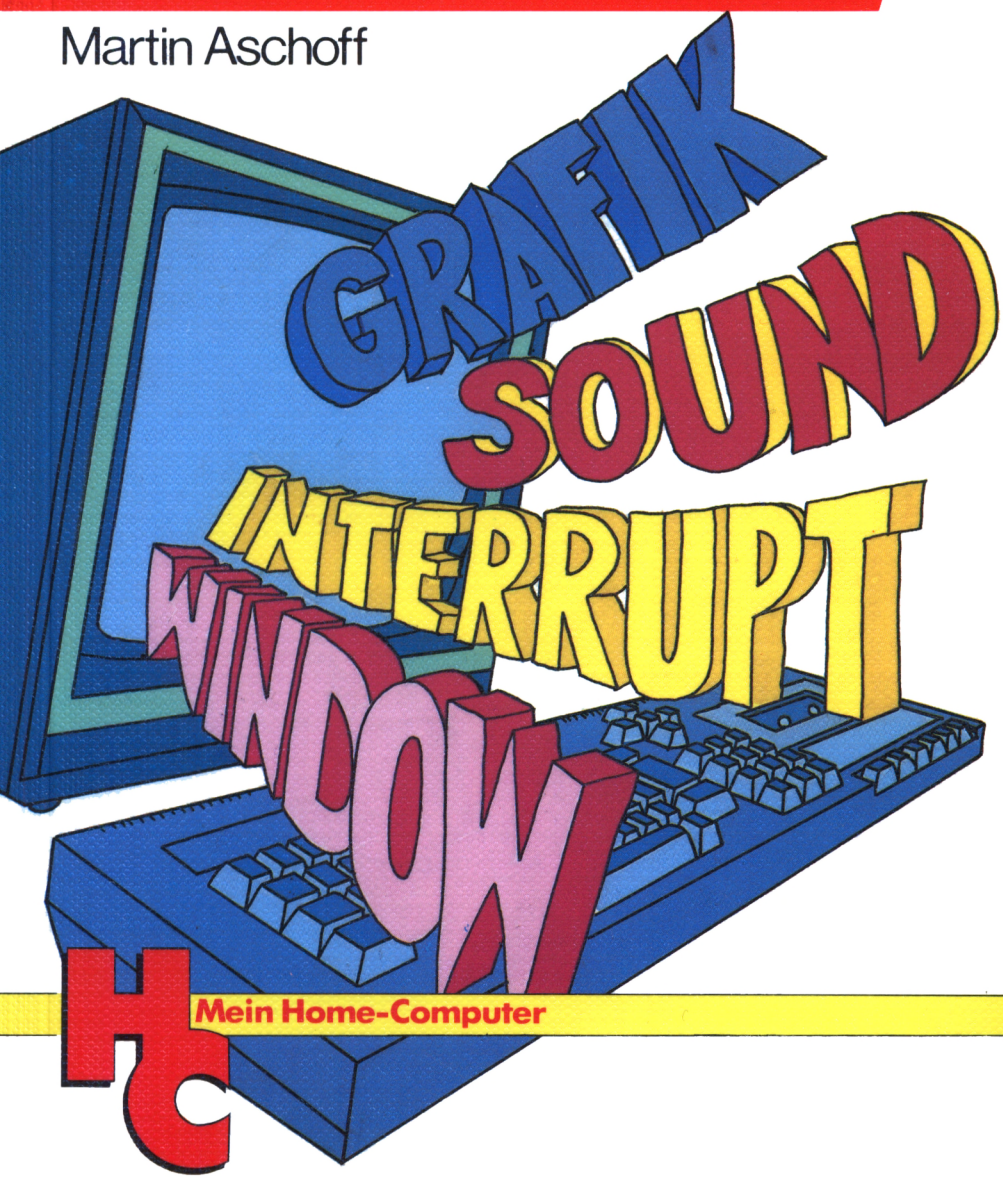
aktiv computern

Was der CPC 464 alles kann

Martin Aschoff



Mein Home-Computer

Martin Aschoff
Was der CPC 464 alles kann

HC – Mein Home-Computer

Martin Aschoff

Was der CPC 464 alles kann

Das Buch, das nach dem Handbuch kommt

VOGEL-BUCHVERLAG
WÜRZBURG

Um Ihnen das fehlerträchtige Eintippen seitenlanger Listings zu ersparen, ist die Software dieses Buches auf Kassette erhältlich. Informationen erhalten Sie von:
Martin Aschoff, W.-Glässing-Straße 31, 6100 Darmstadt

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Aschoff, Martin:
Was der CPC 464 alles kann : d. Buch, d. nach d. Handbuch kommt / [Martin Aschoff]. – 1. Aufl. – Würzburg : Vogel, 1985.
(HC – mein Home-Computer)
ISBN 3-8023-0841-7
NE: HST

ISBN 3-8023-0841-7
1. Auflage. 1985

Umschlaggestaltung: Bernd Schröder, Böhl
Satz: Schmitt u. Köhler, Würzburg
Druck: Alois Erdl KG, Trostberg

Inhaltsverzeichnis

Vorwort

Der CPC 464 ist auf dem besten Wege, sich zum Marktführer unter den Homecomputern zu entwickeln.

Was noch fehlt, ist eine vernünftige, übersichtliche Dokumentation zu dem Gerät, die auch den fortgeschrittenen Anwender nicht im Stich läßt.

Dieses Buch wird Ihnen helfen, Ihren Computer besser zu verstehen und effektiver einzusetzen.

Es stellt für jeden Besitzer des CPC 464 eine Fundgrube an neuen, wichtigen Informationen dar, und beschränkt sich nicht auf die bloße Wiedergabe des Stoffes aus dem Bedienungshandbuch des CPC 464.

Selbst wenn Sie das Handbuch vollständig durchgearbeitet haben, werden bei Ihnen noch viele Fragen offen sein. Weiterhin werden Sie zusätzliche Informationen, Anregungen und Software zu Ihrem Gerät suchen. Um diese Wünsche optimal zu erfüllen, ist dieses Buch in drei große Bereiche aufgeteilt.

Der erste Teil behandelt die *Hardware* des CPC 464. Ein ausführlicher Einblick in die Funktionsweise des Computers macht Sie näher mit Ihrem Gerät vertraut. Die Bedeutung aller Schnittstellen wird erläutert, und es wird erklärt, welche Peripherieeinheiten Sie mit welchem Erweiterungsanschluß verbinden können.

Der zweite Teil des Buches beschäftigt sich eingehend mit der *Firmware* des CPC 464. Ihnen werden Tips zum Programmieren in BASIC und Tricks zum Umgang mit dem Betriebssystem vermittelt. Es folgen Beispiele zur optimalen Nutzung der besonderen Eigenschaften des CPC 464, die ihn von anderen Homecomputern abheben (Grafik, Ton, Interrupt und Windowing).

Der dritte Teil des Buches liefert *Software* zum CPC 464. Weil ein Computer immer nur so gut wie die verfügbare Software sein kann, werden mehrere nützliche und in ihrer Funktion ausgefeilte Programme vorgestellt und vollständig dokumentiert.

Als besonderer Leckerbissen beendet ein Kapitel über das Programmieren von Spielen mit bewegter Grafik und Ton dieses Buch.

Bei den Programmen (abgedruckten Listings) handelt es sich ausschließlich um Originalausdrucke einwandfrei gelaufener Programmversionen. Alle Programme dieses Buches sind daher auf dem CPC 464 zu 100 % lauffähig.

Darmstadt Martin Aschoff

0
Einführung

Mit dem CPC 464 ist ein Homecomputer auf den Markt gekommen, der die Leistungen eines Personalcomputers verspricht und hält! Dieser Computer ist nach dem Kauf ohne jedes Zubehör sofort einsatzbereit, weil Kassettenrecorder und Monitor bereits im Preis enthalten sind. Trotzdem kostet er nicht mehr, als seine Konkurrenz ohne diese Extras.

Der CPC 464 ist ein Computer für jedermann. In der Grundausstattung ist er mit dem Farbmonitor für Spielernaturen ebenso geeignet wie für den Geschäftsmann mit einem monochromen Monitor für die Darstellung von 80 Zeichen je Zeile, Drucker und Diskettenstation mit Betriebssystem CP/M.

Ein gut ausgebautes Händler- und Servicenetz trägt genauso zu seiner Beliebtheit bei wie das ausgezeichnete Preis-Leistungs-Verhältnis und die solide Verarbeitung des Geräts. Der Hauptgrund für die schnelle Verbreitung des CPC 464 liegt jedoch darin, daß er über eine Reihe von bemerkenswerten Eigenschaften verfügt, die ihn deutlich von seiner Konkurrenz abheben. Als wichtigste Merkmale seien genannt:

- ☐ die hochauflösende Grafik mit 128000 Bildpunkten,
- ☐ die Möglichkeit der Darstellung von 27 Farben,
- ☐ die Fenstertechnologie mit bis zu acht Bildschirmfenstern,
- ☐ der Tongenerator mit drei voneinander unabhängigen Kanälen, 5-Ton-Warteschlange und Geräuschgenerator,
- ☐ die hervorragenden Interruptmöglichkeiten im BASIC.

Leider scheint es dem Hersteller unmöglich, auf diese Besonderheiten des CPC 464 ausführlich einzugehen. Das mitgelieferte Bedienungshandbuch des Computers ist zwar 280 Seiten stark, wiederholt sich jedoch öfters und liefert obendrein zum Teil irreführende Informationen.

Im Vorwort des Handbuchs wird darauf hingewiesen, daß ausführliche Informationen über den CPC 464 Bände füllen würden. Das ist vielleicht

übertrieben, aber um die Versäumnisse des Herstellers zu beheben und den Benutzer in die Lage zu versetzen, die Möglichkeiten seines Geräts voll auszuschöpfen, ergibt sich schnell eine Stoffsammlung an Informationen, die dem Umfang eines Buches entspricht.

Dieses Buch halten Sie jetzt in Ihren Händen! Es wird Ihnen – gründliche Lektüre vorausgesetzt – neue Dimensionen in der Anwendung des CPC 464 eröffnen.

1
Die Hardware des CPC 464

Zur Hardware eines Computers gehören alle *mechanischen* und *elektronischen* Komponenten des Geräts, bildlich ausgedrückt: alles, was Sie anfassen können.

Zuerst wird beschrieben, wie der CPC 464 von innen aussieht, und darauf folgt eine Erläuterung der wichtigsten Bestandteile des Computers und Zubehörs.

1.1 Das Innere des Computers

Wahrscheinlich sind Sie kein Elektronikfreak und haben Ihren Computer noch nicht in seine Einzelteile zerlegt. Darum wird Ihnen dieses Kapitel erläutern, wie es im Inneren des CPC 464 aussieht und was dort vor sich geht. Das Öffnen des Gehäuses ist relativ problemlos (Achtung: Vor dem Öffnen von Geräten bitte erst vergewissern, ob nicht etwaige Garantieansprüche verlorengehen.) Sie müssen lediglich sechs Schrauben an der Unterseite des Computers entfernen und sein Oberteil anheben. Das Oberteil ist allerdings immer noch über ein 20poliges Flachbandkabel, das den Kontakt zwischen Tastatur und Rechnerplatine herstellt, mit dem Computer verbunden. Um dieses Teil völlig zu entfernen, müssen Sie den Kabelstecker von der Platine lösen. Aber Vorsicht, der Stecker klemmt, und bevor Sie ihn abbrechen, verzichten Sie lieber darauf! Nachdem Sie das Computeroberteil zur Seite gelegt haben, liegt die Platine des CPC 464 frei vor Ihnen. Sie ist durch neun Schrauben mit dem unteren Teil des Gehäuses verbunden, und es ist unnötig, auch diese zu entfernen. Besonders auffällig ist der schwarze Schaumstoffwürfel auf der Rechnerplatine, der mit einer Kantenlänge von 2 cm anscheinend dem Zweck der Tastaturfederung dient. Weiterhin sind auf der Platine diverse Widerstände, einige Kondensatoren und mehrere ICs zu sehen, vor denen einer

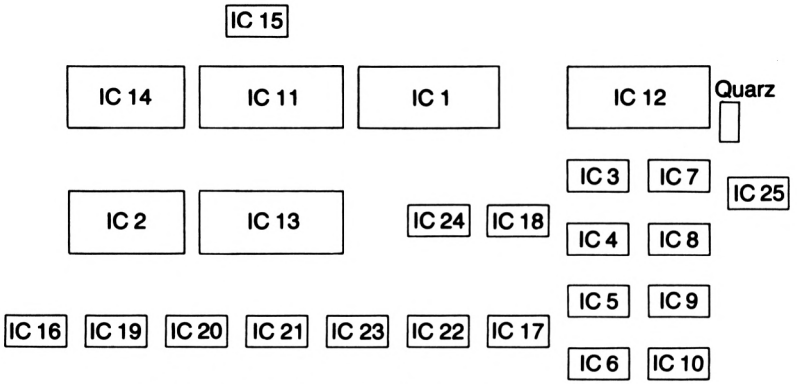


Bild 1.1 Schematische Aufsicht auf die Rechnerplatine

IC-Nr.	Bezeichnung	Bedeutung
IC 1	Z-80A-CPU	Mikroprozessor für 4 MHz Taktfrequenz
IC 2	23256	32-K Byte-ROM
IC 3–10	4164	dynamischer 64-K × 1-Bit-RAM
IC11	8255 PIO	Paralleler Ein-/Ausgabebaustein
IC12	20 RA043	Gate Array, verantwortlich für: – Speicherverwaltung – Refreshing des RAM – Videoausgangssignal – Teilung des Takts
IC13	6845 CRTC	Videosignalgenerator (CRTC)
IC14	AY-3-8912	Tongenerator
IC15	74 LS 273	8-Bit-Puffer für Centronics-Schnittstelle
IC16	74 LS 145	Decoder der Tastaturzeile
IC17	74 LS 244	8-Bit-Datenlesepuffer
IC18	74 LS 373	8-Bit-Datenschreibpuffer
IC19–22	74 LS 153	Datenselektor für Videospeicher → > CRTC
IC23	74 LS 132	NAND-Gatter für logische Verknüpfungen
IC24	74 LS 32	ODER-Gatter für logische Verknüpfungen
IC25	7400 E	NAND-Gatter zur Taktflankenerzeugung
Quarz	HC-18/U	takterzeugender 16-MHz-Quarz

Tabelle 1.1 Bedeutung der ICs des CPC 464

unter einem Kühlkörper versteckt liegt. Wenn Sie ganz genau hinsehen, finden Sie noch drei Transistoren, eine einzelne Diode und den Quarz des CPC 464. Von diesen Bauteilen sind für uns nur die ICs und der Quarz interessant. Bild 1.1 zeigt eine schematische Aufsicht auf die Platine, die nur diese Bauteile berücksichtigt. (IC 12, das Gate Array, liegt unter einem Kühlblech verborgen, weil dieser Baustein besonders viel Wärme produziert.)

Die Bedeutung der einzelnen ICs und des Quarzes wird in Tabelle 1.1 erklärt. Äußerst rechts auf der Platine befindet sich ein 8poliger Stecker, der mit einer 8adrigen Leitung die Verbindung zum Kassettenrecorder herstellt. Die Recorderelektronik braucht uns an dieser Stelle jedoch nicht näher zu interessieren. Sie besteht im Prinzip lediglich aus zwei Verstärkern für das Ein- und Ausgangssignal des Recorders sowie einem Verstärker für den Lautsprecher des CPC 464, der sich ebenfalls unter der Elektronik des Recorders befindet.

1.2 Der Mikroprozessor

Das Herzstück eines Computers besteht heutzutage nur noch aus einem einzigen Baustein, dem Mikroprozessor, der oft auch CPU (*central processing unit*) genannt wird.

Bei dem Mikroprozessor des CPC 464 handelt es sich um die *Z-80 A-CPU* der Firma Zilog, eine der am weitesten verbreiteten Zentraleinheiten für Mikrocomputer. (Das *A* ist der Herstellercode für eine höchste Arbeitsfrequenz von 4 MHz.) Der Mikroprozessor ist das Gehirn des CPC 464. Er verwaltet die Daten des Computers, führt logische und arithmetische Befehle aus, transferiert Daten und kommuniziert über geeignete Peripheriebausteine mit seiner Umwelt.

Die Befehle

Die Anzahl der Befehle, die ein Mikroprozessor verstehen und ausführen kann (z. B. Multiplikation oder Division mit einem einzigen Befehl), ist ein wichtiges Kriterium für seine Leistungsfähigkeit.

Die Z-80-CPU (das A wollen wir künftig weglassen, weil es für uns unerheblich ist) verfügt über einen Befehlssatz von 158 verschiedenen Befehlen (*Operationscodes*).

Die Register

Die Anzahl der Register eines Mikroprozessors ist ebenfalls wichtig. Je mehr Register dem Mikroprozessor zur Verfügung stehen, desto seltener muß er Daten bzw. Adressen nachladen, weil er eine größere Anzahl davon speichern kann. Die Arbeitsweise des Prozessors wird übersichtlicher, effektiver und schneller. Die Z-80-CPU beinhaltet die stattliche Zahl von 22 Registern.

Der Datenbus

Um Daten mit seiner Peripherie austauschen zu können, benötigt der Mikroprozessor einen Kanal für den Datentransport, den sogenannten Datenbus. Die Breite dieses Busses ist auch ein Anzeichen für die Leistungsfähigkeit des Mikroprozessors, denn bei einer Datenbusbreite von 16 Bit können natürlich in einer bestimmten Zeit doppelt so viele Daten übertragen werden wie bei einer Busbreite von 8 Bit.

Die meisten Mikroprozessoren für Homecomputer verfügen über eine Datenbusbreite von 8 Bit, auch die Z-80-CPU. In größeren Computern werden jedoch Mikroprozessoren mit Datenbusbreiten von 16 oder 32 Bit verwendet.

Die interne Datenverarbeitungsbreite

Ebenso wichtig wie die äußere Datenbreite ist die interne Datenverarbeitungsbreite eines Mikroprozessors.

Die Z-80-CPU kann intern 8 Bit parallel verarbeiten und zählt deshalb zur Familie der *8-Bit-Prozessoren.* Zusätzlich ist es möglich, mit den entsprechenden Befehlen je zwei 8-Bit-Register zu einem 16-Bit-Register zusammenzufassen. Die Z-80-CPU ist deshalb neben einem *reellen* 8-Bit-Prozessor auch ein sogenannter *virtueller* 16-Bit-Prozessor, weil mit einem einzigen Befehl 16 Bit lange Datenworte verarbeitet werden können. Der Vorteil dieses Verfahrens liegt klar auf der Hand:

Operationscodes können immer nur ein Register manipulieren. Wenn wir nun anstelle der 8-Bit-Register die doppelt so langen 16-Bit-Register verwenden, werden mit dem gleichen Aufwand auch doppelt soviel Daten angesprochen und verarbeitet.

Manche Mikroprozessoren können intern mehr Daten parallel verarbeiten, als sie auf einmal ausgeben können, weil die Register des Prozessors breiter als sein Datenbus sind.

Das bedeutet, daß man z.B. bei einem Mikroprozessor, der intern 32 Bit manipulieren kann, aber nur über eine Datenbusbreite von 8 Bit verfügt, 24 Datenleitungen spart und so Hardwareaufwand und -kosten erheblich reduziert – freilich auf Kosten der Verarbeitungsgeschwindigkeit!

Der Adreßbus

Neben dem Datenbus benötigt der Mikroprozessor einen Adreßbus, um seine Peripherie (*Speicherzellen* und *I/O-Ports*) adressieren zu können. Die Breite dieses Busses bestimmt, wieviel Peripherieeinheiten angesprochen werden können. Die Adreßbusbreite von Mikroprozessoren beträgt in der Regel 16 Bit, so daß 2^{16} bzw. 65536 verschiedene Einheiten verwaltet werden können. Handelt es sich bei diesen Einheiten ausschließlich um Speicherzellen, so kann der Mikroprozessor 65536 Byte bzw. 64 KByte (1 KByte = 1024 Byte) benutzen. Auf diese Weise kommt die bekannte maximale Speichergröße von 64 KByte zustande.

Das Ansprechen von I/O-Ports gestaltet sich bei manchen Mikroprozessoren ziemlich kompliziert. So muß z.B. bei der oft verwendeten *6502-CPU* für jedes Port eine Speicherzelle reserviert werden. Nicht so bei der Z-80-CPU! Diese hat den Vorteil, daß sie mit einem besonderen Befehl 256 Ports adressieren kann. Diese Portadresse wird über die unteren 8 Adreßbits mit einem zusätzlichen Portansprechsignal ausgegeben. Folglich kann mit der Z-80-CPU der volle Speicherbereich von 64 KByte ausgenutzt und trotzdem genügend I/O-Ports adressiert werden.

Über diese Ports kann man bei entsprechender Hardwarebeschaltung zwischen verschiedenen Speicherblöcken hin und her schalten, so daß maximal 256 × 64 KByte bzw. 16 MByte angesprochen werden können.

Dieses Verfahren verwendet auch die Z-80-CPU des CPC 464 in Verbindung mit dem Gate Array, um zwischen internem ROM und RAM sowie externem ROM und RAM unterscheiden zu können.

Der Steuerbus

Der dritte Bus eines Mikroprozessors ist der Steuerbus, der den *Status* des Prozessors anzeigt. Die einzelnen Signale dieses Busses geben an, ob momentan Daten gelesen oder geschrieben werden, ob auf den Speicher oder ein I/O-Port zugegriffen wird, ob eine Interruptanforderung der Peripherie vorliegt usw.

Signal	Ein-/Ausgang	Bedeutung im aktiven Zustand
D0	Ein-/Ausgang	niederstwertiges Bit des Datenbus gesetzt
D1	Ein-/Ausgang	niederwertiges Bit des Datenbus gesetzt
D2	Ein-/Ausgang	niederwertiges Bit des Datenbus gesetzt
D3	Ein-/Ausgang	niederwertiges Bit des Datenbus gesetzt
D4	Ein-/Ausgang	niederwertiges Bit des Datenbus gesetzt
D5	Ein-/Ausgang	niederwertiges Bit des Datenbus gesetzt
D6	Ein-/Ausgang	niederwertiges Bit des Datenbus gesetzt
D7	Ein-/Ausgang	höchstwertiges Bit des Datenbus gesetzt
A0	Ausgang	niederstwertiges Bit des Adreßbus gesetzt
A1	Ausgang	niederwertiges Bit des Adreßbus gesetzt
A2	Ausgang	niederwertiges Bit des Adreßbus gesetzt
A3	Ausgang	niederwertiges Bit des Adreßbus gesetzt
A4	Ausgang	niederwertiges Bit des Adreßbus gesetzt
A5	Ausgang	niederwertiges Bit des Adreßbus gesetzt
A6	Ausgang	niederwertiges Bit des Adreßbus gesetzt
A7	Ausgang	niederwertiges Bit des Adreßbus gesetzt
A8	Ausgang	niederwertiges Bit des Adreßbus gesetzt
A9	Ausgang	niederwertiges Bit des Adreßbus gesetzt
A10	Ausgang	niederwertiges Bit des Adreßbus gesetzt
A11	Ausgang	niederwertiges Bit des Adreßbus gesetzt
A12	Ausgang	niederwertiges Bit des Adreßbus gesetzt
A13	Ausgang	niederwertiges Bit des Adreßbus gesetzt
A14	Ausgang	niederwertiges Bit des Adreßbus gesetzt
A15	Ausgang	höchstwertiges Bit des Adreßbus gesetzt
$\overline{RD}$	Ausgang	Daten werden gelesen
$\overline{WR}$	Ausgang	Daten werden geschrieben
$\overline{MREQ}$	Ausgang	Adreßbus spricht Speicher an
$\overline{IORQ}$	Ausgang	Unteres Byte des Adreßbus spricht I/O-Port an
$\overline{RESET}$	Eingang	Initialisierung der CPU
$\overline{NMI}$	Eingang	nicht abschaltbare Unterbrechungsanforderung
$\overline{INT}$	Eingang	abschaltbare Unterbrechungsanforderung
$\overline{HALT}$	Ausgang	CPU wartet bis zur nächsten Unterbrechung
$\overline{WAIT}$	Eingang	CPU wartet, bis das Signal inaktiv wird
$\overline{BUSRQ}$	Eingang	CPU ist abgeschaltet, periphere Bausteine können auf Speicher und I/O-Ports zugreifen
$\overline{BUSAK}$	Ausgang	Bestätigung der CPU, daß sie abgeschaltet ist
$\overline{RFSH}$	Ausgang	Untere 7 Bit des Adreßbus enthalten Refresh-adresse für dynamische RAMs
M1	Ausgang	Erster Maschinenzyklus läuft, d.h., CPU holt Befehlscode aus dem Speicher
Φ	Eingang	Takteingang
+5V	Eingang	Versorgungsspannungseingang
GND	Eingang	Masseeingang

Tabelle 1.2 Die Signale der Z-80-CPU

Die Signale der Z-80-CPU können Sie der Tabelle 1.2 entnehmen. Der Balken über einem Signal gibt an, daß dieses Signal negative Logik verwendet, d.h. im aktiven Zustand liegt es nicht auf 1, sondern auf 0 und umgekehrt.

Neben den Anschlüssen für Daten-, Adreß- und Steuerbus besitzt der Mikroprozessor in der Regel zwei Eingänge für die Spannungsversorgung (+5 V und GND) und einen Eingang für den Takt.

Der Takt

Der Mikroprozessor kann nicht alle seine Aufgaben gleichzeitig erfüllen, vielmehr muß er seine Befehle Schritt für Schritt, gegebenenfalls in Unterschriften, ausführen. Die Ausführungszeit für einen solchen Befehl wird durch die anliegende Taktfrequenz bestimmt.

Die Zeit, in der der Mikroprozessor einen Befehl abarbeitet, wird Befehlszyklus genannt. Dieser Befehlszyklus besteht wiederum aus einem oder mehreren Maschinenzyklen. Je komplizierter ein Befehl ist, desto mehr Maschinenzyklen benötigt er zur Ausführung. Im ersten Maschinenzyklus wird immer der Befehl aus dem Speicher gelesen und decodiert, während der nächsten Zyklen wird er ausgeführt.

Ein Maschinenzyklus setzt sich nochmals aus mehreren Taktzyklen zusammen, deren Dauer *direkt* durch die anliegende Taktfrequenz bestimmt wird. Ein Taktzyklus ist genau eine Taktperiode lang.

Der erste Maschinenzyklus eines Befehlszyklus läßt sich immer in die Taktzyklen für das Laden des Befehls in die CPU, Erhöhen des Programmzählers um eins, Decodierung des Befehls und (bei einfachen Befehlen) Ausführung des Befehls sowie (bei einigen Mikroprozessoren) einen zusätzlichen Refreshtakt für dynamische RAMs aufteilen. Wenn der Befehl einfach ist, besteht der Befehlszyklus nur aus diesem ersten Maschinenzyklus, ansonsten folgen weitere Maschinenzyklen.

Je nach Komplexität des zu bearbeitenden Befehls kann sich ein Befehl aus wenigen bis zu Hunderten von Taktzyklen zusammensetzen (z.B. vorzeichenbehaftete Division zweier 32-Bit-Gleitkommazahlen). Die längsten Befehle der Z-80-CPU benötigen 23 Taktzyklen, was bei einer anliegenden Taktfrequenz von 4 MHz (Systemtakt des CPC 464) einer Zeit von 5,75 µs entspricht.

1.3 Der Speicher

Für den Computer ist der Speicher ebenso wichtig wie der Mikroprozessor. Er stellt das *Gedächtnis* des Computers dar. Ohne Speicher kann ein Mikroprozessor nur die Anzahl von Daten verarbeiten, die in seine Register passen. Mit Speicher dagegen ist die Menge der zu verarbeitenden Daten nur durch die Größe des Speichers begrenzt.

So wie der Mensch über Langzeit-, Kurzzeit- und Ultrakurzzeitgedächtnis verfügt, unterscheidet man auch beim Computer zwischen verschiedenen Speichertypen, die durch ihren Aufbau bedingt für unterschiedliche Aufbewahrungszeiten geeignet sind.

Die Register

Die Register des Mikroprozessors lassen sich mit dem menschlichen Ultrakurzzeitgedächtnis vergleichen. Hier werden nur Daten abgelegt, die unmittelbar benötigt werden, z. B. Werte für die nächste logische oder arithmetische Verknüpfung, die Adresse der nächsten Unterprogrammroutine, die Startadresse eines zu verschiebenden Datenblocks usw.

Der Arbeitsspeicher (RAM)

Dem menschlichen Kurzzeitgedächtnis entspricht der Arbeitsspeicher des Computers, im allgemeinen kurz RAM (*random access memory*) genannt. Bei diesem Typ handelt es sich um einen Schreib- *und* Lesespeicher, dessen Inhalt so lange gespeichert bleibt, wie die Versorgungsspannung anliegt. Wird der Computer ausgeschaltet, geht auch der Speicherinhalt verloren.

Beim RAM unterscheidet man zwischen statisch und dynamisch. Eine statische Speicherzelle ist wie ein *Flipflop* aufgebaut. Der Wert wird eingelesen und bleibt so lange erhalten, bis die Betriebsspannung zusammenbricht.

Eine dynamische Speicherzelle besteht im Prinzip nur aus einem FET (*Feldeffekttransistor*) und einem Kondensator. Wenn der Kondensator geladen ist, beträgt der Wert der Speicherzelle 1, und wenn er entladen ist, beträgt ihr Inhalt 0. Der Vorteil des dynamischen RAM besteht darin, daß sich in dieser Technologie wesentlich höhere Integrationsdichten je Baustein erzielen lassen, als es beim statischen RAM der Fall ist. So ist es

möglich, daß der gesamte dynamische RAM des CPC 464 in nur acht ICs enthalten ist. In jedem dieser ICs sind die Daten zu 64 KB × 1 Bit organisiert, d.h., es lassen sich 65 536 Adressen zu je einem Bit ansprechen. Schaltet man acht dieser ICs parallel, so erhält man 64 KB × 8 Bit bzw. 64 KByte RAM, die Größe des Arbeitsspeichers des CPC 464.

Dem Betriebssystem und dem Anwender stehen jedoch nur 48 KByte zur Verfügung, weil die oberen 16 KByte des Speicherbereichs bereits vom Bildschirmspeicher, der die Daten des Videobildes enthält, belegt werden.

Der Nachteil dynamischer RAMs gegenüber ihren statischen Verwandten besteht darin, daß dynamische RAMs wesentlich komplizierter zu handhaben sind. Weil der Kondensator einer dynamischen Speicherzelle durch *Leckströme* mit der Zeit seine Ladung verliert, muß er ungefähr alle zwei Millisekunden neu geladen werden. Dieser Vorgang wird als Refresh bezeichnet.

Moderne Mikroprozessoren unterstützen das Refreshing dynamischer RAMs, indem sie zu der Zeit, in der Daten- und Adreßbus nicht benutzt werden, einen Refreshzyklus einschieben. Für diesen Zweck hat die Z-80-CPU ein besonderes Refreshregister. Der Wert dieses Registers wird periodisch um eins erhöht und über die unteren 7 Bit des Adreßbusses als Refreshadresse ausgegeben. Der RAM eines Computers enthält gewöhnlich das aktuell laufende oder sich in Bearbeitung befindliche Programm, alphanumerische Daten und sonstige Daten des Betriebssystems.

Der Festwertspeicher (ROM)

Daten, die ständig gebraucht werden und niemals vergessen werden dürfen, befinden sich im Langzeitgedächtnis des Computers, dem Festwertspeicher. Man unterscheidet im allgemeinen zwischen ROM, PROM, EPROM und EEPROM. Diese Speichertypen haben alle eine Eigenschaft gemeinsam: Beim Abschalten der Betriebsspannung werden sie nicht gelöscht, sondern behalten ihren Inhalt.

Der ROM (*read only memory*) ist ein Festwertspeicher, dessen Inhalt schon bei der Herstellung durch eine spezielle Ätzmaske festgelegt wird. Die einzelnen Speicherzellen entsprechen jeweils einer leitenden oder nichtleitenden Verbindung von zwei bestimmten Punkten.

Beim PROM (*programmable read only memory*) handelt es sich um einen Festwertspeicher, dessen Inhalt vom Kunden bestimmt werden kann, indem er aus leitenden Verbindungen (Dioden) durch Anlegen einer

hohen (Sperr-)Spannung nichtleitende Verbindungen erzeugt (die Dioden werden zerstört).

Der EPROM (*erasable programmable read only memory*) kann wie ein PROM durch Anlegen hoher Spannungen an die einzelnen Speicherzellen programmiert werden. Im Gegensatz zum PROM kann er jedoch gelöscht werden. Wenn man die Fläche des Chips, die von einem UV-durchlässigen Quarzglasfenster bedeckt wird, ungefähr eine halbe bis eine Stunde lang direkt mit einer UV-Lampe bestrahlt, verliert er seinen Inhalt.

Der EEPROM (*electrically erasable programmable read only memory*) hat gegenüber dem EPROM den Vorteil, daß er elektrisch löschbar ist und der Benutzer exakt bestimmen kann, welche Speicherzellen gelöscht werden sollen. (Der EPROM kann im Gegensatz zum EEPROM nur *vollständig* gelöscht werden.) Dafür ist der EEPROM aber auch der bei weitem teuerste Festwertspeicher!

In Homecomputern ist in der Regel ein ROM als Festwertspeicher eingebaut, der die *Firmware* des Gerätes enthält. Der ROM des CPC 464 verfügt über 32 KByte. Davon liegt die eine Hälfte des Speichers parallel zu den unteren 16 KByte des RAM und die andere Hälfte parallel zum Bildschirmspeicher.

Die externen Speicher

Alle bisher besprochenen Speichertypen befinden sich direkt im Computergehäuse und werden als *interne* Speicher bezeichnet. Die größten Speicher eines Computers befinden sich jedoch außerhalb des Gerätes und werden deshalb als *externe* Speicher bezeichnet. Dazu gehören sämtliche Arten von Datenträgern, vom Buch bis zur Diskette.

Vielleicht wundern Sie sich über das Buch, aber auch das Buch ist ein Speicher, denn jedes Buch enthält Daten. Ihr Computer kann allerdings mit einem Buch herzlich wenig anfangen. Erst müssen Sie die Daten umsetzen und dem Computer mundgerecht servieren. Dies tun Sie, indem Sie z. B. ein Programmlisting aus diesem Buch in den CPC 464 eingeben. Beschränken wir uns lieber auf die Speicher, die der CPC 464 ohne Ihre Hilfe versteht – die *magnetischen* Datenträger Kassette und Diskette.

Die Kassette ist zur Zeit das am weitesten verbreitete Speichermedium. Der Grund liegt in den geringen Speicherkosten. Eine Kassette kostet nur wenige DM, kann aber bis zu 1 MByte an Daten aufnehmen. Hinzu

kommt, daß auch ein Kassettenrecorder relativ preiswert zu erwerben ist.

Nachteile der Kassette sind jedoch die geringe *Aufzeichnungsdichte* und *Übertragungsgeschwindigkeit*. Während die allerersten Mikrocomputer noch mit trägen 300 Baud (Bit/Sekunde) arbeiteten, sind zwar inzwischen 1200 Baud (fast) Standard, doch auch damit läßt sich selbst der langsamsten Diskettenstation nicht das Wasser reichen. Zusätzlich muß oft lange gespultwerden, bis man die Bandstelle, an der das gewünschte Programm beginnt, erreicht hat. Der CPC 464 mit maximal 2000 Baud und einem zügig spulenden Motor bildet hier eine lobenswerte Ausnahme.

