	-	- BLOCK	-
100-05	64-05	-	-	-	-	-	- FCHR	-
100-06	64-06	-	-	-	-	-	- FCOL	-
100-07	64-07	-	-	-	-	-	- FILL	-
100-08	64-08	-	-	-	-	-	- REC	-
100-09	64-09	-	-	-	-	-	- ROT	-
100-10	64-0a	-	-	-	-	-	- DRAW	-
100-11	64-0b	-	-	-	-	-	- CHAR	-
100-13	64-0d	-	-	-	-	-	- INV	-
100-14	64-0e	-	-	-	-	-	- FRAC	-
100-15	64-0f	-	-	-	-	-	- MOVE	-
100-16	64-10	-	-	-	-	-	- PLACE	-
100-25	64-19	-	-	-	-	-	- MULTI	-
100-27	64-1b	-	-	-	-	-	- MMOB	-
100-28	64-1c	-	-	-	-	-	- BFLASH	-
100-29	64-1d	-	-	-	-	-	- MOB SET	-
100-30	64-1e	-	-	-	-	-	- MUSIC	-
100-31	64-1f	-	-	-	-	-	- FLASH	-

Wenn Sie nun planen, BASIC-Programme von CBM 2000 – CBM 3000 – CBM 4000 – CBM 8000 – CBM 600 – CBM 700 – VC20 – C64 – SX64 – C16 – C116 – Plus4 – 128PC – plus verschiedene BASIC-Erweiterungen an Ihrer Rechnerkonfiguration, wie sie auch immer heißen mag, lauffähig zu bekommen, müssen Sie einige Dinge beachten: Inkompatibilität auf der Hardware-Seite kann natürlich nicht von BASIC gelöst werden, aber durch Weglassen oder Hinzufügen können rechnerspezifische Unterschiede kaschiert werden. Ein PET-BASIC-Programm, das keine Farbbefehle besitzt, kann auf dem C64 nachcoloriert oder ein C64-BASIC-Programm entcoloriert auf den PET umgeschrieben werden; dies ist einfach. Aber andere Dinge bereiten da größere Probleme: SPRITE-Befehle auf einem Computer, der keine SPRITE-Befehle hat, SID-SOUND-Befehle für Computer ohne den SOUND INTERFACE DEVICE-Chip, hochauflösende Grafik in einem Computer ohne Bit-mapped Display oder falsche PEEK- und POKE-Adressen (Bildschirm-, Color-CODES) beziehungsweise spezielle SYS-Befehle, schicken den Computer meist ins Nirwana. Hinzu kommen dann noch die Probleme der Bildschirm- und Zeichen-Ausgabe: Unterschiedliche 22 – 40 – 80 Zeichen pro Zeile, verschiedene Charakterzeichen ergeben andere Resultate. Auch die Geschwindigkeit der Commodore-Rechner differiert erheblich, der absolut schnellste ist der VC20, man höre und staune, die langsamsten sind Plus4 und 128PC. Die CBM 600/-700-Rechner sind dagegen so schnell, daß man sie bei bestimmten Problemlösungen verlangsamen muß.
Mit diesen TOKEN-Tabellen wollen wir Ihnen

dabei helfen, BASIC-Programme an Ihren Rechnertyp anzupassen. Jetzt können wir nur noch viel Spaß bei der Umsetzung von Programmen auf Ihr Rechner-System wünschen und viel Geduld bei der Suche der entsprechenden Rechnerroutinen und speziellen Adressen.

Robert Wagner □



Der C64 als Alarm-Anlage

Der C64/C128 hat zwei Joystick-Ports mit je fünf Schaltern. Dies ergibt die Möglichkeit, zehn Ein-/Aus-Schalter zu kontrollieren. Die Pin-Belegung ist wie folgt:

Pin-Belegung:

1 2 3 4 5
6 7 8 9

1: oben
2: rechts
3: unten
4: links
6: Feuerknopf
7: +5V
8: Masse

Die wichtigen Adressen Zum Auslesen der beiden Joystick-Ports sind: j1=56321 und j2=56320. Wird nun zum Beispiel Pin 1 an Port 1 (oben) mit Pin 7 (Masse) verbunden, so ergibt „PRINT PEEK(J1)" die Zahl 1. Im Fachhandel gibt es konfektionierte Kabel mit entsprechendem Stekker. Sicherheitshalber sollte der Anschluß für

100-32	64-20	-	-	-	-	-	-	REPEAT	-	-	-
100-33	64-21	-	-	-	-	-	-	PLAY	-	-	-
100-35	64-23	-	-	-	-	-	-	CENTRE	-	-	-
100-36	64-24	-	-	-	-	-	-	ENVELOPE	-	-	-
100-37	64-25	-	-	-	-	-	-	CGOTO	-	-	-
100-38	64-26	-	-	-	-	-	-	WAVE	-	-	-
100-39	64-27	-	-	-	-	-	-	FETCH	-	-	-
100-41	64-29	-	-	-	-	-	-	UNTIL	-	-	-
100-44	64-2c	-	-	-	-	-	-	USE	-	-	-
100-48	64-30	-	-	-	-	-	-	RESET	-	-	-
100-49	64-31	-	-	-	-	-	-	PROC	-	-	-
100-50	64-32	-	-	-	-	-	-	CALL	-	-	-
100-51	64-33	-	-	-	-	-	-	EXEC	-	-	-
100-52	64-34	-	-	-	-	-	-	END PROC	-	-	-
100-54	64-36	-	-	-	-	-	-	END	-	-	-
100-58	64-3a	-	-	-	-	-	-	LOOP	-	-	-
100-59	64-3b	-	-	-	-	-	-	DELAY	-	-	-
100-64	64-40	-	-	-	-	-	-	SECURE	-	-	-
100-65	64-41	-	-	-	-	-	-	DISAPPA:::-		-	:
100-66	64-42	-	-	-	-	-	-	CIRCLE	-	-	-
100-67	64-43	-	-	-	-	-	-	ON ERROR	-	-	-
100-68	64-44	-	-	-	-	-	-	NO ERROR	-	-	-
100-69	64-45	-	-	-	-	-	-	LOCAL	-	-	-
100-70	64-46	-	-	-	-	-	-	RCOMP	-	-	-
100-71	64-47	-	-	-	-	-	-	ELSE	-	-	-
100-72	64-48	-	-	-	-	-	-	RETRACE	-	-	-
100-73	64-49	-	-	-	-	-	-	TRACE	-	-	-
100-74	64-4a	-	-	-	-	-	-	DIR	-	-	-
100-75	64-4b	-	-	-	-	-	-	PAGE	-	-	-
100-76	64-4c	-	-	-	-	-	-	DUMP	-	-	-
100-77	64-4d	-	-	-	-	-	-	FIND	-	-	-
100-78	64-4e	-	-	-	-	-	-	OPTION	-	-	-
100-79	64-4f	-	-	-	-	-	-	AUTO	-	-	-
100-80	64-50	-	-	-	-	-	-	OLD	-	-	-
100-81	64-51	-	-	-	-	-	-	JOY	-	-	-
100-82	64-52	-	-	-	-	-	-	MOD	-	-	-
100-83	64-53	-	-	-	-	-	-	DIV	-	-	-
100-85	64-55	-	-	-	-	-	-	DUP	-	-	-
100-87	64-57	-	-	-	-	-	-	INST	-	-	-
100-88	64-58	-	-	-	-	-	-	TEST	-	-	-
100-89	64-59	-	-	-	-	-	-	LIN	-	-	-
100-90	64-5a	-	-	-	-	-	-	EXOR	-	-	-
100-91	64-5b	-	-	-	-	-	-	INSERT	-	-	-
100-92	64-5c	-	-	-	-	-	-	POT	-	-	-
100-93	64-5d	-	-	-	-	-	-	PENX	-	-	-
100-95	64-5f	-	-	-	-	-	-	PENY	-	-	-
100-98	64-62	-	-	-	-	-	-	DESIGN	-	-	-
100-99	64-63	-	-	-	-	-	-	RLOCMOB	-	-	-
100-100	64-64	-	-	-	-	-	-	CMOB	-	-	-
100-104	64-68	-	-	-	-	-	-	MOB OFF	-	-	-
100-105	64-69	-	-	-	-	-	-	OFF	-	-	-
100-106	64-6a	-	-	-	-	-	-	ANGL	-	-	-
100-107	64-6b	-	-	-	-	-	-	ARC	-	-	-
100-108	64-6c	-	-	-	-	-	-	COLD	-	-	-
100-109	64-6d	-	-	-	-	-	-	SCRSV	-	-	-
100-110	64-6e	-	-	-	-	-	-	SCRLD	-	-	-
100-111	64-6f	-	-	-	-	-	-	TEXT	-	-	-
100-112	64-70	-	-	-	-	-	-	CSET	-	-	-
100-113	64-71	-	-	-	-	-	-	VOL	-	-	-
100-114	64-72	-	-	-	-	-	-	DISK	-	-	-
100-116	64-74	-	-	-	-	-	-	KEY	-	-	-
100-117	64-75	-	-	-	-	-	-	PAINT	-	-	-
100-118	64-76	-	-	-	-	-	-	LOW COL	-	-	-
100-119	64-77	-	-	-	-	-	-	COPY	-	-	-
100-120	64-78	-	-	-	-	-	-	MERGE	-	-	-
100-121	64-79	-	-	-	-	-	-	RENUMBER	-	-	-
100-122	64-7a	-	-	-	-	-	-	MEM	-	-	-
100-123	64-7b	-	-	-	-	-	-	DETECT	-	-	-
100-124	64-7c	-	-	-	-	-	-	CHECK	-	-	-
100-125	64-7d	-	-	-	-	-	-	DISPLAY	-	-	-
100-126	64-7e 64 59	-	-	-	-	-	-	ERRLN	-	-	-
100-126	64-7e 4e	-	-	-	-	-	-	ERRN	-	-	-
100-127	64-7f	-	-	-	-	-	-	OUT	-	-	-

128	80	END	END	END	END	END	END	END	END	-	-
129	81	FOR	FOR	FOR	FOR	FOR	FOR	FOR	FOR	-	-
130	82	NEXT	NEXT	NEXT	NEXT	NEXT	NEXT	NEXT	NEXT	-	-
131	83	DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA	-	-
132	84	INPUT#	INPUT#	INPUT#	INPUT#	INPUT#	INPUT#	INPUT#	INPUT#	-	-
133	85	INPUT	INPUT	INPUT	INPUT	INPUT	INPUT	INPUT	INPUT	-	-
134	86	DIM	DIM	DIM	DIM	DIM	DIM	DIM	DIM	-	-
135	87	READ	READ	READ	READ	READ	READ	READ	READ	-	-
136	88	LET	LET	LET	LET	LET	LET	LET	LET	-	-
137	89	GOTO	GOTO	GOTO	GOTO	GOTO	GOTO	GOTO	GOTO	-	-
138	8a	RUN	RUN	RUN	RUN	RUN	RUN	RUN	RUN	-	-
139	8b	IF	IF	IF	IF	IF	IF	IF	IF	-	-
140	8c	RESTORE	RESTORE	RESTORE	RESTORE	RESTORE	RESTORE	RESTORE	RESTORE	-	-
141	8d	GOSUB	GOSUB	GOSUB	GOSUB	GOSUB	GOSUB	GOSUB	GOSUB	-	-
142	8e	RETURN	RETURN	RETURN	RETURN	RETURN	RETURN	RETURN	RETURN	-	-
143	8f	REM	REM	REM	REM	REM	REM	REM	REM	-	-
144	90	STOP	STOP	STOP	STOP	STOP	STOP	STOP	STOP	-	-
145	91	ON	ON	ON	ON	ON	ON	ON	ON	-	-
146	92	WAIT	WAIT	WAIT	WAIT	WAIT	WAIT	WAIT	WAIT	-	-
147	93	LOAD	LOAD	LOAD	LOAD	LOAD	LOAD	LOAD	LOAD	-	-
148	94	SAVE	SAVE	SAVE	SAVE	SAVE	SAVE	SAVE	SAVE	-	-
149	95	VERIFY	VERIFY	VERIFY	VERIFY	VERIFY	VERIFY	VERIFY	VERIFY	-	-
150	96	DEF	DEF	DEF	DEF	DEF	DEF	DEF	DEF	-	-
151	97	POKE	POKE	POKE	POKE	POKE	POKE	POKE	POKE	-	-
152	98	PRINT#	PRINT#	PRINT#	PRINT#	PRINT#	PRINT#	PRINT#	PRINT#	-	-
153	99	PRINT	PRINT	PRINT	PRINT	PRINT	PRINT	PRINT	PRINT	-	-
154	9a	CONT	CONT	CONT	CONT	CONT	CONT	CONT	CONT	-	-
155	9b	LIST	LIST	LIST	LIST	LIST	LIST	LIST	LIST	-	-
156	9c	CLR	CLR	CLR	CLR	CLR	CLR	CLR	CLR	-	-
157	9d	CMD	CMD	CMD	CMD	CMD	CMD	CMD	CMD	-	-
158	9e	SYS	SYS	SYS	SYS	SYS	SYS	SYS	SYS	-	-
159	9f	OPEN	OPEN	OPEN	OPEN	OPEN	OPEN	OPEN	OPEN	-	-
160	a0	CLOSE	CLOSE	CLOSE	CLOSE	CLOSE	CLOSE	CLOSE	CLOSE	-	-
161	a1	GET	GET	GET	GET	GET	GET	GET	GET	-	-
162	a2	NEW	NEW	NEW	NEW	NEW	NEW	NEW	NEW	-	-
163	a3	TAB(	TAB(	TAB(	TAB(	TAB(	TAB(	TAB(	TAB(	-	-
164	a4	TO	TO	TO	TO	TO	TO	TO	TO	-	-
165	a5	FN	FN	FN	FN	FN	FN	FN	FN	-	-
166	a6	SPC(	SPC(	SPC(	SPC(	SPC(	SPC(	SPC(	SPC(	-	-
167	a7	THEN	THEN	THEN	THEN	THEN	THEN	THEN	THEN	-	-
168	a8	NOT	NOT	NOT	NOT	NOT	NOT	NOT	NOT	-	-
169	a9	STEP	STEP	STEP	STEP	STEP	STEP	STEP	STEP	-	-
170	aa	+	+	+	+	+	+	+	+	-	-
171	ab	-	-	-	-	-	-	-	-	-	-
172	ac	*	*	*	*	*	*	*	*	-	-
173	ad	/	/	/	/	/	/	/	/	-	-
174	ae	^	^	^	^	^	^	^	^	-	-
175	af	AND	AND	AND	AND	AND	AND	AND	AND	-	-
176	b0	OR	OR	OR	OR	OR	OR	OR	OR	-	-
177	b1	>	>	>	>	>	>	>	>	-	-
178	b2	=	=	=	=	=	=	=	=	-	-
179	b3	<	<	<	<	<	<	<	<	-	-
180	b4	SGN	SGN	SGN	SGN	SGN	SGN	SGN	SGN	-	-
181	b5	INT	INT	INT	INT	INT	INT	INT	INT	-	-
182	b6	ABS	ABS	ABS	ABS	ABS	ABS	ABS	ABS	-	-
183	b7	USR	USR	USR	USR	USR	USR	USR	USR	-	-
184	b8	FRE	FRE	FRE	FRE	FRE	FRE	FRE	FRE	-	-
185	b9	POS	POS	POS	POS	POS	POS	POS	POS	-	-
186	ba	SQR	SQR	SQR	SQR	SQR	SQR	SQR	SQR	-	-
187	bb	RND	RND	RND	RND	RND	RND	RND	RND	-	-
188	bc	LOG	LOG	LOG	LOG	LOG	LOG	LOG	LOG	-	-
189	bd	EXP	EXP	EXP	EXP	EXP	EXP	EXP	EXP	-	-
190	be	COS	COS	COS	COS	COS	COS	COS	COS	-	-
191	bf	SIN	SIN	SIN	SIN	SIN	SIN	SIN	SIN	-	-
192	c0	TAN	TAN	TAN	TAN	TAN	TAN	TAN	TAN	-	-
193	c1	ATN	ATN	ATN	ATN	ATN	ATN	ATN	ATN	-	-
194	c2	PEEK	PEEK	PEEK	PEEK	PEEK	PEEK	PEEK	PEEK	-	-
195	c3	LEN	LEN	LEN	LEN	LEN	LEN	LEN	LEN	-	-
196	c4	STR	STR	STR	STR	STR	STR	STR	STR	-	-
197	c5	VAL	VAL	VAL	VAL	VAL	VAL	VAL	VAL	-	-
198	c6	ASC	ASC	ASC	ASC	ASC	ASC	ASC	ASC	-	-
199	c7	CHR$	CHR$	CHR$	CHR$	CHR$	CHR$	CHR$	CHR$	-	-

+5V isoliert werden (falls er überhaupt verbunden ist), damit kein Kurzschluß entsteht. Die anderen Anschlüsse sind harmlos. Wenn das Geschäft keine passenden Kabel vorrätig hat, kann auch ein Joystick-Adapter verwendet werden, der normalerweise benötigt wird, um Joysticks mit 9poliger Sub-D-Buchse (C64...) an den C16/Plus4 anzuschließen. Hier sind in der Regel nur die sechs für den Joystick notwendigen Kabel angeschlossen. Ein einfacher Test kann mit einem Miniprogramm durchgeführt werden:

```
0 print peek(j1),peek(j2)
  :goto
```

Wird nach dem Starten dieser Zeile eines der Kabel von 1 bis 4 oder 6 mit Pin 7 verbunden, ändert sich der ausgedruckte Wert. Übrigens springt ein GOTO ohne Zeilennummer zu Zeile 0.

PORT 1 UND 2 ALS KONTROLLSTATION

Die Verbindung der entsprechenden Pins mit Masse (Pin 8) kann auf vielfältige Weise geschehen. Die einfachste Methode wird im Joystick angewandt. Hier sind einfach kleine Schalter oder Metallkontakte, die die einzelnen Pins auf Masse legen. Genausogut kann man auch sogenannte Reed-Relais verwenden. Diese werden bevorzugt in Alarmanlagen eingesetzt, um das Öffnen und Schließen von Fenstern und Türen zu kontrollieren. Ein Reed-Relais besitzt zwei metallische Kontakte, die sich schließen, wenn ein Magnet in die Nähe gebracht wird und sich öffnen, wenn er wieder entfernt wird. Diese Schaltkontakte sind sehr klein und preiswert zu erhalten. Genauso kann mit einem Quecksilber-Neigungsschalter das Anheben oder Erschüttern ei-

nes Gegenstandes kontrolliert werden. In gleicher Weise kann eine Matte mit Trittkontakten eingesetzt werden. Soll die Temperatur eines Gerätes überwacht werden, so kann dafür ein Temperaturschalter verwendet werden. Dieser schaltet bei einer bestimmten Temperatur (zum Beispiel 100 Grad) und man kann dies mit dem Joystickport leicht überwachen. In der gleichen Weise kann mit einer Reflexlichtschranke oder einer Gabellichtschranke das Vorhandensein oder Fehlen eines Metall- beziehungsweise Papierbandes überwacht werden.

LICHTSCHRANKE EINFACH GEBAUT

Mit Hilfe einer Taschenlampenbirne oder einer infraroten Leuchtdiode und einem Fototransistor kann eine einfache Lichtschranke gebaut werden. Dazu eignet sich zum Beispiel als Sender der Typ LD261 und als Empfänger BPX81. Der Kontakt ist geschlossen, wenn der Fototransistor beleuchtet wird und offen, wenn Dunkelheit herrscht. Für eine solche Anwendung haben wir das Beispielprogramm „SCHALTER" geschrieben.
Eine typische Anwendung wäre ein Physikversuch mit der sogenannten schiefen Ebene. Entlang der Fahrbahn eines kleinen Modellautos befinden sich zehn Fototransistoren, die mit den entsprechenden Pins an den Ports 1 und 2 (mit gemeinsamer Masse) verbunden werden. Auf dem Wagen befindet sich eine kleine Lampe, die die Transistoren beim Vorbeifahren schaltet. Die Schalterstellung wird angezeigt. Die Zeitdifferenz zwischen den Schaltern wird in Zeile 90 bestimmt. Da es nur zehn Werte sind, braucht keine DIM-An-

200	c8	LEFT$	LEFT$	LEFT$	LEFT$	LEFT$	LEFT$	LEFT$	LEFT$	-	-
201	c9	RIGHT$	RIGHT$	RIGHT$	RIGHT$	RIGHT$	RIGHT$	RIGHT$	RIGHT$	-	-
202	ca	MID$	MID$	MID$	MID$	MID$	MID$	MID$	MID$	-	-
203	cb	GO	GO	GO	GO	GO	GO	GO	GO	-	-
204	cc	-	CONCAT	CONCAT	RGR	RGR	CONCAT 3	-	-	-	-
205	cd	-	DOPEN	DOPEN	RCLR	RCLR	DOPEN 3	-	-	-	-
206	ce	-	DCLOSE	DCLOSE	RLUM	-	DCLOSE 3	-	-	-	-
206-02	ce-02	-	-	-	-	POT	-	-	-	-	-
206-03	ce-03	-	-	-	-	BUMP	-	-	-	-	-
206-04	ce-04	-	-	-	-	PEN	-	-	-	-	-
206-05	ce-05	-	-	-	-	RSPPOS	-	-	-	-	-
206-06	ce-06	-	-	-	-	RSPRITE	-	-	-	-	-
206-07	ce-07	-	-	-	-	RSPCOLOR	-	-	-	-	-
206-08	ce-08	-	-	-	-	XOR	-	-	-	-	-
206-09	ce-09	-	-	-	-	RWINDOW	-	-	-	-	-
206-10	ce-0a	-	-	-	-	POINTER	-	-	-	-	-
207	cf	-	RECORD	RECORD	JOY	JOY	RECORD 3	-	-	-	-
208	d0	-	HEADER	HEADER	RDOT	RDOT	HEADER 3	-	-	-	-
209	d1	-	COLLECT	COLLECT	DEC	DEC	COLLECT 3	-	-	-	-
210	d2	-	BACKUP	BACKUP	HEX$	HEX$	BACKUP 3	-	-	-	-
211	d3	-	COPY	COPY	ERR$	ERR$	COPY 3	-	-	-	-
212	d4	-	APPEND	APPEND	INSTR	INSTR	APPEND 3	-	-	-	-
213	d5	-	DSAVE	DSAVE	ELSE	ELSE	DSAVE 3	-	-	-	-
214	d6	-	DLOAD	DLOAD	RESUME	RESUME	DLOAD 3	-	-	-	-
215	d7	-	CATALOG	CATALOG	TRAP	TRAP	CATALOG 3	-	-	-	-
216	d8	-	RENAME	RENAME	TRON	TRON	RENAME 3	-	-	-	-
217	d9	-	SCRATCH	SCRATCH	TROFF	TROFF	SCRATCH 3	-	-	-	-
218	da	-	DIRECTORY	DIRECTORY	SOUND	SOUND	DIR 3	-	-	-	-
219	db	-	-	DCLEAR	VOL	VOL	-	-	-	-	-
220	dc	-	-	BANK	AUTO	AUTO	RESET	-	-	-	-
221	dd	-	-	BLOAD	PUDEF	PUDEF	MEM	-	-	-	-
222	de	-	-	BSAVE	GRAPHIC	GRAPHIC	TRACE	-	-	-	-
223	df	-	-	KEY	PAINT	PAINT	BASIC	-	-	-	-
224	e0	-	-	DELETE	CHAR	CHAR	RESUME	-	-	-	-
225	e1	-	-	ELSE	BOX	BOX	LETTER	-	-	-	-
226	e2	-	-	TRAP	CIRCLE	CIRCLE	HELP	-	-	-	-
227	e3	-	-	RESUME	GSHAPE	GSHAPE	COKE/BEEP	12/3	-	-	-
228	e4	-	-	DISPOSE	SSHAPE	SSHAPE	GROUND/FAST	12/3	-	-	-
229	e5	-	-	PUDEF	DRAW	DRAW	MATRIX	-	-	-	-
230	e6	-	-	USING	LOCATE	LOCATE	DISPOS	-	-	-	-
231	e7	-	-	ERR$	COLOR	COLOR	PRINT @	-	-	-	-
232	e8	-	-	INSTR	SCNCLR	SCNCLR	HIMEM	-	-	-	-
233	e9	-	-	-	SCALE	SCALE	HARDCOPY	-	-	-	-
234	ea	-	-	-	HELP	HELP	INPUTFORM	-	-	-	-
235	eb	-	-	-	DO	DO	LOCK/MOD/SCREEN	12/3	-	-	-
236	ec	-	-	-	LOOP	LOOP	SWAP	-	-	-	-
237	ed	-	-	-	EXIT	EXIT	USING	-	-	-	-
238	ee	-	-	-	DIRECTORY	DIRECTORY	SEC	-	-	-	-
239	ef	-	-	-	DSAVE	DSAVE	ELSE	-	-	-	-
240	f0	-	-	-	DLOAD	DLOAD	ERROR	-	-	-	-
241	f1	-	-	-	HEADER	HEADER	ROUND	-	-	-	-
242	f2	-	-	-	SCRATCH	SCRATCH	DEEK	-	-	-	-
243	f3	-	-	-	COLLECT	COLLECT	STRING$	-	-	-	-
244	f4	-	-	-	COPY	COPY	POINT	-	-	-	-
245	f5	-	-	-	RENAME	RENAME	INSTR	-	-	-	-
246	f6	-	-	-	BACKUP	BACKUP	CEEK/RND	12/3	-	-	-
247	f7	-	-	-	DELETE	DELETE	MIN	-	-	-	-
248	f8	-	-	-	RENUMBER	RENUMBER	MAX	-	-	-	-
249	f9	-	-	-	KEY	KEY	VARPTR	-	-	-	-
250	fa	-	-	-	MONITOR	MONITOR	FRAC	-	-	-	-
251	fb	-	-	-	USING	USING	ODD	-	-	-	-
252	fc	-	-	-	UNTIL	UNTIL	DEC	-	-	-	-
253	fd	-	-	-	WHILE	WHILE	HEX$	-	-	-	-
254	fe	-	-	-	-	-	EVAL	-	-	-	-
254-02	fe-02	-	-	-	-	BANK	-	-	-	-	-
254-03	fe-03	-	-	-	-	FILTER	-	-	-	-	-
254-04	fe-04	-	-	-	-	PLAY	-	-	-	-	-
254-05	fe-05	-	-	-	-	TEMPO	-	-	-	-	-
254-06	fe-06	-	-	-	-	MOVSPR	-	-	-	-	-
254-07	fe-07	-	-	-	-	SPRITE	-	-	-	-	-
254-08	fe-08	-	-	-	-	SPRCOLOR	-	-	-	-	-
254-09	fe-09	-	-	-	-	RREG	-	-	-	-	-

254-10	fe-0a	-	-	-	-	ENVELOPE	-	-	-	-	-
254-11	fe-0b	-	-	-	-	SLEEP	-	-	-	-	-
254-12	fe-0c	-	-	-	-	CATALOG	-	-	-	-	-
254-13	fe-0d	-	-	-	-	DOPEN	-	-	-	-	-
254-14	fe-0e	-	-	-	-	APPEND	-	-	-	-	-
254-15	fe-0f	-	-	-	-	DCLOSE	-	-	-	-	-
254-16	fe-10	-	-	-	-	BSAVE	-	-	-	-	-
254-17	fe-11	-	-	-	-	BLOAD	-	-	-	-	-
254-18	fe-12	-	-	-	-	RECORD	-	-	-	-	-
254-19	fe-13	-	-	-	-	CONCAT	-	-	-	-	-
254-20	fe-14	-	-	-	-	DVERIFY	-	-	-	-	-
254-21	fe-15	-	-	-	-	DCLEAR	-	-	-	-	-
254-22	fe-16	-	-	-	-	SPRSAVE	-	-	-	-	-
254-23	fe-17	-	-	-	-	COLLISION	-	-	-	-	-
254-24	fe-18	-	-	-	-	BEGIN	-	-	-	-	-
254-25	fe-19	-	-	-	-	BEND	-	-	-	-	-
254-26	fe-1a	-	-	-	-	WINDOW	-	-	-	-	-
254-27	fe-1b	-	-	-	-	BOOT	-	-	-	-	-
254-28	fe-1c	-	-	-	-	WIDTH	-	-	-	-	-
254-29	fe-1d	-	-	-	-	SPRDEF	-	-	-	-	-
254-30	fe-1e	-	-	-	-	QUIT	-	-	-	-	-
254-31	fe-1f	-	-	-	-	STASH	-	-	-	-	-
254-32	fe-20	-	-	-	-	-	-	-	-	-	-
254-33	fe-21	-	-	-	-	FETCH	-	-	-	-	-
254-34	fe-22	-	-	-	-	-	-	-	-	-	-
254-35	fe-23	-	-	-	-	SWAP	-	-	-	-	-
254-36	fe-24	-	-	-	-	OFF	-	-	-	-	-
254-37	fe-25	-	-	-	-	FAST	-	-	-	-	-
254-38	fe-26	-	-	-	-	SLOW	-	-	-	-	-
254-128	fe-80	-	-	-	-	-	-	-	KEY	-	-
254-129	fe-81	-	-	-	-	-	-	-	COLOR	-	-
254-130	fe-82	-	-	-	-	-	-	-	GRAPHIC	-	-
254-131	fe-83	-	-	-	-	-	-	-	SCNCLR	-	-
254-132	fe-84	-	-	-	-	-	-	-	LOCATE	-	-
254-133	fe-85	-	-	-	-	-	-	-	SCALE	-	-
254-134	fe-86	-	-	-	-	-	-	-	BOX	-	-
254-135	fe-87	-	-	-	-	-	-	-	CIRCLE	-	-
254-136	fe-88	-	-	-	-	-	-	-	CHAR	-	-
254-137	fe-89	-	-	-	-	-	-	-	DRAW	-	-
254-138	fe-8a	-	-	-	-	-	-	-	GSHAPE	-	-
254-139	fe-8b	-	-	-	-	-	-	-	PAINT	-	-
254-140	fe-8c	-	-	-	-	-	-	-	SSHAPE	-	-
254-141	fe-8d	-	-	-	-	-	-	-	TUNE	-	-
254-142	fe-8e	-	-	-	-	-	-	-	FILTER	-	-
254-143	fe-8f	-	-	-	-	-	-	-	SPRDEF	-	-
254-144	fe-90	-	-	-	-	-	-	-	TEMPO	-	-
254-145	fe-91	-	-	-	-	-	-	-	MOVSPR	-	-
254-146	fe-92	-	-	-	-	-	-	-	SPRCOL	-	-
254-147	fe-93	-	-	-	-	-	-	-	SPRITE	-	-
254-148	fe-94	-	-	-	-	-	-	-	COLINT	-	-
254-149	fe-95	-	-	-	-	-	-	-	SPRSAV	-	-
254-150	fe-96	-	-	-	-	-	-	-	RBUMP	-	-
254-151	fe-97	-	-	-	-	-	-	-	RCLR	-	-
254-152	fe-98	-	-	-	-	-	-	-	RDOT	-	-
254-153	fe-99	-	-	-	-	-	-	-	RGR	-	-
254-154	fe-9a	-	-	-	-	-	-	-	RJOY	-	-
254-155	fe-9b	-	-	-	-	-	-	-	RPEN	-	-
254-156	fe-9c	-	-	-	-	-	-	-	RPOT	-	-
254-157	fe-9d	-	-	-	-	-	-	-	RSPCOL	-	-
254-158	fe-9e	-	-	-	-	-	-	-	RSPPOS	-	-
255	ff	pi	pi	pi	pi	pi	pi	pi	pi	-	-

weisung gemacht zu werden.
Mit Hilfe von IF-Abfragen und der AND-Funktion kann jede Schalterstellung erfragt werden:

```
100 b=peek(j1):rem port1
110 if band1 then print "oben"
120 if band2 then print "unten"
130 if band4 then print "links"
140 if band8 then print "rechts"
150 goto100
```

Wenn der C64/C128 ein Ereignis an einem der Joystickports feststellt, muß dieses Ergebnis in der Regel zu einem Alarmsignal verarbeitet werden. Die einfachste Möglichkeit besteht in der Verwendung der SOUND-

AUSLÖSEN DES ALARMS

POKEs (siehe Handbuch) und der entsprechenden Bildschirmanzeige. Jedoch kann meist der Fernseher nicht die ganze Zeit eingeschaltet bleiben. Daher hier ein einfacher Tip, wie ein Alarm ausgelöst werden kann. Am Kassettenport stehen 5V zur Verfügung, mit denen normalerweise der Recordermotor betrieben wird. Wird hier ein Piezosummer (verbraucht sehr wenig Strom) angeschlossen, so kann durch Einschalten des Kassettenrecorders dieser Summer betätigt werden. Genauso kann ein Relais gesteuert werden, das eine Sirene in Betrieb setzt. Natürlich können mit Hilfe des User-Ports auch kompliziertere Steuerungen erfolgen.

NOCH EIN KLEINER TIP

Die Joystick-Ports benutzen teilweise die gleichen Leitungen wie die Tastatur. Daher stehen oft bei Benutzung des Joysticks Zeichen im Tastaturpuffer. Für manche Anwendungen ist es besser, die Tastatur zu sperren:

Port normal
1 POKE56323,0
2 POKE56322,255
Port sperren
1 POKE56323,224
2 POKE56322,224

Wir hoffen, daß Sie durch diese Beispiele angeregt wurden, Ihren C64 oder C128 zu Meßzwecken einzusetzen. Wir würden uns freuen, wenn Sie uns Ihre Anwendungen schreiben würden. □

Zeicheneditor

Oftmals möchte man für seine eigenen Programme einen ganz individuellen Zeichensatz oder eigene Sonderzeichen entwerfen. Dieser komfortable Zeichensatz-Editor soll dabei helfen.

BESCHREIBUNG DES PROGRAMMS:

Nach dem Starten des Programms sieht man den Editor, der aus vier verschiedenen Teilen besteht:
- das Editierfeld,
- der Kopierspeicher,
- der Puffer,
- die Anzeige des gesamten Zeichensatzes.

Um nun ein bestimmtes Zeichen zu verändern oder ein vollkommen neues Zeichen zu entwerfen, pickt man sich mit Hilfe des grauen Cursors im unteren Zeichensatz-Anzeigefeld das zu verändernde Zeichen heraus. Die Steuerung dieses Cursors wird mit folgenden Tasten vollzogen:

'+'	für 1 Zeichen nach rechts.
'SHIFT & +'	für 10 Zeichen nach rechts.
'–'	für 1 Zeichen nach links.
'SHIFT & –'	für 10 Zeichen nach links.

Für das eigentliche Editieren sind folgende Funktionen gedacht:

'SPACE'	um einen Punkt zu setzen, beziehungsweise zu löschen.
'Cursor-Tasten	um den grünen Cursor im Editierfeld zu bewegen.
'CLR/HOME'	um das jeweilige Zeichen zu löschen.

Sonderfunktionen:

'I'	invertiert das aktuelle Zeichen.
'R'	verschiebt das jeweilige Zeichen nach links.
'SHIFT & R'	verschiebt das jeweilige Zeichen nach rechts.
'W'	rotiert nach oben.
'A'	rotiert nach links.
'S'	rotiert nach rechts.
'Z'	rotiert nach unten.
'L'	löscht einige Zeilen des jeweiligen Zeichens, was einen schönen Effekt erziehlt.
'SHIFT & L'	ein weiterer Linien-Effekt.
'O'	bewirkt ein UNDO oder auch OLD, das heißt, die Veränderung des aktuellen Zeichens wird wieder zurückgenommen. Das Zeichen nimmt dadurch seine usprüngliche Gestalt wieder an.

Kopierfunktionen:

'C'	Zeichen in Kopierspeicher übernehmen.
'SHIFT & C'	Zeichen aus Kopier-Speicher in Editierfeld übernehmen.

Diese Kopierfunktionen erlauben es zum Beispiel, daß sich ein Zeichen gemerkt wird, und an einer beliebigen Stelle im Zeichensatz wieder abgelegt werden kann.

Pufferfunktionen:

Die Pufferfunktionen erlauben die Erstellung großer Zeichen aus mehreren einzelnen Zeichen.

':'	bewirkt Puffer-Cursor nach links.
';'	bewirkt Puffer-Cursor nach rechts.
'B'	Zeichen in Puffer übernehmen.
'SHIFT & B'	Zeichen aus Puffer in Editierfeld.

Mit diesen Funktionen kann man nun zum Beispiel eine Ecke eines großen Buchstabens editieren, mit den Puffer-Cursor-Tasten positionieren, und dann mit 'B' in den Puffer übernehmen. So überblickt man immer das gesamte große Zeichen und kann die einzelnen Zeichen nacheinander im Zeichensatz ablegen.

Funktionstasten:

'F1/F2'	Rahmenfarbe erhöhen/erniedrigen.
'F3/F4'	Hintergrundfarbe erhöhen/erniedrigen.
'F5/F6'	Zeichenfarbe erhöhen/erniedrigen.
'F7'	Disketten-Operationen. Mit den Disketten-Operationen kann man seinen entworfenen Zeichensatz abspeichern beziehungsweise einen schon vorhandenen einladen. Die Directory Funktion lädt das Inhaltsverzeichnis der Diskette und kann mit 'SPACE' angehalten werden. Mit 'Menu' kommt man wieder zum Editor.
'F8'	Mit dieser Funktionstaste wird der ganz normale Zeichensatz geholt.
'Q'	verläßt den Zeichen-Editor. □

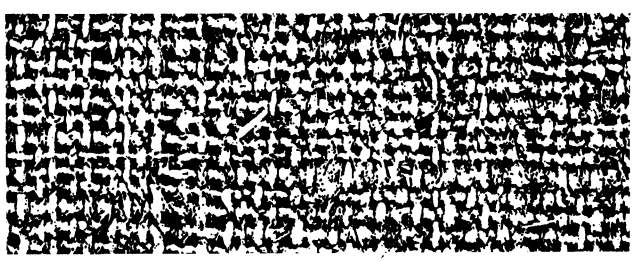
```
10 rem ------------------------64  <bf>
20 rem (p) commodore welt       ==  <hf>
30 rem --------------------------  <mm>
40 rem (c) by                   ==  <pp>
50 rem     marc sachse          ==  <ji>
60 rem                          ==  <nd>
70 rem version 2.0   40z./ascii==  <le>
80 rem  c-64  floppy            ==  <ja>
90 rem --------------------------  <km>
95 gosub 60000                     <jp>
100 open2,8,2,"zeichen-editor,p,w" <ab>
110 print#2,chr$(1)chr$(8);        <ci>
120 printc1$"zeichensatzeditor wir
d auf diskette                     <je>
130 print"gespeichert.             <fl>
140 printc4$"zaehler (1-2873) :    <ap>
150 printc4$                       <mp>
160 fora=1to2873:readb:print#2,chr
$(b);:printc2$a:nexta              <kg>
170 close2:printc1$"zeichensatzedi
tor gespeichert                    <gp>
180 data31,8,195,7,158,50,48,56    <nf>
190 data49,32,32,32,90,69,73,67    <cd>
200 data72,69,78,69,68,73,164,82   <jh>
210 data32,86,49,46,48,0,0,0       <ok>
220 data120,169,51,133,1,162,0,189 <mh>
230 data118,8,157,0,112,189,0,208  <gp>
240 data157,0,32,157,0,48,232,208  <cc>
250 data238,238,42,8,238,45,8,238  <pb>
260 data48,8,238,51,8,238,54,8     <kb>
270 data173,54,8,201,64,208,214,16
9                                  <ke>
280 data55,133,1,88,162,0,189,50   <jd>
290 data19,157,224,48,232,224,8,20
8                                  <ge>
300 data245,169,11,157,0,216,157,0 <ok>
310 data217,157,0,218,157,232,218,
232                                <nc>
320 data208,241,76,232,115,26,5,9  <pn>
330 data3,8,5,14,5,4,9,20          <lo>
340 data15,18,32,22,49,32,32,32    <cp>
350 data32,85,67,67,67,67,67,67    <ad>
360 data67,67,73,32,32,32,32,32    <op>
370 data32,32,32,32,32,32,32,32    <eg>
380 data32,32,32,32,32,32,32,32    <fo>
390 data32,32,32,32,32,32,32,32    <nh>
400 data32,66,32,32,32,32,32,32    <gd>
410 data32,32,66,32,32,32,32,32    <pg>
420 data32,32,32,32,32,32,32,32    <ee>
430 data32,32,32,32,32,32,32,32    <ip>
440 data32,32,32,32,32,32,32,32    <fc>
450 data32,66,32,32,32,32,32,32    <gi>
460 data32,32,66,32,32,32,32,32    <ki>
470 data32,32,32,32,32,26,5,9      <lm>
480 data3,8,5,14,32,32,32,58       <mh>
490 data32,0,32,32,32,32,32,32     <oc>
500 data32,66,32,32,32,32,32,32    <dg>
510 data32,32,66,32,32,32,32,32    <fp>
520 data32,32,32,32,32,32,32,32    <jk>
530 data32,32,32,32,32,32,32,32    <fh>
540 data32,32,32,32,32,32,32,32    <if>
550 data32,66,32,32,32,32,32,32    <mj>
560 data32,32,66,32,32,32,32,32    <pf>
570 data32,32,32,32,32,11,15,16    <fl>
580 data9,5,18,5,14,32,32,58       <on>
590 data32,31,32,32,32,32,32,32    <ea>
600 data32,66,32,32,32,32,32,32    <ag>
610 data32,32,66,32,32,32,32,32    <op>
620 data32,32,32,32,32,32,32,32    <ep>
630 data32,32,32,32,32,32,32,32    <ef>
640 data32,32,32,32,32,32,32,32    <ne>
650 data32,66,32,32,32,32,32,32    <ng>
660 data32,32,66,32,32,32,32,32    <pp>
670 data32,32,32,32,32,32,32,32    <pg>
680 data32,32,32,32,32,32,32,32    <jm>
690 data32,32,32,32,32,32,32,32    <hh>
700 data32,66,32,32,32,32,32,32    <hn>
710 data32,32,66,32,32,32,32,32    <mm>
720 data32,32,32,32,32,32,32,32    <oh>
730 data32,32,32,32,32,32,32,32    <lo>
740 data32,32,32,32,32,32,32,32    <hn>
750 data32,66,32,32,32,32,32,32    <hl>
760 data32,32,66,32,32,32,32,32    <kj>
770 data32,32,32,32,32,16,21,6     <of>
780 data6,5,18,32,32,32,32,32      <ei>
790 data32,32,32,32,32,32,32,32    <dc>
800 data32,74,67,67,67,67,67,67    <mi>
810 data67,67,75,32,32,32,32,32    <hi>
820 data32,32,32,32,32,32,32,32    <ah>
830 data32,32,32,32,32,32,32,32    <fk>
840 data32,32,32,32,32,32,32,32    <bn>
850 data32,32,32,32,32,32,32,32    <bh>
860 data32,32,32,32,32,32,32,32    <do>
870 data32,32,32,32,32,192,193,194 <dc>
880 data195,32,32,32,32,32,32,32   <ab>
890 data32,32,32,32,32,32,32,32    <fo>
900 data32,32,32,32,32,32,32,32    <nh>
910 data32,32,32,32,32,32,32,32    <nb>
920 data32,32,32,32,32,196,197,198 <in>
930 data199,32,32,32,32,32,32,32   <fb>
940 data32,32,32,32,32,32,32,32    <ip>
950 data32,32,32,32,32,32,32,32    <fc>
960 data32,32,32,32,32,32,32,32    <nk>
970 data32,32,32,32,32,200,201,202 <ea>
980 data203,32,32,32,32,32,32,32   <di>
990 data32,32,32,32,32,32,32,32    <ll>
1000 data32,32,32,32,32,32,32,32   <mk>
1010 data32,32,32,32,32,32,32,32   <lp>
1020 data32,32,32,32,32,204,205,20
6                                  <dl>
1030 data207,32,32,32,32,32,32,32  <ba>
1040 data32,32,32,32,32,32,32,32   <fh>
1050 data32,32,32,32,32,32,32,32   <if>
1060 data32,32,32,32,32,32,32,32   <gl>
1070 data32,32,32,32,32,32,32,32   <ag>
```

```
1080 data32,32,32,32,32,32,32,32   <ka>
1090 data32,32,32,32,32,32,32,32   <gh>
1100 data32,32,32,32,32,32,32,32   <cn>
1110 data32,32,32,32,32,32,32,32   <fd>
1120 data32,32,32,32,32,28,32,2    <oe>
1130 data25,32,13,19,32,32,32,32   <ma>
1140 data32,32,32,32,32,32,32,32   <ef>
1150 data32,32,32,32,32,32,32,32   <ne>
1160 data32,32,32,32,32,32,32,32   <gp>
1170 data32,32,32,32,32,32,32,32   <mi>
1180 data32,32,32,32,32,32,32,32   <pg>
1190 data32,32,32,32,32,32,32,32   <jm>
1200 data32,32,32,32,32,32,32,32   <hh>
1210 data32,32,32,32,32,32,32,32   <oa>
1220 data32,32,32,32,32,0,1,2      <lj>
1230 data3,4,5,6,7,8,9,10          <ib>
1240 data11,12,13,14,15,16,17,18   <bd>
1250 data19,20,21,22,23,24,25,26   <fk>
1260 data27,28,29,30,31,32,33,34   <nb>
1270 data35,36,37,38,39,40,41,42   <nj>
1280 data43,44,45,46,47,48,49,50   <ln>
1290 data51,52,53,54,55,56,57,58   <ih>
1300 data59,60,61,62,63,64,65,66   <kc>
1310 data67,68,69,70,71,72,73,74   <ib>
1320 data75,76,77,78,79,80,81,82   <mk>
1330 data83,84,85,86,87,88,89,90   <ka>
1340 data91,92,93,94,95,96,97,98   <pn>
1350 data99,100,101,102,103,104,10
5,106                              <db>
1360 data107,108,109,110,111,112,1
13,114                             <ne>
1370 data115,116,117,118,119,120,1
21,122                             <md>
1380 data123,124,125,126,127,128,1
29,130                             <do>
1390 data131,132,133,134,135,136,1
37,138                             <lp>
1400 data139,140,141,142,143,144,1
45,146                             <aa>
1410 data147,148,149,150,151,152,1
53,154                             <bc>
1420 data155,156,157,158,159,160,1
61,162                             <ml>
1430 data163,164,165,166,167,168,1
69,170                             <ml>
1440 data171,172,173,174,175,176,1
77,178                             <bg>
1450 data179,180,181,182,183,184,1
85,186                             <nl>
1460 data187,188,189,190,191,192,1
93,194                             <ba>
1470 data195,196,197,198,199,200,2
01,202                             <hl>
1480 data203,204,205,206,207,208,2
09,210                             <cm>
1490 data211,212,213,214,215,216,2
17,218                             <nk>
1500 data219,220,221,222,223,224,2
25,226                             <ci>
1510 data227,228,229,230,231,232,2
33,234                             <km>
1520 data235,236,237,238,239,240,2
41,242                             <fm>
1530 data243,244,245,246,247,248,2
49,250                             <lp>
1540 data251,252,253,254,255,32,32
,32                                <cd>
1550 data32,32,32,32,32,32,32,32   <fh>
1560 data32,32,32,32,32,32,32,32   <if>
1570 data32,32,32,32,32,162,0,169  <ja>
1580 data0,157,248,48,232,224,8,20
8                                  <me>
1590 data246,162,0,169,255,157,0,5
4                                  <nm>
1600 data232,224,128,208,246,76,21
1,116                              <pj>
1610 data0,128,64,32,16,8,4,2      <bn>
1620 data1,127,191,223,239,247,251
,253                               <ol>
1630 data254,61,4,101,4,141,4,181  <gj>
1640 data4,221,4,5,5,45,5,85       <lj>
1650 data5,0,0,36,17,17,163,163    <gg>
1660 data163,163,163,163,163,163,1
63,163                             <ce>
1670 data163,163,163,163,163,163,1
3,145                              <de>
1680 data145,0,0,0,0,0,0,0         <fn>
1690 data0,0,0,0,0,0,0,0           <mo>
1700 data0,0,147,49,46,76,79,65    <hh>
1710 data68,13,50,46,83,65,86,69   <df>
1720 data13,51,46,68,73,82,69,67   <nj>
1730 data84,79,82,89,13,52,46,77   <mm>
1740 data69,78,85,13,13,13,0,255   <pp>
1750 data192,0,128,64,0,128,64,0   <hg>
1760 data128,64,0,128,64,0,128,64  <ja>
1770 data0,128,64,0,128,64,0,128   <pn>
1780 data64,0,255,192,0,0,0,0      <po>
1790 data0,0,0,0,0,0,0,0           <ff>
1800 data0,0,0,0,0,0,0,0           <ma>
1810 data0,0,0,0,0,0,0,0           <ge>
1820 data0,0,0,0,0,0,0,184         <ib>
1830 data217,185,217,186,217,187,2
17,224                             <mo>
1840 data217,225,217,226,217,227,2
17,8                               <lk>
1850 data218,9,218,10,218,11,218,4
8                                  <hm>
1860 data218,49,218,50,218,51,218,
0                                  <fp>
1870 data169,0,141,32,208,141,33,2
08                                 <ek>
1880 data169,255,141,138,2,169,11,
141                                <co>
1890 data134,2,32,65,117,162,0,189 <fg>
1900 data114,116,157,64,3,232,224,
64                                 <dk>
```

```
1910 data208,245,169,1,141,21,208,
169                                 <mo>
1920 data13,141,39,208,169,13,141,
248                                 <go>
1930 data7,162,0,189,0,112,157,0    <al>
1940 data4,189,0,113,157,0,5,189    <og>
1950 data0,114,157,0,6,189,232,114  <em>
1960 data157,232,6,169,11,157,0,21
6                                   <fc>
1970 data157,0,217,157,208,217,234
,234                                <la>
1980 data234,232,208,215,32,173,11
7,32                                <hp>
1990 data40,118,32,96,118,32,71,11
8                                   <ch>
2000 data32,251,119,76,53,117,120,
169                                 <ld>
2010 data124,141,20,3,169,117,141,
21                                  <ld>
2020 data3,169,129,141,26,208,173,
17                                  <ao>
2030 data208,41,127,141,17,208,169
,0                                  <cg>
2040 data141,18,208,88,96,120,169,
49                                  <ai>
2050 data141,20,3,169,234,141,21,3  <fm>
2060 data169,240,141,26,208,88,169
,29                                 <ig>
2070 data141,24,208,169,0,141,21,2
08                                  <ia>
2080 data96,173,25,208,141,25,208,
41                                  <il>
2090 data128,208,3,76,126,234,173,
18                                  <pa>
2100 data208,201,192,176,16,169,29
,141                                <le>
2110 data24,208,169,192,141,18,208
,32                                 <ca>
2120 data135,234,76,129,234,169,24
,141                                <jn>
2130 data24,208,169,0,141,18,208,7
6                                   <kc>
2140 data129,234,169,0,141,209,117
,173                                <db>
2150 data3,116,10,46,209,117,10,46  <be>
2160 data209,117,10,46,209,117,141
,208                                <ej>
2170 data117,173,209,117,24,105,32
,141                                <cb>
2180 data209,117,162,0,189,0,240,1
57                                  <eb>
2190 data0,48,232,224,8,208,245,96  <in>
2200 data169,0,141,2,118,173,3,116  <ob>
2210 data10,46,2,118,10,46,2,118    <ll>
2220 data10,46,2,118,141,1,118,173  <gp>
2230 data2,118,24,105,32,141,2,118  <bn>
2240 data162,0,189,0,48,157,0,240   <fh>
2250 data232,224,8,208,245,96,174,
3                                   <ma>
2260 data116,169,11,157,208,218,17
3,210                               <id>
2270 data116,10,170,189,178,116,14
1,37                                <cp>
2280 data118,189,179,116,141,38,11
8,169                               <eg>
2290 data11,141,0,240,96,174,3,116  <od>
2300 data169,15,157,208,218,173,21
0,116                               <ko>
2310 data10,170,189,178,116,141,68
,118                                <bj>
2320 data189,179,116,141,69,118,16
9,15                                <lk>
2330 data141,0,240,96,173,37,116,1
0                                   <if>
2340 data10,10,24,105,191,141,0,20
8                                   <af>
2350 data173,36,116,10,10,10,24,10
5                                   <kp>
2360 data57,141,1,208,96,162,0,138  <bj>
2370 data10,168,185,20,116,141,130
,118                                <em>
2380 data185,21,116,141,131,118,16
0,0                                 <mc>
2390 data189,0,48,57,4,116,240,4    <kc>
2400 data169,160,208,2,169,32,153,
0                                   <aj>
2410 data240,200,192,8,208,234,232
,224                                <fi>
2420 data8,208,212,96,104,104,32,9
6                                   <gl>
2430 data117,162,0,189,77,116,240,
6                                   <np>
2440 data32,210,255,232,208,245,32
,240                                <le>
2450 data119,201,49,240,15,201,50,
240                                 <oj>
2460 data39,201,51,240,63,201,52,2
08                                  <gc>
2470 data237,76,211,116,162,0,189,
80                                  <hc>
2480 data116,32,210,255,232,224,4,
208                                 <fe>
2490 data245,169,13,32,210,255,32,
93                                  <ie>
2500 data119,224,0,240,196,76,167,
119                                 <jk>
2510 data162,0,189,87,116,32,210,2
55                                  <ph>
2520 data232,224,4,208,245,169,13,
32                                  <if>
2530 data210,255,32,93,119,224,0,2
40                                  <oc>
2540 data168,76,203,119,169,1,160,
116                                 <kh>
2550 data162,38,32,189,255,169,1,1
62                                  <de>
```

```
2560 data8,160,0,32,186,255,32,192 <lp>
2570 data255,162,1,32,198,255,160,
5                                  <jn>
2580 data132,251,32,207,255,170,16
4,251                              <ce>
2590 data136,208,245,173,1,220,201
,239                               <il>
2600 data240,249,169,0,133,198,32,
207                                <pa>
2610 data255,164,144,208,35,32,205
,189                               <ia>
2620 data169,32,32,210,255,32,207,
255                                <di>
2630 data133,251,201,0,240,8,165,2
51                                 <fb>
2640 data32,210,255,76,48,119,169,
13                                 <ae>
2650 data32,210,255,160,3,76,11,11
9                                  <eb>
2660 data32,204,255,169,1,32,195,2
55                                 <na>
2670 data32,240,119,201,32,208,249
,76                                <do>
2680 data148,118,162,0,189,39,116,
32                                 <mp>
2690 data210,255,232,224,21,208,24
5,162                              <fk>
2700 data0,169,175,32,210,255,169,
157                                <kf>
2710 data32,210,255,32,240,119,201
,13                                <nf>
2720 data240,38,201,20,208,21,224,
0                                  <aj>
2730 data240,241,169,32,32,210,255
,169                               <ho>
2740 data157,32,210,255,32,210,255
,202                               <ca>
2750 data76,108,119,224,16,240,210
,157                               <la>
2760 data60,116,32,210,255,232,208
,201                               <km>
2770 data142,76,116,96,162,8,160,0 <ge>
2780 data32,186,255,160,116,162,60
,173                               <kp>
2790 data76,116,32,189,255,169,0,1
60                                 <ho>
2800 data32,170,32,213,255,134,174
,134                               <id>
2810 data45,132,175,132,46,76,148,
118                                <cc>
2820 data162,8,160,1,32,186,255,17
3                                  <og>
2830 data76,116,160,116,162,60,32,
189                                <mh>
2840 data255,162,0,160,32,134,251,
132                                <gb>
2850 data252,169,251,162,0,160,40,
32                                 <hp>
2860 data216,255,76,148,118,164,19
8,136                              <do>
2870 data48,251,132,198,185,119,2,
96                                 <ng>
2880 data32,240,119,201,32,208,18,
174                                <ba>
2890 data36,116,172,37,116,189,0,4
8                                  <ep>
2900 data89,4,116,157,0,48,76,96   <ki>
2910 data118,201,17,208,16,238,36,
116                                <aa>
2920 data173,36,116,201,8,208,5,16
9                                  <hb>
2930 data0,141,36,116,96,201,145,2
08                                 <pn>
2940 data16,206,36,116,173,36,116,
201                                <dl>
2950 data255,208,5,169,7,141,36,11
6                                  <cf>
2960 data96,201,29,208,16,238,37,1
16                                 <kp>
2970 data173,37,116,201,8,208,5,16
9                                  <fe>
2980 data0,141,37,116,96,201,157,2
08                                 <ld>
2990 data16,206,37,116,173,37,116,
201                                <ki>
3000 data255,208,5,169,7,141,37,11
6                                  <nc>
3010 data96,201,64,240,4,201,45,20
8                                  <of>
3020 data15,32,9,118,32,219,117,20
6                                  <aj>
3030 data3,116,32,40,118,76,173,11
7                                  <gb>
3040 data201,42,240,4,201,43,208,1
5                                  <cc>
3050 data32,9,118,32,219,117,238,3 <no>
3060 data116,32,40,118,76,173,117,
201                                <ak>
3070 data73,208,16,162,0,189,0,48  <ib>
3080 data73,255,157,0,48,232,224,8 <ne>
3090 data208,243,96,201,79,208,3,7
6                                  <di>
3100 data173,117,201,147,208,13,16
2,0                                <kk>
3110 data169,0,157,0,48,232,224,8  <ej>
3120 data208,248,96,201,87,208,20,
172                                <he>
3130 data0,48,162,0,189,1,48,157   <an>
3140 data0,48,232,224,7,208,245,14
0                                  <ha>
3150 data7,48,96,201,90,208,20,172 <ei>
3160 data7,48,162,6,189,0,48,157   <hc>
3170 data1,48,202,224,255,208,245,
140                                <ml>
3180 data0,48,96,201,65,208,15,162 <la>
3190 data0,189,0,48,42,62,0,48     <ac>
```

```
3200 data232,224,8,208,244,96,201,
83                                  <mh>
3210 data208,15,162,0,189,0,48,106 <lk>
3220 data126,0,48,232,224,8,208,24
4                                   <ok>
3230 data96,201,67,208,13,162,0,18
9                                   <ek>
3240 data0,48,157,248,48,232,224,8 <na>
3250 data208,245,201,195,208,19,16
2,0                                 <bl>
3260 data189,248,48,157,0,48,169,0 <jc>
3270 data157,248,48,232,224,8,208,
240                                 <pc>
3280 data96,201,192,240,4,201,219,
208                                 <ef>
3290 data18,32,9,118,173,3,116,24  <aa>
3300 data105,10,141,3,116,32,40,11
8                                   <pc>
3310 data76,173,117,201,186,240,4,
201                                 <jf>
3320 data221,208,18,32,9,118,173,3 <og>
3330 data116,56,233,10,141,3,116,3
2                                   <ko>
3340 data40,118,76,173,117,201,82,
208                                 <hf>
3350 data29,162,0,189,0,48,42,62    <ki>
3360 data0,48,232,224,2,208,244,16
2                                   <hj>
3370 data5,189,0,48,106,126,0,48    <ag>
3380 data232,224,8,208,244,96,201,
210                                 <eb>
3390 data208,29,162,0,189,0,48,106 <ca>
3400 data126,0,48,232,224,2,208,24
4                                   <pc>
3410 data162,5,189,0,48,42,62,0     <dg>
3420 data48,232,224,8,208,244,96,2
01                                  <id>
3430 data76,208,14,162,0,169,0,157 <la>
3440 data1,48,232,232,224,8,208,24
5                                   <hb>
3450 data96,201,81,208,3,76,226,25
2                                   <df>
3460 data201,133,208,4,238,32,208,
96                                  <hp>
3470 data201,134,208,4,238,33,208,
96                                  <fe>
3480 data201,135,208,8,238,31,117,
104                                 <ni>
3490 data104,76,211,116,201,137,20
8,4                                 <el>
3500 data206,32,208,96,201,138,208
,4                                  <ef>
3510 data206,33,208,96,201,139,208
,8                                  <mo>
3520 data206,31,117,104,104,76,211
,116                                <gk>
3530 data201,140,208,47,120,169,51
,133                                <bl>
3540 data1,162,0,189,0,208,157,0    <hj>
3550 data32,232,208,247,238,16,122
,238                                <ad>
3560 data19,122,173,19,122,201,40,
208                                 <ll>
3570 data234,169,32,141,19,122,169
,208                                <en>
3580 data141,16,122,169,55,133,1,8
8                                   <dp>
3590 data76,173,117,201,204,208,12
,169                                <om>
3600 data0,141,1,48,141,5,48,141    <ai>
3610 data7,48,96,201,136,208,3,76   <al>
3620 data143,118,201,59,208,19,32,
9                                   <mk>
3630 data118,174,210,116,232,224,1
6,208                               <fj>
3640 data2,162,0,142,210,116,76,40 <ck>
3650 data118,201,58,208,19,32,9,11
8                                   <ml>
3660 data174,210,116,202,224,255,2
08,2                                <jo>
3670 data162,15,142,210,116,76,40,
118                                 <fc>
3680 data201,66,208,28,173,210,116
,10                                 <fb>
3690 data10,10,141,147,122,169,54,
141                                 <mk>
3700 data148,122,162,0,189,0,48,15
7                                   <hg>
3710 data0,240,232,224,8,208,245,9
6                                   <nk>
3720 data201,194,208,28,173,210,11
6,10                                <om>
3730 data10,10,141,176,122,169,54,
141                                 <ni>
3740 data177,122,162,0,189,0,240,1
57                                  <jg>
3750 data0,48,232,224,8,208,245,96 <do>
3760 data96,60,66,153,161,161,153,
66                                  <pp>
3770 data60                         <om>
3780 end                            <cf>
60000 rem nachspann ==============
=                                   <da>
60010 rem * farbcodes/steuercodes
*                                   <kg>
60020 c4$=chr$(017):c2$=chr$(145)   <lc>
60030 c1$=chr$(147)                 <pn>
60040 return                        <hc>
```



NUTZEN SIE UNSEREN BEQUEMEN POSTSERVICE

128/C64-SPECIAL KOMMT JETZT DIREKT INS HAUS!