Die Diskette ist 10- bis 100mal so schnell wie eine Kassette. Dieses Speichermedium rotiert in der Diskettenstation mit mehreren hundert Umdrehungen pro Minute, und ein beweglicher Tonkopf kann blitzschnell jede beliebige Position der Diskette ansteuern.

Weiterhin ist eine Diskette im Gegensatz zur Kassette streng *formatiert*. Sie ist in mehrere Spuren aufgeteilt, die sich wiederum aus einzelnen Sektoren zusammensetzen, und ein Sektor enthält 128 Byte (*single density*), oder 256 Byte (*double density*).

Die Diskettenstation des CPC 464 speichert bis zu 180 KByte auf der Seite einer 3-Zoll-Diskette ab. Die Diskette ist dabei in 40 Spuren zu 36 Sektoren mit je 128 Byte aufgeteilt, arbeitet also mit einfacher Schreibdichte.

Der Nachteil der Verwendung von Disketten liegt im relativ hohen Preis der Diskettenstation. Schon eine normale Station mit einem Tonkopf, der nur eine Seite der Diskette nutzen kann (*single sided*), kostet in der Regel mehr als der zugehörige Computer. Eine Station mit zwei Tonköpfen für den direkten Zugriff auf beide Seiten der Diskette (*double sided*) kommt natürlich noch teurer.

Die Diskettenstation des CPC 464 kostet ebenfalls soviel wie das Grundgerät. Dafür sind aber als noble Extras das weltweit verbreitete Betriebssystem CP/M und die immer beliebter werdende Programmiersprache Logo mit im Kaufpreis enthalten. Diese beiden leistungsfähigen Programme werden Sie schnell mit dem Preis der Station versöhnen.

1.4 Der Kassettenrecorder

Der CPC 464 hat den Vorteil, daß der Kassettenrecorder zur Speicherung von Programmen und Daten bereits im Grundgerät enthalten ist. Auf diese Weise ist gewährleistet, daß Computer- und Recorderelektronik optimal einander angepaßt sind und keine Adaptionsprobleme entstehen. Die Mechanik des Recorders hat allerdings ihre Macken. Beim Spulen einer Kassette an Bandanfang oder Bandende quietscht es unter Umständen fürchterlich, und bei rasanter Betätigung der EJECT-Taste kommt Ihnen die eingelegte Kassette im hohen Bogen entgegengeflogen. Eine Federdämpfung wäre in diesem Fall angebracht gewesen. Abgesehen von diesen Eigenheiten, ist die Mechanik des Recorders robust und arbeitet präzise. Auch die Elektronik ist ziemlich zuverlässig, nur bei minderwertigen Kassetten können Lesefehler entstehen.

Die Baud-Raten

Dem Programmierer stehen zwei Datenübertragungsgeschwindigkeiten zur Verfügung: Als erstes die sichere 1000-Baud-Rate und als zweites die schnelle 2000-Baud-Übertragungsrate.

Rein rechnerisch ergeben sich Aufzeichnungszeiten von 8,2 bzw. 4,1 Sekunden pro KByte. Dieser Wert wird in der Praxis allerdings nicht erreicht, weil der CPC 464 Daten immer in einem bestimmten Format zu 2-KByte-Blöcken abspeichert. Format einer 1000-Baud-Aufzeichnung:
3 Sekunden Vorspann, 7 Sekunden Kennung, 17 Sekunden Daten.

Format einer 2000-Baud-Aufzeichnung:
3 Sekunden Vorspann, 3,5 Sekunden Kennung, 8,5 Sekunden Daten.

Dieses Format gilt jeweils für die *vollständige* Aufzeichnung eines 2-KByte-Blocks.

Mit einer Datenübertragungsrate von 1000 Baud benötigt der CPC 464 27 Sekunden für 2 KByte, bei einer Rate von 2000 Baud insgesamt 15 Sekunden für einen kompletten Block. Weil aber tatsächlich nur 17 bzw. 8,5 Sekunden lang die Daten übertragen werden, wird Ihnen die subjektive Übertragungsrate des CPC 464 wesentlich langsamer erscheinen. Bei einer Rate von 1000 Baud werden nur 63 % der Zeit für die reine Datenübertragung genutzt, bei 2000 Baud sogar nur 57 %.

Welche Aufzeichnungsrate ist für Sie am besten? Der vorsichtige Programmierer, der von allen Daten und Programmen Sicherheitskopien

anfertigt, kann getrost mit der schnellen 2000-Baud-Rate arbeiten. Selbst wenn ausnahmsweise ein Lesefehler auftritt, läßt sich noch die Sicherheitskopie verwenden.

Der eilige Benutzer dagegen, der aus Zeitgründen auf eine zusätzliche Kopie verzichten will, sollte ausschließlich die langsamere und sichere 1000-Baud-Rate wählen und nur bestes Kassettenmaterial verwenden. (Achtung: keine Chromdioxidkassetten benutzen!)

1.5 Die Peripherie

Als Peripherie bezeichnet man alle Einheiten eines Computersystems, die nicht direkt zum Grundgerät, bestehend aus Mikroprozessor, Speicher und I/O-Ports, gehören. Beim CPC 464 sind an Peripherie bereits *Tastatur*, *Kassettenrecorder* und *Monitor* im Lieferumfang des Computers enthalten.

Die Tastatur

Die Tastatur des CPC 464 besteht aus 74 einzelnen Tasten, die auch unter Dauerbelastung prellfrei arbeiten und angenehm zu bedienen sind. Sie macht vom Aufbau her einen recht soliden Eindruck und bewährt sich auch bei Dauerbelastung in der Praxis.

Störend wirkt es allerdings, daß es sich um eine *amerikanische* Tastatur handelt. Wesentliche Merkmale sind, daß das Z mit dem Y vertauscht ist und die deutschen Umlaute fehlen.

Dieser Mangel läßt sich jedoch beheben, indem Sie mit dem BASIC-Befehl KEY DEF die Bedeutung der Tasten Z und Y umdefinieren und sich mit dem SYMBOL-Befehl Ihre eigenen Umlaute definieren (siehe Abschnitt 2.4.4).

Der Kassettenrecorder

Der Kassettenrecorder wurde bereits im letzten Abschnitt ausführlich behandelt.

Der Monitor

Die interessanteste mitgelieferte Peripherieeinheit des CPC 464 ist zweifellos der Monitor. Beim Kauf des Grundgerätes kann man zwischen dem *monochromen* (einfarbigen) Grünmonitor und einem Farbmonitor wählen.

Jeder Monitor hat seine Vor- und Nachteile. Während nur mit dem Grünmonitor eine ansprechende Qualität bei der Darstellung von 80 Zeichen je Zeile zu erreichen ist, besticht der Farbmonitor durch seine kräftigen, flimmerfreien Farben. Beiden Monitoren ist gemeinsam, daß sie das Netzteil des CPC 464 enthalten. Folge: Ohne Monitor kein Strom für den Computer!

Das Netzteil der Monitors kann in beiden Fällen einen Ausgangsstrom von 3 A erzeugen. Dies entspricht bei einer gelieferten Spannung von 5 V einer Ausgangsleistung von 15 W. Der CPC 464 benötigt aber nur 10 W für seine Energieversorgung, davon etwa 1 W der Kassettenrecorder im Betrieb. Es bleibt für eigene Zwecke eine *Leistungsreserve* von 5 W, d.h., Sie können bis zu 1 A Strom am Erweiterungsanschluß für eigene Schaltungen entnehmen. Damit ist der CPC 464 einer der wenigen Computer, deren Spannungsversorgung nicht sofort zusammenbricht, wenn man ein paar Leuchtdioden am Expansionsport betreiben will (siehe auch Abschnitt 1.6.1).

Der Modulator

Ebenfalls von Schneider erhältlich ist ein Hf-Modulator mit eingebautem Netzteil. Der Modulator formt das Monitorausgangssignal des CPC 464 so um, daß es von einem Fernsehgerät erkannt werden kann. Auf diese Weise können Sie auch ein Farbfernsehgerät an Ihr Gerät anschließen, wenn Sie nur einen Grünmonitor besitzen.

Was das Netzteil im Modulator zu suchen hat, können Sie sich sicherlich schon denken. Wie wir gesehen haben, erhält der CPC 464 von seinem Monitor den lebensnotwendigen »Saft«, und wenn wir keinen Monitor angeschlossen haben, muß eben der Modulator für den Strom sorgen. Warum ist ein Modulator für den Anschluß eines Fernsehers notwendig? Reicht denn nicht ein normales Antennenkabel?

Ein Monitor besteht, vereinfacht dargestellt, aus einer *Ablenkeinheit* und einer *Bildröhre*. Die Ablenkeinheit generiert aus dem Eingangssignal

das Videobild, und die Bildröhre läßt das erzeugte Bild sichtbar werden. Beim Fernsehgerät kommt prinzipiell nur noch der *Tuner* (Empfänger) dazu, mit dem verschiedene Sender empfangen werden können. Dieser Tuner filtert aus dem Wellensalat, den jede Antenne liefert, eine bestimmte Frequenz heraus und erzeugt durch *Demodulation* der *Trägerwelle* das eigentliche Videosignal, das von der Ablenkeinheit weiterverarbeitet werden kann. Ein Fernseher ist eigentlich nichts anderes als ein Monitor mit eingebautem Empfänger für Fernsehempfang.

Der Tuner des Fernsehgeräts ist daran schuld, daß das für einen Monitor bestimmte Videosignal nicht direkt vom Computer eingespeist werden kann, sondern ein Modulator benötigt wird, der eine Trägerwelle erzeugt (meistens Kanal 36) und dieser Welle das Videosignal aufmoduliert. Der Tuner des Fernsehers trennt darauf wieder Trägerwelle und Nutzsignal durch Demodulation. Dieser technische Umweg und die geringere Auflösung eines Fernsehers gegenüber einem Monitor (5,5 MHz im Vergleich zu 18 MHz) sind der Grund, daß das Fernsehbild ab 50 Zeichen/Zeile unscharf und verschwommen wirkt. Bei ernsthaften Anwendungen (z.B. Textverarbeitung) ist deshalb ein Monitor unerläßlich.

Der Drucker

Neben dem Bildschirm ist der Drucker die wichtigste Ausgabeeinheit eines Computers. Man unterscheidet im allgemeinen zwischen Matrix- und Typenraddruckern. Matrixdrucker setzen jedes zu druckende Zeichen aus einer *Punktmatrix* zusammen. Die Anzahl der Punkte einer Punktmatrix bestimmt in diesem Fall die Qualität des Schriftbildes.

Bei Typenraddruckern hingegen existieren für alle darstellbaren Zeichen kleine *Druckstempel*, die auf einem Rad kreisförmig angeordnet sind. Ein Typenraddrucker entspricht damit annähernd einer elektrischen Schreibmaschine ohne Tastatur.

Wie gewöhnlich hat auch hier jedes System seine Vor- und Nachteile. Vorteile des Matrixdruckers sind hohe Druckgeschwindigkeit, niedriger Preis und die Fähigkeit zur Darstellung von aus Punkten zusammengesetzten Grafiken. Demgegenüber hat der Typenraddrucker den Vorteil eines sehr sauberen Schriftbildes und die Möglichkeit eines schnellen Wechsels der Schriftart durch einen Typenradaustausch.

Die Zukunft gehört jedoch eindeutig dem Matrixdrucker. Ungefähr fünfmal so schnell wie ein Typenraddrucker, aber zum halben Preis, bietet

er die entscheidenden Vorteile. Zusätzlich geben sich moderne Matrixdrucker viel Mühe, um den Ansprüchen eines akzeptablen Schriftbildes gerecht zu werden. Durch Verfeinerung der Punktmatrix und Herabsetzung der Druckgeschwindigkeit zugunsten der Druckqualität ist es inzwischen möglich, eine Schrift zu erzeugen, die von einem Typenraddrucker nur schwer zu unterscheiden ist. Diese Druckqualität wird allgemein als *near-letter-quality* (NLQ) bezeichnet. Die neuesten Matrixdrucker versprechen sogar ein Schriftbild, das von dem eines Typenraddruckers überhaupt nicht mehr zu unterscheiden ist. Sie kosten zur Zeit noch mehrere tausend DM und bezeichnen ihre Druckqualität selbstbewußt als *letter-quality* (LQ).

Der Drucker NLQ 401, der von Schneider für den CPC 464 angeboten wird, verrät bereits durch seinen Namen, daß er über einen NLQ-Modus verfügt. Die Druckgeschwindigkeit mit mageren 50 Zeichen je Sekunde im schnellsten Modus läßt dagegen sehr zu wünschen übrig. Wenn Sie viele Daten auszudrucken haben oder professionell Texte verarbeiten wollen, sind Sie mit einem komfortableren Drucker auf Dauer besser bedient.

Der Plotter

Wenn Ihnen die Grafikmöglichkeiten eines Matrixdruckers nicht ausreichen, dann sollten Sie sich einen Plotter zulegen. Dieses Gerät kann mit einem Stift, der in zwei Richtungen beweglich ist, echte Zeichnungen anfertigen – je nach Größe des Geldbeutels einfarbig oder mehrfarbig und mit einer Schrittauflösung von 1 bis 0,05 mm (letzteres vom menschlichen Auge nicht mehr erfaßbar).

Weitere Peripherie

Inzwischen wird zum CPC 464 auch schon mancherlei Exotisches an Peripherie angeboten. Es soll nur der ADU (*Analog-Digital-Umwandler*) genannt werden, der es ermöglicht, den CPC 464 in ein komplettes Speicheroszilloskop zu verwandeln. Zweifellos eine originelle Idee, aber bei einer oberen Darstellungsfrequenz von 10 KHz für den ernsthaften Anwender ungeeignet.

Findige Bastler werden im Zweifelsfall vor einem Selbstbau von Peripherie für ihren Computer nicht zurückschrecken. Dem Elektronik-

laien bleibt jedoch nur zu hoffen, daß in Zukunft die eine oder andere fertig aufgebaute Hardwareerweiterung für den CPC 464 auf den Markt kommt, die sich für ihn als interessant und nützlich erweisen kann.

1.6 Die Schnittstellen

Eine Schnittstelle soll die Verbindung zwischen Computer und Peripherie herstellen. Der CPC 464 verfügt an seiner Rückseite über mehrere Schnittstellen, die für unterschiedlichste Anwendungen geeignet sind. Die folgenden Abschnitte erläutern *Pinbelegung* und *Funktion* von den Signalen der verschiedenen Schnittstellen und geben an, welche Peripherieeinheiten von Ihnen an welche Schnittstelle in welcher Weise angeschlossen werden können.

1.6.1 Erweiterungsanschluß

Der Erweiterungsanschluß des CPC 464 stellt fast alle Signale des Computers zur Verfügung. Die Pinbelegung dieser Schnittstelle ist zwar im Handbuch des CPC 464 abgedruckt, nicht aber die Bedeutung der einzelnen Signale. Diese können Sie der Tabelle 1.3 entnehmen.

Die einzelnen Signale können in folgende Gruppen aufgeteilt werden:

1. Daten (D0 bis D7)
2. Adressen (A0 bis A15)
3. Steuerbus ($\overline{\text{MREQ}}$, $\overline{\text{M1}}$, $\overline{\text{RFSH}}$, $\overline{\text{IORQ}}$, $\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{HALT}}$, $\overline{\text{INT}}$, $\overline{\text{NMI}}$, $\overline{\text{BUSRQ}}$, $\overline{\text{BUSAK}}$, READY, $\overline{\text{RESET}}$)
4. Speicheraktivierungssignale ($\overline{\text{ROMEN}}$, ROMDIS, $\overline{\text{RAMRD}}$, RAMDIS)
5. Sonstige Signale (SOUND, CURSOR, LIGHTPEN, $\overline{\text{EXP}}$, CLOCK)

Die Signale der ersten drei Gruppen erzeugt der Mikroprozessor des CPC 464. Die Signale für Aktivierung und Desaktivierung von ROM und RAM generiert das Gate Array, und alle Signale der letzten Gruppe – außer CLOCK – werden vom PIO und CRTC erzeugt.

Für das CLOCK-Signal sind der Quarz und das Gate Array, das den Takt durch vier teilt, zuständig.

Normalerweise wird die Diskettenstation mit dem Erweiterungsanschluß verbunden; wenn Sie jedoch alle Ausgänge des Anschlusses puffern

Pin-Nr.	Bezeichnung	Bedeutung der Signale im aktiven Zustand
1	SOUND	Tonausgang
2	GND	Masseanschluß
3	A15	höchstwertiges Bit des Adreßbus gesetzt
4	A14	niederwertiges Bit des Adreßbus gesetzt
5	A13	niederwertiges Bit des Adreßbus gesetzt
6	A12	niederwertiges Bit des Adreßbus gesetzt
7	A11	niederwertiges Bit des Adreßbus gesetzt
8	A10	niederwertiges Bit des Adreßbus gesetzt
9	A9	niederwertiges Bit des Adreßbus gesetzt
10	A8	niederwertiges Bit des Adreßbus gesetzt
11	A7	niederwertiges Bit des Adreßbus gesetzt
12	A6	niederwertiges Bit des Adreßbus gesetzt
13	A5	niederwertiges Bit des Adreßbus gesetzt
14	A4	niederwertiges Bit des Adreßbus gesetzt
15	A3	niederwertiges Bit des Adreßbus gesetzt
16	A2	niederwertiges Bit des Adreßbus gesetzt
17	A1	niederwertiges Bit des Adreßbus gesetzt
18	A0	niederstwertiges Bit des Adreßbus gesetzt
19	D7	höchstwertiges Bit des Datenbus gesetzt
20	D6	niederwertiges Bit des Datenbus gesetzt
21	D5	niederwertiges Bit des Datenbus gesetzt
22	D4	niederwertiges Bit des Datenbus gesetzt
23	D3	niederwertiges Bit des Datenbus gesetzt
24	D2	niederwertiges Bit des Datenbus gesetzt
25	D1	niederwertiges Bit des Datenbus gesetzt
26	D0	niederstwertiges Bit des Datenbus gesetzt
27	+5 V	Versorgungsspannung
28	$\overline{\text{MREQ}}$	Adreßbus spricht Speicher an
29	$\overline{\text{M1}}$	Erster Maschinenzyklus läuft, d.h., CPU holt Befehlscode aus dem Speicher
30	$\overline{\text{RFSH}}$	Untere 7 Bit des Adreßbus enthalten Refresh-adresse für dynamische RAMs
31	$\overline{\text{IORQ}}$	Unteres Byte des Adreßbus spricht I/O-Port an
32	$\overline{\text{RD}}$	Daten werden gelesen
33	$\overline{\text{WR}}$	Daten werden geschrieben
34	$\overline{\text{HALT}}$	CPU wartet bis zur nächsten Unterbrechung
35	$\overline{\text{INT}}$	abschaltbare Unterbrechungsanforderung
36	$\overline{\text{NMI}}$	nicht abschaltbare Unterbrechungsanforderung
37	$\overline{\text{BUSRQ}}$	CPU ist abgeschaltet, periphere Bausteine können auf Speicher und I/O-Ports zugreifen
38	$\overline{\text{BUSAK}}$	Bestätigung der CPU, daß sie abgeschaltet ist

Pin-Nr.	Bezeichnung	Bedeutung der Signale im aktiven Zustand
39	READY	CPU eingeschaltet
40	$\overline{\text{BUS RESET}}$	Initialisierung der CPU (Eingang)
41	$\overline{\text{RESET}}$	Initialisierung der CPU (Ausgang)
42	$\overline{\text{ROMEN}}$	ROM eingeschaltet
43	ROMDIS	ROM ausgeschaltet
44	$\overline{\text{RAMRD}}$	RAM zum Lesen freigegeben
45	RAMDIS	RAM zum Lesen gesperrt
46	CURSOR	Cursor eingeschaltet (Ausgang des CRTC)
47	L.PEN	Lichtgriffel aktiv (Eingang des CRTC)
48	$\overline{\text{EXP}}$	Erweiterung angeschlossen (Eingang)
49	GND	Masseanschluß
50	Φ	Taktausgang

Tabelle 1.3 Pinbelegung des Erweiterungsanschlusses

und an seine Eingänge nur Signale mit *TTL-Pegel* legen, können Sie auch Ihre eigenen Schaltungen anschließen. Wer auf Nummer Sicher gehen will, kann zusätzlich sämtliche Signale über Optokoppler *galvanisch* trennen. So kann auch bei einem Kurzschluß Ihrer Schaltung dem Computer nichts geschehen. Ferner ist es möglich, bis zu 1 A am Erweiterungsanschluß abzugreifen, wenn Sie die +5 V von Pin 27 beziehen und die Masse Ihrer Schaltung an Pin 2 oder 49 legen.

Trotzdem sollten Sie vorsichtig sein und prüfen, ob die Leiterbahnen des CPC 464 diesen Stromfluß auch auf Dauer aushalten, denn praktische Erfahrungswerte liegen zur Zeit noch nicht vor!

1.6.2 Centronics-Schnittstelle

Aufbau und Funktion

Beim Druckeranschluß des CPC 464 handelt es sich um eine abgemagerte Version der Centronics-Schnittstelle. Längst nicht alle Signale, die für diese *Standardschnittstelle* definiert sind, werden vom CPC 464 bereitgestellt. Das wichtigste Signal der Centronics-Schnittstelle, das der CPC 464 nicht zur Verfügung stellt, ist das höchste Datenbit D7. So ist es leider nicht ohne weiteres möglich, mit dem CPC 464 ASCII-Werte von 128 bis 255 auf einem angeschlossenen Drucker auszugeben. Achten Sie deshalb beim Kauf eines Druckers darauf, daß sich *softwaremäßig* das höchste

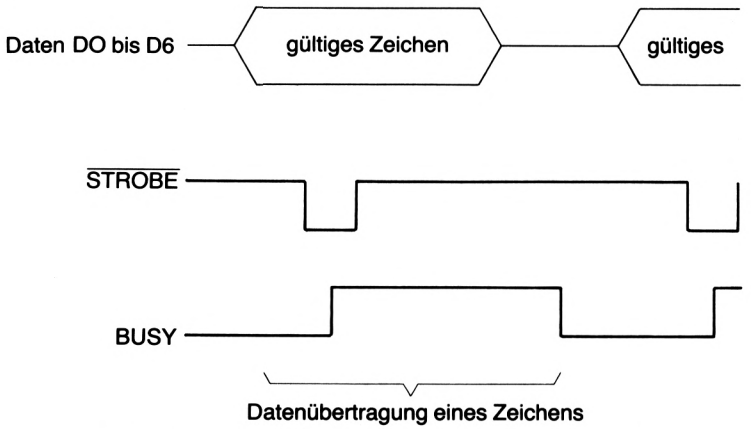


Bild 1.2 Timingdiagramm einer Centronics-Schnittstelle

Datenbit im Drucker setzen und zurücksetzen läßt. Andernfalls können Sonder- und Grafikzeichen, die jeder Drucker im ASCII-Bereich von 128 bis 255 bereitstellt, nicht ausgedruckt werden. Abgesehen von dieser Eigenschaft ist die Schnittstelle des CPC 464 jedoch voll funktionsfähig. Für das Handshaking (Betrieb mit *Quittierungssignalen*) stehen zwar nur die beiden Signale $\overline{\text{STROBE}}$ und BUSY zur Verfügung, doch dies ist für einen ordnungsgemäßen Betrieb der Schnittstelle ausreichend.

Bild 1.2 zeigt den normalen Ablauf der Übertragung eines einzelnen Zeichens vom Computer an den Drucker mit den Quittierungssignalen. Zuerst legt der CPC 464 die Datenbits D0 bis D6 auf den Datenbus. Darauf wird das $\overline{\text{STROBE}}$-Signal gesetzt, das dem Drucker mitteilt, daß er jetzt die Daten übernehmen kann. Der Drucker übernimmt die Daten und zeigt seinerseits mit dem BUSY-Signal an, daß er beschäftigt ist. Sobald dieses Signal inaktiv wird, weiß der Computer, daß der Drucker frei ist und das nächste Zeichen übertragen werden kann.

Das Druckerkabel

Zum Anschluß eines Druckers an den CPC 464 benötigen Sie ein spezielles Druckerkabel, das an der Computerseite einen 34poligen *Direktsteckverbinder* und an der Druckerseite einen normalen *Centronics-Stecker* hat. Leider sind diese Druckerkabel nicht billig. Wenn Sie jedoch ein wenig

Erfahrung haben, können Sie einiges sparen und sich Ihr eigenes Kabel aus einem meterlangen Stück *Flachbandkabel* und den zwei genannten Steckern basteln.

Die Geschichte

Die Centronics-Schnittstelle hat ihren Namen von der gleichlautenden Druckerfirma, die erstmals einen Drucker mit dieser Schnittstelle auf den Markt brachte. Andere Hersteller zogen nach und verwendeten aus praktischen Gründen ebenfalls diese Schnittstelle. Schnell wurde sie zum Standard, und seitdem hat jeder Drucker, der etwas auf sich hält, eine Centronics-Schnittstelle.

So weit, so gut. Der Haken an der Sache ist, daß jeder Druckerhersteller seinen eigenen »Centronics-Standard« entwickelt und produziert. Deswegen ist es heute teilweise nicht ohne weiteres möglich, einen Computer mit eingebauter Centronics-Schnittstelle an einen Drucker mit Centronics-Schnittstelle anzuschließen. Merke: Centronics *ungleich* Centronics!

Achten Sie deshalb beim Kauf von Drucker und Druckerkabel darauf, daß die Pinbelegung beim Anschluß des Gerätes stimmt. Ansonsten funktioniert Ihr Drucker nicht, und unter Umständen sterben einige ICs in ihrem CPC 464 einen schnellen Tod! (Beim Anschluß des Druckers von Schneider entstehen selbstverständlich keine Schnittstellenprobleme.)

1.6.3 Anschluß für den Joystick

Die Pinbelegung des Joystickanschlusses entspricht dem allgemeinen Standard, d.h., es können Joysticks aller Fabrikate an den CPC 464 angeschlossen werden. Dabei ist aber zu beachten, daß Ihr Computer nur über *einen* Joystickanschluß verfügt. Sie können folglich nur einen einzigen Joystick an den CPC 464 anschließen. Es gibt aber viele Spiele, die gleichzeitig zwei Spielern den direkten Wettkampf gegeneinander ermöglichen. Was nun?

Es bleibt Ihnen nichts anderes übrig, als mindestens einen Joystick von Schneider zu erwerben. Dieser Joystick hat nämlich die Besonderheit, daß er über eine *Joystickbuchse* verfügt, an die ein weiterer Joystick angeschlossen werden kann. Wenn Sie jedoch für Ihre Spiele und andere Programme nur einen Joystick benötigen oder an den Schneider-Joystick

einen zweiten Joystick anschließen wollen, können Sie einen ganz normalen, beliebigen Fabrikats verwenden. Auf diese Weise ersparen Sie sich wenigstens den Kauf eines Joysticks, falls Sie noch ein Telespiel mit Joysticks oder Joysticks Ihres alten Computers übrig haben.

Die Pins COMMON und COM2

Vielleicht fragen Sie sich, wie der Computer über *einen* Joystickanschluß *zwei* verschiedene Joysticks erkennen kann. Die Lösung ist einfach: Der Computer gibt über den Joystickanschluß die beiden Signale COMMON und COM2 aus. Wenn das Signal COMMON aktiv ist, wird der erste Joystick abgefragt, und wenn das Signal COM2 aktiviert wird, ist der zweite Joystick angesprochen. Die beiden Signale COMMON und COM2 dienen folglich zur Adressierung der beiden Joysticks.

Theoretisch könnten Sie an den CPC 464 beliebig viele Joysticks von Schneider anschließen, weil aber neben den zwei vorhandenen COM-Pins kein weiteres COM-Pin existiert, wäre es dem Computer nicht möglich, zusätzlich angeschlossene Joysticks abzufragen.

1.6.4 Weitere Schnittstellen

Als weitere Schnittstellen besitzt der CPC 464 einen Videoausgang zum Anschluß des Monitors und einen Stereo-Tonausgang zur Verbindung des Computers mit einer Stereoanlage.

Die Pinbelegung des Videoausgangs ist im Bedienungshandbuch des CPC 464 angegeben und für uns nicht weiter interessant. Der Stereo-Tonausgang des CPC 464 ermöglicht es Ihnen, statt des leisen, dünnen Tons in Mono aus dem Lautsprecher Ihres Computers, den vollen und satten Stereoton Ihrer HiFi-Anlage zu genießen. Dazu müssen Sie lediglich an den Tonausgang ein normales Tonkabel mit einem 3,5-mm-Klinkenstecker anschließen und die andere Seite des Kabels mit dem MONITOR- oder AUX-Eingang Ihrer Stereoanlage verbinden.

2
Die Firmware des CPC 464

Die Firmware ist ein Mittelding zwischen *Hard-* und *Software*. Sie wird wie die Hardware vom Hersteller mit dem Computer geliefert, besteht aber aus einer Sammlung von Programmen (Software), die den Computer verwalten.

Als Firmware bezeichnet man die gesamte Software eines Computers, die bereits beim Kauf des Grundgeräts fest eingebaut ist oder mitgeliefert wird.

Im Fall des CPC 464 handelt es sich um den 32 KByte großen ROM, der das komplette *Betriebssystem* des Computers enthält. In diesem ROM befinden sich ein *BASIC-Interpreter*, ein *Tastatur-, bildschirm-, Ton-* und *Kassettenrecordermanager* sowie ein *zeilen-* und *bildschirmorientierter Editor*, um nur die wichtigsten Systemprogramme zu nennen.

Dieses Kapitel wird sich eingehend mit dem Verständnis und der Anwendung der Firmware des CPC 464 beschäftigen.

2.1 Die Tastaturfunktionen

Die Tastatur des CPC 464 besteht aus einer *amerikanischen* Schreibmaschinentastatur und einem abgegrenzten Ziffernblock. Alle Funktionstasten sind farblich abgehoben, mit Ausnahme der CLR-Taste, um einer Verwechslung mit der DEL-Taste vorzubeugen.

Die Funktion der einzelnen Tasten sind im Handbuch des CPC 464 ausführlich beschrieben und sollen an dieser Stelle nur kurz behandelt und ergänzt werden:

Einmaliges Drücken von ESC während des Listens oder Ausführens eines Programms erzeugt einen Halt, zweimaliges Drücken unterbricht die Tätigkeit.

Die Taste DEL löscht des Zeichen links des Cursors, CLR dagegen die momentane Cursorposition.

Mit COPY wird das Zeichen der Kopiercursorposition auf die Position des regulären Cursors kopiert.

Das Betätigen der Taste CAPSLOCK läßt alle folgenden Buchstaben als Großbuchstaben erscheinen. Erneutes Drücken von CAPSLOCK macht diese Funktion rückgängig.

CTRL und CAPSLOCK entsprechen in ihrer Funktion dem *SHIFT-LOCK* der Schreibmaschine. Aber Vorsicht, in diesem Modus funktioniert das Editieren mit dem Kopiercursor nicht mehr, weil ausschließlich der Kopiercursor mit den Cursortasten bewegt werden kann!

Mit CTRL und einer anderen gedrückten Taste lassen sich die ASCII-Zeichen 1 bis 31 erzeugen, CTRL M entspricht der Taste ENTER, und CTRL P erzeugt einen Piepton.

Die Taste TAB hat wider Erwarten keine Tabulatorfunktion, sondern erzeugt einen nach rechts deutenden Pfeil.

Die Taste ENTER bedarf gewiß keiner näheren Erläuterung, denn dies ist mit Sicherheit die wichtigste und am meisten benutzte Funktion eines Computers.

Die Bedeutung von SHIFT CTRL ESC ist von größter Wichtigkeit für Sie. Das gleichzeitige Drücken von SHIFT, CTRL und zuletzt ESC erzeugt einen softwaremäßigen RESET des CPC 464. Alle Daten werden gelöscht, und der Computer wird neu initialisiert.

Zuletzt ein paar Worte zu dem Zeichen | (SHIFT @):

Wenn Sie dieses Zeichen in eine REM-Zeile einsetzen, geschehen seltsame Dinge. Geben Sie folgende Zeile ein, und lassen Sie diese Zeile vom CPC 464 listen:

```
10 REM |!
```

Sie sehen, das Ausrufezeichen scheint verschwunden zu sein. Es befindet sich zwar noch im Programmtext, das Zeichen | verhindert jedoch das Listen des unmittelbar folgenden Zeichens.

Geben Sie zusätzlich folgende Zeile ein, und starten Sie das Programm:

```
20 PRINT "HALLO!"
```

Wieder werden Sie überrascht sein, denn der CPC 464 druckt gar nichts aus. Die Programmzeile 20 läßt sich wohl listen, aber die Ausführung dieser Zeile wird durch das | in der vorhergehenden REM-Zeile

verhindert. Durch Ausprobieren können Sie weitere Eigenheiten dieses Zeichens feststellen. Sicher ist, daß Sie für einen ordnungsgemäßen Programmablauf das | in keiner REM-Zeile verwenden dürfen.

Wenn Sie Umlaute auf dem CPC 464 definiert haben, wird aus dem | das kleine *ö*. Weil der ASCII-Wert jedoch unverändert bleibt, gilt auch für das *ö*, daß es in keine REM-Zeile eingesetzt werden darf. Schauen Sie sich dazu die REM-Zeilen des Abfrageprogramms und der Adressenverwaltung an!

2.2 Der Editor

Der Editor des CPC 464 stellt Ihnen mehrere Möglichkeiten zur Korrektur eines Schreibfehlers oder Verbesserung einer BASIC-Programmzeile zur Verfügung. Es gibt insgesamt vier Verfahrensweisen, einen Fehler zu beheben:

- Eine falsche Eingabe kann durch Verwendung der Cursortasten und der Tasten CLR und DEL behoben werden.
- Sie können eine fehlerhafte Programmzeile korrigieren, indem Sie die Zeile neu eingeben.
- Mit dem Befehl EDIT kann in eine zu verbessernde Zeile gesprungen und der Fehler durch Löschen oder Einfügen von Zeichen behoben werden.
- Der Kopiercursor ermöglicht Ihnen das Kopieren von Zeilen und Zeilenausschnitten an jede beliebige Stelle des Bildschirms und gleichzeitig in den Speicher des CPC 464.

Die schnellste Editiermethode ist für jeden Fehler unterschiedlich.

Kleine Eingabefehler werden sofort nach der erstgenannten Methode behoben. Ganz kurze Zeilen geben Sie am besten neu ein. Programmzeilen, deren Fehler ziemlich weit am Anfang der Zeile liegen, lassen sich am schnellsten mit EDIT korrigieren, denn in diesem Fall ist es nicht notwendig, die ganze Zeile mit dem Cursor zu durchfahren.

Ebenso werden Programmzeilen mit wenigen Fehlern am besten durch EDIT behoben, weil Sie in diesem Fall kreuz und quer mit dem Cursor durch die Zeile huschen können.

Für das Zusammenfügen von Programmzeilen eignet sich ausschließlich der Kopiercursor, denn im EDIT-Modus kann die aktuell editierte Zeile nicht verlassen werden.

Gleichfalls ist bei größeren Fehlern an mehreren Stellen einer Zeile diese Methode am günstigsten, weil sie übersichtlicher ist. Nur die Zeichen, die Sie mit COPY kopieren und neu über die Tastatur eingeben, werden auf dem Bildschirm angezeigt und als Inhalt der neuen Programmzeile abgespeichert.

Die Theorie hört sich ziemlich trocken an, doch in der Praxis werden Sie mit der Zeit merken, welche Editiermöglichkeit in welcher Situation für Sie am schnellsten ist. Jede Methode hat ihre Vor- und Nachteile. Wenn Sie jedoch die jeweils beste Möglichkeit des Editierens anwenden, wird das Korrigieren von Fehlern mit dem CPC 464 komfortabler als bei vielen anderen Computern.

2.3 Der BASIC-Interpreter

Der BASIC-Interpreter ist das größte und zugleich wichtigste *Systemprogramm* der *Firmware*. Dieses Programm interpretiert die einzelnen BASIC-Befehle, wandelt sie in einen computergerechten Code um und führt sie (soweit möglich) aus. Ein BASIC-Interpreter ist folglich ein *Übersetzungsprogramm*, das die höhere Programmiersprache BASIC in einen für den Computer verständlichen Zahlencode, bestehend aus Nullen und Einsen, umformt.

Je mehr Befehle ein BASIC-Interpreter verstehen und verarbeiten kann, desto leistungsfähiger ist er für den Programmierer.

Der BASIC-Interpreter des CPC 464 beherrscht stolze 170 Befehle und verfügt damit über einen der mächtigsten serienmäßigen BASIC-Dialekte für Homecomputer.

2.3.1 Variable

Variablennamen

Die Namen der Variablen des CPC 464 dürfen bis zu 40 Zeichen lang sein und aus alphanumerischen Zeichen bestehen. Einzige Bedingungen sind, daß der Variablenname mit einem Buchstaben beginnt und keinem BASIC-Befehlswort entspricht (z.B. END). Ein BASIC-Befehlswort als Teil eines Variablennamens (z.B. ende) ist dagegen erlaubt. Das ist auch der Grund, warum Sie zwischen Befehl und Variable (oder Zahl) immer

ein Leerzeichen setzen müssen. Andernfalls würde der BASIC-Interpreter den gesamten Ausdruck als Variable interpretieren.

Wenn Sie deutsche Umlaute definiert haben, dürfen diese Umlaute *nicht* im Variablennamen erscheinen, weil sie immer noch die ASCII-Werte der ursprünglichen Zeichen besitzen und diese Zeichen vom Interpreter nicht als Buchstaben anerkannt werden.

Variablentypen

Der CPC 464 verfügt über drei verschiedene Variablentypen:

Die Realvariable kann einen Wert im Bereich von $-1{,}70141\,E+38$ bis $+1{,}70141\,E+38$ zugewiesen bekommen. Läßt sie sich ohne Exponentialschreibweise darstellen, werden neun Ziffern angezeigt, in Exponentialschreibweise dagegen nur sechs Ziffern. Eine Realvariable kann durch den Befehl DEFREAL oder durch ein Ausrufezeichen direkt hinter dem Variablennamen definiert werden.

Der Integervariablen kann ein ganzzahliger Wert im Bereich von -32768 bis $+32767$ zugewiesen werden. Dieser Zahlenbereich wird mit einer 16-Bit- bzw. 2-Byte-Darstellung abgedeckt.

Eine Integervariable muß durch den Befehl DEFINT oder durch ein Prozentzeichen direkt hinter dem Variablennamen definiert werden.

Die Stringvariable enthält einen Text von bis zu 255 Zeichen Länge, der aus allen ASCII-Zeichen von CHR$(32) bis CHR$(255) zusammengesetzt sein darf. Mit dem Befehl DEFSTR oder einem Dollarzeichen hinter dem Variablennamen wird dieser Variablentyp definiert. Wenn der Typ einer Variablen nicht ausdrücklich durch einen DEF-Befehl oder durch ein Sonderzeichen hinter dem Variablennamen definiert wird, nimmt der Interpreter automatisch REAL als Typ an.

Variablenspeicherung

Wenn einer Variablen ein Wert zugewiesen oder ihr Inhalt verändert wird, muß dieser neue Wert vom BASIC-Interpreter abgespeichert werden. Für diesen Zweck existiert direkt hinter dem Programmtext im Arbeitsspeicher des CPC 464 eine Variablentabelle.

Während der Programmerstellung verändert sich die Position des Programmtextendes durch Einfügen und Löschen von Befehlen ständig. Bei vielen Computern wird dadurch die Variablentabelle zerstört und alle

Werte gelöscht. Der CPC 464 kümmert sich jedoch auch beim Verändern eines Programms um die Variablen, indem er die Tabelle in seinem Speicher entsprechend mitverschiebt.

Wenn Sie Ihr Programm mit RUN starten, werden automatisch alle Variablen gelöscht. Mit einem direkten GOTO-Befehl kann dies verhindert werden. Alle Variablenwerte bleiben erhalten, und das Programm beginnt in der angesprungenen Zeile. Steht in dieser Zeile allerdings ein CLEAR-Befehl, werden die Daten doch gelöscht!

2.3.2 BASIC-Befehle

CALL

Mit dem Befehl CALL können Sie eigene RAM-Routinen oder ROM-Routinen der Firmware aufrufen. Wenn Sie diesen Befehl verwenden und Parameter übergeben, müssen alle Werte im Bereich von −32768 bis 65535 liegen.

CHR$

Der Befehl CHR$(1)+CHR$(*n*) gibt das Zeichen mit dem ASCII-Wert *n* aus. Auf diese Art ist es möglich, die ASCII-Zeichen 1 bis 31 zu erzeugen.

Auf die Eingabe von CHR$(7) antwortet Ihr Computer mit einem Piepton. Mit CHR$(22)+CHR$(1) schalten Sie den Transparentmodus ein, mit CHR$(22)+CHR$(0) können Sie ihn wieder ausschalten.

CHR$(24) tauscht die Papierfarbe mit der Schreibstiftfarbe aus. Auf diese Weise können Sie als Besitzer eines Grünmonitors das Videobild invertieren.

DEC$ *)

Der Befehl DEC$ ist im Computer als BASIC-Token implementiert. Dies sehen Sie daran, daß dec$ beim Listen automatisch – wie alle anderen BASIC-Befehle – in die Großschreibweise DEC$ umgewandelt wird. Es ist allerdings nicht möglich, diesen Befehl auszuführen, weil bei jedem Versuch eine Fehlermeldung erscheint. Das ist wiederum halb so schlimm, denn äquivalent zu den Befehlen BIN$ und HEX$ müßte der

Befehl DEC$ eine Zahl in das Dezimalsystem umwandeln. Weil binäre und hexadezimale Zahlen jedoch automatisch beim Ausdruck durch ein PRINT-Kommando in Dezimalzahlen umgewandelt werden, ist diese Funktion überflüssig.

ENT

Siehe Abschnitt 2.5.4

ENV

Siehe Abschnitt 2.5.3

INKEY$

Mit dem Befehl INKEY$ ist eine Dateneingabe ohne Programmunterbrechung möglich.

Manchmal ist es nützlich, mit INKEY$ den Befehl INPUT zu simulieren. So kann der Computer z. B. parallel zu einer Dateneingabe andere Operationen ausführen, indem er nur gelegentlich die Tastatur abfragt.

Eine weitere Möglichkeit bietet sich, wenn Sie Ihre Programme mit CALL 47944 gegen eine Unterbrechung schützen wollen. Jeder INPUT-Befehl würde diese Funktion aufheben. Die Verwendung von INKEY$ hebt dagegen den BREAK-Schutz nicht auf.

Eine einfache Simulation in INPUT läßt sich mit folgender Befehlsfolge erreichen:

```
10 text$=""
20 FOR lauf=1 TO n
30 a$=INKEY$:IF a$="" THEN 30
40 text$=text$+a$
50 NEXT lauf
```

Dieses kleine Programm erlaubt Ihnen die Eingabe eines Textes, bestehend aus *n* Zeichen. Allerdings sind die Editiermöglichkeiten der Tasten CLR und DEL außer Funktion gesetzt.

Merge

Mit dem Befehl MERGE ist es möglich, mehrere einzelne BASIC-Programme zu einem großen Programm zusammenzufügen. Einzige Bedingung ist, daß sich die Zeilennummerbereiche der Programme nicht überschneiden, weil sonst die neuesten Programmzeilen jeweils die älteren Zeilen überschreiben würden. Wenn Sie mit dem Befehl RENUM die Zeilennummern der Programme vor dem Zusammenfügen umnumerieren, kann eigentlich nichts schiefgehen.

*MOD und *

Die beiden Befehle MOD und \ sind im Handbuch des CPC 464 nicht aufgeführt, funktionieren aber trotzdem einwandfrei auf dem Gerät. Wenn Sie statt des Verknüpfungszeichens / das Zeichen \ bei einer Division verwenden, erhalten Sie das *ganzzahlige Ergebnis* der Division. An gleicher Stelle der Befehl MOD eingesetzt, ergibt den *ganzzahligen Rest* dieser Division.

OPENIN und OPENOUT

Wenn Sie eine Datei mit OPENOUT *name* unter dem Namen *name* abspeichern, sollte man erwarten, daß sie sich mit OPENIN *name* wieder einlesen läßt. Doch weit gefehlt! Dateinamen können nur aus Großbuchstaben bestehen, folglich wird die Datei nicht unter *name*, sondern unter *NAME* abgelegt. Wenn Sie jetzt diese Datei unter dem Namen *name* einlesen wollen, fängt der Computer an zu spinnen, weil *name* und *NAME* noch lange nicht dasselbe für ihn sind.

Wenn Ihre Dateien keine festen Namen haben, sondern eine Variable den Namen angibt, können noch mehr Komplikationen entstehen.

Wenn Sie mit OPENOUT *name*$ eine Datei abspeichern und die Variable *name*$ Kleinbuchstaben enthält, kann diese Datei nur mit der Befehlsfolge OPENIN UPPER$(*name*$) wieder eingelesen werden. Sollte die Länge des Textes der Variablen *name*$ schwanken, muß vor der Textzuweisung INPUT *name*$ der Befehl *name*$ = STRING$(16,' ') erfolgen. Dieser Befehl weist der Variablen einen Text bestehend aus 16 Leerzeichen zu. Mit Recht werden Sie sich fragen, was das nun wieder soll. Wenn Sie mit OPENOUT *name*$ eine Datei abspeichern wollen und die

Länge des Textes der Variablen *name*$ von Mal zu Mal länger wird, erhält die abzulegende Datei mit Sicherheit nicht den Namen *name*$. Dieser Fehler der Firmware kann behoben werden, indem man der Variablen vor ihrer eigentlichen Textzuweisung einen Text der maximalen Länge (bei Dateinamen 16 Stellen) zuweist.

Alles klar? Wenn nicht, dann schauen Sie sich mal das Ein- und Auslesen von Daten im Abfrageprogramm von Kapitel 3 an.

RANDOMIZE

Zu dem Befehl RANDOMIZE muß eine Zahl eingegeben werden, die den Anfang einer Zufallszahlenkette darstellt. Wenn Sie RANDOMIZE TIME eingeben, wird garantiert bei jedem Programmlauf eine andere Zufallszahlenkette erzeugt, weil die Variable TIME bei jedem Durchlauf des RANDOMIZE-Befehls einen anderen Wert enthält.

RENUM

Der Befehl RENUM ist besonders komfortabel, denn mit RENUM können Sie die Zeilen Ihres Programms nach Belieben umnumerieren. Allerdings ist es nicht möglich, neben dem Startpunkt auch einen Endpunkt für die Numerierung festzusetzen. Sie müssen Ihr Programm immer bis zum Ende durchnumerieren lassen. Wenn Sie jedoch die Programmblöcke in ihrer Reihenfolge im Programm neu numerieren, kommen Sie auch zum Ziel, denn ein RENUM-Befehl hebt logischerweise ab seinem Startpunkt die Numerierung des vorherigen RENUM auf.

Auf diese Weise können Sie eine Einteilung, wie sie in Abschnitt 2.3.3 vorgeschlagen wird, realisieren.

RND

Der Befehl RND funktioniert nicht so, wie man es von den gebräuchlichen BASIC-Dialekten gewohnt ist. Um eine Zufallszahl von *minimum* bis *maximum* zu erzeugen, müssen Sie den Ausdruck INT(RND∗(*maximum* − *minimum* + 1) + *minimum*) verwenden.

SOUND

Siehe Abschnitt 2.5.2.

SYMBOL

Der Umgang mit dem Befehl SYMBOL wird im Handbuch des CPC 464 etwas stiefmütterlich behandelt.

Wenn Sie ein Zeichen neu definieren wollen, müssen Sie dem Computer zuerst durch SYMBOL AFTER *n* mitteilen, daß die Punktmatrizen für ASCII-Zeichen ab *n* in den RAM kopiert werden sollen. Erst jetzt können Sie mit dem Befehl SYMBOL alle ASCII-Werte von *n* bis 255 ändern. Wenn *n* jedoch größer als 239 ist, kann der Befehl SYMBOL AFTER weggelassen werden, weil die Punktmatrizen der ASCII-Zeichen 240 bis 255 standardmäßig im RAM liegen und deswegen ein Kopieren überflüssig ist.

Der erste Parameter des Befehls SYMBOL gibt den ASCII-Wert des neu zu definierenden Zeichens an. Die folgenden acht Parameter sind die Bitmuster für das neue Symbol. Der erste Wert gibt durch seine gesetzten Bits an, welche Punkte in der obersten Zeile des neuen Zeichens gesetzt werden sollen, der letzte Wert vermittelt das Muster für die unterste Zeile.

Ein Beispiel: Der Befehl SYMBOL 240,255,129,129,129,129,129,129,255 definiert für den ASCII-Wert 240 ein Rechteck. Der binäre Wert von 255 beträgt 11111111, d.h., erste und achte Zeile des Zeichens 240 werden durchgehend gesetzt und bilden obere wie untere Seite des Rechtecks. Der binäre Wert von 129 beträgt 10000001. Folglich werden in allen Zeilen zwischen der ersten und achten Zeile nur der erste und der letzte Punkt gesetzt. Als Ganzes betrachtet ergeben Zeile zwei bis sieben die seitlichen Begrenzungen des Rechtecks. Alle Zeilen zusammengenommen ergeben das komplette Rechteck. Übung macht – gerade bei dieser Prozedur – den Meister.

TRON und TROFF

Die Befehle TRON und TROFF eignen sich hervorragend zur Suche von Fehlern im Programm. Wenn Sie mit TRON den Tracemodus einschalten, druckt der Computer alle Zeilennummern der Programmzeilen aus, die er abarbeitet. So können Sie sehen, ob und wie oft das Programm welche Zeile durchläuft. Mit TROFF können Sie den Tracemodus abschalten.

ZONE

Mit dem Befehl ZONE können Sie die Breite einer Druckzone bestimmen. Diese Druckzonen werden angesprochen, wenn Sie mit einem PRINT-Kommando durch Komma getrennte Daten ausgeben. Die Daten vor dem ersten Komma werden in die erste Druckzone ausgegeben, die Daten nach dem ersten Komma in die zweite Druckzone usw.

Wenn eine Druckzone breiter als der Rest des Bildschirms ist, wird sie automatisch an den Anfang der nächsten Zeile verlegt. So werden z.B. bei MODE 1 und ZONE 41 alle Daten untereinander ausgegeben, weil nur eine einzige Zone vollständig in eine Zeile paßt.

Für das Erstellen von Tabellen ist der Befehl ZONE besonders geeignet.

Befehle zum Runden

Zum Runden von Zahlen stehen Ihnen die vier Befehle CINT, FIX, INT und ROUND zur Verfügung.

CINT rundet Zahlen im Integerbereich (−32768 bis 32767) auf, wenn der Nachkommateil der Zahl mindestens 0,5 beträgt. Achtung, bei negativen Zahlen wird betragsmäßig gerundet, d.h., die Zahlen werden nicht auf −, sondern abgerundet!

Der Befehl FIX gibt lediglich den Vorkommateil einer beliebigen Zahl zurück, und INT gibt den nächstkleineren ganzzahligen Wert einer Zahl aus, es wird also immer abgerundet.

Mit ROUND können Zahlen auf eine bestimmte Anzahl von Nachkommastellen gerundet werden.

Je nach gewünschter Rundungsart müssen Sie sich für einen der genannten Befehle entscheiden.

Die Funktion von ROUND erfüllt auch die Befehlsfolge INT(*zahl*∗1E*n* +0.5)/1E*n*. Sie rundet die Variable *zahl* auf *n* Stellen hinter dem Komma, z.B. ergibt INT(123.456∗100+0.5)/100 als Ergebnis die Zahl 123, 46.

Die Ein-/Ausgabeeinheiten

Der CPC 464 verfügt über die zehn Ein-/Ausgabeeinheiten #0 bis #9.

Die Einheiten #0 bis #7 sind Bildschirmfenster. Das PRINT#*n*-Kommando gibt Daten auf das Bildschirmfenster *n* aus, und der INPUT#*n*-Befehl fordert eine Dateneingabe in Fenster *n*.

#8 spricht über die Centronics-Schnittstelle den Drucker an. Mit dieser Einheit können nur Daten ausgegeben werden. Wenn kein Drucker an den CPC 464 angeschlossen ist, wartet der Computer umsonst auf eine Meldung des Druckers und muß mit zweimaligem Drücken der ESC-Taste unterbrochen werden.

Die Einheit #9 repräsentiert den Kassettenrecorder. Um Daten ein- oder ausgeben zu können, müssen Sie zuerst eine Ein- oder Ausgabedatei öffnen, die Daten mit INPUT#9 bzw. PRINT#9 ein- oder auslesen und anschließend die Datei wieder schließen.

Wenn Sie bei einer Dateneingabe oder -ausgabe keine Einheit angeben, wird vom CPC 464 automatisch #0 benutzt.

2.3.3 Strukturiert programmiert

Mittlerweile dürfte es sich bis zum letzten BASIC-Programmierer herumgesprochen haben, daß der sogenannte Spaghetticode-Programmierstil verpönt ist, strukturiertes Programmieren dagegen im Trend liegt. Der CPC 464 besitzt mehrere Eigenschaften, die das strukturierte Programmieren unterstützen.

Die erlaubte Länge der Variablennamen ermöglicht es dem Programmierer, sinnvolle Namen für die veränderlichen Größen seines Programms zu verwenden. Mit dem Befehl REM oder dem Zeichen ' können Sie Kommentarzeilen in Ihr Programm zur Dokumentation der Funktion einzelner Abschnitte einfügen, damit Sie sich auch nach mehreren Wochen noch in dem Programm zurechtfinden.

Übersichtliche Programme erhalten Sie, wenn alle Programmteile, die mehr als zweimal im Programm benötigt werden, als Unterprogramme definiert werden. Die übrigen Programmteile sollten in einzelne Programmblöcke mit genau einem Ein- und einem Ausgang (keine wilden GOTO-Sprünge!) zerlegt und dokumentiert werden.

In der Praxis hat es sich als zweckmäßig erwiesen, gewissen Zeilennummerbereichen bestimmte Funktionen zuzuordnen. Ein Beispiel:

1	bis 9	REM Programmierer, Copyright usw.
10	bis 99	Definition von Variablen, Farben usw.
100	bis 299	Programmü
300	bis 999	Programmblöcke (Hauptroutinen)

1000 bis 9999	Unterprogramme (Subroutinen)
10000 bis 59999	verschachtelte Unterprogramme
60000 bis 65535	Programmdaten

Dieses Schema muß natürlich nicht sklavisch befolgt werden, sondern soll lediglich als Anregung für eigene Einteilungen dienen.

2.3.4 Fehlersuche im Programm

Sobald die Länge der von Ihnen erstellten Programme über einige Zeilen hinausgeht, werden sich zwangsläufig Fehler einschleichen. Bei Fehlern in Programmen kann man grob zwischen Schreibfehlern und Denkfehlern unterscheiden.

Schreibfehler zu finden ist in der Regel nicht allzu schwer, weil der Computer meistens eine Fehleranzeige mit Angabe der falschen Programmzeile ausgibt.

Das Auffinden von Denkfehlern ist dagegen weitaus schwieriger. Um die Fehlersuche zu vereinfachen, können Sie wichtige Variablenwerte an kritischen Stellen durch eingeschobene PRINT-Kommandos ausdrucken lassen. Hilft diese Maßnahme nicht weiter, müssen Sie durch künstlich erzeugte Wertzuweisungen heikle Daten, die das Programm nicht richtig verarbeiten könnte, simulieren.

Mit RUN *anfang* (löscht alle Variablen) oder GOTO *anfang* (löscht keine Variablen) und *ende* END können Sie den Fehler im Programm zwischen Zeile *anfang* und Zeile *ende* einkreisen. Je kleiner dieser Bereich wird, desto einfacher ist es, den Fehler zu lokalisieren.

Schließlich existieren die Befehle TRON und TROFF, die Ihnen zeigen können, ob das Programm in der Weise abgearbeitet wird, die Sie erwarten.

Wenn Sie auch mit dieser Methode den Fehler nicht finden, sollten Sie das Programm abspeichern und ein paar Tage später erneut auf Fehlersuche gehen. Oft findet sich der Fehler dann schnell, weil man unbelastet ist und sich nicht zu tief in das Problem verbohrt hat. Eine weitere Möglichkeit bietet sich, wenn Sie einen Freund haben, der sich in BASIC auskennt. Schon die Erläuterung der Problemstellung kann den Fehler aufdecken, spätestens jedoch ein noch gänzlich unvoreingenommener Programmierer.

2.4 Die Grafik

Die Grafik des CPC 464 ist recht leistungsfähig. Aus 27 verfügbaren Farben können bis zu 16 Farben gleichzeitig dargestellt werden, und wenn Sie sich mit zwei Farben begnügen, können 128000 Bildpunkte angesprochen werden. Der CPC 464 verfügt über drei Bildschirmmodi, die jeweils unterschiedlich viele Farben und Bildpunkte bereitstellen.

Der Modus 0 wird als *Vielfarbenmodus* bezeichnet. Es können zwar nur 20 Zeichen je Zeile und 160 × 200 Bildpunkte dargestellt werden, dafür sind aber 16 verschiedene Farben gleichzeitig möglich. Modus 1 ist der *Standardmodus*. 40 Zeichen je Zeile und 320 × 200 Bildpunkte werden angezeigt, und maximal vier verschiedene Farben können verwendet werden. Der *hochauflösende Modus* des CPC 464 ist der Modus 2. 80 Zeichen je Zeile und 640 × 200 Bildpunkte können dargestellt werden, allerdings nur zweifarbig.

Obwohl es mehrere Auflösungsmöglichkeiten gibt, muß beim Ansprechen der Punkte mit BASIC-Befehlen kein Unterschied zwischen den einzelnen Modi gemacht werden.

Besonders komfortabel ist die Tatsache, daß bei der Angabe von Koordinaten, die außerhalb des Bildschirms liegen, keine Fehlermeldung erfolgt, sondern lediglich kein Punkt gesetzt wird. Dies ist bei der Ausgabe von Grafiken, die am Rand des Bildschirms liegen, sehr nützlich.

2.4.1 Auflösung

Die hohe Grafikauflösung des CPC 464 macht es möglich, hervorragende Funktionsgraphen und dreidimensionale Darstellungen zu realisieren.

Bei dem Abbilden von mathematischen Funktionen ist vor allem der Befehl ORIGIN sehr nützlich. Mit diesem Befehl kann der Nullpunkt des Grafikkoordinatensystem beliebig gesetzt werden. Wenn Sie diesen Nullpunkt auf den Koordinatennullpunkt des Funktionsgraphen setzen, müssen Sie keine relativen Verschiebungen der Funktionspunkte mit einberechnen, weil auch negative Werte dargestellt werden können.

Bildpunkte können mit dem Befehl PLOT gesetzt und Linien mit dem Befehl DRAW gezogen werden. Nur ein CIRCLE-Befehl zum Zeichnen von Kreisen und Ellipsen fehlt. In Abschnitt 2.4.5 wird gezeigt, wie Sie sich behelfen können.

Manchmal ist es notwendig, Schrift und Grafik gleichzeitig auf den Bildschirm zu bringen. In diesem Fall gibt es die Möglichkeit, daß der

Text die Grafik überschreibt oder Text und Grafik sich überschneiden. Letzteres ist der sogenannte *Transparentmodus* des CPC 464, der mit CHR$(22)+CHR$(1) eingeschaltet und mit CHR$(22)+CHR$(0) ausgeschaltet werden kann.

Vielleicht haben Sie sich schon gefragt, warum über die Software des CPC 464 zwar 400 senkrechte Bildpunkte angesprochen werden können, die Hardware des Computers aber nur 200 Punkte je Spalte bereitstellt. Zur Erklärung muß etwas weiter ausgeholt werden:

Im Modus 0 kann der CPC 464 insgesamt 32000 Bildpunkte darstellen. Jeder Bildpunkt kann 16 Farben annehmen und benötigt daher im Speicher 4 Bit, weil sich mit 4 Bit genau 16 Zustände darstellen lassen. Wenn nun von 32000 Bildpunkten jeder Punkt genau 4 Bit belegt, dann werden insgesamt 128000 Bit für den Bildschirmspeicher gebraucht.

Im Modus 1 kann der CPC 464 insgesamt 64000 Bildpunkte darstellen. Die Bildpunkte können in diesem Fall vier Farben annehmen und belegen jeweils 2 Bit. Daraus ergibt sich, daß ebenfalls 128000 Bit für den Bildschirmspeicher benötigt werden.

Zu guter Letzt lassen sich im Modus 2 128000 Bildpunkte auf dem CPC 464 darstellen. Diese Bildpunkte können nur zwei verschiedene Farben annehmen, und der Computer benötigt ein Bit pro Punkt, um die beiden Farben auseinanderhalten zu können. Es ergibt sich wieder ein Speicherbedarf von 128000 Bit für das Videobild.

Sie sehen, in jedem Fall werden 128000 Bit für den Bildschirmspeicher belegt. Diese 128000 Bit entsprechen in etwa den oberen 16-KByte-RAM des CPC 464, die für das Videobild reserviert sind.

Jetzt ist das Phänomen der 400 Bildpunkte leicht erklärbar. Würde die Grafik des CPC 464 nicht 200, sondern 400 senkrechte Bildpunkte pro Spalte zur Verfügung stellen, dann müßte auch doppelt so viel Speicherplatz für den Bildschirmspeicher bereitgestellt werden, weil in jedem Fall 256000 Bit belegt werden würden. Als Resultat ständen dem Programmierer nicht 43,5 KByte RAM, sondern nur 27,5-KByte-RAM zur freien Verfügung, zweifellos nicht im Sinne des Erfinders.

Weil das BASIC des CPC 464 jedoch insgesamt 256000 Bildpunkte ansprechen kann, ist zu erwarten, daß in nicht allzu ferner Zukunft eine externe RAM-Erweiterung für den CPC 464 angeboten wird, die die Grafikauflösung des Computers verdoppelt, ohne seinen Arbeitsspeicher drastisch zu verkleinern. Eine höchste Auflösung von 256000 Bildpunkten würde sicher auch anspruchsvolle Benutzer zufriedenstellen!

2.4.2 Farben

Die Farben des CPC 464 sind durch Zahlen codiert. Diese Zahlen werden Farbwerte genannt.

Mit dem Befehl PEN kann die Schriftfarbe, mit dem Befehl PAPER die Hintergrundfarbe und mit dem Befehl BORDER die Randfarbe bestimmt werden. Nur bei dem Befehl BORDER kann direkt ein Farbwert angegeben werden. Bei den anderen beiden Befehlen müssen die Nummern von Farbregistern genannt werden, die die einzelnen Farbwerte enthalten.

Je höher die Grafikauflösung des CPC 464 ist, desto weniger Farbregister können benutzt werden und desto weniger Farben können gleichzeitig auf dem Bildschirm dargestellt werden. Durch Verändern von Farbregisterinhalten ist es jedoch möglich, einzelne Farben miteinander auszutauschen.

Die Farben und Farbwerte des CPC 464 lassen sich in folgende Gruppen aufteilen:

Bunte Farben:	Rote Farben:	Gelbe Farben:	Grüne Farben:
	16 Rosa	25 Pastellgelb	22 Pastellgrün
	7 Purpur	24 Hellgelb	21 Limonengrün
	6 Hellrot	15 Orange	19 Seegrün
	3 Rot	12 Gelb	18 Hellgrün
			9 Grün
	Türkise Farben:	Blaue Farben:	Violette Farben:
	23 Pastelltürkis	14 Pastellblau	17 Pastellmagenta
	20 Helltürkis	11 Himmelblau	8 Hellmagenta
	10 Türkis	2 Hellblau	5 Hellviolett
		1 Blau	4 Magenta
Unbunte Farben:	26 Weiß		
	13 Grau		
	0 Schwarz		

Verwenden Sie in Ihren Programmen möglichst nur eine Farbe aus jeder Gruppe, um einen ausreichenden Kontrast auf dem Bildschirm hervorzurufen.

Wenn Ihre Programme auch auf dem monochromen Bildschirm des CPC 464 einen guten optischen Eindruck machen sollen, müssen die Farbwerte von nebeneinanderliegenden Farben größer als vier sein. Die Farbwerte erzeugen ihrer Größe nach geordnet eine zunehmende Helligkeit auf dem Grünmonitor, und wenn die Farbwertdifferenz zweier benachbarter Farben zu gering ist, können die Farben vom Betrachter nicht mehr unterschieden werden.

Ein Beispiel: Auf dem Farbmonitor heben sich die Farben Hellblau (2) und Rot (3) klar voneinander ab, aber auf dem monochromen Bildschirm sind sie kaum zu unterscheiden.

Wenn Sie dagegen Blau (1) und Hellrot (6) wählen oder – noch besser – Himmelblau (11) und Rot (3), beträgt die Farbwertdifferenz fünf bzw. acht und ist auch auf dem Grünmonitor deutlich zu erkennen.

2.4.3 Fenster

Für die Ausgabe von Text auf den Bildschirm hat der CPC 464 einen besonderen Clou, die Bildschirmfenster.

Bildschirmfenster sind vom Benutzer definierte Bereiche des Bildschirms, die wie eigenständige Bildschirme behandelt werden können. Es ist z. B. möglich, in diesen Bereichen Programme zu listen und laufen zu lassen.

Auf dem CPC 464 können acht verschiedene, voneinander unabhängige Fenster mit dem Befehl WINDOW definiert werden. (Über solch eine Fenstertechnik verfügen neben dem CPC 464 nur Computer, die mindestens doppelt so teuer sind!)

Wenn Sie viele Bildschirmfenster verwenden, kann es geschehen, daß sich zwei Fenster zufällig oder absichtlich überschneiden. In diesem Fall gilt, daß der Text im neuen Fenster die Informationen des älteren Fensters überschreibt und löscht.

Wenn Sie zusätzlich zu den Bildschirmfenstern Grafik verwenden, müssen Sie wissen, daß sich die Befehle PLOT und DRAW über die Bildschirmfensterdefinitionen hinwegsetzen und «querfeldein» zeichnen.

Mit dem ORIGIN-Befehl ist es jedoch möglich, ein Grafikfenster zu definieren, wenn Sie ausschließlich positive Koordinatenwerte verwenden. So erzeugt z. B. der Befehl ORIGIN 320,200 ein Grafikfenster im rechten oberen Viertel des Bildschirms.

2.4.4 Umlaute

Sicherlich haben Sie sich schon darüber geärgert, daß der CPC 464 nicht über die deutschen Umlaute verfügt. Dem kann abgeholfen werden, indem Sie mit dem Befehl SYMBOL Ihre eigenen Umlautzeichen definieren.

Zweckmäßigerweise verwenden Sie für Umlaute diejenigen ASCII-Werte, die auch von Druckern bei aktiviertem deutschen Zeichensatz als Umlaute akzeptiert und ausgedruckt werden.

Folgende Übersicht zeigt für jeden Umlaut den ASCII-Wert und das alte Zeichen, mit dem der neue Umlaut über die Tastatur abgerufen werden kann:

Ä– 91–[ä–123–SHIFT [
Ö– 92–\ ö–124–SHIFT @
Ü– 93–] ü–125–SHIFT]
ß–126–CTRL 2

(Wie Sie sehen, wird auch das scharfe *s* mit berücksichtigt.)

Um diese Zeichen zu definieren, müssen Sie zuerst mit SYMBOL AFTER 91 die Matrizen der ASCII-Zeichen ab dem Wert 91 in den RAM laden, damit sie geändert werden können.

Folgende SYMBOL-Befehle definieren die neuen Zeichen:

```
SYMBOL 91,102,24,60,102,102,126,102,0
SYMBOL 92,102,60,102,102,102,102,60,0
SYMBOL 93,102,0,102,102,102,102,60,0
SYMBOL 123,102,0,120,12,124,204,118,0
SYMBOL 124,102,0,60,102,102,102,60,0
SYMBOL 125,102,0,102,102,102,102,62,0
SYMBOL 126,60,102,102,108,102,102,124,96
```

Die Matrizen der einzelnen Zeichen wurden so gewählt, daß sich die neuen Zeichen harmonisch in das Erscheinungsbild des bestehenden Zeichensatzes einfügen. An dieser Stelle sei nochmals darauf hingewiesen, daß Umlaute *nicht* in Variablennamen verwendet werden dürfen, weil sie aufgrund ihrer ASCII-Werte vom BASIC-Interpreter als Sonderzeichen behandelt werden.

2.4.5 Beispiele

In diesem Abschnitt sollen einige praktische Anwendungen den gelernten Stoff vertiefen.

Das folgende Programm zeichnet einen Kreis in die Mitte des Bildschirms:

```
10 REM *** Kreis ***
20 MODE 2:CLEAR:DEG
30 ORIGIN 320,200:MOVE 320,399
40 FOR lauf=0 TO 360
50 DRAW 199*SIN(lauf),199*COS(lauf)
60 NEXT lauf
```

Wenn Sie die Multiplikatoren in Zeile 50 vergrößern, stößt der Kreis an den Rand des Bildschirms; wenn Sie die beiden Werte verkleinern, wird auch der Kreis kleiner. Interessant wird es, wenn Sie den beiden Multiplikatoren jeweils unterschiedliche Werte zuweisen. In diesem Fall entsteht eine Ellipse, die umso gestreckter wird, je größer die Abweichung zwischen den beiden Multiplikatoren ist.

Das nächste Programm zeichnet zufällig geformte und verteilte Ellipsen auf den Bildschirm:

```
10 REM *** Ellipsen ***
20 MODE 2:CLEAR:DEFINT a-z:RANDOMIZE TIM
E:DEG
30 x=RND*640
40 y=RND*400
50 m=RND*50+50
60 n=RND*50+50
70 IF x+m>639 OR y+n>399 OR x-m<0 OR y-n
<0 THEN 30
80 ORIGIN x,y
90 MOVE 0,n
100 FOR lauf=1 TO 360
110 DRAW m*SIN(lauf),n*COS(lauf)
120 NEXT lauf
130 GOTO 30
```

In den Zeilen 30 und 40 wird der Koordinatenursprung, der die Position der Ellipse bestimmt, generiert, und in den Zeilen 50 und 60 werden die beiden Multiplikatoren aus einem Zahlenbereich von 50 bis 99 zufällig ausgewählt.

Zeile 70 prüft, ob die Ellipse vollständig innerhalb des Bildschirmbereichs liegt, und wenn dies der Fall ist, wird die Ellipse ausgegeben, andernfalls werden neue Zufallswerte erzeugt.