128 C64 SPECIAL

SECHSMAL IM JAHR FÜR 70 DM! SIE SPAREN ÜBER 20 PROZENT!

Finden Sie Ihre 128/C64-SPECIAL nicht immer am Kiosk? Vielleicht, weil schon ausverkauft? Möchten Sie 128/C64 schon vor der Kioskbelieferung in Händen haben? Dann gibt es jetzt eine Möglichkeit! Wir beliefern Sie im Abonnement mit sechs Ausgaben für ganze 70 DM (Inland) oder 80 DM (Ausland). Sie erhalten dann das jeweils druckfrische Heft, in der Regel früher, als es am Kiosk hängt (so die Bundespost will). Sechsmal.

WICHTIGE RECHTLICHE GARANTIE!
Sie können diesen Abo-Auftrag binnen einer Woche nach Eingang der Abo-Bestätigung durch den Verlag widerrufen – Postkarte genügt. Zur Wahrung der Frist genügt die rechtzeitige Absendung. Bitte bestätigen Sie durch Ihre zweite Unterschrift, daß Sie von diesem Widerspruchsrecht Kenntnis genommen haben. Ansonsten läuft dieser Auftrag jeweils für sechs Ausgaben, wenn ihm nicht vier Wochen vor Ablauf widersprochen wird, weiter.

ABO-SERVICE-KARTE

COUPON

Ich nehme zur Kenntnis, daß die Belieferung erst beginnt, wenn die Abo-Gebühr dem Verlag zugegangen ist.

Ja, ich möchte von Ihrem Angebot Gebrauch machen.
Bitte senden Sie mir bis auf Widerruf ab sofort jeweils die nächsten sechs Ausgaben an untenstehende Anschrift. Wenn ich nicht einen Monat vor Ablauf kündige, läuft diese Abmachung automatisch weiter.

Name _____
Vorname _____
Straße/Hausnr. _____
PLZ/Ort _____
Ich bezahle:
☐ per beiliegendem Verrechnungsscheck
☐ gegen Rechnung
☐ bargeldlos per Bankeinzug von meinem Konto
bei (Bank) und Ort _____
Kontonummer _____
Bankleitzahl _____
(steht auf jedem Kontoauszug)

Unterschrift _____
Von meinem Widerspruchsrecht habe ich Kenntnis genommen.

Unterschrift _____ 128/III

COMMODORE WELT
ABO-SERVICE 128/III
POSTFACH 1161
D-8044
UNTERSCHLEISSHEIM

PROGRAMMSERVICE
Achtung! Preis-Senkung!!!

Hiermit bestelle ich in Kenntnis Ihrer Verkaufsbedingungen die Listings dieses Heftes auf

☐ Diskette (30 DM)
Name _____
Vorname _____
Straße/Hausnr. _____
PLZ/Ort _____
Ich bezahle:
☐ per beiliegendem Verrechnungsscheck / Bargeld
☐ bargeldlos per Bankeinzug von meinem Konto (nur möglich in der Bundesrepublik!)
bei (Bank) und Ort _____
Kontonummer _____
Bankleitzahl _____
(steht auf jedem Kontoauszug)
Verkaufsbedingungen: Lieferung nur gegen Vorkasse oder Bankabbuchung. Keine Nachnahme.
Umtausch bei Nichtfunktionieren.

Unterschrift _____ 128/III
Bitte ausschneiden und einsenden an

COMMODORE WELT
KASSETTENSERVICE 128/III
POSTFACH 1161
D-8044 UNTERSCHLEISSHEIM

Jede Diskette nur noch 30 DM! Sie sparen 10 DM

LOAD&RUN

Das Spiele-Magazin

Hallo, liebe Leser,

Die Redaktion von Load & Run wurde mal wieder vom Fieber befallen. Kaum waren die Epidemien Shanghai und Tetris überstanden, machte sich eine neue Krankheit breit, der sich keiner der Redakteure erwehren konnte. Ich spreche vom neuesten Amiga-Game: ‚Ports of Call'. Dieser Handelsimulation ist für kurze Zeit sogar unser Verleger zum Opfer gefallen. Leider hat diese Krankheit aber auch einen Haken: Ist ein Spieler lange genug infiziert, so verflüchtigt sie sich wieder mit einem Schlag. Nach langem Spiel deckte unser Verleger einige sehr bedeutende Schwächen und Schlampereien in der Programmierung auf, die den Spielwitz erheblich senken oder gar völlig verschwinden lassen. So wurde durch schlampige Arbeit einmal mehr ein hervorragendes Spiel kaputtgemacht. Doch mehr dazu im ausführlichen Testbericht in dieser Ausgabe.

Ansonsten finden Sie in dieser Ausgabe wieder alles wie gewohnt vor. Ausführliche Spieletests berichten über die neuesten Games, Kurzberichte behandeln die Umsetzungen und Neuerscheinungen auf dem Spielemarkt, und die Player's Pages geben Hilfestellung für jeden verzweifelten Joystick-Akrobaten. Ich hoffe, es ist für jeden etwas dabei.

Viel Spaß beim Lesen wünscht Ihnen die Redaktion

Impressum

LOAD & RUN erscheint in der München Aktuell Verlags GmbH, Heßstraße 90, 8000 München 40.

Verantwortlich für den Inhalt:
Gert Seidel

Redaktion:
Thomas Bosch, Michael Nebauer.

Layout:
Sonja Anderle

Anzeigenverwaltung:
ADV-Mediendienste, Aindlingerstr. 8900 Augsburg, Tel. 0821/790 42 43

Inhalt

Darts auf dem Amiga

Jetzt brauchen auch die Amiga-Besitzer nicht länger zu trauern. Mastertronic bringt in diesen Tagen eine Darts-Simulation auf den Markt.

Beim Darts-Spiel müssen Sie durch geschicktes Pfeilwerfen, im Volksmund auch „Spikkern" genannt, Ihren Punkteabstand so schnell wie möglich auf Null bringen.
Bis zu zwei Spieler können an der Darts-Simulation von Mastertronic teilnehmen. Wenn man mit Werfen an der Reihe ist, erscheint auf dem Bildschirm die Darts-Scheibe und eine Hand, welche einen Pfeil hält und leider ständig hin und her wackelt (fast wie in der Realität). Mit dem Joystick müssen Sie die Hand in die richtige Position bringen und anschließend mit dem Feuerknopf den Pfeil werfen. Danach schreibt eine weitere Hand Ihren aktuellen Punktestand auf die nebenstehende Tafel. Sieger ist derjenige Spieler, der zuerst bei Null angelangt ist.
Grafisch ist World Darts sehr gut gemacht. Viele kleine Details wie zum Beispiel wechselnde Gesichtszüge einer passiven Spielfigur sorgen immer wieder für ein kleines Schmunzeln. Digitalisierte Soundeffekte lockern die Atmosphäre zusätzlich auf. Wer sich für den Darts-Sport begeistert und zudem noch einen Amiga mit mindestens 512 Kbyte RAM und einen Farbmonitor besitzt, kann sich World Darts ja mal ansehen. TB ■

Titel:	**Darts**
Getestet:	**Amiga**
Umsetzungen:	---

Im Test: **Preis (DM):** k.A.

- [x] Joystick
- [ ] Tastatur
- [ ] Maus

Wertung	0	25	50	75	100
Grafik					
Sound					
Bedienung					
Motivation					

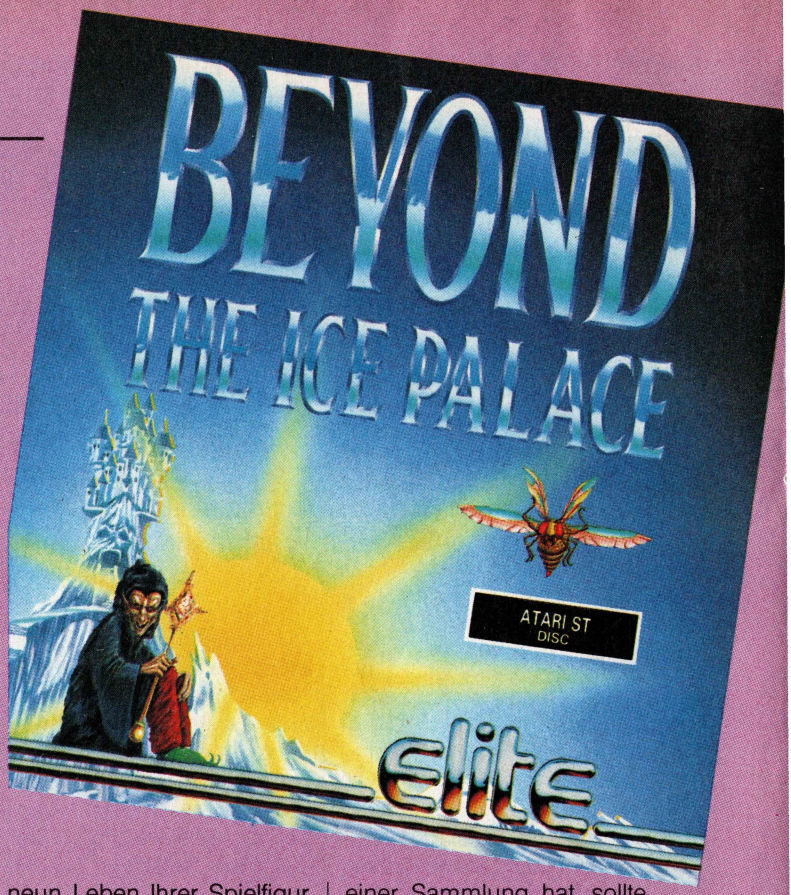
Rockford is still alive

Die Boulder-Dash-Reihe reißt nicht ab. Auch für den Amiga gibt es jetzt ein solches Game.

Eine hervorragende Adaption der beliebten Boulder-Dash-Reihe liegt jetzt auch für den Amiga vor. Das Spielprinzip ist dasselbe geblieben. Rockford besitzt aber im Gegensatz zu seinen Vorgängern viele neue Features, die das Spielgeschehen auflockern. So können Sie sich zu Beginn des Spieles eine von fünf Rockford-Figuren aussuchen. Je nach Spielfigur ändert sich der Aufbau des Spielfeldes. Haben Sie sich beispielsweise für Rockford, den Meisterkoch entschieden, finden sie sich in einem Chaos aus Bestecken, Tellern und Tischen wieder. Glauben Sie aber nicht, daß sich am Spiel etwas ändert, denn auch hier müssen Sie Objekte sammeln und dabei aufpassen, daß Sie nicht von herunterfallenden Suppenschüsseln zerschmettert werden.
Jedes Spielfeld ist in vier Level unterteilt, die mit steigender Nummer schwieriger werden. Unter anderem nimmt die zur Verfügung stehende Zeit immer schneller ab.
Rockford ist grafisch sehr gut gelungen. Auch über mangelnde Sounduntermalung kann sich der Spieler nicht beklagen. Das Spiel kann ich jedem empfehlen, der einen Amiga mit Farbmonitor besitzt und noch nicht genug von der Boulder-Dash-Welle hat. TB ■

Alte Idee - gutes Spiel

Elite, der Spezialist in Sachen Spielautomaten-Umsetzungen, hat wieder zugeschlagen. Ein gut aufgemachtes Ballerspiel für den ST ist das Ergebnis.



Lange Zeit war es still um den englischen Softwarehersteller Elite-Systems Limited, doch jetzt endlich gibt es ein neues Spiel, erhältlich für den C 64 und den ST. Die Spielidee kann man nicht mehr als alt bezeichnen, uralt trifft wohl eher zu: Ein unbekanntes Land wird von den Mächten des Bösen unterdrückt. Nach heftigen Diskussionen beschließt man, einen einsamen Helden, der zufällig über neun Leben verfügt, hinab in die Dungeons zu schicken, damit er den Zauberern und ihren Monstergehilfen ein für allemal den Garaus macht: Es darf geballert werden.

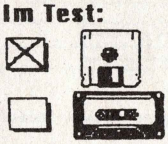
Trotz der etwas einfallslosen Spielidee dürfen Sie aber nicht glauben, daß auch das Game selbst verschrottet gehöre – im Gegenteil! Beyond The Ice Palace ist ein gutes bis sehr gutes Jump-and-run-Spiel mit hervorragender Grafik und zahlreichen Bildschirmen. Ein für ST-Verhältnisse sehr ordentliches Scrolling und gelungen animierte Sprites sorgen für einen flüssigen Spielablauf. Auch die passenden Soundeffekte lockern die Atmosphäre stark auf.

Auf dem Weg durch die Dungeons sind neben den Gegnern auch zahlreiche Hindernisse wie tiefe Schluchten oder steile Berghänge zu überwinden. Die neun Leben Ihrer Spielfigur sind wirklich nicht zu viel.

Beyond The Ice Palace ist für Freunde und Liebhaber der Jump-and-run-Spiele ein absolutes Muß. Wer noch nicht allzu viele davon in einer Sammlung hat, sollte sich das Game so schnell wie möglich zulegen. Die Atari-ST-Version benötigt übrigens ROM-TOS und einen Farbmonitor oder TV-Modulator.

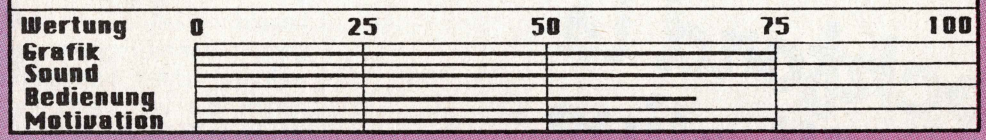
TB ■

Titel:	Beyond The Ice Palace
Getestet:	Atari ST
Umsetzungen:	---

Im Test: 

Preis (DM): 59,-

Joystick ☒
Tastatur ☐
Maus ☐



Feuer frei

Gute Nachrichten für alle Amiga-Besitzer: Wer vor einem Jahr noch mit neidischen Blicken auf Westbank für den CPC geschaut hat, darf dieses Game jetzt auch auf seinem Amiga spielen. Gunshoot ist da.

Axxion hat sich der Umsetzung von Westbank angenommen. Und weil die Amiga-Version noch ein paar Features mehr bot als die des CPC, hat man ihr gleich einen neuen Namen verpaßt: Gunshoot, was übersetzt etwa Revolverschuß heißt. Für Nicht-Kenner von Westbank hier kurz die Handlung:

Sie sind ein kleiner Bankangestellter und möchten gerne ein Held sein. Deswegen kommt Ihnen auch der Umstand, daß in letzter Zeit ziemlich viele Banken überfallen wurden, gerade recht. Mit einem 45er Revolver bewaffnet, warten Sie hinter Ihrem Schalter auf Banditen. Darum herum sind zwölf Türen, die Sie ständig im Auge behalten müssen. Von Zeit zu Zeit öffnen sich eine oder zwei und Personen betreten die Bank. Sind diese aber in der Absicht gekommen, ihr Vermögen anzulegen oder wollen sie gar die Bank um ein paar Tausender erleichtern?

Keine Chance für Bankräuber

Beobachten Sie die Leute genau. Es kann sehr schnell passieren, daß plötzlich einer seinen Colt zieht. Manche kommen sogar schon be-

waffnet herein. Sie müssen augenblicklich reagieren, wenn Sie einen Überfall vereiteln wollen. Dazu bewegen Sie mit dem Joystick ein Fadenkreuz auf die jeweilige Person. Mit dem Feuerknopf wird der Abzug betätigt. Haben Sie einen Räuber erwischt, gibt's Punkte en masse. Wenn Sie aber auf einen ehrlichen Kunden ballern, gibt's jede Menge Ärger mit dem Sheriff. Also genau hinsehen, bevor Sie abdrücken.

Gunshoot ist ein Game, das jedem zu empfehlen ist. Die ohnehin relativ hohe Spielmotivation wird durch hervorragende Grafik und digitalisierte Soundeffekte noch gesteigert. Und wenn's trotzdem mal zu langweilig werden sollte, bietet das Spiel noch einen speziellen Zwei-Player-Mode, in dem Spieler eins das Fadenkreuz steuert und sein Partner den Abzug betätigt. Gunshoot gibt's für alle Amigas ab 512 KByte RAM mit Farbmonitor. TB ■

Titel:	**Gunshoot**
Getestet:	**Amiga**
Umsetzungen:	---

Im Test: ☒ Diskette ☐ Kassette

Preis (DM): k.A.

☒ Joystick ☐ Tastatur ☐ Maus

Wertung	0	25	50	75	100
Grafik					
Sound					
Bedienung					
Motivation					

Vorsicht: Suchtgefahr!

Ein einfaches Puzzle-Spiel – das suggeriert die Aufmachung von Powerstyx. Doch hinter diesem Namen verbirgt sich sehr viel mehr.

In Powerstyx müssen Sie verschiedene Bilder freilegen, die zu Beginn noch unsichtbar sind. Mit einem Cursor, der nur mit dem Joystick gesteuert werden kann, schneiden Sie aus dem Bildschirm beliebig große Rechtecke heraus. Dadurch wird ein Teil des verdeckten Bildes sichtbar, der jeweils so groß ist wie das ausgeschnittene Rechteck. Sind mehr als 75 Prozent des Bildes freigelegt, gelangen Sie in den nächsten Level. Es gibt 15 Bilder zu entdecken, jedes repräsentiert einen Level.

Mit dem Cursor Rechtecke ausschneiden – das klingt doch wirklich nicht schwierig. Und tatsächlich weckt das Game zunächst keine große Begeisterung. Mißmutig bewege ich meinen Cursor über den Bildschirm. Hier ein Rechteck, da ein Rechteck. Doch was ist das? Aus der linken oberen Ecke tauchen plötzlich zwei Totenköpfe auf, die meinen Linien folgen. Da die beiden nicht gerade vertrauenerweckend aussehen, entschließe ich mich zur Flucht nach hinten. Doch diverse fliegende Buchstaben versperren mir den Weg. Nach wenigen Sekunden gebe ich den Kampf auf – und bin eines meiner fünf Leben los.

Der zweite Anlauf gelingt endlich. Nach knapp vier Minuten, die zu 80 Prozent aus Flucht vor diversen bösartigen Sprites bestehen, ist das erste Bild freigelegt oder besser gesagt: 75 Prozent davon. Das Bild wird komplett aufgebaut und ich kann mich knappe 30 Sekunden lang auf die wirklich einmalig schöne Grafik konzentrieren. Noch bevor ich mich richtig sattgesehen habe, ertönt ein akustisches Warnsignal und der Bildschirm wird wieder schwarz, was den nächsten Level ankündigt. Nun gut, ich habe den ersten in nur vier Minuten überstanden, da wirft mich der zweite auch nicht um – denkste! Die fliegenden Totenschädel haben Verstärkung bekommen und auch die restlichen Sprites scheinen wesentlich angriffslustiger. Nach drei Minuten muß ich kapitulieren, da meine Leben aufgebraucht sind. Ein neuer Start, ein neues Spiel.

Untermalt wird das Spektakel von diversen Hintergrundgeräuschen, die nicht sonderlich gut gelungen sind. Das fällt aber nicht allzu stark ins Gewicht, da sich der Spieler ständig auf das Spielgeschehen konzentrieren muß. Bis in den 15. Level habe ich es zwar nicht geschafft, aber die hervorragende Qualität der Grafiken dürfte sich wohl kaum verschlechtern. Übrigens sind die Bilder bei der C64-Version fast bessser als die der Amiga-Version. Auch läuft auf dem C64 alles ein klein wenig schneller ab. Aber keine Angst, liebe Amiga-Freaks, auch die Geschwindigkeit auf dem 16-Bit-Computer reicht doppelt und dreifach aus.

Powerstyx sollte in keiner Sammlung fehlen. Ich kann es nur wärmstens weiterempfehlen. Doch lassen Sie sich warnen: Wer es einmal gespielt hat, kommt so schnell nicht mehr davon los.

Thomas Bosch ■

Ports of Call

Stellen Sie sich vor, Sie bekommen fünf Millionen Dollar und wollen daraus 500 Millionen Dollar machen. Bei Ports of Call kann das leicht gelingen. Wer sich allerdings keine Mühe gibt, bei dem wird bald der Pleitegeier über der Reederei schweben.

Ports of Call ist ein Spiel, bei dem der Spieler die Rolle eines Reeders übernimmt. Mit fünf Millionen Dollar müssen nach dem Spielstart erst einmal Schiffe gekauft werden. Am billigsten sind Low-Coast-Ships, diese sehen allerdings auch so aus. Die verbeulten Kähne sollten erst einmal repariert werden. Der Zustand des Schiffes wird in jedem Hafen angegeben, die Tauglichkeit sollte niemals 60% unterschreiten, da sonst das Schiff von einer Rattenplage befallen werden und später sogar untergehen kann. Ist das Schiff also gerade noch schwimmfähig, kann die Ladung ausgewählt werden. Manche Güter können sehr viel Frachterträge bringen, doch bei der Auswahl ist auf sehr viele Dinge zu achten: Wie weit ist es vom Starthafen zum Zielhafen, wieviel Treibstoff wird verbraucht, lohnen die Frachten am Zielhafen, wie hoch ist die Gefahr, Eisberge oder Riffe umfahren zu müssen, sind in diesem Gebiet viele Stürme oder drohen Freibeuter den Tanker zu plündern ? Ist die richtige Entscheidung gefallen, kann es schon losgehen. Zum Ablegen können Schlepper zu Hilfe genommen werden, diese kosten allerdings auch Geld. Außerdem ist auf die Schlepperkapitäne auch kein Verlaß, hin und wieder streiken diese und da ein Reeder keine Zeit zu verlieren hat, heißt es, selbst im Hafen ab- bzw. anzulegen. Dies hört sich zwar sehr einfach an, muß aber erst einmal ausdauernd geübt werden. Ein kleiner Crash gegen die Hafenmauer kostet nämlich viel Zeit und Geld. Sobald das beladene Schiff sicher den Hafen verlassen hat, kann der Spieler auf der Weltkarte sein kleines Schiff entlangtuckern sehen. Unterwegs kann auch so manches Ereignis geschehen, so ist es manchmal nötig, einem Sturm auszuweichen oder Schiffbrüchige zu retten. Im Zielhafen heißt es, nach neuen ertragreichen Ladungen umsehen. Ist keine gewinnbringende Fracht in Sicht, kann ja mal das Schiff auf Vordermann gebracht werden. Nach ein paar Tagen, das Schiff glänzt nun wieder in der Abendsonne, muß dann Geld verdient werden. Deshalb sollten Häfen, in denen es keine ertragreichen Güter gibt, gemieden werden.

Dunkle Geschäfte können natürlich auch getätigt werden. Für einen Schrankkoffer, der irgendwo im Schiff versteckt wird, erhält der Reeder in den meisten Fällen 100.000 Dollar. An 50 Kisten Waffen hingegen kann schon einmal eine Millon Dollar verdient werden. Nur: Der Zoll findet diese Schmugglerei nicht so toll. In den meisten Fällen genügt aber ein saftiges Schmiergeld, um das betroffene Schiff wieder freizukaufen. Sind die ersten 20 Millionen Dollar erwirtschaftet, kann der Reeder ein High-Tech-Schiff mit Querstrahlruder anzahlen. Mit 40% Anzahlung kann sich fast jede Reederei dieses Superschiff leisten. Aber aufgepaßt, mehr als 40 Millionen Dollar sollte für so einen großen Pott nicht bezahlt werden. Natürlich hat jeder Reeder auch ein Büro, von hier aus können die Gewinne und die Verluste abgefragt werden, Hypotheken getilgt oder Kredite aufgenommen werden. Falls mal irgendwann die Geschäfte nicht mehr so gut laufen, kann mit dem Menüpunkt Registerwechsel der Heimathafen des Reeders gewechselt werden. Zu den ertragreichsten Frachtrouten gehören: Rotterdam – Kalkutta und Vancouver – Karachi.

Mittlerweile wird dieses Spiel so gepriesen, als ob es noch kein besseres Wirtschaftsspiel gegeben hätte. Anfangs macht es sicherlich unwahrscheinlich viel Spaß. Doch nach einigen Tagen, wenn der Spieler genug Geld erwirtschaftet hat, verschwindet die Spieldiskette wieder in der Versenkung. Denn die Fehler, die dieses Spiel hat, machen es nach einer gewissen Zeit total lächerlich.

Das fängt schon damit an, daß ein auf Reede liegendes Schiff in Rio de Janeiro von einem Eisberg gerammt werden kann. Es kann weiter mal passieren, daß einem Spieler unterwegs der Treibstoff ausgeht. Das betroffene Schiff muß dann natürlich abgeschleppt werden.

Doch wenn im Hafen wieder aufgetankt werden soll, ist der Dieselbunker noch fast voll. Dem Milliardär macht dieses lächerliche Milliönchen Abschleppkosten überhaupt nichts aus. Doch der kleine Reeder sieht mal wieder den Pleitegeier.

Oder folgender Fall tritt ein: Wegen mangelnder Aufträge müssen die Schiffe im Hafen festgemacht werden. Diese Schiffe rühren sich dann, für die angegebene Zeit, nicht mehr vom Fleck. Doch wenn dann ein anderes Schiff den „Kurs“ des eigenen Pottes kreuzt, welches man mittels einem Radar umfahren sollte, muß man sich schon an den Kopf greifen. Das gleiche Desaster haben die armen Freibeuter, diese greifen meist erst an, wenn das Schiff leer im Hafen liegt oder sie es bereits ausgeraubt hatten. Wenn beim Einlaufen mal wieder die Schlepperkapitäne streiken sollten, ist es immer billiger, das Schiff rückwärts aus dem Hafen zu steuern, denn der Strafzettel ist allemal niedriger als die folgenden Reparaturkosten-

Rechnung. Daß ein Schiff mal auf Kollisionskurs fahren kann, ist die natürlichste Sache der Welt, aber dann dem geisterfahrenden Schiff im Rückwärtsgang zu entfliehen, ist sicher keine Lösung. Irgendwann mal kommt der Tag, an dem sich der Spieler entscheidet, 100 High-Tech-Schiffe zu kaufen. Das benötigt dann schon eine Ewigkeit, aber dann den Spieler hundertfach zu zwingen, alle seine noch im Hafen liegenden Schiffe aus dem Kollisionsbereich zu fahren, ist schon eine Zumutung. Irgendwann pfeift jeder auf Frachten, er kauft die hochmodernen Schiffe für unter ca. 40 Millionen und verkauft diese wieder für ca. 70 Millionen, ohne daß das Schiff jemals den Hafen verläßt.

Im Grunde kann man dieses Spiel niemandem empfehlen. ah ■

Freistoß für Amiga

Die Fußball-EM ist zwar vorbei, aber das bedeutet noch lange nicht, daß jetzt eine Fußball-tote Zeit folgt. Besitzer eines Atari ST oder Amiga können die EM auf ihrem Computer nachvollziehen.

Von Grandslam, bekannt durch Spiele wie Terramex oder Fred Feuerstein, kommt ein neues Sportspiel zu uns: Euro Soccer '88, eine Fußball-Simulation. Da bislang kein übermäßig großes und qualitativ hochwertiges Soccerspiel auf dem Markt ist, war ich entsprechend skeptisch. Allerdings sollte ich eine Überraschung erleben, denn Euro Soccer '88 dürfte nur schwer zu schlagen sein. Sowohl grafisch als auch vom Sound her gehört dieses Spiel zum Besten, was für die bekannten 16-Biter zu haben ist. Große detaillierte Sprites und zahlreiche grafische Effekte motivieren den Spieler.

Zu Beginn wählen Sie sich Ihre Mannschaft aus. Wenn Ihnen die Trikot-Farben nicht zusagen, können diese selbstverständlich geändert werden. Nachdem die wichtigsten Parameter eingestellt worden sind, geht's los.

Wie von den bisherigen Fußball-Simulationen gewohnt, können Sie nur die Steuerung für jeweils einen Spieler übernehmen; welchen, wählt der Computer aus. In den meisten Fällen ist es derjenige, der dem Ball am nächsten ist. Damit Ihnen keine Verwechslung unterlaufen kann, schwebt über der aktuellen Spielfigur ein blinkender Pfeil. Auch die Figur des (Computer)-Gegners ist mit einem Pfeil gekennzeichnet.

Der oder die Spieler steuern die Fußballteams im Turnier, der Computer simuliert die Resultate der übrigen Matches. Damit ist schon in den Entscheidungsspielen Spannung und Nervenkitzel garantiert. Dann geht's ins Halbfinale und in die Endrunden.

Euro Soccer '88 wird in einer stabilen Kunststoffverpakkung auf Diskette geliefert. Als kleine Zugaben finden Sie in der Packung noch ein DIN-A2-Poster und einen Ansteck-Button. Die Anleitung ist komplett in Deutsch verfaßt, so daß auch Spieler, die der englischen Sprache nicht mächtig sind, keine Probleme haben werden.

Das Spiel kann jedem, der sich für Fußball begeistert und ein hervorragend aufgemachtes Spiel schätzt, wärmstens empfohlen werden. Den Preis von knapp 60 Mark dürfen Sie ohne Bedenken auf den Tisch legen. TB ■

Titel:	Euro Soccer '88
Getestet:	Amiga
Umsetzungen:	Atari ST

Im Test: ☒ Diskette ☐ Kassette

Preis (DM): k.A.

☒ Joystick ☒ Tastatur ☒ Maus

Wertung	0	25	50	75	100
Grafik					
Sound					
Bedienung					
Motivation					

Die Wargames-Edition

Vor kurzer Zeit erreichte uns eine wahre Flut Kriegsstrategie-Spiele von Electronic Arts Deutschland. Die Spielefreaks und auch die Programmierer scheinen neuerdings Gefallen an dieser zweifelhaften Art von Games zu finden.

Ich mußte lange überlegen, ehe ich mnich entschloß, die Kriegsspiel-Serie von Electronic Arts nehmen. Mit Vorliebe wird der Zweite Weltkrieg behandelt. Aber auch Kollektionen, mit dem jedes nennenswerte Gefecht durchgeführt werden kann, sind dabei. Die Palette reicht vom Flugzeugträgerkrieg im Pazifik über die Schlacht Rommel gegen Patton bis hin zum Bombardement Englands und Deutschlands.

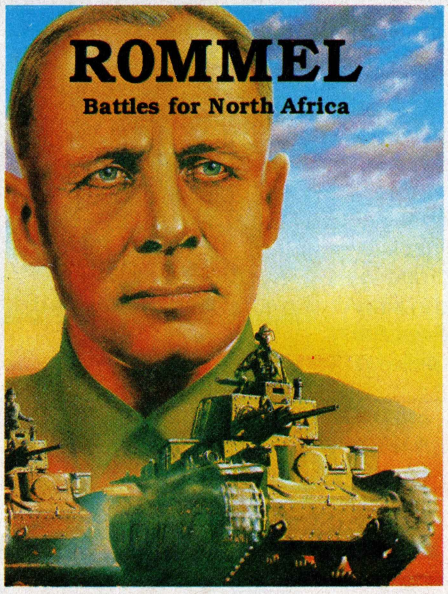
Patton versus Rommel

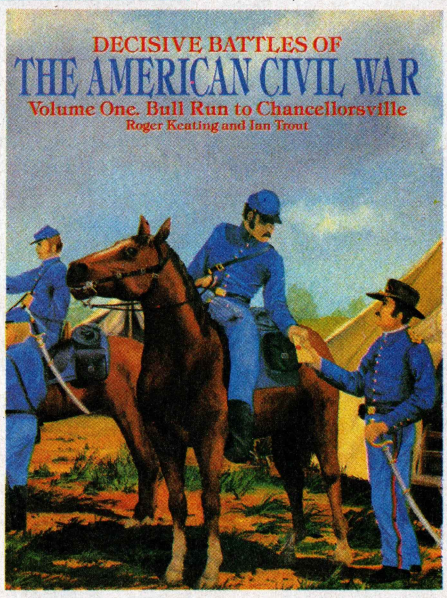
nennt sich das erste und zugleich beste Spiel dieser Wargames. Es simuliert das Gefecht in der Normandie, als sich der britische General Patton dem deutschen Feldmarschall Rommel gegenübersah.
Das Kriegsspektakel spielt sich im Computer nur auf einer grafisch ganz gut gelungenen Landkarte ab. Alle Einheiten erscheinen auf der Karte als gut erkennbare Symbole, die mit dem Joystick anzuwählen und zu steuern sind. Meldungen und Bestätigungen über eventuelle Aktivitäten erscheinen als gut lesbare Fenster innerhalb der Kartengrafik.
Über den Witz dieser Simulation läßt sich streiten. Ich konnte jedenfalls nur kurze Zeit vor dem Bildschirm ausharren, ehe mir die ganze Sache zu dumm wurde. Aber vielleicht gibt es doch so ausgefuchste Strategen, die dieses Game als eine Herausforderung betrachten.

Sie jedoch auch die Möglichkeit, unter mehreren Schlachten in Nordafrika zu wählen.
Auch bei diesem Game läuft die Handlung auf der Landkarte ab. Vergleiche mit dem oben genannten bei der Grafik unterlasse ich lieber. Die Darstellung der Karte ist der reinste Horror.
Durch die vielen verschiedenen Farben werden Sie so stark verwirrt, daß Sie sich auf der ohnehin schon Blockgrafik-artigen Karte kaum noch zurechtfinden. Überhaupt ist das ganze Spiel reichlich mit Farbe bestückt, oder besser gesagt: überladen.
Dafür kommt der Sound zu kurz. Wenn überhaupt, dann tönt nur ein langweiliges Gedudel aus dem Lautsprecher, das dem C-64 nicht im geringsten würdig ist.
Gehen wir jetzt etwas in der Zeit zurück, nämlich ins Jahr 1861, als in Amerika der große Ärger begann.

The American Civil War

Der Amerikanische Bürgerkrieg unterscheidet sich im Aufbau, zumindest auf dem





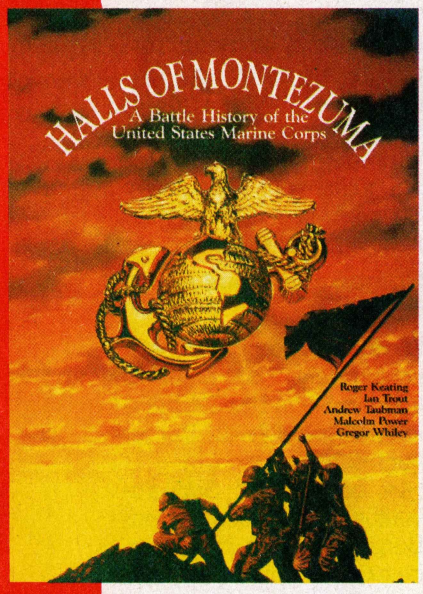
Rommel – Battles for North Africa

Game zwei beschäftigt sich ebenfalls mit dem Wüstenfuchs Rommel. Bei diesem Spiel können Sie gegen den Computer oder einen zweiten Kommandeur spielen, der entweder die Rolle der Alliierten oder die von Rommel übernimmt. Hier haben Computer, fast nicht von Rommels Schlacht um Nordafrika. Der Bildschirmaufbau ist vollkommen identisch, die Farbgebung auch, und am Sound wurde ebenfalls

nichts verbessert. Einziger Unterschied ist das Szenario. Bedient wird das Spiel über die Cursortasten. Eine Joystick-Steuerung, die wesentlich komfortabler gewesen wäre, wurde nicht eingebaut. Auch hier flaut die Motivation schon nach kurzer Spieldauer ab. Hinzu kommen die langen Wartezeiten beim Laden der Disk, da oft nachgeladen werden muß. Insgesamt also ein sehr mäßiges Spiel.

Halls of Montezuma

Für ganz Kriegslüsterne kommt von denselben Programmierern ein Simulationsspiel, das mehrere



Schlachten, unter anderem auch in den zwei Weltkriegen, zusammenfaßt. Viel mehr kann ich dazu nicht mehr sagen, außer daß sich die Kämpfe vom ersten Weltkrieg über den zweiten und den Koreakrieg bis hin nach Vietnam ziehen. Die ehrenvollen Gemetzel werden natürlich von der amerikanischen Elitetruppe Marines ausgetragen.
Da dieselben Designer am Werk waren, hat sich auch hier gegenüber den anderen Games nicht das geringste geändert. Die Grafik blieb gleich, der Sound auch.

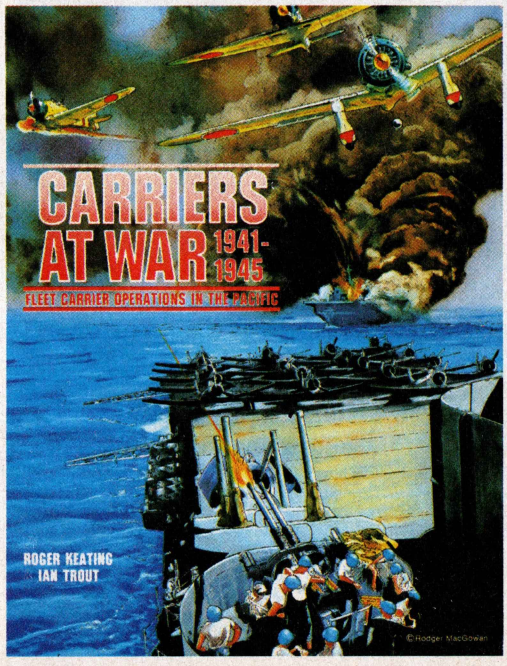
Gehen wir wieder zurück zum Zweiten Weltkrieg und sehen den Flugzeugträgern im Pazifischen Ozean zu.

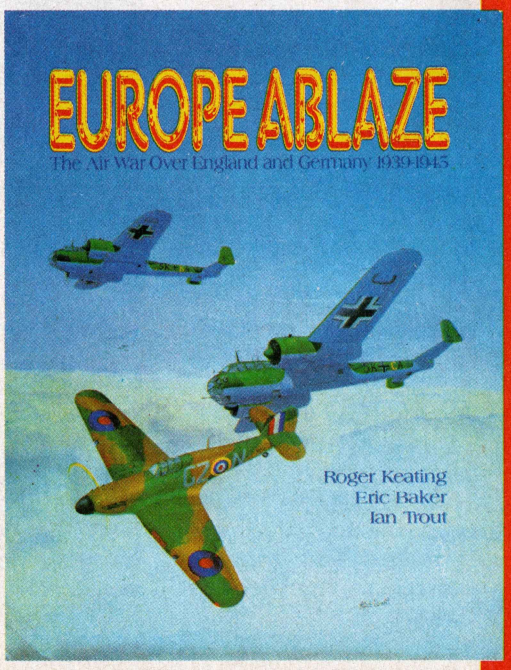
Carriers AT War 1941–1945

Es ist nicht leicht, einen richtigen Eindruck von diesem Game zu bekommen, ohne die beiden ausführlichen, sechzehn- und vierundzwanzigseitigen Handbücher komplett durchstudiert zu haben.
Ort und Zeit der Handlung ist wieder einmal der Zweite Weltkrieg im Pazifischen Ozean. Zur Auswahl stehen sechs verschiedene Schauplätze, unter anderem Pearl Harbour, Coral Sea, die Midway-Inseln und die Philippinische See. Alles Orte, die erfahrenen Veteranen wohl ein Begriff sind. Hier haben sich zwei Flugzeugträger erbitterte Schlachten geliefert: Die amerikanische „Saratoga" traf auf die japanische „Kaga".
Der Bildschirmaufbau ist wieder einmal der gleiche wie bei den obengenannten Games. Die Steuerung ebenfalls, der schlechte Sound auch.
Trotzdem scheint Carriers at War umfangreicher ausgefallen zu sein, da dem Spiel eine ganze Menge an Handbüchern und Kartenmaterial, das nicht nur bloßer Schnick-Schnack ist, beiliegen.
Kommen wir zum letzten Spiel dieser Episode: der Bombardierung Europas.

Europe Ablaze

In derselben Aufmachung wie Carriers at War geliefert, enthält auch Europe Ablaze jede Menge an Handbüchern und Karten. Diesmal findet die Handlung jedoch nicht auf hoher See statt, sondern hoch in den Lüften. Europe Ablaze simuliert den Luftkrieg über England und Deutschland.
Wie sollte es anders sein, das weitere entspricht den anderen Games. Grafik und Soundeffekte sind völlig identisch, an Komplexität erreicht es mit Leichtigkeit das Pensum von Carriers at War. Gespielt werden kann in drei Szenarien: die Schlacht um England, die Nachtoffensive britischer Bomber und der Angriff auf Berlin.
Ein Wort zu allen hier getesteten Kriegsstrategie-Spielen: Die Aufmachung und die Illustration könnte nicht besser sein. Dem Spieler werden die historischen Hintergründe und Gegebenheiten genau erklärt sowie die verwendeten Waffen, Flugzeuge, Panzer und Schiffe detailgetreu erläutert. Die Handbücher lassen keine Fragen offen und sind sorgfältig und mit viel Liebe zum Detail erarbeitet.
Trotzdem habe ich erhebliche Bedenken gegen das Thema dieser Games. Sie befassen sich leider nur – ohne Ausnahme – mit Krieg. Natürlich, Kriege sind die beste Übung für strategisches Denken, trotzdem sollte man es damit nicht übertreiben. Die Programmierer haben sich große Mühe gegeben, die Schauplätze und Situationen so realitätsgetreu wie möglich zu gestalten. Technisch gesehen sind die Games daher kaum zu übertreffen, obwohl ich das von der grafischen Ausführung nicht gerade behaupten würde.
Kurz gesagt: Es ist Geschmackssache, ob die Spiele gefallen oder nicht. Mich sprechen sie nicht besonders an. Es war außerordentlich schwierig, völlig objektiv zu bleiben, und ich gebe gerne zu, daß mir das nicht ganz gelang. Ausgefuchste Strategen sehen darin aber sicher eine große Herausforderung. Fraglich ist nur, ob sich nicht die Bundesprüfstelle bald für diese Art von Herausforderung interessiert. mn ■

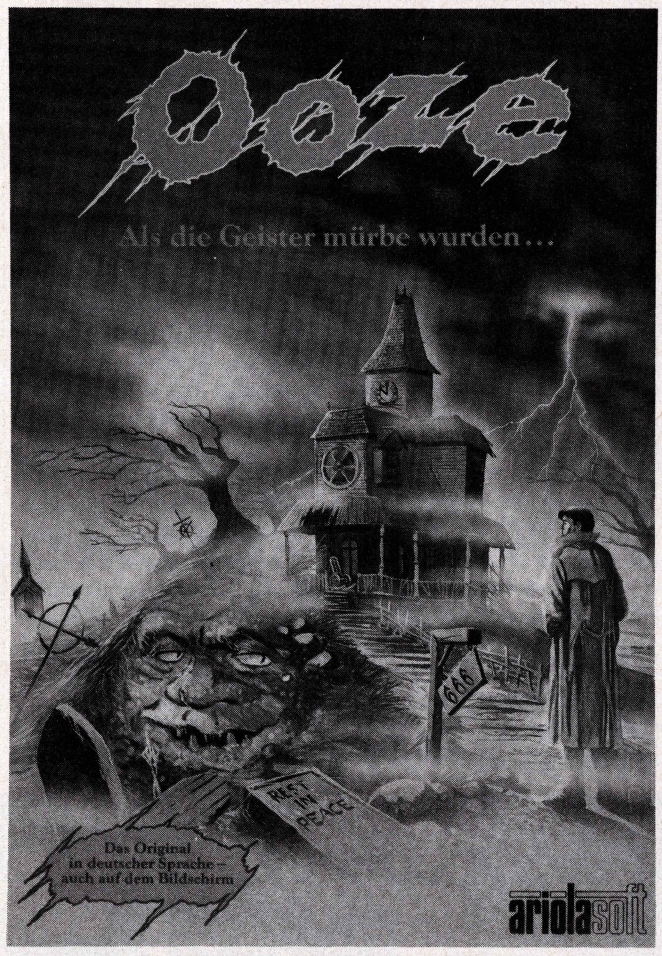




Horrortrip als Comicstrip

Sie haben gerade Lust, sich mal wieder kräftig zu gruseln? Sie haben auch gute Nerven? Und Sie wollen nebenbei auch mal ganz herzhaft lachen? Dann habe ich einen guten Tip für Sie: Besuchen Sie doch mal Rue Morgue, Hausnummer 666. Bitte klingeln bei „Ooze".

Mr. Cheez Burger, wohnhaft in 4233 Denbrough 3575, Rue Morgue No. 666, ist am 28. August 82 um 10 Uhr 25 Minuten in Denbrough, Rue Morgue 666 tot aufgefunden worden.
Der gräßlich verstümmelte Leichnam, die abgetrennten Glieder und die völlige Blutleere des Körpers weisen auf einen möglicherweise rituellen Mord hin. Ebenso die Striemen auf der Haut des Ermordeten, deren Entstehung noch immer unklar bleibt, da sie tiefe Brandmale mit aufweisen. Weitere Ergebnisse der Obduktion können beim zuständigen Gerichtsarzt eingesehen werden."
So steht es in der Sterbeurkunde Ihres so merkwürdig dahingeschiedenen Onkels Cheez Burger. Als Zeugen unterzeichneten unter anderem Mr. Stephen King und Mr. Edgar Allen Poe.
Wenn Sie das gelesen haben, werden Sie sich wohl fragen, was es daran so herzlich zu lachen gibt. Richtig, hier noch gar nichts. Das war nur eine kleine stimmungsmachende Beilage zum deutschen Grafikadventure „OOZE - Als die Geister mürbe wurden."



Einfallsreiche Hintergrundgeschichte

Wenn Sie die mit einem hervorragenden Cover bedeckte Verpackung öffnen, fallen Ihnen zuerst die besagte Sterbeurkunde und ein dreißigseitiges Heft entgegen, das sich als Tagebuch Ihres verstorbenen Onkels entpuppt. Der Roman ist sehr stimmungsvoll geschrieben und eignet sich hervorragend als Einführung in das Abenteuerspiel, zumal die Geschichte dort aufhört, wo das Spiel beginnt.
Als Cheez Burger in die alte Villa einzog, bekam er zuerst große Schwierigkeiten mit den dort wohnenden Geistern aller Art. Diese waren es nämlich nicht gewöhnt, ein fremdes lebendes Wesen zu sehen, das sie möglicherweise in ihrer Ruhe stören könnte. Also bereiteten sie dem alten Cheez kurzerhand eine Menge Terror – bis der Hausgeist Ludus, ehemaliger Hofwitzbold des Königs, herausfand, daß Mr. Burger eigentlich ganz nett ist und den Geistern gar nichts anhaben will. Nach einigen skurrilen Streichen fanden die beiden endlich zusammen und wurden unzertrennliche Freunde – kein Wunder, wenn ein Teil unsterblich war. Nachdem Ludus seine Spukkollegen, T-Bone das Skelett, Vino, den ewig betrunkenen Weingeist und Holunder, dessen liebstes Hobby ist, mit Köpfen zu kegeln, davon überzeugt hatte, daß Cheez Burger zu ihnen hält, hatte dieser eine Gruppe fester Freunde gewonnen.
Leider jedoch hatten auch die Geister ein großes Problem, in das sie Ihren Onkel einweihten: Haustyrann in Carfax Abbey ist ein Monster, das sich Ooze nennt und alle anderen Geister in Schach hält. Dieser Ooze, der laut Ludus nicht mal ein wirklicher Geist ist, weil er noch nie gelebt hat, ist durch und durch böse und hat überall seine Handlanger im Haus versteckt. Ooze hat aber auch eine sehr bedeutende Stärke: Er kann seine Gestalt beliebig verändern.

Roman zum Spiel als Beigabe

Der Plan, Ooze zu vernichten, schlug fehl. Das Ergebnis können Sie in der Sterbeurkunde von Cheez Burger nachlesen.
Hier beginnt das Grafikadventure. Sie müssen als Ham Burger, der Erbe von Cheez, das Haus übernehmen und versuchen, Ooze mit all seinen Spionen zu vertreiben. Daß das gar nicht so einfach ist, hat auch schon Ihr Onkel zu spüren bekommen.
Das wichtigste an einem Abenteuerspiel ist der Eingabeinterpreter. In diesem Fall setzt der deutsche Parser wohl neue Maßstäbe. Er versteht ein großes Stück des Vokabulars und auch die deutsche Grammatik einwandfrei. Lange Sätze, durch Kommas und Bindewörter verknüpft, stellen für ihn kein Problem dar. Wenn er einmal eine Eingabe nicht versteht, dann läßt das Programm Sie das deutlich spüren, indem

es Sie durch einen frechen Kommentar darauf aufmerksam macht.
Eine kleine Schwäche hat der Parser jedoch: Schnelltipper werden sich umgewöhnen und einen langsameren Gang einlegen müssen. Tippen Sie nämlich zu schnell, so kommt der Interpreter nicht mehr mit oder läßt der Bequemlichkeit halber einfach ein paar Buchstaben aus. Nach einer kurzen Eingewöhnungszeit stellt das jedoch auch kein Problem mehr dar.
Der Textteil ist in Deutsch verfaßt und enthält nicht den kleinsten Rechtschreib- oder Grammatikfehler. Vom Umfang her stellt Ooze sogar die Infocom-Produkte in den Schatten. Man kann ohne Übertreibung sagen, daß das neue Dragonware-Game einem illustrierten Horrorroman in der Ich-Perspektive sehr nahe kommt, so detailreich und durchdacht sind die riesigen Textmassen.
Zu einem Grafikadventure gehören natürlich auch Grafiken. Denen von Ooze kann wirklich das Prädikat „Hervorragend" verliehen werden. Zu jedem der siebzig Räume gibt es eine eigene Grafik, an der ich stundenlang die Augen weiden könnte. Teilweise lassen die Bilder sogar die von Jinxter und The Pawn hinter sich.
Mit den Texten und Bildern gelang es den Programmierern ausgezeichnet, eine Horrorstimmung zu erzeugen. Am besten, Sie versuchen nachts, bei Kerzenlicht und geeigneter Musik, Ooze zu schlagen. Aber auch tagsüber läßt die Spannung nicht lange auf sich warten.