Die Grafik des CPC 464 macht es Ihnen leicht, mathematische Funktionen darzustellen. Folgendes Programm zeichnet eine Sinuskurve:

```
10 REM *** Sinuskurve ***
20 MODE 2:CLEAR:DEG
30 ORIGIN 0,200:DRAW 639,0
40 FOR lauf=0 TO 720
50 DRAW 640*lauf/720,199*SIN(lauf)
60 NEXT lauf
```

Wieder ist der Befehl ORIGIN zum Setzen des Koordinatenursprungs äußerst nützlich, weil keine Grafikwerte verschoben werden müssen, um in den sichtbaren Bereich des Bildschirms zu gelangen. Auch die zweite Hälfte der Sinusschwingung mit ihren negativen Koordinaten wird korrekt ausgegeben.

In Zeile 50 wird der Laufwert durch 720 geteilt, um zwei volle Sinusschwingungen auf dem Bildschirm zu erzeugen ($720 = 2 \times 360$ Grad), und mit 640 multipliziert, um die gesamte Bildschirmbreite auszunutzen.

Abgesehen von dieser Zeile dürfte Ihnen das Verständnis des Programms jedoch keine Schwierigkeiten bereiten.

Das letzte Programmbeispiel beschäftigt sich mit den Bildschirmfenstern. Es werden zufällig verteilte Bildschirmfenster von zufälliger Größe erzeugt und mit einer Zufallsfarbe ausgefüllt:

```
10 REM *** Bildschirmfenster ***
20 MODE 0:DEFINT a-z:RANDOMIZE TIME
30 FOR fenster=0 TO 7
40 a=RND*20+1
50 b=RND*20+1
60 c=RND*25+1
70 d=RND*25+1
```

```
80 farbe=RND*26
90 farbregister=RND*15
100 BORDER farbe
110 WINDOW #fenster,a,b,c,d
120 PAPER #fenster,farbregister
130 CLS #fenster
140 NEXT fenster
150 GOTO 30
```

In den Zeilen 40 bis 90 werden die Zufallswerte generiert und in den Zeilen 100 bis 130 ausgegeben. Dieser Programmteil wird von einer FOR-NEXT-Schleife eingeschlossen, die dafür sorgt, daß alle Bildschirmfenster von null bis sieben genau einmal definiert werden. Nach Beendigung der Programmschleife beginnt das ganze Spiel von vorne.

Anhand dieses Programms können Sie deutlich sehen, daß der Inhalt von neuen Bildschirmfenstern alte Fenster überschreibt.

2.5 Der Tongenerator

Der Tongenerator des CPC 464 ist in seiner Handhabung erst mit etwas Übung in den Griff zu bekommen. Für die Ansteuerung des Tongenerators stehen dem Programmierer zwar nur drei Befehle zur Verfügung (SOUND, ENV und ENT), doch diese Befehle haben es in sich!

Rein rechnerisch können fast $2{,}5 \times 10^{84}$ Ton- und Geräuschfolgen mit diesen Befehlen erzeugt werden.

Das Bedienungshandbuch des CPC 464 erklärt die Bedeutung und Anwendung der Befehle zum Programmieren des Tongenerators zum Teil recht verwirrend. Deshalb werden sich die nächsten Abschnitte dieses Kapitels ausführlich mit den Möglichkeiten der Programmierung des Tongenerators beschäftigen.

2.5.1 Töne und Rauschen

Der Tongenerator des CPC 464 verfügt über drei Kanäle zur Ausgabe von Tönen und über einen Kanal zur Ausgabe von Rauschen.

Töne können gespielt werden, indem der Parameter *Frequenzteiler* des SOUND-Befehls auf einen Wert ungleich null gesetzt wird. Das Rauschen

wird über den letzten Parameter des SOUND-Befehls *Rauschen* gesteuert. Wenn dieser Wert ungleich null ist, wird ein sogenanntes *Weißes Rauschen* erzeugt, das aus einer Überlagerung aller hörbaren Frequenzen besteht.

Es ist möglich, gleichzeitig Töne und Rauschen zu erzeugen, wenn beide Parameter, Frequenzteiler und Rauschen, einen von null verschiedenen Wert erhalten. Weil für die Tonausgabe drei getrennte Kanäle zur Verfügung stehen, ist es beim Anschluß einer Stereoanlage an den CPC 464 möglich, Stereoeffekte zu erzielen, z. B. kann bei einer zweistimmigen Melodie die erste Stimme über den linken Lautsprecher und die zweite Stimme über den rechten Lautsprecher ausgegeben werden und umgekehrt.

Für die Ausgabe des Rauschens steht dagegen nur ein Kanal bereit, und folglich ist keine Stereoausgabe von Rauschen möglich. Es kann z. B. kein hohes Rauschen über den einen und gleichzeitig ein dumpfes Rauschen über den anderen Lautsprecher der Stereoanlage ausgegeben werden. Nur die Ausgabe von Rauschen über eine bestimmte Seite oder beide Seiten simultan ist möglich.

2.5.2 SOUND

Der Befehl SOUND ist das wichtigste Kommando zur Tonausgabe und kann bis zu sieben Parameter verarbeiten.

Folgende Übersicht führt die einzelnen Parameter der Reihe nach auf und gibt in Klammer jeweils den Wertebereich an:

Kanalstatus	(0 bis 255)
Frequenzteiler	(0 bis 4095)
Dauer	(−32768 bis 32767)
Lautstärke	(0 bis 15)
Lautstärkenhüllkurvennummer	(0 bis 15)
Frequenzhüllkurvennummer	(0 bis 15)
Rauschen	(0 bis 31)

Zu den einzelnen Parametern einige Kommentare:

Kanalstatus

Der Wertebereich von *Kanalstatus* beträgt genau ein Byte. Jedes Bit dieses Bytes hat eine besondere Bedeutung:

0. Bit – SOUND über den linken Kanal ausgeben (1)
1. Bit – SOUND über den mittleren Kanal ausgeben (2)
2. Bit – SOUND über den rechten Kanal ausgeben (4)
3. Bit – Synchronisation mit dem linken Kanal (8)
4. Bit – Synchronisation mit dem mittleren Kanal (16)
5. Bit – Synchronisation mit dem rechten Kanal (32)
6. Bit – SOUND abwarten (64)
7. Bit – SOUND sofort ausführen (128)

Wenn Sie eine bestimmte Funktion wünschen, müssen Sie die Zahl in der Klammer als Wert für den *Kanalstatus* verwenden. Wenn mehrere Funktionen parallel durchgeführt werden sollen, müssen Sie die Werte der entsprechenden Bits addieren und als *Kanalstatus* angeben, z. B. bewirkt der *Kanalstatus* 135, daß sofort über alle drei Kanäle der folgende SOUND ausgegeben wird, denn $135 = 1 + 2 + 4 + 128$.

Wenn Sie einen *Kanalstatus* mit gesetztem Bit 6 angeben, wird der folgende SOUND erst ausgeführt, wenn der Wartezustand mit dem Befehl RELEASE aufgehoben wird.

Frequenzteiler

Die Ausgabefrequenz eines Tons wird errechnet, indem 125000 durch den *Frequenzteiler* geteilt wird. Folglich gibt der *Frequenzteiler* die Höhe des auszugebenden Tons an.

Weil der Wert des *Frequenzteilers* in einem Bereich von 0 bis 4095 liegen darf, sind Frequenzen von 30,5 Hz bis 125 kHz möglich. (Wenn der *Frequenzteiler* auf null gesetzt wird, erfolgt keine Tonausgabe.)

Wenn Ihnen diese Frequenzangaben wenig sagen, schauen Sie sich folgende Aufstellung an, die angibt, welches Frequenzspektrum der Mensch in welchem Alter wahrnehmen kann:

20 Jahre – 16 Hz bis 20 kHz
30 Jahre – 16 Hz bis 16 kHz
45 Jahre – 16 Hz bis 12 kHz
60 Jahre – 16 Hz bis 6 kHz

Wie Sie sehen, nimmt das Hörvermögen von hohen Tönen mit zunehmendem Alter drastisch ab. Man kommt jedoch auch mit einer oberen Hörgrenze von 6 kHz noch gut zurecht, weil selbst höchste Töne der Musik diesen Wert selten überschreiten. Lediglich das Differenzieren von Instrumenten fällt schwerer, weil die klangfarbencharakteristischen Oberschwingungen nicht mehr wahrgenommen werden können.

Die menschliche Sprache ist ein Gemisch verschiedener, sich überlagernder Frequenzen und geht selbst bei hohen Stimmen über 1 kHz nicht hinaus.

Mit diesem kleinen Ausflug in die Akustik wird Ihnen das Einschätzen von Frequenzen sicherlich leichterfallen.

Dauer

Der Parameter *Dauer* bestimmt die Länge des Tons oder des Rauschens. Wenn der Wert positiv ist, gibt er die Länge in Hundertstelsekunden an, ist die Zahl negativ, wird die Lautstärkenhüllkurve dem positiven Wert entsprechend oft wiederholt.

Lautstärke

Je größer der Wert des Parameters *Lautstärke* ist, desto voluminöser ist der Ausgabeton bzw. das Rauschen.

Ohne Lautstärkenhüllkurve entspricht der Wertebereich 0 bis 7 dem Bereich 8 bis 15. Mit Lautstärkenhüllkurve kann feiner differenziert werden, weil dann für das Lautstärkenspektrum alle 16 Werte zur Verfügung stehen.

Lautstärkenhüllkurvennummer

Diese Nummer gibt an, welche ENV-Zuweisung für den SOUND-Befehl gültig ist.

Frequenzhüllkurvennummer

Diese Nummer gibt an, welche ENT-Zuweisung für den SOUND-Befehl gültig ist.

Rauschen

Der letzte Parameter des SOUND-Befehls gibt die Höhe des Rauschens an. Je größer dieser Wert ist, desto dumpfer wird das Rauschen. Wenn Sie als Wert eine Null angeben, wird kein Rauschen erzeugt.

Allgemein ist zu den Parametern noch folgendes zu sagen: Alle Werte müssen ganzzahlig sein. Wenn Sie andere Zahlen angeben, rundet der BASIC-Interpreter automatisch zur nächstliegenden ganzen Zahl. Wenn ein Wert null beträgt, muß er nicht angegeben werden. Steht er am Ende des SOUND-Befehls, kann er weggelassen werden, ansonsten können Sie ersatzweise zwei aufeinanderfolgende Kommas setzen.

2.5.3 ENV

Der Befehl *ENV* generiert die Lautstärkenküllkurve. Es können bis zu 16 Parameter angegeben werden. Der erste Parameter bestimmt die Hüllkurvennummer, die weiteren Werte formen jeweils als Dreiergruppen einzelne Ausschnitte der Hüllkurve.

Es folgt eine Aufstellung von Parametern und Wertebereichen:

Hüllkurvennummer (1 bis 15)
1. Schrittanzahl (0 bis 127)
1. Schrittgröße (−128 bis 127)
1. Schrittzeit (0 bis 255)
⋮
5. Schrittanzahl (0 bis 127)
5. Schrittgröße (−128 bis 127)
5. Schrittzeit (0 bis 255)

Weil die Lautstärke des SOUND-Befehls bei großen oder vielen Schritten leicht den Wert 15 überschreiten kann, gilt: *Lautstärke* = *Lautstärke* MOD 16, d.h., die *Lautstärke* 16 entspricht der *Lautstärke* null usw.

Damit ist nicht zu theoretisch wird, zeigt Bild 2.1 ein konkretes Beispiel, das aus drei Hüllkurvenabschnitten besteht.

Der erste Hüllkurvenabschnitt steigert die Lautstärke in 15 Schritten von je 4 hundertstel Sekunden Länge auf ihren Maximalwert, sofern der Parameter *Lautstärke* des zugehörigen SOUND-Befehls gleich null ist.

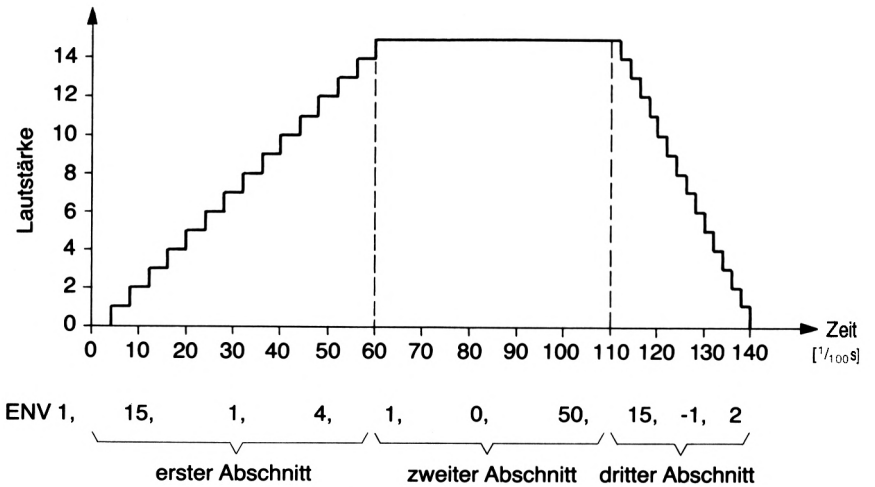


Bild 2.1 Beispiel einer Lautstärkenhüllkurve

(Wenn der Wert ungleich null ist, wird die *Lautstärke* 15 überschritten und beginnt wieder mit null.)

Der zweite Hüllkurvenabschnitt hält die Lautstärke für die Dauer von einer halben Sekunde konstant, und der dritte Abschnitt läßt die Lautstärke in 15 Schritten zu je 2 hundertstel Sekunden Länge auf ihren Ausgangswert (null) abklingen.

Wenn Sie mit dem ENV-Befehl eine Hüllkurve formen, die die Lautstärke eines Tons periodisch an- und abschwellen läßt, können Sie mit diesem Befehl auf dem CPC 464 hervorrangende Vibratoeffekte erzeugen.

2.5.4 ENT

Der Befehl ENT generiert die Frequenzhüllkurve. Es können bis zu 16 Parameter angegeben werden. Der erste Parameter bestimmt die Hüllkurvennummer, die weiteren Werte formen jeweils als Dreiergruppen einzelne Ausschnitte der Hüllkurve.

Es folgt eine Aufstellung von Parametern und Wertebereichen:

Hüllkurvennummer (−15 bis 15)
1. Schrittanzahl (0 bis 239)
1. Schrittgröße (−128 bis 127)
1. Schrittzeit (0 bis 255)

⋮

5. Schrittanzahl (0 bis 239)
5. Schrittgröße (−128 bis 127)
5. Schrittzeit (0 bis 255)

Die Hüllkurvennummer der Frequenzhüllkurve kann im ENT-Befehl ein negatives Vorzeichen erhalten. Dieses Vorzeichen gibt an, daß die Hüllkurve so oft wiederholt wird, bis der Ton des SOUND-Befehls zum Ende gekommen ist. Erhält die Nummer der Hüllkurve kein negatives Vorzeichen, läßt der CPC 464 die Hüllkurve nur ein einziges Mal erzeugen.

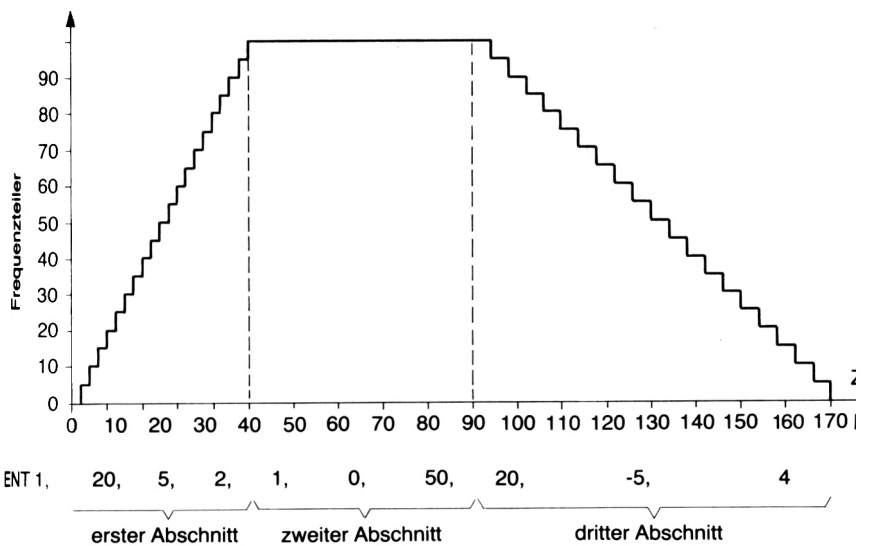


Bild 2.2 Beispiel einer Frequenzhüllkurve

Die Dimensionierung der Schrittgröße des ENT-Befehls fällt leichter, wenn man weiß, daß eine positive Schrittweite den Ton tiefer und eine negative Schrittweite den Ton höher klingen läßt.

Zur Verdeutlichung der Informationen zeigt Bild 2.2 ein konkretes Beispiel, das aus drei Hüllkurvenabschnitten besteht.

Der erste Hüllkurvenabschnitt erhöht den Frequenzteiler des zugehörigen SOUND-Befehls in 20 Schritten von je 2 hundertstel Sekunden Länge jeweils um 5.

Der zweite Hüllkurvenabschnitt hält den Frequenzteiler für die Dauer von einer halben Sekunde konstant, und der dritte Abschnitt läßt den Frequenzteiler wieder in 20 Schritten zu je 4 hundertstel Sekunden Länge auf seinen Ursprungswert zurückschrumpfen.

Mit dem ENT-Befehl können Sie auf dem CPC 464 gute Sireneneffekte erzielen, wenn Sie die Frequenz eines Tons periodisch vergrößern und verkleinern.

2.5.5 Stereo

Der CPC 464 verfügt nur über einen einzelnen, eingebauten Lautsprecher und kann deshalb lediglich Töne in Mono ausgeben. Es lassen sich allerdings Stereotöne erzeugen, und wenn Sie eine Stereoanlage besitzen, können sie Ihren Computer daran anschließen (siehe auch Abschnitt 1.6.4) und diese Töne hören.

Mit der Stereoausgabe von Tönen und Geräuschen (Geräusche mit Einschränkung, siehe Abschnitt 2.5.1) lassen sich überraschende Effekte erzielen. Mehrstimmige Lieder können über verschiedene Lautsprecher ausgegeben werden, Töne und Geräuse können akustisch bewegt werden.

Um einen Ton oder ein Geräusch so naturgetreu wie möglich von links nach rechts wandern zu lassen, sollten Sie diese Reihenfolge bei der Ausgabe über die verschiedenen Kanäle einhalten:

1. Kanal 1		
2. Kanal 1	Kanal 2	
3. Kanal 1		Kanal 3
4. Kanal 1	Kanal 2	Kanal 3
5. Kanal 1		Kanal 3
6.	Kanal 2	Kanal 3
7.		Kanal 3

Es ergibt sich der Effekt eines vorüberziehenden sowie an- und abschwellenden Tons oder Geräuschs.

Natürlich ist auch eine Bewegung von rechts nach links möglich. Sie müssen lediglich die Reihenfolge der Aufstellung umkehren.

2.5.6 Anwendungen

Dieser Abschnitt bietet zu dem umfangreichen Thema der Tongeneratorprogrammierung einige Anwendungsbeispiele.

Wenn Sie auf dem CPC 464 musikalische Töne bestimmter Oktaven und Notenwerte erzeugen wollen, ist diese Befehlsfolge recht nützlich:

$$125000/(32{,}7032*2((ton-1)/12+oktave-1))$$

Das Ergebnis dieser Rechnung muß als *Frequenzteiler* in einen SOUND-Befehl eingesetzt werden.

Für die Variable *oktave* müssen Sie die Nummer der gewünschten Oktave ausgehend von der Kontra-Oktave angeben, und für die Variable *ton* geben Sie die Nummer des zu spielenden Halbtons innerhalb der Oktave an.

Nummer	Oktave	Ton
1	Kontraoktave	C
2	große Oktave	Cis
3	kleine Oktave	D
4	eingestrichene Oktave	Dis
5	zweigestrichene Oktave	E
6	dreigestrichene Oktave	F
7	viergestrichene Oktave	Fis
8	fünfgestrichene Oktave	G
9	sechsgestrichene Oktave	Gis
10	siebengestrichene Oktave	A
11	achtgestrichene Oktave	Ais
12	neungestrichene Oktave	H

Tabelle 2.1 Oktaven und Töne auf dem CPC 464

Tabelle 2.1 zeigt alle Oktaven und Töne, die auf dem CPC 464 mit dieser Befehlsfolge möglich sind.

Wie Sie wissen, verfügt der CPC 464 über drei verschiedene Kanäle zur Tonausgabe. Die SOUNDs von Ton- und Geräuschfolgen werden in die entsprechenden Warteschlangen der Kanäle geladen und so bald wie möglich abgespielt.

Wenn Sie Töne oder Geräusche nur über einen Kanal ausgeben, werden diese SOUNDs nacheinander gespielt.

Wenn Sie jedoch verschiedene Kanäle ansprechen, werden die SOUNDs parallel über diese Kanäle ausgegeben.

Manchmal ist es erwünscht, SOUND-Befehle nacheinander über verschiedene Kanäle auszugeben. In diesem Fall darf die Warteschlange des einen Kanals erst geladen werden, wenn die Tonausgabe des anderen Kanals beendet ist.

Folgende Warteschleife wird so lange durchlaufen, bis die Ausgabe über Kanal *kanal* zum Ende gekommen ist:

WHILE SQ(*kanal*)>127:WEND

Diese WHILE-WEND-Schleife prüft bei jedem Durchgang, ob Bit 7 des Warteschlangenzustandes von Kanal *kanal* gesetzt ist, d.h. der Kanal *kanal* aktiv ist. Ist dies der Fall, wird die Schleife erneut durchlaufen, ansonsten wird sie beendet, und ein folgender SOUND-Befehl kann die Warteschlange des nächsten Kanals zur Tonausgabe laden.

Abschließend eine kleine Demonstration der Geräuschmöglichkeiten in Form eines Raketenstarts:

```
10 REM *** Raketenstart ***
20 FOR lauf=31 TO 1 STEP -1
30 SOUND 2,0,n,(lauf-2)/4,,,lauf
40 NEXT lauf
50 GOTO 10
```

Die Schleife dieses Programmbeispiels schlägt zwei Fliegen mit einer Klappe: Das ausgegebene Rauschen wird während der Erhöhung der Variablen *lauf* leiser und gleichzeitig höher, um den Effekt einer sich entfernenden Rakete zu erzielen.

Wenn Sie für die Variable *n* den Wert 50 einsetzen, ergibt sich der Start einer Großrakete, und bei $n = 5$ hören Sie eine Kleinstrakete starten.

Weil das Rauschen dieses Beispielprogramms ausschließlich über den mittleren Kanal ausgegeben wird und sich somit die einzelnen „Rauscher" nicht in die Quere kommen, kann auf die WHILE-WEND-Schleife verzichtet werden.

Viele weitere Beispiele zu den Ton- und Geräuschmöglichkeiten des CPC 464 mit Stereoeffekten und ausführlichen Erläuterungen bietet das Demonstrationsprogramm in Kapitel 3.

2.6 Unterbrechungen

Aus Abschnitt 1.2 wissen Sie bereits, daß der Mikroprozessor des CPC 464 drei verschiedene *Interruptmöglichkeiten* besitzt. Doch was genau ist ein Interrupt? Exakt formuliert ist ein Interrupt eine hard- oder softwaremäßige Unterbrechung der momentanen Tätigkeit eines Mikroprozessors. Allgemeiner ausgedrückt bewirkt ein Interrupt den Abbruch der Bearbeitung eines aktuellen Programmteils und den Sprung in einen neuen Programmabschnitt.

Die Z-80-CPU verfügt über hervorragende Interruptmöglichkeiten, und der CPC 464 als Homecomputer mit einer Z-80-CPU berücksichtigt diese Tatsache in seiner Firmware. Es ist nämlich mit einfachen BASIC-Befehlen möglich, leistungsfähige Interruptverarbeitungsmöglichkeiten zu erzielen.

Ein schönes Beispiel ist der Befehl ON SQ(*kanal*) GOSUB *zeile*. Wenn Sie diesen Befehl an den Anfang eines Programmblocks setzen, bewirkt er, daß bei leerer Tonwarteschlange von Kanal *kanal* augenblicklich die Bearbeitung dieses Blocks unterbrochen und das Unterprogramm ab Zeile *zeile* aufgerufen wird. Im Unterprogramm kann die Warteschlange wieder aufgefüllt werden und mit einem normalen RETURN ein Rücksprung an die Stelle des Programmblocks erfolgen, an der er verlassen wurde.

Dies ist möglich, weil sich der CPC 464 bei einem Interrupt automatisch den genauen Punkt der Unterbrechung für einen eventuellen Rücksprung merkt.

An weiteren Interruptmöglichkeiten existieren die Befehle AFTER, EVERY und REMAIN.

Der Befehl AFTER *zeit, timer* GOSUB *zeile* lädt den Zeitgeber *timer* mit dem Wert *zeit* und startet den Zeitgeber. Wenn die Zeit *zeit* auf dem

Zeitgeber *timer* verstrichen ist, erfolgt ein Unterprogrammaufruf ab Programmzeile *zeile*.

EVERY *zeit, timer* GOSUB *zeile* lädt den Zeitgeber *timer* mit dem Wert *zeit* und startet den Zeitgeber. Jedesmal, wenn die Zeit des Zeitgebers *timer* abgelaufen ist, wird ein Unterprogramm ab Zeile *zeile* aufgerufen, und gleichzeitig beginnt die Ausführung dieses Befehls erneut.

Bei beiden Befehlen kann das Unterprogramm mit einem normalen RETURN-Befehl beendet werden, und es erfolgt ein Rücksprung an die Stelle des Hauptprogramms, an der es verlassen wurde. Mit REMAIN *timer* kann die Restzeit des Zeitgebers *timer* abgefragt und gleichzeitig der Zeitgeber *timer* auf null zurückgesetzt werden. Auf diese Weise lassen sich Zeitgeber einfach abschalten.

Es kann nun vorkommen, daß mehrere Zeitgeber gleichzeitig eine Programmunterbrechung und einen Unterprogrammaufruf anfordern. Wenn nun jeder Interrupt ein anderes Unterprogramm aufrufen möchte, könnte der CPC 464 in Schwierigkeiten geraten. Deswegen sind die Zeitgeber nach einer bestimmten Priorität geordnet. Zeitgeber drei hat die höchste Priorität und gibt vor allen anderen Zeitgebern das aufzurufende Unterprogramm an. Entsprechend hat Zeitgeber null die niedrigste Priorität, und alle anderen Zeitgeber gehen vor.

2.7 Das Betriebssystem

Das Betriebssystem eines Computers ist für die Verwaltung des Gerätes, der Peripherie und der Eingaben des Benutzers zuständig.

Die folgenden Abschnitte beschäftigen sich speziell mit der Verwaltung der Benutzerdaten. Sie gehen der Frage nach, wo welche Daten im Arbeitsspeicher des CPC 464 abgespeichert werden. Zu diesem Zweck werden Speicherbelegung und Formate der abgelegten Daten beleuchtet.

Der letzte Abschnitt zeigt schließlich ein praktisches Verfahren zum Retten von nichtlesbaren Programmen, das sich aus der Anwendung der folgenden Informationen ergibt.

2.7.1 Speicheraufteilung

Der CPC 464 verfügt über 32 KByte ROM und 64 KByte RAM. Aus Kapitel 1 wissen Sie aber, daß die Z-80-CPU nur 64 KByte Speicher adressieren kann. Die Konstrukteure des CPC 464 haben dieses Problem gelöst, indem sie den ROM parallel zum RAM legten. Die unteren 16 KByte des ROM liegen parallel zu den unteren 16 KByte des RAM und die oberen 16 KByte des ROM parallel zu den oberen 16 KByte des RAM. Bild 2.3 verdeutlicht diese Aufteilung.

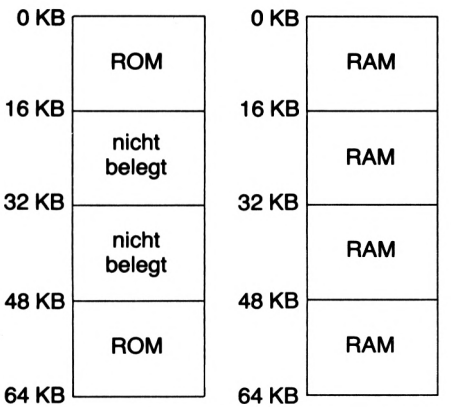


Bild 2.3 Speicheraufteilung von ROM und RAM im CPC 464

Damit die Z-80-CPU mit ihrem Speicher nicht durcheinander kommt, lassen sich die diverse RST-Routinen ROM und RAM jeweils aktivieren und desaktivieren, um darauf zugreifen zu können. Diese RST-Routinen werden von Assemblerspezialisten benutzt, um einen Blick in die Firmware des CPC 464 zu werfen. Dem BASIC-Programmierer bleibt diese Möglichkeit allerdings verschlossen, weil mit den Befehlen PEEK und POKE nicht auf den ROM des CPC 464 zugegriffen werden kann.

Im RAM kann dagegen nach Herzenslust gePEEKt und gePOKEt werden. Die Tabelle 2.2 zeigt ausführlich, wo im RAM welche Daten des Betriebssystems abgespeichert werden.

Adresse	Inhalt
0 bis 63	RST-Routinen für Speicherverwaltung und Unterbrechungen (Kopie der gleichnamigen ROM-Adressen)
64 bis 321	Ausführungspuffer (enthält aktuelle Befehls- oder Programmzeile, von drei Nullen beendet)
322 bis 367	Null
368	Beginn des BASIC-Programms
368 bis *n*	BASIC-Programm, von drei Nullen beendet
n + 1 bis ...	Variablentabellenbeginn
... bis 43903	Stringspeicher (die zuletzt eingegebenen Strings stehen am Anfang)
43903	Stringspeicherende und maximaler HIMEN-Wert
43904 bis 43911	Punktmatrix des ASCII-Zeichens 240
43912 bis 43919	Punktmatrix des ASCII-Zeichens 241
43920 bis 43927	Punktmatrix des ASCII-Zeichens 242
43928 bis 43935	Punktmatrix des ASCII-Zeichens 243
43936 bis 43943	Punktmatrix des ASCII-Zeichens 244
43944 bis 43951	Punktmatrix des ASCII-Zeichens 245
43952 bis 43959	Punktmatrix des ASCII-Zeichens 246
43960 bis 43967	Punktmatrix des ASCII-Zeichens 247
43968 bis 43975	Punktmatrix des ASCII-Zeichens 248
43976 bis 43983	Punktmatrix des ASCII-Zeichens 249
43984 bis 43991	Punktmatrix des ASCII-Zeichens 250
43992 bis 43999	Punktmatrix des ASCII-Zeichens 251
44000 bis 44007	Punktmatrix des ASCII-Zeichens 252
44008 bis 44015	Punktmatrix des ASCII-Zeichens 253
44916 bis 44023	Punktmatrix des ASCII-Zeichens 254
44024 bis 44031	Punktmatrix des ASCII-Zeichens 255
44032 bis 44195	Betriebssystemdaten
44196 bis 44450	Eingabepuffer (enthält aktuell eingegebene ASCII-Zeichen, von einer Null beendet)
44451 bis 49151	Betriebssystemdaten
44592, 44593	Pointer, zeigt auf BASIC-Programmanfang-1
44667, 44668	aktueller HIMEM-Wert
44669, 44670	maximaler HIMEM-Wert
44673, 44674	Pointer, zeigt auf BASIC-Programmanfang-1
44675, 44676	Pointer, zeigt auf Variablentabellenbeginn
44677, 44678	Pointer, zeigt auf Variablentabellenbeginn
45197, 45198	Pointer, zeigt auf Stingspeicherbeginn
45199, 45200	Pointer, zeigt auf Stingspeicherende
47360 bis 47393	Einstiegsadressen zu ROM-Routinen
47872 bis 48441	Einstiegsadressen zu ROM-Routinen
48589 bis 48627	Einstiegsadressen zu ROM-Routinen
49152 bis 65535	Bildschirmspeicher
49152 bis 51151	1. Zeile der Bildschirmausgabepositionen
51200 bis 53199	2. Zeile der Bildschirmausgabepositionen
53248 bis 55247	3. Zeile der Bildschirmausgabepositionen
55296 bis 57295	4. Zeile der Bildschirmausgabepositionen
57344 bis 59343	5. Zeile der Bildschirmausgabepositionen
59392 bis 61391	6. Zeile der Bildschirmausgabepositionen
61440 bis 63439	7. Zeile der Bildschirmausgabepositionen
63488 bis 65487	8. Zeile der Bildschirmausgabepositionen

Tabelle 2.2 RAM-Speicherbelegung des CPC 464

2.7.2 POKEN von Daten

Das POKEN von Zahlen, die kleiner als 256 sind, ist nicht schwer, denn dies ist mit einem einzigen POKE-Befehl möglich.

Manchmal ist es jedoch notwendig, eine Adresse in den RAM zu POKEN, z. B. um einen Zeiger zu verändern. Weil jedoch eine Adresse bis zu 65535 betragen kann, muß sie in zwei Speicherzellen abgelegt werden. Doch wie gelangen wir von einer Adresse zu zwei POKE-Werten?

In die erste Speicherzelle muß immer das LSB (*niederstwertiges Byte*) der Adresse gePOKEt werden, und in die zweite Speicherzelle wird das MSB (*höchstwertiges Byte*) der Adresse geladen. Dazu führen Sie folgende Berechnung durch:

```
MSB = INT(Adresse/256)
LSB = Adresse-MSB*256
```

Die beiden Ergebnisse dieser Operation werden in die entsprechenden Speicherstellen gePOKEt – fertig!

2.7.3 ROM-Routinen

Wie Sie bereits in Abschnitt 2.7.1 gesehen haben, existieren im RAM spezielle Einstiegsadressen für ROM-Routinen. Die Tabelle 2.3 führt einige dieser Adressen auf, die in ihrer Handhabung auch für den Nicht-Assembler-Programmierer völlig problemlos sind, weil keine Parameterübergabe erforderlich ist und die Routinen mit dem BASIC-Befehl CALL aufgerufen werden können.

Das Betriebssystem des CPC 464 selbst verwendet diese Adressen nicht. Wenn Sie ebenfalls keine dieser Adressen benötigen, können Sie diesen Speicherplatz sinnvoll nutzen, indem Sie an dieser Stelle Ihre eigenen RAM-Routinen plazieren und über den CALL-Befehl anspringen.