Fazit

Mit „Ooze - Als die Geister mürbe wurden" wurde das bislang beste deutsche Adventure programmiert, das sich hinter amerikanischen Produkten wie The Prawn nicht zu verstecken braucht. Durch den gelungenen Parser, die hervorragenden Texte, die sehenswerte Grafik und die witzige Vorgeschichte wird es für deutsche Abenteuerspiele neue Maßstäbe setzen, die zu übertreffen für andere Firmen schwierig sein wird.

mn ■

Titel:	**Ooze**
Getestet:	**Amiga**
Umsetzungen:	**Atari ST, MS-DOS, C-64**

Im Test: ☒ 💾 ☐ 📼

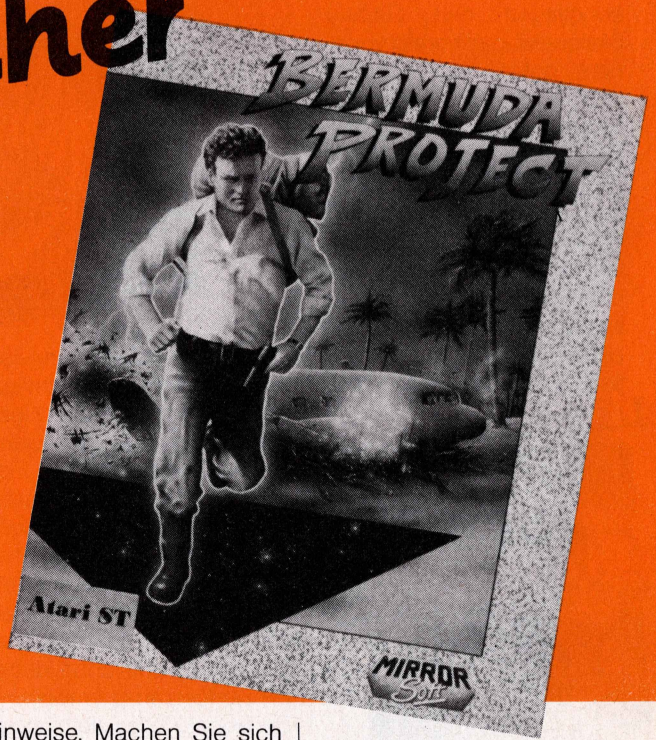
Preis (DM): 79,90

☐ Joystick ☒ Tastatur ☐ Maus

Wertung	0	25	50	75	100
Grafik					
Sound					
Bedienung					
Motivation					

Ein mörderischer Auftrag

Schon seit über 50 Jahren häufen sich die Rätsel um verschwundene Schiffe und Flugzeuge im Bereich des Bermuda-Dreiecks. Nun soll ein Reporter Licht in die Sache bringen.



Das Rätsel, welches das Bermuda-Dreieck umgibt, beschäftigt schon lange die Wissenschaftler und Theologen. Ganze Schiffskonvois und Flugzeugkolonnen verschwinden plötzlich spurlos vom Radarschirm und tauchen nie mehr auf. Von Mirrorsoft kommt jetzt ein Computerspiel zu uns, das sich mit dem Bermuda-Mysterium beschäftigt.
Sie übernehmen die Rolle eines hochbezahlten Reporters, der von seiner Zeitung den Auftrag erhält, sich auf den Bermuda-Inseln ein wenig umzusehen. Doch auch Sie entkommen dem berüchtigten Dreieck nicht. Ihr Flugzeug stürzt ab.
Nach mehrstündiger Bewußtlosigkeit finden Sie sich auf einer offenbar unbewohnten Insel wieder. Ihr einziger Begleiter, der Pilot, ist tot. Was nun?
Wie sie weiter vorgehen, bleibt Ihnen überlassen. Auch die Anleitung gibt keine Hinweise. Machen Sie sich also auf die Suche nach Einwohnern und spüren sie mit deren Hilfe die Wracks der verschwundenen Schiffe und Flugzeuge auf. Dabei gelten die gleichen Bedingungen wie in der Realität, das heißt, Sie müssen auch für Nahrung und Trinkwasser sorgen und dürfen Ihrer

Fortsetzung auf S. 14

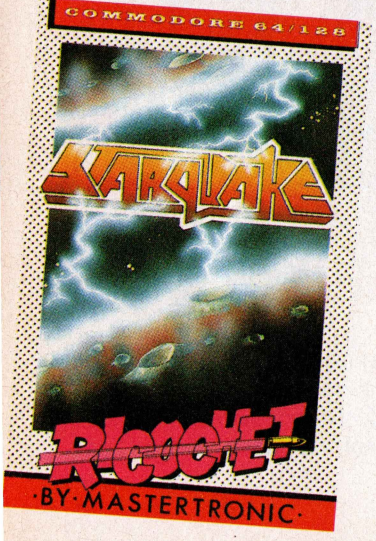
Games für kleine Geldbeutel

Daß gute Software nicht unbedingt teuer sein muß, haben uns diverse Hersteller wieder einmal mit einer geballten Ladung Billigspiele bewiesen.

Alle hier getesteten Billigspiele werden in der gewohnten Kunststoffverpackung auf Kassette geliefert. Zu erhalten sind die Games für etwa zehn bis fünfzehn Mark in den Fachabteilungen der Kaufhäuser und im Computer-Fachhandel.

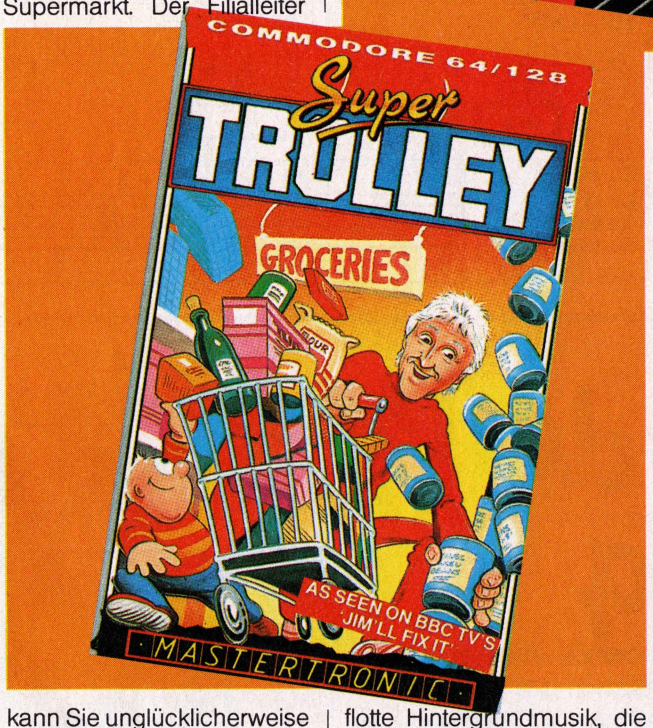
Starquake

Eine geheime Macht hat vom Universum Besitz ergriffen. Zahlreiche Bösewichte haben die friedliebenden Bewohner verjagt und treiben nun ihr zerstörerisches Unwesen. Ihre Aufgabe liegt auf der Hand: Stürzen Sie sich ins Getümmel und machen Sie den unerwünschten Eindringlingen den Garaus. Starquake, das bereits vor einem guten Jahr auf den Markt kam, ist ein Arcade-Action-Game für einen Spieler. Mastertronic hat, wie in letzter Zeit schon oft, die Vertriebsrechte erworben und bringt Starquake als Billigspiel unters Volk. Wer das Game noch nicht hat, der sollte es sich so schnell wie möglich besorgen. Gute Grafiken und flotte Musikuntermalung lassen so schnell keine Langeweile aufkommen. Starquake gibts für CPC, Commodore 64 und Spectrum 48K.



Super Trolley

Sie sind ein kleiner Hilfsangestellter in einem bekannten Supermarkt. Der Filialleiter kann Sie unglücklicherweise nicht ausstehen und schikaniert Sie, wo er nur kann. Heute hat er sich wieder was ganz Fieses ausgedacht: Erst müssen Sie diverse Waren sortieren, dann mit einem Einkaufswagen die Regale leeren und die Artikel anderswo wieder einräumen. Das alles natürlich in möglichst kurzer Zeit und ohne dabei die Kunden anzurempeln.



Bei Super Trolley, dem neuesten Billigspiel von Mastertronic, ist die Grafik zwar nicht gerade berauschend, aber noch ausreichend. Als kleine Entschädigung ertönt zu dem ganzen Spektakel eine flotte Hintergrundmusik, die vom Sound-Hexer Rob Hubbard programmiert wurde. Ob das Game auch Spaß macht, hängt vom persönlichen Geschmack ab. Computerneulingen kann man es vielleicht empfehlen, da die Aufgaben nicht allzu anspruchsvoll sind. Am besten, Sie sehen sich das Game vor dem Kauf genau an. Super Trolley gibt's für Commodore 64.

Subterranea – Rack it

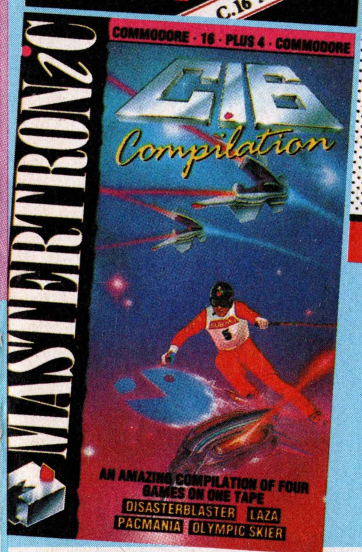
In 16 Levels müssen Sie bei Subterranea Ihr Joystick-Gefühl unter Beweis stellen. Wieder einmal sind bösartige Aliens in unser friedliches Universum eingefallen. Die sollen Sie vertreiben, am besten durch kräftiges Drücken auf den Feuerknopf. Wie bei Airwolf steuern sie in diesem Game von Hewson ein Raumschiff durch die unterschiedlichsten Höhlensysteme. Dabei begegnen Ihnen ständig feindliche Flugobjekte, die sofort das Feuer eröffnen. Es empfiehlt sich, ebenfalls auf die Feuertaste zu drücken und zwar möglichst schnell, denn sonst sind Sie Ihr Raumschiff und damit auch eines der drei Leben unweigerlich los.

Subterranea könnte durchaus als Ballerspiel Erfolg haben, zumal auch Grafik und Sound nicht gerade schlecht sind. Leider jedoch existiert keine Möglichkeit, das Raumschiff mit dem Joystick zu steuern, so daß Sie sich mit einer äußerst umständlichen Tastaturbelegung die Finger verrenken dürfen. So kommt einfach nicht der rechte Spielspaß auf. Getestet haben wir Subterranea – Rack it auf dem Commodore 64 mit Farbmonitor.

Super Gran

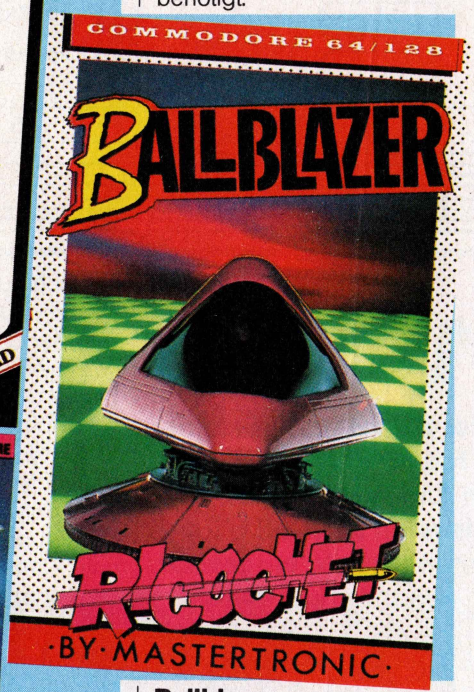
Von Tynesoft, vor allem bekannt durch das Sportspiel

Winter Olympiad '88, kommt ein neues Spiel für den Commodore 16.
Bei Super Gran handelt es sich um das offizielle Computerspiel zur gleichnamigen englischen Fernsehserie. In einem amateurhaft gebastelten Helicopter müssen Sie einige Abenteuer bestehen. Grafisch schneidet Super Gran nicht sonderlich gut ab, da sich Sprites und Hintergrund kaum unterscheiden lassen. Auch das Spielprinzip reißt den Spieler nicht gerade vom Hocker. Wem's aber gefällt...



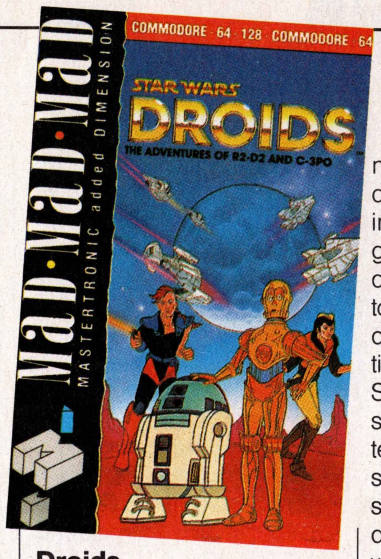
C16 Compilation

Eine preiswerte Spielesammlung bietet Mastertronic jetzt für alle C16/Plus4-Besitzer an. Auf der Kassette, die den eindrucksvollen Namen C16 Compilation trägt, sind vier Spiele, die in Sachen Grafik und Motivation eine preiswerte Alternative zu den meist überteuerten Spielesammlungen sind.
Im einzelnen handelt es sich um Pacmaina, ein Pac-Man-Verschnitt, der von der Geschwindigkeit her seinesgleichen sucht, Disasterblaster, ein gelungenes Weltraum-Ballerspiel, Laza, ebenfalls ein Ballerspiel, allerdings nicht ganz so motivierend, und schließlich das Sportspiel Downhill. Hier müssen Sie einen Abfahrtslauf hinter sich bringen. Für alle vier Spiele wird ein Joystick benötigt.



Ballblazer

Vor etwa 18 Monaten brachte die Firma Lucasfilm Games das Geschicklichkeitsspiel Ballblazer heraus, für das nun Mastertronic die Vertriebsrechte aufgekauft hat. Angeboten wird Ballblazer für den Commodore 64/128 und den Schneider PC. Es ist ein Spiel, bei dem der Bildschirm gesplittet ist, damit zwei Spieler gleichzeitig gegeneinander antreten können. Spaß macht Ballblazer allemal, so daß dem Kauf nichts im Wege steht. Beim CPC empfiehlt sich ein Farbmonitor.

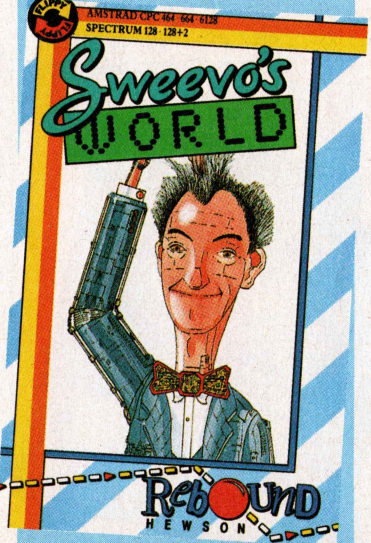


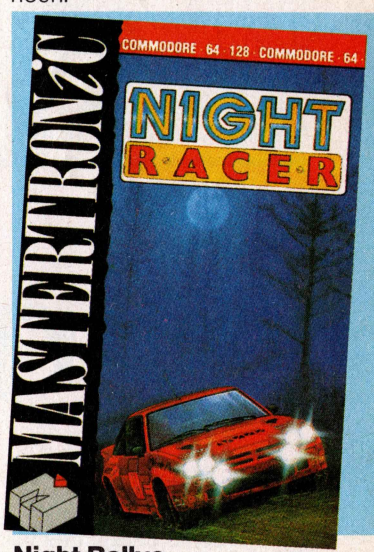
Droids

Droids ist ein neues Ballerspiel aus der Mastertronic-MAD-Serie, das durch hervorragende Grafiken und flotten Sound angenehm überrascht. Ihre Aufgabe ist es, die beiden Roboter R2-D2 und C-3P0 durch ein feindliches Raumschiff zu steuern (möchte bloß wissen, wie die da reingekommen sind!) und dabei so viele Aliens wie möglich zu zerstören. Wenn Ihnen die Namen der Robotis bekannt vorkommen: Ja, es sind die aus Star Wars, wie Sie am Untertitel erkennen können. Droids können Sie künftig auf dem Spectrum 48/128/Plus2 und dem Schneider CPC 464 spielen.

Sweevo's World

Den Vertrieb des vor zwei Jahren sehr erfolgreichen Spiels Sweevo's World hat die Firma Mastertronic übernommen und für den Schneider CPC und den Spectrum in neuer Verpackung herausgebracht. Sie übernehmen in diesem Game die Rolle der tollpatschigen Ente Sweevo, die ihren Planeten von bösartigen Sprites säubern muß. Sweevo's World ist ein typischer Vertreter der bekannten 3D-Adventures, wie es sie seit Knight Lore dutzendweise auf dem Markt gibt. Allerdings bietet das Spiel wesentlich bessere Grafiken als seine „Soft-Kollegen". Wer also noch nicht allzu viele Games dieser Kategorie zu Hause hat, darf sich Sweevo's World ohne Bedenken zulegen. Der Preis von knapp 15 Mark ist sicher nicht zu hoch.





Night Rallye

Eine Rallye-Simulation für den Commodore 64/128 bietet ebenfalls Mastertronic an. Allerdings darf man sich vom Titel nicht irreführen lassen, handelt es sich doch wieder einmal nur um ein simples Autorennspiel, das überhaupt nichts Neues bietet. Night Rallye kann nur allein und nur mit Joystick gespielt werden. Grafisch ist Night Rallye ganz gut gelungen, auch die Soundeffekte können sich hören lassen. Da das Spiel auch nur knapp über 10 Mark kostet, dürfte es eine interessante Alternative zu den teuren Autorennspielen sein. Thomas Bosch ■

Fortsetzung von S. 11

Spielfigur keine allzu große Anstrengung zumuten. Um ans Ziel zu kommen, ist es außerdem ratsam, jedes Objekt, sei es ein Stein, ein Baum oder eine Trinkwasserquelle, peinlich genau zu untersuchen.
Gesteuert wird die Spielfigur mit der Maus. Über die beiden Maustasten lassen sich auch durch Pulldown-Menüs die verschiedenen Aktionen anwählen. Ihre Spielfigur kann Objekte untersuchen, aufnehmen, ablegen, benutzen, verbinden und trennen. Außerdem können Sie sich jederzeit den Status Ihres Reporters betrachten. Dort finden Sie auch eine entsprechende Warnung, wenn der Held kurz vor der Erschöpfung steht.
Daß das abzusuchende Gebiet relativ groß ist, kann es ziemlich lange dauern, bis das Spielende erreicht ist. Glücklicherweise haben die Autoren auch Optionen zum Abspeichern und Wiedereinladen des aktuellen Spielstandes eingebaut. Hierzu muß sich eine frisch formatierte Diskette im Laufwerk A befinden.
Die Grafik von Bermuda-Projekt beschränkt sich zwar lediglich auf ein paar Objekte, ist aber völlig ausreichend. Da es notwendig ist, sämtliche Objekte zu untersuchen, würde sich das Spiel lange hinziehen, wenn der Bildschirm damit überfüllt wäre. Außerdem dürfen Sie nicht vergessen, daß Sie sich vorwiegend am Sandstrand einer einsamen Insel aufhalten, wo es keine Zivilisation gibt.
Zu Beginn des Spieles erklingt aus dem Lautsprecher ein digitalisiertes Musikstück. Ansonsten sind nur diverse Hintergrundgeräusche zu hören.

Fazit

Bermuda-Projekt ist ein Spiel für Leute mit viel Zeit und Geduld. Wer also einmal länger als eine Stunde an einem Computergame sitzen möchte und außerdem eine Vorliebe für Rätsel hat, dem kann ich dieses Spiel nur empfehlen. TB ■

Titel:	**Bermuda Project**
Getestet:	**Atari ST**
Umsetzungen:	**Amiga (in Vorbereitung)**

Im Test: ☒ ☐

Preis (DM): 59,-

☐ Joystick
☐ Tastatur
☒ Maus

Wertung	0	25	50	75	100
Grafik					
Sound					
Bedienung					
Motivation					

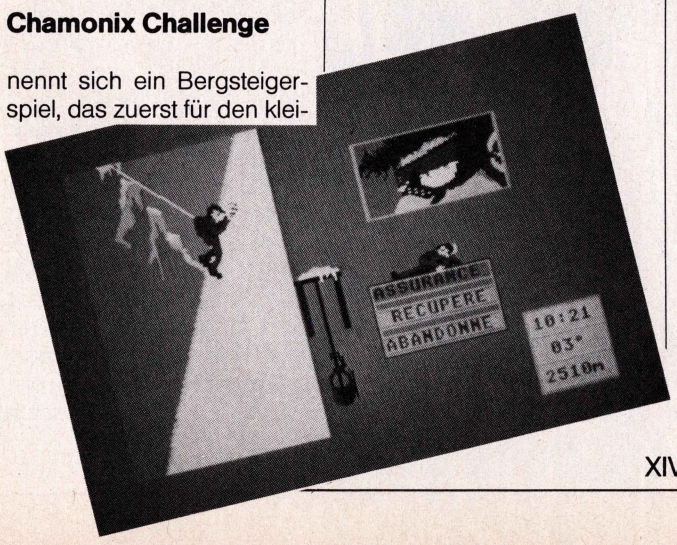
Kurzberichte

Dieser Monat brachte leider nicht viele Neuerscheinungen und Umsetzungen für alle Computer. Sowohl für den C-64 als auch für Atari ST und Amiga hat wohl die Sommerpause eingesetzt.

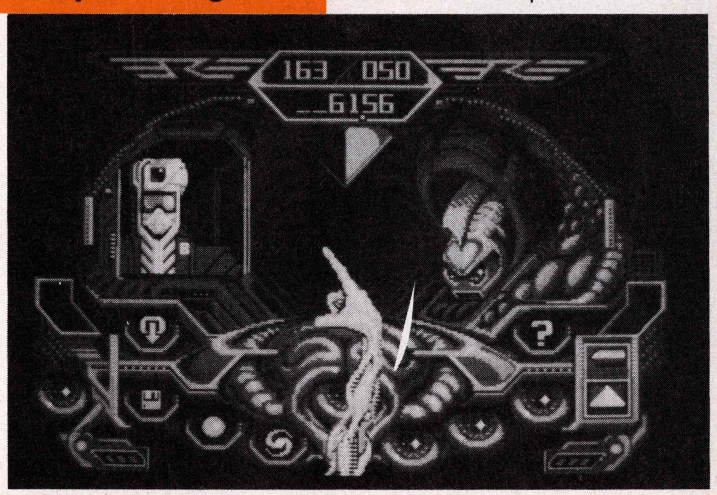
Zwei Umsetzungen brachte Infogames für den Amiga auf den Markt. Beide sind schon seit einiger Zeit für andere Computer wie C-64 und ST erhältlich und wurden von uns getestet.

Chamonix Challenge

nennt sich ein Bergsteigerspiel, das zuerst für den kleinen Commodore erschien. Ziel des Spiels ist die Besteigung eines oder mehrerer Berggipfel. Dazu müssen sowohl Gletscher als auch Felswände erfolgreich überwunden werden. Das Game ist ein technisch hochwertiges und recht schwieriges Geschicklichkeitsspiel.



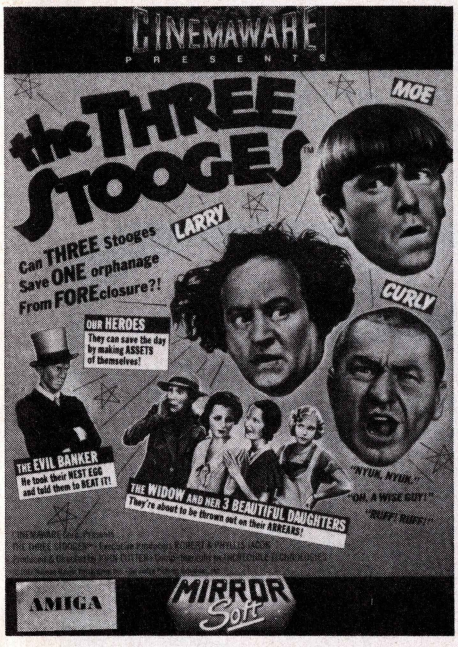
Die Grafik ist für Amiga-Maßstäbe jedoch enttäuschend ausgefallen. Sie unterscheidet sich nur geringfügig von der des Commodore 64. Die Schwierigkeit und der Spielwitz sind jedoch gleichgeblieben. Bergfans sollten sich Chamonix Challenge einmal ansehen. Für andere ist es empfehlenswert, erst einmal probezuspielen.
Das andere Game wurde vom Atari ST adaptiert:



Die Arche des Captain Blood

Captain Blood ist ein sehr kompliziertes Spiel, das in

keine Kategorie so recht einzuordnen ist. Aufgabe ist es, als Programmierer seine Einzelteile wieder aufzusammeln, die in einem Universum innerhalb des Computer verstreut sind. Dazu muß er Kontakt mit fremden Lebewesen aufnehmen. Exotische Planeten anzufliegen und gegebenenfalls auch zu zerstören ist hier angesagt.
Die Grafik ist identisch mit der des Atari ST. Das Spiel hat rasante Fraktalgrafik und herrliche Farbeffekte vorzuweisen. Musik und Sounduntermalung sind auch nicht zu verachten. Insgesamt eine gelungene Umsetzung eines gelungenen Spiels.



The Tree Stooges

Nur einen Kurztest war uns The Tree Stooges für den Amiga wert. Ziel des Spiels, das sich an der Fernsehserie orientiert, ist, für eine verarmte Familie Geld aufzutreiben. Dazu lasssen sich die drei Clowns allerlei einfallen.
Grafisch ist das Spiel allererste Güte, der Spielwitz bleibt jedoch auf der Null-Marke. Das, was sich die Programmierer haben einfallen lassen, mag ja als Grafikdemo ganz gut sein, für ein Spiel ist jedoch nichts drin. Oder finden Sie daran Gefallen, wenn sich einige Leute minutenlange Tortenschlachten liefern oder ein Watschenspiel veranstalten, bei dem es einzige Aufgabe des Spielers bleibt, den Feuerknopf zu drücken?
Um ein gutes Spiel zu gestalten, hätten sich die Designer mehr einfallen lassen müssen als einige schwachsinnige Comicstrips mit teilweise recht langen Sequenzen dazwischen, in denen der Spieler nichts zu tun hat als kräftig zu gähnen.

Gauntlet II für Atari ST

Der glänzende Spielautomat und Nachfolger des Arcade-Games Gauntlet wurde jetzt auch für den Atari ST adaptiert. Die Umsetzung ist vom Automaten kaum zu unterscheiden. Das Game hat schnelle und hochauflösende Grafik und hervorragenden Sound vorzuweisen. Sogar an die Sprachausgabe des Automaten wurde gedacht. Ein besonderer Gag ist, daß auch am ST vier Spieler mitspielen können. Allerdings ist das leider nur mit einem speziellen Joystick-Interface möglich. Trotzdem ist diese Version wesentlich besser gelungen als sein Vorgänger Gauntlet I.

Atari-Umsetzung von ZARCH

Das bekannte Archimedes-Spiel Zarch können nun auch ST-Besitzer genießen. Das schnelle Aktiongame, bei dem außerirdische Wesen aller Art abgewehrt werden müssen, nennt sich auf dem ST Virus. Die Grafik bleibt im wesentlichen gleich, nur die Geschwindigkeit hat gelitten, was aber nichts heißen will. Das Game ist noch schnell genug, um einen fließenden Spielverlauf zu garantieren. Atari-Besitzer sollten sich Virus unbedingt ansehen.

Aktion-Game für Atari ST

She-Fox nennt sich ein neues Aktion-Spiel, das unter anderem auch für den ST erscheint. Das Game ist ein simples Scroll-Spiel, bei dem ausnahmsweise einmal eine Heldin die Hauptrolle spielt. Die Amazone muß sich am laufenden Band mit ihrer Peitsche gegen wilde Tiere und ein Zeitlimit behaupten. Grafisch ist das Spiel ganz gut gelungen, vom Spielwitz blieb allerdings nichts mehr übrig. Bereits nach einigen Spielminuten wird die ständige Raserei in eine vorgegebene Richtung langweilig. Insgesamt also ein sehr mittelmäßiges Spiel.

Pink Panther für den Commodore 64

Das in der letzten Ausgabe getestete Amiga-Spiel The Pink Panther wurde für den C-64 adaptiert. Kaum zu glauben: Die Grafik ist zwar nicht so detailreich, dafür aber schneller als auf dem Amiga. Auch der Sound ist für C-64-Maßstäbe hervorragend. Der Spielwitz hat nicht im geringsten gelitten. Aus dem kleinen Commodore haben die Programmierer herausgeholt, was er hergibt. Eine Umsetzung, wie sie nicht besser sein könnte. mn ■

Players' Pages

Hallo, liebe Spielefreaks und Joystickartisten. Willkommen bei den Players Pages der LOAD & RUN. Wer Tips, Tricks, Pokes und Lösungen zu jedem x-beliebigen Spiel und für jedes x-beliebige System hat oder sucht, ist hier richtig!

Letzten Monat hat sich wieder viel getan bei uns. Entsprechend umfangreich fallen die Players Pages aus. Doch ich will mich nicht mit langen Vorreden aufhalten, sondern gleich loslegen.

Ranarama

Für Ranarama-Spieler auf dem C64/128 haben wir folgenden Poke bereit, der nach dem üblichen load-reset-poke-sys-Rezept eingebaut wird.

POKE 37104,96
POKE 33969,234
POKE 33970,234

Game Over Part II

Für Part II von Game Over finden viele kein passendes Codewort. Wie wär's denn mal mit ZAPPA? Übrigens werden für dieses Spiel noch jede Menge Strategie-Hinweise gesucht. Für alle Game-Over-Spieler auf dem Spectrum habe ich auch noch einen Code anzubieten: 18024.

Terramex

Haben Sie schon mal versucht, mit dem Ballon in die Lüfte zu steigen? Sicherlich, denn mit dem Schirm geht es ja einwandfrei. Aber wer kann uns verraten, wie der Ballon wieder zum Landen überredet werden kann? Die LOAD & RUN-Redaktion hat es auch nach mehreren durchwachten Nächten noch nicht geschafft. Übrigens: Wenn Sie es nicht schaffen, warum gehen sie dann nicht in den Untergrund. Wenn Sie in den Brunnen klettern, kommen noch jede Menge weitere Räume, in denen unzählige Gefahren lauern wie beispielsweise eine Schlange, die aber durch Flötenmusik beruhigt werden kann.

Die Erbschaft

Viele Leser haben sich über die Tips zum dritten Teil der Erbschaft in LOAD & RUN 4/88 gefreut. Andere dagegen konnten damit nichts anfangen, weil sie nicht mal den ersten Teil beenden konnten. Um ihrem Leiden nun ein Ende zu bereiten, hier eine Liste mit den Gegenständen, welche die einzelnen Personen erhalten:

Neger.....Trompete
Mann mit Glatze .....Taschenlampe
Mafioso.....Pistole
Chinese.....Kerzenhalter
Mann mit Scheiten..... Kaktus
Punkerin ..... Bügeleisen
Mann mit Glatze & Bart..... Kette

Bei der Gelegenheit noch ein kleiner Tip: Es empfiehlt sich, stets zu Fuß zu gehen und den Aufzug zu meiden. Im Zimmer gleich links nach Ihrem eigenen Wohnraum finden Sie in der Schublade ein zusätzliches Taschengeld.

ANZEIGENSERVICE

Die große Börse für jeden Zweck in 64/128 SPECIAL. Kostenlos für Privat-Inserenten. Einfach Coupon ausschneiden, fotokopieren o.ä., ausfüllen und ab die Post – Freimachen nicht vergessen! – Unsere Adresse steht auf dem Coupon, ebenso die Preise für gewerbliche Anbieter! Achtung! Wir weisen ausdrücklich darauf hin, daß wir offensichtlich gewerbliche Anzeigen nicht kostenlos veröffentlichen und uns jedweden Abdruck kostenloser Anzeigen vorbehalten müssen, insbesondere, wenn deren Inhalt nicht CBM-typisch ist oder gegen geltendes Recht verstößt. Private Chiffreanzeigen werden nicht aufgenommen. Für Privatanbieter: maximal acht Zeilen a 28 Anschläge. Für gewerbliche Anbieter: 4,80 DM pro Zeile.

COMMODORE WELT
ANZEIGENABTLG **128/III**
POSTFACH 1161
D-8044 UNTERSCHLEISSHEIM

Name _____
Vorname _____
Straße/Hausnr. _____
Plz/Ort _____

DAS SONDERANGEBOT: PRIVATE KLEINANZEIGEN KOSTENLOS!

Das bietet Ihnen 128 SPECIAL: KLEINANZEIGEN SIND KOSTENLOS FÜR PRIVATANBIETER! Suchen Sie etwas, haben Sie etwas zu verkaufen, zu tauschen, wollen Sie einen Club gründen? Coupon ausfüllen, auf Postkarte kleben oder in Briefumschlag stecken und abschicken. So einfach geht das. Wollen Sie das Heft nicht zerschneiden, können Sie den Coupon auch fotokopieren. Oder einfach den Anzeigentext uns so schicken, auf Postkarte oder im Brief. Aber bitte mit Druckbuchstaben oder in Schreibmaschinenschrift!
Und: Einschließlich Ihrer Adresse und/oder Telefonnummer sollten acht Zeilen à 28 Anschläge nicht überschritten werden.
ACHTUNG: WICHTIGER HINWEIS!
Wir veröffentlichen nur Kleinanzeigen privater Inserenten kostenlos. Gewerbliche Anzeigen kosten pro Zeile DM 4,80 plus Mehrwertsteuer!
Wir versenden für Privat-Inserenten keine Beleg-Exemplare!
Chiffre-Anzeigen sind nicht gestattet! Wir behalten uns vor, Anzeigen, die gegen rechtliche, sittliche oder sonstige Gebote verstoßen, abzulehnen!
Anzeigenabdruck in der Reihenfolge ihres Eingangs, kein Rechtsanspruch auf den Abdruck in der nächsten Ausgabe!
Die Insertion ist nicht vom Kauf des Heftes abhängig! Wir behalten uns vor, Anzeigen, die nicht zum Themenkreis des Heftes – Computer – gehören, nicht abzudrucken oder sie nur insoweit zu berücksichtigen, wie es der Umfang des kostenlosen Anzeigenteils zuläßt.

VERDIENEN SIE GELD MIT IHREM COMPUTER!

Haben Sie einen Commodore VC 20 oder C 64? Einen 16/116, Plus 4? Oder einen 128? Können Sie programmieren? In Basic oder Maschinensprache? Dann bietet COMMODORE-WELT Ihnen die Möglichkeit, mit diesem Hobby Geld zu verdienen!

Wie? Ganz einfach. Sie senden uns die Programme, die Sie für einen Abdruck als geeignet halten, zusammen mit einer Kurzbeschreibung, aus der auch die verwendete Hardware – eventuelle Erweiterungen – benutzte Peripherie – hervorgehen muß (Schauen Sie sich dazu den Kopf unserer Programmlistings an.)

Benötigt werden: Zwei Listings des Programms sowie eine Datenkassette oder Diskette! Wenn die Redaktion sich überzeugt hat, daß dieses Programm läuft und sich zum Abdruck eignet, zahlen wir Ihnen pro Programm je nach Umfang bis zu DM 300,–!

Sollten Sie keinen Drucker haben, genügt der Datenträger.

Sie erhalten Ihre Kassette/Diskette selbstverständlich zurück, wenn Sie einen ausreichend frankierten Rückumschlag mit Ihrer Adresse beifügen.

Bei der Einsendung müssen Sie mit Ihrer Unterschrift garantieren, daß Sie der alleinige Inhaber der Urheber-Rechte sind! Benutzen Sie bitte anhängendes Formular! (Wir weisen darauf hin, daß auch die Redaktion amerikanische und englische Fachzeitschriften liest und „umgestaltete" Programme ziemlich schnell erkennt).

Um Ihnen die Arbeit zu erleichtern, finden Sie hier ein Formular. Sie können es ausschneiden oder fotokopieren.

Name des Einsenders: _____
Straße/Hausnr./Tel.: _____
Plz/Ort: _____

Hiermit biete ich Ihnen zum Abdruck folgende(s) Programm(e) an:

_____
_____
_____
_____

Benötigte Geräte: _____
_____

Beigefügt ( ) Listings ( ) Kassette ( ) Diskette

Ich versichere, der alleinige Urheber des Programmes zu sein!
Hiermit ermächtige ich die Redaktion, dieses Programm abzudrucken und wirtschaftlich zu verwerten. Sollte es in den Kassetten-Service aufgenommen werden, erhalte ich auch dafür eine entsprechende Vergütung, das Copyright geht insoweit auf den Verlag über.

_____
Rechtsverbindliche Unterschrift

COMMODORE WELT
PROGRAMM-REDAKTION
POSTFACH 1161
D-8044 UNTERSCHLEISSHEIM

Pokern auf dem C128

Keine Angst, Ihr Computer zieht Ihnen nicht das Geld aus der Tasche. Der C128 begnügt sich mit Punkten. Er schenkt dem Spieler sogar 150 dieser wertvollen Spielermünzen. Ein Spiel mit diesem Pokerautomat kostet zehn Punkte. Außerdem besteht ein Spiel aus zwei Teilen. Im ersten Teil werden alle Karten neu aufgelegt. Wenn dies geschehen ist, kann der Spieler mit den Tasten eins bis fünf die Karten markieren, die nicht erneut aufgelegt werden sollen. Wird zum Beispiel die Taste eins betätigt, so leuchtet der Kreis unter der ersten Karte auf und die Karte wird gehalten. Wird erneut die Taste eins betätigt, wird die Karte nicht mehr gehalten. Mit der Leertaste wird der zweite Durchlauf gestartet und der Computer zieht für alle nicht markierten