2.7.4 Bildschirmspeicher

Der Bildschirmspeicher des CPC 464 belegt die obersten 16 KByte des RAMs. Folgendes kleine Programm füllt den Bildschirm mit der Farbe aus Farbregister Eins aus:

Adresse	Funktion
0	RESET
47878	wartet auf die Eingabe eines Zeichens (kein SHIFT oder CRTL)
47896	wartet auf die Eingabe eines Zeichens (kein SHIFT oder CRTL)
47941	nach BREAK folgt automatisch RESET (Die Befehle CLEAR und INPUT heben diese Routine auf)
47944	verhindert BREAK (Die Befehle CLEAR und INPUT heben diese Routine auf)
47947	entspricht dem einmaligen Drücken der Taste ESC
47956	Bildschirmfreigabe (nur im laufenden Programm verwendbar)
47959	Bildschirmsperrung (nur im laufenden Programm verwendbar)
47980	entspricht dem BASIC-Befehl CLS
48028	entspricht dem BASIC-Befehl PRINT CHR$(24)
48091	entspricht dem BASIC-Befehl CLG
48148	entspricht der Befehlsfolge PAPER 0:CLS
48238	startet den Motor des Kassettenrecorders
48341	stoppt den Motor des Kassettenrecorders
48307	Tonfreigabe
48310	stoppt aktuellen Ton
48313	startet aktuellen Ton
48409	wartet auf den vollständigen Bildaufbau
48439	restauriert den Einstiegsadressenblock 47872–48441

Tabelle 2.3 Einstiegsadressen für ROM-Routinen

```
10 FOR zeile=1 TO 8
20 zusatz=zusatz+2048
30 FOR lauf=47104+zusatz TO 49103+zusatz
40 POKE lauf,255
50 NEXT lauf
60 NEXT zeile
```

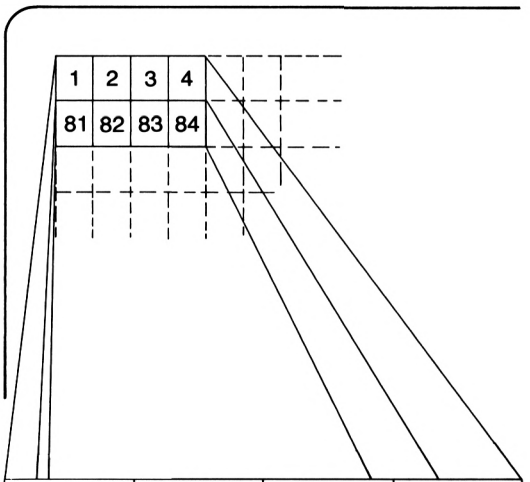
Anhand dieses Beispiels können Sie sehen, daß das POKEn in den Bildschirmspeicher ziemlich kompliziert ist. Der CPC 464 besitzt 2000 Bildschirmpositionen, auf die ASCII-Zeichen ausgegeben werden können. Weil die Zeichen des Computers aus einer 8×8-Punktmatrix aufgebaut sind, besteht jede Bildschirmausgabeposition aus acht Zeilen, und jede Zeile setzt sich wiederum aus acht Punkten zusammen.

Die Zeilen der Bildschirmpositionen werden im Bildschirmspeicher des CPC 464 jeweils durch ein Byte dargestellt. Das Besondere an der Belegung des Bildschirmspeichers ist, daß nicht die Zeilen *einer* Bildschirmposition hintereinander im Speicher folgen, sondern eine Zeile *aller* Bildschirmausgabepositionen in einem 2000-Byte-Block zusammengefaßt ist, so daß der *erste* Block des Bildschirmspeichers die *erste* Zeile aller Bildschirmpositionen enthält, der *zweite* Block die *zweite* Bildschirmzeile aller Ausgabepositionen usw.

Bild 2.4 zeigt für die linke obere Ecke des Bildschirms, welche Speicherzellen des Bildschirmspeichers für welche Bildschirmzeilen zuständig sind. Das Schema läßt sich für die anderen Bildschirmpositionen entsprechend erweitern.

Weil dem CPC 464 je Zeile genau ein Byte an Speicherplatz zur Verfügung steht, kann nur eine begrenzte Anzahl von Informationen für jede Bildschirmzeile abgespeichert werden. Der Bildschirmmodus des Computers gibt an, wieviel Bildpunkte einer Zeile unabhängig voneinander angesprochen werden können und wieviel Farben für diese Punkte möglich sind. Die Farbe eines Bildpunktes oder einer Gruppe von Bildpunkten wird durch den Inhalt eines Farbregisters bestimmt. Die Nummer dieses Registers wird durch den dezimalen Wert einer Binärzahl angegeben, und die Ziffern dieses Binärwertes setzen sich letztendlich aus den einzelnen Bits der angesprochenen Bildpunkte zusammen.

Im hochauflösenden Modus 2 ist dieses System noch einfach zu durchschauen:



49152	49153	49154	49155
51200	51201	51202	51203
53248	53249	53250	53251
55296	55297	55298	55299
57344	57345	57346	57347
59392	59393	59394	59395
61440	61441	61442	61443
63488	63489	63490	63491
49232	49233	49234	49235
51280	51281	51282	51283
53328	53329	53330	53331
55376	55377	55378	55379
57424	57425	57426	57427
59472	59473	59474	59475
61520	61521	61522	61523
63568	63569	63570	63571

Bild 2.4 Speicherzellen für Bildschirmzeilen

Jeder Bildpunkt einer Bildschirmzeile kann direkt durch ein Bit des zugehörigen Zeilenbytes angesteuert werden. Die Farbe des Bildpunkts wird durch den Zustand des entsprechenden Bits festgelegt. Wenn das Bit auf *eins* gesetzt ist, erhält der Punkt die Farbe aus Farbregister *eins*, wird das Bit auf null zurückgesetzt, nimmt der Punkt die Farbe des Farbregisters *null* an.

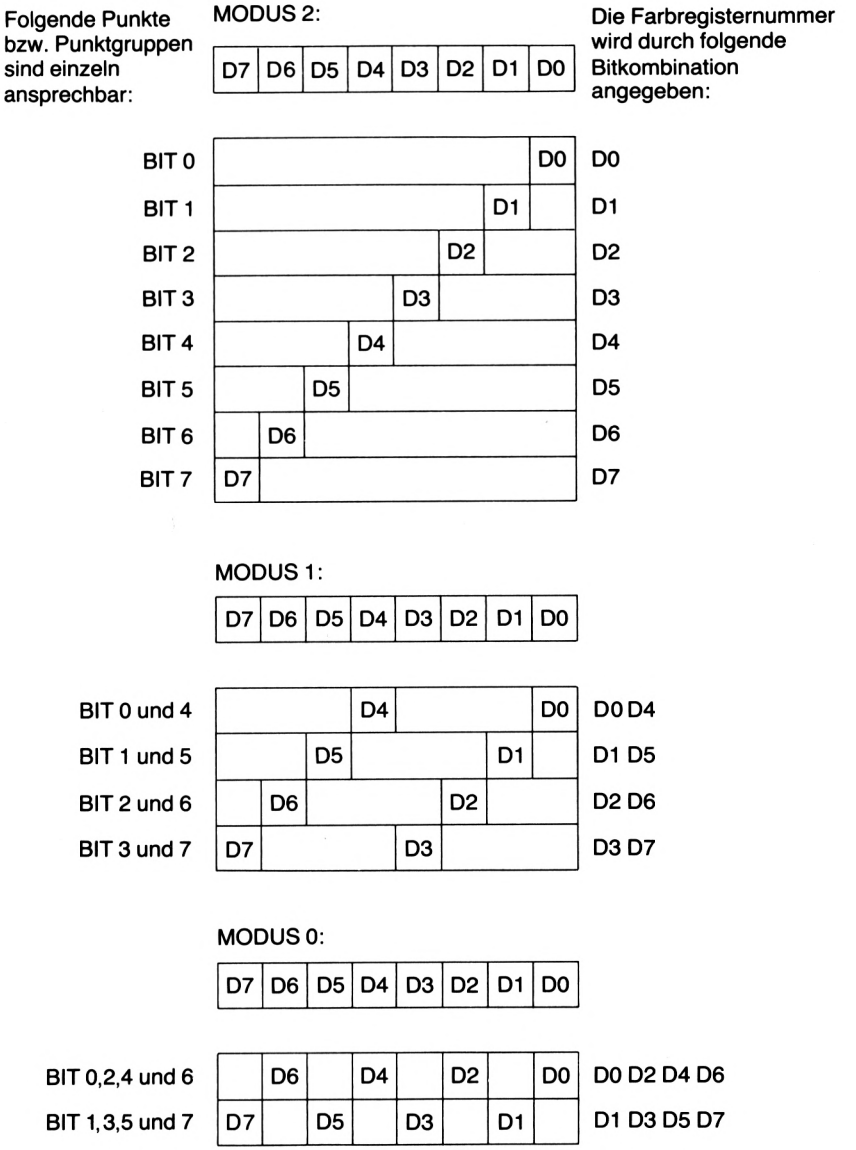


Bild 2.5 Funktion der einzelnen Bits eines Bildschirmzeilenbytes

In den beiden anderen Modi ist das Verfahren komplizierter:

Der Standardmodus 1 erlaubt die Darstellung von vier Farben, aber es kann immer nur zwei bestimmten Bildpunkten gleichzeitig eine Farbe zugeordnet werden. Die Farbe dieser beiden Punkte wird durch das Farbregister festgelegt, dessen Nummer aus dem Binärwert der beiden Bits für die zwei Bildpunkte gebildet wird.

Im Vielfarbenmodus 0 muß mit 16 möglichen Farben jeweils die Farbe von vier Bildpunkten gleichzeitig angegeben werden. Die Farbe dieser Bildpunktgruppe wird wieder durch den Inhalt des Farbregisters, dessen Nummer von den entsprechenden Bits gebildet wird, bestimmt.

Um das Verständnis dieses Systems zu vereinfachen, zeigt Ihnen Bild 2.5, in welchem Modus welche Bits eines Bildschirmzeilenbytes unabhängig voneinander gesetzt werden können und wie sich die Nummer eines Farbregisters aus diesen Bits zusammensetzt.

Sie sehen, daß jedes Bit in den verschiedenen Modi unterschiedliche Funktionen hat. Wenn Ihnen dieses Verfahren noch Schwierigkeiten bereitet, sollten Sie einige Experimente zu diesem Thema durchführen.

Keine Sorge, durch zielloses POKEn in den Bildschirmspeicher kann der CPC 464 nicht abstürzen. Allerdings sollten Sie nicht in den 48 «freien» Bytes, die nach jedem Bildschirmzeilenblock folgen, Werte abspeichern. An dieser Stelle werden teilweise Informationen über das Bildschirmformat abgelegt, und wenn Sie dort etwas verändern, kann auf dem Bildschirm das Chaos ausbrechen!

2.7.5 Format von BASIC-Programmzeilen

Die Aneinanderreihung von Zeilen eines Programms im Arbeitsspeicher des Computers nennt man Programmtext.

Das Format einer einzelnen BASIC-Zeile im Programmtext sieht folgendermaßen aus:

1. Byte	LSB der Zeilenlänge *n*
2. Byte	MSB der Zeilenlänge *n*
3. Byte	LSB der Zeilennummer
4. Byte	MSB der Zeilennummer
5. Byte	BASIC-Token oder Variable
6. Byte	BASIC-Token, Variable, Text usw. (optional)
⋮	

⋮

$n-1$. Byte BASIC-Token, Variable, Text usw. (optional)
n. Byte 0 für Zeilenende

Wenn diese Zeile am Anfang eines Programms steht, beginnt sie mit dem ersten Byte in der Speicherzelle 368.

Handelt es sich bei dieser Zeile um die letzte eines Programms, beenden drei Nullen in den Speicherzellen $n+1$, $n+2$ und $n+3$ das Programm.

Die Bytes in der Mitte der Programmzeile, die als optional gekennzeichnet sind, erscheinen nur im Speicher, wenn die Zeile mehr als ein einzelnes *BASIC-Token* enthält.

2.7.6 BASIC-Token

Im letzten Abschnitt war unter anderem von *BASIC-Token* die Rede. Vielen Lesern wird sich sicherlich die Frage stellen, was ein BASIC-Token ist.

Wenn ein BASIC-Befehl im Programmtext Buchstabe für Buchstabe abgespeichert werden würde, wäre viel Speicherplatz belegt, eine Verwechslungsgefahr mit Variablennamen gegeben und ein erhöhter Zeitaufwand beim Erkennen des Befehls durch den Interpreter nötig.

Sie sehen, mehrere Gründe sprechen gegen diese Art der Abspeicherung im Arbeitsspeicher.

Es hat sich bei den meisten Computern durchgesetzt, jedem BASIC-Befehl eine bestimmte Codenummer, bestehend aus ein oder zwei Bytes, zuzuordnen.

Dieses Verfahren spart Speicherplatz, Programmlaufzeit und beugt zusätzlich einer Verwechslung mit Variablennamen im Programmtext vor.

Die Codenummern der BASIC-Befehle des CPC 464 bestehen bei normalen Befehlen aus einem Byte mit gesetztem Bit 7. Bei Funktionsbefehlen wird ein Byte mit dem Wert 225 vorangestellt.

Die Tabelle 2.4 zeigt die Token aller BASIC-Befehle des CPC 464.

BASIC-Token	BASIC-Befehl	BASIC-Token	BASIC-Befehl	BASIC-Token	BASIC-Befehl
255+ 0	ABS	255+121	RIGHT$	168	LOAD
255+ 1	ASC	255+122	ROUND	169	LOCATE
255+ 2	ATN	255+123	STRING$	170	MEMORY
255+ 3	CHR$	255+124	TEST	171	MERGE
255+ 4	CINT	255+125	TESTR	172	MID$
255+ 5	COS	255+127	VPOS	173	MODE
255+ 6	CREAL	128	AFTER	174	MOVE
255+ 7	EXP	129	AUTO	175	MOVER
255+ 8	FIX	130	BORDER	176	NEXT
255+ 9	FRE	131	CALL	177	NEW
255+ 10	IMKEY	132	CAT	178	ON
255+ 11	INP	133	CHAIN	179	ON BREAK
255+ 12	INT	134	CLEAR	180	ON ERROR GOTO 0
255+ 13	JOY	135	CLG		
255+ 14	LEN	136	CLOSEIN	181	ON SQ
255+ 15	LOG	137	CLOSEOUT	182	OPENIN
255+ 16	LOG10	138	CLS	183	OPENOUT
255+ 17	LOWER$	139	CONT	184	ORIGIN
255+ 18	PEEK	140	DATA	185	OUT
255+ 19	REMAIN	141	DEF	186	PAPER
255+ 20	SGN	142	DEFINT	187	PEN
255+ 21	SIN	143	DEFREAL	188	PLOT
255+ 22	SPACE$	144	DEFSTR	189	PLOTR
255+ 23	SQ	145	DEG	190	POKE
255+ 24	SQR	146	DELETE	191	PRINT
255+ 25	STR$	147	DIM	192	' (' = REM)
255+ 26	TAN	148	DRAW	193	RAD
255+ 27	UNT	149	DRAWR	194	RANDOMIZE
255+ 28	UPPER$	150	EDIT	195	READ
255+ 29	VAL	151	ELSE	196	RELEASE
255+ 64	EOF	152	END	197	REM
255+ 65	ERR	153	ENT	198	RENUM
255+ 66	HIMEM	154	ENV	199	RESTORE
255+ 67	INKEY$	155	ERASE	200	RESUME
255+ 68	PI	156	ERROR	201	RETURN
255+ 69	RND	157	EVERY	202	RUN
255+ 70	TIME	158	FOR	203	SAVE
255+ 71	XPOS	159	FOR	204	SOUND
255+ 72	YPOS	159	GOSUB	205	SPEED
255+113	BIN$	160	GOTO	206	STOP
255+114	DEC$	161	IF	207	SYMBOL
255+115	HEX$	162	INK	208	TAG
255+116	INSTR	163	INPUT	209	TAGOFF
255+117	LEFT$	164	KEY	210	TROFF
255+118	MAX	165	LET	211	TRON
255+119	MIN	166	LINE	212	WAIT
255+120	POS	167	LIST	213	WEND

BASIC-Token	BASIC-Befehl	BASIC-Token	BASIC-Befehl	BASIC-Token	BASIC-Befehl
214	While	235	TAB	246	*
215	WIDTH	235	THEN	247	/
216	WINDOW	236	TO	248	^
217	WRITE	237	USING	249	\
218	ZONE	238	>	250	AND
	DI	239	=	251	MOD
220	EI	240	>=	252	OR
227	ERL	241	<	253	XOR
228	FN	242	<>	254	NOT
229	SPC	243	<=	255+ *n*	siehe oben!
230	STEP	244	+		
231	SWAP	245	−		

Tabelle 2.4 Die Token der BASIC-Befehle

2.7.7 Format von BASIC-Variablen

Wenn Variablen im Programmtext erscheinen, haben sie ein besonderes Format, um nicht mit BASIC-Token oder anderem Text verwechselt zu werden.

Weil bei den Variablen zwischen drei verschiedenen Typen unterschieden wird, existieren drei geringfügig voneinander abweichende Formate.

Realvariable werden in diesem Format abgespeichert:

13, Länge des Variablennamens +4,0, Buchstabe... Buchstabe +128

Das Format für Integervariable sieht folgendermaßen aus:

2, Länge des Variablennamens +4,0, Buchstabe... Buchstabe +128

Zuletzt das Format für Stringvariable:

3, Länge des Variablennamens +4,0, Buchstabe... Buchstabe +128

Wie Sie sehen, bestimmt ausschließlich das *erste Byte* in den Formaten den Typ der Variablen, und der *letzte Buchstabe* des Variablennamens wird mit gesetztem Bit 7 abgespeichert (der ASCII-Wert wird um 128 erhöht).

Wenn den Variablen ihre Werte erst während des laufenden Programms zugewiesen werden, können diese Werte nicht im Programmtext

Realvariable	Integervariable	Stringvariable
0	0	0
0	0	0
erster Großbuchstabe des Variablennamens	erster Großbuchstabe des Variablennamens	erster Großbuchstabe des Variablennamens
⋮	⋮	⋮
letzter Großbuchstabe des Variablennamens +128	letzter Großbuchstabe des Varaiablennamens +128	letzter Großbuchstabe des Variablennamens +128
4	1	2
Mantisse	LSB des Zahlenwerts	Textlänge
Mantisse	MSB des Zahlenwerts	LSB der Adresse
Mantisse	0	MSB der Adressse
Mantisse	–	0
Exponent	–	–
0	–	–

Tabelle 2.5 Format der Variablentypen in der Variablentabelle

abgespeichert, sondern müssen in einer speziellen Variablentabelle abgelegt werden.

Tabelle 2.5 zeigt die Formate der verschiedenen Variablentypen in der Variablentabelle.

Der Text einer Stringvariable wird nicht in der Variablentabelle, sondern im Stringspeicher abgelegt. Die Adresse in der Variablentabelle gibt an, ab welcher Speicherzelle im Stringspeicher der Text einer Variablen beginnt.

2.7.8 Einlesen unleserlicher Programme

Weil der CPC 464 nicht über den allgemein üblichen BASIC-Befehl *VERIFY* verfügt und somit eine Überprüfung von geSAVEten Programmen *nicht* möglich ist, kann es passieren, daß Ihr Computer ein Programm nicht einlesen kann.

Wenn Sie eine Sicherheitskopie angefertigt haben, ist das nicht weiter tragisch. Sollte es sich bei dem unleserlichen Programm jedoch um Ihre einzige Kopie handeln, ist die Arbeit oder das Geld, das Sie in dieses Programm investiert haben, in der Regel verloren.

Oft ist an einem Lesefehler nicht einmal der CPC 464 schuld, sondern eine fehlerhafte Bandstelle der Kassette (*dropout*), die bei Musikaufnahmen nicht weiter auffällt, bei der Datenspeicherung jedoch schmerz-

liche Datenlücken verursachen kann. (Ein drop-out können Sie daran erkennen, daß beim erneuten Abspeichern des Programms an der gleichen Stelle wieder ein Lesefehler auftritt. Kassetten mit mehreren drop-outs verwenden Sie am besten nur für Musikaufnahmen!)

Was können Sie tun, wenn Ihr Computer ein Programm nicht mehr laden will, sind die – unter Umständen wertvollen – Daten unwiederbringlich verloren?

Zum Glück gibt es eine Methode, mit der Sie einen Teil, unter Umständen sogar das ganze Programm, retten können: Schalten Sie Ihren Computer aus und wieder ein, und laden Sie das unleserliche Programm. Tritt ein *Load error b* auf, haben Sie Glück gehabt, und der CPC 464 lädt trotz gefundenem Lesefehler das komplette Programm in seinen Arbeitsspeicher. Bei einem *Load error a* lädt der Computer das Programm nur bis zur fehlerhaften Stelle, und der Rest fehlt. Manchmal hat man allerdings Glück und erwischt nach mehrmaligen Versuchen den *Load error b*.

Sie haben jetzt einen Teil oder sogar das gesamte Programm im Speicher, können es aber nicht listen. Dies liegt daran, daß der CPC 464 die ersten zwei Speicherzellen des BASIC-Programmbereichs (368 und 369) auf null gesetzt hat. Diese beiden Stellen des BASIC-Programmbereichs zeigen die Zeilenlänge der ersten Zeile eines BASIC-Programms an. Wenn sie auf null liegen, bedeutet das, daß die Länge der ersten Zeile eines Programms null beträgt, die Zeile folglich nicht vorhanden ist. Der BASIC-Interpreter sieht beim Auftreten des Befehls LIST diese Marke, glaubt, daß das Ende des Programms erreicht ist, und listet im Endeffekt gar nichts.

Um das Programm wieder listfähig zu machen, müssen Sie die beiden Speicherzellen mit den ursprünglichen Werten des Programms überschreiben. Aus Abschnitt 2.7.5 wissen Sie folgendes:

Speicherzelle 368 enthält das untere Byte der Zeilenlänge, und Speicherzelle 369 enthält das obere Byte der Zeilenlänge (gewöhnlich null).

Wie Sie sehen, muß in 368 das LSB und in 369 das MSB der Zeilenlänge geladen werden. Weil die erste Zeile eines Programms (fast) nie länger als 255 Byte ist, kann die Speicherzelle 369 unverändert bleiben.

Die Zeilenlänge der ersten Programmzeile finden Sie durch einfaches Ausprobieren heraus. Sie muß mindestens 6 Byte lang sein (siehe Abschnitt 2.7.5), und ihr Wert kann durch weiteres Abschätzen präzisiert werden.

Den vermuteten Wert POKEn Sie in die Speicherzelle 368. Wenn sich jetzt das Programm korrekt listen läßt, haben Sie einen Volltreffer gelandet, andernfalls müssen Sie es weiterprobieren.

Eine Alternative bietet sich, indem Sie mit dem PEEK-Befehl nach der Null für das Ende der ersten Programmzeile suchen. Wenn Sie diese Null gefunden haben (Vorsicht, auch das Variablenformat im Programmtext enthält eine Null!), ziehen Sie von der Speicherzellenadresse der Null 367 ab und laden den erhaltenen Wert in die Zelle 368.

Wenn Sie alles gePOKEt haben, müßte sich das Programm listen lassen. Ansonsten haben Sie einen Fehler gemacht und fangen am besten von vorn an.

Probieren Sie jetzt nicht, das Programm laufen zu lassen! Die Variablentabellenzeiger deuten nämlich noch auf Speicherzelle 370, und wenn Sie das Programm laufen lassen würden, könnten zugewiesene Variablenwerte den Anfang des BASIC-Programms überschreiben, und es würde früher oder später abstürzen.

Bevor Sie das Programm laufen oder editieren lassen, müssen Sie die Variablentabellenzeiger auf das Programmende setzen. Das Programmende finden Sie einfach, indem Sie nach drei aufeinanderfolgenden Nullen suchen. Notieren Sie sich den Inhalt der Speicherzellen 370 bis 378 (werden von der Variablen *i* überschrieben), und geben Sie im Direktmodus Folgendes ein:

```
FOR i=379 TO 43901:IF PEEK(i)=0 AND PEEK(i+1)=0
AND PEEK(i+2)=0 THEN i=i+3:PRINT i ELSE NEXT i
```

Nach einiger Zeit (nicht ungeduldig werden, es kann unter Umständen mehrere Minuten dauern) wird von dieser Befehlsfolge eine Zahl ausgegeben. Dieser Wert gibt den neuen Anfang der Variablentabellen an.

Mit

```
POKE 44476,i-INT(i/256)*256
POKE 44676,INT(i/256)
POKE 44677,i-INT(i/256)*256
POKE 44678,INT(i/256)
```

können Sie die Variablentabellenzeiger korrigieren.

Anschließend müssen Sie die Speicherzellen 370 bis 378 mit den notierten Werten restaurieren, weil dieser Speicherplatz vorübergehend für die Zwischenspeicherung der Variablen *i* benutzt wurde.

Nun kann Ihr Programm wieder laufen und läßt sich editieren.

Wenn Sie einen *load error a* hatten, hat der CPC 464 das Einlesen des Programms beim Auftreten des ersten Lesefehlers abgebrochen, und Sie müssen gezwungenermaßen den Rest des Programms neu erstellen.

War dagegen ein *load error b* aufgetreten, hat Ihr Computer das Programm vollständig geladen, allerdings steckt der Lesefehler noch in irgendeiner Programmzeile.

Durch das zeitliche Auftreten des Fehlers beim Laden können Sie schon ungefähr vermuten, wo er sitzt. Entweder suchen Sie das ganze Programm durch und verbessern die entsprechende Zeile, oder Sie warten beim Ablauf des Programms, daß eine Fehlermeldung auftritt. Wenn der Lesefehler allerdings raffiniert genug war und z.B. lediglich eine Variablenzuweisung geändert hat, können Sie unter Umständen recht lange oder sogar umsonst auf eine Fehlermeldung warten.

Auch wenn dieses Verfahren zum Retten von unleserlichen Programmen nicht ganz einfach ist, erleichtert es Ihnen doch das Verständnis der Firmwareabläufe im CPC 464 und kann Ihnen im Ernstfall eine Menge Zeit ersparen!

3
Die Software des CPC 464

Dieses Kapitel enthält eine Reihe nützlicher vollständig ausgearbeiteter Programme und Routinen sowie Richtlinien zum Programmieren von Spielen.

Lassen Sie sich bitte durch die seitenlangen Listings der Programme nicht abschrecken, denn nur an konkreten Beispielen können die meisten Tips und Tricks zum CPC 464 vermittelt werden. Wenn Sie die Software dieses Kapitels durcharbeiten, werden Sie sehen, daß jedes Programm eine Vielzahl von Kurzroutinen enthält, die Sie bei der Erstellung eigener Programme ebenfalls verwenden können.

Alle Programme erklären die Möglichkeiten ihrer Anwendung selbst. Deshalb ist es nicht notwendig, weitere allgemeine Erläuterungen zu der Software zu geben. Zum Verständnis der einzelnen Routinen eines Programms ist es jedoch nötig, die Funktion aller gedanklich schwer nachvollziehbaren Programmzeilen zu dokumentieren.

Weil die Software dieses Kapitels relativ viel Text ausgibt bzw. hochauflösende Grafik benötigt, wird in allen Programmen der Modus 2 verwendet. Ebenso benutzen alle Programme die deutschen Umlaute. Zum Initialisieren dieser Umlaute auf dem CPC 464 und als Übersicht über das Softwareangebot dieses Kapitels können Sie das Programm des nächsten Abschnitts verwenden.

3.1 Inhaltsverzeichnis

```
100 REM *** INHALTSVERZEICHNIS ***
110 MODE 2:WIDTH 80:MEMORY 43903
120 KEY 128,"LIST ":KEY 138,"EDIT ":KEY
139,"RUN"+CHR$(13):KEY 140,"PRINT#n,"+CH
R$(34)
130 SYMBOL AFTER 91
140 SYMBOL 91,102,24,60,102,102,126,102,
0
150 SYMBOL 92,102,60,102,102,102,102,60,
0
160 SYMBOL 93,102,0,102,102,102,102,60,0
170 SYMBOL 123,102,0,120,12,124,204,118,
0
180 SYMBOL 124,102,0,60,102,102,102,60,0
190 SYMBOL 125,102,0,102,102,102,102,62,
0
200 SYMBOL 126,60,102,102,108,102,102,12
4,96
210 PRINT STRING$(80,"_")
220 PRINT TAB(15);"* * *  I N H A L T S
V E R Z E I C H N I S  * * *"
230 PRINT STRING$(80,"_"):PRINT
240 PRINT"D e m o n s t r a t i o n";TAB
(50);"DEMO";TAB(70);"()":PRINT
250 PRINT"M o n a t s k a l e n d e r";T
AB(50);"MONAT";TAB(70);"()":PRINT
260 PRINT"B i o r h y t h m u s";TAB(50)
;"BIO";TAB(70);"()":PRINT
270 PRINT"F i n a n z b e r e c h n u n
g e n";TAB(50);"FINANZEN";TAB(70);"()":P
RINT
280 PRINT"F u n k t i o n s g r a p h";T
AB(50);"FUNKTION";TAB(70);"()":PRINT
290 PRINT"A b f r a g e p r o g r a m m"
;TAB(50);"ABFRAGE";TAB(70);"()":PRINT
```

```
300 PRINT"A d r e s s e n v e r w a l t
u n g";TAB(50);"ADRESSEN";TAB(70);"()":P
RINT:PRINT
310 PRINT CHR$(7);:PRINT"Bitte spulen Si
e die Kassette bis zu der Bandnummer des
 gewünschten Programms,  und laden Sie d
as betreffende Programm unter der Abkürz
ung in Großbuchstaben !"
320 END
```

100 Programmname
110 Initialisierung
120 Belegung der Funktionstasten
130 Kopieren der Zeichenmatrizen ab dem ASCII-Wert 91 in den RAM
140 Definition des Zeichens Ä
150 Definition des Zeichens Ö
160 Definition des Zeichens Ü
170 Definition des Zeichens ä
180 Definition des Zeichens ö
190 Definition des Zeichens ü
200 Definition des Zeichens ß

210 Ausgabe des Programmnamens
bis
230

240 Ausgabe der Programmnamen, Abkürzungen und Bandnummern
bis
300

310 Textausdruck
320 Programmende

3.2 Demonstrationsprogramm

```
100 REM *** DEMONSTRATIONSPROGRAMM ***
110 MODE 2:CLEAR:DEFINT a-z:RANDOMIZE TI
ME:DEG
120 LOCATE 30,13:PRINT"CHAOS IM COMPUTER
"
200 REM
210 REM *** Einzelne Schüsse ***
220 ENV 1,15,-1,1
230 SOUND 7,0,-5,15,1,0,15
300 REM
310 REM *** Maschinengewehr 1 ***
320 SOUND 1,4095,200,15,0,0,10
330 WHILE SQ(1)>127:WEND
400 REM
410 REM *** Maschinengewehr 2 ***
420 ENV 1,5,-1,1
430 FOR lauf=1 TO 25
440 SOUND 4,0,10,15,1,0,10
450 WHILE SQ(4)>127:WEND
460 NEXT lauf
500 REM
510 REM *** Laserfeuer ***
520 FOR laser=1 TO 5
530 FOR lauf=31 TO 1 STEP -1
540 SOUND 1,0,1,15,0,0,lauf
550 NEXT lauf
560 FOR lauf=1 TO 31
570 SOUND 4,0,1,15,0,0,lauf
580 NEXT lauf
590 NEXT laser
600 REM
610 REM *** Explosion ***
620 FOR lauf=1 TO 31
630 SOUND 4,0,10,15,0,0,lauf
640 NEXT lauf
```

```
700 REM
710 REM *** Sirene ***
720 ENV 1,5,-1,20,5,1,20
730 ENT -1,100,1,1,100,-1,1
740 SOUND 2,60,-3,15,0,1
750 WHILE SQ(4)>127:WEND
800 REM
810 REM *** Martinshorn ***
820 ENV 1,15,1,1,1,0,40,5,-1,1
830 FOR lauf=1 TO 5
840 SOUND 1,190,60,0,1
850 WHILE SQ(1)>127:WEND
860 SOUND 4,142,60,0,1
870 WHILE SQ(4)>127:WEND
880 NEXT lauf
900 REM
910 REM *** Raketenstart ***
920 FOR lauf=31 TO 4 STEP -1
930 SOUND 7,0,50,(lauf-2)/4,0,0,lauf
940 NEXT lauf
1000 REM
1010 REM *** Hubschrauber ***
1020 LOCATE 30,13:PRINT"FLUG IM HUBSCHRA
UBER"
1030 ENV 1,15,1,1
1040 FOR lauf=30 TO 15 STEP -1
1050 SOUND 1,0,lauf,0,1,0,lauf
1060 WHILE SQ(1)>127:WEND
1070 NEXT lauf
1080 SOUND 5,0,-30,0,1,0,15
1090 WHILE SQ(1)>127:WEND
1100 FOR lauf=15 TO 30
1110 SOUND 4,0,lauf,0,1,0,lauf
1120 WHILE SQ(4)>127:WEND
1130 NEXT lauf
1140 ENV 1,15,1,5,15,-1,10
1150 SOUND 4,0,250,0,1,0,31
```

```
2000 REM
2010 REM *** Eisenbahn ***
2020 MODE 1:LOCATE 15,13:PRINT"EISENBAHN
"
2030 kanal=1
2040 FOR lauf=24 TO 6 STEP -2
2050 GOSUB 10000
2060 NEXT lauf
2070 kanal=3
2080 GOSUB 20000
2090 ENV 1,15,1,1,15,-1,2
2100 SOUND 7,0,-20,0,1,0,3
2110 WHILE SQ(2)>127:WEND
2120 kanal=6
2130 GOSUB 20000
2140 kanal=4
2150 FOR lauf=6 TO 16 STEP 2
2160 GOSUB 10000
2170 NEXT lauf
3000 REM
3010 REM *** Muster 1 ***
3020 MODE 2
3030 FOR y=0 TO 399 STEP 5
3040 MOVE 0,y:DRAW 639,399-y
3050 NEXT y
3060 FOR x=0 TO 639 STEP 5
3070 MOVE x,0:DRAW 639-x,399
3080 NEXT x
3090 FOR warten=1 TO 2000:NEXT warten
3100 REM
3110 REM *** Muster 2 ***
3120 CLS:MOVE 0,0
3130 FOR lauf=0 TO 399 STEP 6
3140 DRAW 639,lauf:DRAW 639-lauf,399:DRA
W 0,399-lauf:DRAW lauf,0
3150 NEXT lauf
3160 FOR warten=1 TO 2000:NEXT warten
```

```
3200 REM
3210 REM*** Muster 3 ***
3220 CLS:MOVE 0,0
3230 FOR x=0 TO 639 STEP 8
3240 MOVE 0,0:DRAW x,399
3250 MOVE 639,399:DRAW x,0
3260 MOVE 639,0:DRAW x,399
3270 MOVE 0,399:DRAW x,0
3280 NEXT x
3290 FOR warten=1 TO 2000:NEXT warten
3300 REM
3310 REM *** Sinuskurve ***
3320 CLS:ORIGIN 0,200:DRAW 639,0
3330 FOR lauf=0 TO 720
3340 DRAW 640*lauf/720,199*SIN(lauf)
3350 NEXT lauf
3360 FOR warten=1 TO 2000:NEXT warten
3400 REM
3410 REM *** Kreis ***
3420 CLS:ORIGIN 320,200:MOVE 320,399
3430 LOCATE 38,13:PRINT"KREIS"
3440 FOR lauf=0 TO 360
3450 DRAW 199*SIN(lauf),199*COS(lauf)
3460 NEXT lauf
3470 FOR lauf=0 TO 360
3480 MOVE 0,0
3490 DRAW 199*SIN(lauf),199*COS(lauf)
3500 NEXT lauf
3510 FOR warten=1 TO 2000:NEXT warten
4000 REM
4010 REM *** Strecken ***
4020 CLS:ORIGIN 320,200
4030 x=INT(RND*5+2)
4040 y=INT(RND*5+2)
4050 FOR lauf=1 TO 3600 STEP 6
4060 z=100*COS(lauf)
4070 MOVE z,z
```

```
4080 DRAW 199*COS(lauf/x),199*SIN(lauf/y
)
4090 NEXT lauf
4100 GOTO 4020
10000 REM
10010 REM *** UP Beschleunigen und Verzo
egern ***
10020 ENV 1,15,1,lauf/2,15,-1,lauf/2
10030 SOUND kanal,0,-1,0,1,0,lauf
10040 WHILE SQ(kanal)>127:WEND
10050 RETURN
20000 REM
20010 REM *** UP Konstante Geschwindigke
it ***
20020 ENV 1,15,1,2,15,-1,2
20030 SOUND kanal,0,-5,0,1,0,4
20040 WHILE SQ(2)>127:WEND
20050 RETURN
```