Karten eine neue Karte. Falls nun ein Gewinn angezeigt wird, so kann dieser mit der ESC-Taste übernommen oder mit der Space-Taste die Risikoleiter eingeschaltet werden. Der Faktor, mit dem der Gewinn multipliziert wird, und die Null leuchten nun abwechselnd auf. Durch Drücken der Space-Taste wird diese Ausgabe gestoppt. Leuchtet die Zahl Null auf, so ist der Gewinn verloren. Leuchtet der Multiplikationsfaktor auf, so kann dessen Wert übernommen werden oder wieder die Risikoleiter eingeschaltet werden. Der Multiplikator ist dann natürlich höher. Der Gewinn kommt in das Feld High-Score und wenn alle 150 Punkte verbraucht sind, so können aus diesem Feld Punkte zum Weiterspielen geholt werden. Doch jeder Punkt, der zum Weiterspielen benötigt wird, kostet vier Punkte aus dem High-Score-Feld. Außerdem übernimmt der Computer nur maximal 100 Punkte im Punktefeld. □

```
10 rem ========================128  <ob>
20 rem (p) commodore welt       ==  <hf>
30 rem ==========================  <mm>
40 rem (c) by                   ==  <pp>
50 rem    siegfried ederer jun.==   <ee>
60 rem                          ==  <nd>
70 rem version 7.0   40z./ascii==   <ah>
80 rem pc-128 floppy/datasette ==   <ho>
90 rem ==========================  <km>
100 dimy(11)                        <oi>
110 scnclr                          <me>
120 graphic1,1                      <me>
130 color1,1:color0,2:color4,2      <ib>
140 scnclr:char,5,10,"bitte 45 sec
 warten"                            <pp>
150 char,5,11,"sshapes werden defi
niert":sleep3                       <ma>
160 gosub15120:gosub13810           <bh>
170 rem herz neun teil eins         <lp>
180 width1                          <ag>
190 scnclr                          <gf>
200 gshapehe$,44,1                  <jm>
210 gshapehe$,28,1                  <ca>
220 gshapehe$,36,23                 <nl>
230 gshapehe$,44,15                 <ad>
240 gshapehe$,28,15                 <al>
250 gshapene$,17,0                  <eh>
260 gshapene$,56,0                  <dg>
270 sshapen1$,15,0,63,32            <ka>
280 scnclr                          <bk>
290 rem herz neun teil zwei         <fe>
300 gshapehe$,44,1                  <pb>
310 gshapehe$,28,1                  <mf>
320 gshapehe$,44,16                 <ej>
330 gshapehe$,28,16                 <jb>
340 gshapene$,17,17                 <da>
350 gshapene$,57,17                 <pf>
360 sshapen2$,15,0,63,32            <gn>
370 scnclr                          <mo>
380 rem herz vier teil eins         <jb>
390 gshapehe$,44,1                  <cc>
400 gshapehe$,28,1                  <ko>
410 gshapevi$,16,0                  <mb>
420 gshapevi$,53,0                  <pb>
430 sshapev1$,15,0,63,32            <pm>
440 scnclr                          <fl>
450 rem herz vier teil zwei         <ni>
460 gshapehe$,44,16                 <kd>
470 gshapehe$,28,16                 <dk>
480 gshapevi$,16,17                 <ob>
490 gshapevi$,53,17                 <cb>
500 sshapev2$,15,0,63,32            <ag>
510 scnclr                          <oh>
520 rem herz as teil eins           <hf>
530 width2                          <ac>
540 draw1,17,11to21,0to25,11        <fg>
550 draw1,19,6to23,6                <om>
560 draw1,54,11to58,0to62,11        <ga>
570 draw1,56,6to60,6                <gd>
580 width1                          <jd>
590 draw1,34,27to36,32to44,32to46,
27                                  <bf>
600 draw1,34,27to36,25to38,27to40,
25to42,27to44,25to46,27             <ne>
610 sshapea1$,15,0,63,32            <lb>
620 scnclr                          <me>
630 rem herz as teil zwei           <mf>
640 width2                          <hg>
650 draw1,17,28to21,17to25,28       <og>
660 draw1,19,23to23,23              <ha>
670 draw1,54,28to58,17to62,28       <jd>
680 draw1,56,23to60,23              <hj>
690 width1                          <ag>
700 draw1,34,4to40,16to46,4         <lc>
710 circle1,37,4,3,4,270,90         <jj>
720 circle1,43,4,3,4,270,90         <jk>
730 paint1,40,10                    <bf>
740 sshapea2$,15,0,63,32            <pf>
750 scnclr                          <mj>
760 width1                          <dh>
770 rem herz fuenf teil eins        <jk>
780 gshapehe$,44,1                  <lf>
790 gshapehe$,28,1                  <hl>
800 gshapehe$,36,24                 <lb>
810 gshapefu$,20,0                  <nh>
820 gshapefu$,55,0                  <mh>
830 sshapef1$,15,0,63,32            <ki>
840 scnclr                          <ho>
850 rem herz fuenf teil zwei        <jm>
860 gshapehe$,44,16                 <oc>
870 gshapehe$,28,16                 <ok>
880 gshapefu$,20,17                 <hp>
890 gshapefu$,55,17                 <hp>
900 sshapef2$,15,0,63,32            <bo>
910 scnclr                          <al>
920 rem herz sechs teil eins        <gi>
930 gshapehe$,44,1                  <og>
940 gshapehe$,28,1                  <pc>
950 gshapehe$,44,23                 <nc>
960 gshapehe$,28,23                 <cj>
970 gshapese$,17,0                  <bo>
980 gshapese$,57,0                  <kd>
990 sshapes1$,15,0,63,32            <lg>
1000 scnclr                         <lp>
1010 rem herz sechs teil zwei       <mb>
1020 gshapehe$,44,16                <fk>
1030 gshapehe$,28,16                <je>
1040 gshapese$,17,17                <da>
1050 gshapese$,57,17                <pf>
1060 sshapes2$,15,0,63,32           <gn>
1070 scnclr                         <em>
1080 rem herz sieben teil eins      <mb>
1090 gshapehe$,44,1                 <kh>
1100 gshapehe$,28,1                 <cl>
1110 gshapehe$,44,23                <fd>
```

```
1120 gshapehe$,28,23                <pn>
1130 gshapehe$,36,12                <cc>
1140 gshapesi$,18,0                 <mb>
1150 gshapesi$,54,0                 <ko>
1160 sshapec1$,15,0,63,32           <cn>
1170 scnclr                         <bf>
1180 rem herz sieben teil zwei      <bf>
1190 gshapehe$,44,16                <bh>
1200 gshapehe$,28,16                <lp>
1210 gshapesi$,18,17                <ce>
1220 gshapesi$,54,17                <jj>
1230 sshapec2$,15,0,63,32           <dd>
1240 scnclr                         <kb>
1250 rem herz acht teil eins        <lm>
1260 gshapehe$,44,1                 <gl>
1270 gshapehe$,28,1                 <ep>
1280 gshapehe$,44,23                <hh>
1290 gshapehe$,28,23                <ka>
1300 gshapehe$,36,12                <ci>
1310 gshapeac$,17,0                 <lm>
1320 gshapeac$,56,0                 <pm>
1330 sshapee1$,15,0,63,32           <fh>
1340 scnclr                         <gk>
1350 rem herz acht teil zwei        <cm>
1360 gshapehe$,44,16                <di>
1370 gshapehe$,28,16                <oc>
1380 gshapehe$,36,1                 <fk>
1390 gshapeac$,17,17                <pd>
1400 gshapeac$,56,17                <pd>
1410 sshapee2$,15,0,63,32           <po>
1420 scnclr                         <al>
1430 rem herz zehn teil eins        <di>
1440 gshapehe$,44,1                 <og>
1450 gshapehe$,28,1                 <pc>
1460 gshapehe$,44,23                <nc>
1470 gshapehe$,28,23                <cj>
1480 gshapehe$,36,12                <jp>
1490 gshapeze$,15,0                 <po>
1500 gshapeze$,54,0                 <gp>
1510 sshapez1$,15,0,63,32           <jc>
1520 scnclr                         <nd>
1530 rem herz zehn teil zwei        <ee>
1540 gshapehe$,44,1                 <cp>
1550 gshapehe$,28,1                 <hj>
1560 gshapehe$,44,16                <of>
1570 gshapehe$,28,16                <pn>
1580 gshapehe$,36,8                 <og>
1590 gshapeze$,15,17                <ch>
1600 gshapeze$,54,17                <ki>
1610 sshapez2$,15,0,63,32           <fm>
1620 scnclr                         <jm>
1630 rem herz zwei teil eins        <ia>
1640 gshapehe$,36,1                 <hi>
1650 gshapezw$,18,0                 <im>
1660 gshapezw$,57,0                 <lb>
1670 sshapew1$,15,0,63,32           <ka>
1680 scnclr                         <bf>
1690 rem herz zwei teil zwei        <no>
1700 gshapehe$,36,16                <bh>
1710 gshapezw$,18,17                <md>
1720 gshapezw$,57,17                <kd>
1730 sshapew2$,15,0,63,32           <jk>
1740 scnclr                         <in>
1750 rem herz drei teil eins        <ca>
1760 gshapehe$,36,1                 <gh>
1770 gshapehe$,36,23                <al>
1780 gshapedr$,17,0                 <hn>
1790 gshapedr$,56,0                 <hc>
1800 sshaped1$,15,0,63,32           <gj>
1810 scnclr                         <bk>
1820 rem herz drei teil zwei        <jg>
1830 gshapehe$,36,16                <bj>
1840 gshapedr$,17,17                <ce>
1850 gshapedr$,56,17                <ne>
1860 sshaped2$,15,0,63,32           <hp>
1870 scnclr                         <jc>
1880 rem herz koenig teil eins      <ka>
1890 gshapehe$,28,1                 <fk>
1900 width2                         <pn>
1910 draw1,16,0to16,10:draw1,22,0t
o17,4to22,10                        <eb>
1920 draw1,38,12to41,20to59,20to62
,12                                 <ma>
1930 draw1,38,12to42,8to46,12to50,
8to54,12to58,8to62,12               <ao>
1940 char,2,3,"koenig"              <ea>
1950 width1                         <io>
1960 sshapek1$,15,0,63,32           <jc>
1970 scnclr                         <fl>
1980 rem herz koenig teil zwei      <kd>
1990 gshapehe$,44,16                <kd>
2000 width2                         <ca>
2010 draw1,55,17to55,27:draw1,61,1
7to56,21to61,27                     <hm>
2020 draw1,15,12to18,20to36,20to39
,12                                 <aj>
2030 draw1,15,12to19,8to23,12to27,
8to31,12to34,8to38,12               <np>
2040 char,2,0,"koenig"              <oh>
2050 width1                         <lb>
2060 sshapek2$,15,0,63,32           <fm>
2070 scnclr                         <ce>
2080 rem herz dame teil eins        <fb>
2090 gshapehe$,28,1                 <np>
2100 gshapeda$,16,0                 <lp>
2110 char,4,2,"dame"                <oi>
2120 sshapeg1$,15,0,63,32           <gc>
2130 scnclr                         <jm>
2140 rem herz dame teil zwei        <mo>
2150 gshapehe$,44,16                <cg>
2160 gshapeda$,55,17                <nn>
2170 char,2,1,"dame"                <af>
2180 sshapeg2$,15,0,63,32           <lk>
2190 scnclr                         <bf>
2200 rem herz bube teil eins        <jo>
2210 gshapehe$,28,1                 <pa>
```

```
2220 gshapebu$,16,0                    <la>
2230 char,4,2,"bube"                   <bk>
2240 sshapeb1$,15,0,63,32              <oo>
2250 scnclr                            <in>
2260 rem herz bube teil zwei           <bd>
2270 gshapehe$,44,16                   <ae>
2280 gshapebu$,55,17                   <cl>
2290 char,2,1,"bube"                   <nl>
2300 sshapeb2$,15,0,63,32              <fb>
2310 scnclr                            <ag>
2320 rem karo neun teil eins           <lf>
2330 circle,48,5,5,,,,,90              <gm>
2340 circle1,32,5,5,,,,,90             <an>
2350 circle,40,26,5,,,,,90             <kb>
2360 circle,48,19,5,,,,,90             <de>
2370 circle,32,19,5,,,,,90             <oh>
2380 paint1,48,5:paint1,32,5:paint
1,40,28:paint1,48,22:paint1,32,22      <ke>
2390 gshapene$,17,0                    <do>
2400 gshapene$,57,0                    <ic>
2410 sshapen3$,15,0,63,32              <pn>
2420 scnclr                            <oc>
2430 rem karo neun teil zwei           <im>
2440 circle,48,5,5,,,,,90              <la>
2450 circle,32,5,5,,,,,90              <ha>
2460 circle,48,19,5,,,,,90             <io>
2470 circle,32,19,5,,,,,90             <pd>
2480 paint1,48,5:paint1,32,5:paint
1,48,20:paint1,32,20                   <nj>
2490 gshapene$,17,17                   <gc>
2500 gshapene$,57,17                   <mk>
2510 sshapen4$,15,0,63,32              <dh>
2520 scnclr                            <kl>
2530 rem karo vier teil eins           <fe>
2540 circle,48,5,5,,,,,90              <oh>
2550 circle,32,5,5,,,,,90              <pd>
2560 gshapevi$,16,0                    <cg>
2570 gshapevi$,55,0                    <io>
2580 paint1,48,6:paint1,32,6           <fp>
2590 sshapev3$,15,0,63,32              <nj>
2600 scnclr                            <em>
2610 rem karo vier teil zwei           <op>
2620 circle,48,19,5,,,,,90             <gl>
2630 circle,32,19,5,,,,,90             <bg>
2640 paint1,48,20:paint1,32,20         <dn>
2650 gshapevi$,16,17                   <mp>
2660 gshapevi$,55,17                   <ap>
2670 sshapev4$,15,0,63,32              <ga>
2680 scnclr                            <on>
2690 rem karo as teil eins             <mi>
2700 width2                            <bg>
2710 draw1,17,11to21,0to25,11          <dk>
2720 draw1,19,6to23,6                  <gk>
2730 draw1,54,11to58,0to62,11          <lc>
2740 draw1,56,6to60,6                  <fl>
2750 char,4,3,"as"                     <dl>
2760 sshapea3$,15,0,63,32              <md>
2770 scnclr                            <kb>
2780 rem karo as teil zwei             <dj>
2790 draw1,17,28to21,17to25,28         <nf>
2800 draw1,19,23to23,23                <io>
2810 draw1,54,28to58,17to62,28         <hg>
2820 draw1,56,23to60,23                <od>
2830 width1                            <cj>
2840 circle,40,9,9,,,,,90              <bh>
2850 paint1,40,7                       <ca>
2860 sshapea4$,15,0,63,32              <ai>
2870 scnclr                            <gk>
2880 width1                            <ll>
2890 rem karo fuenf teil eins          <dd>
2900 circle,49,5,5,,,,,90              <al>
2910 circle,32,5,5,,,,,90              <fn>
2920 circle,40,26,5,,,,,90             <fm>
2930 paint1,48,6:paint1,32,6:paint
1,40,27                                <nf>
2940 gshapefu$,19,0                    <be>
2950 gshapefu$,56,0                    <km>
2960 sshapef3$,15,0,63,32              <pa>
2970 scnclr                            <dd>
2980 rem karo fuenf teil zwei          <jc>
2990 circle,48,19,5,,,,,90             <hc>
3000 circle,32,19,5,,,,,90             <ap>
3010 paint1,48,20:paint1,32,20         <gf>
3020 gshapefu$,19,17                   <cc>
3030 gshapefu$,56,17                   <nd>
3040 sshapef4$,15,0,63,32              <aj>
3050 scnclr                            <nd>
3060 rem karo sechs teil eins          <ng>
3070 circle,49,5,5,,,,,90              <ae>
3080 circle,32,5,5,,,,,90              <kf>
3090 circle,48,26,5,,,,,90             <gj>
3100 circle,32,26,5,,,,,90             <io>
3110 paint1,48,6:paint1,32,6:paint
1,32,27:paint1,48,27                   <dh>
3120 gshapese$,17,0                    <al>
3130 gshapese$,57,0                    <cp>
3140 sshapes3$,15,0,63,32              <mm>
3150 scnclr                            <jm>
3160 rem karo sechs teil zwei          <kh>
3170 circle,48,19,5,,,,,90             <ld>
3180 circle,32,19,5,,,,,90             <gb>
3190 paint1,48,20:paint1,32,20         <kf>
3200 gshapese$,17,17                   <ie>
3210 gshapese$,57,17                   <fb>
3220 sshapes4$,15,0,63,32              <de>
3230 scnclr                            <dn>
3240 rem karo sieben teil eins         <bn>
3250 circle,48,5,5,,,,,90              <bd>
3260 circle,32,5,5,,,,,90              <ok>
3270 circle,48,26,5,,,,,90             <pn>
3280 circle,32,26,5,,,,,90             <ko>
3290 circle,40,15,5,,,,,90             <mg>
3300 paint1,48,6:paint1,32,6:paint
1,32,27:paint1,48,27:paint1,40,17      <ch>
3310 gshapesi$,17,0                    <ik>
3320 gshapesi$,54,0                    <kj>
```

```
3330 sshapec3$,15,0,63,32          <gj>
3340 scnclr                        <bk>
3350 rem karo sieben teil zwei     <jk>
3360 circle,48,19,5,,,,,90         <aj>
3370 circle,32,19,5,,,,,90         <bl>
3380 paint1,48,20:paint1,32,20     <ag>
3390 gshapesi$,17,17               <in>
3400 gshapesi$,54,17               <ea>
3410 sshapec4$,15,0,63,32          <il>
3420 scnclr                        <lk>
3430 rem karo acht teil eins       <md>
3440 circle,49,5,5,,,,,90          <kg>
3450 circle,32,5,5,,,,,90          <ap>
3460 circle,48,26,5,,,,,90         <oo>
3470 circle,32,26,5,,,,,90         <ge>
3480 circle,40,15,5,,,,,90         <aj>
3490 paint1,48,6:paint1,32,6:paint
1,32,27:paint1,48,27:paint1,40,17  <co>
3500 gshapeac$,17,0                <lf>
3510 gshapeac$,56,0                <jn>
3520 sshapee3$,15,0,63,32          <oj>
3530 scnclr                        <jh>
3540 rem karo acht teil zwei       <jf>
3550 circle,48,19,5,,,,,90         <pe>
3560 circle,32,19,5,,,,,90         <oa>
3570 circle,40,5,5,,,,,90          <pd>
3580 paint1,48,20:paint1,32,20:pai
nt1,40,6                           <me>
3590 gshapeac$,17,17               <fo>
3600 gshapeac$,56,17               <go>
3610 sshapee4$,15,0,63,32          <ap>
3620 scnclr                        <em>
3630 rem karo zehn teil eins       <no>
3640 circle,48,5,5,,,,,90          <op>
3650 circle,32,5,5,,,,,90          <mo>
3660 circle,48,26,5,,,,,90         <km>
3670 circle,32,26,5,,,,,90         <eh>
3680 circle,40,15,5,,,,,90         <ep>
3690 paint1,48,6:paint1,32,6:paint
1,32,27:paint1,48,27:paint1,40,17  <nm>
3700 gshapeze$,15,0                <ba>
3710 gshapeze$,54,0                <db>
3720 sshapez3$,15,0,63,32          <pa>
3730 scnclr                        <cj>
3740 rem karo zehn teil zwei       <ii>
3750 circle,48,5,5,,,,,90          <pc>
3760 circle,32,5,5,,,,,90          <nd>
3770 circle,48,19,5,,,,,90         <pb>
3780 circle,32,19,5,,,,,90         <hm>
3790 circle,40,11,5,,,,,90         <jk>
3800 paint1,48,5:paint1,32,5:paint
1,48,20:paint1,32,20:paint1,40,14  <hg>
3810 gshapeze$,15,17               <jb>
3820 gshapeze$,54,17               <hb>
3830 sshapez4$,15,0,63,32          <pp>
3840 scnclr                        <ag>
3850 rem karo zwei teil eins       <dk>
3860 circle,40,5,5,,,,,90          <cn>
3870 paint1,40,6                   <bp>
3880 gshapezw$,18,0                <em>
3890 gshapezw$,57,0                <de>
3900 sshapew3$,15,0,63,32          <oc>
3910 scnclr                        <jc>
3920 rem karo zwei teil zwei       <jk>
3930 circle,40,19,5,,,,,90         <em>
3940 paint1,40,20                  <lg>
3950 gshapezw$,18,17               <fo>
3960 gshapezw$,57,17               <na>
3970 sshapew4$,15,0,63,32          <dj>
3980 scnclr                        <bp>
3990 rem karo drei teil eins       <cg>
4000 circle,40,5,5,,,,,90          <dk>
4010 circle,40,25,5,,,,,90         <pj>
4020 paint1,40,6:paint1,40,27      <lc>
4030 gshapedr$,17,0                <ok>
4040 gshapedr$,56,0                <el>
4050 sshaped3$,15,0,63,32          <pl>
4060 scnclr                        <lp>
4070 rem karo drei teil zwei       <ej>
4080 circle,40,19,5,,,,,90         <md>
4090 paint1,40,20                  <jc>
4100 gshapedr$,17,17               <ki>
4110 gshapedr$,56,17               <ih>
4120 sshaped4$,15,0,63,32          <cp>
4130 scnclr                        <em>
4140 rem karo koenig teil eins     <ij>
4150 circle,32,5,5,,,,,90          <kk>
4160 paint1,32,6                   <jj>
4170 width2                        <df>
4180 draw1,16,0to16,10:draw1,22,0t
o17,4to22,10                       <ei>
4190 draw1,38,12to41,20to59,20to62
,12                                <bc>
4200 draw1,38,12to42,8to46,12to50,
8to54,12to58,8to62,12              <ej>
4210 char,2,3,"koenig"             <la>
4220 width1                        <mf>
4230 sshapek3$,15,0,63,32          <ah>
4240 scnclr                        <cj>
4250 rem karo koenig teil zwei     <fi>
4260 circle,48,19,5,,,,,90         <kb>
4270 paint1,48,20                  <di>
4280 width2                        <ki>
4290 draw1,55,17to55,27:draw1,61,1
7to56,21to61,27                    <gc>
4300 draw1,15,12to18,20to36,20to39
,12                                <am>
4310 draw1,15,12to19,8to23,12to27,
8to31,12to34,8to38,12              <dc>
4320 char,2,0,"koenig"             <dc>
4330 width1                        <dh>
4340 sshapek4$,15,0,63,32          <mf>
4350 scnclr                        <ae>
4360 rem karo dame teil eins       <dh>
4370 circle,32,5,5,,,,,90          <cn>
4380 paint1,32,6                   <bp>
```

```
4390 gshapeda$,16,0                    <lf>
4400 char,4,2,"dame"                   <no>
4410 sshapeg3$,15,0,63,32              <gn>
4420 scnclr                            <jc>
4430 rem karo dame teil zwei           <jd>
4440 circle,48,19,5,,,,,90             <mn>
4450 paint1,48,20                      <pg>
4460 gshapeda$,56,17                   <oj>
4470 char,2,1,"dame"                   <na>
4480 sshapeg4$,15,0,63,32              <dd>
4490 scnclr                            <bp>
4500 rem karo bube teil eins           <fa>
4510 circle,32,5,5,,,,,90              <dk>
4520 paint1,32,6                       <gm>
4530 gshapebu$,16,0                    <ik>
4540 char,4,2,"bube"                   <me>
4550 sshapeb3$,15,0,63,32              <ih>
4560 scnclr                            <kl>
4570 rem karo bube teil zwei           <ie>
4580 circle,48,19,5,,,,,90             <fl>
4590 paint1,48,20                      <gl>
4600 width2                            <kk>
4610 gshapebu$,55,17                   <fi>
4620 char,2,1,"bube"                   <ej>
4630 sshapeb4$,15,0,63,32              <oo>
4640 scnclr                            <em>
4650 rem kreuz neun teil eins          <hj>
4660 gshapepi$,43,0                    <jh>
4670 gshapepi$,27,0                    <bl>
4680 gshapepi$,35,21                   <fd>
4690 gshapepi$,43,13                   <pj>
4700 gshapepi$,27,13                   <ab>
4710 gshapene$,17,0                    <ea>
4720 gshapene$,57,0                    <dh>
4730 sshapen5$,15,0,63,32              <id>
4740 scnclr                            <bf>
4750 rem kreuz neun teil zwei          <bi>
4760 gshapepi$,43,0                    <om>
4770 gshapepi$,27,0                    <ma>
4780 gshapepi$,43,13                   <ec>
4790 gshapepi$,27,13                   <ig>
4800 gshapene$,17,17                   <cg>
4810 gshapene$,57,17                   <ap>
4820 sshapen6$,15,0,63,32              <op>
4830 scnclr                            <mj>
4840 rem kreuz vier teil eins          <ob>
4850 gshapepi$,43,0                    <cd>
4860 gshapepi$,27,0                    <ch>
4870 gshapevi$,16,0                    <ma>
4880 gshapevi$,56,0                    <pe>
4890 sshapev5$,15,0,63,32              <pd>
4900 scnclr                            <fg>
4910 rem kreuz vier teil zwei          <co>
4920 gshapepi$,43,13                   <ji>
4930 gshapepi$,27,13                   <da>
4940 gshapevi$,16,17                   <jb>
4950 gshapevi$,56,17                   <ee>
4960 sshapev6$,15,0,63,32              <ah>
4970 scnclr                            <oc>
4980 rem kreuz as teil eins            <dg>
4990 width2                            <oo>
5000 draw1,17,11to21,0to25,11          <nb>
5010 draw1,19,6to23,6                  <gf>
5020 draw1,54,11to58,0to62,11          <bp>
5030 draw1,56,6to60,6                  <cl>
5040 char,4,3,"as"                     <ih>
5050 sshapea5$,15,0,63,32              <ei>
5060 scnclr                            <jh>
5070 rem kreuz as teil zwei            <cm>
5080 draw1,17,28to21,17to25,28         <gf>
5090 draw1,19,23to23,23                <mi>
5100 draw1,54,28to58,17to62,28         <fp>
5110 draw1,56,23to60,23                <fe>
5120 width1                            <ab>
5130 draw1,40,3to40,18:draw1,34,11
to46,11                                <bc>
5140 circle,40,3,3,3:circle,34,11,
3,3:circle,46,11,3,3                   <no>
5150 width2                            <pd>
5160 draw1,37,20to40,18to43,20         <me>
5170 paint1,42,3:paint1,34,9:paint
1,48,11                                <ag>
5180 sshapea6$,15,0,63,32              <pc>
5190 scnclr                            <jm>
5200 width1                            <ie>
5210 rem kreuz fuenf teil eins         <bm>
5220 gshapepi$,43,0                    <ek>
5230 gshapepi$,27,0                    <bm>
5240 gshapepi$,35,20                   <jg>
5250 gshapefu$,19,0                    <hm>
5260 gshapefu$,54,0                    <mg>
5270 sshapef5$,15,0,63,32              <ln>
5280 scnclr                            <fb>
5290 rem kreuz fuenf teil zwei         <hg>
5300 gshapepi$,43,13                   <ig>
5310 gshapepi$,27,13                   <co>
5320 gshapefu$,19,17                   <fn>
5330 gshapefu$,54,17                   <pp>
5340 sshapef6$,15,0,63,32              <od>
5350 scnclr                            <nn>
5360 rem kreuz sechs teil eins         <fe>
5370 gshapepi$,43,0                    <ch>
5380 gshapepi$,27,0                    <if>
5390 gshapepi$,43,21                   <pi>
5400 gshapepi$,27,21                   <ba>
5410 gshapese$,17,0                    <ea>
5420 gshapese$,57,0                    <an>
5430 sshapes5$,15,0,63,32              <ed>
5440 scnclr                            <jc>
5450 rem kreuz sechs teil zwei         <ge>
5460 gshapepi$,43,13                   <pn>
5470 gshapepi$,27,13                   <lf>
5480 gshapese$,17,17                   <ec>
5490 gshapese$,57,17                   <ma>
5500 sshapes6$,15,0,63,32              <fm>
5510 scnclr                            <bp>
```

```
5520 rem kreuz sieben teil eins    <fk>
5530 gshapepi$,43,2                <nm>
5540 gshapepi$,27,2                <mi>
5550 gshapepi$,43,19               <ia>
5560 gshapepi$,27,19               <ji>
5570 gshapepi$,35,11               <eh>
5580 gshapesi$,18,0                <ni>
5590 gshapesi$,54,0                <pm>
5600 sshapec5$,15,0,63,32          <pn>
5610 scnclr                        <oh>
5620 rem kreuz sieben teil zwei    <ch>
5630 gshapepi$,43,14               <dj>
5640 gshapepi$,27,14               <mf>
5650 gshapesi$,18,17               <ad>
5660 gshapesi$,54,17               <ml>
5670 sshapec6$,15,0,63,32          <bf>
5680 scnclr                        <he>
5690 rem kreuz acht teil eins      <fo>
5700 gshapepi$,43,0                <ap>
5710 gshapepi$,27,0                <gc>
5720 gshapepi$,43,21               <ab>
5730 gshapepi$,27,21               <mj>
5740 gshapepi$,35,10               <gk>
5750 gshapeac$,17,0                <cd>
5760 gshapeac$,56,0                <jd>
5770 sshapee5$,15,0,63,32          <ji>
5780 scnclr                        <dn>
5790 rem kreuz acht teil zwei      <ld>
5800 gshapepi$,43,14               <ed>
5810 gshapepi$,27,14               <ih>
5820 gshapepi$,35,0                <pm>
5830 gshapeac$,17,17               <dp>
5840 gshapeac$,56,17               <bn>
5850 sshapee6$,15,0,63,32          <ic>
5860 scnclr                        <nn>
5870 rem kreuz zehn teil eins      <in>
5880 gshapepi$,43,0                <ch>
5890 gshapepi$,27,0                <if>
5900 gshapepi$,43,21               <pi>
5910 gshapepi$,27,21               <ba>
5920 gshapepi$,35,10               <mf>
5930 gshapeze$,15,0                <jb>
5940 gshapeze$,54,0                <lj>
5950 sshapez5$,15,0,63,32          <hf>
5960 scnclr                        <kg>
5970 rem kreuz zehn teil zwei      <hh>
5980 gshapepi$,43,0                <ea>
5990 gshapepi$,27,0                <an>
6000 gshapepi$,43,15               <il>
6010 gshapepi$,27,15               <bp>
6020 gshapepi$,35,8                <nk>
6030 gshapeze$,15,17               <oo>
6040 gshapeze$,54,17               <lo>
6050 sshapez6$,15,0,63,32          <ll>
6060 scnclr                        <gp>
6070 rem kreuz zwei teil eins      <ko>
6080 gshapepi$,35,0                <ak>
6090 gshapezw$,18,0                <nd>
6100 gshapezw$,57,0                <pd>
6110 sshapew5$,15,0,63,32          <hh>
6120 scnclr                        <oh>
6130 rem kreuz zwei teil zwei      <hd>
6140 gshapepi$,35,19               <do>
6150 gshapezw$,18,17               <on>
6160 gshapezw$,57,17               <ek>
6170 sshapew6$,15,0,63,32          <ih>
6180 scnclr                        <ga>
6190 rem kreuz drei teil eins      <hp>
6200 gshapepi$,35,0                <bl>
6210 gshapepi$,35,21               <fd>
6220 gshapedr$,17,0                <je>
6230 gshapedr$,56,0                <me>
6240 sshaped5$,15,0,63,32          <nh>
6250 scnclr                        <on>
6260 rem kreuz drei teil zwei      <ec>
6270 gshapepi$,35,13               <cb>
6280 gshapedr$,17,17               <dc>
6290 gshapedr$,56,17               <ld>
6300 sshaped6$,15,0,63,32          <em>
6310 scnclr                        <gf>
6320 rem kreuz koenig teil eins    <cb>
6330 gshapepi$,27,0                <pm>
6340 width2                        <ej>
6350 draw1,16,0to16,10:draw1,22,0t
o17,4to22,10                       <ld>
6360 draw1,38,12to41,20to59,20to62
,12                                <fg>
6370 draw1,38,12to42,8to46,12to50,
8to54,12to58,8to62,12              <ok>
6380 char,2,3,"koenig"             <mo>
6390 width1                        <nb>
6400 sshapek5$,15,0,63,32          <hf>
6410 scnclr                        <co>
6420 rem kreuz koenig teil zwei    <ad>
6430 gshapepi$,43,13               <mi>
6440 width2                        <gm>
6450 draw1,55,17to55,27:draw1,61,1
7to56,21to61,27                    <ki>
6460 draw1,15,12to18,20to36,20to39
,12                                <ao>
6470 draw1,15,12to19,8to23,12to27,
8to31,12to34,8to38,12              <pm>
6480 char,2,0,"koenig"             <bb>
6490 width1                        <pm>
6500 sshapek6$,15,0,63,32          <ll>
6510 scnclr                        <ph>
6520 rem kreuz dame teil eins      <ak>
6530 gshapepi$,27,0                <nc>
6540 gshapeda$,16,0                <mk>
6550 char,4,2,"dame"               <nh>
6560 sshapeg5$,15,0,63,32          <bg>
6570 scnclr                        <gp>
6580 rem kreuz dame teil zwei      <hi>
6590 gshapepi$,43,13               <ej>
6600 gshapeda$,55,17               <oh>
6610 char,2,0,"dame"               <bf>
```

```
6620 sshapeg6$,15,0,63,32          <gh>
6630 scnclr                        <oh>
6640 rem kreuz bube teil eins      <kg>
6650 gshapepi$,27,0                <ib>
6660 gshapebu$,16,0                <mh>
6670 char,4,2,"bube"               <kd>
6680 sshapeb5$,15,0,63,32          <pd>
6690 scnclr                        <ga>
6700 rem kreuz bube teil zwei      <dd>
6710 gshapepi$,43,13               <gl>
6720 gshapebu$,55,17               <gp>
6730 char,2,0,"bube"               <fo>
6740 sshapeb6$,15,0,63,32          <cm>
6750 scnclr                        <ni>
6760 rem pik neun teil eins        <dc>
6770 gshapekr$,44,1                <pf>
6780 gshapekr$,28,1                <dl>
6790 gshapekr$,36,22               <dd>
6800 gshapekr$,44,15               <ji>
6810 gshapekr$,28,15               <ee>
6820 gshapene$,17,0                <ie>
6830 gshapene$,57,0                <mc>
6840 sshapen7$,15,0,63,32          <on>
6850 scnclr                        <kb>
6860 rem pik neun teil zwei        <dk>
6870 gshapekr$,44,1                <bn>
6880 gshapekr$,28,1                <bb>
6890 gshapekr$,44,16               <ah>
6900 gshapekr$,28,16               <ap>
6910 gshapene$,17,17               <bl>
6920 gshapene$,57,17               <ci>
6930 sshapen8$,15,0,63,32          <cd>
6940 scnclr                        <fg>
6950 rem pik vier teil eins        <go>
6960 gshapekr$,44,1                <hg>
6970 gshapekr$,28,1                <mc>
6980 gshapevi$,16,0                <cm>
6990 gshapevi$,53,0                <hj>
7000 sshapev7$,15,0,63,32          <dg>
7010 scnclr                        <oc>
7020 rem pik vier teil zwei        <ko>
7030 gshapekr$,44,16               <ki>
7040 gshapekr$,28,16               <oa>
7050 gshapevi$,16,17               <pe>
7060 gshapevi$,53,17               <pe>
7070 sshapev8$,15,0,63,32          <ne>
7080 scnclr                        <gp>
7090 rem pik as teil eins          <pf>
7100 width2                        <ca>
7110 draw1,17,11to21,0to25,11      <gl>
7120 draw1,19,6to23,6              <lj>
7130 draw1,54,11to58,0to62,11      <ie>
7140 draw1,56,6to60,6              <ak>
7150 width1                        <kj>
7160 char,4,3,"as"                 <eb>
7170 sshapea7$,15,0,63,32          <pb>
7180 scnclr                        <di>
7190 rem pik as teil zwei          <an>
7200 width2                        <ec>
7210 draw1,17,28to21,17to25,28     <il>
7220 draw1,19,23to23,23            <of>
7230 draw1,54,28to58,17to62,28     <fg>
7240 draw1,56,23to60,23            <bo>
7250 width1                        <ne>
7260 draw1,34,11to40,0to46,12      <fm>
7270 circle1,37,12,3,4,90,270      <ap>
7280 circle1,43,12,3,4,90,270      <ge>
7290 paint1,40,10                  <np>
7300 draw1,40,18to40,20            <ld>
7310 draw1,38,22to40,20to42,22     <gj>
7320 sshapea8$,15,0,63,32          <an>
7330 scnclr                        <gf>
7340 width1                        <kh>
7350 rem pik fuenf teil eins       <lc>
7360 gshapekr$,44,1                <bf>
7370 gshapekr$,28,1                <bj>
7380 gshapekr$,36,22               <go>
7390 gshapefu$,19,0                <ep>
7400 gshapefu$,54,0                <cn>
7410 sshapef7$,15,0,63,32          <ed>
7420 scnclr                        <bi>
7430 rem pik fuenf teil zwei       <fj>
7440 gshapekr$,44,16               <jl>
7450 gshapekr$,28,16               <dc>
7460 gshapefu$,19,17               <cp>
7470 gshapefu$,54,17               <la>
7480 sshapef8$,15,0,63,32          <dj>
7490 scnclr                        <kg>
7500 rem pik sechs teil eins       <be>
7510 gshapekr$,44,1                <ag>
7520 gshapekr$,28,1                <nc>
7530 gshapekr$,44,22               <bd>
7540 gshapekr$,28,22               <kg>
7550 gshapese$,17,0                <el>
7560 gshapese$,57,0                <gp>
7570 sshapes7$,15,0,63,32          <pm>
7580 scnclr                        <fl>
7590 rem pik sechs teil zwei       <eo>
7600 gshapekr$,44,16               <bi>
7610 gshapekr$,28,16               <nf>
7620 gshapese$,17,17               <oi>
7630 gshapese$,57,17               <ep>
7640 sshapes8$,15,0,63,32          <ag>
7650 scnclr                        <oh>
7660 rem pik sieben teil eins      <dj>
7670 gshapekr$,44,1                <ec>
7680 gshapekr$,28,1                <im>
7690 gshapekr$,44,22               <jf>
7700 gshapekr$,28,22               <dn>
7710 gshapekr$,36,12               <oc>
7720 gshapesi$,18,0                <ba>
7730 gshapesi$,54,0                <kb>
7740 sshapec7$,15,0,63,32          <cg>
7750 scnclr                        <la>
7760 rem pik sieben teil zwei      <kj>
7770 gshapekr$,44,16               <fg>
```

```
7780 gshapekr$,28,16                <pj>
7790 gshapesi$,18,17                <io>
7800 gshapesi$,54,17                <gd>
7810 sshapec8$,15,0,63,32           <ek>
7820 scnclr                         <dn>
7830 rem pik acht teil eins         <kh>
7840 gshapekr$,44,1                 <el>
7850 gshapekr$,28,1                 <gp>
7860 gshapekr$,44,22                <lh>
7870 gshapekr$,28,22                <fo>
7880 gshapekr$,36,12                <ge>
7890 gshapeac$,17,0                 <gk>
7900 gshapeac$,56,0                 <ek>
7910 sshapee7$,15,0,63,32           <ge>
7920 scnclr                         <ae>
7930 rem pik acht teil zwei         <bg>
7940 gshapekr$,44,16                <id>
7950 gshapekr$,28,16                <jl>
7960 gshapekr$,36,1                 <ho>
7970 gshapeac$,17,17                <ek>
7980 gshapeac$,56,17                <ik>
7990 sshapee8$,15,0,63,32           <bn>
8000 scnclr                         <kg>
8010 rem pik zehn teil eins         <km>
8020 gshapekr$,44,1                 <ag>
8030 gshapekr$,28,1                 <nc>
8040 gshapekr$,44,22                <bd>
8050 gshapekr$,28,22                <kg>
8060 gshapekr$,36,12                <nm>
8070 gshapeze$,15,0                 <nk>
8080 gshapeze$,54,0                 <mc>
8090 sshapez7$,15,0,63,32           <kl>
8100 scnclr                         <gp>
8110 rem pik zehn teil zwei         <ok>
8120 gshapekr$,44,1                 <mp>
8130 gshapekr$,28,1                 <cd>
8140 gshapekr$,44,16                <kg>
8150 gshapekr$,28,16                <cn>
8160 gshapekr$,36,8                 <mg>
8170 gshapeze$,15,17                <hl>
8180 gshapeze$,54,17                <em>
8190 sshapez8$,15,0,63,32           <ib>
8200 scnclr                         <di>
8210 rem pik zwei teil eins         <en>
8220 gshapekr$,36,1                 <fi>
8230 gshapezw$,18,0                 <cl>
8240 gshapezw$,57,0                 <ji>
8250 sshapew7$,15,0,63,32           <fl>
8260 scnclr                         <la>
8270 rem pik zwei teil zwei         <cn>
8280 gshapekr$,36,16                <fg>
8290 gshapezw$,18,17                <ge>
8300 gshapezw$,57,17                <je>
8310 sshapew8$,15,0,63,32           <oa>
8320 scnclr                         <cj>
8330 rem pik drei teil eins         <lh>
8340 gshapekr$,36,1                 <ih>
8350 gshapekr$,36,22                <mk>
8360 gshapedr$,17,0                 <mm>
8370 gshapedr$,56,0                 <gj>
8380 sshaped7$,15,0,63,32           <ia>
8390 scnclr                         <lf>
8400 rem pik drei teil zwei         <id>
8410 gshapekr$,36,16                <fi>
8420 gshapedr$,17,17                <of>
8430 gshapedr$,57,17                <ek>
8440 sshaped8$,15,0,63,32           <nh>
8450 scnclr                         <co>
8460 rem pik koenig teil eins       <hh>
8470 gshapekr$,28,1                 <ho>
8480 width2                         <gm>
8490 draw1,16,0to16,10:draw1,22,0t
o17,4to22,10                        <kb>
8500 draw1,38,12to41,20to59,20to62
,12                                 <cj>
8510 draw1,38,12to42,8to46,12to50,
8to54,12to58,8to62,12               <gm>
8520 char,2,3,"koenig"              <ap>
8530 width1                         <pm>
8540 sshapek7$,15,0,63,32           <kl>
8550 scnclr                         <ph>
8560 rem pik koenig teil zwei       <of>
8570 gshapekr$,44,16                <oa>
8580 width2                         <ip>
8590 draw1,55,17to55,27:draw1,61,1
7to56,21to61,27                     <hp>
8600 draw1,15,12to18,20to36,20to39
,12                                 <of>
8610 draw1,15,12to19,8to23,12to27,
8to31,12to34,8to38,12               <bn>
8620 char,2,0,"koenig"              <ao>
8630 width1                         <bp>
8640 sshapek8$,15,0,63,32           <ab>
8650 scnclr                         <lp>
8660 rem pik dame teil eins         <an>
8670 gshapekr$,28,1                 <lp>
8680 gshapeda$,16,0                 <bb>
8690 char,4,2,"dame"                <hj>
8700 sshapeg7$,15,0,63,32           <df>
8710 scnclr                         <di>
8720 rem pik dame teil zwei         <bl>
8730 gshapekr$,44,16                <og>
8740 gshapeda$,55,17                <ji>
8750 char,2,1,"dame"                <ej>
8760 sshapeg8$,15,0,63,32           <if>
8770 scnclr                         <la>
8780 rem pik bube teil eins         <pe>
8790 gshapekr$,28,1                 <ba>
8800 gshapebu$,16,0                 <om>
8810 char,4,2,"bube"                <pi>
8820 sshapeb7$,15,0,63,32           <ai>
8830 scnclr                         <cj>
8840 rem pik bube teil zwei         <ih>
8850 gshapekr$,44,16                <ef>
8860 gshapebu$,55,17                <hk>
8870 char,2,1,"bube"                <db>
```

```
8880 sshapeb8$,15,0,63,32                <hd>
8890 scnclr                              <kb>
8900 rem sshape halten                   <bh>
8910 graphic1,1:circle1,12,11,8,8        <lc>
8920 sshapeha$,4,3,20,20                 <om>
8930 scnclr                              <pa>
8940 gosub14220                          <eh>
8950 rem sshape ende                     <hm>
8960 slow:poke53265,27                   <mf>
8970 graphic0,1                          <al>
8980 color0,2:color1,1:color4,2:co
lor5,1                                   <dn>
8990 print:print:print:print"kenne
n sie die regeln  (j/n)":inputx$         <ip>
9000 ifx$<>"j"andx$<>"n"thensleep1
:goto8970                                <cg>
9010 ifx$="n"thengosub12440              <bm>
9020 poke53265,0:fast                    <pj>
9030 graphic1,1                          <nd>
9040 rem *****aufbau des spielfeld
es **                                    <kd>
9050 color0,2:color1,1                   <de>
9060 box1,8,90,66,175                    <hj>
9070 box1,68,90,126,175                  <bd>
9080 box1,128,90,186,175                 <ol>
9090 box1,188,90,246,175                 <oc>
9100 box1,248,90,306,175                 <jg>
9110 gshapeha$,27,179                    <ho>
9120 gshapeha$,92,179                    <bn>
9130 gshapeha$,148,179                   <nb>
9140 gshapeha$,212,179                   <op>
9150 gshapeha$,268,179                   <bg>
9160 char,4,23,"1"                       <gg>
9170 char,12,23,"2"                      <kc>
9180 char,19,23,"3"                      <jl>
9190 char,27,23,"4"                      <mi>
9200 char,34,23,"5"                      <gg>
9210 char,1,3,"gewinn"                   <aj>
9220 char,1,5," 00000"                   <bo>
9230 char,25,8,"x2"                      <lp>
9240 char,22,7,"x4"                      <de>
9250 char,19,6,"x8"                      <gl>
9260 char,15,5,"x16"                     <gf>
9270 char,2,9,"geben"                    <ln>
9280 char,29,7,"risiko"                  <pd>
9290 char,31,9,"000"                     <pn>
9300 char,11,4,"x32"                     <kd>
9310 char,30,5,"vorrat"                  <lm>
9320 circle1,299,75,10,10                <pg>
9330 char,37,9,"0"                       <cl>
9340 char,10,9,"halten"                  <nh>
9350 char,25,3,":4"                      <ha>
9360 char,19,3,"000  "                   <nf>
9370 color1,5:width2                     <gd>
9380 box1,230,66,285,82                  <kp>
9390 circle,37,75,29,8                   <ng>
9400 circle,103,75,35,8                  <dj>
9410 box1,8,52,120,65                    <nf>
9420 box1,8,34,63,50                     <ge>
9430 draw1,230,79to220,79to73,37to
63,37                                    <dm>
9440 box1,235,20,290,36                  <cj>
9450 box1,135,20,195,36                  <eo>
9460 color1,4                            <aj>
9470 draw1,63,34to100,25to135,25         <nc>
9480 draw1,125,19to135,25to125,31        <jl>
9490 draw1,195,35to235,35                <np>
9500 draw1,225,29to235,35to225,41        <am>
9510 width1                              <ll>
9520 slow:poke53265,27                   <go>
9530 vo=150                              <gc>
9540 color1,1:play"c":char,32,3,"1
":play"c":char,33,3,"5":play"c":ch
ar,34,3,"0"                              <oa>
9550 a$=" full house (c)1987 by si
egfried ederer"                          <ib>
9560 fori=1to40:char,1,1,a$:forx=1
to10:nextx                               <gg>
9570 a$=right$(a$,len(a$)-1)+left$
(a$,1):nexti:goto9650                    <ii>
9580 rem ****** blinken von geben*
**                                       <ih>
9590 color1,8:char,2,9,"geben":for
i=1to20:nexti:color1,1:char,2,9,"g
eben":fori=1to10:nexti                   <li>
9600 geta$:ifa$=chr$(32)thenreturn       <hk>
9610 goto9590                            <mh>
9620 rem ****** blinken von halten
 ***                                     <nb>
9630 color1,8:char,10,9,"halten":f
ori=1to20:nexti:color1,1:char,10,
9,"halten":fori=1to10:nexti              <mo>
9640 color1,8:char,2,9,"geben":for
i=1to20:nexti:color1,1:char,2,9,"g
eben":fori=1to10:nexti:return            <ee>
9650 z6=0:gosub9590                      <im>
9660 gosub12300                          <al>
9670 rem ******* ziehen der karten
 **                                      <an>
9680 aa(5)=54:aa(7)=55:aa(9)=56:z6
=0                                       <kc>
9690 fori=1to9step2                      <gb>
9700 aa(i)=int(rnd(x)*52)+1              <cl>
9710 ifaa(3)=aa(1)then9700               <nj>
9720 ifaa(5)=aa(1)oraa(5)=aa(3)the
n9700                                    <dh>
9730 ifaa(7)=aa(1)oraa(7)=aa(3)ora
a(7)=aa(5)then9700                       <kc>
9740 ifaa(9)=aa(1)oraa(9)=aa(3)ora
a(9)=aa(5)oraa(9)=aa(7)then9700          <oe>
9750 ifaa(i)=1theny$=w1$:x$=w2$          <il>
9760 ifaa(i)=2theny$=d1$:x$=d2$          <kn>
9770 ifaa(i)=3theny$=v1$:x$=v2$          <oe>
9780 ifaa(i)=4theny$=f1$:x$=f2$          <ad>
9790 ifaa(i)=5theny$=s1$:x$=s2$          <gb>
9800 ifaa(i)=6theny$=c1$:x$=c2$          <ng>
```

```
9810 ifaa(i)=7theny$=e1$:x$=e2$     <ag>
9820 ifaa(i)=8theny$=n1$:x$=n2$     <eo>
9830 ifaa(i)=9theny$=z1$:x$=z2$     <ai>
9840 ifaa(i)=10theny$=b1$:x$=b2$    <pk>
9850 ifaa(i)=11theny$=g1$:x$=g2$    <lk>
9860 ifaa(i)=12theny$=k1$:x$=k2$    <cl>
9870 ifaa(i)=13theny$=a1$:x$=a2$    <ch>
9880 ifaa(i)=14theny$=w3$:x$=w4$    <ld>
9890 ifaa(i)=15theny$=d3$:x$=d4$    <ah>
9900 ifaa(i)=16theny$=v3$:x$=v4$    <ma>
9910 ifaa(i)=17theny$=f3$:x$=f4$    <ak>
9920 ifaa(i)=18theny$=s3$:x$=s4$    <cm>
9930 ifaa(i)=19theny$=c3$:x$=c4$    <ef>
9940 ifaa(i)=20theny$=e3$:x$=e4$    <hb>
9950 ifaa(i)=21theny$=n3$:x$=n4$    <kh>
9960 ifaa(i)=22theny$=z3$:x$=z4$    <kb>
9970 ifaa(i)=23theny$=b3$:x$=b4$    <ik>
9980 ifaa(i)=24theny$=g3$:x$=g4$    <on>
9990 ifaa(i)=25theny$=k3$:x$=k4$    <df>
10000 ifaa(i)=26theny$=a3$:x$=a4$   <mi>
10010 ifaa(i)=27theny$=w5$:x$=w6$   <dj>
10020 ifaa(i)=28theny$=d5$:x$=d6$   <jd>
10030 ifaa(i)=29theny$=v5$:x$=v6$   <mc>
10040 ifaa(i)=30theny$=f5$:x$=f6$   <cc>
10050 ifaa(i)=31theny$=s5$:x$=s6$   <nf>
10060 ifaa(i)=32theny$=c5$:x$=c6$   <lm>
10070 ifaa(i)=33theny$=e5$:x$=e6$   <kn>
10080 ifaa(i)=34theny$=n5$:x$=n6$   <ck>
10090 ifaa(i)=35theny$=z5$:x$=z6$   <gc>
10100 ifaa(i)=36theny$=b5$:x$=b6$   <kh>
10110 ifaa(i)=37theny$=g5$:x$=g6$   <oo>
10120 ifaa(i)=38theny$=k5$:x$=k6$   <bk>
10130 ifaa(i)=39theny$=a5$:x$=a6$   <cf>
10140 ifaa(i)=40theny$=w7$:x$=w8$   <km>
10150 ifaa(i)=41theny$=d7$:x$=d8$   <ij>
10160 ifaa(i)=42theny$=v7$:x$=v8$   <hi>
10170 ifaa(i)=43theny$=f7$:x$=f8$   <ak>
10180 ifaa(i)=44theny$=s7$:x$=s8$   <kd>
10190 ifaa(i)=45theny$=c7$:x$=c8$   <ea>
10200 ifaa(i)=46theny$=e7$:x$=e8$   <db>
10210 ifaa(i)=47theny$=n7$:x$=n8$   <kp>
10220 ifaa(i)=48theny$=z7$:x$=z8$   <eo>
10230 ifaa(i)=49theny$=b7$:x$=b8$   <jp>
10240 ifaa(i)=50theny$=g7$:x$=g8$   <hn>
10250 ifaa(i)=51theny$=k7$:x$=k8$   <lm>
10260 ifaa(i)=52theny$=a7$:x$=a8$   <an>
10270 ifz6=1then return             <kc>
10280 ifi=1theny1$=y$:y2$=x$:gosub
10330:goto10350                     <pb>
10290 ifi=3theny3$=y$:y4$=x$:gosub
10330:goto10370                     <lo>
10300 ifi=5theny5$=y$:y6$=x$:gosub
10330:goto10390                     <kd>
10310 ifi=7theny7$=y$:y8$=x$:gosub
10330:goto10410                     <bm>
10320 ifi=9theny9$=y$:y10$=x$:gosu
b10330:goto10430                    <dl>
10330 ifaa(i)>=1andaa(i)<=26thenco
lor1,3:return                       <kj>
10340 color1,1:return               <dl>
10350 gshapey1$,13,100              <ap>
10360 gshapey2$,13,140:nexti        <ia>
10370 gshapey3$,73,100              <la>
10380 gshapey4$,73,140:nexti        <hf>
10390 gshapey5$,133,100             <ca>
10400 gshapey6$,133,140:nexti       <ci>
10410 gshapey7$,193,100             <if>
10420 gshapey8$,193,140:nexti       <pn>
10430 gshapey9$,253,100             <la>
10440 gshapey10$,253,140:nexti      <np>
10450 gosub9630:geta$               <ba>
10460 ifa$="1"andz1=0thencolor1,8:
gshapeha$,27,179:z1=1:goto10450     <mn>
10470 ifa$="2"andz2=0thencolor1,8:
gshapeha$,92,179:z2=1:goto10450     <gm>
10480 ifa$="3"andz3=0thencolor1,8:
gshapeha$,148,179:z3=1:goto10450    <pg>
10490 ifa$="4"andz4=0thencolor1,8:
gshapeha$,212,179:z4=1:goto10450    <bm>
10500 ifa$="5"andz5=0thencolor1,8:
gshapeha$,268,179:z5=1:goto10450    <pc>
10510 ifa$="1"andz1=1thencolor1,1:
gshapeha$,27,179:z1=0               <md>
10520 ifa$="2"andz2=1thencolor1,1:
gshapeha$,92,179:z2=0               <ko>
10530 ifa$="3"andz3=1thencolor1,1:
gshapeha$,148,179:z3=0              <eo>
10540 ifa$="4"andz4=1thencolor1,1:
gshapeha$,212,179:z4=0              <pc>
10550 ifa$="5"andz5=1thencolor1,1:
gshapeha$,268,179:z5=0              <dc>
10560 ifa$=chr$(32)then10580        <ph>
10570 goto10450                     <nn>
10580 z6=1                          <ap>
10590 ifz1=0theni=1:aa(1)=int(rnd(
x)*52)+1:gosub10650                 <kn>
10600 ifz2=0theni=3:aa(3)=int(rnd(
x)*52)+1:gosub10660                 <ab>
10610 ifz3=0theni=5:aa(5)=int(rnd(
x)*52)+1:gosub10670                 <mh>
10620 ifz4=0theni=7:aa(7)=int(rnd(
x)*52)+1:gosub10680                 <fd>
10630 ifz5=0theni=9:aa(9)=int(rnd(
x)*52)+1:gosub10690                 <ch>
10640 goto10890                     <kh>
10650 ifaa(1)=aa(3)oraa(1)=aa(5)or
aa(1)=aa(7)oraa(1)=aa(9)then10590:
elsegosub9750:gosub10850:gosub1080
0:gosub10700:return                 <pc>
10660 ifaa(3)=aa(1)oraa(3)=aa(5)or
aa(3)=aa(7)oraa(3)=aa(9)then10600:
elsegosub9750:gosub10850:gosub1081
0:gosub10720:return                 <fc>
10670 ifaa(5)=aa(1)oraa(5)=aa(3)or
aa(5)=aa(7)oraa(5)=aa(9)then10610:
elsegosub9750:gosub10850:gosub1082
```

```
0:gosub10740:return                                    <dp>
10680 ifaa(7)=aa(1)oraa(7)=aa(3)or
aa(7)=aa(5)oraa(7)=aa(9)then10620:
elsegosub9750:gosub10850:gosub1083
0:gosub10760:return                                    <fp>
10690 ifaa(9)=aa(1)oraa(9)=aa(3)or
aa(9)=aa(5)oraa(9)=aa(7)then10630:
elsegosub9750:gosub10850:gosub1084
0:gosub10780:return                                    <bb>
10700 gshapey1$,13,100                                 <ck>
10710 gshapey2$,13,140:return                          <ik>
10720 gshapey3$,73,100                                 <jj>
10730 gshapey4$,73,140:return                          <ma>
10740 gshapey5$,133,100                                <gp>
10750 gshapey6$,133,140:return                         <oj>
10760 gshapey7$,193,100                                <kd>
10770 gshapey8$,193,140:return                         <cl>
10780 gshapey9$,253,100                                <bn>
10790 gshapey10$,253,140:return                        <pg>
10800 ifi=1theny1$=y$:y2$=x$:retur
n                                                      <em>
10810 ifi=3theny3$=y$:y4$=x$:retur
n                                                      <jk>
10820 ifi=5theny5$=y$:y6$=x$:retur
n                                                      <cb>
10830 ifi=7theny7$=y$:y8$=x$:retur
n                                                      <mp>
10840 ifi=9theny9$=y$:y10$=x$:retu
rn                                                     <bi>
10850 ifaa(i)>=1andaa(i)<=26thenco
lor1,3:return                                          <ah>
10860 color1,1:return                                  <lm>
10870 rem gewinnermittlung                             <jl>
10880 rem farbenermittlung                             <cl>
10890 fori=1to9step2                                   <go>
10900 ifaa(i)<14thenfa(i)=1:rem he
rz                                                     <en>
10910 ifaa(i)>13andaa(i)<27thenfa(
i)=2:rem karo                                          <oi>
10920 ifaa(i)>26andaa(i)<40thenfa(
i)=3:rem kreuz                                         <mj>
10930 ifaa(i)>39thenfa(i)=4:rem bl
att                                                    <pa>
10940 nexti                                            <oh>
10950 rem **                                           <do>
10960 ca(1)=aa(1):ca(2)=aa(3):ca(3
)=aa(5):ca(4)=aa(7):ca(5)=aa(9)                        <ei>
10970 fori=1to4                                        <jb>
10980 fory=i+1to5                                      <ji>
10990 ifca(i)<=ca(y)then11030                          <fg>
11000 letca(0)=ca(i)                                   <em>
11010 letca(i)=ca(y)                                   <jk>
11020 letca(y)=ca(0)                                   <ka>
11030 nexty                                            <gb>
11040 nexti                                            <hj>
11050 ifca(1)=ca(2)-13thenc=c+1                        <lf>
11060 ifca(1)=ca(2)-26thenc=c+1                        <dp>
11070 ifca(1)=ca(2)-39thenc=c+1                        <ln>
11080 ifca(1)=ca(3)-13thenc=c+1                        <ib>
11090 ifca(1)=ca(3)-26thenc=c+1                        <ml>
11100 ifca(1)=ca(3)-39thenc=c+1                        <on>
11110 ifca(1)=ca(4)-13thenc=c+1                        <kf>
11120 ifca(1)=ca(4)-26thenc=c+1                        <cd>
11130 ifca(1)=ca(4)-39thenc=c+1                        <jb>
11140 ifca(1)=ca(5)-13thenc=c+1                        <gi>
11150 ifca(1)=ca(5)-26thenc=c+1                        <od>
11160 ifca(1)=ca(5)-39thenc=c+1                        <pn>
11170 ifca(2)=ca(3)-13thend=d+1                        <mc>
11180 ifca(2)=ca(3)-26thend=d+1                        <ln>
11190 ifca(2)=ca(3)-39thend=d+1                        <dh>
11200 ifca(2)=ca(4)-13thend=d+1                        <on>
11210 ifca(2)=ca(4)-26thend=d+1                        <gp>
11220 ifca(2)=ca(4)-39thend=d+1                        <ph>
11230 ifca(2)=ca(5)-13thend=d+1                        <ef>
11240 ifca(2)=ca(5)-26thend=d+1                        <nd>
11250 ifca(2)=ca(5)-39thend=d+1                        <fc>
11260 ifca(3)=ca(4)-13thene=e+1                        <bk>
11270 ifca(3)=ca(4)-26thene=e+1                        <jb>
11280 ifca(3)=ca(4)-39thene=e+1                        <hp>
11290 ifca(3)=ca(5)-13thene=e+1                        <db>
11300 ifca(3)=ca(5)-26thene=e+1                        <ml>
11310 ifca(3)=ca(5)-39thene=e+1                        <fh>
11320 ifca(4)=ca(5)-13thenf=f+1                        <bm>
11330 ifca(4)=ca(5)-26thenf=f+1                        <ig>
11340 ifca(4)=ca(5)-39thenf=f+1                        <bo>
11350 rem vier gleiche                                 <jh>
11360 ifc=3ord=3thenl=7:goto12060                      <bp>
11370 rem full house                                   <cg>
11380 ifc=2andd=1ande=1thenl=4:got
o12060                                                 <ii>
11390 ifc=2andd=1andf=1thenl=4:got
o12060                                                 <fh>
11400 ifc=2ande=1andf=1thenl=4:got
o12060                                                 <fi>
11410 ifd=2andc=1ande=1thenl=4:got
o12060                                                 <hl>
11420 ifd=2andc=1andf=1thenl=4:got
o12060                                                 <fg>
11430 ifd=2ande=1andf=1thenl=4:got
o12060                                                 <hd>
11440 ife=2andc=1andd=1thenl=4:got
o12060                                                 <el>
11450 ife=2andc=1andf=1thenl=4:got
o12060                                                 <ci>
11460 ife=2andd=1andf=1thenl=4:got
o12060                                                 <cb>
11470 rem drei gleiche                                 <in>
11480 ifc=2ord=2ore=2thenl=3:goto1
2060                                                   <oo>
11490 rem zwei paare                                   <nd>
11500 ifc=1andd=1thenl=2:goto12060                     <ec>
11510 ifc=1ande=1thenl=2:goto12060                     <dc>
11520 ifc=1andf=1thenl=2:goto12060                     <pl>
11530 ifd=1ande=1thenl=2:goto12060                     <dn>
11540 ifd=1andf=1thenl=2:goto12060                     <io>
11550 ife=1andf=1thenl=2:goto12060                     <bc>
```

```
11560 rem ein paar                          <fg>
11570 ifc=1ord=1ore=1orf=1then1=1:
goto12060                                   <pj>
11580 rem 5 in einer reihe vers. f
arbe                                        <he>
11590 fori=1to5                             <ei>
11600 ifca(i)=1orca(i)=14orca(i)=2
7orca(i)=40thenca(i)=1                      <ik>
11610 ifca(i)=2orca(i)=15orca(i)=2
8orca(i)=41thenca(i)=2                      <dh>
11620 ifca(i)=3orca(i)=16orca(i)=2
9orca(i)=42thenca(i)=3                      <cb>
11630 ifca(i)=4orca(i)=17orca(i)=3
0orca(i)=43thenca(i)=4                      <bh>
11640 ifca(i)=5orca(i)=18orca(i)=3
1orca(i)=44thenca(i)=5                      <ie>
11650 ifca(i)=6orca(i)=19orca(i)=3
2orca(i)=45thenca(i)=6                      <ik>
11660 ifca(i)=7orca(i)=20orca(i)=3
3orca(i)=46thenca(i)=7                      <ai>
11670 ifca(i)=8orca(i)=21orca(i)=3
4orca(i)=47thenca(i)=8                      <pn>
11680 ifca(i)=9orca(i)=22orca(i)=3
5orca(i)=48thenca(i)=9                      <gb>
11690 ifca(i)=10orca(i)=23orca(i)=
36orca(i)=49thenca(i)=10                    <mb>
11700 ifca(i)=11orca(i)=24orca(i)=
37orca(i)=50thenca(i)=11                    <hb>
11710 ifca(i)=12orca(i)=25orca(i)=
38orca(i)=51thenca(i)=12                    <di>
11720 ifca(i)=13orca(i)=26orca(i)=
39orca(i)=52thenca(i)=13                    <cl>
11730 nexti                                 <em>
11740 fori=1to4                             <ne>
11750 fory=i+1to5                           <id>
11760 ifca(i)<=ca(y)then11800               <gi>
11770 letca(0)=ca(i)                        <bo>
11780 letca(i)=ca(y)                        <mh>
11790 letca(y)=ca(0)                        <el>
11800 nexty                                 <hf>
11810 nexti                                 <in>
11820 ifca(1)=ca(2)-1then11840              <lb>
11830 goto11910                             <dm>
11840 ifca(2)=ca(3)-1then11860              <po>
11850 goto11910                             <co>
11860 ifca(3)=ca(4)-1then11880              <hf>
11870 goto11910                             <fo>
11880 ifca(4)=ca(5)-1andfa(1)=fa(3
)andfa(3)=fa(5)andfa(5)=fa(7)andfa
(7)=fa(9)andca(5)=13then1=9:goto12
060                                         <dk>
11890 ifca(4)=ca(5)-1andfa(1)=fa(3
)andfa(3)=fa(5)andfa(5)=fa(7)andfa
(7)=fa(9)then1=8:goto12060                  <if>
11900 ifca(4)=ca(5)-1then1=5:goto1
2060                                        <mb>
11910 iffa(1)=fa(3)andfa(3)=fa(5)a
ndfa(5)=fa(7)andfa(7)=fa(9)then1=6
:goto12060                                  <dk>
11920 rem kein gewinn                       <pg>
11930 color1,1                              <an>
11940 gshapeha$,27,179                      <bd>
11950 gshapeha$,92,179                      <hh>
11960 gshapeha$,148,179                     <if>
11970 gshapeha$,212,179                     <cg>
11980 gshapeha$,268,179                     <ef>
11990 char,4,23,"1"                         <ik>
12000 char,12,23,"2"                        <oi>
12010 char,19,23,"3"                        <fn>
12020 char,27,23,"4"                        <gm>
12030 char,34,23,"5"                        <pm>
12040 z1=0:z2=0:z3=0:z4=0:z5=0              <kb>
12050 play"c":goto9650                      <fp>
12060 ifl=1thenla$="  ein paar   "
:vg=10                                      <fp>
12070 ifl=2thenla$=" zwei paare  "
:vg=20                                      <me>
12080 ifl=3thenla$="drei gleiche "
:vg=30                                      <mf>
12090 ifl=4thenla$=" full house  "
:vg=50                                      <kb>
12100 ifl=5thenla$="5 einer reihe"
:vg=70                                      <bo>
12110 ifl=6thenla$="5 einer farbe"
:vg=100                                     <od>
12120 ifl=7thenla$="vier gleiche "
:vg=150                                     <cf>
12130 ifl=8thenla$="5 einer reihe"
:vg=200                                     <kk>
12140 ifl=9thenla$="fuenf bis as "
:vg=250                                     <ld>
12150 fori=1to27:char,2,7,la$               <ea>
12160 la$=right$(la$,len(la$)-1)+l
eft$(la$,1):nexti                           <om>
12170 vr=vg:vg$=str$(vg):char,31,9
,"    ":char,31,9,vg$:char,2,5,"
   ":char,2,5,vg$:gosub13290                <od>
12180 color1,1                              <ln>
12190 gshapeha$,27,179                      <fo>
12200 gshapeha$,92,179                      <dm>
12210 gshapeha$,148,179                     <jg>
12220 gshapeha$,212,179                     <bg>
12230 gshapeha$,268,179                     <ng>
12240 char,4,23,"1"                         <hg>
12250 char,12,23,"2"                        <fb>
12260 char,19,23,"3"                        <oa>
12270 char,27,23,"4"                        <pc>
12280 char,34,23,"5"                        <ej>
12290 l=0:z1=0:z2=0:z3=0:z4=0:z5=0
:c=0:d=0:e=0:f=0:goto9650                   <ep>
12300 rem geld                              <jk>
12310 vo=vo-10:ifvo<0then12360              <ji>
12320 vo$=str$(vo)                          <bh>
12330 char,31,3,"    "                      <aa>
12340 char,31,3,vo$                         <bi>
12350 return                                <gk>
```

```
12360 a$="geld zu ende  high score
=1 weiter=2   "                    <fa>
12370 fori=1to52:char,1,1,a$:forx=
1to10:nextx                        <nf>
12380 a$=right$(a$,len(a$)-1)+left
$(a$,1):nexti                      <ga>
12390 getkeya$:ifa$="1"then14470   <bo>
12400 ifa$="2"then12410:else12370  <ic>
12410 ifhs<=0orhs<40then9020       <oe>
12420 ifhs<400thenvo=int(hs/40)*10
:hs=hs-vo*4:hs$=str$(hs):char,18,3
,"     ":char,18,3,hs$:vo$=str$(vo
):char,31,3,"    ":char,31,3,vo$:g
oto9550                            <fn>
12430 hs=hs-400:vo=100:char,18,3,"
     ":hs$=str$(hs):vo$=str$(vo):c
har,31,3,"    ":char,31,3,vo$:char
,18,3,hs$:goto9550                 <ma>
12440 rem                          <ca>
12450 scnclr                       <in>
12460 print:print"    spielanleitu
ng"                                <lm>
12470 print:print"dieses programm
simuliert den merkur-"             <ic>
12480 print"automaten full house" <jd>
12490 print"es ist eine art von po
ker"                               <ab>
12500 print:print"am anfang erhalt
en sie 150 punkte"                 <ok>
12510 print"pro spiel gehen 10 pun
kte weg. pro spiel gibt es zwei ka
rtendurchlaeufe."                  <ch>
12520 print"beim ersten durchlauf,
 den sie mit "                     <nj>
12530 print"'space' starten, zieht
 der computer 5"                   <om>
12540 print"karten. mit hilfe der
tasten 1 - 5 mues-"                <na>
12550 print"sen sie nun die karten
 kennzeichen die"                  <ek>
12560 print"nicht mehr gezogen wer
den sollen. wenn"                  <aj>
12570 print"sie z.b. die taste '1'
 druecken leuchtet"                <fn>
12580 print"der kreis unter der 1.
 karte gelb. die "                 <bi>
12590 print"erste karte wird gehal
ten. wenn sie nun"                 <ak>
12600 print"erneut die '1' druecke
n wird der kreis"                  <nn>
12610 print"wieder schwarz, die ka
rte wird nicht"                    <bm>
12620 print"mehr gehalten. nachdem
 sie alle karten"                  <en>
12630 print"gekennzeichnet haben," <nd>
12640 print:print"weiter taste dru
ecken"                             <kh>
12650 getkeyx$:scnclr:print:print  <mf>
12660 print"druecken sie die space
-taste. der zweitedurchlauf wird
gestartet."                        <pp>
12670 print"der computer zieht fue
r alle nichtge-"                   <dp>
12680 print"kennzeichneten karten
eine neue karte"                   <eo>
12690 print"falls sie nun gewinnen
 wird ihr gewinn  im risiko und im
 gewinnfeld angezeigt"             <hd>
12700 print"wenn sie den gewinn nu
n uebernehmen    wollen druecken
sie die 'esc' taste."              <hk>
12710 print"durch druecken der 'sp
ace' taste wird"                   <fi>
12720 print"risiko eingeschaltet.
der faktor mit    dem der gewinn m
iltipliziert wird und   die null b
linken abwechselnd auf. durch"     <ng>
12730 print"druecken der taste 'sp
ace' wird gestopt"                 <od>
12740 print"wenn nun die 0 leuchte
t haben sie den "                  <an>
12750 print"gewinn verloren. anson
sten erscheint im gewinnfeld der n
eue gewinn. nun kann"              <hj>
12760 print"man wieder risiko spie
len oder ueber-"                   <di>
12770 print"nehmen."               <lb>
12780 print:print"weiter taste dru
ecken"                             <aj>
12790 getkeyx$:scnclr              <df>
12800 print:print:print"der gewinn
 kommt in das high-score "         <md>
12810 print"feld. wenn die 150 pun
kte verbraucht"                    <kj>
12820 print"sind koennen sie aus d
iesem feld punkte"                 <gp>
12830 print"zum weiterspielen uebe
rnehmen. aber"                     <pp>
12840 print"achtung fuer jeden ueb
ernommenen punkt"                  <kn>
12850 print"gehen 4 punkte vom hig
h-score feld. der"                 <pg>
12860 print"computer uebernimmt ma
ximal 100 punkte"                  <gd>
12870 print"ins punktefeld."       <cc>
12880 print:print"der sinn des spi
eles ist, moeglichst"              <na>
12890 print"viele punkte ins high-
score feld zu be-"                 <af>
12900 print"kommen, da man sich da
nn in die "                        <on>
12910 print"high-score liste eintr
agen darf."                        <dc>
12920 print:print"weiter taste dru
ecken"                             <ia>
12930 getkeyx$:scnclr:goto12950    <nb>
```

```
12940 rem ******* tabelle ********
**                                <kj>
12950 color1,1                    <an>
12960 char,5,2,"     uebersicht
"                                 <ko>
12970 char,33,4,"punkte"          <ga>
12980 char,1,5,"ein paar
                  10"             <op>
12990 char,1,6,"zwei paare
                  20"             <ic>
13000 char,1,7,"drei gleiche
                  30"             <fb>
13010 char,1,8,"full house (3gleic
he und 2gleiche)  50"             <eg>
13020 char,1,9,"fuenf in einer rei
he ung. farbe     70"             <jg>
13030 char,1,10,"fuenf von einer f
arbe              100"            <jo>
13040 char,1,11,"vier gleiche
                  150"            <be>
13050 char,1,12,"fuenf in einer re
ihe gl. farbe     200"            <ap>
13060 char,1,13,"5 in einer reihe
bis as gl. farbe  250"            <go>
13070 char,1,15,"5 in einer reihe
heisst z.b. 2,3,4,5,6"            <ee>
13080 char,1,16,"kann aber auch du
rcheinander sein."                <pb>
13090 char,1,17,"z.b. 4 3 5 7 6"  <lo>
13100 char,1,20,"alles verstanden
 (j/n)"                           <oh>
13110 getkeya$:ifa$<>"j"anda$<>"n"
then13100                         <io>
13120 ifa$="n"then goto12440      <kg>
13130 scnclr:return               <in>
13140 la$="
      siegfried "                 <ji>
13150 printla$                    <ln>
13160 la$=right$(la$,len(la$)-1)+l
eft$(la$,1)                       <gj>
13170 goto 13150                  <kc>
13180 rem **                      <di>
13190 tempo15                     <df>
13200 play"o5cdcecfco4c"          <fn>
13210 for y=1to10                 <ah>
13220 j=int(rnd(x)*16)+1          <bh>
13230 color4,j                    <mi>
13240 nexty                       <al>
13250 return                      <hj>
13260 sound1,1000,40,,,,3         <lj>
13270 return                      <ka>
13280 rem risiko                  <on>
13290 s=0                         <ml>
13300 b$="   esc = uebernehmen  sp
ace = risko    "                  <na>
13310 char,1,1,b$                 <ph>
13320 rem blinken von risiko      <pc>
13330 color1,8:char,29,7,"risiko":
fori=1to20:nexti:color1,1:char,29,
7,"risiko":fori=1to10:nexti       <ak>
13340 geta$                       <kl>
13350 ifa$=chr$(27)andh=1thengosub
13770:goto13390                   <fp>
13360 ifa$=chr$(27)then13390      <lo>
13370 ifa$=chr$(32)thenh=1:goto134
10                                <jd>
13380 goto13330                   <ck>
13390 gosub13180:hs=hs+vr:char,31,
9,"   ":char,2,5,"     ":char,2,7,
"          ":char,18,3,"     ":
gosub13690                        <ja>
13400 hs$=str$(hs):char,18,3,hs$:r
eturn                             <ae>
13410 rem **                      <hb>
13420 s=s+1                       <ko>
13430 color1,8:char,37,9,"0"      <gl>
13440 color1,8:char,37,9,"0"      <fo>
13450 getb$:ifb$=chr$(32)then13660 <hg>
13460 color1,1:char,37,9,"0"      <ci>
13470 color1,8:gosub13510         <ca>
13480 getb$:ifb$=chr$(32)then13560 <on>
13490 color1,1:gosub13510         <an>
13500 goto13430                   <fb>
13510 ifs=5thenchar,11,4,"x32":ret
urn                               <bp>
13520 ifs=4thenchar,15,5,"x16":ret
urn                               <dm>
13530 ifs=3thenchar,19,6,"x8":retu
rn                                <hg>
13540 ifs=2thenchar,22,7,"x4":retu
rn                                <fg>
13550 ifs=1thenchar,25,8,"x2":retu
rn                                <lj>
13560 ifs=1thenvr=vg*2:goto13610  <ck>
13570 ifs=2thenvr=vg*4:goto13610  <nj>
13580 ifs=3thenvr=vg*8:goto13610  <if>
13590 ifs=4thenvr=vg*16:goto13610 <pa>
13600 ifs=5thenvr=vg*32:goto13610 <ep>
13610 vr$=str$(vr)                <aa>
13620 color1,1                    <bl>
13630 char,2,5,"     ":char,2,5,vr
$                                 <al>
13640 ifs=5thenhs=hs+vr:hs$=str$(h
s):char,18,3,"     ":char,18,3,hs$
:gosub13180:goto13670             <hd>
13650 goto13320                   <bm>
13660 gosub13260                  <ch>
13670 gosub13690:char,31,9,"   ":c
har,2,5,"     "                   <ce>
13680 gosub13770:return           <pp>
13690 color1,1                    <ia>
13700 char,25,8,"x2"              <bh>
13710 char,22,7,"x4"              <pb>
13720 char,19,6,"x8"              <ag>
13730 char,15,5,"x16"             <nn>
13740 char,11,4,"x32"             <pp>
```

```
13750 char,37,9,"0"                    <eg>
13760 return                           <hj>
13770 a$="full house (c)1987 by si
egfried ederer"                        <kp>
13780 char,1,1,a$                      <an>
13790 h=0:return                       <dh>
13800 play"c"                          <na>
13810 rem herz sshape                  <jf>
13820 scnclr                           <ek>
13830 draw1,0,2to4,8to8,2              <jd>
13840 circle1,2,2,2,2,270,90           <jp>
13850 circle1,6,2,2,2,270,90           <ii>
13860 paint1,4,6                       <ee>
13870 sshapehe$,0,0,8,8                <kk>
13880 scnclr                           <me>
13890 gshapehe$,6,7                    <ii>
13900 box1,2,2,18,21                   <lp>
13910 sshapea$,1,1,18,21               <fg>
13920 sprsava$,1                       <oc>
13930 scnclr:circle,10,11,5,,,,,90     <ce>
13940 paint1,10,11:box1,2,2,18,21      <ne>
13950 sshapea$,1,1,18,21               <fj>
13960 sprsava$,2                       <of>
13970 scnclr                           <hj>
13980 rem pik sshape                   <pe>
13990 draw1,0,6to4,0to8,6              <ad>
14000 circle1,2,6,2,2,90,270           <ge>
14010 circle1,6,6,2,2,90,270           <ng>
14020 paint1,4,3                       <gj>
14030 draw1,4,9to4,11                  <ma>
14040 sshapekr$,0,0,8,12               <oj>
14050 scnclr                           <bi>
14060 gshapekr$,6,7                    <lm>
14070 paint1,10,11:box1,2,2,18,21      <ad>
14080 sshapea$,1,1,18,21               <dg>
14090 sprsava$,3                       <jd>
14100 scnclr                           <ho>
14110 draw1,5,11to5,3:draw1,3,7to7
,7                                     <eb>
14120 circle,5,2,2,2:circle,2,7,2,
2:circle,8,7,2,2                       <cp>
14130 paint1,5,1:paint1,1,7:paint1
,9,7                                   <hg>
14140 paint1,5,1:paint1,1,7:paint1
,9,7                                   <in>
14150 sshapepi$,0,0,10,11              <lc>
14160 scnclr                           <ph>
14170 gshapepi$,5,7                    <cd>
14180 box1,2,2,18,21                   <fc>
14190 sshapea$,1,1,18,21               <ai>
14200 sprsava$,4                       <dd>
14210 scnclr:return                    <gp>
14220 slow:poke53265,27                <md>
14230 color1,11:char,7,24,"full ho
use : automatenpoker"                  <kd>
14240 sprite1,1,3,0,1,1,0              <gb>
14250 sprite2,1,3,0,1,1,0              <ee>
14260 sprite3,1,1,0,1,1,0              <im>
14270 sprite4,1,1,0,1,1,0              <ld>
14280 movspr1,50,50                    <ha>
14290 a=int(rnd(x)*7)+1:movspr1,90
#a                                     <ei>
14300 movspr2,75,140                   <gl>
14310 a=int(rnd(x)*7)+1:movspr2,90
#a                                     <hd>
14320 movspr3,100,95                   <af>
14330 a=int(rnd(x)*7)+1:movspr3,27
0#a                                    <do>
14340 movspr4,125,185                  <ma>
14350 a=int(rnd(x)*7)+1:movspr4,27
0#a                                    <pj>
14360 rem die lorelei von friedric
h silcher (1789 - 1860)                <lp>
14370 let i=0                          <jk>
14380 envelope1,12,0,12,0,1:play"t
1"                                     <mj>
14390 tempo 8                          <kn>
14400 play" o4ig i.g sa ig o5ico4i
b ia q.g qf if qe  ie id ic id"        <ma>
14410 play" o4q.e ie ig i.g sa ig
o5ic o4ib ia q.g gf if qe ie ig if
 id"                                   <ck>
14420 play "o4q.c qc ie i.d se id
ig id id q.b qa ia qg ig i#f ig ia
"                                      <co>
14430 play "o4q.g qg ig i.g sa ia
o5ic o4ib ia qg o5ie qd id qc ic o
4ib ia ib"                             <hg>
14440 play "o5q.c ic":envelope0,0,
9,0,0,2,1536:play"t0":tempo4           <nb>
14450 sprite1,0:sprite2,0:sprite3,
0:sprite4,0:return                     <gn>
14460 rem high score                   <cg>
14470 hs(10)=hs                        <ig>
14480 dopen#9,"high-score",d0,u8       <jh>
14490 fori=1to9                        <dd>
14500 input#9,hs(i)                    <of>
14510 input#9,na$(i)                   <da>
14520 nexti                            <ap>
14530 dclose#9                         <fj>
14540 gosub14800                       <ne>
14550 ifhs(10)<hs(9)then15070          <op>
14560 gosub14990                       <pd>
14570 fori=1to9                        <bp>
14580 fory=i+1to10                     <ca>
14590 ifhs(i)>=hs(y)then14660          <fd>
14600 leths(0)=hs(i)                   <hc>
14610 leths(i)=hs(y)                   <ca>
14620 leths(y)=hs(0)                   <oh>
14630 letna$(0)=na$(i)                 <ba>
14640 letna$(i)=na$(y)                 <ma>
14650 letna$(y)=na$(0)                 <ba>
14660 nexty                            <fb>
14670 nexti                            <gm>
14680 fori=1to9                        <ad>
14690 ifna$(i)=""thenna$(i)="....?
```

```
...."                                   <pc>
14700 nexti                             <oe>
14710 gosub14800                        <ld>
14720 scratch"high-score"               <fk>
14730 dopen#9,"high-score",d0,u8,w      <ac>
14740 fori=1to9                         <pe>
14750 print#9,hs(i)                     <pc>
14760 print#9,na$(i)                    <an>
14770 nexti                             <pl>
14780 dclose#9                          <ak>
14790 goto15070                         <cm>
14800 graphic0,1                        <mn>
14810 color0,15:color4,7:color5,1       <ak>
14820 char,10,0,"high-score"            <eb>
14830 color5,7                          <an>
14840 fori=1to10                        <nh>
14850 hs$(i)=str$(hs(i))                <eb>
14860 i$(i)=str$(i)                     <pa>
14870 nexti                             <in>
14880 y=1                               <mk>
14890 fori=1to9                         <io>
14900 y=y+2                             <mn>
14910 color5,14                         <gm>
14920 char,2,y,i$(i)                    <nm>
14930 color5,1                          <fo>
14940 char,5,y,hs$(i)                   <cc>
14950 color5,3                          <kb>
14960 char,20,y,na$(i)                  <nh>
14970 nexti                             <ca>
14980 return                            <al>
14990 rem                               <ca>
15000 char,2,22,"ihre punkte"           <ka>
15010 char,21,22,hs$(10)                <bl>
15020 char,2,23,"namen eingeben"        <ga>
15030 char,20,23,""                     <cm>
15040 inputna$(10)                      <ik>
15050 na$(10)= mid$(na$(10),1,12)       <pd>
15060 return                            <kk>
15070 char,2,22,"wollen sie es noc
hmal versuchen?"                        <cc>
15080 char,2,23,"j fuer nochmal n
fuer spiel beenden"                     <ng>
15090 getkeya$:ifa$="j"thenhs=0:co
lor0,2:goto9020                         <id>
15100 ifa$="n"thenscnclr:end            <cd>
15110 goto15090                         <ib>
15120 poke53265,0:fast                  <nj>
15130 scnclr                            <jc>
15140 circle1,3,3,3,3                   <ef>
15150 draw1,6,3to6,7                    <ec>
15160 circle1,3,7,3,3,90,220            <bf>
15170 sshapene$,0,0,6,10                <ia>
15180 scnclr                            <ph>
15190 draw1,5,0to0,7to10,7              <hf>
15200 draw1,5,0to5,11                   <fc>
15210 sshapevi$,0,0,10,11               <co>
15220 scnclr                            <eh>
15230 draw1,4,0to0,0to0,5to2,5          <ob>
15240 circle1,2,8,3,3,0,220             <ep>
15250 sshapefu$,0,0,5,11                <dl>
15260 scnclr                            <jh>
15270 circle1,3,7,3,3                   <gh>
15280 draw1,0,7to0,3                    <co>
15290 circle1,3,3,3,3,270,40            <pp>
15300 sshapese$,0,0,6,10                <bi>
15310 scnclr                            <pk>
15320 draw1,0,0to6,0to2,10:draw1,1
,5to6,5                                 <dp>
15330 sshapesi$,0,0,6,10                <of>
15340 scnclr                            <dg>
15350 circle1,3,7,3,3                   <co>
15360 circle1,3,2,2,2                   <nf>
15370 sshapeac$,0,0,6,10                <gd>
15380 scnclr                            <ii>
15390 circle1,7,5,2,5                   <jc>
15400 draw1,0,3to3,0to3,10              <fa>
15410 sshapeze$,0,0,9,10                <ak>
15420 scnclr                            <ni>
15430 circle1,2,2,2,2,270,100           <mc>
15440 draw1,4,2to0,10to4,10             <pk>
15450 sshapezw$,0,0,4,10                <ia>
15460 scnclr                            <cj>
15470 circle1,3,2,3,2,280,180           <mg>
15480 circle1,3,7,3,3,0,270             <bp>
15490 sshapedr$,0,0,6,10                <em>
15500 scnclr                            <hj>
15510 width2                            <pi>
15520 draw1,0,0to0,10:draw1,0,0to2
,0:draw1,0,10to2,10                     <cp>
15530 circle1,4,5,3,5,0,180             <fn>
15540 sshapeda$,0,0,10,10               <fm>
15550 scnclr                            <nn>
15560 draw1,0,0to0,10:draw1,0,0to2
,0:draw1,0,10to2,10                     <hd>
15570 circle1,4,2,3,2,0,180             <pk>
15580 circle1,4,7,3,3,0,180             <mn>
15590 draw1,0,4to4,4                    <oe>
15600 sshapebu$,0,0,10,10               <gm>
15610 width1                            <bc>
15620 return                            <ba>
```