100 Programmname
110 Initialisierung
120 Textausdruck

200 Routine zur Ausgabe von Schüssen
220 Definition der Lautstärkenhüllkurve
230 Fünffache Ausgabe der Hüllkurve über alle drei Kanäle

300 Routine zur Ausgabe von Maschinengewehrfeuer 1
320 Parallele Ausgabe eines tiefen Tons und Rauschens für zwei Sekunden über den linken Kanal
330 Schleife wartet auf das Ende der Geräuschausgabe

400 Routine zur Ausgabe von Maschinengewehrfeuer 2
420 Definition der Lautstärkenhüllkurve
430 Schleifenbeginn für die fünfundzwanzigfache Ausgabe der Hüllkurve
440 Ausgabe eines Rauschens mit Lautstärkenhüllkurve

450 Schleife wartet auf das Ende der Ausgabe
460 Schleifenende für die fünfundzwanzigfache Ausgabe der Hüllkurve

500 Routine zur Ausgabe von Laserfeuer
520 Schleifenbeginn für fünfmal Laserfeuer
530 Schleifenbeginn für das Herunterzählen des Rauschens
540 Ausgabe des Rauschens über den linken Kanal
550 Schleifenende für das Herunterzählen des Rauschens
560 Schleifenbeginn für das Heraufzählen des Rauschens
570 Ausgabe des Rauschens über den rechten Kanal
580 Schleifenende für das Heraufzählen des Rauschens
590 Schleifenende für fünfmal Laserfeuer
600 Routine zur Ausgabe einer Explosion
620 Schleifenbeginn für das Heraufzählen des Rauschens
630 Ausgabe des Rauschens über den rechten Kanal
640 Schleifenende für das Heraufzählen des Rauschens

700 Routine zur Ausgabe einer Sirene
720 Definition der Lautstärkenhüllkurve für das An- und Abschwellen des Sirenentons
730 Definition der Frequenzhüllkurve für das Heulen des Tons
740 Dreifache Ausgabe des Sirenentons über den mittleren Kanal
750 Schleife wartet auf das Ende der Geräuschausgabe über den rechten Kanal (Explosion)

800 Routine zur Ausgabe eines Martinshorns
820 Definition der Lautstärkenhüllkurve für weiche Tonausgabe
830 Schleifenbeginn für die fünffache Ausgabe des Martinshorns
840 Ausgabe des tiefen Tons über den linken Kanal
850 Schleife wartet auf das Ende des Tons
860 Ausgabe des hohen Tons über den rechten Kanal
870 Schleife wartet auf das Ende des Tons
880 Schleifenende für die fünffache Ausgabe des Martinshorns

900 Routine zur Ausgabe eines Raketenstarts
920 Schleifenbeginn für das Herunterzählen des Rauschens und Verminderung der Lautstärke
930 Ausgabe des Rauschens über alle drei Kanäle

940 Schleifenende für das Herunterzählen des Rauschens und Verminderung der Lautstärke

1000 Routine zur Ausgabe eines Hubschrauberflugs
1020 Textausdruck
1030 Definition der Lautstärkenhüllkurve
1040 Schleifenbeginn für Herunterzählen und Verkürzung des Rauschens
1050 Ausgabe des Rauschens über den linken Kanal
1060 Schleife wartet auf das Ende des Rauschens
1070 Schleifenende für Herunterzählen und Verkürzung des Rauschens
1080 dreißigfache Ausgabe des Rauschens über den linken und rechten Kanal
1090 Schleife wartet auf das Ende des Rauschens
1100 Schleifenbeginn für Heraufzählen und Verlängerung des Rauschens
1110 Ausgabe des Rauschens über den rechten Kanal
1120 Schleife wartet auf das Ende des Rauschens
1130 Schleifenende für Heraufzählen und Verlängerung des Rauschens
1140 Definition der Lautstärkenhüllkurve für das Anhalten
1150 Ausgabe des Rauschens über den rechten Kanal

2000 Routine zur Ausgabe einer Eisenbahnfahrt
2020 Textausdruck
2030 Kanalzuweisung «links»
2040 Schleifenbeginn für Herunterzählen und Verkürzung des Rauschens
2050 Aufruf des UPs zum Beschleunigen
2060 Schleifenende für Herunterzählen und Verkürzung des Rauschens
2070 Kanalzuweisung «Mitte links»
2080 Aufruf des UPs «Konstante Geschwindigkeit»
2090 Definition der Lautstärkenhüllkurve
2100 zwanzigfache Ausgabe des Rauschens über alle drei Kanäle
2110 Schleife wartet auf das Ende des Rauschens
2120 Kanalzuweisung «Mitte rechts»
2130 Aufruf des UPs «Konstante Geschwindigkeit»
2140 Kanalzuweisung «rechts»
2150 Schleifenbeginn für Heraufzählen und Verlängerung des Rauschens
2160 Aufruf des UPs zum Verzögern
2170 Schleifenende für Heraufzählen und Verlängerung des Rauschens

```
3000 Routine zur Ausgabe von Muster 1
3020 Löschen des Bildschirms durch Umschalten in den hochauflösenden
     Modus
3030 Schleifenbeginn für die Erhöhung der y-Koordinate
3040 Ausgabe einer Linie schräg über den Bildschirm
3050 Schleifenende für die Erhöhung der y-Koordinate
3060 Schleifenbeginn für die Erhöhung der x-Koordinate
3070 Ausgabe einer Linie schräg über den Bildschirm
3080 Schleifenende für die Erhöhung der x-Koordinate
3090 Warteschleife

3100 Routine zur Ausgabe von Muster 2
3120 Löschen des Bildschirms und Positionierung des Grafikcursors
3130 Schleifenbeginn für den Verschiebefaktor
3140 Ausgabe eines verschobenen Rechtecks
3150 Schleifenende für den Verschiebefaktor
3160 Warteschleife

3200 Routine zur Ausgabe von Muster 3
3220 Löschen des Bildschirms und Positionierung des Grafikcursors
3230 Schleifenbeginn für die Erhöhung der x-Koordinate
3240 Linie beginnt links unten
3250 Linie beginnt rechts oben
3260 Linie beginnt rechts unten
3270 Linie beginnt links oben
3280 Schleifenende für die Erhöhung der x-Koordinate
3290 Warteschleife
3300 Routine zur Ausgabe einer Sinuskurve
3320 Löschen des Bildschirms, Setzen des Koordinatennullpunktes und
     Ausgabe der x-Achse
3330 Schleifenbeginn für zwei volle Sinusschwingungen (720 Grad)
3340 Berechnung der x-Koordinate, daß zwei Schwingungen (720 Grad)
     genau eine Bildschirmbreite (640 Punkte) einnehmen
     Berechnung der y-Koordinate, daß eine Amplitude genau die Hälfte
     der Bildschirmhöhe (200 Punkte) einnimmt
     Ausgabe des Kurventeilstücks
3350 Schleifenende für zwei volle Sinusschwingungen
3360 Warteschleife
```

3400 Routine zur Ausgabe eines Kreises
3420 Löschen des Bildschirms, Setzen des Koordinatennullpunktes und Positionierung des Grafikcursors an den Kreisanfang
3430 Textausdruck
3440 Schleifenbeginn für einen Vollkreis (360 Grad)
3450 Berechnung der x- und y-Koordinaten, daß der Kreis oben und unten bis an den Bildschirmrand reicht, Ausgabe des Kreissegments
3460 Schleifenende für einen Vollkreis
3470 Schleifenende für einen Vollkreis (360 Grad)
3480 Positionierung des Grafikcursors im Mittelpunkt des Kreises
3490 Ausgabe einer Linie vom Kreismittelpunkt zum Kreisrand
3500 Schleifenende für einen Vollkreis
3510 Warteschleife

4000 Routine zur Ausgabe von Strecken
4020 Löschen des Bildschirms und Setzen des Koordinatennullpunktes
4030 Erzeugung eines Zufallswertes zwischen 1 und 7 für die x-Koordinate
4040 Erzeugung eines Zufallswertes zwischen 1 und 7 für die y-Koordinate
4050 Schleifenbeginn für die Anzahl und Position der Strecken
4060 Definition der Streckenanfangskoordinate
4070 Positionierung des Grafikcursors auf der Streckenanfangskoordinate
4080 Berechnung der Streckenendkoordinate
Ausgabe einer Linie zwischen Streckenanfang und Streckenende
4090 Schleifenende für die Anzahl und Position der Strecken
4100 Endlosschleife

10000 UP für Beschleunigen und Verzögern der Eisenbahn
10020 Definition der Lautstärkenhüllkurve
10030 Ausgabe des Rauschens über den zugewiesenen Kanal
10040 Schleife wartet auf das Ende der Geräuschausgabe
10050 Rückkehr in das Hauptprogramm

20000 UP für eine konstante Geschwindigkeit der Eisenbahn
20020 Definition der Lautstärkenhüllkurve
20030 Ausgabe des Rauschens über den zugewiesenen Kanal

20040 Schleife wartet auf das Ende der Geräuschausgabe
20050 Rückkehr in das Hauptprogramm

3.3 Monatskalender

```
100 REM *** MONATSKALENDER ***
110 MODE 2:CLEAR:RESTORE
120 REM
130 REM *** Programmenü ***
140 PRINT STRING$(80,"_")
150 PRINT TAB(20);"* * *  M O N A T S K
A L E N D E R  * * *"
160 PRINT STRING$(80,"_")
170 PRINT:PRINT:PRINT
180 PRINT"Dieses Programm berechnet für
einen beliebigen Monat eines beliebigen
Jahres    die Wochentage."
190 PRINT:PRINT
200 PRINT CHR$(7);:PRINT"Bitte geben Sie
 nun Monatsnummer und Jahreszahl durch K
omma getrennt ein!"
210 INPUT monat,jahr
220 IF monat<1 OR monat>12 THEN PRINT:PR
INT"Der Gregorianische Kalender kennt ab
er nur Monat eins bis zwölf!":GOTO 190
230 IF jahr<100 AND jahr>0 THEN jahr=jah
r+1900:GOTO 250
240 IF jahr<1582 OR jahr<1583 AND monat<
11 THEN PRINT:PRINT"Erst im Oktober 1582
 wurde der Gregorianische Kalender einge
führt. Auf den      4.10.1952 julianisch
er Zeitrechnung folgte direkt der 15.10.
1582 gregorianischerZeitrechnung!":GOTO
190
250 REM
260 REM *** Einlesen von Monatsname und
Monatslänge ***
```

```
270 FOR lauf=1 TO 12
280 READ a$,a
290 IF lauf=monat THEN monat$=a$:monatsl
aenge=a
300 NEXT lauf
310 REM
320 REM *** Schaltjahrkorrektur ***
330 IF jahr MOD 400=0 OR jahr MOD 4=0 AN
D jahr MOD 100<>0 THEN schaltjahr=1
340 IF schaltjahr=1 AND monat=2 THEN mon
atslaenge=29
350 REM
360 REM *** Berechnung der Wochentage **
*
370 tage=jahr*365+INT(jahr/4)-INT(jahr/1
00)+INT(jahr/400)-365
380 FOR lauf=1 TO 12
390 READ monatstage
400 IF lauf=monat THEN tage=tage+monatst
age
410 NEXT lauf
420 IF monat<3 AND schaltjahr=1 THEN tag
e=tage-1
430 wochentag=tage-INT(tage/7)*7
440 IF wochentag=0 THEN wochentag=7
450 REM
460 REM *** Kalenderausdruck ***
470 CLS:LOCATE 30,1:PRINT monat$;jahr
480 FOR spalte=1 TO 73 STEP 12
490 LOCATE spalte,3
500 READ wochentag$:PRINT TAB(spalte);wo
chentag$
510 NEXT spalte
520 FOR reihe=6 TO 21 STEP 3
530 FOR spalte=2 TO 74 STEP 12
540 LOCATE spalte,reihe
550 wochentag=wochentag-1
```

```
560 IF wochentag<1 THEN monatslaenge=mon
atslaenge-1:IF monatslaenge>=0 THEN datu
m=datum+1:PRINT datum
570 NEXT spalte
580 NEXT reihe
590 PRINT:PRINT:PRINT
600 PRINT CHR$(7);:PRINT"Noch ein weiter
er Kalenderausdruck (j/n) ?"
610 a$=INKEY$:IF a$="j"THEN 110 ELSE IF
a$<>"n" THEN 610
620 END
1000 REM
1010 REM *** Kalenderdaten ***
1020 DATA JANUAR,31,FEBRUAR,28,MÄRZ,31,A
PRIL,30,MAI,31,JUNI,30,JULI,31,AUGUST,31
,SEPTEMBER,30,OKTOBER,31,NOVEMBER,30,DEZ
EMBER,31
1030 DATA 1,32,60,91,121,152,182,213,244
,274,305,335
1040 DATA Montag,Dienstag,Mittwoch,Donne
rstag,Freitag,Samstag,Sonntag
```

100 Programmname
110 Initialisierung
140 Ausgabe des Programmnamens
bis
160

170 Textausdruck
bis
200

210 Eingabe von Monat und Jahr
220 Überprüfen des Monats
230 Überprüfen, ob die Jahrhunderte des Jahres fehlen
240 Überprüfen des Jahres

270 Schleifenbeginn für das Einlesen von Monatsname und Monatslänge
280 Einlesen von Monatsnahme und Monatslänge
290 Zuweisung des gesuchten Namens und der gesuchten Länge
300 Schleifenende für das Einlesen von Monatsname und Monatslänge

330 Schaltjahrüberprüfung
340 Im Falle eines Schaltjahrs und gewünschten Monats Februar
Verlängerung der Monatslänge auf 29 Tage
370 Berechnung der vergangenen Tage bis zum gewünschten Jahr

380 Schleifenbeginn für das Einlesen der vergangenen Tage bis zum gewünschten Monat
390 Einlesen der Anzahl der Tage
400 Zuweisung der gesuchten Tage
410 Schleifenende für das Einlesen der vergangenen Tage bis zum gewünschten Monat

420 Korrektur der Tagesanzahl im Falle eines Schaltjahrs und gewünschten Monats Januar oder Februar
430 Berechnung des Rests bei der Division der Tage durch 7
440 Korrektur des Rests für den Sonntag
470 Löschen des Bildschirms und Ausgabe von Jahr und Monat

480 Schleifenbeginn für den Ausdruck der Wochentage
490 Positionierung des Cursors
500 Einlesen und Ausdruck des Wochentags
510 Schleifenende für den Ausdruck der Wochentage

520 Schleifenbeginn für die horizontale Positionierung des Cursors
530 Schleifenbeginn für die vertikale Positionierung des Cursors
540 Positionierung des Cursors
550 Vermindern der auszugebenden Leerpositionen um eins
560 Bei vollzogener Ausgabe aller Leerpositionen Reduzierung der Monatslänge um eins
zusätzlich im Falle einer Monatslänge größer als null Erhöhung des Datums um eins und Ausgabe des Datums
570 Schleifenende für die vertikale Positionierung des Cursors
580 Schleifenende für die horizontale Positionierung des Cursors

590 Textausdruck
bis
600

610 Abfrage der Benutzereingabe und Verzweigung
620 Programmende

3.4 Biorhythmusprogramm

```
100 REM *** BIORHYTHMUS ***
110 MODE 2:CLEAR:RESTORE:DEG
120 REM
130 REM *** Programmenü ***
140 PRINT STRING$(80,"_")
150 PRINT TAB(22);"* * *  B I O R H Y T
H M U S  * * *"
160 PRINT STRING$(80,"_")
170 PRINT:PRINT:PRINT
180 PRINT"Dieses Programm berechnet Ihre
n persönlichen Biorhythmus für ein belie
biges     Datum (nach Ihrer Geburt)."
190 PRINT:PRINT
200 REM
210 REM *** erste Abfrage ***
220 PRINT CHR$(7);:INPUT"Geben Sie bitte
 Ihr Geburtsdatum in der Form Tag, Monat
, Jahr durch Komma ge-   trennt ein ";ge
btag,gebmonat,gebjahr
230 REM
240 REM *** Datenaufbereitung ***
250 monat=gebmonat:jahr=gebjahr
260 GOSUB 1000
270 gebschaltjahr=schaltjahr:gebmonat$=m
onat$
280 IF gebtag<1 OR gebtag>monatslaenge O
R gebmonat<1 OR gebmonat>12 OR gebjahr<1
 OR gebjahr>2000 THEN PRINT"Kennen Sie n
```

```
icht einmal Ihr eigenes Geburtsdatum ?":
GOTO 190
290 IF gebjahr<100 THEN gebjahr=gebjahr+
1900
300 PRINT:PRINT
310 REM
320 REM *** zweite Abfrage ***
330 PRINT CHR$(7);:INPUT"Bitte geben Sie
 nun durch Komma getrennt Monatsnummer u
nd Jahreszahl des Datums ein, dessen Aus
druck Sie wünschen ";monat,jahr
340 REM
350 REM *** Datenaufbereitung ***
360 IF monat<1 OR monat>12 THEN PRINT:PR
INT"Der Gregorianische Kalender kennt ab
er nur Monat eins bis zwölf!":GOTO 300
370 IF jahr<100 THEN jahr=jahr+1900
380 IF jahr<gebjahr OR jahr=gebjahr AND
monat<gebmonat THEN PRINT:PRINT"Eigentli
ch müßte Ihnen klar sein, daß zumindest
ein Leben vor der Geburt sehr  unwahrsc
heinlich ist!":GOTO 300
390 GOSUB 1000
400 REM
410 REM *** Berechnung der Differenz in
Tagen ***
420 tage=jahr*365+INT(jahr/4)-INT(jahr/1
00)+INT(jahr/400)
430 gebtage=gebjahr*365+INT(gebjahr/4)-I
NT(gebjahr/100)+INT(gebjahr/400)
440 FOR lauf=1 TO 12
450 READ monatstage
460 IF lauf=monat THEN tage=tage+monatst
age
470 IF lauf=gebmonat THEN gebtage=gebtag
e+monatstage
480 NEXT lauf
```

```
490 IF monat<3 AND schaltjahr=1 THEN tag
e=tage-1
500 IF gebmonat<3 AND gebschaltjahr=1 TH
EN gebtage=gebtage-1
510 gebtage=gebtage+gebtag
520 differenz=tage-gebtage
530 REM
540 REM *** Bildschirmtext ***
550 CLS:PRINT monat$:PRINT jahr
560 PRINT:PRINT:PRINT"1ste Kurve":PRINT"
physisch"
570 PRINT:PRINT:PRINT"2te Kurve":PRINT"e
motional"
580 PRINT:PRINT:PRINT"3te Kurve":PRINT"g
eistig"
590 PRINT:PRINT:PRINT"Geburtstag:":PRINT
600 IF gebtag=1 OR gebtag=6 OR gebtag>19
 THEN PRINT gebtag;"ster" ELSE PRINT geb
tag;"ter"
610 PRINT gebmonat$:PRINT gebjahr
620 REM
630 REM *** Koordinatenkreuz ***
640 ORIGIN 116,80
650 DRAW 0,320:MOVE -8,150:DRAW 520,150
660 MOVE -8,0:DRAW 520,0:MOVE -8,300:DRA
W 520,300
670 LOCATE 12,2:PRINT"+1":LOCATE 12,11:P
RINT" 0":LOCATE 12,20:PRINT"-1"
680 FOR x=0 TO 496 STEP 16
690 MOVE x,0:DRAW x,-8
700 NEXT x
710 zaehler=1
720 FOR spalte=16 TO 76 STEP 4
730 LOCATE spalte,22:PRINT zaehler:zaehl
er=zaehler+2
740 NEXT spalte
750 REM
```

```
760 REM *** Kurvenausgabe ***
770 ORIGIN 116,230
780 bereich=32
790 FOR laenge=23 TO 33 STEP 5
800 MOVE 0,0
810 index=(differenz-INT(differenz/laeng
e)*laenge)/laenge*360
820 IF jahr=gebjahr AND monat=gebmonat T
HEN bereich=32-gebtag:index=0:start=16*g
ebtag:MOVE start,0
830 FOR lauf=0 TO bereich/laenge*360
840 DRAW lauf*laenge/22.5+start,150*SIN(
lauf+index)
850 NEXT lauf
860 NEXT laenge
870 LOCATE 1,25:PRINT CHR$(7);:PRINT"Noc
h ein weiterer Biorhythmusausdruck (j/n)
 ?"
880 a$=INKEY$:IF a$="j" THEN 100 ELSE IF
 a$<>"n" THEN 880
890 END
1000 REM
1010 REM *** UP Monatslänge und Schaltja
hrtest ***
1020 RESTORE
1030 FOR lauf=1 TO 12
1040 READ a$,a
1050 IF lauf=monat THEN monat$=a$:monats
laenge=a
1060 NEXT lauf
1070 IF jahr MOD 400=0 OR jahr MOD 4=0 A
ND jahr MOD 100<>0 THEN schaltjahr=1
1080 IF schaltjahr=1 AND monat=2 THEN mo
natslaenge=29
1090 RETURN
10000 REM
10010 REM *** Kalenderdaten ***
```

```
10020 DATA JANUAR,31,FEBRUAR,28,MÄRZ,31,
APRIL,30,MAI,31,JUNI,30,JULI,31,AUGUST,3
1,SEPTEMBER,30,OKTOBER,31,NOVEMBER,30,DE
ZEMBER,31
10030 DATA 1,32,60,91,121,152,182,213,24
4,274,305,335
```

100 Programmname
110 Initialisierung

140 Ausgabe des Programmnamens
bis
160

170 Textausdruck
bis
190

220 Eingabe des Geburtsdatums
250 Zuweisung der Variablen *für* das UP
260 Aufruf des UPs
270 Zuweisung der Variablen *von* dem UP
280 Überprüfung des Geburtsdatums
290 Überprüfung, ob die Jahrhunderte des Geburtsjahres fehlen

330 Eingabe des gewünschten Monats und Jahres
360 Überprüfung des Monats
370 Überprüfung, ob die Jahrhunderte des Jahres fehlen
380 Überprüfung, ob der gewünschte Monat vor dem Geburtsmonat liegt
390 Aufruf des UPs

420 Berechnung der vergangenen Tage bis zum gewünschten Jahr
430 Berechnung der vergangenen Tage bis zum Geburtsjahr

440 Schleifenbeginn für das Einlesen der vergangenen Tage bis zum gewünschten Monat und Geburtsmonat
450 Einlesen der Anzahl der Tage
460 Zuweisung der gesuchten Tage für den gewünschten Monat

470 Zuweisung der gesuchten Tage für den Geburtsmonat
480 Schleifenende für das Einlesen der vergangenen Tage bis zum gewünschten Monat und Geburtsmonat

490 Korrektur der Tagesanzahl im Fall eines Schaltjahres und gewünschten Monats Januar oder Februar
500 Korrektur der Tagesanzahl im Fall eines Schaltjahres und Geburtsmonats Januar oder Februar
510 Zuweisung der vergangenen Tage bis zum Geburtstag
520 Berechnung der Differenz zwischen Geburtstag und gewünschtem Monat und Jahr

550 Löschen des Bildschirms und Ausgabe von Monat und Jahr
560 Textausdruck
bis
590

600 Ausgabe des Geburtstags
610 Ausgabe von Geburtsmonat und -jahr

640 Setzen des Koordinatennullpunktes
650 Ausgabe der y- und x-Achse
660 Ausgabe der Begrenzungslinien
670 Ausdruck der y-Achsenbeschriftung

680 Schleifenbeginn für das Setzen der x-Achsenmarkierungen
690 Setzen einer Markierung
700 Schleifenende für das Setzen der x-Achsenmarkierungen

710 Zuweisung des x-Achsenbeschriftungswerts

720 Schleifenbeginn für den Ausdruck der x-Achsenbeschriftung
730 Ausdruck der Beschriftung
Erhöhung des Beschriftungswertes um eins
740 Schleifenende für den Ausdruck der x-Achsenbeschriftung

770 Setzen des Koordinatennullpunktes
780 Zuweisung des Ausgabebereichs

790 Schleifenbeginn für die Ausgabe der drei Kurven
800 Positionierung des Grafikcursors
810 Berechnung des Zustands der Sinuskurve zum Monatsanfang
820 Korrektur des Ausgabebereichs, des Sinuskurvenzustands, Definition des Startpunktes der Sinusschwingung und Neupositionierung des Grafikcursors im Falle der Kurvenausgabe für den Geburtsmonat
830 Schleifenbeginn für die Ausgabe einer Kurve
840 Berechnung der x-Koordinate unter Einbeziehung der Schwingungslänge
Berechnung der y-Koordinate unter Berücksichtigung des Sinuskurvenzustands zum Monatsanfang
Ausgabe des Kurventeilstücks
850 Schleifenende für die Ausgabe einer Kurve

860 Schleifenende für die Ausgabe der drei Kurven

870 Textausdruck
880 Abfrage der Benutzereingabe und Verzweigung
890 Programmende

1000 UP für Monatslänge und Schaltjahrtest
1020 Rücksetzen des DATA-Zeigers

1030 Schleifenbeginn für das Einlesen von Monatsname und Monatslänge
1040 Einlesen von Monatsname und Monatslänge
1050 Zuweisung des gesuchten Namens und der gesuchten Länge
1060 Schleifenende für das Einlesen von Monatsname und Monatslänge

1070 Schaltjahrüberprüfung
1080 Im Falle eines Schaltjahrs und des Monats Februar Verlängerung der Monatslänge auf 29 Tage
1090 Rückkehr in das Hauptprogramm

3.5 Finanzprogramm

```
100 REM *** FINANZBERECHNUNGEN ***
110 MODE 2:CLEAR
120 REM
130 REM *** Programmenü ***
140 PRINT STRING$(80,"_")
150 PRINT TAB(15);"* * *  F I N A N Z B
E R E C H N U N G E N  * * *"
160 PRINT STRING$(80,"_")
170 PRINT:PRINT:PRINT
180 PRINT"Einmalige Einzahlung auf ein K
onto";TAB(60);"(1)":PRINT:PRINT
190 PRINT"Regelmäßige jährliche Einzahlu
ng zu Jahresbeginn";TAB(60);"(2)":PRINT:
PRINT
200 PRINT"Regelmäßige monatliche Einzahl
ung zu Monatsbeginn";TAB(60);"(3)":PRINT
:PRINT
210 PRINT"Tilgungsdauer eines Kredites";
TAB(60);"(4)":PRINT:PRINT
220 PRINT:PRINT"Bitte geben Sie die Numm
er der gewünschten Funktion ein!"
230 GOSUB 3000
240 funktion=a:ON funktion GOSUB 1000,10
00,1000,2000
250 CLS:LOCATE 15,13:PRINT CHR$(7);:PRIN
T"Wünschen Sie weitere Berechnungen (j/n
) ?"
260 a$=INKEY$:IF a$="j" THEN 100 ELSE IF
 a$<>"n" THEN 260
270 END
1000 REM
1010 REM *** Bearbeitung der Funktionen
1-3 ***
1020 CLS:PRINT"Die folgende Berechnung b
enötigt mehrere Variablen."
```

```
1030 PRINT"Eine der Variablen kann ausge
rechnet werden, die restlichen Parameter
 müssen     von Ihnen eingegeben werden."
1040 PRINT"Bitte geben Sie nun an, welch
er Parameter berechnet werden soll!"
1050 PRINT:PRINT:PRINT
1060 IF funktion=1 THEN PRINT"Startkapit
al";:GOTO 1090
1070 IF funktion=2 THEN PRINT"Jährliche
Einzahlung";:GOTO 1090
1080 PRINT"Monatliche Einzahlung";
1090 PRINT TAB(30);"(1)":PRINT:PRINT
1100 PRINT"Zinsen in Prozent";TAB(30);"(
2)":PRINT:PRINT
1110 PRINT"Laufzeit";TAB(30);"(3)":PRINT
:PRINT
1120 PRINT"Endkapital";TAB(30);"(4)":PRI
NT:PRINT
1130 GOSUB 3000
1140 IF (funktion=2 OR funktion=3) AND a
=2 THEN LOCATE 40,11:PRINT"Berechnung le
ider nicht möglich!":GOTO 1220
1150 IF a<>1 THEN LOCATE 40,8:PRINT CHR$
(7);:INPUT kapital ELSE position=8
1160 IF a<>2 THEN LOCATE 40,11:PRINT CHR
$(7);:INPUT zinsen ELSE position=11
1170 IF a<>3 THEN LOCATE 40,14:PRINT CHR
$(7);:INPUT laufzeit ELSE position=14
1180 IF a<>4 THEN LOCATE 40,17:PRINT CHR
$(7);:INPUT endkapital ELSE position=17
1190 IF funktion=1 THEN GOSUB 4000:GOTO
1220
1200 IF funktion=2 THEN GOSUB 5000:GOTO
1220
1210 GOSUB 6000
1220 LOCATE 1,25:PRINT CHR$(7);:PRINT"Wü
nschen Sie eine Wiederholung dieser Funk
```

```
tion (j/n) ?"
1230 a$=INKEY$:IF a$="j" THEN 1000 ELSE
IF a$<>"n" THEN 1230
1240 RETURN
2000 REM
2010 REM *** Bearbeitung der Funktion 4
***
2020 CLS:ZONE 25:PRINT"Kredithöhe","Kred
itzins","Monatsrate"
2030 LOCATE 15,1:PRINT CHR$(7);:INPUT kr
edit:LOCATE 40,1:PRINT CHR$(7);:INPUT zi
ns:LOCATE 65,1:PRINT CHR$(7);:INPUT rate
2040 kredit=INT(kredit*100+0.5)/100:zins
=INT(zins*100+0.5)/100:rate=INT(rate*100
+0.5)/100
2050 PRINT:PRINT:ZONE 15:monat=0
2060 PRINT"Monat","Monatsrate","Kreditzi
nsen","Tilgung","Restschuld":PRINT
2070 monat=monat+1:zinsen=kredit*zins/12
00:tilgung=rate-zinsen:kredit=kredit-til
gung
2080 zinsen=INT(zinsen*100+0.5)/100:tilg
ung=INT(tilgung*100+0.5)/100:kredit=INT(
kredit*100+0.5)/100
2090 PRINT monat,rate,zinsen,tilgung,kre
dit
2100 IF kredit>0 THEN 2070 ELSE PRINT
2110 jahre=INT(monat/12):monate=monat-ja
hre*12
2120 PRINT"Sie brauchen";jahre;:IF jahre
>1 OR jahre=0 THEN PRINT"Jahre";ELSE PRI
NT"Jahr";
2130 PRINT" und";monate;:IF monate>1 OR
monate=0 THEN PRINT"Monate";ELSE PRINT"M
onat";
2140 PRINT" zum Abbezahlen Ihres Kredite
s!":PRINT
```

```
2150 PRINT CHR$(7);:PRINT"Wünschen Sie e
ine Wiederholung dieser Funktion (j/n) ?
"
2160 a$=INKEY$:IF a$="j" THEN 2000 ELSE
IF a$<>"n" THEN 2160
2170 RETURN
3000 REM
3010 REM *** UP Zifferneingabe ***
3020 PRINT CHR$(7);
3030 a$=INKEY$:a=VAL(a$):IF a>0 AND a<5
THEN RETURN ELSE 3030
4000 REM
4010 REM *** Formeln für Funktion 1 ***
4020 IF a=1 THEN kapital=endkapital/((1+
zinsen/100)^laufzeit):ausdruck=kapital:G
OTO 7000
4030 IF a=2 THEN zinsen=100*((endkapital
/kapital)^(1/laufzeit)-1):ausdruck=zinse
n:GOTO 7030
4040 IF a=3 THEN laufzeit=LOG(endkapital
/kapital)/LOG(1+zinsen/100):ausdruck=lau
fzeit:GOTO 7040
4050 endkapital=kapital*(1+zinsen/100)^l
aufzeit:ausdruck=endkapital:GOTO 7050
5000 REM
5010 REM *** Formeln für Funktion 2 ***
5020 IF a=1 THEN kapital=endkapital/(((1
+zinsen/100)^(laufzeit+1)-1)/(zinsen/100
)-1):GOTO 7000
5030 IF a=3 THEN laufzeit=LOG((zinsen/10
0)*(endkapital/kapital)+(1+zinsen/100))/
LOG(1+zinsen/100)-1:GOTO 7040
5040 endkapital=kapital*(((1+zinsen/100)
^(laufzeit+1)-1)/(zinsen/100)-1):GOTO 70
50
6000 REM
6010 REM *** Formeln für Funktion 3 ***
```

```
6020 IF a=1 THEN kapital=endkapital/((12
+78*zinsen/1200)*(((1+zinsen/100)^laufze
it-1)/(zinsen/100))):GOTO 7000
6030 IF a=3 THEN laufzeit=LOG((zinsen/10
0)*(endkapital/(kapital*(12+78*zinsen/12
00)))+1)/LOG(1+zinsen/100):GOTO 7040
6040 endkapital=kapital*(12+78*zinsen/12
00)*(((1+zinsen/100)^laufzeit-1)/(zinsen
/100)):GOTO 7050
7000 REM
7010 REM *** Ergebnisausgabe ***
7020 ergebnis=kapital:GOTO 7060
7030 ergebnis=zinsen:GOTO 7060
7040 ergebnis=laufzeit:GOTO 7060
7050 ergebnis=endkapital
7060 LOCATE 40,position:ergebnis=INT(erg
ebnis*100+0.5)/100:PRINT ergebnis
7070 RETURN
```

100 Programmname
110 Initialisierung

140 Ausgabe des Programmnamens
bis
160

170 Ausdruck des Menütextes
bis
220

230 Aufruf des UPs «Zifferneingabe»
240 Verzweigung zum gewünschten Unterprogramm

250 Löschen des Bildschirms und Textausdruck
260 Abfrage der Benutzereingabe und Verzweigung
270 Programmende

1000 UP für die Funktionen 1 bis 3

```
1020 Ausdruck des Menütextes
bis
1120

1130 Aufruf des UPs «Zifferneingabe»
1140 Textausdruck, wenn für die Berechnung keine Formel existiert
1150 Eingabe der erforderlichen Daten
bis
1180
1190 Aufruf des UPs, das für die gewünschte Funktion zuständig ist
bis
1210

1220 Textausdruck
1230 Abfrage der Benutzereingabe und Verzweigung
1240 Rückkehr in das Hauptmenü

2000 UP für die Funktion 4
2020 Textausdruck
2030 Eingabe der erforderlichen Daten
2040 Runden der Werte
2050 Ausgabe des Tabellenkopfs
bis
2060

2070 Erhöhung des Monats um eins und Berechnung der Werte für diesen
bis  Monat
2080 Berechnung der Werte für diesen Monat
2090 Ausgabe der Werte
2100 Tabellenende bei vollständiger Tilgung der Schuld
2110 Berechnung des Tilgungszeitraums
2120 Ausdruck des Tilgungszeitraums
bis
2140

2150 Textausdruck
2160 Abfrage der Benutzereingabe und Verzweigung
2170 Rückkehr in das Hauptmenü
```

3000 UP für die Zifferneingabe
3030 Abfrage der Ziffer und Rückkehr in das Hauptmenü, falls die Ziffer zwischen 0 und 5 liegt

4000 UP mit den Formeln für die Funktion 1
4020 Auswahl der gewünschten Formel,
bis
4050 Berechnung des gesuchten Wertes und Sprung in die Ergebnisausgabe

5000 UP mit den Formeln für die Funktion 2
5020 Auswahl der gewünschten Formel, Berechnung des gesuchten
bis Wertes und Sprung in die Ergebnisausgabe
5040

6000 UP mit den Formeln für die Funktion 3
6020 Auswahl der gewünschten Formel, Berechnung des gesuchten
bis Wertes und Sprung in die Ergebnisausgabe
6040

7000 Beginn des Ergebnisausgabeteils
7020 Zuweisung des gesuchten Wertes zum Ergebnis
bis
7050
7060 Runden und Ausgabe des Ergebnisses
7070 Rückkehr in das UP für die Funktionen 1 bis 3

3.6 Funktionsgraph

```
100 REM *** FUNKTIONSGRAPH ***
110 MODE 2:CLEAR
120 DEF FN funktion(x)=x^3-x^2+3*x
130 xmini=-10:xmaxi=10
140 REM
150 REM *** Programmenü ***
160 PRINT STRING$(80,"_")
170 PRINT TAB(20);"* * *  F U N K T I O
N S G R A P H  * * *"
```

```
180 PRINT STRING$(80,"_")
190 PRINT:PRINT
200 PRINT"Dieses Programm zeichnet zu ei
ner gegebenen Funktion den Graphen."
210 PRINT"(Asymptoten parallel zur Y-Ach
se werden mit eingezeichnet.)":PRINT
220 PRINT"Folgende Möglichkeiten stehen
zur Auswahl:":PRINT:PRINT
230 PRINT"Neue Funktion eingeben";TAB(30
);"(1)":PRINT
240 PRINT"x-Achsengrenzen festlegen";TAB
(30);"(2)":PRINT
250 PRINT"y-Achsengrenzen festlegen";TAB
(30);"(3)":PRINT
260 PRINT"Funktionsgraph ausgeben";TAB(3
0);"(4)":PRINT:PRINT:PRINT
270 PRINT CHR$(7);:PRINT"Bitte geben Sie
 die gewünschte Nummer ein!"
280 a$=INKEY$:a=VAL(a$):IF a<1 OR a>4 TH
EN 280
290 CLS:ON a GOSUB 1000,2000,3000,4000
300 CLS:IF a=4 THEN LOCATE 12,13:PRINT C
HR$(7);:PRINT"Soll dieses Programm fortg
esetzt werden (j/n) ?" ELSE 150
310 a$=INKEY$:IF a$="j" THEN CLS:GOTO 15
0 ELSE IF a$<>"n" THEN 310
320 END
1000 REM
1010 REM *** Neue Funktion eingeben ***
1020 PRINT"Bitte geben Sie Ihre Funktion
 in folgender Form in Zeile 120 ein:":PR
INT
1030 PRINT"120 DEF FN funktion(x)=x^4+2*
x^3-x ..... etc.":PRINT
1040 PRINT"Anschließend können Sie das P
rogramm mit dem Befehl RUN wieder starte
n.":PRINT
```

```
1050 PRINT"Wenn Sie keine Programmänderu
ng vorgenommen haben, kann das Programm
mit CONT   fortgesetzt werden.":PRINT:PR
INT:STOP
1060 RETURN
2000 REM
2010 REM *** x-Achsengrenzen festlegen *
**
2020 PRINT"Bitte geben Sie die untere un
d obere Grenze der x-Achse ein !":PRINT:
PRINT
2030 PRINT CHR$(7);:INPUT"Untere Grenze
 ";xmini:PRINT
2040 PRINT CHR$(7);:INPUT"Obere Grenze
 ";xmaxi:PRINT
2050 IF xmaxi<=xmini THEN PRINT"Fehlerha
fte Eingabe !":PRINT:GOTO 2020
2060 RETURN
3000 REM
3010 REM *** y-Achsengrenzen festlegen *
**
3020 PRINT"Bitte geben Sie die untere un
d obere Grenze der y-Achse ein !":PRINT:
PRINT
3030 PRINT"(Wenn Sie zweimal Null eingeb
en, berechnet der Computer die Grenzen s
elbst.)":PRINT:PRINT
3040 PRINT CHR$(7);:INPUT"Untere Grenze
 ";ymini:PRINT
3050 PRINT CHR$(7);:INPUT"Obere Grenze
 ";ymaxi:PRINT
3060 IF ymini=0 AND ymaxi=0 THEN 3070 EL
SE IF ymaxi<=ymini THEN PRINT"Fehlerhaft
e Eingabe !":PRINT:GOTO 3020
3070 RETURN
4000 REM
4010 REM *** Achsengrenzen berechnen ***
```

```
4020 xdifferenz=xmaxi-xmini
4030 xschritt=639/xdifferenz
4040 IF ymini<>0 OR ymaxi<>0 THEN 4400 E
LSE flag=1
4200 REM
4210 REM *** Wenn nicht eingegeben, y-Ac
hsengrenzen berechnen ***
4220 LOCATE 10,13:PRINT"Einen Moment - e
s werden die y-Achsengrenzen berechnet !
4230 FOR x=xmini TO xmaxi STEP xschritt^
-1
4240 y=FN funktion(x)
4250 IF ymini>y THEN ymini=y
4260 IF ymaxi<y THEN ymaxi=y
4270 NEXT x
4400 REM
4410 REM *** Koordinatennullpunkt berech
nen ***
4420 ydifferenz=ymaxi-ymini
4430 yschritt=399/ydifferenz
4440 xachse=xmini/-xdifferenz*639
4450 yachse=ymini/-ydifferenz*399
4460 CLS:ORIGIN xachse,yachse
4600 REM
4610 REM *** Koordinatenkreuz zeichnen *
**
4620 MOVE xmini*xschritt,0
4630 DRAW xmaxi*xschritt,0
4640 MOVE 0,ymini*yschritt
4650 DRAW 0,ymaxi*yschritt
4660 ymini=INT(ymini*1000+0.5)/1000
4670 ymaxi=INT(ymaxi*1000+0.5)/1000
4680 ycursor=INT((-yachse+399)/16+1)
4690 IF yachse<0 THEN ycursor=25 ELSE IF
 yachse>399 THEN ycursor=1
4700 LOCATE 1,ycursor:PRINT INT(xmini*10
00+0.5)/1000;
```

```
4710 LOCATE 80-LEN(STR$(INT(xmaxi*1000+0
.5)/1000)),ycursor:PRINT INT(xmaxi*1000+
0.5)/1000;
4720 IF LEN(STR$(ymini))>LEN(STR$(ymaxi)
) THEN laenge=LEN(STR$(ymini)) ELSE laen
ge=LEN(STR$(ymaxi))
4730 xcursor=INT(xachse/8+1)-laenge
4740 IF xcursor<1 THEN xcursor=1 ELSE IF
 xachse>639 THEN xcursor=80-laenge
4750 LOCATE xcursor,1:PRINT ymaxi;
4760 LOCATE xcursor,25:PRINT ymini;
4770 IF flag=1 THEN ymini=0:ymaxi=0:flag
=0
4800 REM
4810 REM *** Funktionsgraph ausgeben ***
4820 yposition=FN funktion(xmini)*yschri
tt
4830 IF yposition>32767 THEN yposition=3
2767 ELSE IF yposition<-32768 THEN yposi
tion=-32768
4840 MOVE xmini*xschritt,yposition
4850 FOR x=xmini TO xmaxi STEP xschritt^
-1
4860 y=FN funktion(x)
4870 yposition=y*yschritt
4880 IF yposition<32768 AND yposition>-3
2769 THEN DRAW x*xschritt,yposition
4890 NEXT x
4900 PRINT CHR$(7);
4910 a$=INKEY$:IF a$="" THEN 4910
4920 RETURN
```

100 Programmname
110 Initialisierung
120 Definition der Funktion
130 Zuweisung der Standardgrenzen für die x-Achse

```
160 Ausgabe des Programmnamens
bis
180
190 Ausdruck des Menütextes
bis
270

280 Abfrage der gewünschten Möglichkeit
290 Löschen des Bildschirms und Verzweigung zum gewünschten
    Unterprogramm
300 Löschen des Bildschirms und Textausdruck
310 Abfrage der Benutzereingabe und Verzweigung
320 Programmende

1000 UP zur Eingabe einer neuen Funktion
1020 Textausdruck und Programmstop
bis
1050
1060 Rückkehr in das Menü

2000 UP zum Festlegen der x-Achsengrenzen
2020 Textausdruck
2030 Eingabe der unteren Grenze
2040 Eingabe der oberen Grenze
2050 Überprüfung auf fehlerhafte Eingabe
2060 Rückkehr in das Menü

3000 UP zum Festlegen der y-Achsengrenzen
3020 Textausdruck
bis
3030
3040 Eingabe der unteren Grenze
3050 Eingabe der oberen Grenze
3060 Überprüfung auf fehlerhafte Eingabe
3070 Rückkehr in das Menü

4000 UP zum Ausgeben des Funktionsgraphen
4020 Berechnung der x-Achsenlänge
```

4030 Berechnung der Schrittgröße für eine x-Einheit
4040 Überprüfung, ob die y-Achsengrenzen definiert wurden oder berechnet werden müssen (*flag* = 1)

4220 Textausdruck

4230 Schleifenbeginn für die Berechnung des kleinsten und größten Wertes von y innerhalb der x-Achsengrenzen
4240 Berechnung von y
4250 Überprüfung, ob der bisher kleinste Wert für y vorliegt
4260 Überprüfung, ob der bisher größte Wert für y vorliegt
4270 Schleifenende für die Berechnung des kleinsten und größten Werts von y innerhalb der x-Achsengrenzen

4420 Berechnung der y-Achsenlänge
4430 Berechnung der Schrittgröße für eine y-Einheit
4440 Berechnung der x-Koordinate des Koordinatennullpunktes
4450 Berechnung der y-Koordinate des Koordinatennullpunktes

4460 Löschen des Bildschirms und Setzen des Koordinatennullpunktes

4620 Positionierung des Grafikcursors
4630 Ausgabe der x-Achse
4640 Positionierung des Grafikcursors
4650 Ausgabe der y-Achse

4660 Runden der unteren y-Achsengrenze
4670 Runden der oberen y-Achsengrenze

4680 Berechnung der vertikalen Position des Cursors
4690 Korrektur der vertikalen Cursorposition, falls die x-Achse außerhalb des Bildschirms liegt
4700 Ausgabe der gerundeten unteren x-Achsengrenze am linken Rand des Bildschirms
4710 Ausgabe der gerundeten oberen x-Achsengrenze am rechten Rand des Bildschirms

4720 Längenberechnung des längeren y-Achsengrenzentexts

4730 Berechnung der horizontalen Position des Cursors
4740 Korrektur der horizontalen Cursorposition, falls die y-Achse außerhalb des Bildschirms liegt
4750 Ausgabe der oberen y-Achsengrenze am oberen Rand des Bildschirms
4760 Ausgabe der unteren y-Achsengrenze am unteren Rand des Bildschirms
4770 Rücksetzen der y-Achsengrenzen, falls sie berechnet wurden (*flag* = 1)

4820 Berechnung der y-Koordinate für die untere x-Achsengrenze
4830 Korrektur der y-Koordinate, falls ihr Wert außerhalb des Integerbereichs liegt, um eine Fehlermeldung zu vermeiden
4840 Positionierung des Grafikcursors

4850 Schleifenbeginn für die Ausgabe des Funktionsgraphen
4860 Berechnung von y
4870 Berechnung der y-Koordinate
4880 Ausgabe des Graphenteilstücks, falls der Wert der y-Koordinate im Integerbereich liegt
4890 Schleifenende für die Ausgabe des Funktionsgraphen

4900 Signalton als "READY"-Meldung
4910 Abfrage der Tastatur, bis eine Taste gedrückt wird
4920 Rückkehr in das Menü

3.7 Abfrageprogramm

```
100 REM *** ABFRAGEPROGRAMM ***
110 MODE 2:CLEAR:RANDOMIZE TIME
120 DIM datei$(1,1000),flag(1000)
130 REM
140 REM *** Programmenü ***
150 PRINT STRING$(80,"_")
160 PRINT TAB(15);"* * *  A B F R A G E
P R O G R A M M  * * *"
170 PRINT STRING$(80,"_")
```

```
180 LOCATE 10,15:PRINT CHR$(7);:PRINT"Wü
nschen Sie Informationen zu diesem Progr
amm (j/n) ?"
190 a$=INKEY$:IF a$="j"THEN GOSUB 10000
ELSE IF a$<>"n" THEN 190
200 MODE 1:PRINT"Wollen Sie:":PRINT:PRIN
T
210 PRINT"Daten eingeben,";TAB(30);"(1)"
:PRINT
220 PRINT"Daten berichtigen,";TAB(30);"(
2)":PRINT
230 PRINT"Daten löschen,";TAB(30);"(3)":
PRINT
240 PRINT"Daten einlesen,";TAB(30);"(4)"
:PRINT
250 PRINT"Daten auslesen,";TAB(30);"(5)"
:PRINT
260 PRINT"abgefragt werden";TAB(30);"(6)
":PRINT
270 PRINT:PRINT"oder aufhören ?";TAB(30)
;"(7)":PRINT CHR$(7);
280 a$=INKEY$:a=VAL(a$):IF a<1 OR a>7 TH
EN 280
290 MODE 2:IF zeiger=0 AND a<>1 AND a<>4
 AND a<>7 THEN LOCATE 20,13:PRINT"Es sin
d keine Daten vorhanden !":FOR lauf=1 TO
 2000:NEXT:GOTO 200
300 ON a GOSUB 1000,2000,3000,4000,5000,
6000,310:GOTO 200
310 LOCATE 25,13:PRINT"Auf Wiedersehen !
"
320 END
1000 REM
1010 REM *** Eingabe von Daten ***
1020 PRINT"Bitte geben Sie die zusammeng
ehörenden Datenfelder immer direkt nache
inander    ein, so daß sie auf dem Bilds
```

```
chirm nebeneinander erscheinen."
1030 PRINT"Drücken Sie bei der Eingabe e
ines Datenpaares versehentlich einmal di
e Taste    ENTER, so wird die Eingabe wi
ederholt."
1040 PRINT"Wenn Sie dagegen zweimal die
Taste ENTER drücken, wird die Dateneinga
be abge-    schlossen.":PRINT
1050 zeiger=zeiger+1:PRINT zeiger;
1060 LOCATE 20,VPOS(#0):PRINT CHR$(7);:I
NPUT a$:LOCATE 50,VPOS(#0)-1:PRINT CHR$(
7);:INPUT b$
1070 IF a$<>"" AND b$<>"" THEN datei$(0,
zeiger)=a$:datei$(1,zeiger)=b$:GOTO 1050
 ELSE IF a$<>"" OR b$<>"" THEN 1060
1080 zeiger=zeiger-1
1090 RETURN
2000 REM
2010 REM *** Berichtigung von Daten ***
2020 PRINT CHR$(7);:INPUT"Bitte geben Si
e das falsche Datenfeld ein ";falsch$:PR
INT:PRINT
2030 FOR datensatz=0 TO 1
2040 FOR datenfeld=1 TO zeiger
2050 IF datei$(datensatz,datenfeld)=fals
ch$ THEN PRINT CHR$(7);:INPUT"Wie lautet
 das korrekte Datenfeld ";richtig$:PRINT
:PRINT:datei$(datensatz,datenfeld)=richt
ig$:GOTO 2090
2060 NEXT datenfeld
2070 NEXT datensatz
2080 PRINT"Das Datenfeld ";falsch$;" exi
stiert nicht !":PRINT:PRINT
2090 PRINT CHR$(7);:PRINT"Wollen Sie wei
tere Daten berichtigen (j/n) ?":PRINT:PR
INT
2100 a$=INKEY$:IF a$="j" THEN 2000 ELSE
```

```
IF a$<>"n" THEN 2100
2110 RETURN
3000 REM
3010 REM *** Loeschen von Daten ***
3020 PRINT CHR$(7);:PRINT"Bitte geben Si
e das zu löschende Datenpaar durch Komma
 getrennt ein !"
3030 INPUT a$,b$:PRINT:PRINT
3040 FOR datenfeld=1 TO zeiger
3050 IF datei$(0,datenfeld)=a$ AND datei
$(1,datenfeld)=b$ OR datei$(0,datenfeld)
=b$ AND datei$(1,datenfeld)=a$ THEN 3080
3060 NEXT datenfeld
3070 PRINT"Das Datenpaar ";a$;", ";b$;"
existiert nicht !":GOTO 3150
3080 FOR lauf=datenfeld TO zeiger-1
3090 datei$(0,datenfeld)=datei$(0,datenf
eld+1)
3100 datei$(1,datenfeld)=datei$(1,datenf
eld+1)
3110 NEXT lauf
3120 zeiger=zeiger-1
3130 PRINT"Das Datenpaar ";a$;", ";b$;"
ist gelöscht !"
3140 IF zeiger=0 THEN FOR warten=1 TO 20
00:NEXT warten:GOTO 3170
3150 PRINT CHR$(7);:PRINT:PRINT:PRINT"Wo
llen Sie weitere Datenpaare löschen (j/n
) ?":PRINT:PRINT
3160 a$=INKEY$:IF a$="j" THEN 3000 ELSE
IF a$<>"n" THEN 3160
3170 RETURN
4000 REM
4010 REM *** Einlesen von Daten ***
4020 PRINT CHR$(7);:INPUT"Wie lautet der
 Name der einzulesenden Datei ";name$:na
me$=UPPER$(name$):PRINT
```

```
4030 PRINT"Spulen Sie bitte die Kassette
 bis zum Anfang der einzulesenden Datei.
":PRINT
4040 OPENIN name$
4050 INPUT#9,zeiger
4060 FOR datensatz=0 TO 1
4070 FOR datenfeld=0 TO zeiger
4080 INPUT#9,datei$(datensatz,datenfeld)
4090 NEXT datenfeld
4100 NEXT datensatz
4110 CLOSEIN
4120 RETURN
5000 REM
5010 REM *** Auslesen von Daten ***
5020 name$=STRING$(16," "):PRINT CHR$(7)
;:INPUT"Wie soll der Name der auszulesen
den Datei lauten ";name$:PRINT
5030 PRINT"Suchen Sie sich bitte auf der
 Kassette einen freien Platz für die Dat
ei.":PRINT
5040 OPENOUT name$
5050 PRINT#9,zeiger
5060 FOR datensatz=0 TO 1
5070 FOR datenfeld=0 TO zeiger
5080 PRINT#9,datei$(datensatz,datenfeld)
5090 NEXT datenfeld
5100 NEXT datensatz
5110 CLOSEOUT
5120 RETURN
6000 REM
6010 REM *** Abfragen der Daten ***
6020 PRINT CHR$(7);:INPUT"Soll der erste
  oder der zweite Datensatz abgefragt we
rden (1/2) ";wahl:wahl=2-wahl:PRINT:PRIN
T
6030 PRINT CHR$(7);:INPUT"Wie oft wollen
 Sie insgesamt abgefragt werden ";anzahl
```

```
:PRINT:PRINT
6040 FOR abfrage=1 TO anzahl
6050 zufall=INT(RND*zeiger+1):IF flag(zu
fall)=1 THEN 6050 ELSE flag(zufall)=1
6060 PRINT:PRINT abfrage;TAB(20);datei$(
wahl,zufall);:LOCATE 60,VPOS(#0):PRINT C
HR$(7);:INPUT antwort$
6070 IF antwort$="" THEN antwort$="error
"
6080 IF antwort$=datei$(1-wahl,zufall) T
HEN PRINT"Richtig !":GOTO 6110
6090 fehler=fehler+1:PRINT"Das war leide
r falsch!"
6100 PRINT"Die richtige Antwort lautet:
";datei$(1-wahl,zufall)
6110 IF abfrage/zeiger=INT(abfrage/zeige
r) THEN GOSUB 20000
6120 NEXT abfrage
6130 PRINT:PRINT:PRINT"Sie wußten bei";a
nzahl;"Abfragen";fehler;"mal nicht die L
ösung!"
6140 GOSUB 20000:FOR warten=1 TO 2000:NE
XT warten
6150 RETURN
10000 REM
10010 REM *** UP Benutzerinformation ***
10020 LOCATE 1,10:PRINT"Dieses Programm
kann Sie beim Lernen unterstützen!"
10030 PRINT"Sie geben zwei Datensätze, d
ie in einer gewissen Beziehung zueinande
r stehen,  ein, und der Computer fragt
Sie ab, indem er ein Feld eines Datensat
zes nennt  und Sie das zugehörige Feld
des anderen Datensatzes eingeben müssen.
":PRINT
10040 PRINT"Bei den Datensätzen kann es
sich um Vokabeln verschiedener Sprachen,
```

```
 Fremdwörterund ihre Definitionen, Forme
ln und ihre Bezeichnungen, Jahreszahlen
und ge-     schichtliche Ereignisse etc.
 handeln.":PRINT
10050 PRINT"Damit Sie Ihre Daten nicht j
edesmal wieder zu Beginn des Programmes
neu eingebenmüssen, können die Datensätz
e auf Band ausgelesen und später ebenso
wieder ein- gelesen werden.":PRINT
10060 PRINT"Bitte drücken Sie die Taste
ENTER, wenn Sie sich alles durchgelesen
haben!":PRINT CHR$(7);
10070 a$=INKEY$:IF a$<>CHR$(13) THEN 100
70 ELSE RETURN
20000 REM
20010 REM *** UP Loeschen aller Abfragep
ositionen ***
20020 FOR index=1 TO zeiger
20030 flag(index)=0
20040 NEXT index
20050 RETURN
```

100 Programmname
110 Initialisierung
120 Dimensionierung der Datenmatrizen
150 Ausgabe des Programmnamens
bis
170
180 Textausdruck
190 Abfrage der Benutzereingabe, eventuell Aufruf des UPs "Benutzer-informationen"
200 Ausgabe des Menütexts
bis
270

280 Abfrage der gewünschten Funktion

290 Wenn Funktionsaufruf wegen fehlender Daten nicht sinnvoll, dann erneute Ausgabe des Menütexts
300 Verzweigung zum gewünschten Unterprogramm
310 Textausdruck
320 Programmende

1000 UP für die Eingabe von Daten
1020 Textausdruck
bis
1040

1050 Erhöhung des Datenzeigers um eins und Ausgabe des Zeigers
1060 Eingabe eines Datenpaares
1070 Überprüfung der Eingabe, eventuell weitere Eingaben oder Neueingabe
1080 Korrektur des Datenzeigers durch Vermindern des Zeigers um eins
1090 Rückkehr in das Menü

2000 UP für die Berichtigung von Daten
2020 Eingabe des falschen Datenfeldes

2030 Schleifenbeginn für den Durchlauf der Datensätze
2040 Schleifenbeginn für den Durchlauf der Datenfelder
2050 Beim Auffinden des falschen Datenfeldes:
Eingabe des richtigen Feldes und Ersetzen des falschen Feldes
Sprung aus der Schleife
2060 Schleifenende für den Durchlauf der Datenfelder
2070 Schleifenende für den Durchlauf der Datensätze

2080 Textausdruck bei erfolgloser Beendigung der Suchschleife und Überspringen der nächsten Programmzeile

2090 Textausdruck
2100 Abfrage der Benutzereingabe und Verzweigung
2110 Rückkehr in das Menü

3000 UP für das Löschen von Daten
3020 Textausdruck
3030 Eingabe des zu löschenden Datenpaars

3040 Schleifenbeginn für den Durchlauf der Datenfelder
3050 Beim Auffinden des zu löschenden Datenpaares:
Sprung aus der Schleife
3060 Schleifenende für den Durchlauf der Datenfelder

3070 Textausdruck bei erfolgloser Beendigung der Suchschleife und Überspringen der folgenden Schleife

3080 Schleifenbeginn für das Nachrücken der Datenpaare hinter dem zu löschenden Datenpaar
3090 Nachrücken eines Datenpaars
bis
3100
3110 Schleifenende für das Nachrücken der Datenpaare hinter dem zu löschenden Datenpaar

3120 Korrektur des Datenzeigers durch Vermindern des Zeigers um eins
3130 Textausdruck
3140 Beendigung des UPs, wenn das letzte Datenpaar gelöscht wurde
3150 Textausdruck
3160 Abfrage der Benutzereingabe und Verzweigung
3170 Rückkehr in das Menü

4000 UP für das Einlesen von Daten
4020 Eingabe des Dateinamens
4030 Textausdruck

4040 Öffnen der Einlesedatei
4050 Einlesen des Datenzeigers

4060 Schleifenbeginn für das Einlesen der Datensätze
4070 Schleifenbeginn für das Einlesen der Datenfelder
4080 Einlesen eines Datenfeldes
4090 Schleifenende für das Einlesen der Datenfelder
4100 Schleifenende für das Einlesen der Datensätze

4110 Schließen der Einlesedatei
4120 Rückkehr in das Menü

5000 UP für das Auslesen von Daten
5020 Eingabe des Dateinamens
5030 Textausdruck

5040 Öffnen der Auslesedatei
5050 Auslesen des Datenzeigers

5060 Schleifenbeginn für das Auslesen der Datensätze
5070 Schleifenbeginn für das Auslesen der Datenfelder
5080 Auslesen eines Datenfeldes
5090 Schleifenende für das Auslesen der Datenfelder
5100 Schleifenende für das Auslesen der Datensätze

5110 Schließen der Auslesedatei

5120 Rückkehr in das Menü

6000 UP für das Abfragen der Daten
6020 Eingabe des abzufragenden Datensatzes
6030 Eingabe der Anzahl der Abfragen

6040 Schleifenbeginn für das Abfragen der Daten
6050 Erzeugung einer Zufallszahl und Überprüfung, ob die Zahl bereits abgefragt wurde (*flag* = 1)
6060 Abfrage und Eingabe eines Datenfeldes
6070 Korrektur bei leerer Eingabe
6080 Überprüfung des eingegebenen Datenfeldes und Textausdruck
bis
6100

6110 Aufruf des UPs "Löschen aller Abfragepositionen", wenn alle Datenfelder genau einmal abgefragt wurden
6120 Schleifenende für das Abfragen der Daten

6130 Ausgabe der Fehleranzahl
6140 Aufruf des UPs "Löschen aller Abfragepositionen" und Warteschleife
6150 Rückkehr in das Menü

10000 UP zur Information des Benutzers
10020 Textausdruck
bis
10060
10070 Rückkehr in das Menü beim Drücken von ENTER

20000 UP zum Löschen aller Abfragepositionen

20020 Schleifenbeginn für das Durchlaufen aller Abfragepositionen
20030 Löschen einer Position (*flag* = 0)
20040 Schleifenende für das Durchlaufen aller Abfragepositionen

20050 Rückkehr in das UP für das Abfragen der Daten

3.8 Adressenverwaltung

```
100 REM *** ADRESSENVERWALTUNG ***
110 CLEAR
120 maxi=100
130 DIM anrede$(maxi),vorname$(maxi),nam
e$(maxi),strasse$(maxi),nummer$(maxi),po
stfach$(maxi),leitzahl$(maxi),ort$(maxi)
,vorwahl$(maxi),telefon$(maxi)
140 REM
150 REM *** Programmenü ***
160 MODE 2
170 PRINT STRING$(80,"_")
180 PRINT TAB(15);"* * *  A D R E S S E
N V E R W A L T U N G  * * *"
190 PRINT STRING$(80,"_")
200 PRINT:PRINT
210 PRINT"Adressen eingeben";TAB(30);"(1
)":PRINT
220 PRINT"Adressen berichtigen";TAB(30);
"(2)":PRINT
```

```
230 PRINT"Adressen löschen";TAB(30);"(3)
";:PRINT
240 PRINT"Adressen einlesen";TAB(30);"(4
)":PRINT
250 PRINT"Adressen auslesen";TAB(30);"(5
)":PRINT
260 PRINT"Adressen ausdrucken";TAB(30);"
(6)":PRINT
270 PRINT#n,"Programm beenden";TAB(30);"
(7)":PRINT:PRINT
280 PRINT CHR$(7);:PRINT"Bitte geben Sie
 die Nummer der gewünschten Funktion ein
!"
290 a$=INKEY$:a=VAL(a$):IF a<1 OR a>7 TH
EN 290
300 IF zeiger=0 AND a<>1 AND a<>4 AND a<
>7 THEN CLS:LOCATE 20,13:PRINT"Es sind k
eine Adressen vorhanden !":FOR warten=1
TO 2000:NEXT warten:GOTO 140
310 ON a GOSUB 1000,2000,3000,4000,5000,
6000,320:GOTO 140
320 CLS:LOCATE 25,13:PRINT"Auf Wiederseh
en !"
330 END
1000 REM
1010 REM *** Adresseneingabe ***
1020 CLS:PRINT"Bitte geben Sie jetzt die
 Adresse ein."
1030 PRINT"Wenn Sie zu einem Punkt keine
 Angaben machen können oder wollen, dann
 drücken   Sie nur die Taste ENTER ohne
Eingabe.
1040 zeiger=zeiger+1:index=zeiger
1050 GOSUB 10000
1060 PRINT CHR$(7);:PRINT"Wünschen Sie e
ine weitere Adresseneingabe (j/n) ?"
1070 a$=INKEY$:IF a$="j" THEN 1000 ELSE
```

```
IF a$<>"n" THEN 1070
1080 RETURN
2000 REM
2010 REM *** Adressenberichtigung ***
2020 CLS:PRINT CHR$(7);:INPUT"Wie heißt
die Person, deren Adresse geändert werde
n soll (Vorname, Nachname)  ";vorname$,n
ame$:PRINT:PRINT
2030 FOR index=1 TO zeiger
2040 IF vorname$=vorname$(index) AND nam
e$=name$(index) THEN 2070
2050 NEXT index
2060 PRINT"Die Adresse von "vorname$;" "
;name$" existiert nicht !":GOTO 2090
2070 PRINT"Die Adresse von "vorname$;" "
;name$" lautet:":GOSUB 20000
2080 PRINT"Bitte geben Sie jetzt die geä
nderte Adresse ein.":GOSUB 10000
2090 PRINT CHR$(7);:PRINT:PRINT"Wollen S
ie weitere Adressen ändern (j/n) ?"
2100 a$=INKEY$:IF a$="j" THEN 2000 ELSE
IF a$<>"n" THEN 2100
2110 RETURN
3000 REM
3010 REM *** Adressenloeschung ***
3020 CLS:PRINT CHR$(7);:INPUT"Wie heißt
die Person, deren Adresse gelöscht werde
n soll (Vorname, Nachname)  ";vorname$,n
ame$:PRINT:PRINT
3030 FOR index=1 TO zeiger
3040 IF vorname$=vorname$(index) AND nam
e$=name$(index) THEN 3070
3050 NEXT index
3060 PRINT"Die Adresse von "vorname$;" "
;name$" existiert nicht !":GOTO 3220
3070 FOR lauf=index TO zeiger-1
3080 anrede$(lauf)=anrede$(lauf+1)
```

```
3090 vorname$(lauf)=vorname$(lauf+1)
3100 name$(lauf)=name$(lauf+1)
3110 strasse$(lauf)=strasse$(lauf+1)
3120 nummer$(lauf)=nummer$(lauf+1)
3130 postfach$(lauf)=postfach$(lauf+1)
3140 leitzahl$(lauf)=leitzahl$(lauf+1)
3150 ort$(lauf)=ort$(lauf+1)
3160 vorwahl$(lauf)=vorwahl$(lauf+1)
3170 telefon$(lauf)=telefon$(lauf+1)
3180 NEXT lauf
3190 zeiger=zeiger-1
3200 PRINT"Die Adresse von "vorname$;" "
;name$" ist gelöscht !"
3210 IF zeiger=0 THEN FOR lauf=1 TO 2000
:NEXT:GOTO 3240
3220 PRINT CHR$(7);:PRINT:PRINT"Wollen S
ie weitere Adressen löschen (j/n) ?"
3230 a$=INKEY$:IF a$="j" THEN 3000 ELSE
IF a$<>"n" THEN 3230
3240 RETURN
4000 REM
4010 REM *** Einlesen der Adressen ***
4020 CLS:PRINT CHR$(7);:PRINT"Spulen Sie
 bitte die Kassette bis zum Anfang der e
inzulesenden Adressen.":PRINT
4030 OPENIN "ADRESSEN"
4040 INPUT#9,zeiger
4050 FOR index=1 TO zeiger
4060 INPUT#9,anrede$(index)
4070 INPUT#9,vorname$(index)
4080 INPUT#9,name$(index)
4090 INPUT#9,strasse$(index)
4100 INPUT#9,nummer$(index)
4110 INPUT#9,postfach$(index)
4120 INPUT#9,leitzahl$(index)
4130 INPUT#9,ort$(index)
4140 INPUT#9,vorwahl$(index)
```

```
4150 INPUT#9,telefon$(index)
4160 NEXT index
4170 CLOSEIN
4180 RETURN
5000 REM
5010 REM *** Auslesen der Adressen ***
5020 CLS:PRINT CHR$(7);:PRINT"Suchen Sie
 sich bitte auf der Kassette einen freie
n Platz für die Adressen.":PRINT
5030 OPENOUT "ADRESSEN"
5040 PRINT#9,zeiger
5050 FOR index=1 TO zeiger
5060 PRINT#9,anrede$(index)
5070 PRINT#9,vorname$(index)
5080 PRINT#9,name$(index)
5090 PRINT#9,strasse$(index)
5100 PRINT#9,nummer$(index)
5110 PRINT#9,postfach$(index)
5120 PRINT#9,leitzahl$(index)
5130 PRINT#9,ort$(index)
5140 PRINT#9,vorwahl$(index)
5150 PRINT#9,telefon$(index)
5160 NEXT index
5170 CLOSEOUT
5180 RETURN
6000 REM
6010 REM *** Adressenausdruck ***
6020 MODE 2:PRINT"Welche Funktion wünsch
en Sie ?":PRINT:PRINT:PRINT
6030 PRINT"Alle Adressen auf dem Bildsch
irm ausdrucken";TAB(60);"(1)":PRINT
6040 PRINT"Alle Adressen über den Drucke
r ausgeben";TAB(60);"(2)":PRINT
6050 PRINT"Ausgewählte Adressen auf dem
Bildschirm ausdrucken";TAB(60);"(3)":PRI
NT
6060 PRINT"Ausgewählte Adressen über den
```

```
 Drucker ausgeben";TAB(60);"(4)":PRINT
6070 PRINT"Eine Adresse auf dem Bildschi
rm ausdrucken";TAB(60);"(5)":PRINT
6080 PRINT"Eine Adresse über den Drucker
 ausgeben";TAB(60);"(6)":PRINT:PRINT:PRI
NT
6090 PRINT CHR$(7);:PRINT"Bitte geben Si
e die gewünschte Nummer ein."
6100 a$=INKEY$:a=VAL(a$):IF a<1 OR a>6 T
HEN 6100
6110 MODE 1:IF a/2=INT(a/2) THEN n=8
6120 ON a GOSUB 7000,7000,8000,8000,9000
,9000
6130 n=0:PRINT"Wünschen Sie weitere Ausd
rucke (j/n) ?"
6140 a$=INKEY$:IF a$="j" THEN 6000 ELSE
IF a$<>"n" THEN 6140
6150 RETURN
7000 REM
7010 REM *** Ausdruck aller Adressen ***
7020 FOR index=1 TO zeiger
7030 GOSUB 20000
7040 NEXT index
7050 RETURN
8000 REM
8010 REM *** Ausdruck ausgewählter Adres
sen ***
8020 PRINT CHR$(7);:PRINT"Sollen Adresse
n eines bestimmten Nachna-mens oder Wohn
ortes ausgedruckt werden ? (Bitte geben
Sie 'N' oder 'W' ein !)":PRINT:PRINT
8030 a$=INKEY$:a$=UPPER$(a$):IF a$="W" T
HEN 8090 ELSE IF a$<>"N" THEN 8030
8040 PRINT CHR$(7);:INPUT"Wie lautet der
 zu suchende Nachname    ";name$
8050 FOR index=1 TO zeiger
8060 IF name$=name$(index) THEN GOSUB 20
```

```
000:a$="gefunden"
8070 NEXT index
8080 GOTO 8130
8090 PRINT CHR$(7);:INPUT"Wie heißt der
aufzufindende Wohnort    ";ort$
8100 FOR index=1 TO zeiger
8110 IF ort$=ort$(index) THEN GOSUB 2000
0:a$="gefunden"
8120 NEXT index
8130 IF a$<>"gefunden" THEN PRINT:PRINT:
PRINT"Es ist keine Adresse vorhanden !":
PRINT
8140 RETURN
9000 REM
9010 REM *** Ausdruck einer Adresse ***
9020 PRINT CHR$(7);:INPUT"Wie heißt die
Person, deren Adresse aus-gegeben werden
 soll (Vorname, Nachname)";vorname$,name
$:PRINT:PRINT
9030 FOR index=1 TO zeiger
9040 IF vorname$=vorname$(index) AND nam
e$=name$(index) THEN GOSUB 20000:GOTO 90
70
9050 NEXT index
9060 PRINT"Die Adresse von "vorname$;" "
;name$:PRINT"existiert nicht !":PRINT
9070 RETURN
10000 REM
10010 REM *** Eingabe einer Adresse ***
10020 PRINT:PRINT
10030 PRINT CHR$(7);:INPUT"Anrede    ";an
rede$(index)
10040 PRINT CHR$(7);:INPUT"Vorname   ";vo
rname$(index)
10050 PRINT CHR$(7);:INPUT"Nachname ";na
me$(index)
10060 PRINT CHR$(7);:INPUT"Straße
```

```
";strasse$(index)
10070 PRINT CHR$(7);:INPUT"Hausnummer
";nummer$(index)
10080 PRINT CHR$(7);:INPUT"Postfach
";postfach$(index)
10090 PRINT CHR$(7);:INPUT"Postleitzahl
";leitzahl$(index)
10100 PRINT CHR$(7);:INPUT"Wohnort
";ort$(index)
10110 PRINT CHR$(7);:INPUT"Telefonvorwah
l ";vorwahl$(index)
10120 PRINT CHR$(7);:INPUT"Telefonnummer
  ";telefon$(index)
10130 PRINT:PRINT
10140 RETURN
20000 REM
20010 REM *** Ausgabe einer Adresse ***
20020 PRINT#n
20030 IF anrede$(index)<>"" THEN PRINT#n
,anrede$(index)
20040 IF vorname$(index)<>"" THEN PRINT#
n,vorname$(index);" ";
20050 IF name$(index)<>"" THEN PRINT#n,n
ame$(index)
20060 IF strasse$(index)<>"" THEN PRINT#
n,strasse$(index);" ";
20070 IF nummer$(index)<>"" THEN PRINT#n
,nummer$(index)
20080 IF postfach$(index)<>"" THEN PRINT
#n,"Postfach ";postfach$(index)
20090 IF leitzahl$(index)<>"" THEN PRINT
#n,leitzahl$(index);" ";
20100 IF ort$(index)<>"" THEN PRINT#n,or
t$(index)
20110 IF vorwahl$(index)<>"" THEN PRINT#
n,"Tel. (";vorwahl$(index);") ";
20120 IF telefon$(index)<>"" THEN PRINT#
```

```
n,telefon$(index)
20130 PRINT#n
20140 RETURN
```