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

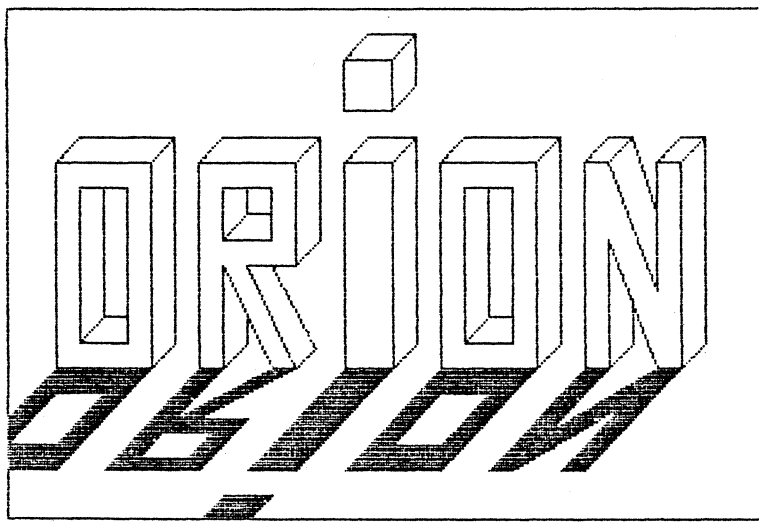
Den „Großen und Kleinen Wagen" kennt vermutlich jeder, dann hackt's aber schon bei den meisten. Nach diesem Menüpunkt können Sie ohne weiteres 24 Sternbilder am Himmel erkennen und identifizieren. Jedes Sternbild wird grafisch dargestellt, die imaginäre Verbindung der einzelnen Fixsterne untereinander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

```
10 rem ======================128  <ob>
20 rem (p) commodore welt      ==  <hf>
30 rem ==========================  <mm>
40 rem (c) by                  ==  <pp>
50 rem     rolf lange          ==  <mj>
60 rem                         ==  <nd>
70 rem version 7.0   40z./ascii==  <ah>
80 rem pc-128 floppy/datasette ==  <ho>
90 rem ==========================  <km>

95 gosub 60000                     <jp>
100 gosub10310                     <jk>
110 scnclr                         <me>
120 width1                         <ce>
130 color4,1:color0,1:color5,2:col
or1,2                              <oe>
140 color2,3:color3,15       .     <km>
150 print:print"          h a u p
t m e n u e"                       <oh>
160 print:print:print:print        <md>
170 print"  0- helligkeit der ster
ne"                                <bf>
180 print"  1- farben der sterne"  <bj>
190 print"  2- die spektralklassen
"                                  <ol>
200 print"  3- die himmelskoordina
ten"                               <ee>
210 print"  4- die sternbilder"    <fg>
220 print"  5- die hellsten sterne
"                                  <bn>
230 print"  6- unser planetensyste
m "                                <pp>
240 print"  7- die galaxien"       <hl>
250 print"  8- besondere sternkons
tellationen"                       <gi>
260 print"  9- ende"               <oi>
270 print:print:print:print:print  <dn>
280 print"              "zu$       <oe>
290 printc2$"  ihre wahl";:inputw  <ei>
300 w=w+1                          <im>
310 onwgoto320,590,1020,1710,2380,
7030,7610,7980,8580,10650          <jl>
320 scnclr                         <gk>
330 print                          <cp>
340 print"       die helligkeit de
r sterne"                          <af>
350 print"       ================
========"                         <dm>
360 print                          <gl>
370 print" nach der helligkeit ein
es sterns wird"                    <hj>
380 print" seine groesse definiert
. die einheit"                     <jg>
390 print" heisst auch groesse. di
e abk. dafuer"                     <kg>
400 print" ist ein kleines m, von
lat. magnitudo"                    <mm>
410 print" (groesse). je heller ei
n stern ist,"                      <ob>
420 print" desto kleiner ist der w
ert seiner"                        <ia>
430 print" groesse. da in den anfa
engen der as-"                     <ek>
440 print" tronomie aber noch nich
t alle sterne"                     <pp>
450 print" bekannt waren, ist es h
eute der fall, "                   <ll>
460 print" dass einige sterne nega
tive groessen-"                    <nc>
470 print" werte haben, da sie hel
ler als 0.0 m"                     <dk>
480 print" sind (z.b. sirius -1.4m
)."                                <od>
490 print" nun gibt es aber zwei v
ersch. hellig-"                    <bi>
500 print" keiten, die visuelle un
d die tat-"                        <gp>
510 print" saechliche h. die vis.
helligkeit ist"                    <go>
520 print" die, die wir wahrnehmen
. die tat. h."                     <gl>
530 print" gibt die helligkeit all
er sterne an,"                     <db>
540 print" wenn sie alle in gleich
er entfernung"                     <ag>
550 print" waeren. uns interessier
t aber nur die"                    <nl>
560 print" visuelle helligkeit."   <eo>
570 getm$:ifm$=""then570           <ob>
580 goto110                        <pp>
590 scnclr                         <ii>
600 print                          <en>
610 print"          die farbe der
sterne"                            <ek>
620 print"          ==============
======"                           <pb>
630 print:print                    <dd>
640 print" die helleren sterne leu
chten in "                         <jl>
650 print" versch. farben. wie kom
mt das? die"                       <mi>
660 print" ursache hierfuer liegt
in der tempera-"                   <pm>
670 print" tur eines sterns. je ho
eher die t."                       <ge>
680 print" naemlich ist, desto meh
r verschiebt"                      <em>
690 print" sich das sichtbare spek
trum des sterns"                   <gp>
700 print" zum kurzwelligen blauen
 licht. (siehe"                    <dk>
710 print" auch spektralklassen) d
.h., dass sehr"                    <lh>
720 print" kuehle sterne roetlich
erscheinen,"                       <fl>
730 print" sehr heisse dagegen bla
```

```
eulich-weiss."                                  <cl>
740 print" unter '2-spektralklasse
n' finden sie"                                  <fp>
750 print" genauere unterteilungen
 zu den themen"                                 <ko>
760 print" farbe und temperatur."               <pp>
770 getm$:ifm$=""then770                        <ja>
780 scnclr                                      <ag>
790 print:print" im vorgriff auf d
en abschnitt 'spek-"                            <ji>
800 print" tralklassen' wird hier
nun noch der zu-"                               <gi>
810 print" sammenhang zw. den spek
tralklassen"                                    <dk>
820 print" und den farben der ster
ne gezeigt."                                    <bi>
830 print" die einzelnen spektralk
lassen sind:"                                   <mc>
840 print:print                                 <mk>
850 print"     o - b - a - f - g -
 k - m"                                         <ae>
860 print                                       <fh>
870 print" die dazugehoerigen farb
en lauten:                                      <df>
880 print                                       <hp>
890 print"     o : blaeulich/weiss
"                                               <bo>
900 print"     b : weiss"                       <da>
910 print"     a : weissgelb"                   <nc>
920 print"     f : weissgelb-gelb"              <lj>
930 print"     g : gelb-orange"                 <hm>
940 print"     k : orange"                      <gk>
950 print"     m : rot"                         <he>
960 print                                       <ca>
970 print" jede dieser klassen ist
 dann nochmal"                                  <ba>
980 print" in 10 unterklassen (0-9
) unterteilt."                                  <lo>
990 print" z.b.:g9,b0,a7,g2,usw."               <lk>
1000 getq$:ifq$=""then1000                      <bl>
1010 goto110                                    <pg>
1020 scnclr                                     <oh>
1030 print                                      <kn>
1040 print"              spektralkl
assen"                                          <eh>
1050 print"              ==========
=====""                                         <ig>
1060 print:print                                <ld>
1070 print" die spektralklassen la
ssen sich"                                      <if>
1080 print" am besten anhand eines
 diagramms, des"                                <hf>
1090 print" sogenannten hertzsprun
g-russel-diagr."                                <oo>
1100 print" erklaeren. bei den tem
peraturangaben"                                 <aj>
1110 print" wird in der astronomie
 die einheit"                                   <mo>
1120 print" kelvin ÄkÜ verwendet.
0 grad ent-"                                    <gb>
1130 print" sprechen hierbei 273 k
."                                              <hn>
1140 print:print                                <dd>
1150 print" druecken sie eine tast
e !"                                            <oa>
1160 getd$:ifd$=""then1160                      <fk>
1170 scnclr                                     <bf>
1180 color0,1:color4,1:color5,2:co
lor1,2                                          <gk>
1190 graphic1,1                                 <oh>
1200 box1,16,16,304,184                         <pg>
1210 char1,4,24,"b0     a0     f0
    g0     k0   m"                              <he>
1220 char1,0,12,"+5m"                           <gj>
1230 char1,0,6,"0m"                             <ip>
1240 char1,0,1,"-5m"                            <pn>
1250 char1,0,18,"10m"                           <km>
1260 char1,0,24,"15m"                           <hl>
1270 char1,2,1,"   20000k        8
000k     5000k"                                 <ce>
1280 circle1,230,23,2                           <mp>
1290 char1,28,4,"polarstern"                    <id>
1300 circle1,300,45,2                           <an>
1310 char1,30,5,"antares"                       <im>
1320 circle1,46,50,2                            <ag>
1330 char1,4,4,"spica"                          <ib>
1340 circle1,265,67,2                           <pp>
1350 char1,31,7,"capella"                       <ec>
1360 circle1,100,70,2                           <co>
1370 circle1,140,90,2                           <il>
1380 circle1,150,110,2                          <hh>
1390 circle1,230,130,2                          <ji>
1400 char1,13,8,"regulus"                       <bc>
1410 char1,19,11,"sirius"                       <bj>
1420 char1,20,13,"atair"                        <bd>
1430 char1,30,16,"sonne"                        <jn>
1440 circle1,87,135,30,15                       <id>
1450 char1,8,16,"weisse",1                      <jc>
1460 char1,8,17,"zwerge",1                      <ne>
1470 char1,18,3,"ueber-",1                      <mf>
1480 char1,18,4,"riesen=>",1                    <bj>
1490 char1,24,20,"zwerge=>",1                   <ag>
1500 char1,24,19,"rote",1                       <fg>
1510 draw1,304,184to290,135to58,45
to30,16                                         <fn>
1520 draw1,304,184to230,140to32,80
to16,35                                         <ge>
1530 char1,29,12,"  haupt-",1                   <em>
1540 char1,29,13,"<=reihe",1                    <ng>
1550 color4,1                                   <nc>
1560 paint0,20,20,0                             <fk>
1570 getm$:ifm$=""then1570                      <ip>
1580 graphic0,0                                 <fl>
1590 scnclr                                     <ga>
1600 print:print" in der hauptreih
e liegen die meisten"                           <cj>
```

```
1610 print:print" sterne. die uebe
rriesen sind roetlich"                <kl>
1620 print:print" bis rot, relativ
 kuehl und hell."                     <kj>
1630 print:print" die zwerge sind
nicht so leuchtstark,"                <fc>
1640 print:print" weisse sind rela
tiv heiss und blau-"                  <ph>
1650 print:print" weiss. die roten
 zwerge sind dagegen"                 <lg>
1660 print:print" sehr kalt und vo
n der farbe her rot."                 <pi>
1670 print:print:print                 <ha>
1680 print" taste druecken !!"        <fa>
1690 getq$:ifq$=""then1690            <ne>
1700 goto110                          <em>
1710 scnclr                           <fb>
1720 print                            <bg>
1730 print"         die himmelskoo
rdinaten"                             <bi>
1740 print                            <do>
1750 print" um die position eines
sterns am himmel"                     <gd>
1760 print" genau zu beschreiben,
gibt es fuer den"                     <fk>
1770 print" himmel aehnliche koord
inaten wie auf  "                     <ik>
1780 print" der erde laengen- und
breitengrade."                        <bp>
1790 print" die hoehe ueber dem ho
rizont heisst"                        <kd>
1800 print" deklination Ädekl.Ü un
d wird in grad"                       <ic>
1810 print" gemessen Ä0-90Ü. ein s
tern der 0 grad"                      <ao>
1820 print" dekl. hat liegt demnac
h auf dem him-"                       <lc>
1830 print" melsaequator. die rich
tung, z.b."                           <jp>
1840 print" n,s,e,w, heisst rektas
zension. sie"                         <ol>
1850 print" wird in stunden, minut
en und sekunden"                      <dd>
1860 print" gemessen. 24h sind hie
rbei 360 grad."                       <ol>
1870 print" uebrigens: an jedem or
t der welt hat"                       <fb>
1880 print" der polarstern genau d
ie dekl., die"                        <di>
1890 print" dem breitengrad des or
tes entspricht."                      <bg>
1900 print" z.b.: ueber hamm hat d
er polarstern"                        <oh>
1910 print" 51 grad dekl. also lie
gt hamm auf dem"                      <ag>
1920 print" 51. breitengrad."         <pk>
1930 getw$:ifw$=""then1930            <dm>
1940 scnclr                           <bp>
1950 graphic1,1                       <jh>
1960 circle1,160,100,100              <hp>
1970 circle1,160,100,20,100           <em>
1980 circle1,160,100,100,20           <mj>
1990 circle1,160,100,100,20,,,23      <ak>
2000 char1,1,1,"die himmels-"         <hk>
2010 char1,1,2,"kugel"                <je>
2020 char1,15,15,"0h"                 <jd>
2030 char1,0,12,"ost"                 <mo>
2040 char1,35,12,"west"               <oa>
2050 char1,23,9,"12h"                 <ai>
2060 char1,29,12,"18h"                <ie>
2070 char1,9,12,"6h"                  <cm>
2080 char1,18,1,"90g"                 <ai>
2090 char1,8,14,"ha"                  <gf>
2100 char1,15,7,"ea"                  <cb>
2110 char1,0,20,"ea=erd-"             <cc>
2120 char1,0,21,"aequator"            <ai>
2130 char1,0,22,"ha=himmels-"         <of>
2140 char1,0,23,"aequator"            <cg>
2150 char1,18,23,"-90g"               <pk>
2160 char1,30,22,"h=stunden"          <jm>
2170 char1,30,23,"g=grad"             <fg>
2180 circle1,160,100,2:paint1,160,
100                                   <fg>
2190 char1,21,12,"e"                  <bk>
2200 char1,30,21,"e=erde"             <pm>
2210 getm$:ifm$=""then2210            <dl>
2220 graphic0,0                       <id>
2230 scnclr                           <gf>
2240 print:print" uebrigens: die s
cheinbare bahn der"                   <oa>
2250 print:print" sonne um die erd
e heisst ekliptik."                   <km>
2260 print:print" die sternbilder,
 durch die die eklip-"                <ob>
2270 print:print" tik verlaeuft, s
ind die tierkreiszei-"                <do>
2280 print:print" chen. die eklipt
ik ist gegen den "                    <ff>
2290 print:print" himmelsaequator
um 23.5 grad geneigt."                <bp>
2300 print:print" die beiden scnit
tpunkte zw. ekl. und"                 <gl>
2310 print:print" himmelsaequator
heissen fruehlings-"                  <lj>
2320 print:print" und herbstpunkt.
 im fruehlingspunkt"                  <ne>
2330 print:print" betraegt die rek
taszension 0 stunden"                 <am>
2340 print:print" und die deklinat
ion 0 grad (herbst-"                  <lb>
2350 print:print" punkt: 12 h, 0 g
rad)"                                 <on>
2360 getq$:ifq$=""then2360            <ac>
2370 goto110                          <eo>
2380 scnclr                           <jc>
2390 graphic1,1                       <db>
```

```
2400 char1,11,0,"zeichenerklaerung
"                                  <fn>
2410 circle,5,16,5:paint1,5,16     <fb>
2420 circle,5,34,4:paint1,5,34     <gc>
2430 circle,5,51,3:paint1,5,51     <pj>
2440 circle,5,68,2:paint1,5,68     <kh>
2450 circle,5,83,1:paint1,5,83     <kb>
2460 circle,5,99                   <go>
2470 char1,2,2,"sterne heller als
1.0m"                              <pg>
2480 char1,2,4,"sterne von 1.0-2.0
m"                                 <go>
2490 char1,2,6,"sterne von 2.0-3.0
m"                                 <oj>
2500 char1,2,8,"sterne von 3.0-4.0
m"                                 <mg>
2510 char1,2,10,"sterne von 4.0-5.
0m"                                <ib>
2520 char1,2,12,"sterne unter 5.0m
"                                  <fj>
2530 circle,5,115,5,2              <ig>
2540 char1,2,14,"nebel und sternha
ufen"                              <ml>
2550 char1,1,17,"bei jedem sternbi
ld steht folgendes :               <jn>
2560 char1,1,19,"dt. name"         <gh>
2570 char1,1,20,"lat. name, abk."  <cn>
2580 char1,1,21,"rektaszension Äin
 stunden,minuten(h,')Ü"            <ne>
2590 char1,1,22,"deklination Äin g
radÜ"                              <fo>
2600 getq$:ifq$=""then2600else2610 <ii>
2610 graphic0                      <nm>
2620 scnclr                        <he>
2630 print:print"           sternb
ilder"                             <nd>
2640 print:print                   <ec>
2650 print"  1-orion          13-
fuhrmann"                          <cg>
2660 print"  2-stier          14-
zwillinge                          <cc>
2670 print"  3-gr.hund        15-
gr.wagen"                          <fo>
2680 print"  4-kl.hund        16-
kl.wagen"                          <ad>
2690 print"  5-adler          17-
leier"                             <ba>
2700 print"  6-schwan         18-
cassiopeia"                        <eo>
2710 print"  7-centaur        19-
andromeda"                         <he>
2720 print"  8-kreuz d.suedens 20-
steinbock"                         <ad>
2730 print"  9-schuetze       21-
wassermann"                        <aj>
2740 print" 10-bootes         22-
noerdl.krone"                      <eg>
2750 print" 11-pegasus        23-
loewe"                             <gc>
2760 print" 12-jungfrau       24-
skorpion"                          <nf>
2770 print:print" "                <lp>
2780 print" sternbild nr."         <ca>
2790 print"              "c2$;:in
puts                               <ip>
2800 onsgoto2810,3110,3340,3510,36
70,3830,4020,4200,4330,4510,4680,4
860,5040,5230,5420,5620,5800,5940,
6080,6240,6400,6560,6700,6840      <mk>
2810 scnclr                        <pa>
2820 graphic1,1                    <dc>
2830 color0,1:color4,1:color5,2    <ni>
2840 circle1,94,28,5:circle1,175,3
4,4:circle1,102,185,3:circle1,200,
180,5                              <na>
2850 circle1,134,116,4:circle1,150
,107,4:circle1,160,101,4           <li>
2860 paint1,94,28:paint1,175,34:pa
int1,102,185:paint1,200,180:paint1
,134,116                           <ja>
2870 paint1,150,107:paint1,160,101 <lm>
2880 draw1,170,94to188,80to188,84:
draw1,188,80to184,80               <ga>
2890 draw1,126,124to106,136to110,1
36:draw1,106,136to108,132          <mk>
2900 char1,23,11,"zum"             <ma>
2910 char1,23,12,"aldebaran"       <fc>
2920 draw1,160,101to175,34to94,28t
o134,116                           <mg>
2930 char1,10,14,"sirius"          <lc>
2940 char1,10,13,"zum"             <jb>
2950 draw1,134,116to150,107to160,1
01to200,180to102,185to134,116      <mc>
2960 char1,26,22,"rigel"           <kg>
2970 char1,30,17,"orion"           <aj>
2980 char1,30,18,"orion,ori"       <no>
2990 char1,30,19,"r:6h-7h"         <af>
3000 char1,30,20,"d:+20- -10"      <al>
3010 char1,0,3,"beteigeuze"        <mi>
3020 char1,23,4,"bellatrix"        <el>
3030 circle1,150,145,3:circle1,150
,120,2:circle1,180,120,2:circle1,1
54,140,1                           <po>
3040 circle1,157,146,1:circle1,160
,150,1                             <cp>
3050 paint1,150,145:paint1,150,120
:paint1,180,120:paint1,154,140     <lg>
3060 paint1,157,146:paint1,160,150 <da>
3070 circle1,153,146,7,10,,,165    <dc>
3080 char1,21,18,"m42"             <nm>
3090 gets$:ifs$=""then3090         <oh>
3100 goto10250                     <co>
3110 scnclr                        <em>
3120 graphic1,1                    <aa>
3130 circle1,140,100,5:circle1,77,
22,4:circle1,50,75,3:circle1,250,4
```

```
0,3                                                    <lk>
3140 circle1,244,50,2:circle1,254,
47,2:circle1,260,48,2:circle1,258,
43,2                                                   <ke>
3150 circle1,260,37,2:circle1,256,
36,1:circle1,244,40,1:circle1,249,
48,1                                                   <fk>
3160 paint1,140,100:paint1,77,22:p
aint1,50,75:paint1,250,40                              <np>
3170 paint1,244,50:paint1,254,47:p
aint1,260,48:paint1,258,43:paint1,
260,37                                                 <gl>
3180 paint1,256,36:paint1,244,40:p
aint1,249,48                                           <pd>
3190 circle1,140,140,2:circle1,133
,45,2:circle1,151,78,2:circle1,166
,95,2                                                  <po>
3200 circle1,170,100,2:circle1,156
,98,2:circle1,154,93,1:circle1,157
,103,1                                                 <kh>
3210 circle1,216,120,3                                 <fm>
3220 paint1,140,140:paint1,133,45:
paint1,151,78:paint1,166,95:paint1
,170,100                                               <bh>
3230 paint1,156,98:paint1,154,93:p
aint1,157,103:paint1,216,120                           <le>
3240 draw1,140,100to77,22:draw1,14
0,100to50,75:draw1,140,100to250,40                     <ik>
3250 draw1,140,100to140,140:draw1,
140,100to216,120                                       <fb>
3260 char1,29,19,"stier"                               <on>
3270 char1,29,20,"taurus,tau"                          <ko>
3280 char1,29,21,"r:3h-4h20m"                          <gn>
3290 char1,29,22,"d:0- +30"                            <hh>
3300 char1,7,13,"aldebaran"                            <ep>
3310 char1,28,2,"plejaden"                             <ci>
3320 gets$:ifs$=""then3090                             <oc>
3330 goto10250                                         <om>
3340 scnclr                                            <bk>
3350 graphic1,1                                        <gn>
3360 circle1,196,48,5:circle1,250,
60,4:circle1,154,89,3:circle1,180,
94,3                                                   <ck>
3370 circle1,130,120,4:circle1,146
,135,3:circle1,160,150,4:circle1,1
14,124,3                                               <bb>
3380 circle1,102,118,3:circle1,96,
158,4                                                  <fd>
3390 paint1,196,48:paint1,250,60:p
aint1,154,89:paint1,180,94:paint1,
130,120:paint1,146,135                                 <ge>
3400 paint1,160,150:paint1,114,124
:paint1,102,118:paint1,96,158                          <eb>
3410 draw1,96,158to130,120to146,13
5to160,150:draw1,130,120to154,89to
196,48to250,60                                         <cd>
3420 draw1,154,89to180,94                              <md>
3430 char1,28,19,"gr. hund"                            <np>
3440 char1,28,20,"canis maior"                         <bf>
3450 char1,28,21,"cma"                                 <me>
3460 char1,28,22,"r:5h40m-6h"                          <jk>
3470 char1,28,23,"d:-10- -30"                          <lk>
3480 char1,22,4,"sirius"                               <lj>
3490 gets$:ifs$=""then3090                             <ho>
3500 goto10250                                         <ih>
3510 scnclr                                            <gp>
3520 graphic1,1                                        <bf>
3530 circle1,120,100,5:circle1,160
,60,3:circle1,160,50,2:circle1,166
,45,1                                                  <ji>
3540 circle1,40,130,1:circle1,80,1
50,1:circle1,152,134:circle1,152,1
50,1                                                   <dp>
3550 paint1,120,100:paint1,160,60:
paint1,160,50:paint1,166,45:paint1
,40,130                                                <lg>
3560 paint1,80,150:paint1,152,150                      <em>
3570 draw1,120,100to160,60to160,50
to166,45                                               <io>
3580 draw1,40,130to80,150to120,100
to152,134to152,150                                     <dh>
3590 char1,28,19,"kl. hund"                            <np>
3600 char1,28,20,"canis minor"                         <hi>
3610 char1,28,21,"cmi"                                 <ko>
3620 char1,28,22,"r:8h-8h20m"                          <jc>
3630 char1,28,23,"d:0 - +10"                           <im>
3640 char1,7,12,"procyon"                              <fm>
3650 gets$:ifs$=""then3090                             <na>
3660 goto10250                                         <jp>
3670 scnclr                                            <la>
3680 graphic1,1                                        <cd>
3690 circle1,100,70,5:circle1,106,
60,3:circle1,108,50,2:circle1,170,
33,3                                                   <ao>
3700 circle1,178,14,2:circle1,90,8
6,2:circle1,50,126,2:circle1,157,1
14,2                                                   <ke>
3710 circle1,184,144,3:circle1,196
,150,2:circle1,90,72,2                                 <an>
3720 paint1,100,70:paint1,106,60:p
aint1,108,50:paint1,170,33:paint1,
178,14                                                 <ic>
3730 paint1,90,86:paint1,50,126:pa
int1,157,114:paint1,184,144:paint1
,196,150:paint1,90,72                                  <cp>
3740 draw1,100,70to106,60to108,50t
o170,33to178,14:draw1,100,70to90,8
6to50,126                                              <pf>
3750 draw1,100,70to157,114to184,14
4to196,150                                             <ne>
3760 char1,29,19,"adler"                               <na>
3770 char1,29,20,"aquila,aql"                          <hd>
3780 char1,29,21,"r:20h-21h"                           <mm>
3790 char1,29,22,"d:-10- +20"                          <oc>
3800 char1,5,8,"altair"                                <hb>
3810 gets$:ifs$=""then3090                             <jo>
```

```
3820 goto10250                                  <lh>
3830 scnclr                                     <pa>
3840 graphic1,1                                 <dc>
3850 circle1,140,20,4:circle1,170,
90,4:circle1,236,126,2:circle1,270
,170,3                                          <hp>
3860 circle1,224,50,3:circle1,228,
14,2:circle1,233,8,2:circle1,120,1
40,3                                            <jc>
3870 circle1,70,160,3:circle1,176,
42,2:circle1,185,50,2:circle1,100,
84,2                                            <nb>
3880 paint1,140,20:paint1,170,90:p
aint1,236,126:paint1,270,170:paint
1,224,50                                        <oo>
3890 paint1,228,14:paint1,233,8:pa
int1,120,140:paint1,70,160:paint1,
176,42                                          <nc>
3900 paint1,185,50:paint1,100,84                <fp>
3910 draw1,140,20to170,90to236,126
to270,170:draw1,170,90to224,50to22
8,14to233,8                                     <af>
3920 draw1,170,90to120,140to70,160              <dl>
3930 char1,15,1,"deneb"                         <hk>
3940 char1,3,3,"schwan"                         <cg>
3950 char1,3,4,"cygnus"                         <ob>
3960 char1,3,5,"cyg"                            <po>
3970 char1,3,6,"r:19h-22h"                      <id>
3980 char1,3,7,"d:+30- +60"                     <ed>
3990 char1,23,10,"schedir"                      <pe>
4000 gets$:ifs$=""then3090                      <ho>
4010 goto10250                                  <ih>
4020 scnclr                                     <gp>
4030 graphic1,1                                 <bf>
4040 circle,70,30,3:circle,82,64,2
:circle,90,84,3:circle,110,90,4:ci
rcle,186,88,3                                   <jd>
4050 circle,196,122,3:circle,208,1
23,2:circle,216,124,2:circle,246,1
00,3:circle,235,140,4                           <ho>
4060 circle,180,160,3:circle,160,6
0,3:circle,190,30,5:circle,200,20,
5                                               <di>
4070 paint1,70,30:paint1,82,64:pai
nt1,90,84:paint1,110,90:paint1,186
,88                                             <hm>
4080 paint1,196,122:paint1,208,123
:paint1,216,124:paint1,246,100:pai
nt1,235,140                                     <bj>
4090 paint1,180,160:paint1,160,60:
paint1,190,30:paint1,200,20                     <ei>
4100 draw1,70,30to82,64to90,84to11
0,90to186,88to196,122to208,123to21
6,124to246,100to235,140to180,160                <lp>
4110 draw1,186,88to160,60to190,30t
o200,20:draw1,186,88to246,100:draw
1,235,140to216,124                              <jf>
4120 char1,2,19,"centaur"                       <mf>
4130 char1,2,20,"centaurus,cen"                 <bg>
4140 char1,2,21,"r:11 40'-14 20'"               <cl>
4150 char1,2,22,"d:-30- -55"                    <oi>
4160 char1,26,2,"toliman"                       <di>
4170 char1,25,4,"hadar"                         <af>
4180 gets$:ifs$=""then3090                      <lc>
4190 goto10250                                  <jm>
4200 scnclr                                     <ni>
4210 graphic1,1                                 <gl>
4220 circle1,140,40,5:circle1,140,
120,4:circle1,180,90,4:circle1,112
,98,3                                           <op>
4230 circle1,110,25,2:circle1,190,
102,1:circle1,133,80,2:circle1,190
,138,2                                          <di>
4240 paint1,140,40:paint1,140,120:
paint1,180,90:paint1,112,98:paint1
,110,25                                         <nn>
4250 paint1,190,102:paint1,133,80:
paint1,190,138                                  <ep>
4260 draw1,140,40to140,120:draw1,1
80,90to112,98                                   <ad>
4270 char1,30,19,"kreuz"                        <ma>
4280 char1,30,20,"crux,cru"                     <hm>
4290 char1,30,21,"r:12-12,5h"                   <id>
4300 char1,30,22,"d:ca. -60"                    <lg>
4310 gets$:ifs$=""then3090                      <ad>
4320 goto10250                                  <df>
4330 scnclr                                     <nn>
4340 graphic1,1                                 <id>
4350 circle1,8,140,3:circle1,100,1
10,3:circle1,120,80,4:circle1,138,
86,3:circle1,173,70,3                           <ai>
4360 circle1,195,114,3:circle1,188
,160,4:circle1,200,170,3:circle1,2
20,40,2:circle1,226,118,3                       <ba>
4370 circle1,96,44,2:circle1,84,40
,2:circle1,60,14,2:circle1,60,4,2:
circle1,6,134,1                                 <fg>
4380 paint1,8,140:paint1,100,110:p
aint1,120,80:paint1,138,86:paint1,
173,70                                          <ge>
4390 paint1,195,114:paint1,188,160
:paint1,200,170:paint1,220,40:pain
t1,226,118                                      <gg>
4400 paint1,96,44:paint1,84,40:pai
nt1,60,14:paint1,60,4:paint1,6,134              <ai>
4410 draw1,8,140to100,110to120,80t
o138,86to173,70to195,114to188,160t
o200,170                                        <eg>
4420 draw1,173,70to220,40:draw1,19
5,114to226,118                                  <lb>
4430 draw1,120,80to96,44to84,40to6
0,14to60,4                                      <ck>
4440 char1,27,19,"schuetze"                     <pb>
4450 char1,27,20,"sagittarius"                  <fn>
4460 char1,27,21,"sgr"                          <li>
4470 char1,27,22,"r:18h-20h"                    <og>
```

```
4480 char1,27,23,"d:-15- -40"                    <me>
4490 gets$:ifs$=""then3090                       <cg>
4500 goto10250                                   <ja>
4510 scnclr                                      <eh>
4520 graphic1,1                                  <ni>
4530 circle1,160,160,5:circle1,204
,166,3:circle1,210,180,2:circle1,1
20,180,2                                         <cn>
4540 circle1,146,82,2:circle1,149,
54,3:circle1,112,36,2:circle1,84,8
0,2:circle1,118,102,3                            <gl>
4550 circle1,170,6,2:circle1,184,5
,2:circle1,176,10,2:circle1,180,27
,2                                               <lp>
4560 paint1,160,160:paint1,204,166
:paint1,210,180:paint1,120,180:pai
nt1,146,82:paint1,149,54                         <en>
4570 paint1,112,36:paint1,84,80:pa
int1,118,102:paint1,170,6:paint1,1
84,5:paint1,176,10:paint1,180,27                 <kf>
4580 draw1,120,180to160,160to204,1
66to210,180                                      <hg>
4590 draw1,160,160to146,82to149,54
to112,36to84,80to118,102to160,160                <nj>
4600 char1,30,19,"bootes"                        <bk>
4610 char1,30,20,"bootes"                        <mi>
4620 char1,30,21,"boo"                           <gi>
4630 char1,30,22,"r:13-14.5h"                    <ck>
4640 char1,30,23,"d:+10- +55"                    <jl>
4650 char1,21,19,"arctur"                        <pd>
4660 gets$:ifs$=""then3090                       <mf>
4670 goto10250                                   <bm>
4680 scnclr                                      <jm>
4690 graphic1,1                                  <ib>
4700 circle,280,160,3:circle,240,1
80,2:circle,184,154,2:circle,168,1
46,2:circle,146,130,3                            <in>
4710 circle,30,130,3:circle,40,40,
3:circle,140,40,4:circle,170,64,2:
circle,178,70,2                                  <ph>
4720 circle,240,60,2:circle,280,56
,2:circle,180,30,3:circle,230,10,2               <ne>
4730 paint1,280,160:paint1,240,180
:paint1,184,154:paint1,168,146:pai
nt1,146,130                                      <dm>
4740 paint1,30,130:paint1,40,40:pa
int1,140,40:paint1,170,64:paint1,1
78,70:                                           <cb>
4750 paint1,240,60:paint1,280,56:p
aint1,180,30:paint1,230,10                       <ok>
4760 draw1,280,160to240,180to184,1
54to168,146to146,130to30,130to40,4
0to140,40to170,64to178,70to240,60t
o280,56                                          <mh>
4770 draw1,140,40to180,30to230,10                <on>
4780 char1,28,11,"pegasus"                       <lp>
4790 char1,28,12,"pegasus"                       <en>
4800 char1,28,13,"peg"                           <nf>
4810 char1,28,14,"r:0-22 20'"                    <lk>
4820 char1,28,15,"d:10 - 30"                     <cc>
4830 draw1,140,40to146,130                       <ll>
4840 gets$:ifs$=""then3090                       <jj>
4850 goto10250                                   <cj>
4860 scnclr                                      <ae>
4870 graphic1,1                                  <mo>
4880 circle1,140,160,5:circle1,170
,120,2:circle1,224,100,3:circle1,2
58,96,2                                          <im>
4890 circle1,304,80,2:circle1,312,
52,2:circle1,270,46,2:circle1,200,
68,2                                             <jp>
4900 circle1,190,30,3:circle1,120,
100,2:circle1,72,95,2:circle1,10,9
0,2                                              <ik>
4910 paint1,140,160:paint1,170,120
:paint1,224,100:paint1,258,96                    <da>
4920 paint1,304,80:paint1,312,52:p
aint1,270,46:paint1,200,68                       <mc>
4930 paint1,190,30:paint1,120,100:
paint1,72,95:paint1,10,90                        <ic>
4940 draw1,140,160to170,120to224,1
00to258,96to304,80to312,52                       <ii>
4950 draw1,258,96to270,46:draw1,22
4,100to200,68to190,30                            <cm>
4960 draw1,170,120to120,100to72,95
to10,90                                          <kg>
4970 char1,25,19,"jungfrau"                      <ko>
4980 char1,25,20,"virgo,vir"                     <nf>
4990 char1,25,21,"r:11.40-14.40"                 <dh>
5000 char1,25,22,"d:+10- -20"                    <co>
5010 char1,13,21,"spica"                         <ec>
5020 gets$:ifs$=""then3090                       <ho>
5030 goto10250                                   <ih>
5040 scnclr                                      <gp>
5050 graphic1,1                                  <bf>
5060 circle1,160,40,5:circle1,184,
50,3:circle1,178,64,2:circle1,192,
68,2                                             <gk>
5070 circle1,200,120,3:circle1,140
,160,4:circle1,72,104,3:circle1,80
,56,4                                            <jb>
5080 paint1,160,40:paint1,184,50:p
aint1,178,64:paint1,192,68                       <oe>
5090 paint1,200,120:paint1,140,160
:paint1,72,104:paint1,80,56                      <gd>
5100 draw1,160,40to184,50to178,64t
o192,68to200,120to140,160to72,104t
o80,56to160,40                                   <ac>
5110 char1,29,19,"fuhrmann"                      <pb>
5120 char1,29,20,"auriga,aur"                    <dc>
5130 char1,29,21,"r:6h-7h"                       <fm>
5140 char1,29,22,"d:+30- +50"                    <lp>
5150 char1,17,3,"capella"                        <fn>
5160 gets$:ifs$=""then3090                       <lg>
5170 graphic0,0                                  <ji>
5180 printcl$:print                              <fl>
```

```
5190 printc3$"wollen sie nochein s
ternbild (j/n)";:inputa$          <em>
5200 ifa$="j"thengoto2380         <dp>
5210 ifa$="n"thengoto110          <cm>
5220 ifa$<>chr$(74)ora$<>chr$(78)t
hen4860                           <la>
5230 scnclr                       <on>
5240 graphic1,1                   <aj>
5250 circle1,80,20,4:circle1,162,8
2,3:circle1,200,100,3:circle1,196,
150,4:circle1,120,126,2           <ep>
5260 circle1,90,110,2:circle1,54,7
0,4:circle1,60,64,2:circle1,212,10
0,3                               <pd>
5270 paint1,80,20:paint1,162,82:pa
int1,200,100:paint1,196,150:paint1
,120,126                          <pe>
5280 paint1,90,110:paint1,54,70:pa
int1,60,64:paint1,212,100         <lh>
5290 draw1,80,20to162,82to200,100t
o196,150to120,126to90,110to54,70to
60,64                             <ml>
5300 char1,29,19,"zwillinge"      <mo>
5310 char1,29,20,"gemini,gem"     <lk>
5320 char1,29,21,"r:6h-8h"        <oa>
5330 char1,29,22,"d:+10- +35"     <fb>
5340 char1,11,2,"castor"          <gk>
5350 char1,0,8,"pollux"           <ee>
5360 gets$:ifs$=""then3090        <oc>
5370 goto10250                    <ke>
5380 scnclr                       <bk>
5390 graphic1,1                   <gn>
5400                              <cn>
5420 scnclr                       <gk>
5430 graphic1,1                   <on>
5440 circle,162,62,2:circle,232,68
,4:circle,220,106,3:circle,163,100
,3                                <ga>
5450 circle,112,42,4:circle,63,37,
4:circle,58,32,1:circle,25,56,4   <bm>
5460 paint1,162,62:paint1,232,68:p
aint1,220,106:paint1,163,100      <og>
5470 paint1,112,42:paint1,63,37:pa
int1,58,32:paint1,25,56           <ho>
5480 draw1,162,62to232,68to220,106
to163,100to162,62to112,42to63,37to
25,56                             <hf>
5490 draw1,236,60to244,40:draw1,24
4,40to246,46:draw1,244,40to238,44 <bl>
5500 draw1,16,64to10,70to4,80to4,9
0to6,86:draw1,4,90to2,86          <he>
5510 char1,28,19,"gr.wagen"       <me>
5520 char1,28,20,"ursa maior"     <fn>
5530 char1,28,21,"uma"            <el>
5540 char1,28,22,"r:8h-14h"       <fd>
5550 char1,28,23,"d:40 - 70"      <lp>
5560 char1,26,2,"richtung"        <ll>
5570 char1,26,3,"himmelsnordpol"  <cj>
5580 char1,2,11,"zum"             <gn>
5590 char1,2,12,"arctur (boo)"    <ko>
5600 gets$:ifs$=""then3090        <bn>
5610 goto10250                    <ai>
5620 scnclr                       <pk>
5630 graphic1,1                   <go>
5640 circle,140,85,2:circle,141,14
0,4:circle,101,138,3:circle,104,80
,2                                <ji>
5650 circle,160,40,2:circle,198,20
,1:circle,234,14,4                <ii>
5660 paint1,140,85:paint1,141,140:
paint1,101,138:paint1,104,80      <mk>
5670 paint1,160,40:paint1,198,20:p
aint1,234,14                      <ba>
5680 draw1,140,85to141,140to101,13
8to104,80to140,85to160,40to198,20t
o234,14                           <bm>
5690 draw1,232,24to236,24:draw1,23
4,22to234,26                      <kf>
5700 char1,28,19,"kl.wagen"       <jg>
5710 char1,28,20,"ursa minor"     <ci>
5720 char1,28,21,"umi"            <ii>
5730 char1,28,22,"r:12h-18h "     <cg>
5740 char1,28,23,"d:70 - 90"      <am>
5750 char1,30,1,"polaris"         <pl>
5760 char1,30,3,"himmels-"        <mh>
5770 char1,30,4,"nordpol"         <fa>
5780 gets$:ifs$=""then3090        <kb>
5790 goto10250                    <ee>
5800 scnclr                       <gf>
5810 graphic1,1                   <mf>
5820 circle,160,60,5:circle,150,70
,1:circle,142,110,2:circle,122,112
,2:circle,128,80,2                <hg>
5830 circle,134,104,5,3,,,145     <nm>
5840 paint1,160,60:paint1,150,70:p
aint1,142,110:paint1,122,112:paint
1,128,80                          <jk>
5850 draw1,160,60to150,70to142,110
to122,112to128,80to150,70         <og>
5860 char1,28,19,"leier"          <dg>
5870 char1,28,20,"lyra,lyr"       <en>
5880 char1,28,21,"r:17 20'-18"    <mc>
5890 char1,28,22,"d:30 - 40"      <fj>
5900 char1,22,7,"wega"            <na>
5910 char1,12,12,"m57"            <ke>
5920 gets$:ifs$=""then3090        <cl>
5930 goto10250                    <fj>
5940 scnclr                       <ho>
5950 graphic1,1                   <jc>
5960 circle,180,100,3:circle,160,1
10,4:circle,150,90,4:circle,126,96
,3:circle,110,74,2                <pb>
5970 paint1,180,100:paint1,160,110
:paint1,150,90:paint1,126,96:paint
1,110,74                          <mh>
5980 char1,28,19,"cassiopeia"     <mh>
```

```
5990 char1,28,20,"cassiopeia"                <bl>
6000 char1,28,21,"cas"                       <lk>
6010 char1,28,22,"r:0h - 4h"                 <ef>
6020 char1,28,23,"d:50 - 70"                 <ae>
6030 char1,7,5,"das himmels - w"             <mj>
6040 char1,17,15,"schedir"                   <di>
6050 draw1,180,100to160,110to150,9
0to126,96to110,74                            <df>
6060 gets$:ifs$=""then3090                   <ie>
6070 goto10250                               <hc>
6080 scnclr                                  <jh>
6090 graphic1,1                              <fj>
6100 circle,40,60,4:circle,120,100
,4:circle,180,120,3:circle,240,130
,4:circle,130,80,2                           <an>
6110 circle,146,50,7,5,,,55:circle
,156,40,5,3,,,95:circle,136,60,5,3
,,,140                                       <en>
6120 char1,28,19,"andromeda"                 <jk>
6130 char1,28,20,"andromeda"                 <nc>
6140 char1,28,21,"and"                       <gp>
6150 char1,28,22,"r:0h - 2h"                 <ok>
6160 char1,28,23,"d:30 - 50"                 <gn>
6170 paint1,40,60:paint1,120,100:p
aint1,180,120:paint1,240,130:paint
1,130,80                                     <ne>
6180 draw1,40,60to120,100to180,120
to240,130:draw1,120,100to130,80to1
44,56                                        <lp>
6190 char1,19,7,"m31(andromedanebe
l)"                                          <cb>
6200 char1,21,4,"ngc205"                     <kd>
6210 char1,13,8,"m32"                        <ib>
6220 gets$:ifs$=""then3090                   <lc>
6230 goto10250                               <jm>
6240 scnclr                                  <ni>
6250 graphic1,1                              <gl>
6260 circle,210,40,2:circle,204,60
,3:circle,190,90,2:circle,152,138,
2                                            <cm>
6270 circle,138,150,2:circle,76,12
0,2:circle,50,94,1:circle,36,74,3            <md>
6280 circle,48,78,2:circle,80,78,2
:circle,110,82,2:circle,160,85,2             <lb>
6290 paint1,210,40:paint1,204,60:p
aint1,190,90:paint1,152,138                  <on>
6300 paint1,138,150:paint1,76,120:
paint1,50,94:paint1,36,74                    <bn>
6310 paint1,48,78:paint1,80,78:pai
nt1,110,82:paint1,160,85                     <pl>
6320 draw1,210,40to204,60to190,90t
o152,138to138,150to76,120to50,94to
36,74to48,78to80,78to110,82to160,8
5to204,60                                    <lo>
6330 char1,28,19,"steinbock"                 <nb>
6340 char1,28,20,"capricornus"               <ff>
6350 char1,28,21,"cap"                       <hm>
6360 char1,28,22,"r:20 - 22"                 <pk>
6370 char1,28,23,"d:-10- +30"                <bg>
6380 gets$:ifs$=""then3090                   <oc>
6390 goto10250                               <ke>
6400 scnclr                                  <bk>
6410 graphic1,1                              <gn>
6420 circle,310,56,2:circle,300,56
,2:circle,276,70,2:circle,244,40,3
:circle,196,10,3                             <jh>
6430 circle,172,16,2:circle,112,46
,2:circle,116,70,2:circle,114,86,3
:circle,80,120,2                             <gp>
6440 circle,66,46,2:circle,20,80,2
:circle,166,10,2:circle,158,8,2              <he>
6450 paint1,310,56:paint1,300,56:p
aint1,276,70:paint1,244,40:paint1,
196,10                                       <lg>
6460 paint1,172,16:paint1,112,46:p
aint1,116,70:paint1,114,86:paint1,
80,120                                       <no>
6470 paint1,66,46:paint1,20,80:pai
nt1,166,10:paint1,158,8                      <de>
6480 draw1,310,56to300,56to276,70t
o244,40to196,10to172,16to112,46to1
16,70to114,86to80,120                        <id>
6490 draw1,112,46to66,46to20,80:dr
aw1,172,16to166,10to158,8                    <pm>
6500 char1,24,19,"wassermann"                <fe>
6510 char1,24,20,"aquarius,aqr"              <hi>
6520 char1,24,21,"r:21 20'-22 40'"           <le>
6530 char1,24,22,"d:0 - (-20)"               <ea>
6540 gets$:ifs$=""then3090                   <bf>
6550 goto10250                               <ae>
6560 scnclr                                  <fl>
6570 graphic1,1                              <hh>
6580 circle,172,50,2:circle,180,75
,2:circle,192,88,2                           <li>
6590 circle,184,104,3:circle,170,1
06,2:circle,150,96,2                         <jl>
6600 paint1,172,50:paint1,180,75:p
aint1,192,88:paint1,184,104:paint1
,170,106:paint1,150,96                       <gn>
6610 draw1,172,50to180,75to192,88t
o184,104to170,106to150,96                    <kd>
6620 char1,24,19,"nordl. krone"              <ho>
6630 char1,24,20,"corona borealis"           <in>
6640 char1,24,21,"crb"                       <pf>
6650 char1,24,22,"r:15 40' - 16"             <he>
6660 char1,24,23,"d:25 - 40"                 <ek>
6670 char1,24,13,"gemma"                     <ic>
6680 gets$:ifs$=""then3090                   <lp>
6690 goto10250                               <bh>
6700 scnclr                                  <he>
6710 graphic1,1                              <do>
6720 circle,224,64,3:circle,216,44
,2:circle,176,60,2:circle,166,81,3
:circle,186,110,2                            <dj>
6730 circle,180,140,4:circle,80,12
0,2:circle,20,120,3:circle,60,90,3           <ld>
```

```
6740 paint1,224,64:paint1,216,44:p
aint1,176,60:paint1,166,81:paint1,
186,110                          <cl>
6750 paint1,180,140:paint1,80,120:
paint1,20,120:paint1,60,90       <fe>
6760 draw1,224,64to216,44to176,60t
o166,81to186,110to180,140to80,120t
o20,120to60,90to166,81           <md>
6770 char1,28,19,"loewe"         <pi>
6780 char1,28,20,"leo,leo"       <eg>
6790 char1,28,21,"r:9 40'-12"    <mn>
6800 char1,28,22,"d:10 - 30"     <ed>
6810 char1,15,18,"regulus"       <bc>
6820 gets$:ifs$=""then3090       <ph>
6830 goto10250                   <dh>
6840 scnclr                      <in>
6850 graphic1,1                  <ak>
6860 circle,62,102,3:circle,56,102
,3:circle,46,112,3:circle,40,120,3
:circle,60,140,4                 <kc>
6870 circle,110,150,3:circle,140,1
40,2:circle,146,120,3:circle,152,1
00,3:circle,170,70,3             <gd>
6880 circle,180,60,5:circle,194,52
,3:circle,206,20,2:circle,220,22,3 <dd>
6890 circle,224,40,3:circle,226,60
,3:circle,230,80,2:circle,30,102,3 <gi>
6900 paint1,62,102:paint1,56,102:p
aint1,46,112:paint1,40,120:paint1,
60,140:paint1,110,150            <bl>
6910 paint1,140,140:paint1,146,120
:paint1,152,100:paint1,170,70:pain
t,180,60:paint1,194,52           <ef>
6920 paint1,206,20:paint1,220,22:p
aint1,224,40:paint1,226,60:paint1,
230,80:paint1,30,102             <kf>
6930 draw1,56,102to46,112to40,120t
o60,140to110,150to140,140to146,120
to152,100to170,70to180,60to194,52t
o206,20                          <ne>
6940 draw1,194,52to220,22:draw1,19
4,52to224,40:draw1,194,52to226,60:
draw1,194,52to230,80:draw1,40,120t
o30,102                          <lb>
6950 char1,28,19,"skorpion"      <kl>
6960 char1,28,20,"scorpius"      <ne>
6970 char1,28,21,"sco"           <jh>
6980 char1,28,22,"r:16-17"       <cl>
6990 char1,28,23,"d:-10- -45"    <dc>
7000 char1,15,5,"antares"        <ip>
7010 gets$:ifs$=""then3090       <fj>
7020 goto10250                   <ag>
7030 scnclr                      <al>
7040 print:print" zuerst sehen sie
 eine aufstellung"               <ah>
7050 print:print" der wichtigsten
sternbilder mit lat."            <cd>
7060 print:print" namen und die da
zugehoerige abk. ,"              <hm>
7070 print:print" da diese bei der
 nachfolgenden ta-"              <mj>
7080 print:print" elle benoetigt w
erden."                          <jn>
7090 print:print:print:print:print
" taste druecken "               <ih>
7100 getq$:ifq$=""thengoto7100   <cf>
7110 scnclr                      <kl>
7120 print" dt.name          lat.n
ame       abk."                  <mi>
7130 print:print" adler
 aquila          aql"            <df>
7140 print" fuhrmann         aurig
a          aur"                  <kb>
7150 print" ochsentreiber    boote
s          boo"                  <le>
7160 print" gr. hund         canis
 maior     cma"                  <id>
7170 print" kl. hund         canis
 minor     cmi"                  <cd>
7180 print" kiel d. schiffs  carin
a          car"                  <og>
7190 print" cassiopeia       cassi
opeia      cas"                  <el>
7200 print" zentaur          centa
ur         cen"                  <jp>
7210 print" noerdl. krone    coron
a borealis crb"                  <ni>
7220 print" kreuz d. suedens crux
           cru"                  <ng>
7230 print" schwan           cygnu
s          cyg"                  <pl>
7240 print" zwillinge        gemin
i          gem"                  <oj>
7250 print" loewe            leo
           leo"                  <lg>
7260 print" leier            lyra
           lyr"                  <ko>
7270 print" orion            orion
           ori"                  <pd>
7280 print" pegasus          pegas
us         peg"                  <me>
7290 print" schuetze         sagit
tarius     sgr"                  <gp>
7300 print" skorpion         scorp
ius        sco"                  <hg>
7310 print" stier            tauru
s          tau"                  <gl>
7320 print" gr. wagen        ursa
maior      uma"                  <bh>
7330 print" kl. wagen        ursa
minor      umi"                  <im>
7340 print" jungfrau         virgo
           vir"                  <km>
7350 getq$:ifq$=""then7350       <ph>
7360 scnclr                      <kb>
7370 print" stern      sternbild
```

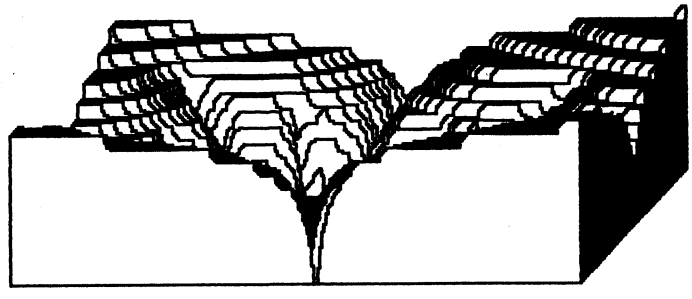
```
spektr.    hell."                    <lc>
7380 print:print" sirius     gr.hu
nd      a0      -1.37"               <ae>
7390 print" canopus    kiel
  f0      -0.86"                     <kb>
7400 print" toliman    centaur
  g2       0.06"                     <bj>
7410 print" wega       leier
  a1       0.14"                     <kb>
7420 print" capella    fuhrmann
  g5       0.21"                     <bk>
7430 print" arctur     bootes
  k2       0.24"                     <pa>
7440 print" rigel      orion
  b8       0.34"                     <kk>
7450 print" beteigeuze orion
  m2    0.4-1.3"                     <ki>
7460 print" procyon    kl.hund
  f3       0.48"                     <ei>
7470 print" achernar   eridanus
  b9       0.68"                     <jn>
7480 print" hadar      centaur
  b3       0.86"                     <kd>
7490 print" altair     adler
  a7       0.89"                     <ig>
7500 print" acrux      kreuz
  b1       1.05"                     <mi>
7510 print" aldebaran  stier
  k5       1.06"                     <in>
7520 print" spica      jungfrau
  b2       1.21"                     <gp>
7530 print" pollux     zwillinge
  k0       1.21"                     <gg>
7540 print" antares    skorpion
  m0       1.22"                     <el>
7550 print" formalhaut sued.fisch
  a3       1.29"                     <cg>
7560 print" deneb      schwan
  a2       1.33"                     <he>
7570 print" regulus    loewe
  b8       1.34"                     <lk>
7580 getq$:ifq$=""then7580           <kc>
7590 goto110                         <nh>
7600 sleep5                          <lb>
7610 scnclr                          <jh>
7620 graphic1,1                      <fj>
7630 circle,1,100,2                  <gb>
7640 circle1,1,100, 9,6:circle,9,1
00,2:paint1,9,100                    <ni>
7650 circle,1,100,27,18:circle,27,
100,2:paint1,27,100                  <nh>
7660 circle,1,100,35,23:circle,35,
100,2:paint1,35,100                  <pn>
7670 circle,1,100,106,69:circle1,1
06,100,2:paint1,106,100              <ic>
7680 circle,1,100,317,215:circle,3
17,100,2:paint1,317,100              <bf>
7690 circle,1,100,200,140:circle1,
200,100,2:paint1,200,100             <am>
7700 char1,21,11,"jupiter"           <pm>
7710 char1,1,13,"m"                  <bc>
7720 char1,1,10,"v"                  <md>
7730 char1,4,13,"e"                  <jn>
7740 char1,12,10,"mars"              <oj>
7750 char1,33,12,"saturn"            <hm>
7760 char1,27,4,"m=merkur"           <lg>
7770 char1,27,5,"v=venus"            <hc>
7780 char1,27,6,"e=erde"             <fn>
7790 char1,27,7,"W=sonne"            <mf>
7800 char1,10,0,"die inneren plane
ten"                                 <cn>
7810 getq$:ifq$=""then7810           <pm>
7820 scnclr                          <dn>
7830 circle1,1,100,2                 <hp>
7840 circle,1,100,12,7:circle,12,1
00,2:paint1,12,100                   <ma>
7850 circle,1,100, 40,25:circle1,4
0,100,2:paint1,40,100                <lj>
7860 circle,1,100,120, 75:circle,1
20,100,2:paint1,120,100              <jf>
7870 circle,1,100,160,100:circle,1
60,100,2:paint1,160,100              <ki>
7880 circle,1,100,317,210:circle,3
17,100,2:paint1,317,100              <fn>
7890 char1,34,12,"pluto"             <ok>
7900 char1,21,12,"neptun"            <ob>
7910 char1,12,10,"uranus"            <fk>
7920 char1,3,9,"saturn"              <ge>
7930 char1,0,13,"erde"               <ee>
7940 char1,10,0,"die aeusseren pla
neten"                               <fm>
7950 getq$:ifq$=""then7950           <ng>
7960 graphic0                        <mi>
7970 goto110                         <kp>
7980 scnclr                          <ho>
7990 print"                galaxie
n"                                   <nh>
8000 print:print" galaxien sind an
sammlungen von mil-"                 <mh>
8010 print" liarden von sternen. d
ie ausmasse einer"                   <ba>
8020 print" solchen galaxie sind u
nvorstellbar"                        <mc>
8030 print" gross. unsere galaxie
(milchstrasse)"                      <hc>
8040 print" hat einen laengsdurchm
esser von ca."                       <kn>
8050 print" 100000 lichtjahren.(11
j=9460600000000"                     <jl>
8060 print" km)die galaxien gliede
rn sich prozent."                    <ag>
8070 print" in folgende typen :"     <cj>
8080 print:print:print" 1- ellipti
sche nebel (18%)"                    <ij>
8090 print" 2- normale spiralen  (
78%)"                                <lp>
```

```
8100 print" 3- irregulaere nebel (
 4%)"                                <gh>
8110 print:print:input" nr.:";t      <mi>
8120 ift<1ort>3then7980              <lj>
8130 ontgoto8140,8260,8390           <em>
8140 scnclr                          <lp>
8150 graphic1,1                      <jm>
8160 circle,60,80,20:circle,140,80
,30,12                               <aj>
8170 circle,240,80,40,5              <ob>
8180 char1,10,2,"elliptische nebel
"                                    <ni>
8190 char1,6,6,"e0"                  <kk>
8200 char1,17,6,"e3"                 <kg>
8210 char1,29,6,"e7"                 <me>
8220 char1,1,16,"die unterklassen
der ell. nebel e0-e7"                <nj>
8230 char1,1,18,"geben die staerke
 der abplattung an."                 <ob>
8240 char1,1,20,"e0 mit 1:1 und e7
 mit 3:1"                            <jk>
8250 goto8490                        <ai>
8260 scnclr                          <la>
8270 print:print"          spiraln
ebel"                                <ib>
8280 print:print" spiralnebel sehe
n optisch aus, wie "                 <jb>
8290 print:print" der name schon s
agt, wie spiralen."                  <fk>
8300 print: print" der bekannteste
 spiralnebel ist wohl"               <on>
8310 print:print" der andromedaneb
el (m31) im sternbild"               <lj>
8320 print:print" andromeda. unser
e galaxie ist ebenfalls"             <ma>
8330 print:print" ein spiralnebel.
 die spiralnebelarme"                <ka>
8340 print:print" liegen je nach s
taerke der drehung"                  <ci>
8350 print:print" mehr oder wenige
r an den kern des"                   <mf>
8360 print:print" nebels an."        <ab>
8370 print:print:print:print" tast
e druecken !"                        <mg>
8380 goto8490                        <fj>
8390 scnclr                          <lf>
8400 print:print"            irregu
laere nebel"                         <ej>
8410 print:print:print" irr. nebel
 sind nun nebel, die sich"           <od>
8420 print:print" weder in die kla
ssifikation der"                     <bh>
8430 print:print" ell. nebel noch
in die der spiralnebel"              <in>
8440 print:print" einordnen lassen
. ihre namen richten"                <od>
8450 print:print" sich dann einfac
h nach ihrem aussehen."              <fb>
8460 print:print" z.b. der nordame
rika-nebel im stern-"                <ll>
8470 print:print" bild schwan.       <nf>
8480 goto8490                        <of>
8490 getq$:ifq$=""then8490           <mf>
8500 graphic0                        <eb>
8510 scnclr                          <kg>
8520 printc4$" h - hauptmenue"       <kb>
8530 printc4$" g - andere galaxie"   <ck>
8540 getg$:ifg$=""then8540           <pf>
8550 ifg$="h"then110                 <jn>
8560 ifg$="g"then7980                <gj>
8570 ifg$<>"h"org$<>"g"then8510      <ae>
8580 scnclr                          <dd>
8590 print:print:print" wenn man z
u best. zeiten einige"               <kc>
8600 print:print" sternbilder zusa
mmenfasst, erhaelt"                  <oj>
8610 print:print" man bestimmte, m
arkante gebilde, die"                <ph>
8620 print:print" dann nur aus den
 hellsten sternen der"               <ee>
8630 print:print" beteiligten ster
nbilder bestehen."                   <be>
8640 print:print:print:print" bitt
e taste druecken!"                   <ec>
8650 getq$:ifq$=""then8650           <he>
8660 scnclr                          <nd>
8670 print:print"  da bei jeder st
ernkonstellation ca."                <af>
8680 print:print"  120-150 sterne
gezeichnet werden, dau-"             <le>
8690 print:print"  ert es einige z
eit, bis das ganze"                  <df>
8700 print:print"  bild fertiggest
ellt ist. wenn die"                  <lc>
8710 print:print"  schrift erschei
nt ist der vorgang be-"              <hn>
8720 print:print"  endet und sie k
oennen mit einer bel."               <jm>
8730 print:print"  taste das progr
amm weiterlaufen "                   <fa>
8740 print:print"  lassen."           <lk>
8750 print:print:print"  bitte tas
te druecken !"                       <fo>
8760 getq$:ifq$=""then8760           <ln>
8770 scnclr                          <la>
8780 print:print:print:print"  a -
 das wintersechseck"                 <ak>
8790 print:print"  b - das sommerd
reieck"                              <ie>
8800 print:print"  c - der fruehli
ngshimmel"                           <jo>
8810 print:print"  d - der herbsth
immel"                               <eh>
8820 print:print:input"  ihre wahl
";j$                                 <pg>
8830 ifj$="a"then8880                <ni>
```

```
8840 ifj$="b"then9210                <fh>
8850 ifj$="c"then9500                <ih>
8860 ifj$="d"then9800                <co>
8870 ifj$<>"a"andj$<>"b"andj$<>"c"
andj$<>"d"then8770                   <ip>
8880 scnclr                          <in>
8890 graphic1,1                      <ak>
8900 fast                            <lm>
8910 circle,202,7,3:circle,180,16,
1:circle,194,42,1:circle,214,40,1:
circle,214,14,1                      <fk>
8920 circle,228,52,1:circle,230,72
,1:circle,270,74,3:circle,306,68,1
:circle,190,100,1                    <ae>
8930 circle,218,104,1:circle,199,1
35,1:circle,204,132,1:circle,208,1
30,1:circle,188,154,1                <gn>
8940 circle,220,156,3:circle,148,1
62,1:circle,132,170,3:circle,118,1
86,1:circle,126,194,1                <gi>
8950 circle,104,193,1:circle,100,1
02,3:circle,111,96,1:circle,110,11
0,1:circle,114,60,3                  <fb>
8960 circle,124,50,3:circle,129,76
,1:circle,152,68,1:circle,151,90,1
:circle,163,75,1                     <cc>
8970 paint1,202,7:paint1,270,74:pa
int1,220,156:paint1,132,170:paint1
,100,102:paint1,163,75:paint1,124,
50:paint1,114,60                     <ao>
8980 paint1,180,16:paint1,194,42:p
aint1,214,40:paint1,214,14           <hg>
8990 paint1,228,52:paint1,230,72:p
aint1,306,68:paint1,190,100          <ok>
9000 paint1,218,104:paint1,199,135
:paint1,204,132:paint1,208,130:pai
nt1,188,154                          <jg>
9010 paint1,148,162:paint1,118,186
:paint1,126,194:paint1,104,193:pai
nt1,111,96                           <if>
9020 paint1,110,110:paint1,129,76:
paint1,152,68:paint1,151,90:paint1
,163,75                              <pm>
9030 paint1,114,60                   <hf>
9040 fori=1to75                      <mj>
9050 x=int(rnd(x)*319)+1             <ni>
9060 y=int(rnd(x)*200)+1             <pf>
9070 circle1,x,y                     <ch>
9080 nexti                           <lm>
9090 char1,26,0,"capella (aur)"      <ho>
9100 char1,31,7,"aldebaran"          <dk>
9110 char1,12,22,"sirius (cma) "     <lm>
9120 char1,28,18,"rigel (ori)        <jm>
9130 char1,6,11,"procyon"            <da>
9140 char1,5,7,"pollux"              <bl>
9150 char1,3,5,"castor (gem)"        <eo>
9160 char1,7,12,"(cmi)"              <kb>
9170 char1,35,6,"(tau)"              <ib>
9180 slow                            <cb>
9190 getq$:ifq$=""then9190           <mm>
9200 goto10170                       <on>
9210 scnclr                          <cc>
9220 graphic1,1                      <lp>
9230 fast                            <oj>
9240 circle,96,52,1:circle,120,60,
1:circle,160,42,3:circle,156,54,1:
circle,158,106,1                     <hd>
9250 paint1,96,52:paint1,120,60:pa
int1,160,42:paint1,156,54:paint1,1
58,106                               <ib>
9260 circle,186,54,1:circle,194,46
,1:circle,208,40,1:circle,244,52,3
:circle,236,58,1                     <fm>
9270 paint1,186,54:paint1,194,46:p
aint1,208,40:paint1,244,52:paint1,
236,58                               <lo>
9280 circle,234,68,1:circle,242,64
,1:circle,82,168,1:circle,112,144,
3:circle,117,141,1                   <mh>
9290 paint1,234,68:paint1,242,64:p
aint1,82,168:paint1,112,144:paint1
,117,141                             <hp>
9300 circle,158,137,1:circle,138,1
84,1:circle,164,196,1:circle,168,1
99,1:circle,162,134,1                <ea>
9310 paint1,158,137:paint1,138,184
:paint1,164,196:paint1,168,199:pai
nt1,162,134                          <ka>
9320 fori=1to75                      <kk>
9330 x=int(rnd(x)*320)+1             <gp>
9340 y=int(rnd(y)*200)+1             <ha>
9350 circle1,x,y                     <gk>
9360 nexti                           <ca>
9370 char1,31,5,"wega "              <cm>
9380 char1,16,3,"deneb (schwan)"     <ee>
9390 char1,26,19,"sommerdreieck"     <ok>
9400 char1,26,20,"zu sehen :"        <io>
9410 char1,26,21,"juli-sep."         <pl>
9420 char1,26,22,"von 20 - 23 h"     <oh>
9430 char1,26,23,"ort:im zenit"      <ji>
9440 char1,7,18,"altair"             <do>
9450 char1,7,19,"(aql)"              <pa>
9460 char1,31,6,"(lyr)"              <lk>
9470 slow                            <kk>
9480 getq$:ifq$=""then9480           <ni>
9490 goto10170                       <oc>
9500 scnclr                          <gk>
9510 graphic1,1                      <on>
9520 fast                            <hg>
9530 circle,144,30,1:circle,124,44
,1:circle,130,68,1:circle,118,106,
1:circle,158,50,1                    <ee>
9540 circle,146,66,1:circle,140,10
0,3:circle,150,106,1:circle,140,12
6,1:circle,160,128,1                 <ik>
9550 circle,146,161,1:circle,160,1
```

```
50,3:circle,178,140,1:circle,200,1
35,1:circle,192,118,1                <fi>
9560 circle,190,100,1:circle,228,1
36,1:circle,240,116,1:circle,253,1
38,1:circle,250,100,1                <kp>
9570 circle,262,80,1:circle,274,90
,1:circle,304,86,3:circle,300,70,1
:circle,294,64,1                     <be>
9580 circle,290,54,1:circle,300,40
,1:circle,310,50,1                   <pl>
9590 paint1,118,106:paint1,158,50:
paint1,146,66:paint1,140,100:paint
1,150,106:paint1,140,126:paint1,16
0,128:paint1,146,161                 <in>
9600 paint1,160,150:paint1,178,140
:paint1,200,135:paint1,192,118:pai
nt1,190,100:paint1,228,136:paint1,
240,116:paint1,253,138               <hc>
9610 paint1,250,100:paint1,262,80:
paint1,274,90:paint1,304,86:paint1
,300,70:paint1,294,64:paint1,290,5
4:paint1,300,40:paint1,310,50        <jc>
9620 fori=1to75                      <bm>
9630 x=int(rnd(x)*319)+1             <db>
9640 y=int(rnd(x)*200)+1             <pe>
9650 circle1,x,y                     <lf>
9660 nexti                           <nd>
9670 char1,21,19,"spica (vir)"       <ph>
9680 char1,32,12,"regulus"           <ji>
9690 char1,32,13,"(leo)"             <om>
9700 char1,10,8,"gemma"              <fm>
9710 char1,10,9,"(crb)"              <ci>
9720 char1,17,11,"arctur (boo)"      <bg>
9730 char1,2,19,"fruehlingshimmel"   <io>
9740 char1,2,20,"zu sehen :"         <ek>
9750 char1,2,21,"apr.-juni"          <bm>
9760 char1,2,22,"20h - 00h"          <no>
9770 slow                            <gd>
9780 getq$:ifq$=""then9780           <mj>
9790 goto10170                       <cb>
9800 scnclr                          <me>
9810 graphic1,1                      <me>
9820 fast                            <cl>
9830 circle,10,70,3:circle,4,90,1:
circle,7,154,3:circle,142,13,1:cir
cle,152,19,1                         <cc>
9840 circle,160,13,1:circle,166,19
,1:circle,172,10,1:circle,75,55,1:
circle,100,70,2:                     <cb>
9850 circle,110,60,1:circle,130,92
,1:circle,160,100,2:circle,160,140
,1:circle,204,100,1                  <be>
9860 circle,206,140,1:circle,226,9
4,1:circle,240,82,1:circle,220,144
,1:circle,236,154,1                  <bi>
9870 circle,260,152,1:circle,272,1
34,1:circle,260,40,3:circle,274,20
,1:circle,280,40,1                   <bp>
9880 circle,278,56,1:circle,272,66
,1:circle,312,34,1                   <cl>
9890 paint1,10,70:paint1,4,90:pain
t1,7,154:paint1,142,13:paint1,152,
19                                   <ah>
9900 paint1,160,13:paint1,166,19:p
aint1,172,10:paint1,75,55:paint1,1
00,70:                               <ph>
9910 paint1,110,60:paint1,130,92:p
aint1,160,100:paint1,160,140:paint
1,204,100                            <hh>
9920 paint1,206,140:paint1,226,94:
paint1,240,82:paint1,220,144:paint
1,236,154                            <mg>
9930 paint1,260,152:paint1,272,134
:paint1,260,40:paint1,274,20:paint
1,280,40                             <cd>
9940 paint1,278,56:paint1,272,66:p
aint1,312,34                         <an>
9950 circle,120,40,8,4,,,55          <oc>
9960 circle,44,152:circle,48,152:c
ircle,46,154:circle,45,153:circle,
42,151:circle,45,152:circle,44,154   <ao>
9970 fori=1to70                      <gh>
9980 x=int(rnd(x)*320)+1             <ck>
9990 y=int(rnd(y)*200)+1             <mi>
10000 circle,x,y                     <hh>
10010 nexti                          <fb>
10020 char1,27,3,"deneb"             <nn>
10030 char1,27,4,"(cyg)"             <le>
10040 char1,16,5,"andromeda-"        <ee>
10050 char1,16,6,"nebel"             <ao>
10060 char1,1,7,"capella"            <hg>
10070 char1,1,21,"aldebaran"         <ap>
10080 char1,6,19,"plejaden"          <jn>
10090 char1,20,18,"pegasus"          <bn>
10100 draw1,142,13to152,19to160,13
to166,19to172,10:draw1,75,55to100,
70to130,92to160,100to160,140to206,
140to204,100to160,100                <ja>
10110 draw1,204,100to226,94to240,8
2:draw1,206,140to220,144to236,154t
o260,152to272,134                    <ib>
10120 draw1,260,40to280,40to312,34
:draw1,274,20to280,40to278,56to272
,66                                  <he>
10130 draw1,0,62to10,70to4,90to0,9
2:draw1,0,122to7,154to0,152:draw1,
7,154to42,152                        <kc>
10140 slow                           <db>
10150 getq$:ifq$=""then10150         <jc>
10160 goto10170                      <pk>
10170 scnclr                         <kl>
10180 graphic0                       <jl>
10190 print:print:print"  h - haup
tmenue"                              <pc>
10200 print:print"  k - andere kon
stellation"                          <mh>
```

```
10210 getq$:ifq$=""then10210          <cc>
10220 ifq$="h"then110                 <ck>
10230 ifq$="k"then8770                <ae>
10240 ifq$<>"h"orq$<>"k"then10170     <gl>
10250 graphic0                        <le>
10260 scnclr                          <ga>
10270 input"wollen sie noch ein st
ernbild (j/n)";p$                     <ce>
10280 ifp$="j"thengoto2620            <fb>
10290 ifp$="n"thengoto110             <cl>
10300 ifp$<>"n"andp$<>"j"then10260    <fo>
10310 scnclr                          <me>
10320 color0,2:color4,1:color5,1:c
olor1,1                               <fh>
10330 graphic1,1                      <gl>
10340 fast                            <fd>
10350 box1,20,60,60,140:box1,30,70
,50,130                               <bf>
10360 box1,140,60,160,140:box1,180
,60,220,140:box1,190,70,210,130       <cm>
10370 draw1,240,60to240,140to250,1
40to250,90to270,140to280,140to280,
60to270,60to270,110to250,60to240,6
0                                     <bc>
10380 draw1,80,60to80,140to90,140t
o90,100to110,140to120,140to100,100
to120,100to120,60to80,60:box1,90,7
0,110,90                              <ae>
10390 draw1,20,60to30,50to70,50to7
0,130to60,140:draw1,60,60to70,50:d
raw1,40,70to40,120to50,120:draw1,4
0,120to30,130                         <kc>
10400 draw1,180,60to190,50to230,50
to230,130to220,140:draw1,220,60to2
30,50:draw1,200,70to200,120to210,1
20:draw1,190,130to200,120             <mm>
10410 draw1,140,60to150,50to170,50
to170,130to160,140:draw1,160,60to1
70,50                                 <ke>
10420 draw1,270,60to280,50to290,50
to290,130to280,140:draw1,280,60to2
90,50                                 <ek>
10430 draw1,240,60to250,50to260,50
to270,80:draw1,250,140to260,130to2
60,113                                <ah>
10440 draw1,250,60to260,50            <mi>
10450 draw1,80,60to90,50to130,50to
130,90to120,100:draw1,120,60to130,
50                                    <cd>
10460 draw1,100,70to100,80to110,80
:draw1,90,90to100,80                  <bl>
10470 draw1,90,140to100,130to100,1
20:draw1,120,140to128,132to110,100    <pm>
10480 draw1,140,180to180,140to220,
140to180,180to140,180:draw1,160,17
0to180,150to200,150to180,170to160,
170                                   <ik>
10490 draw1,100,180to140,140to160,
140to120,180to100,180                 <eo>
10500 draw1,0,170to20,150to40,150t
o20,170to0,170                        <pp>
10510 draw1,0,160to20,140to60,140t
o20,180to0,180                        <al>
10520 draw1,240,140to200,180to210,
180to255,155to230,180to240,180to28
0,140                                 <cg>
10530 draw1,250,140to225,165to270,
140                                   <pa>
10540 draw1,80,140to40,180to80,180
to100,160to80,160to120,140            <em>
10550 draw1,110,140to70,160to90,14
0:draw1,55,175to75,175to85,165to65
,165to55,175                          <eg>
10560 paint1,20,172:paint1,64,160:
paint1,130,160:paint1,160,175:pain
t1,220,170                            <ji>
10570 box1,140,20,160,40              <bi>
10580 draw1,140,20to150,10to170,10
to170,30to160,40:draw1,160,20to170
,10                                   <fi>
10590 draw1,90,190to110,190to100,2
00to80,200to90,190                    <cg>
10600 paint1,90,197                   <jf>
10610 slow                            <ja>
10620 getq$:ifq$=""then10620          <kk>
10630 graphic0                        <kk>
10640 return                          <ab>
10650 scnclr                          <gp>
10660 print:print:print" wollen si
e das programm"                       <mo>
10670 print:print" (b)eenden oder"    <hl>
10680 print:print" (w)eiter benutz
en"                                   <aj>
10690 geta$:ifa$=""then10690          <gf>
10700 ifa$="b"then10730               <jf>
10710 ifa$="w"then110                 <ml>
10720 ifa$<>"b"ora$<>"w"then10650     <dp>
10730 scnclr                          <ao>
10740 end                             <lk>
60000 rem nachspann ============== <pj>
60010 rem *farbcodes/steuercodes * <kg>
60020 c4$=chr$(017):c3$=chr$(029)     <kh>
60030 c2$=chr$(145):c1$=chr$(147)     <no>
60040 rem ***zeichensatz/graphik * <lm>
60050 zu$=chr$(191)                   <in>
60060 return                          <jl>
```



Patience

Stellen Sie sich vor, Sie haben zwei Kartenspiele und wollen Patience spielen. Alleine das Auflegen der Karten nimmt sehr viel Zeit in Anspruch. Diese Arbeit nimmt Ihnen jetzt der C128 ab.
Das Spiel besteht aus 104 Karten, die bunt gemischt in zehn Reihen aufgebaut werden. Das Ziel des Spieles ist, die Karten von den höchsten bis zur niedrigsten zusammenzulegen. Insgesamt gibt es 13 Reihen, daher haben Sie drei Reihen zu Ihrer Verfügung frei. Führen Sie nun den Pfeil mit dem an Port zwei angesteckten Joystick an die obere Kante der Karte, die Sie bewegen wollen. Drücken Sie den Feuerknopf. Die ausgewählte Karte färbt sich weiß und kann nun an die neue Position gebracht werden. Abgelegt wird die Karte durch erneutes Drücken des Feuerknopfes. Liegen in einem Stapel mehrere Karten in der richtigen Reihenfolge, so können diese zusammen bewegt werden. Dazu führen Sie den Pfeil auf die Karte, ab der Sie die Bewegung durchführen wollen, klicken sie an und legen sie an der neuen Position ab. Haben Sie eine Reihe fertig, eine Reihe besteht aus 13 Karten vom As bis zur Zwei, können Sie eine Karte an das Ende der Reihe zur späteren Verwendung ablegen. Dabei muß diese Karte nicht mit der Kartenfarbe der Reihe übereinstimmen. Am Ende des Spiels müssen acht fertige Reihen in jeweils einer Farbe vorhanden sein. Wollen Sie ein Spiel abbrechen oder nach gelöster Patience neubeginnen, so drücken Sie eine beliebige Taste. Der Bildschirm wird gelöscht und ein neuer Kartenstapel erscheint.
Geben Sie das Programm bitte mit unserem im Heft abgedruckten Checksummer ein und speichern dieses ab. Wundern Sie sich nicht, wenn nach dem Aufbau des Titelbilds und anschließend gedrückter Taste der Bildschirm gelöscht wird. Ihr Computer ist nicht abgestürzt, vielmehr wird in den Fast-Modus umgeschaltet, um das Spielfeld zu kreieren. □

```
10 rem ========================128  <ob>
20 rem (p) commodore welt       ==  <hf>
30 rem ========================== <mm>
40 rem (c) by                   ==  <pp>
50 rem     helmut karner        ==  <mo>
60 rem                          ==  <nd>
70 rem version 7.0   40z./ascii==  <ah>
80 rem pc-128 floppy/datasette ==  <ho>
90 rem ========================== <km>
95 gosub 60000                     <jp>
100 gosub830:color0,8:color4,7:col
or5,7                              <ll>
110 fast:graphic1,1:circle,18,18,6
,,,,,120:paint,18,18:poke208,0     <dd>
120 sshapeb$,10,10,30,30:sprsavb$,
1:sprite1,1,2                      <jl>
130 graphic1,1:box,7,7,24,32:dimk$
(104),p(12,25),k(104),x(12),g(104)
:k=0                               <hc>
140 f$(1)="S":f$(2)="Z":f$(3)="A":
f$(4)="X"                          <mh>
150 forf=1to4:forw=2to14:w$=right$
(str$(w),1):k=k+1                  <kd>
160 g(k)=w:g(k+52)=w               <ng>
170 ifw>9thenw$="t"                <de>
180 ifw>10thenw$="j"               <dd>
190 ifw>11thenw$="q"               <lj>
200 ifw>12thenw$="k"               <hb>
210 ifw>13thenw$="a"               <fc>
220 char1,1,1,f$(f):char1,2,1,w$   <mk>
230 char1,1,3,f$(f):char1,2,3,w$   <in>
240 sshapek$(k),7,7,24,32:k$(k+52)
=k$(k)                             <fn>
250 nextw,f:graphic1,1:x=0:y=1:slo
w:color1,3                         <of>
255 cw$=zn$+zn$+zn$+zn$+zn$+zn$+zn
$+zn$+zn$+zn$+zn$+zn$+zn$+zn$+zn$+
zn$+zn$+zn$+zn$+zn$+zn$+zn$        <nj>
260 char1,1,0,zn$+zn$+zn$+zn$+zn$+
zn$+zn$+zn$+zn$+zn$+zn$+zn$+zn$+zn
$+zn$+zn$+zn$+cw$,1                <dg>
270 char1,1,1," ********* p a t i
e n c e ********** ",1             <dj>
275 wc$=zo$+zo$+zo$+zo$+zo$+zo$+zo
$+zo$+zo$+zo$+zo$+zo$+zo$+zo$+zo$+
zo$+zo$+zo$+zo$                    <hc>
280 char1,1,2,zo$+zo$+zo$+zo$+zo$+
zo$+zo$+zo$+zo$+zo$+zo$+zo$+zo$+zo
$+zo$+zo$+zo$+zo$+wc$,1            <bb>
290 char1,1,24,"           * k a r
 n e r *           ",1             <lh>
300 char1,1,23,"           * h e l
 m u t *           ",1             <pn>
310 char1,2,23,"tries":char1,32,23
," time ":char1,2,24,"     ":char1
,32,24,"      "                    <pl>
320 fort=1to104                    <fe>
330 x=x+1:ifx>10thenx=1:y=y+1      <ke>
340 p(x,y)=int(rnd(1)*104)+1       <mg>
350 ifk(p(x,y))=1goto340           <ll>
360 k(p(x,y))=1:x(x)=y:l=p(x,y):go
sub690                             <mp>
370 gshapek$(p(x,y)),x*24+8,y*8+16
:next                              <nc>
380 ti$="000000":u$="0000"         <bd>
390 poke2564,192:movspr1,x*24+32,y
*8+66:movspr2,x*24+32,y*8+66:poke2
564,193                            <oh>
400 color1,3:char1,7-len(u$),24,ri
ght$(u$,(len(u$)-1)):char1,32,24,t
i$                                 <db>
410 v=joy(2):ifv=1theny=y-1:ify<1t
heny=1                             <pm>
420 ifv=3thenx=x+1:ifx>12thenx=12  <hi>
430 ifv=5theny=y+1:ify>22theny=22  <la>
440 ifv=7thenx=x-1:ifx<0thenx=0    <oa>
450 ifpeek(208)=1thenrun110        <fb>
460 ifv<128goto390                 <pn>
470 ifz=1goto500                   <kg>
480 x1=x:y1=y:z=1:sprsavk$(p(x1,y1
)),2                               <np>
490 movspr2,x*24+32,y*8+66:sprite2
,1,2:goto390                       <io>
```

```
500 ifx1=xthenz=0:sprite2,0:goto39
0                                   <op>
510 ifp(x,x(x))=0goto530            <af>
520 ifg(p(x,x(x)))<>g(p(x1,y1))+1g
oto730                              <kn>
530 ify1<>x(x1)goto800              <hd>
540 ifx(x1)>y1goto620               <go>
550 fort=y1+2toy1+5:char1,x1*3+1,t
,"   ":next                         <nd>
560 l=p(x1,y1-1):gosub690           <kp>
570 gshapek$(p(x1,y1-1)),x1*24+8,(
y1+1)*8                             <dj>
580 l=p(x1,y1):gosub690:y=x(x)+1    <jd>
590 gshapek$(p(x1,y1)),x*24+8,y*8+
16:u=u+1:u$=str$(u)                 <ga>
600 p(x,y)=p(x1,y1):p(x1,y1)=0:spr
ite2,0                              <pk>
610 x(x1)=y1-1:x(x)=y:z=0:goto390   <an>
620 q=x(x1):fort=y1+2toq+5:char1,x
1*3+1,t,"   ":next                  <ie>
630 l=p(x1,y1-1):gosub690           <bb>
640 gshapek$(p(x1,y1-1)),x1*24+8,(
y1+1)*8:x(x1)=y1-1:y=x(x)+1         <ic>
650 fort=y1toq:l=p(x1,t):gosub690   <nb>
660 gshapek$(p(x1,t)),x*24+8,y*8+1
6                                   <po>
670 p(x,y)=p(x1,t):p(x1,t)=0:sprit
e2,0:y=y+1:u=u+1:u$=str$(u)         <oa>
680 next:x(x)=y-1:z=0:goto390       <pj>
690 color1,3:ifl>26thencolor1,1     <dm>
700 ifl>52thencolor1,3              <pj>
710 ifl>78thencolor1,1              <cn>
720 return                          <dd>
730 ify1<>x(x1)thenx1=x:goto500     <ej>
740 ifg(p(x,1))<14thenx1=x:goto500  <hm>
750 ifg(p(x,x(x)))<>2thenx1=x:goto
500                                 <cc>
760 ifx(x)<>13thenx1=x:goto500      <me>
770 o=0:ford=2to13:ifg(p(x,d))<>g(
p(x,d-1))-1theno=1                  <bo>
780 next:ifo=1thenx1=x:goto500      <hj>
790 goto540                         <el>
800 o=0:ford=y1+1tox(x1):ifg(p(x1,
d))<>g(p(x1,d-1))-1theno=1          <gd>
810 next:ifo=1thenx1=x:goto500      <mh>
820 goto540                         <ci>
830 graphic0,1:color4,3:color0,8:c
olor5,7                             <ag>
840 char1,5,4,"p r o u d  t o  p r
 e s e n t :"+c4$+c4$+c4$:print     <fg>
850 printleft$(qr$,4)re$zf$zv$"I U
"zv$"I "zf$zh$zd$"  "zh$"  "zf$zv$
"  UI"zh$"  U"zv$"  "zf$zv$         <ap>
860 printleft$(qr$,4)pu$yo$" "yo$"
 "yo$" "yo$"  "yo$"  "yo$"  "yo$"
   "yo$yo$yo$"  "yo$"  "yo$         <kb>
870 printleft$(qr$,4)wh$za$zv$"K "
za$zv$zi$"  "yo$"  "yo$"  "za$zv$
"  "yo$yo$yo$"  "yo$"  "za$zv$      <if>
880 printleft$(qr$,4)cy$yo$"  "yo
$" "yo$"  "yo$"  "yo$"  "yo$"  "
yo$yo$yo$"  "yo$"  "yo$             <oo>
890 printleft$(qr$,4)oe$zg$"  "zg
$" "zg$"  "zg$"  "zg$"  "zc$zv$"
 "zg$"JK  J"zv$"  "zc$zv$           <io>
900 printgr$c4$c4$c4$left$(qr$,12)
"c. 1 9 8 7  b y"                   <kf>
910 printlb$c4$c4$c4$left$(qr$,7)"
k a r n e r  h e l m u t":sleep5    <lm>
920 graphic0,1:color4,8:color0,3:c
olor5,8                             <mg>
930 char1,3,12,"kennen sie die spi
elregeln,(j/n) ?":getkey a$         <on>
940 ifa$="j"thenreturn              <cc>
950 ifa$<>"n"goto930                <hj>
960 fast:graphic0,1:color4,5:color
0,8:color5,3                        <mg>
970 fort=1to40:m$=m$+chr$(160):nex
t                                   <lb>
980 fort=1to3:char1,0,t,m$,1:next   <mp>
990 fort=21to23:char1,0,t,m$,1:nex
t                                   <jf>
1000 char1,9,2,"s p i e l r e g e
l n :",1                            <ea>
1010 char1,8,22,"druecken sie eine
 taste",1                           <pp>
1020 window0,4,39,20:slow           <al>
1030 printc1$b1$c4$c4$"patience is
t ein altes kartenlege-spiel."      <cn>
1040 printc4$"es besteht aus 104 k
arten ."                            <hh>
1050 printc4$"das ziel des spieles
 ist ,"                             <ha>
1060 printc4$"die karten von der h
oechsten bis zur"                   <en>
1070 printc4$"niedrigsten zusammen
zulegen."                           <pg>
1080 printc4$"die reihenfolge der
karten finden sie"                  <mb>
1090 printc4$"auf der naechsten se
ite :"                              <cg>
1100 getkey a$                      <jc>
1110 printc1$re$c4$c4$left$(qr$,4)
"<-- hoechste ---- niedrigste -->"  <hc>
1120 printc4$c3$c3$c3$"U"zv$zv$"I
U"zv$zv$"I U"zv$zv$"I U"zv$zv$"I U
"zv$zv$"I U"zv$zv$"I U"zv$zv$"I"    <ne>
1130 printc3$c3$c3$yo$"Sa"yo$" "yo
$"Sk"yo$" "yo$"Sq"yo$" "yo$"Sj"yo$
" "yo$"St"yo$" "yo$"S9"yo$" "yo$"S
8"yo$                               <id>
1140 printc3$c3$c3$yo$"  "yo$" "yo
$"  "yo$" "yo$"  "yo$" "yo$"  "yo$
" "yo$"  "yo$" "yo$"  "yo$" "yo$"
 "yo$                               <je>
1150 printc3$c3$c3$yo$"Sa"yo$" "yo
```

```
$"Sk"yo$" "yo$"Sq"yo$" "yo$"Sj"yo$
" "yo$"St"yo$" "yo$"S9"yo$" "yo$"S
8"yo$                                   <nc>
1160 printc3$c3$c3$"J"zv$zv$"K J"z
v$zv$"K J"zv$zv$"K J"zv$zv$"K J"zv
$zv$"K J"zv$zv$"K J"zv$zv$"K"          <pk>
1170 printc4$left$(qr$,5)"U"zv$zv$
"I U"zv$zv$"I U"zv$zv$"I"s2$"U"zv$
zv$"I"s2$"U"zv$zv$"I"s2$"U"zv$zv$"
I"                                      <nl>
1180 printleft$(qr$,5)yo$"S7"yo$"
"yo$"S6"yo$" "yo$"S5"yo$" "yo$"S4"
yo$" "yo$"S3"yo$" "yo$"S2"yo$          <bj>
1190 printleft$(qr$,5)yo$s2$s2$yo$
s2$yo$s2$s2$yo$s2$yo$s2$s2$yo$s2$y
o$s2$s2$yo$s2$yo$s2$s2$yo$s2$yo$s2
$s2$yo$                                 <cd>
1200 printleft$(qr$,5)yo$"S7"yo$"
"yo$"S6"yo$" "yo$"S5"yo$;s2$yo$"S4
"yo$" "yo$"S3"yo$" "yo$"S2"yo$         <lo>
1210 printleft$(qr$,5)"J"zv$zv$"K"
s2$"J"zv$zv$"K"s2$"J"zv$zv$"K"s2$"
J"zv$zv$"K"s2$"J"zv$zv$"K"s2$"J"zv
$zv$"K"                                 <cd>
1220 getkeya$                           <bl>
1230 printcl$bk$c4$"die karten wer
den mit dem joystick in"                <ii>
1240 printc4$"port 2 wie folgt bew
egt :"                                  <om>
1250 printc4$"fuehren sie den pfei
l ueber die karte ,"                    <bd>
1260 printc4$"und klicken sie mit
dem feuerknopf an ."                    <nc>
1270 printc4$"die karte faerbt sic
h weiss und kann nun"                   <bp>
1280 printc4$"an die neue position
 gebracht werden ."                     <in>
1290 printc4$"dort wird sie mit de
m feuerknopf wieder"                    <lm>
1300 printc4$"ausgeklickt."             <lb>
1310 getkeya$                           <mj>
1320 printcl$bl$c4$"sie koennen au
ch mehrere karten zugleich"             <lm>
1330 printc4$"bewegen , wenn sie i
n der richtigen rei="                   <ma>
1340 printc4$"henfolge liegen ."        <fd>
1350 printc4$"dazu muessen sie nur
 den pfeil auf die"                     <nd>
1360 printc4$"karte fuehren , ab d
er sie die bewegung"                    <pj>
1370 printc4$"durchfuehren wollen
."                                      <do>
1380 printc4$"die bewegung selbst
ist die gleiche wie"                    <pg>
1390 printc4$"bei einer karte ."        <mk>
1400 getkeya$                           <gh>
1410 printcl$re$c4$"beim ausklicke
n genuegt es , wenn sie in"             <ak>
1420 printc4$"der richtigen reihe
sind !"                                 <hj>
1430 printc4$"die karten werden in
 zehn reihen unter="                    <eb>
1440 printc4$"einander gelegt ."        <dj>
1450 printc4$"drei reihen bleiben
zu ihrer verwendung"                    <gp>
1460 printc4$"frei ."                   <pa>
1470 printc4$"eine fertige reihe b
esteht aus 13 karten"                   <lk>
1480 printc4$"vom as bis zur 2 ."       <ca>
1490 getkeya$                           <am>
1500 printcl$bk$c4$"eine wichtige
regel ist das 'aufheben' :"             <cd>
1510 printc4$"wenn sie eine fertig
e reihe haben ,"                        <kc>
1520 printc4$"koennen sie am ende
der reihe eine karte"                   <mh>
1530 printc4$"fuer spaetere verwen
dung aufheben ."                        <fa>
1540 printc4$"die kartenfarbe (her
z, karo, pik, treff)"                   <le>
1550 printc4$"muss in der reihe ni
cht uebereinstimmen!"                   <nn>
1560 printc4$"am ende des spieles
muessen 8 fertige"                      <ei>
1570 printc4$"reihen vorhanden sei
n ."                                    <ek>
1580 getkeya$                           <lc>
1590 printcl$wh$c4$c4$c4$c3$"ich w
uensche ihnen fuer die naechsten"       <hc>
1600 printc4$left$(qr$,5)"stunden
viel kopfzerbrechen !"                  <hd>
1610 printlb$c4$c4$c4$c3$c3$"nach
geloester patience druecken sie"        <ng>
1620 printc4$left$(qr$,15)"eine ta
ste"                                    <ob>
1630 getkeya$:return                    <fj>
60000 rem nachspann ============== <pj>
60010 rem *farbcodes/steuercodes * <kg>
60020 wh$=chr$(005):c4$=chr$(017)       <ee>
60030 re$=chr$(028):c3$=chr$(029)       <ie>
60040 gr$=chr$(030):bl$=chr$(031)       <do>
60050 oe$=chr$(129):bk$=chr$(144)       <gh>
60060 cl$=chr$(147):lb$=chr$(154)       <bc>
60070 pu$=chr$(156):cy$=chr$(159)       <fg>
60080 rem ***zeichensatz/graphik *      <pg>
60090 s2$=chr$(160):za$=chr$(171)       <ap>
60100 zc$=chr$(173):zd$=chr$(174)       <hh>
60110 zf$=chr$(176):zg$=chr$(177)       <fh>
60120 zh$=chr$(178):zi$=chr$(179)       <eg>
60130 zn$=chr$(184):zo$=chr$(185)       <ho>
60140 zv$=chr$(192):yo$=chr$(221)       <fj>
60150 rem *********zeichenfolgen *      <gp>
60160 for q=1 to 40                     <pp>
60170 qr$=qr$+c3$                       <fg>
60180 next q                            <fe>
60190 return                            <ka>
```

Gelbe Kugeln

Gelbe Kugeln ist ein Programm für den Commodore 64, an dem sich bis zu acht Spieler beteiligen können.
Nach RUN werden Sie aufgefordert, für jeden Mitspieler einzugeben, ob er am Spiel mit Joystick oder Tastatur teilnimmt. Geben Sie eine der Ziffern Eins bis Drei ein, wie auf dem Bild angegeben. Nach dem achten Spieler, oder bei weniger Spielern nach Eingabe einer Null, beginnt das Spiel.
Zuerst wird das Spielfeld aufgebaut. Für jeden Spieler erscheint ein Feld mit 72 gelben oder roten Kugeln und Kreisen. Dazu wird angegeben, ob der betreffende Spieler den Joystick oder die Tastatur benutzt, sowie die Anzahl gelber Kugeln. Auf dem darüber- oder darunterliegenden Bildschirmrand werden zur Nummer des Spielers die erreichten Punkte angezeigt. Dazu wird in der oberen Zeile der Bonus angegeben, der am Anfang bei 900 liegt, in jeder Runde aber um 60 vermindert wird.
Sind alle Felder aufgebaut, kann das Spiel beginnen. Im Feld des ersten Spielers erscheint ein blauumrandetes Quadrat. Dieses kann nun mittels Joystick oder Cursortasten bewegt werden. Durch Druck auf Feuer oder Leertaste werden die neun innerhalb des Quadrates liegenden Zeichen verändert. Aus gelben Kugeln werden gelbe Kreise, aus diesen dann rote Kugeln. Rote Kugeln verwandeln sich in rote Kreise, und diese werden zu gelben Kugeln.
Das Ziel des Spieles ist es, alle Zeichen in gelbe Kugeln umzuwandeln. Dazu kommen die Spieler der Reihe nach zum Zuge. In jeder Runde kann ein Spieler fünf Mal je neun Zeichen verändern. Nach der fünften Eingabe werden dem Spieler die Anzahl der vorhandenen gelben Kugeln als Punkte gutgeschrieben. Erreicht ein Spieler das Ziel in einer Runde mit weniger als fünf Zügen, können alle noch nicht in dieser Runde beteiligten Spieler nur noch ebensoviele Züge machen. Wenn am Ende einer Runde ein oder mehrere Spieler 72 gelbe Kugeln vorweisen können, so erhalten diese den restlichen Bonus gutgeschrieben, und für jeden Spieler wird ein neues Spielfeld aufgebaut. So werden pro Mitspieler nacheinander sechs Bilder gegeben, dann ist das Spiel zu Ende.
Eine Schikane sei noch erwähnt. Wenn während eines Spieles der Bildrand rot wird, dann werden nicht nur im Feld dieses einen Spielers, sondern in allen Feldern die Zeichen an derselben Stelle verändert. Wenn nur ein Spieler alleine spielt, wird per Zufall ein Quadrat gewählt, dessen Zeichen umgewandelt werden.
Dies ist oft zum Vorteil, gegen Ende des Spieles aber meist ein Nachteil. Es kann sogar vorkommen, daß ein Spieler alle 72 gelben Kugeln erreicht hat und sich auf den Sieg freut, aber ein nachfolgender Spieler macht ihm einen Strich durch die Rechnung, indem er wieder neun gelbe Kreise erzeugt. In diesem Falle geht das Spiel weiter, denn der Bonus wird nur gewährt, wenn am Ende der Runde noch alle 72 gelben Kugeln vorhanden sind.
Sollte der Bonus auf Null laufen, wird ein neues Bild erzeugt, so, als ob das Ziel erreicht wurde, jedoch wird natürlich kein Bonus gegeben.
Gelbe Kugeln ist mit einer Highscore-Wertung ausgestattet. Am Anfang wird, falls vorhanden, die alte Tabelle von der Diskette gelesen, und am Ende, nach Eingabe von vier Zeichen des Spielers, wieder weggespeichert. Sollte sich keine beschreibungsfähige Diskette im Laufwerk befinden, werden Sie dazu aufgefordert, dies nachzuholen. Wollen Sie keine Highscore, so geben Sie einfach „N" ein.
Nachdem das Programm Gelbe Kugeln komplett abgetippt ist, SAVEn Sie es erst mal auf Floppy oder Datasette. Sie können noch nicht damit arbeiten, denn es fehlt noch das Maschinenprogramm, das für die Gestaltung des oberen und unteren Randes und für einige andere, nur im Interrupt möglichen Effekte, zuständig ist.
Tippen Sie dazu nun den LADER GELBE KUGELN ab und dieser erzeugt nun nach RUN das erforderliche Maschinenprogramm und SAVEd es je nach Ihren Angaben auf Diskette oder Datasette.
Nun können Sie das Hauptprogramm laden und starten. Achten Sie darauf, daß sich das Maschinenprogramm beim Starten des Programmes im Zugriff befindet, denn es wird vom Hauptprogramm nachgeladen. Sollten Sie die Datasette benutzen, so ist die Zeile 120 zu ändern in:

```
120 LOAD"MP GELBE KUGELN",1,1
```

G. Kramer □

```
10 rem ========================64  <bf>
20 rem (p) commodore welt       ==  <hf>
30 rem ==========================  <mm>
40 rem (c) by                   ==  <pp>
50 rem     guenter kramer       ==  <hi>
60 rem                          ==  <nd>
70 rem version 2.0   40z./ascii==  <le>
80 rem  c-64  floppy/datasette ==  <gl>
90 rem ==========================  <km>
95 gosub 60000                     <jp>
100 s=0                            <ab>
110 a=peek(61)+256*peek(62)        <pl>
120 poke65,aand255                 <gn>
130 poke66,a/256                   <gn>
140 printchr$(147)"laden assembler
"                                  <ai>
150 fori=0to6                      <dk>
160 reada                          <nc>
170 s=s+a                          <pj>
180 ad(i)=a                        <ef>
190 next                           <pp>
200 reada                          <hb>
210 ifs=athen240                   <mn>
220 print"fehler in pruefsummen"   <mp>
230 end                            <en>
240 an=49152                       <gb>
250 en=50708                       <cm>
260 s=0                            <ai>
270 forj=0to6                      <gm>
280 a=an+256*j                     <ai>
290 e=a+255                        <ac>
300 ife>enthene=en                 <cd>
310 fori=atoe                      <kb>
320 reada                          <fd>
330 d2=peek(63)+256*peek(64)       <oa>
340 ifd1=0thend1=d2                <bh>
350 if(a<0)or(a>255)then420        <ki>
360 ifa-int(a)<>0then420           <cp>
370 s=s+a                          <ae>
380 pokei,a:next                   <bl>
390 ifs<>ad(j)then450              <fm>
400 print"teil"j"in ordnung"       <nb>
410 goto470                        <hf>
420 print"datafehler in zeile"s   <hi>
430 print"falscher wert ="a       <jm>
440 end                            <pd>
450 print"datafehler zeilen"d1"-"d
2                                  <bo>
460 end                            <bl>
470 d1=0                           <pf>
480 s=0                            <on>
490 next                           <fj>
500 goto540                        <am>
510 ****************************** <pf>
520 *** save maschinenprogramm *** <ek>
530 ****************************** <na>
540 print                          <nf>
550 print"save maschinenprogramm"  <od>
560 print                          <pn>
570 print"adresse ? 1 = kassette"  <bn>
580 printtab(8)"8/9 = diskette"    <ie>
590 poke198,0                      <ci>
600 geta$:ifa$=""then600           <jh>
610 ifa$="1"then630                <ea>
620 if(a$<"8")or(a$>"9")then600    <gc>
630 poke2,val(a$)                  <nf>
640 fori=0to3                      <cn>
650 poke251+i,peek(43+i)           <pn>
660 next                           <kp>
670 poke43,0                       <ic>
680 poke44,192                     <kp>
690 poke45,158                     <ka>
700 poke46,198                     <lh>
710 save"mp gelbe kugeln",peek(2)  <im>
720 poke43,peek(251)               <pn>
730 poke44,peek(252)               <ki>
740 poke45,peek(253)               <pc>
750 poke46,peek(254)               <lm>
760 ifpeek(2)=1thenend             <oe>
770 print                          <kd>
780 open1,8,15                     <gc>
790 get#1,a$                       <in>
800 get#1,b$                       <aj>
810 if(a$="0")and(b$="0")then910   <oe>
820 printa$;b$;                    <jc>
830 get#1,a$                       <gp>
840 printa$;                       <ab>
850 ifst<>64then830                <al>
860 close1                         <on>
870 print"floppy ok?"s2$"(j/e)"    <km>
880 print"e = ende, nicht speicher
n"                                 <ap>
890 geta$:ifa$="j"then670          <be>
900 ifa$<>"e"then890               <dg>
910 close1:end                     <nh>
920 ************************       <eh>
930 *** datas pruefsummen ***      <cj>
940 ************************       <hg>
950 data32926,36175,31393,36129    <fk>
960 data34879,35514,2443,209459    <bp>
970 ******************************
*                                  <fc>
980 *** datas maschinenprogramm **
*                                  <ca>
990 ******************************
*                                  <do>
1000 data76,9,192,76,149,192,76,6,
194,32,160,229,120,169,51,133,1,16
9,208                              <pj>
1010 data160,0,133,96,132,95,132,9
0,132,88,169,224,133,91,169,240,13
3,89,32                            <pa>
1020 data191,163,169,55,133,1,169,
196,141,0,221,169,204,141,136,2,32
,68,229                            <ld>
```

```
1030 data169,56,141,24,208,169,3,1
41,20,3,169,192,141,21,3,141,9,3,1
69,6,141                                <ck>
1040 data8,3,169,248,141,18,208,14
1,219,192,173,17,208,41,127,141,17
,208,169                                <po>
1050 data129,141,26,208,169,76,141
,164,192,160,46,185,0,208,153,0,20
3,153                                   <ca>
1060 data48,203,153,96,203,136,16,
241,88,169,24,141,3,193,141,9,193,
141,87                                  <je>
1070 data196,160,15,169,250,153,0,
203,136,240,3,136,208,247,96,173,2
5,208                                   <np>
1080 data141,25,208,48,7,173,13,22
0,88,76,49,234,76,188,254,44,230,1
92,160                                  <ch>
1090 data7,185,0,203,153,0,208,185
,8,203,153,8,208,185,39,203,153,39
,208,136                                <lj>
1100 data16,235,173,16,203,141,16,
208,160,2,185,21,203,153,21,208,18
5,27,203                                <of>
1110 data153,27,208,136,16,241,162
,248,142,248,207,232,238,219,192,2
08,247                                  <ml>
1120 data45,157,193,169,19,141,17,
208,162,0,160,0,173,18,208,201,250
,208,249                                <lg>
1130 data169,6,56,233,1,208,251,14
0,32,208,142,33,208,24,144,3,76,14
4,193                                   <be>
1140 data24,144,3,76,82,193,162,24
0,142,217,192,162,48,142,173,192,1
62,69                                   <bm>
1150 data142,202,192,162,75,142,20
8,192,162,56,142,179,192,162,64,14
2,194                                   <fl>
1160 data192,162,87,142,185,192,16
9,56,141,9,193,173,17,208,9,11,41,
123,141                                 <oj>
1170 data231,192,162,0,142,18,208,
162,9,142,243,192,162,0,160,0,76,2
27,193                                  <ad>
1180 data162,232,142,217,192,162,9
6,142,173,192,162,117,142,202,192,
162,123                                 <ka>
1190 data142,208,192,162,104,142,1
79,192,162,112,142,194,192,162,135
,142,185                                <ef>
1200 data192,169,56,141,3,193,162,
40,142,18,208,162,50,142,243,192,1
62,76                                   <dd>
1210 data142,167,192,162,0,160,0,7
6,227,193,162,44,142,167,192,162,7
6,142                                   <gn>
1220 data227,192,76,170,192,162,0,
142,173,192,162,248,142,217,192,16
2,21,142                                <jb>
1230 data202,192,162,27,142,208,19
2,162,8,142,179,192,162,16,142,194
,192,162                                <ff>
1240 data39,142,185,192,162,45,142
,227,192,160,240,140,18,208,162,25
0,142                                   <cc>
1250 data243,192,173,17,208,41,119
,141,231,192,162,0,160,0,169,24,14
1,3,193                                 <ka>
1260 data141,9,193,142,238,192,140
,236,192,162,248,142,219,192,76,18
8,254                                   <ae>
1270 data162,0,32,121,0,240,13,32,
253,174,32,138,173,32,184,177,164,
100,166                                 <hl>
1280 data101,96,32,115,0,201,64,24
0,3,76,231,167,32,115,0,201,82,240
,3,76                                   <nl>
1290 data167,194,32,115,0,201,69,2
08,8,169,44,141,164,192,76,228,167
,133,251                                <ef>
1300 data32,115,0,32,241,193,134,2
52,32,241,193,134,253,165,251,201,
67,208                                  <gh>
1310 data54,169,1,166,252,36,253,2
40,3,142,76,193,10,36,253,240,3,14
2,216                                   <ai>
1320 data193,10,36,253,240,3,142,1
38,193,10,36,253,240,3,142,78,193,
10,36                                   <po>
1330 data253,240,3,142,218,193,10,
36,253,240,3,142,140,193,76,174,16
7,201                                   <bk>
1340 data70,208,8,166,252,142,255,
255,76,174,167,76,8,175,169,1,36,2
,240,4                                  <dd>
1350 data162,0,240,18,169,2,36,2,2
40,4,162,48,208,8,169,4,36,2,240,9
,162,96                                 <jm>
1360 data73,255,37,2,133,2,138,96,
201,83,240,3,76,232,197,32,115,0,1
62,7,201                                <ba>
1370 data176,208,8,162,2,134,2,169
,82,208,29,201,79,208,2,162,2,201,
77,208                                  <dl>
1380 data2,162,4,201,82,208,2,162,
3,201,85,208,2,162,1,134,2,32,115,
0,141                                   <pm>
1390 data250,194,32,115,0,32,241,1
93,134,251,192,1,240,2,160,0,132,2
52,32                                   <lb>
1400 data241,193,134,253,32,241,19
3,134,254,169,69,201,65,208,6,169,
73,162                                  <ll>
1410 data45,208,8,201,69,208,57,16
9,36,162,13,141,52,195,142,54,195,
```

```
32,131                                          <kf>
1420 data194,201,4,240,37,24,105,2
1,141,55,195,141,58,195,165,251,20
8,4,169                                         <fj>
1430 data255,208,9,168,169,1,136,2
40,3,10,208,250,36,255,13,117,203,
141,117                                         <db>
1440 data203,76,19,195,76,174,167,
201,67,208,42,32,131,194,201,4,240
,32,24                                          <cd>
1450 data105,39,141,104,195,141,95
,195,165,251,166,253,208,10,160,7,
153,87                                          <ll>
1460 data203,136,16,250,208,224,20
2,157,87,203,76,70,195,76,174,167,
201,70                                          <dn>
1470 data208,91,32,131,194,201,4,2
40,81,201,0,208,2,169,254,201,48,2
08,2,169                                        <if>
1480 data252,201,96,208,2,169,250,
141,188,195,165,251,166,253,208,12
,142,187                                        <ba>
1490 data195,162,2,238,188,195,160
,254,208,24,202,138,41,7,162,6,10,
202,208                                         <cf>
1500 data252,144,3,238,188,195,141
,187,195,160,62,162,1,165,251,153,
0,249                                           <jp>
1510 data136,192,255,208,248,136,2
06,188,195,202,208,241,76,116,195,
76,174                                          <le>
1520 data167,201,75,240,3,76,211,1
96,32,131,194,201,4,208,3,76,148,1
96,141                                          <mh>
1530 data83,196,141,94,196,24,105,
29,141,181,196,41,242,141,154,196,
141,157                                         <jj>
1540 data196,141,161,196,141,164,1
96,162,240,164,254,240,7,136,192,7
,240,2                                          <mc>
1550 data162,208,142,99,196,201,16
,208,2,162,250,201,112,208,14,166,
253,224                                         <ef>
1560 data59,176,2,162,59,224,196,1
44,2,162,196,165,251,72,165,252,72
,152,10                                         <ja>
1570 data72,165,252,208,13,169,1,3
2,167,196,32,151,196,169,144,76,68
,196,32                                         <fp>
1580 data167,196,32,160,196,169,36
,141,109,196,169,1,133,252,32,175,
196,104                                         <nj>
1590 data168,165,251,153,96,203,24
,105,24,133,251,8,200,138,153,96,2
03,200                                          <eh>
1600 data192,16,208,37,6,252,32,17
5,196,165,252,40,36,21,32,160,196,
169,36                                          <ci>
1610 data141,109,196,165,251,201,1
92,144,4,169,192,133,251,76,80,196
,32,151                                         <pk>
1620 data196,76,80,196,40,104,133,
252,104,133,251,76,214,195,76,174,
167,73                                          <ek>
1630 data255,45,112,203,141,112,20
3,96,13,112,203,141,112,203,96,136
,48,4                                           <lg>
1640 data10,76,167,196,96,169,48,1
41,87,196,173,125,203,37,252,208,5
,169,24                                         <mi>
1650 data141,87,196,173,94,196,201
,48,208,11,162,29,173,71,203,37,25
2,240                                           <gb>
1660 data2,162,9,96,201,66,240,3,7
6,145,197,160,3,165,251,10,133,251
,165,252                                        <cn>
1670 data42,133,252,136,208,243,9,
224,141,252,196,165,251,141,251,19
6,160                                           <oa>
1680 data7,120,169,53,133,1,185,16
8,226,162,55,134,1,88,153,240,207,
136,16                                          <de>
1690 data237,166,253,240,45,32,60,
197,166,254,32,76,197,32,131,194,2
01,4,240                                        <fo>
1700 data30,32,123,197,141,47,197,
165,253,141,46,197,160,7,162,21,18
5,240                                           <nn>
1710 data207,157,192,250,202,202,2
02,136,16,244,76,20,197,76,174,167
,240,1                                          <an>
1720 data202,224,14,176,251,134,25
1,138,10,101,251,133,251,96,240,5,
202,224                                         <kb>
1730 data24,176,251,138,160,0,132,
254,201,3,144,6,200,56,233,3,208,2
46,133                                          <km>
1740 data252,152,162,6,10,133,253,
165,254,42,133,254,165,253,202,208
,243,101                                        <di>
1750 data251,101,252,133,253,96,20
1,0,208,2,169,254,201,48,208,2,169
,252,201                                        <ao>
1760 data96,208,2,169,250,24,101,2
54,96,201,88,240,16,201,80,208,4,1
69,27                                           <cf>
1770 data208,10,201,89,208,68,169,
23,208,2,169,29,141,179,197,32,131
,194,201                                        <bd>
1780 data4,240,49,24,105,29,141,21
2,197,141,218,197,141,221,197,166,
253,208                                         <ci>
1790 data4,169,255,208,8,169,1,202
,240,3,10,208,250,166,251,208,8,73
,255,45                                         <ki>
1800 data29,203,76,220,197,13,29,2
```

```
03,141,29,203,76,170,197,76,174,16
7,76,8                                  <dm>
1810 data175,201,128,240,3,76,8,17
5,120,169,4,141,136,2,169,49,141,2
0,3,169                                 <gn>
1820 data234,141,21,3,169,167,141,
9,3,169,228,141,8,3,32,24,229,88,1
69,199                                  <bc>
1830 data141,0,221,76,228,167           <ak>
60000 rem nachspann ==============
=                                       <da>
60010 rem *** zeichensatz/graphik
*                                       <ig>
60020 s2$=chr$(160)                     <bb>
60030 return                            <fo>
```

Gelbe Kugeln — 1 —

```
10 rem ========================64  <bf>
20 rem (p) commodore welt       ==  <hf>
30 rem ==========================  <mm>
40 rem (c) by                   ==  <pp>
50 rem     guenter kramer       ==  <hi>
60 rem                          ==  <nd>
70 rem version 2.0   40z./ascii==  <le>
80 rem  c-64  floppy/datasette ==  <gl>
90 rem ==========================  <km>
95 gosub 60000                      <jp>
100 ifa=1then130                    <nd>
110 a=1                             <ak>
120 load"mp gelbe kugeln",8,1       <hf>
130 gosub4740                       <do>
140 gosub2920                       <hb>
150 gosub3130                       <ka>
160 gosub4180                       <pd>
170 gosub1160                       <be>
180 gosub290                        <bo>
190 gosub5270                       <lb>
200 printchr$(147)                  <nd>
210 print"ein neues spiel?"s2$"(j/
n)"                                 <eh>
220 geta$                           <ap>
230 ifa$="n"thenend                 <ac>
240 ifa$<>"j"then220                <pn>
250 run130                          <fd>
260 ********************            <be>
270 *** spiel laeuft ***            <in>
280 ********************            <ff>
290 sn=1:bi=0                       <pl>
300 a=rnd(-ti)                      <lg>
310 gosub1710                       <la>
320 gz=5:ru=1                       <cb>
330 goto370                         <gc>
340 sn=sn+1                         <jk>
350 ifsn>anthensn=1:ru=ru+1:bo=bo-
60:gz=5                             <ng>
360 ifbo=0then1030                  <ob>
370 pokeze,0:pokesp,0:syscu         <cn>
380 a$=str$(bi)                     <dn>
390 printchr$(154)a$". bild";       <fo>
400 a$=str$(ru)                     <bh>
410 printtab(15)a$". runde";        <da>
420 a$=right$(chr$(32)+str$(bo),4)  <oe>
430 printtab(28)"bonus ="a$         <ko>
440 zf=int((10+2*an)*rnd(1))+2      <ah>
450 a=ko(sn,0)                      <hj>
460 a0=ko(sn,1)                     <md>
470 ifms(sn,2)=72then920            <kb>
480 gosub2690                       <bb>
490 zu=1                            <hd>
500 poke198,0                       <jg>
510 gosub2480                       <ml>
520 ifei=32then630                  <md>
530 ifei=29thena=a+1                <oa>
540 ifei=157thena=a-1               <ka>
550 ifei=145thena0=a0-1             <kj>
560 ifei=17thena0=a0+1              <ac>
570 ifa=0thena=1                    <nj>
580 ifa=7thena=6                    <fj>
590 ifa0=0thena0=1                  <nl>
600 ifa0=8thena0=7                  <bi>
610 gosub2690                       <ga>
620 goto510                         <ja>
630 $sma                            <kk>
640 ifru=1then840                   <mo>
650 ifzu<>zfthen840                 <pe>
660 ifan>1then760                   <ma>
670 gosub2190                       <of>
680 $rc,2,31                        <ea>
690 a4=a:a5=a0                      <kn>
700 a=int(rnd(0)*6)+1               <cj>
710 a0=int(rnd(0)*7)+1              <ke>
720 gosub2190                       <bl>
730 a=a4:a0=a5                      <ka>
740 $rc,0,31                        <ol>
750 goto850                         <mk>
760 a4=sn                           <mc>
770 $rc,2,31                        <jo>
780 forsn=1toan                     <ha>
790 gosub2190                       <pe>
800 next                            <mi>
810 $rc,0,31                        <ng>
820 sn=a4                           <al>
830 goto850                         <nd>
840 gosub2190                       <cg>
850 ko(sn,0)=a                      <hc>
860 ko(sn,1)=a0                     <pc>
870 zu=zu+1                         <hj>
880 ifms(sn,2)=72then910            <mj>
890 ifzu>gzthen920                  <bm>
900 goto510                         <cl>
910 gz=zu-1                         <il>
920 ms(sn,1)=ms(sn,1)+ms(sn,2)      <nl>
930 gosub2800                       <io>
```

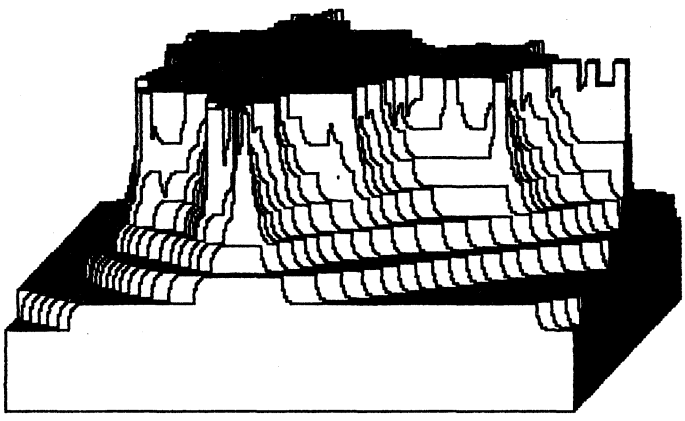
```
940 ifsn<anthen340                          <ml>
950 en$="0"                                 <ng>
960 forsn=1toan                             <je>
970 ifms(sn,2)<72then1010                   <mb>
980 ms(sn,1)=ms(sn,1)+bo                    <gh>
990 gosub2800                               <ah>
1000 en$="1"                                <kn>
1010 next                                   <gn>
1020 ifen$="0"then340                       <ec>
1030 sn=1                                   <ag>
1040 ifbi<6then310                          <in>
1050 pokeze,24:pokesp,10:syscu              <pf>
1060 printchr$(153)"spielende";             <be>
1070 print"(return)"                        <fk>
1080 geta$                                  <ak>
1090 ifa$<>chr$(13)then1080                 <af>
1100 $end                                   <eb>
1110 printchr$(147)                         <ji>
1120 return                                 <fg>
1130 **********************                 <gk>
1140 *** sprites laden ***                  <ni>
1150 **********************                 <cf>
1160 $soa:$sua                              <el>
1170 $sof,0                                 <oo>
1180 $suf,0                                 <kj>
1190 a1$=""                                 <dc>
1200 a2$=""                                 <mg>
1210 a3=0                                   <fg>
1220 a$=chr$(32)+chr$(32)                   <hh>
1230 fori=1toan                             <me>
1240 a0$=right$(str$(i),1)                  <ak>
1250 a0$=a$+a0$+a$                          <kj>
1260 ifi>4then1290                          <mo>
1270 a1$=a1$+a0$+chr$(32)                   <bm>
1280 goto1300                               <bp>
1290 a2$=a2$+a0$+chr$(32)                   <ka>
1300 next                                   <le>
1310 $rc,0,255                              <jc>
1320 a2=1:a1=5:a=1                          <ja>
1330 a$=a1$                                 <fj>
1340 gosub3890                              <an>
1350 a$=a2$                                 <fg>
1360 a2=4:a1=1                              <ma>
1370 gosub3890                              <nb>
1380 a$=".....0"                            <dg>
1390 a$=a$+a$                               <fb>
1400 a$=a$+a$                               <gp>
1410 a$=left$(a$,len(a1$))                  <ko>
1420 a2=1:a1=14                             <ge>
1430 gosub3890                              <gj>
1440 a2=4:a1=10                             <gp>
1450 a$=left$(a$,len(a2$))                  <mk>
1460 gosub3890                              <bm>
1470 $sox,0:$sux,0                          <jb>
1480 fori=1to7step2                         <ep>
1490 a=i*40                                 <mk>
1500 $sok,a,0,i                             <la>
1510 $sok,a+24,0,i+1                        <oj>
1520 $suk,a,0,i                             <he>
1530 $suk,a+24,0,i+1                        <hf>
1540 next                                   <jg>
1550 $src,12                                <oe>
1560 $sre:$sma                              <ih>
1570 printchr$(147)                         <md>
1580 a$="x"                                 <np>
1590 a2=2:a3=0                              <lb>
1600 $smf,0                                 <nc>
1610 forj=1toan                             <kf>
1620 a1=3*j-2                               <jg>
1630 gosub3450                              <fi>
1640 $smk,80*j-60,65,j                      <bm>
1650 ifj>4then:$smk,80*j-380,145,j          <io>
1660 next                                   <ih>
1670 return                                 <kh>
1680 **********************                 <bo>
1690 *** neues bild ***                     <dc>
1700 **********************                 <kf>
1710 printchr$(147)                         <on>
1720 bi=bi+1                                <op>
1730 fori=1toan                             <lb>
1740 ko(i,0)=1                              <mg>
1750 ko(i,1)=1                              <fe>
1760 ms(i,2)=18                             <eb>
1770 next                                   <gd>
1780 a$=chr$(119)+chr$(119)+chr$(1
19)                                         <pb>
1790 a$=a$+chr$(113)+chr$(113)+a$           <bo>
1800 a$=a$+chr$(32)+chr$(32)+a$             <cl>
1810 a$=a$+chr$(32)+chr$(32)+a$             <il>
1820 a0$=a$                                 <ie>
1830 ifan>3then1850                         <be>
1840 a0$=left$(a$,an*10-2)                  <lc>
1850 fori=3to11                             <ma>
1860 pokeze,i:pokesp,1:syscu                <gk>
1870 printchr$(158)a0$                      <kc>
1880 next                                   <ea>
1890 a0$=a$                                 <pd>
1900 ifan<5then1970                         <ff>
1910 ifan=8then1930                         <de>
1920 a0$=left$(a$,(an-4)*10-2)              <jn>
1930 fori=13to21                            <ml>
1940 pokeze,i:pokesp,1:syscu                <an>
1950 printchr$(158)a0$                      <on>
1960 next                                   <ob>
1970 forsn=1toan                            <nm>
1980 fork=1to10                             <cm>
1990 a=1+int(6*rnd(1))                      <ek>
2000 a0=1+int(7*rnd(1))                     <hf>
2010 gosub2190                              <pn>
2020 next                                   <fj>
2030 a=2:a0=6:gosub2190                     <dn>
2040 a=5:a0=2:gosub2190                     <ke>
2050 pokeze,2                               <cm>
2060 pokesp,10*sn-9                         <pi>
2070 ifsn<5then2100                         <ge>
2080 pokeze,22                              <ad>
```

```
2090 pokesp,10*sn-49                    <hb>
2100 syscu                              <dd>
2110 printchr$(30)chr$(18)e$            <cp>
2120 next                               <cc>
2130 sn=1                               <af>
2140 bo=900                             <df>
2150 return                             <gk>
2160 ***********************            <kk>
2170 *** setzen felder ***              <fj>
2180 ***********************            <oh>
2190 fori=atoi+2                        <mn>
2200 forj=a0toj+2                       <el>
2210 a1=40*j+i+70+10*sn                 <kn>
2220 ifsn>4thena1=a1+360                <do>
2230 a2=peek(f+a1)and15                 <pf>
2240 a3=peek(b+a1)                      <ci>
2250 ifa3=81then2310                    <mk>
2260 ifa2=7thenpokef+a1,10:goto229
0                                       <cp>
2270 ms(sn,2)=ms(sn,2)+1                <ab>
2280 pokef+a1,7                         <ll>
2290 pokeb+a1,81                        <ol>
2300 goto2330                           <an>
2310 ifa2=7thenms(sn,2)=ms(sn,2)-1      <kb>
2320 pokeb+a1,87                        <ji>
2330 a$=str$(ms(sn,2))                  <id>
2340 a$=right$(a$,2)                    <ng>
2350 pokeze,2                           <lo>
2360 pokesp,10*sn-3                     <me>
2370 ifsn<5then2400                     <hn>
2380 pokeze,22                          <dl>
2390 pokesp,10*sn-43                    <no>
2400 syscu                              <jl>
2410 printchr$(159)a$                   <hd>
2420 nextj,i                            <kk>
2430 e$=e$(ms(sn,0))                    <ie>
2440 return                             <lb>
2450 ************************          <mn>
2460 *** eingabe abfragen ***           <eo>
2470 ************************          <gf>
2480 ei=0                               <ho>
2490 geta$                              <eh>
2500 $sme,sn                            <nn>
2510 ifms(sn,0)<3then2550               <ai>
2520 ifa$=""then2640                    <lk>
2530 ei=asc(a$)                         <mk>
2540 goto2640                           <ea>
2550 poke56322,224                      <de>
2560 a2=peek(56320)                     <dm>
2570 ifms(sn,0)=1thena2=peek(56321
)                                       <ml>
2580 poke56322,255                      <ld>
2590 if(a2and1)=0thenei=145             <ei>
2600 if(a2and2)=0thenei=17              <if>
2610 if(a2and4)=0thenei=157             <al>
2620 if(a2and8)=0thenei=29              <jh>
2630 if(a2and16)=0thenei=32             <nc>
2640 ifei=0then2490                     <ei>
2650 return                             <fg>
2660 ***********************           <ia>
2670 *** bewegen sprite ***             <fm>
2680 ***********************           <pg>
2690 a1=80*sn-68+8*a                    <gf>
2700 a2=57                              <nm>
2710 ifsn<5then2740                     <fg>
2720 a1=a1-320                          <gn>
2730 a2=137                             <ha>
2740 a2=a2+8*a0                         <nm>
2750 $smk,a1,a2,sn                      <ff>
2760 return                             <dd>
2770 *************************         <gg>
2780 *** zaehler anzeigen ***           <di>
2790 *************************         <cc>
2800 a$=str$(ms(sn,1))                  <mo>
2810 a$=right$(a$,len(a$)-1)            <hj>
2820 a$=right$("......"+a$,6)           <cf>
2830 a2=1:a1=14:a3=0                    <ad>
2840 ifsn>4thena2=4                     <df>
2850 a=sn*6-5                           <hc>
2860 ifsn>4thena=a-24:a1=10             <fo>
2870 gosub3890                          <an>
2880 return                             <ce>
2890 ********************              <ch>
2900 *** anfangswerte ***               <fc>
2910 ********************              <fi>
2920 sys12*4096                         <ka>
2930 $rf,0:$rc,4,255                    <fn>
2940 $rc,3,31:$re:$saa:$saf,0           <on>
2950 printchr$(150)chr$(147)            <ob>
2960 printchr$(158)                     <hn>
2970 b=52224:      rem bildspeiche
r                                       <cn>
2980 f=55296:      rem farbspeiche
r                                       <fo>
2990 cu=58640:     rem cursorsteue
rung                                    <hd>
3000 ze=214:       rem cursorzeile      <ba>
3010 sp=211:       rem cursorspalt
e                                       <gb>
3020 si=54272:     rem sid              <da>
3030 dime$(3):     rem eingabe          <hn>
3040 dimms(8,2):   rem spielerwert
e                                       <cd>
3050 dimko(8,1):   rem koordinaten      <oi>
3060 e$(1)="joy-1"                      <kf>
3070 e$(2)="joy-2"                      <fi>
3080 e$(3)="taste"                      <ol>
3090 return                             <mk>
3100 ****************                  <fj>
3110 *** titelbild ***                  <fp>
3120 ****************                  <ok>
3130 a$="gelbe kugeln"                  <jm>
3140 a1=1:a2=7:a3=128                   <fh>
3150 gosub3450                          <bi>
3160 a=2:a1=8:a2=7                      <mb>
3170 gosub3890                          <kl>
```

```
3180 a=int(len(a$)/2):a1=176-8*a   <cc>
3190 $src,3:$smc,3:$srx,1:$sry,1   <hl>
3200 $srk,168-8*len(a$)            <lh>
3210 $smk,a1,67:$soe:$sme          <ep>
3220 pokeze,3:pokesp,20-a:syscu    <dh>
3230 printa$                       <ga>
3240 fori=1to200:next              <hd>
3250 pokeze,7:pokesp,12:syscu      <na>
3260 print"ein programm von"        <gb>
3270 fori=1to200:next              <pl>
3280 fori=67to200                  <pl>
3290 $smk,a1,i                     <gc>
3300 next                          <gd>
3310 $sma:$sue                     <di>
3320 $smf,0:a$="guenter kramer"    <ak>
3330 a1=1:a2=2:a3=0:$sme           <ig>
3340 $smx,1:$smy,1                 <hh>
3350 $smk,56,92                    <ce>
3360 gosub3450                     <hl>
3370 a=2:a2=2:a1=14                <ob>
3380 gosub3890                     <an>
3390 $rc,2,56                      <on>
3400 $rc,14,31                     <og>
3410 return                        <em>
3420 **********************        <fm>
3430 *** rahmen setzen ***         <ld>
3440 **********************        <mf>
3450 a=len(a$)                     <ck>
3460 if(a2and1)=1thengosub3530     <mm>
3470 if(a2and2)=2thengosub3650     <al>
3480 if(a2and4)=4thengosub3770     <fb>
3490 return                        <on>
3500 ********************          <ml>
3510 *** oberer rand ***           <fm>
3520 ********************          <no>
3530 fori=1+a1toa+a1               <ce>
3540 $sob,64+a3,1,i:next           <op>
3550 $sob,73+a3,1,i:$sob,66+a3,9,i <dj>
3560 $sob,75+a3,14,i               <fb>
3570 fori=a+a1to1+a1step-1         <pe>
3580 $sob,64+a3,14,i:next          <de>
3590 $sob,74+a3,14,i               <oe>
3600 $sob,66+a3,9,i:$sob,85+a3,1,i <ki>
3610 return                        <no>
3620 ****fn*****************       <cm>
3630 *** mittelsprites ***         <hm>
3640 ********************          <la>
3650 fori=1+a1toa+a1               <mc>
3660 $smb,64+a3,1,i:next           <kc>
3670 $smb,73+a3,1,i:$smb,66+a3,9,i <ej>
3680 $smb,75+a3,14,i               <co>
3690 fori=a+a1to1+a1step-1         <oa>
3700 $smb,64+a3,14,i:next          <fn>
3710 $smb,74+a3,14,i               <aa>
3720 $smb,66+a3,9,i:$smb,85+a3,1,i <gp>
3730 return                        <mp>
3740 ********************          <jl>
3750 *** unterer rand ***          <nc>
3760 ********************          <nb>
3770 fori=1+a1toa+a1               <dm>
3780 $sub,64+a3,1,i:next           <ob>
3790 $sub,73+a3,1,i:$sub,66+a3,9,i <op>
3800 $sub,75+a3,14,i               <hp>
3810 fori=a+a1to1+a1step-1         <eg>
3820 $sub,64+a3,14,i:next          <nj>
3830 $sub,74+a3,14,i               <jm>
3840 $sub,66+a3,9,i:$sub,85+a3,1,i <bj>
3850 return                        <ma>
3860 ********************          <ob>
3870 *** text setzen ***           <fj>
3880 ********************          <oh>
3890 iflen(a$)=0then3930           <pg>
3900 if(a2and1)=1thengosub3970     <bh>
3910 if(a2and2)=2thengosub4040     <fm>
3920 if(a2and4)=4thengosub4110     <mp>
3930 return                        <ga>
3940 ******************            <jh>
3950 *** text oben ***             <pd>
3960 ******************            <jm>
3970 fori=0tolen(a$)-1             <mh>
3980 a4=asc(mid$(a$,i+1,1))and63   <oi>
3990 $sob,a4+a3,a1,a+i:next        <il>
4000 return                        <on>
4010 ********************          <ip>
4020 *** text mitte ***            <eg>
4030 ********************          <bj>
4040 fori=0tolen(a$)-1             <ec>
4050 a4=asc(mid$(a$,i+1,1))and63   <mk>
4060 $smb,a4+a3,a1,a+i:next        <ik>
4070 return                        <hj>
4080 ********************          <ek>
4090 *** text unten ***            <nm>
4100 ********************          <gb>
4110 fori=0tolen(a$)-1             <pm>
4120 a4=asc(mid$(a$,i+1,1))and63   <db>
4130 $sub,a4+a3,a1,a+i:next        <ff>
4140 return                        <ag>
4150 **************                <oo>
4160 *** anfang ***                <ff>
4170 **************                <op>
4180 printchr$(150)                <jl>
4190 pokeze,12:pokesp,10:syscu     <ho>
4200 print"wieviele mitspieler?"   <ba>
4210 print                         <jo>
4220 print"1 = joystick 1"         <kg>
4230 print"2 = joystick 2"         <eb>
4240 print"3 = tastatur"           <og>
4250 print                         <oo>
4260 print"0 = spiel beginnt"      <hm>
4270 print"9 = high score"         <lh>
4280 pokeze,14:pokesp,10:syscu     <lp>
4290 an=0                          <ab>
4300 an=an+1                       <el>
4310 printtab(19)"spieler"an" = "; <ap>
4320 poke204,0                     <pi>
4330 poke198,0                     <nh>
```

```
4340 geta$                                <en>
4350 ifa$="0"then4440                     <bc>
4360 ifa$="9"then4470                     <nm>
4370 ifa$<"1"then4340                     <lb>
4380 ifa$>"3"then4340                     <hk>
4390 ms(an,0)=val(a$)                     <fi>
4400 poke204,1                            <bn>
4410 printe$(ms(an,0))                    <ii>
4420 ifan<8then4300                       <ap>
4430 goto4550                             <if>
4440 ifan=1then4340                       <ko>
4450 an=an-1                              <ia>
4460 goto4550                             <ej>
4470 $rc,0,255                            <ej>
4480 $sma                                 <dp>
4490 poke204,1                            <ni>
4500 gosub5020                            <ik>
4510 $rc,2,56                             <gi>
4520 $rc,14,31                            <cl>
4530 printchr$(150)chr$(147)              <jl>
4540 goto4620                             <en>
4550 poke204,1                            <mh>
4560 print"-"                              <oh>
4570 printtab(19)"in ordnung? ";          <me>
4580 poke204,0                            <em>
4590 geta$                                <cj>
4600 ifa$="j"then4680                     <ch>
4610 ifa$<>"n"then4590                    <bi>
4620 poke204,1                            <hh>
4630 fori=12to24                          <dn>
4640 poke781,i                            <ak>
4650 sys59903                             <fe>
4660 next                                 <ao>
4670 goto4180                             <jj>
4680 poke204,1                            <gi>
4690 printa$                              <ck>
4700 return                               <gk>
4710 ************************            <ko>
4720 *** lesen highscore ***              <bd>
4730 ************************            <kc>
4740 open15,8,15                          <cf>
4750 open3,8,3,"0:h.s. kugeln,u,r"        <fk>
4760 input#15,a,a$,a0,a1                  <fi>
4770 ifa=0thenhs=1:goto4880               <dg>
4780 close3:close15                       <lk>
4790 printchr$(147)a;a$;a0;a1             <eb>
4800 print"wenn highscore vorhande
n,"                                       <pd>
4810 print"diskette einlegen und"         <io>
4820 print"'j' eingeben."                 <ib>
4830 print"sonst 'n' eingeben."          <np>
4840 geta$                                <aa>
4850 ifa$="j"then4740                     <pa>
4860 ifa$<>"n"then4840                    <gf>
4870 hs=0                                 <la>
4880 dimhs$(20),hs(20)                    <ai>
4890 a$="    "                            <ah>
4900 a=0                                  <fc>
4910 fori=1to20                           <lb>
4920 ifhs=0then4940                       <cp>
4930 input#3,a$,a                         <oi>
4940 hs$(i)=a$                            <pi>
4950 hs(i)=a                              <ai>
4960 next                                 <gi>
4970 close3:close15                       <gk>
4980 return                               <jn>
4990 ************************            <li>
5000 *** anzeigen highscore ***           <mo>
5010 ************************            <mn>
5020 printchr$(159)chr$(147)              <mj>
5030 printspc(15)"ehrentafel"             <dc>
5040 print:print                          <in>
5050 a$=""                                <bl>
5060 fori=1to7                            <gl>
5070 a$=a$+chr$(32)                       <pa>
5080 next                                 <fj>
5090 fori=1to10                           <pf>
5100 a0$=right$(a$+str$(hs(i)),7)         <gm>
5110 a1$=right$(a$+str$(hs(i+10)),
7)                                        <ae>
5120 ifhs(i)=0thena0$=a$                  <aa>
5130 ifhs(i+10)=0thena1$=a$               <jo>
5140 printspc(6)hs$(i)spc(1)a0$;          <gd>
5150 printspc(1)chr$(122)chr$(122)
;                                         <jj>
5160 printspc(1)hs$(i+10)spc(1)a1$        <jn>
5170 next                                 <ao>
5180 pokeze,22:pokesp,0:syscu             <eg>
5190 print"weiter = return"               <gi>
5200 poke198,0                            <cj>
5210 geta$                                <jk>
5220 ifa$<>chr$(13)then5210               <dj>
5230 return                               <jd>
5240 ************************            <pb>
5250 *** update highscore ***             <ke>
5260 ************************            <gp>
5270 pokeze,10:pokesp,13:syscu            <kc>
5280 print"namen eingeben"                <pd>
5290 print                                <bg>
5300 fori=1toan                           <lb>
5310 ifms(i,1)<hs(20)then5530             <li>
5320 a0$=""                               <ka>
5330 poke198,0                            <mk>
5340 printspc(13)"spieler"i" = ";         <pg>
5350 forj=1to4                            <da>
5360 geta$                                <en>
5370 ifa$=chr$(13)thenj=4:goto5420        <gk>
5380 ifa$<chr$(35)then5360                <ha>
5390 ifa$>chr$(90)then5360                <la>
5400 printa$;                             <cd>
5410 a0$=a0$+a$                           <kh>
5420 next                                 <ae>
5430 forj=19to1step-1                     <jn>
5440 hs$(j+1)=hs$(j)                      <ao>
5450 hs(j+1)=hs(j)                        <gf>
5460 a=j+1                                <he>
```

```
5470 ifms(i,1)>hs(j)thena=j          <bd>
5480 hs(a)=ms(i,1)                   <aj>
5490 hs$(a)=left$(a0$+"     ",4)     <dp>
5500 ifa<>jthenj=0                   <oi>
5510 next                            <lj>
5520 print                           <oe>
5530 next                            <ob>
5540 gosub5020                       <po>
5550 printchr$(147)                  <fd>
5560 ifhs=0then5600                  <dm>
5570 open1,8,15                      <ig>
5580 print#1,"s:h.s. kugeln"         <in>
5590 close1                          <bp>
5600 open15,8,15                     <ae>
5610 open3,8,3,"0:h.s. kugeln,u,w"   <gp>
5620 poke198,0                       <hl>
5630 input#15,a,a$,a0,a1             <pj>
5640 ifa=0then5740                   <gc>
5650 close3:close15                  <ci>
5660 printchr$(147)a;a$;a0;a1        <jf>
5670 print"wenn highscore gewuensc
ht,"                                 <bc>
5680 print"diskette einlegen und"    <mf>
5690 print"'j' eingeben."            <en>
5700 print"sonst 'n' eingeben."      <nc>
5710 geta$:ifa$="n"then5790          <lm>
5720 ifa$="j"then5570                <kf>
5730 goto5710                        <ki>
5740 fori=1to20                      <ja>
5750 ifhs(i)=0thenhs$(i)="     "     <fp>
5760 print#3,hs$(i);chr$(13);hs(i)   <jb>
5770 next                            <md>
5780 close3:close15                  <mj>
5790 return                          <ph>
60000 rem nachspann ==============
=                                    <da>
60010 rem *** zeichensatz/graphik
*                                    <ig>
60020 s2$=chr$(160)                  <bb>
60030 return                         <fo>
```



Text 128

Das Programm Text 128 ist nur für die 80-Zeichen-Darstellung des C128 geeignet.
Im Hauptmenü können die Punkte Text eingeben, Text löschen, Disc Funktionen, Text drucken und Programmende mit Hilfe der Spacetaste angewählt und mit der Returntaste aufgerufen werden.

Text eingeben:
In der Kopfzeile steht immer die aktuelle Zeilen- und Spaltennummer der momentanen Cursorposition. Außerdem steht in der Kopfzeile die Belegung der Funktionstasten. Mit der Taste F1 springt der Cursor zehn Zeichen nach rechts. Mit F2 kann der Tabulator eingestellt werden. Es muß nur der voreingestellte Tabulator mit dem Wert zehn überschrieben und mit der Return-Taste bestätigt werden. Je nach Tabulator-Einstellung ertönt kurz vor dem Zeilenende ein akustisches Signal. Wird die Return-Taste betätigt, springt der Cursor an den nächsten Zeilenanfang und führt den eingestellten Tabulatorsprung aus.
Mit F3 wird der Cursor veranlaßt, auf den Textanfang zu springen und F4 veranlaßt genau das Gegenteil, der Cursor springt zum Textende. Mit F5 wird der Text rechts vom Cursor nach rechts verschoben. Ist das Zeilenende erreicht, kann nicht weiter verschoben werden. Mit F6 wird das Zeichen auf dem sich der Cursor befindet gelöscht. Der Text rechts daneben wird um ein Zeichen nach links verschoben. Befindet sich der Cursor auf einer Zeile ohne Text, so wird die ganze Zeile aus dem Speicher gelöscht und der Text unterhalb des Cursors um eine Zeile nach oben verschoben. Zurück ins Hauptmenü gelangt man mit F7. Um eine Zeile einzufügen, muß F8 betätigt werden. Die Zeile, auf der sich der Cursor befindet, wird um eine Zeile nach unten verschoben. Das Einfügen und Löschen von Textzeilen kann, je nach Textlänge, zwischen einer und fünf Sekunden dauern. Es steht ein Textspeicher von 420 Zeilen mit je 79 Zeichen zur Verfügung.

Text löschen:
Es wird der gesamte Text, nach einer Sicherheitsabfrage, gelöscht.

Disc Funktionen:
In diesem Untermenü können Texte geladen, gespeichert und gelöscht werden. Wird nur die Return-Taste betätigt, ohne einen Textnamen einzugeben, erfolgt der Rücksprung ins Diskmenü. Wird ein Text abgespeichert, so wird dem Dateinamen 'text.' vorangestellt. Beim Laden und Löschen von Files, kann der Joker '*' verwendet werden. Wird beim Menüpunkt Text löschen der Joker eingegeben, so werden alle Files deren Namen mit 'text.' beginnt gelöscht.
Die Ausgabe des Directory kann mit der „no scroll"-Taste angehalten werden.

Text drucken:
Nach Anwahl dieses Menüpunktes muß die Anzahl der Ausdrucke eingegeben werden. Es werden auf eine DIN-A4-Seite genau 60 Zeilen gedruckt. Das heißt, in der ersten Seite werden die Zeilen eins bis 60 ausgedruckt, die zweite Seite enthält dann die Zeilen 61 bis 120. □

```
10 rem ======================128  <ob>
20 rem (p) commodore welt      ==  <hf>
30 rem ========================== <mm>
40 rem (c) by                  ==  <pp>
50 rem     andreas hink        ==  <mj>
60 rem                         ==  <nd>
70 rem version 7.0   80z./ascii==  <gi>
80 rem pc-128 floppy/drucker   ==  <on>
90 rem ========================== <km>

95 gosub 60000                     <jp>
100 rem -------------------------
- initialisieren -----------------
---------                          <em>
110 fast                           <nk>
120 max=420                        <op>
130 printchr$(27)+"m"              <ko>
140 dimte$(max+10),aw$(13)         <ae>
150 ze=1:sp=1:ta=9:ae=0:ges=1:colo
r6,1:color5,14:vol1                <je>
160 printchr$(14);chr$(11);he$he$c
l$;                                <eo>
170 data" Text eingeben "," Text l
oeschen "," Disc Funktionen "      <bl>
180 data" Text drucken "," Program
m Ende "," zurueck mit <F7> "      <ae>
190 data" neues Blatt einlegen und
 <return> druecken, "              <ec>
200 data" Inhaltsverzeichnis "," T
ext laden "," Text speichern "," T
ext loeschen "," Disc Status "," *
 zurueck * "                       <mg>
210 fori=1to13:readaw$(i):next     <hl>
220 fori=1to28:printze$;:next:prin
trn$"    Textverarbeitung    "rf$;
:fori=1to28:printze$;:next         <fo>
230 fori=1to4:printzk$tab(79)z1$;:
next                               <bk>
240 fori=1to80:printzm$;:next      <on>
250 gosub2240                      <bk>
260 key1,chr$(133)                 <pd>
270 key2,chr$(137)                 <hm>
280 key3,chr$(134)                 <bc>
290 key4,chr$(138)                 <mh>
300 key5,chr$(135)                 <do>
310 key6,chr$(139)                 <fk>
320 key7,chr$(136)                 <ai>
330 key8,chr$(140)                 <ob>
340 g1$="
                   "               <ip>
350 fori=1tomax:te$(i)=g1$:nexti   <jn>
360 rem -------------------------
- hauptmenue ---------------------
----------                         <mp>
370 trap1670:dclose:close4,4,7     <oe>
380 window1,1,78,4,1:a=1           <kj>
390 printhe$tab(13)"waehlen Sie mi
t <space>,   dann <return> zum bes
taetigen"c4$                       <kc>
400 printtab(4)rn$aw$(1)rf$tab(28)
aw$(2)tab(49)aw$(3)                <bh>
410 printtab(4)aw$(4)tab(28)aw$(5) <bi>
420 getkeyg$:sound1,50000,1        <mn>
430 ifg$=chr$(32)then460           <nh>
440 ifg$=chr$(13)thenonagoto540,11
60,1230,1790,2190                  <kl>
450 goto420                        <of>
460 a=a+1:ifa=6thena=1             <pn>
470 ifa=1thenchar1,4,2,aw$(1),1:ch
ar1,28,3,aw$(5),0                  <dm>
480 ifa=2thenchar1,28,2,aw$(2),1:c
har1,4,2,aw$(1),0                  <lj>
490 ifa=3thenchar1,49,2,aw$(3),1:c
har1,28,2,aw$(2),0                 <cf>
500 ifa=4thenchar1,4,3,aw$(4),1:ch
ar1,49,2,aw$(3),0                  <lc>
510 ifa=5thenchar1,28,3,aw$(5),1:c
har1,4,3,aw$(4),0                  <gi>
520 goto420                        <fa>
530 rem -------------------------
- text eingeben -----------------
----------                         <lf>
540 printcl$" Zeile:      Spalte:
     "rn$" **  "aw$(1)" ** ":prin
t                                  <om>
550 print"Tabulator -----<F1>  Tex
t Anfang -<F3>  einfuegen --<F5>
zurueck --------<F7>";             <ja>
560 print"Tab einstellen <F2>  Tex
t Schluss <F4>  loeschen ---<F6>
Zeile einfuegen <F8>";             <eg>
570 window8,1,24,1:printze;c1$"  "
tab(13)sp;c1$" "                   <pc>
580 window0,6,79,24                <dm>
590 gosub770                       <oe>
600 getkeya$:sound1,50000,1:ifges<
zethenges=ze                       <ff>
610 s=asc(a$)                      <cp>
620 ifs<141ands>132thengoto910     <op>
630 ifs=34thena$=chr$(39)          <je>
640 ifs=13 thengosub760:gosub780:g
osub760:goto570                    <ma>
650 ifs=17 thengosub760:gosub790:g
osub760:goto570                    <dn>
660 ifs=145thengosub760:gosub860:g
osub760:goto570                    <ad>
670 ifs=157thengosub760:gosub890:g
osub770:goto570                    <ca>
680 ifs=29 thengosub760:gosub830:g
osub770:goto570                    <ce>
690 ifs<13then600                  <mc>
700 ifs=14ors=15ors=30ors=31then60
0                                  <fb>
710 ifs<29ands>17then600           <em>
720 ifs<160ands>127then600         <gm>
730 mid$(te$(ze),sp,1)=a$          <nd>
```

```
740 gosub760:gosub830                                   <mi>
750 goto570                                             <mc>
760 char1,sp-1,ae,mid$(te$(ze),sp,
1):return                                               <gl>
770 char1,sp-1,ae,mid$(te$(ze),sp,
1),1:return                                             <cd>
780 sp=1+ta                                             <hj>
790 ze=ze+1:ifze=max+1thenchar1,27
,ae,chr$(15)+" 7 Din-A4 Seiten vol
l ",1:ze=max-1:ae=ae-2                                  <lc>
800 ifae<18thenae=ae+1                                  <mc>
810 ifae=18thenprintchr$(27)+"l":p
rintchr$(27)+"m";te$(ze)                                <hj>
820 return                                              <pm>
830 sp=sp+1:ifsp=80thengosub780                         <pf>
840 ifsp>79-tathenvol5:sound2,8000
,5:vol1                                                 <jj>
850 return                                              <di>
860 ifae<19andae>0thenae=ae-1:ze=z
e-1:return                                              <oa>
870 ifae=0andze>1thenze=ze-1:print
chr$(27)+"w":window0,6,79,6:printt
e$(ze-ae)                                               <kl>
880 ifae=0thenreturn                                    <ok>
890 ifsp>1thensp=sp-1:return                            <nc>
900 ifsp=1thensp=79:gosub860:retur
n                                                       <lo>
910 ifs=136then380                                      <ea>
920 ifs=133thengosub760:sp=sp+ta+1
:ifsp>79thensp=79                                       <ga>
930 ifs=133thengosub770:goto570                         <hf>
940 ifs=137thenbegin:window1,1,78,
4,1:print                                               <gp>
950 print"Wie gross soll der Tabul
ator sein "rn$"(01-40)"rf$"  ";ta+
1                                                       <fp>
960 window43,2,46,2:inputta:ifta<1
orta>40thenta=10                                        <gg>
970 ta=ta-1:window1,1,78,4:bend:go
to540                                                   <gm>
980 ifs=134thenprinthe$;:fori=1to1
9:printte$(i):next:ze=1:sp=1:ae=0:
goto570                                                 <lj>
990 ifs=138thenbegin:ze=1:ifges>19
thenze=ges-18                                           <ml>
1000 printhe$;:fori=1to19:printte$
(i+ze-1):next:sp=1:ae=18:ze=ze+18                       <jb>
1010 bend:goto570                                       <pa>
1020 ifs=135thenbegin                                   <fc>
1030 ifright$(te$(ze),1)=" "thente
$(ze)=left$(te$(ze),sp-1)+" "+mid$
(te$(ze),sp,79-sp):char1,0,ae,te$(
ze)                                                     <hg>
1040 bend:goto570                                       <pp>
1050 ifs=139thenbegin                                   <il>
1060 ifte$(ze)=g1$andges>1thenfori
=zetoges:te$(i)=te$(i+1):next:te$(
ges)=g1$:ges=ges-1:printhe$;:fori=
1to19:printte$(i+ze-ae-1):next:got
o1090                                                   <ho>
1070 te$(ze)=left$(te$(ze),sp-1)+r
ight$(te$(ze),79-sp)+chr$(32)                           <hd>
1080 char1,0,ae,te$(ze)                                 <ma>
1090 bend:goto570                                       <ci>
1100 ifs=140andges<maxthenbegin:ge
s=ges+1                                                 <ic>
1110 fori=gestozestep-1:te$(i+1)=t
e$(i):next:te$(ze)=g1$                                  <na>
1120 printhe$;:fori=1to19:printte$
(i+ze-ae-1):next                                        <oj>
1130 bend:goto570                                       <df>
1140 goto570                                            <dl>
1150 rem ------------------------
- text loeschen ----------------
----------                                              <mf>
1160 printcl$:char1,27,0," **  "+a
w$(2)+" ** " ,1:print                                   <le>
1170 printtab(20)c4$"Text "chr$(15
)"loeschen"chr$(27)+"o";", sind Si
e sicher ? j/n ";                                       <nn>
1180 getkeyg$:sound1,50000,1:ifg$=
"j"thenbegin:printrn$" ja "rf$                          <fh>
1190 fori=1toges:te$(i)=g1$:next:p
rintcl$:window0,6,79,24,1:ze=1:sp=
1:ae=0:ges=1                                            <ic>
1200 bend                                               <oh>
1210 goto380                                            <jn>
1220 rem ------------------------
- disc funktionen ---------------
----------                                              <ff>
1230 window1,1,78,4,1:a=1                               <di>
1240 printhe$tab(13)"waehlen Sie m
it <space>,   dann <return> zum be
staetigen"c4$                                           <ae>
1250 char1,4,2,aw$(8),1:char1,28,2
,aw$(9):char1,49,2,aw$(10)                              <jh>
1260 char1,4,3,aw$(11):char1,28,3,
aw$(12):char1,49,3,aw$(13)                              <bl>
1270 window20,6,59,24                                   <ah>
1280 printrn$"              "aw$(3)"
        "rf$;                                           <oa>
1290 fori=1to17:printzk$tab(39)z1$
:next                                                   <gk>
1300 printzm$zm$zm$zm$zm$zm$zm$zm$
zm$zm$zm$zm$zm$zm$zm$zm$zm$zm$zm$z
m$zm$zm$zm$zm$zm$zm$zm$zm$zm$zm$zm
$zm$zm$zm$zm$zm$zm$zm$zm$zm$        <ef>
1310 window1,1,78,4:print                               <eb>
1320 getkeyg$:sound1,50000,1                            <ce>
1330 ifg$=chr$(32)then1370                              <pp>
1340 ifg$=chr$(13)anda=6thengoto16
70                                                      <pg>
1350 ifg$=chr$(13)then1450                              <ap>
1360 goto1320                                           <bn>
1370 a=a+1:ifa=7thena=1                                 <ia>
1380 ifa=1thenchar1,4,2,aw$(8),1:c
```

```
har1,49,3,aw$(13),0                                    <ff>
1390 ifa=2thenchar1,28,2,aw$(9),1:
char1,4,2,aw$(8),0                                     <no>
1400 ifa=3thenchar1,49,2,aw$(10),1
:char1,28,2,aw$(9),0                                   <hm>
1410 ifa=4thenchar1,4,3,aw$(11),1:
char1,49,2,aw$(10),0                                   <nl>
1420 ifa=5thenchar1,28,3,aw$(12),1
:char1,4,3,aw$(11),0                                   <hb>
1430 ifa=6thenchar1,49,3,aw$(13),1
:char1,28,3,aw$(12),0                                  <jp>
1440 goto1320                                          <dg>
1450 window21,7,58,23                                  <gn>
1460 ifa=1thenprinttab(9)aw$(8);ch
r$(27)+"l":print:directory:printch
r$(27)+"m";:getkeyg$:printcl$:goto
1310                                                   <ai>
1470 ifa=5thenprint:printtab(12)aw
$(12):char1,1,5," "+ds$+" ",1:getk
eyg$:printcl$:goto1310                                 <ci>
1480 ifa=2thenbegin:printc4$tab(13
)aw$(9):a1=ges                                         <ie>
1490 gosub1710:ifna$=""then1550                        <hk>
1500 dopen#2,(na$):input#2,ges:ifd
s>0thenprinthe$he$:char1,29,17,ds$
,1:getkeyg$:dclose:goto1550                            <ei>
1510 fori=1toa1:te$(i)=g1$:next                        <io>
1520 fori=1toges:input#2,te$(i):ne
xt                                                     <ef>
1530 dclose                                            <ak>
1540 ze=1:ae=0:sp=1                                    <do>
1550 bend                                              <gf>
1560 ifa=3thenbegin:printc4$tab(11
)aw$(10)                                               <lp>
1570 gosub1710:ifna$=""then1610                        <df>
1580 dopen#2,(na$),w:print#2,ges:i
fds>0thenprinthe$he$:char1,30,17,d
s$,1:getkeyg$:dclose:goto1610                          <ob>
1590 fori=1toges:print#2,chr$(34)+
te$(i):next                                            <hj>
1600 dclose                                            <ce>
1610 bend                                              <fg>
1620 ifa=4thenbegin:printc4$tab(12
)aw$(11)                                               <nd>
1630 gosub1710:ifna$=""then1650                        <cn>
1640 scratch(na$):ifval(mid$(ds$,2
1,2))>0thenprinthe$he$:char1,28,17
,ds$,1:char1,28,19,"validate .....
":open15,8,15:print#15,"v":close15                     <kl>
1650 bend                                              <pg>
1660 window21,7,58,23,1:goto1310                       <oo>
1670 window0,6,79,24                                   <go>
1680 fori=1to19:trap1670                               <ki>
1690 printte$(i+ze-ae-1):next                          <ol>
1700 goto370                                           <ga>
1710 printc4$"       Geben Sie den
Textname ein :"                                        <no>
1720 print"            "c4$ze$ze$z
e$ze$ze$ze$ze$ze$ze$ze$ze$ze$ze$ze$ze$ze
$ze$                                                   <op>
1730 print"            "rn$"
       "rf$                                            <ck>
1740 print"            "zm$zm$zm$z
m$zm$zm$zm$zm$zm$zm$zm$zm$zm$zm$zm
$                                                      <am>
1750 window34,13,46,13,1:printrn$"
           "he$;:inputna$:iflen(
na$)>11then1750                                        <mi>
1760 na$="text."+na$:iflen(na$)=5t
henna$=""                                              <ai>
1770 return                                            <gp>
1780 rem ------------------------
- text drucken ------------------
-----------                                            <gp>
1790 printcl$:char1,27,0," **  "+a
w$(4)+"  ** " ,1:an=0                                  <af>
1800 char1,20,2,"Wieviel Ausdrucke
 (max.99) ?    <="                                     <bl>
1810 window48,3,51,3,1                                 <fb>
1820 print"  0":inputg$:an=val(g$)                     <bg>
1830 ifan<1oran>99then380                              <mn>
1840 window1,2,78,4,1                                  <hk>
1850 open4,4,7                                         <ca>
1860 printhe$c4$tab(22)an"Ausdruck
";                                                     <kh>
1870 ifan>1thenprint"e";                               <ko>
1880 print" mit jeweils"int((ges+5
9)/60)"seite";                                         <ei>
1890 ifges>60thenprint"n";                             <eb>
1900 window1,4,78,4,1                                  <ch>
1910 foru=1toan                                        <gn>
1920 printtab(8)aw$(7);aw$(6)                          <im>
1930 getkeyg$:ifg$=chr$(136)thencl
ose4,4,7:goto380                                       <jd>
1940 ifg$<>chr$(13)then1930                            <gf>
1950 printcl$                                          <om>
1960 a=0:fori=1toges                                   <fm>
1970 a=a+1                                             <gc>
1980 ifa=61thenbegin:fore=1to12:pr
int#4:next                                             <mf>
1990 a=1:printtab(8)aw$(7);aw$(6)                      <bm>
2000 getkeyg$:ifg$=chr$(136)thencl
ose4,4,7:goto380                                       <ej>
2010 ifg$<>chr$(13)then2000                            <ke>
2020 printcl$                                          <ej>
2030 bend                                              <om>
2040 ifte$(i)=g1$thenprint#4:goto2
140                                                    <gf>
2050 ifright$(te$(i),65)=right$(g1
$,65)thenprint#4,left$(te$(i),15):
goto2140                                               <ld>
2060 ifright$(te$(i),60)=right$(g1
$,60)thenprint#4,left$(te$(i),20):
goto2140                                               <dc>
2070 ifright$(te$(i),50)=right$(g1
$,50)thenprint#4,left$(te$(i),30):
```

```
goto2140                                    <ii>
2080 ifright$(te$(i),40)=right$(g1
$,40)thenprint#4,left$(te$(i),40):
goto2140                                    <hi>
2090 ifright$(te$(i),30)=right$(g1
$,30)thenprint#4,left$(te$(i),50):
goto2140                                    <nj>
2100 ifright$(te$(i),20)=right$(g1
$,20)thenprint#4,left$(te$(i),60):
goto2140                                    <ce>
2110 ifright$(te$(i),15)=right$(g1
$,15)thenprint#4,left$(te$(i),65):
goto2140                                    <bh>
2120 ifright$(te$(i),10)=right$(g1
$,10)thenprint#4,left$(te$(i),70):
goto2140                                    <fk>
2130 print#4,te$(i)                         <ln>
2140 nexti                                  <nn>
2150 ifa<=60thenfore=1to72-a:print
#4:next                                     <kh>
2160 print"   "u;c1$". Ausdruck is
t fertig":getkeyg$:printcl$                 <hk>
2170 nextu:close4,4,7:goto380               <ih>
2180 rem ------------------------
- programm ende ------------------
-----------                                 <ga>
2190 printcl$:char1,27,0," **  "+a
w$(5)+" ** " ,1:print:print                 <jk>
2200 printtab(21)"Programm "chr$(1
5)"Ende"chr$(27)+"o";", sind Sie s
icher ? j/n"                                <cf>
2210 getkeyg$:ifg$="j"thenprinthe$
+he$+cl$chr$(27)+"l":end                    <al>
2220 goto380                                <aa>
2230 rem ------------------------
-- vorspann ---------------------
-----------                                 <de>
2240 window1,1,78,4,1                       <km>
2250 printrn$"

                        ";                  <od>
2260 print"   (c)                 ge
schrieben von Andreas Hink    * 19
88 *                  ";                    <lf>
2270 print"                       Pa
noramastr.6,  7480 Sigmaringen-Jun
gnau                  ";                    <jo>
2280 print"

                     "rf$;                  <ap>
2290 sleep3:return                          <gc>
60000 rem nachspann ============            <pj>
60010 rem *farbcodes/steuercodes *          <kg>
60020 c4$=chr$(017):rn$=chr$(018)           <pj>
60030 he$=chr$(019):rf$=chr$(146)           <nh>
60040 cl$=chr$(147):c1$=chr$(157)           <jl>
60050 rem ***zeichensatz/graphik *          <gd>
60060 ze$=chr$(175):zk$=chr$(181)           <ok>
60070 zl$=chr$(182):zm$=chr$(183)           <fm>
60080 return                                <md>
```

Jetzt perfekt: Unser Checksummer

Hatte bisher unser Checksummer an Buchstabenvertauschungen nichts auszusetzen, so zeigt er sich nun nicht mehr so kulant.

Ob Sie mit der alten Version nun eingegeben hatten:

10 print"ab"
oder
10 print "ba",

der Checksummer brachte in beiden Fällen die Prüfsumme < gk > . Leicht kann es vorkommen, daß beim schnellen Tippen, besonders im Zehnfingersystem, die Taste, die eigentlich erst als übernächste drankommen sollte, ein wenig zu früh erwischt wird. Dem Checksummer, der lediglich die ASCII-Werte der Buchstaben addierte, konnte dieses natürlich nicht auffallen. Was also tun? Ob etwas früher oder später addiert wird, ändert nichts am Resultat der Summe. Anders ist es, wenn man zwei Verknüpfungsarten kombiniert. So ist z.B. 2*30+40 etwas anderes als 2*40+30. Und genau dieses war dann die Lösung. Die Summe wird nun einfach durch eine Linksverschiebung vor jeder Addition verdoppelt. Dadurch, daß im Falle, wenn das Ergebnis größer als 255 ist, der dabei entstehende Übertrag als Wert 1 zusätzlich addiert wird, verflüchtigen die Werte der am Anfang der Zeile gefundenen Codes sich nicht nach 8 weiteren Zeichen. Damit bleibt nicht nur die Aussagekraft der Prüfsumme voll erhalten, sondern erfährt sogar eine erhebliche Steigerung. Und vor allen Dingen wird nur eine klitzekleine Änderung erforderlich, die dieses zu vollbringen in der Lage ist. Ein einziges Byte ist nur zu ändern. Wir tun dieses mit "poke 345,10" in der Zeile 470. Dadurch wird das hier ursprünglich ansässige CLC (Clear Carry) durch ASL (Arithmetik Shift Left) ersetzt. Die nachfolgende Addition mit ADC (Addiere mit Carry) addiert den ASCII-Code des gefundenen Zeichens und den nach links herausgeshifteten Übertrag. Da einige unserer Leser beklagten, daß das Checksummerlisting nachher noch im Programmspeicher stehen würde, haben wir diesem noch mit einem "new" abgeholfen. New bzw. neu ist nun folgendes.

10 print"ab" ergibt die Prüfsumme < jd >
10 print"ba" die Prüfsumme < jf >

Sie brauchen den Checksummer nicht neu einzutippen. Alles, was Sie tun müssen, ist, die Zeile 470 anzufügen. An der Bedienung des Checksummers hat sich nichts geändert. Die Eingabehinweise bleiben daher wie gehabt.

EINGABEHINWEISE

Am rechten Rand jedes Listings, jeweils am Ende einer Eingabezeile, finden Sie zwei Buchstaben zwischen einem Kleiner- und einem Größerzeichen eingeschlossen. Diese dürfen Sie nicht mit in Ihr Listing eintippen, sondern sie dienen Ihnen zur Überprüfung Ihrer Eingabe.

Zwischen dem Kleiner- und dem Größerzeichen am rechten Rand befinden sich zwei Buchstaben. Mit einem speziellen Programm können Sie beim Eintippen Ihre Eingabe auf ihre Richtigkeit überprüfen. Dieses Programm, der Checksummer, sorgt nämlich dafür, daß nach erfolgter Zeileneingabe am linken oberen Bildschirmeck zwei Buchstaben ausgegeben werden. Wenn diese Buchstaben nicht mit den vorher erwähnten Buchstaben in unserem Listing übereinstimmen, so können Sie davon ausgehen, daß Sie sich vertippt haben und können sich so die Zeile nochmals näher ansehen, ob Sie Ihren Eingabefehler finden. Wenn Sie dann alles richtig getippt haben, so stimmen die Buchstaben überein und Sie können sich getrost der nächsten Zeile zuwenden.

ERST SICHERN, DANN AUSPROBIEREN

Das Checksummerlisting hat noch keine Prüfsummen. Seien Sie deshalb besonders aufmerksam, daß alles paßt und **speichern Sie dieses Programm unbedingt ab, bevor Sie es starten!** Bei einem Tippfehler würde es sich wahrscheinlich auf Nimmerwiedersehen verabschieden und Sie müßten die ganze Arbeit vermutlich nochmals machen. Wenn Sie es gestartet haben, so geschieht nichts Besonderes. Der Computer meldet sich einfach kurz darauf mit „READY", und das war auch schon alles. Alles sollte nun wie immer funktionieren, mit der kleinen Ausnahme, daß nunmehr nach jeder Eingabe im Direktmodus eine Prüfsumme erscheint. Nehmen Sie zum Testen irgendeine kurze BASIC-zeile aus unserem Heft her und testen sie aus. Wenn die Summen übereinstimmen, so können Sie sich freuen, denn Fehler beim Abtippen werden Ihnen nun in Zukunft viel weniger passieren, als vorher.

EINER FÜR ALLE, EIN ECHTES UNIVERSAL-PROGRAMM

Unseren Checksummer können Sie verwenden, ob Sie einen C16/116/Plus4 oder ob Sie einen C64 oder gar einen C128 haben. Nur müssen Sie beim letzteren beachten, ob Sie auch wirklich im 40-Zeichenmodus sind. Nachdem Sie den Checksummer geladen und gestartet haben, können Sie Ihr BASIC-Programm eingeben wie gewohnt, Sie können es abspeichern, Sie können auch laden, Sie können Kürzel verwenden und ob Sie ein paar Leerzeichen mehr oder weniger verwenden, der Checksummer läßt sich dadurch nicht aus der Fassung bringen. Ein bißchen Vorsicht sollte man allerdings walten lassen, wenn man Programme eingetippt hat, in denen Peeks und Pokes vorkommen. Es wird zwar nicht besonders häufig vorkommen, aber es könnte bisweilen ge-

!A! 65XX-Assembler — CMP

'CoMPare with akku' (Vergleiche Speicherinhalt mit Akku)

HEX-Code	Assembler	Takte/Adressierung
C9 OP	CMP #$OP	2 direkt (OP = 0 bis FF)
C5 ZP	CMP $ZP	3 Zeropage
D5 ZP	CMP $ZP,X	4 Zeropage
CD LO HI	CMP $HILO	4 absolut
DD LO HI	CMP $HILO,X	4 absolut X
D9 LO HI	CMP $HILO,Y	4 absolut Y
C1 ZP	CMP ($ZP,X)	6 indiziert indirekt
D1 ZP	CMP ($ZP),Y	5 indirekt indiziert

Es werden Negativ- und Zero-Flagge sowie Carrybit beeinflußt.

C64 + 128 — Kanäle schließen

Ordnungsgemäßes Schließen aller Files.

Man kann zwar mit SYS65511 alle Kanäle schließen, aber dabei werden z.B. auf Diskette geöffnete Files nicht geschlossen. Es erscheint dann ein * im Directory; das File kann nicht mehr gelesen werden.
In 152 steht die Anzahl der geöffneten Kanäle, in 601–610 stehen die Logischen Filenummern.

!A! 65XX-Assembler — CPX, CPY

'ComPare with X-register' (Vergleiche Speicher mit X-Register)

HEX-Code	Assembler	Takte/Adressierung
DO OP	CPX #$OP	2 direkt (OP=0 bis FF)
E4 ZP	CPX $ZP	3 Zeropage
EC LO HI	CPX $HILO	4 absolut
CO OP	CPY #$OP	2 direkt (OP=0 bis FF)
C4 ZP	CPY $ZP	3 Zeropage
CC LO HI	CPY $HILO	4 absolut

Es werden Negativ- und Zero-Flagge sowie Carrybit beeinflußt.

C64 + 128 — Formatierhilfe

Formatierhilfe

Neue Disketten müssen vor Gebrauch formatiert werden. Dies sollte stets bei noch kaltem Laufwerk gemacht werden, da sich sonst der Schreib-/Lese-Kopf verstellen kann. Mit diesem kleinen Programm kann man einen Satz von 10 Disketten hintereinander formatieren, wobei alle den gleichen Namen und verschiedene IDs bekommen.

```
10 input"name";n$:forn=0to9
20 open1,8,15,"n:"+n$+",a"+mid$(str$(n),2):close1
30 printchr$(147)"neue diskette - taste"
40 get a$:if a$=" " then40
50 next
```

!A! 65XX-Assembler — BEQ, BNE

'Branch if EQual' (Verzweige, wenn Ergebnis der letzten Operation gleich Null war)

HEX-Code	Assembler	Takte/Adressierung
F0 OP	BEQ SHILO	2 relativ (OP <255)

Es werden keine Register-Flaggen beeinflußt.

Wird meist zusammen mit den Vergleichsbefehlen (CMP, CPX, CPY oder LDA...) angewandt. Ist dem JMP-Befehl vorzuziehen, da Programm verschiebbar bleibt. Adresse $HILO darf nicht mehr als 127 größer oder kleiner als die Adresse sein, an der BNE steht.

C64 + 128 — Floppy-Befehlskanal #15

Validate und Formatieren ohne warten.

Das Formatieren einer Diskette mit:

```
open15,8,15,"n:name,id":close15
```

dauert bei der 1541 ca. 80 s, ein Validate (V:) kann bei voller Diskette mehrere Minuten in Anspruch nehmen.
Man kann sich die Wartezeit ersparen, wenn man den CLOSE-Befehl wegläßt und erst nach Beendigung des Vorgangs den Kanal schließt. Die Arbeit verrichtet nämlich die Floppy ganz allein, nur, wenn man den Kanal schließen will, muß man warten.

C64 + 128 — File-Test

Test ob File auf Diskette vorhanden.

Werden im Programm Diskettenfiles angesprochen, so muß vorher getestet werden, ob das File vorhanden ist, da das Programm sonst abstürzen kann, wenn es nicht gefunden wird.

Beispiel:

```
10 input"filename,typ
   (p,s,u)";p$,t$:gosub1000:ifft>0then10
20 end
1000 open8,8,8,p$+","+t$+",r":close8
1010 open1,8,15:input#1,ft,f$,c$,cS:close1:print ft,
     f$:return
```

C64 + 128 — GET mit Auswahl

GET mit logischer Stringverknüpfung

Bei einer Eingabe soll vom Programm oft nur ein bestimmter Satz von Zeichen akzeptiert werden. Dies kann mit einer Reihe von IF-Befehlen erfolgen. Dies verbraucht aber viel Speicherplatz. Durch logischen Vergleich des eingegebenen Zeichens mit den erlaubten kann Speicherplatz gespart werden.
'a$<>chr$(32)' hat den Wert 0, wenn A$=" " und –1, wenn A$<>" ". Mit dem ON-GOTO-Befehl kombiniert, kann man Programmzeilen sparen. Beim C128 kann hierbei auch der INSTR-Befehl verwendet werden.

Dieses Wissen kann man benutzen, um alle Files ordnungsgemäß zu schließen:

```
10 open1,5:open3,6:open2,7:open8,3
20 fori=1topeek(152)print"Peek(600+i)"offen":next:
   print
30 f=peek(601):n=peek(152):print"peek(601):"f,
   "peek(152):"n
40 if peek(152)>0 then closef:goto30
```

Noch einfacher geht es mit dieser Unterroutine, die nur mit GOSUB 1000 angesprungen werden muß:

```
1000 if peek(152)>0 then close peek(601):goto1000
1010 return
```

Beispiele:

```
033C    LDA  #$04
033E    CMP  $028D
0341    BNE  $033C
```

Der Inhalt der Speicherstelle $028D (SHIFT/CTRL-Flagge) wird mit dem Akku (=4) verglichen. Solange die CTRL-Taste nicht gedrückt wird, ist die Zero-Flagge gesetzt und der BNE-Befehl verzweigt nach $033C (Schleife).

CMP $0277,X: Der Inhalt von $0277+X (X=Inhalt des X-Registers) wird mit dem Akkuinhalt verglichen.

Einer solchen formatierten Leerdiskette kann man dann auch bei warmem Laufwerk einen neuen Namen geben, indem ohne ID formatiert wird:

```
open1,8,15,"n:name":close1
```

Oder beim C128:

```
header"name",d0
```

Es ist beim Laufwerk 1541 wichtig, daß die IDs verschiedener Disketten sich unterscheiden. Will man einen 2. Satz formatieren, so ändert man ',a' in ',b'...

Beispiele:

```
033C LDX  #$00
033E INX
0341 CPX  #$FF
0344 BNE  $033E
```

Das X-Register wird mit INX immer um 1 erhöht und dann das X-Register mit 255 verglichen. Solange der Vergleich ungleich 0 ist, wird zu INX verzweigt.
CPY $0277: Der Inhalt von $0277 wird mit dem Y-Register verglichen.

Man kann auch mehrere Befehle hintereinander abschicken. Dazu wird zuerst mit OPEN15,8,15 der Befehlskanal geöffnet. Danach können die Befehle mit PRINT#15,... an die Floppy übermittelt werden.

Ein sinnvolles Beispiel dafür ist das Erzeugen eines Backup-Files. Dazu wird das File p$ bak (falls schon vorhanden) gelöscht und p$ in p$.bak umbenannt:

```
10 open15,8,15,"s:"+p$+".bak":print#1,"r:"+p$
   +".bak"+"="+p$:close15
```

'Branch if Not Equal' (Verzweige, wenn Ergebnis der letzten Operation ungleich Null war)

HEX-Code	Assembler	Takte/Adressierung
DO OP	BNE $HILO	2 relativ

Es werden keine Register-Flaggen beeinflußt.

Beispiel:

```
033E INX
0341 CPX  #$FF
0344 BNE  $033E
```

Solange X kleiner ist $FF wird nach $033E verzweigt.

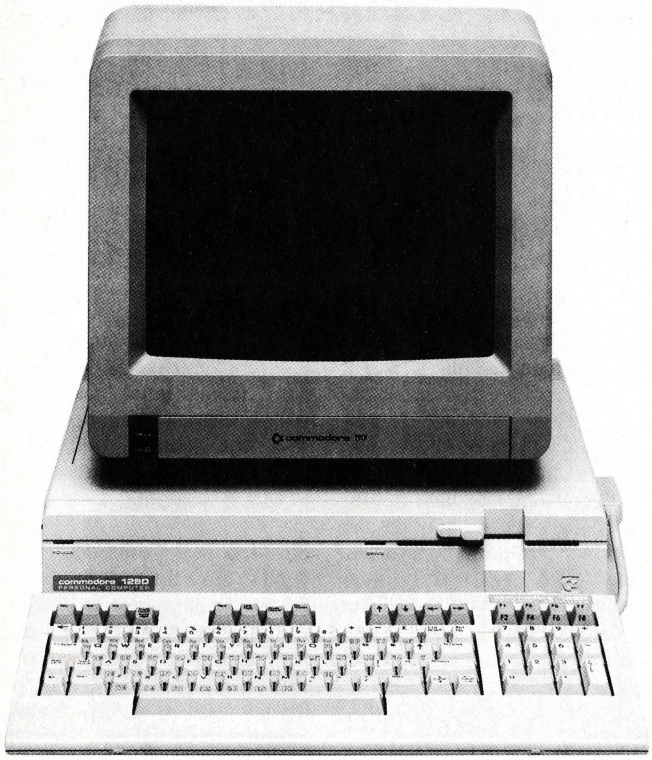
Beispiele:

```
 1 print"Taste druecken'"
 2 geta$:on (a$<chr$(1))+2 goto2,3
 3 print"OK"
 4 print"SPACE druecken!"
 5 geta$:on (a$<>chr$(32))+2 goto5,6
 6 print"OK"
 7 geta$:on (a$<chr$(1))+2 goto7:print"fortsetzung"
 8 geta$:on -(a$="a")-2*(a$+"b") goto9,10:goto8
 9 print"9"
10 print"10"
```

Erklärungen:

In Zeile 1000 wird das File zum Lesen geöffnet und gleich wieder geschlossen. Da kein Schreibversuch unternommen wurde (PRINT#8,...), blinkt bei nicht vorhandenem File nur die Kontrollampe an der Floppy. In Zeile 1010 wird der Fehlerkanal ausgelesen. In ft steht dann der Fehlertyp (z.B. 62 bei 'FILE NOT FOUND') und in f$ der Text.

Sollte mit Einschalttest (Karteikarten von CW 7/86) verbunden sein.



```
10 rem =checksummer==c16 c64 c128==
20 rem (p) 05/87 commodore welt  ==
30 rem ============================
40 rem (c) alfons mittelmeyer    ==
50 rem                           ==
60 rem c16/116/plus4             ==
70 rem c64                       ==
80 rem c128 (40-zeichen)         ==
90 rem ============================
100 rem --------------------------
110 rem   grundroutine (c16)
120 rem --------------------------
130 data165,059,072,165,060,072,032
140 data086,137,104,133,060,104,133
150 data059,152,072,160,000,165,020
160 data024,101,021,170,024,144,011
170 data201,032,240,006,138,024,113
180 data059,234,170,200,177,059,234
190 data208,240,169,031,072,138,074
200 data074,074,074,072,138,041,015
205 data072,169,031,072,162,003,104
210 data024,105,129,157,000,012,202
220 data016,246,104,168,096
230 lt=peek(772):ht=peek(773)
240 fori=312to386:readx:pokei,x:nex
t
250 iflt<>124then350
260 rem --------------------------
270 rem        anpassung c64
280 rem --------------------------
290 fori=312to317:pokei,234:next
300 fori=321to326:pokei,234:next
310 fori=1to6:readad:readx:pokead,x
:next
320 poke380,4:poke319,lt:poke320,ht
:goto430
330 data346,121,347,000,348,002
340 data351,185,352,000,353,002
350 iflt<>13then430
360 rem --------------------------
370 rem anpassung c128 (40 zeichen)
380 rem --------------------------
390 restore410:poke332,22
400 poke335,23:goto310
410 data313,061,316,062,323,062
420 data326,061,347,061,352,061
430 poke772,056:poke 773,1
440 rem --------------------------
450 rem ergaenzung 10/87
460 rem --------------------------
470 poke 345,10:new
480 rem ============================
490 rem = fuer hefte cw 7/87 bis  =
500 rem = cw 9/87 sowie cw128 5/87=
510 rem = und c16 6/87 ist die    =
520 rem = poke-anweisung in zeile =
530 rem = 470 wegzulassen         =
540 rem ============================
```

schehen, daß nach dem Laufenlassen eines Programmes weder der Checksummer noch sonst etwas mehr funktioniert, auch wenn dies bisher ohne Checksummer nicht der Fall gewesen sein sollte. Also bitte sichern Sie in jedem Falle Ihre Programme, bevor Sie sie ausprobieren.

Ein paar Dinge sollten Sie noch wissen. Wir drucken in unseren Listings des öfteren Punkte statt Leerzeichen. Wenn Ihnen nun aber Leerzeichen besser gefallen, so liefert der Checksummer natürlich eine falsche Summe. Wenn Sie diese auf Richtigkeit überprüfen wollen, so können Sie dies tun, indem Sie sie zuerst einmal so wie im Heft abtippen, und nachher, nachdem Sie sie nachgeprüft haben, einfach wieder die Punkte durch Leerzeichen ersetzen.

A. Mittelmeyer

Der C128 steht Kopf

Mit diesem kurzen Listing wird der Zeichensatz des Commodore 128 kurzerhand umgedreht. Nach dem Start braucht der Computer allerdings erst einmal etwas Zeit, um alle Zeichen umzurechnen.
Der neue Zeichensatz kann dann mit POKE 2604.24 eingeschaltet und mit POKE 2604.20 ausgeschaltet werden. □

```
100 rem kopfstand                          <kl>
110 rem von sascha kuczil                  <ge>
120 fast:rem fast-modus ein                <ep>
130 graphic1,1:graphic0                    <cn>
140 trap 190:rem bei fehler in zei
le 70                                      <dj>
150 bank 14:a=0:rem in bank 15             <be>
160 for t=0 to 4100 step 8                 <mf>
170 a=a+1:if a=9 then a=0:next             <lg>
180 poke 8200+t-a-1,peek(53248+a+t
-1):goto 170                               <mh>
190 slow:rem 40-zeichen-modus              <bf>
```

Das Superdirektory

Das Programm „Easyload“ erleichtert das Laden von Programmen erheblich.
Nach dem Starten des BASIC-Laders wird das Maschinenprogramm in den Bereich von 52800 bis 53233 geschrieben. Dieser Bereich kann dann in einem Monitorprogramm mit dem Befehl ‘S “Easyload“,08,CE40, CFF2 auf Diskette gespeichert werden. Mit Sys 52800 wird das Programm initialisiert. Um nun das erweiterte Directory auf den Bildschirm auszugeben, muß die CTRL- zusammen mit der D-Taste betätigt werden. Natürlich kann die Ausgabe mit CTRL verlangsamt und mit RUN/STOP unterbrochen werden.
Die Ausgabe des Directory erfolgt,ohne ein eventuell im Speicher stehendes BASIC-Programm zu löschen. □

Easyload

```
10 rem ========================64   <bf>
20 rem (p) commodore welt        ==   <hf>
30 rem ==========================   <mm>
40 rem (c) by                    ==   <pp>
50 rem     matthias kling        ==   <lc>
60 rem                           ==   <nd>
70 rem version 2.0   40z./ascii==   <le>
80 rem  c-64  floppy             ==   <ja>
90 rem ==========================   <km>
95 gosub 60000                       <jp>
100 rem *** easyload.52800.d ***     <ko>
110 for i = 52800 to 53233           <aj>
120 read q : poke i,q : p=p+q : z=
z+1                                  <hf>
130 h=int(z/16) : if z<>16*h then
next                                 <ma>
140 read q : if q=p goto 160         <mj>
150 print cl$"fehler in"107+h-(z/1
6<>h)"!" : end                       <mo>
160 p=0 : if i <= 53233 then next    <ob>
170 print cl$"programm ist initial
isiert!" : sys 52800 : new           <fm>
180 data 120,169,077,141,143,002,1
69,206,141,144,002,088,096,165,197
,201,2061                            <hp>
190 data 018,208,007,173,141,002,2
01,004,240,003,076,072,235,169,000
,133,1682                            <hj>
200 data 207,169,001,133,204,133,2
53,032,068,229,169,207,160,207,032
,030,2234                            <co>
210 data 171,169,034,032,210,255,1
69,036,032,210,255,169,034,032,210
,255,2273                            <fd>
220 data 169,215,160,207,032,030,1
71,169,036,133,003,169,003,133,187
,169,1986                            <hk>
230 data 000,133,188,169,001,133,1
83,169,008,133,186,169,096,133,185
,032,1918                            <pd>
240 data 213,243,165,186,032,180,2
55,165,185,032,150,255,160,003,132
,002,2358                            <jd>
```

Tips-Too

```
250 data 032,165,255,133,003,164,1
44,240,003,076,176,207,032,165,255
,164,2214                            <gn>
260 data 144,240,003,076,176,207,1
64,002,136,208,227,166,253,240,011
,166,2419                            <kn>
270 data 003,032,205,189,032,063,1
71,076,234,206,162,001,134,006,166
,003,1683                            <da>
280 data 134,251,133,252,005,251,2
08,002,133,006,169,000,133,004,133
,212,2026                            <lc>
290 data 133,005,032,165,255,164,1
44,240,003,076,176,207,170,240,043
,164,2217                            <cm>
300 data 253,240,006,032,210,255,0
76,242,206,224,034,208,011,160,001
,132,2290                            <eb>
310 data 005,165,212,073,001,133,2
12,138,164,212,208,004,224,032,240
,210,2233                            <cp>
320 data 166,004,230,004,157,000,0
02,076,242,206,164,253,208,105,164
,005,1986                            <gi>
330 data 240,091,165,004,233,002,1
70,189,000,002,201,080,240,004,160
,000,1781                            <bj>
```

```
340 data 132,006,164,006,208,007,1
69,211,160,207,076,081,207,169,207
,160,2170                                <ee>
350 data 207,032,030,171,165,004,0
72,233,003,133,004,160,000,032,182
,207,1635                                <ed>
360 data 166,214,164,006,208,009,1
60,028,024,032,240,255,076,124,207
,160,2073                                <aj>
370 data 022,024,032,240,255,169,2
20,160,207,032,030,171,104,133,004
,164,1967                                <ia>
380 data 002,032,182,207,032,063,1
71,032,199,207,076,151,207,032,199
,207,1999                                <ff>
390 data 169,227,160,207,032,030,1
71,032,215,170,160,002,169,000,133
,253,2130                                <am>
400 data 165,145,201,127,240,003,0
76,174,206,169,129,160,163,032,030
,171,2191                                <ba>
410 data 032,066,246,076,134,227,1
32,002,230,002,185,000,002,032,210
,255,1831                                <kf>
420 data 164,002,196,004,208,240,0
96,166,251,165,252,032,205,189,096
,076,2342                                <ad>
```

Is-Utilities

```
430 data 207,032,000,032,032,032,0
00,044,056,013,013,000,044,056,044
,049,0654                                <ke>
440 data 032,032,000,032,066,076,0
79,069,067,075,069,032,070,082,069
,073,0923                                <cm>
450 data 046,000,0046                    <fj>
60000 rem nachspann ==============
=                                        <da>
60010 rem * farbcodes/steuercodes
*                                        <kg>
60020 cl$=chr$(147)                      <bg>
60030 return                             <fo>
```


BASIC in allen Sprachen

Mit diesem Programm können die Befehle und die Fehlermeldungen des C128 verändert werden. Das Programm bietet auch die Möglichkeit, den erstellten Befehlssatz abzuspeichern und später wieder einzuladen. Nach Anwahl des Editors werden alle Befehle und Fehlermeldungen nacheinander ausgegeben. Nach jedem Fragezeichen, welches nach jedem zu verändernden Befehl steht, kann eine andere Zeichenfolge oder die Taste „Return", falls der Befehl nicht geändert werden soll, eingegeben werden. Die einzige Beschränkung besteht in der Zeichenlänge, die neuen Anweisungen müssen genau so lang wie die alten Befehle sein. □

Eigener Befehls-Satz

```
10 rem ========================64   <bf>
20 rem (p) commodore welt       ==   <hf>
30 rem ==========================   <mm>
40 rem (c) by                   ==   <pp>
50 rem     dietmar schorr       ==   <hi>
60 rem                          ==   <nd>
70 rem version 2.0   40z./ascii==   <le>
80 rem  c-64  floppy            ==   <ja>
90 rem ==========================   <km>
95 gosub 60000                       <jp>
100 fora=49152to49175:readx:pokea,
x:next                               <ak>
110 fora=28672to28874:readb:pokea,
b:next                               <hj>
120 clr                              <im>
130 sys49152:poke1,54                <dg>
140 printcl$"            "rn$"(c) 1
987 by s.c.s"                        <co>
150 print"       "rn$"written by d
ietmar schorr"                       <ie>
160 printc4$"ein programm fuer all
 die leute die ein-"                 <em>
170 print"en eigenen befehlssatz u
nd eigene fehler"                    <kl>
180 print"meldungen haben wollen !
"                                    <nn>
190 printleft$(qd$,4)"    sollen w
ir beginnen?"                        <ki>
200 geta$:ifa$=""then200             <gk>
210 goto400                          <nc>
220 printcl$                         <fk>
230 i=0                              <on>
240 h=peek(41118+a):ifa=255then280   <je>
250 i=i+1                            <bm>
260 ifh<>32andh<65andh<>39orh>90th
en300                                <kp>
270 printchr$(peek(41118+a));        <ec>
280 a=a+1:goto240                    <bc>
```

```
290 end                                          <mg>
300 printchr$(abs(peek(41118+a)-12
8));" = ";                                       <hc>
310 a$="":inputa$:ifa$=""then390                 <po>
320 ifa$="s.c.s"then400                          <jo>
330 iflen(a$)<>ithena=a+1-i:printc
4$"bitte"i"nicht"len(a$)"buchstabe
n":goto230                                       <jl>
340 forj=1tolen(a$)-1                            <ga>
350 poke41117+a+j+1-i,asc(mid$(a$,
j,1))                                            <fh>
360 next                                         <fe>
370 ifj>len(a$)thenj=j-1                         <nl>
380 poke41117+a+j+1-i,asc(mid$(a$,
j,1))+128                                        <ld>
390 a=a+1:ifa<650then230                         <ke>
400 printcl$tab(16)rn$"menue"                    <jh>
410 printleft$(qd$,5)tab(14)"1) ed
itor"                                            <fj>
420 printc4$tab(14)"2) speichern"                <ib>
430 printc4$tab(14)"3) laden"                    <md>
440 printc4$tab(14)"4) ende"                     <mb>
450 geta$:ifa$<>"1"anda$<>"2"anda$
<>"3"anda$<>"4"then450                           <em>
460 ifa$="1"thenclr:goto220                      <pb>
470 ifa$="4"thenprintcl$:end                     <nj>
480 printleft$(qd$,4)"     1) flopp
y   oder   2) datasette"                         <of>
490 getb$:ifb$<>"1"andb$<>"2"then4
90                                               <cj>
500 ifb$="2"then540                              <gn>
510 open1,8,0:close1:ifst<>-128the
n540                                             <ec>
520 printc4$c4$c4$"   d e v i c e
 n o t  p r e s e n t":forq=0to150
:next:sys28672                                   <cj>
530 goto400                                      <oo>
540 printc4$c4$"name:";:inputn$                  <fo>
550 ifb$="2"thenpoke16000,1:goto57
0                                                <co>
560 poke16000,8                                  <kg>
570 ifa$="2"thensys28682n$:goto400               <da>
580 sys28759n$:goto400                           <ko>
590 data160,0,132,248,169,160,133,
249,177,248,145,248,200,208,249,23
0,249                                            <kb>
600 data169,192,197,249,208,241,96               <nh>
610 data238,32,208,165,203,201,64,
240,247,96,76,16,112,54,208,34,160
,0,132                                           <hg>
620 data248,169,160,133,249,169,20
3,133,250,169,112,133,251,177,248,
145,250                                          <oe>
630 data200,208,249,230,249,230,25
1,169,164,197,249,208,239,169,209,
133,174                                          <de>
640 data169,116,133,175,169,142,13
3,193,169,112,133,194,32,87,226,16
0,2,169                                          <gj>
650 data2,174,128,62,32,186,255,32
,234,245,144,3,76,249,224,96,32,87
,226,160                                         <gn>
660 data2,169,2,174,128,62,32,186,
255,32,213,255,144,3,76,249,224,16
0,0,132                                          <db>
670 data250,169,160,133,251,169,20
3,133,248,169,112,133,249,177,248,
145,250                                          <hf>
680 data200,208,249,230,249,230,25
1,169,164,197,251,208,239,96,160,0
,132,248                                         <hh>
690 data169,160,133,249,177,248,14
5,248,200,208,249,230,249,169,192,
197,249                                          <pk>
700 data208,241,169,54,133,1,160,0
,132,248,169,160,133,249,169,203,1
33,250                                           <nh>
710 data169,112,133,251,177,250,14
5,248,200,208,249,230,249,230,251,
169,164                                          <bb>
720 data197,249,208,239,96                       <lo>
60000 rem nachspann ==============
=                                                <da>
60010 rem * farbcodes/steuercodes
*                                                <kg>
60020 c4$=chr$(017):rn$=chr$(018)                <pj>
60030 cl$=chr$(147)                              <pn>
60040 rem ********* zeichenfolgen
*                                                <gd>
60050 for q=1 to 40                              <dg>
60060 qd$=qd$+c4$                                <ed>
60070 next q                                     <ji>
60080 return                                     <md>

ready.
```

La Paloma

Geben Sie das Listing mit unserem Checksummer ein und starten Sie das Programm mit Run. Vorher sollten Sie aber nicht vergessen, das Spiel abzuspeichern. Ist das Titelbild aufgebaut,so drücken Sie die SPACE-Taste. Erschrecken Sie nicht, wenn der Bildschirm gelöscht wird. Ihr Computer ist nicht abgestürzt! Vielmehr schaltet das Programm in den FAST-Modus, um die Grafik schneller zu erzeugen.
Nach einigen Sekunden ist eine Landschaft aufgebaut und Sie können mit dem Spiel beginnen. Ihr Gewehr (Fadenkreuz) steuern Sie mit Hilfe des in Port 2 eingesteckten Joysticks. Schießen Sie durch Drücken des Feuerknopfes. Aber aufgepaßt. Sie haben jeweils nur einen Schuß pro Tontaube. Das Ziel des Spiels ist alle 25 Tontauben „zu erwischen".
Nach einem Durchgang bekommen Sie mitgeteilt, wieviele Treffer Sie erzielt haben. Wollen Sie ein neues Spiel starten, so drücken Sie eine beliebige Taste und anschließend den Feuerknopf.
Nun viel Spaß und Weidmannsheil. □

La Paloma

```
10 rem ========================128   <ob>
20 rem (p) commodore welt       ==   <hf>
30 rem ==========================   <mm>
40 rem (c) by                   ==   <pp>
50 rem     helmut karner        ==   <mo>
60 rem                          ==   <nd>
70 rem version 7.0   40z./ascii==   <ah>
80 rem pc-128 floppy/datasette  ==   <ho>
90 rem ==========================   <km>
95 gosub 60000                       <jp>
100 color0,13:color4,13:color5,2     <po>
110 printcl$rn$c4$left$(qr$,9)"T
"left$(qr$,9)"T          "           <oi>
120 printrn$left$(qr$,9)"T  "left$
(qr$,9)"T          "                 <gf>
130 printrn$left$(qr$,9)"T  "left$
(qr$,9)"T  "left$(qr$,4)"T  "        <on>
140 printrn$left$(qr$,9)"T  "left$
(qr$,9)"T  "left$(qr$,4)"T  "        <dj>
150 printrn$left$(qr$,9)"T  "left$
(qr$,9)"T  "left$(qr$,4)"T  "        <ki>
160 printrn$left$(qr$,9)"T  "left$
(qr$,9)"T          "                 <jd>
170 printrn$left$(qr$,9)"T  "left$
(qr$,9)"T          "                 <pp>
180 printrn$left$(qr$,9)"T  "left$
(qr$,9)"T  "left$(qr$,4)"T  "        <bh>
190 printrn$left$(qr$,9)"T  "left$
(qr$,9)"T  "left$(qr$,4)"T  "        <gj>
200 printrn$left$(qr$,9)"T  "left$
(qr$,9)"T  "left$(qr$,4)"T  "        <ng>
210 printrn$left$(qr$,9)"T
"c3$c3$c3$"T  "left$(qr$,4)"T  "     <cb>
220 printrn$left$(qr$,9)"T
"c3$c3$c3$"T  "left$(qr$,4)"T  "     <fd>
230 printc4$c4$c3$c3$rn$"T    "c3$
"T    "c3$"T "left$(qr$,4)"T    "c
3$"T"yq$c3$c3$z8$" "c3$"T    "      <dk>
240 printc3$c3$rn$"T "c3$"T "c3$"T
 "c3$"T "c3$"T "left$(qr$,4)"T "c3
$"T "c3$"T "yq$z8$"  "c3$"T "c3$"T
 "                                   <ph>
250 printc3$c3$rn$"T "c3$"T "c3$"T
 "c3$"T "c3$"T "left$(qr$,4)"T "c3
$"T "c3$"T     "c3$"T "c3$"T "      <ip>
260 printc3$c3$rn$"T    "c3$"T
"c3$"T "left$(qr$,4)"T "c3$"T "c3$
"T "rf$yq$z8$rn$"T "c3$"T    "      <la>
270 printc3$c3$rn$"T "left$(qr$,4)
"T "c3$"T "c3$"T "left$(qr$,4)"T "
c3$"T "c3$"T "c3$c3$"T "c3$"T "c3$
"T "                                 <jl>
280 printc3$c3$rn$"T "left$(qr$,4)
"T "c3$"T "c3$"T "left$(qr$,4)"T "
c3$"T "c3$"T "c3$c3$"T "c3$"T "c3$
"T "                                 <pc>
290 printc3$c3$rn$"T "left$(qr$,4)
"T "c3$"T "c3$"T    "c3$"T    "c3$
"T "c3$c3$"T "c3$"T "c3$"T "         <ij>
300 printc4$c3$rn$"c.1988 fuer den
 c128 von karner helmut":getkeya$    <hm>
310 fast:color0,7:color1,6:color4,
15:graphic1,1:envelope0,0,10,5,9,3
:play"t0"                            <mf>
320 circle,10,10,8:draw,10,7to10,1
3,7,10to13,10:sshapea$,0,0,23,21:s
prsava$,1                            <ee>
330 circle,40,10,3,2:paint,40,10:s
shapea$(0),30,0,53,21:sprsava$(0),
6:sprsava$(0),7                      <ag>
340 fort=1to6:graphic1,1:box,10-t-
2,10-t-2,10-t,10-t,,1:box,10-t-2,1
0+t,10-t,10+t+2,,1:box,10+t,10-t-2
,10+t+2,10-t,,1:box,10+t,10+t,10+t
+2,10+t+2,,1                         <ec>
350 sshapea$(t),0,0,23,21:next:gra
phic1,1:box,0,0,23,21,,1:sshapea$,
0,0,23,21:sprsava$,2:sprsava$,3      <kd>
360 graphic1,1:circle,6,10,4,18,,,
96,120:paint,6,10:sshapea$,0,0,23,
21:sprsava$,4                        <af>
370 graphic1,1:circle,17,10,4,18,,
,264,120:paint,17,10:sshapea$,0,0,
23,21:sprsava$,5                     <dl>
380 a$="
                ":graphic1,1:z$(0)="
 ":z$(1)="."                         <mj>
390 x=0:y=80:yu=80:locate0,80:deff
na(a)=cos(w* /180)*e:deffnb(b)=sin
(w* /180)*(e-20)                     <gp>
400 x=x+9:y=y+rnd(0)*7-3             <cc>
410 ify>yuthenyu=y:j=yu/8+1:ifj-in
t(j)=0thenj=j+1                      <pi>
420 drawtox,y:ifx<320goto400         <ag>
430 fort=jto21:forb=0to39:char,b,t
,z$(rnd(0)*2),1:nextb,t              <ca>
440 paint,0,83:fort=0to39:char,t,2
1,"#",1:next                         <kj>
450 color1,14:fort=22to24:char,0,t
```

```
,a$,1:next                                    <ih>
460 color1,2:fora=1to5:x=a*55:y=in
t(rnd(1)*50)+15:x1=x-2:y1=y-2:f=rn
d(0)*10+20:forw=0to380step10                  <ba>
470 e=rnd(1)*3+f:x2=x+fna(a):y2=y+
fnb(b):draw,x1,y1tox2,y2:x1=x2:y1=
y2:next:paint,x,y:next                        <eh>
480 movspr1,175,200:sprite1,1,1:fo
rt=2to5:spritet,1,10:next:movspr2,
315,160:movspr4,315,147:movspr3,30
,160:movspr5,30,147:slow:o=25:p=0             <hi>
490 j=joy(2):ifj<128goto490:elsefo
rt=1to100:next                                <dl>
500 o=o-1:ifo<0goto610                        <lb>
510 ifrnd(0)<.5thenx=50:w=70:w1=85
:elsex=295:w=290:w1=275                       <ac>
520 movspr6,x,150:sprite6,1,1:movs
pr6,w#10:movspr7,x,150:sprite7,1,1
2:movspr7,w1#10:z=ti                          <hc>
530 j=joy(2)-1:ifj=-1orj=127thenmo
vspr1,180#3:elseifj=0thenmovspr1,j
#12:elsemovspr1,j*45#15                       <og>
540 ifti-z>100goto600                         <ej>
550 v=bump(1)                                 <ca>
560 ifj<127orsz=1goto530                      <no>
570 sz=1:play"c"                              <ip>
580 ifv<>33goto530                            <nd>
590 sprite7,0:movspr1,0#0:movspr6,
w#2:fort=1to6:sprsava$(t),6:fora=1
to50:nexta,t:p=p+1                            <mc>
600 sz=0:sprite7,0:sprite6,0:sprsa
va$(0),6:movspr1,rnd(0)*210+80,200
:movspr1,0#0:fort=1to500:next:fort
=1tornd(0)*500+100:next:goto500               <ai>
610 color1,14:char,10,24,"geschoss
ene tauben:":char,29,24,str$(p):po
ke208,0:getkeya$:char,10,24,"
                    ",1:goto480               <ef>
620 sound1,1000,5,2,1000,,3:sound2
,10000,5,2,100,,3                             <ag>
60000 rem nachspann ============              <pj>
60010 rem *farbcodes/steuercodes *            <kg>
60020 c4$=chr$(017):rn$=chr$(018)             <pj>
60030 c3$=chr$(029):rf$=chr$(146)             <fm>
60040 c1$=chr$(147)                           <gg>
60050 rem ***zeichensatz/graphik *            <gd>
60060 z8$=chr$(169):yq$=chr$(223)             <ei>
60070 rem *********zeichenfolgen *            <pg>
60080 for q=1 to 40                           <kh>
60090 qr$=qr$+c3$                             <nb>
60100 next q                                  <ba>
60110 return                                  <pp>
```

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 Diskenteninhaltsverzeichnis 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 sich 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 – Gruppe 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

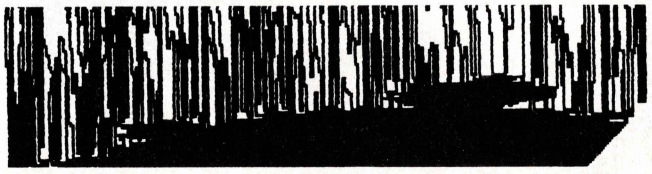
Zum Abschluß dieses „Krankenhaus-Reports" noch eine simple Methode, den Virus wieder loszuwerden. Dazu benötigen Sie einen Disketten-MONITOR. Suchen Sie die Spur und den Sektor des eigentlichen Programms, bei unserem Beispiel war es Spur 21, Sektor 0.
Dieser Wert wird im DIRECTORY im 4. und 5. Byte übernommen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



COMPUTERN LEICHT GEMACHT

14,80

Das PC-Magazin

NEU

Jetzt an ausgewählten Kiosken und im Bahnhofs-Buchhandel