100 Programmname
110 Initialisierung
120 Zuweisung der Adressenkapazität
130 Dimensionierung der Datenmatrizen
170 Ausgabe des Programmnamens
bis
190
200 Ausgabe des Hauptmenütextes
bis
280

290 Abfrage der gewünschten Funktion
300 Wenn Funktionsaufruf wegen fehlenden Adressen nicht sinnvoll, dann erneute Ausgabe des Hauptmenütextes
310 Verzweigung zum gewünschten Unterprogramm
320 Textausdruck
330 Programmende

1000 UP für die Eingabe von Adressen
1020 Textausdruck
bis
1030
1040 Erhöhung des Adressenzeigers um eins und Zuweisung des Datenzeigers für die UP-Variable
1050 Eingabe einer Adresse
1060 Textausdruck
1070 Abfrage der Benutzereingabe und Verzweigung
1080 Rückkehr in das Hauptmenü

2000 UP für die Berichtigung von Adressen
2020 Eingabe des Namens der Person, deren Adresse geändert werden soll
2030 Schleifenbeginn für den Durchlauf der Adressen

2040 Beim Auffinden der zu ändernden Adresse:
Sprung aus der Schleife
2050 Schleifenende für den Durchlauf der Adressen
2060 Textausdruck bei erfolgloser Beendigung der Suchschleife und Überspringen der nächsten Programmzeilen

2070 Ausgabe der zu ändernden Adresse
2080 Eingabe der geänderten Adresse
2090 Textausdruck
2100 Abfrage der Benutzereingabe und Verzweigung
2110 Rückkehr in das Hauptmenü

3000 UP für das Löschen von Adressen
3020 Eingabe des Namens der Person, deren Adresse gelöscht werden soll

3030 Schleifenbeginn für den Durchlauf der Adressen
3040 Beim Auffinden der zu löschenden Adresse:
Sprung aus der Schleife
3050 Schleifenende für den Durchlauf der Adressen

3060 Textausdruck bei erfolgloser Beendigung der Suchschleife und Überspringen der folgenden Schleife

3070 Schleifenbeginn für das Nachrücken der Adressen hinter der zu löschenden Adresse
3080 Nachrücken einer Adresse
bis
3170
3180 Schleifenende für das Nachrücken der Adressen hinter der zu löschenden Adresse

3190 Korrektur des Adressenzeigers durch Vermindern des Zeigers um eins
3200 Textausdruck
3210 Beendigung des UPs, wenn die letzte Adresse gelöscht wurde
3220 Textausdruck
3230 Abfrage der Benutzereingabe und Verzweigung
3240 Rückkehr in das Hauptmenü

```
4000 UP für das Einlesen der Adressen
4020 Textausdruck

4030 Öffnen der Einlesedatei
4040 Einlesen des Adressenzeigers

4050 Schleifenbeginn für das Einlesen der Adressen
4060 Einlesen einer Adresse
bis
4150
4160 Schleifenende für das Einlesen der Adressen

4170 Schließen der Einlesedatei
4180 Rückkehr in das Hauptmenü

5000 UP für das Auslesen der Adressen
5020 Textausdruck

5030 Öffnen der Auslesedatei
5040 Auslesen des Adressenzeigers

5050 Schleifenbeginn für das Auslesen der Adressen
5060 Auslesen einer Adresse
bis
5150
5160 Schleifenende für das Auslesen der Adressen

5170 Schließen der Auslesedatei
5180 Rückkehr in das Hauptmenü

6000 UP zur Ausgabe von Adressen
6020 Ausgabe des Menütexts
bis
6090

6100 Abfrage der gewünschten Funktion
6110 Zuweisung der Ausgabeeinheit
6120 Verzweigung zum gewünschten Unterprogramm
```

6130 Textausdruck
6140 Abfrage der Benutzereingabe und Verzweigung
6150 Rückkehr in das Hauptmenü

7000 UP zum Ausdruck aller Adressen

7020 Schleifenbeginn für das Ausdrucken der Adressen
7030 Ausdruck einer Adresse
7040 Schleifenende für das Ausdrucken der Adressen

7050 Rückkehr in das Menü

8000 UP zum Ausdruck ausgewählter Adressen
8020 Textausdruck
8030 Abfrage der Benutzereingabe und Verzweigung
8040 Eingabe des Nachnamens

8050 Schleifenbeginn für das Durchlaufen der Adressen
8060 Beim Auffinden der gesuchten Adresse:
Ausdruck der Adresse
8070 Schleifenende für das Durchlaufen der Adressen

8080 Überspringen der nächsten Schleife

8090 Eingabe des Wohnorts

8100 Schleifenbeginn für das Durchlaufen der Adressen
8110 Beim Auffinden der gesuchten Adresse:
Ausdruck der Adresse
8120 Schleifenende für das Durchlaufen der Adressen

8130 Textausdruck bei erfolgloser Beendigung der Suchschleifen
8140 Rückkehr in das Menü

9000 UP zum Ausdruck einer Adresse
9020 Eingabe des Namens der Person, deren Adresse ausgedruckt werden soll

```
9030 Schleifenbeginn für das Durchlaufen der Adressen
9040 Beim Auffinden der gesuchten Adresse:
     Ausdruck der Adresse und Sprung aus der Schleife
9050 Schleifenende für das Durchlaufen der Adressen

9060 Textausdruck bei erfolgloser Beendigung der Suchschleife
9070 Rückkehr in das Menü

10000 UP zur Eingabe einer Adresse
10020 Eingabe einer Adresse
bis
10130
10140 Rückkehr in das Hauptprogramm

20000 UP zur Ausgabe einer Adresse
20020 Ausgabe aller vorhandenen Teile einer Adresse
bis
20130
20140 Rückkehr in das Hauptprogramm
```

3.9 Sortierroutinen

Zum Sortieren von großen Datenmengen ist ein Computer ideal geeignet, weil er eine hohe Datenverarbeitungsgeschwindigkeit besitzt und auch bei extrem monotoner Arbeit nicht die Geduld verliert.

Bis zum heutigen Tag sind mehrere hundert Methoden zum Ordnen von Zahlen mit dem Computer ersonnen worden. Die drei ältesten und kürzesten Verfahren werden in diesem Abschnitt vorgestellt.

Alle drei Routinen sortieren ein Zahlenfeld *feld* mit *anzahl* Werten nach der Größe der Zahlen, so daß am Ende die kleinste Zahl an erster Stelle und die größte Zahl an letzter Position des Feldes stehen.

Der einfachste Sortieralgorithmus ist, wie der Name schon sagt, der EASY-SORT.

```
10 REM *** EASY SORT ***
20 FOR lauf=1 TO anzahl
30 minimum=1E+38:position=0
40 FOR index=lauf TO anzahl
50 IF minimum>feld(index) THEN minimum=f
eld(index):position=index
60 NEXT index
70 tausch=feld(lauf):feld(lauf)=feld(pos
ition):feld(position)=tausch
80 NEXT lauf
```

Die äußere Schleife mit der Laufvariablen *lauf* sorgt dafür, daß die innere Schleife mit der Laufvariablen *index* nach jedem Durchlauf eine Position später als vorher mit der Bearbeitung des Zahlenfeldes *feld* beginnt. Auf diese Weise bleiben bereits sortierte Zahlen von der inneren Schleife unberührt.

Bevor die innere Schleife startet, werden die Variablen für diese Schleife initialisiert. Der Variablen *minimum* wird ein Höchstwert zugeordnet, und die Variable *position* wird auf null gesetzt.

Darauf prüft die innere Schleife, ob die Position des Feldes, die durch die aktuelle Laufvariable *index* repräsentiert wird, den kleinsten Wert enthält, der bisher vorgekommen ist. Wenn dies der Fall ist, wird dieser Wert der Variablen *minimum* und seine Position der Variablen *position* zugeordnet.

Wenn die innere Schleife beendet ist, wird das erste Element des Feldes, das von der inneren Schleife im letzten Durchlauf erfaßt wurde, mit dem Element des Feldes, das den kleinsten Wert enthält, ausgetauscht. Auf diese Weise gelangt jeweils die kleinste Zahl an den Feldanfang, und größere Werte werden an das Ende des Feldes gedrängt. Wenn die äußere Schleife komplett durchlaufen wurde, ist das Zahlenfeld *feld* fertig sortiert.

Das nächste Sortierverfahren ist das sogenannte BUBBLE-SORT.

```
10 REM *** BUBBLE SORT ***
20 FOR lauf=2 TO anzahl
30 FOR index=anzahl TO lauf STEP-1
40 IF feld(index-1)>feld(index) THEN tau
```

```
sch=feld(index):feld(index)=feld(index-1
):feld(index-1)=tausch
50 NEXT index
60 NEXT lauf
```

Wieder existieren eine äußere und eine innere Schleife. Diesmal sorgt die äußere Schleife dafür, daß die innere Schleife, die das Zahlenfeld *feld* rückwärts bearbeitet, bei jedem Durchlauf eine Position früher als beim vorigen Durchlauf endet. So ist wieder gewährleistet, daß bereits sortierte Elemente des Feldes von der weiteren Bearbeitung durch die innere Schleife ausgeschlossen bleiben.

In der inneren Schleife wird überprüft, ob der Wert der vorhergehenden Position des Feldes größer ist als der Wert der aktuellen, durch die Laufvariable *index* dargestellten Position. Wenn dieser Vergleich positiv ausfällt, werden die beiden Elemente gegeneinander ausgetauscht.

So gelangt eine kleine Zahl mit jedem Durchlauf der inneren Schleife Schritt für Schritt an den Feldanfang. Sie wandert wie eine Blase zu ihrer endgültigen Position, daher der Name BUBBLE-SORT.

Das Zahlenfeld *feld* ist komplett durchsortiert, wenn die innere und die äußere Schleife beendet sind.

Das letzte Sortierverfahren wird RIPPLE-SORT genannt.

```
110 REM *** RIPPLE SORT ***
20 flag=0
30 FOR index=1 TO anzahl-1
40 IF feld(index)>feld(index+1) THEN tau
sch=feld(index):feld(index)=feld(index+1
):feld(index+1)=tausch:flag=1
50 NEXT index
60 IF flag=1 THEN 20
```

Am Anfang dieser Sortierroutine wird der Merker *flag* auf null gesetzt. Darauf beginnt eine Schleife, die das gesamte Zahlenfeld *feld* durchläuft.

Wenn der Wert der Position des Feldes, die durch die Laufvariable *index* bestimmt ist, größer als der Wert der folgenden Position ist, werden diese beiden Elemente miteinander vertauscht, und der Merker *flag* wird auf eins gesetzt.

Nachdem die Schleife beendet ist, wird überprüft, ob ein Zahlentausch stattgefunden hat, d.h. der Merker *flag* gesetzt ist.

Wenn ein Austausch stattgefunden hat, wird die Schleife erneut durchlaufen. Wieder wird nach Elementen gesucht, die mit ihren Nachbarn ausgetauscht werden müssen.

Erst dann, wenn die Sortierschleife so lange abgearbeitet worden ist, daß ein kompletter Schleifendurchlauf ohne ein einziges Vertauschen erfolgte, ist das Zahlenfeld *feld* vollständig sortiert.

Die Geschwindigkeit einer Sortierroutine ist ihr wesentlichstes Leistungskriterium. Deshalb folgt eine Gegenüberstellung der drei Routinen für das Sortieren von einem Zahlenfeld, das aus 100 zufällig erzeugten neunstelligen Werten besteht:

EASY-SORT: 27,5 Sekunden
BUBBLE-SORT: 60 bis 65 Sekunden
RIPPLE-SORT: 90 bis 110 Sekunden

Wie Sie sehen, ist EASY-SORT nicht nur das einfachste, sondern auch mit Abstand das schnellste Verfahren. Dies gilt allerdings nur für kleine, völlig unsortierte Felder.

EASY-SORT vergleicht stets sämtliche Felder miteinander, deshalb kann auch die Laufzeit relativ präzise angegeben werden.

Bei den anderen beiden Sortierverfahren schwanken die Werte für die Laufzeit dagegen erheblich. Der Grund dafür ist, daß die verwendeten Zahlenfelder durch Zufall mehr oder weniger vorsortiert sind.

Wenn ein Zahlenfeld bereits grob vorsortiert ist, sind das BUBBLE-SORT- und besonders das RIPPLE-SORT-Verfahren im Vorteil:

BUBBLE-SORT erspart sich eine große Anzahl von zeitaufwendigen Vertauschoperationen, und weil sich RIPPLE-SORT für jeden Sortierdurchlauf merkt, ob das Zahlenfeld schon fertig geordnet ist, beendet es seine Tätigkeit sofort, wenn es nichts mehr zu sortieren gibt. EASY-SORT würde auch dann 27,5 Sekunden lang brav sortieren, wenn ein Feld bereits vollständig geordnet wäre.

Neben diesen drei Sortierverfahren existieren viele weitere Methoden. Die leistungsfähigsten Sortierroutinen bestehen jedoch aus einer Kombination von verschiedenen Sortieralgorithmen und sind in ihrer Funktion für gewöhnlich nur schwer verständlich.

Dieses Thema wird in weiterführender Spezialliteratur ausführlich abgehandelt.

3.10 Textverarbeitung

Komfortable Textverarbeitungssysteme können dem Anwender beim Erstellen von Briefen und Manuskripten viel Zeit ersparen. Mit Vorliebe werden Personalcomputer für diese Aufgabe eingesetzt.

Auch der CPC 464 eignet sich wegen seiner Hardwareeigenschaften (professionelle Tastatur, Darstellung von 80 Zeichen je Zeile) für den Einsatz in der Textverarbeitung.

Es ist allerdings nicht sinnvoll, an dieser Stelle ein komplettes Textverarbeitungssystem mit Dokumentation und den notwendigen Anwendungsbeispielen abzudrucken, weil sonst der Umfang dieses Buches gesprengt werden würde. In einem bescheidenen Rahmen ist es jedoch möglich, mit dem CPC 464 auch ohne ein Textverarbeitungsprogramm Texte zu bearbeiten.

Einen Text können Sie auf dem Computer erstellen, indem Sie Programmzeilen mit PRINT-Befehlen verwenden. Durch Einfügen zusätzlicher Programmzeilen können Sie den Text ergänzen und durch Löschen von Zeilen kürzen. Zusätzlich kann jede Programmzeile durch Editieren behandelt und verändert werden. Auf diese Weise ist es möglich, auch längere Texte auf dem Computer zu erstellen und auf Kassette zu speichern. Doch wie bekommen Sie den Text auf den Drucker?

Bei den meisten Homecomputern müssen Sie alle PRINT-Befehle in LPRINT-Befehle umändern, um eine Textausgabe auf dem Drucker zu erreichen. Beim CPC 464 ist das nicht nötig!

Sicherlich ist Ihnen schon aufgefallen, daß das Inhaltsverzeichnisprogramm aus Abschnitt 3.1 eine Funktionstaste mit der Befehlsfolge PRINT♯*n*,' belegt. Diese Befehlsfolge hat den Vorteil, daß sie alle folgenden Zeichen bis zum Ende der Programmzeile sowohl über den Bildschirm als auch über einen angeschlossenen Drucker ausgeben kann. In der Befehlsfolge wird eine Ein-/Ausgabeeinheit genannt, die durch die Variable *n* repräsentiert ist. Wenn Sie *n* nicht definieren, ist der Wert automatisch null, und der dem Befehl folgende Text wird auf den Bildschirm ausgegeben. Weisen Sie dagegen der Variablen *n* den Wert acht zu, wird der Text über den Drucker ausgegeben.

Das Verfahren zum Erstellen von Texten sieht jetzt folgendermaßen aus:

Mit dem Befehl AUTO können Sie vom CPC 464 die Zeilennummern erzeugen lassen, CTRL ENTER gibt die erforderliche Befehlsfolge aus,

und Sie müssen nur noch den Text eingeben. Selbst das Anführungszeichen am Ende der Programmzeile können Sie sich sparen.

Das folgende Listing zeigt einen Text, der auf diese Weise erstellt wurde und mit *n=8* und *GOTO 10* über den Drucker ausgegeben werden kann.

```
AUTO 10
10 PRINT#n,"Demonstrationstext
20 PRINT#n
30 PRINT#n,"Dieses kleine Beispiel soll
Ihnen verdeutlichen, wie auf
40 PRINT#n,"dem CPC 464 auch ohne Textve
rarbeitungssystem längere
50 PRINT#n,"Texte erstellt und über eine
n angeschlossenen Drucker
60 PRINT#n,"ausgegeben werden können.
70 PRINT#n,"Durch Einfügen und Löschen v
on Programmzeilen kann der
80 PRINT#n,"Text ebenso verändert und ko
rrigiert werden  wie durch
90 PRINT#n,"Anwendung der Editiermöglich
keiten des CPC 464.
```

In diesem Beispiel benötigt jede Textzeile genau eine Programmzeile.

Wenn Sie Texte mit 40 oder 80 Zeichen je Zeile schreiben, können Sie es sich noch einfacher machen, weil Sie in diesem Fall im Modus 1 oder 2 auf dem Bildschirm sehen können, welche Worte zusammen in eine Textzeile passen und wo sie getrennt werden müssen. So ist es möglich, in eine Programmzeile 8 bzw. 4 Textzeilen zu packen. Das folgende Beispiel zeigt einen Text mit 40 Zeichen je Zeile:

```
AUTO 100
100 PRINT#n,"Demonstrationstext
110 PRINT#n
120 PRINT#n,"Anhand dieses Beispielprogr
amms können  Sie sehen, daß es auch mögl
ich ist, meh-rere Textzeilen in eine Pro
grammzeile   einzugeben.
```

```
130 PRINT#n,"Dieses Verfahren funktionie
rt allerdingsnur dann, wenn Sie 40 oder
80 Zeichen jeZeile verwenden. In diesem
Format ent-  spricht im Modus 1 bzw. 2 d
ie Textzei-  lenbreite der Bildschirmbre
ite, und Sie wissen genau, wann eine Tex
tzeile zu En-
140 PRINT#n,"de ist  und wo die Zeile ge
trennt werdenmuß.
```

Diese Methode ist sehr einfach, aber wirkungsvoll. Immerhin entstand mit dem vorgestellten Verfahren das Manuskript für dieses Buch.

3.11 Programmieren von Spielen

Zur Zeit werden Homecomputer überwiegend für Videospiele eingesetzt, und weil jedes Spiel mit der Zeit seinen Reiz verliert, besteht ein ständiger Bedarf nach neuen Spielen auf dem Markt. Manch ein Programmierer hat sich mit dem Erstellen von Spielprogrammen schon eine goldene Nase verdient.

Es ist daher nützlich, wenn Sie sich als Computerbesitzer in der Technik des Spieleprogrammierens etwas auskennen. Vielleicht gelingt es gerade Ihnen, das beliebteste Spiel des Jahres zu schreiben und zu vermarkten! (Allerdings kann nicht garantiert werden, daß Sie dabei Millionen verdienen. Wenn Sie Pech haben, wird Ihr Programm weitaus öfter illegal kopiert als gekauft.)

Im Unterschied zu den «seriösen» Programmen wird bei Spielen hauptsächlich mit Grafik gearbeitet. Eine aufwendige Grafik ist fast schon die Garantie für ein gutes Spiel. Deshalb ist es wichtig, daß Sie als Spieleprogrammierer die Grafik Ihres Computers optimal beherrschen und einsetzen können. Nur unter bester Ausnutzung aller Möglichkeiten kann die Konkurrenz von Ihrem Spiel geschlagen werden. (Daß das Thema des Spiels auch interessant sein muß, ist klar!)

Weil die Grafik einen wichtigen Stellenwert in einem Spielprogramm einnimmt, ist ihr ein gesonderter Abschnitt in diesem Kapitel gewidmet.

Ein Spiel sollte – wie jedes andere Programm auch – sorgfältig geplant und strukturiert werden, um das Verständnis zu erleichtern und das

Verändern einzelner Programmteile zu vereinfachen. Es ist sinnvoll, vor der Erstellung ein Spielprogramm in mehrere Hauptblöcke einzuteilen, und zwar in

das Titelbild
die Spielanleitung
das Spielfeld
den Spielablauf
das Spielende

Auch zu diesen Teilen eines Spiels geben Ihnen die folgenden Abschnitte wichtige Informationen und Anreize für eigene Programme.

3.11.1 Auflösung, Farben, Sprites

Auflösung

Um mit einer annehmbaren Grafikauflösung arbeiten zu können, vergessen Sie besser den Vielfarbenmodus 0 und wählen den Standardmodus 1 mit vier Farben. Vier Farben sind für jedes Spiel ausreichend, weil bei jedem Szenenwechsel neue Farben verwendet werden können.

Detailenthusiasten sollten dagegen den hochauflösenden Grafikmodus 2 verwenden. In diesem Modus kann auch die Trikotwerbung der Spieler eines Fußballprogramms noch mühelos entziffert werden.

Doch Spaß beiseite, der Modus 2 eignet sich mit nur zwei möglichen Farben, aber 80 Zeichen je Zeile, eher für mathematische Darstellungen und Textverarbeitung als für Spielprogramme. Ein Spiel mit nur zwei Farben wirkt auf die Dauer fade und leblos, es sei denn, der Spieler wird pausenlos mit neuen grafischen Effekten überrascht.

Farben

Ein Spiel verliert meist wesentlich an Reiz, wenn es für einen Farbmonitor gedacht ist, aber auf einem monochromen Monitor gesehen wird. Das liegt im allgemeinen daran, daß sich der Autor des Programms nicht überlegt hat, wie seine Farbkombinationen auf einem einfarbigen Bildschirm wirken. Fehlender Kontrast zwischen den Farben und die Tatsache, daß zwei völlig unterschiedliche Farben auf dem monochromen Monitor dengleichen Grünton hervorrufen können, sind die Folgen.

Beachten Sie bitte beim Programmieren mit Farben, daß die Farbwerte des CPC 464 nicht wild zusammengewürfelt sind, sondern eine klare Abstufung vom dunkelsten bis zum hellsten Farbton auf einem monochromen Monitor darstellen.

Deshalb ist es sinnvoll, bei im Spiel benachbarten oder konkurrierenden Farben möglichst solche mit einer großen Farbwertdifferenz zu wählen. In der Praxis hat sich eine Minimaldifferenz von *fünf* bewährt (siehe auch Abschnitt 2.4.2).

Wenn Sie alle auf dem CPC 464 möglichen Farben thematisch ordnen, erhalten Sie sechs große Gruppen, die sich intern lediglich um Farbnuancen unterscheiden. Diese Gruppen repräsentieren die Garben Gelb, Rot, Violett (Magenta), Blau, Türkis und Grün. Suchen Sie sich aus jeder Gruppe Ihre Lieblingsfarbe für das Spielprogramm heraus, aber verwenden Sie nicht mehrere Farbtöne einer Farbgruppe gleichzeitig, sonst wirkt das Bild zu langweilig, und unter Umständen wird der Farbunterschied vom Spieler gar nicht wahrgenommen.

Sprites

Ein Sprite ist ein selbstdefiniertes Grafikzeichen von etwa 20×20 Bildpunkten. Das ist ausreichend, um *player-missiles* wie Männchen oder Raumschiffe darzustellen.

Der Vorteil eines Sprites liegt darin, daß es mit einem einzigen Befehl gelöscht und gesetzt werden kann. Weil das Darstellen bewegter Grafik aus einer Folge von Löschen und Setzen eines Objekts mit verschobenen Koordinaten besteht, erleichtert die Sprite-Technik das Programmieren von bewegter Grafik ungemein.

Leider ist es mit dem CPC 464 nicht möglich, Sprites zu definieren, wie man es von anderen Homecomputern mit hochauflösender Grafik gewöhnt ist.

Eine akzeptable Alternative bietet das Erstellen von eigenen Grafikzeichen mit dem Befehl SYMBOL. Nebeneinandergesetzt können diese Zeichen auch größere Objekte darstellen, und mit TAG und TAGOFF ist eine Ausgabe über die Grafikkoordinaten möglich, so daß auch fein abgestufte Bewegungen produziert werden können.

Um auf diese Weise bewegte Grafik zu erzeugen, müssen Sie die Grafikzeichen immer um einen Schritt versetzen und die alte Position mit dem ASCII-Zeichen CHR$(32) löschen. Wenn Sie zusätzlich die ROM-

Routine für den flüssigeren Bewegungsablauf mit CALL 48409 aufrufen, ist die Illusion einer Bewegung perfekt.

Das folgende Programmbeispiel läßt ein Karo, das aus vier Standardgrafikzeichen zusammengesetzt ist, von links nach rechts langsam über den Bildschirm wandern:

```
10 MODE 2:CLEAR:TAG:ORIGIN 0,200
20 a=212:b=213:c=214:d=215
30 CALL 48409:MOVE x,y:PRINT CHR$(b);CHR
$(a);
40 CALL 48409:MOVE x,y+16:PRINT CHR$(c);
CHR$(d);
50 FOR lauf=1 TO 50:NEXT lauf
60 MOVE x,y:PRINT"  ";
70 MOVE x,y+16:PRINT"  ";
80 x=x+4:IF x=644 THEN x=0
90 GOTO 30
```

Die Grafikzeichen werden als Variable definiert, um den Programmablauf zu bescheunigen und ein problemloses Austauschen der Symbole zu ermöglichen.

3.11.2 Titelbild

Den ersten Eindruck eines Spielprogramms vermittelt das Titelbild, das nach dem Laden des Programms auf dem Bildschirm erscheint. Dieses Titelbild entspricht dem Einband eines Buches oder dem Cover einer Schallplatte. Deshalb ist besondere Sorgfalt darauf zu legen, denn Ihr Spiel soll den Benutzer ansprechen und zum Spielen animieren.

Es gibt mehrere Dinge, die zu einem guten Titelbild gehören. Als erstes natürlich der Programmname, der Name des Programmierers und der Copyright-Vermerk. Weiterhin sollten das Datum der Fertigstellung oder die Nummer der Programmversion nicht fehlen, damit der Benutzer sofort sehen kann, ob er schon die neueste Variante Ihres Spiels besitzt.

Mit SYMBOL können Sie Ihren eigenen Zeichensatz definieren und den Text in futuristischen Buchstaben ausdrucken. Noch besser ist es, wenn Sie den Programmnamen groß in dreidimensionalen Schriftzügen anzeigen.

Das ist aber längst noch nicht alles. Stellen Sie sich einen Buchumschlag vor, auf dem nur Titel, Autor und Verlag abgedruckt sind. Der Inhalt des Buchs mag noch so spannend sein, der Umschlag vermittelt jedenfalls Langeweile.

Sie sehen, Grafik gehört zum Titelbild eines Programms dazu, schließlich bestehen die Elemente Ihres Spiels auch nicht (nur) aus Buchstaben.

Am effektvollsten ist eine bewegte Grafik, z.B. ein sich verändernder Bildrahmen oder einige Elemente Ihres Spiels (Raumschiffe usw.), um dem Spieler gleich die Qualität der grafischen Darstellung Ihres Spielprogramms zu demonstrieren.

Jetzt besteht das Titelbild Ihres Spiels schon aus Text und Grafik. Der Benutzer gewinnt durch das Titelbild einen ersten optischen Eindruck Ihres Programms. Fehlt jetzt noch etwas? Ja, und zwar die Musik! Eine Untermalung des Titelbildes mit einer fröhlichen Melodie lockert das Ganze auf und beweist, daß Ihr Spiel nicht nur optisch, sondern auch akustisch ansprechend gestaltet ist. Zusätzlich kann sich Ihre Melodie zum Hit entwickeln. Melodien zu so bekannten Spielen wie Pac-Man bleiben im Ohr und machen für das Spiel Reklame.

Wenn das Titelbild Ihres Spiels raffiniert gestaltet ist, vermarktet sich das Programm fast von selbst!

3.11.3 Spielanleitung

Die Spielanleitung ist der Teil eines Spiels, dem der Programmierer meistens viel zu wenig Beachtung schenkt.

Eine Spielanleitung muß klar verständlich geschrieben sein und alle Varianten eines Spiels erklären, sonst befindet sich der Benutzer auf einmal in einer Situation, die er nicht meistern kann, weil ihm die nötigen Informationen fehlen.

Sie kennen sicherlich die Spielhallenautomaten, die nur über eine dürftige oder gar keine Spielanleitung verfügen. Der Spieler muß sich seine Informationen erfragen oder für teures Geld erspielen. Das wirkt frustrierend und verhilft dem Programm schlecht oder gar nicht zu einer angemessenen Popularität.

Weil in der Regel alle Spiele, außer den Adventures, Punktspiele sind, muß die Spielanleitung genau erklären, für welche Aktion in welcher Situation dem Spieler wieviel Punkte auf sein Konto gutgeschrieben werden. Zweckmäßigerweise werden zu zerstörende oder aufzufindende

Objekte neben der zugehörigen Punktzahl abgebildet, um die Anleitung für den Benutzer übersichtlicher zu gestalten.

Der Sinn Ihres Spiels muß deutlich gemacht werden, damit der Benutzer weiß, auf welches Ziel er hinarbeiten muß. Das bloße Sammeln von Punkten sollte dabei *nicht* im Vordergrund stehen. In anspruchsvollen Spielen muß der Benutzer von Szene zu Szene schwierigere Aufgaben lösen, um die jeweils nächste Spielsequenz zu erreichen.

Zusätzlich muß erklärt werden, wie das player-missile, das den Spieler auf dem Spielfeld repräsentiert, gesteuert werden kann, z. B. mit bestimmten Buchstaben der Tastatur oder mit einem angeschlossenen Joystick.

Erst dann, wenn Ihre Spielanleitung all diese Informationen enthält, kann sie als vollständig bezeichnet werden.

3.11.4 Spielfeld

Während des Spielablaufs wird das Spielfeld bzw. werden die Spielfelder abgebildet. Besonders dann, wenn Ihr Spiel nur über ein einziges Spielfeld verfügt, der Hintergrund folglich stationär ist, müssen Sie Wert auf ein interessantes Feld legen, damit der Spieler nicht zu schnell gelangweilt wird. Vielleicht überlassen Sie die Detailplanung des Spielfeldes dem Zufall, so daß es in jedem Durchgang in etwas anderer Form erscheint (sofern dies bei dem Thema Ihres Spiels möglich ist).

Wenn das Spielfeld Ihres Spiels dagegen beweglich ist oder wenn es öfters gewechselt wird, brauchen Sie sich hinsichtlich dieses Punktes nicht ganz so viel Mühe zu geben.

Häufig besteht ein Spielfeld aus zwei Teilen, einer Landschaftsszene und einer Anzeigeeinheit. Der erste Teil zeigt dem Benutzer an, wo er sich momentan befindet, und der zweite Teil gibt den aktuellen Zustand des Spielers (z. B. Sauerstoffvorrat) oder seines Fortbewegungsmittels (z. B. Geschwindigkeit) an. Die Anzeigeeinheit muß die Daten möglichst in grafischer Form darstellen, um einen schnelleren Überblick zu gewährleisten.

Nur wirklich wichtige Informationen – wie Restmunition oder Treibstoff – dürfen angezeigt werden, sonst wird der Spieler durch die vielen Anzeigen eher verwirrt als aufgeklärt.

3.11.5 Ablauf des Spiels

Der wichtigste Teil eines Spiels ist der Spielablauf. Dazu kann eigentlich am wenigsten gesagt werden, weil dieser Punkt am variabelsten ist und das Thema des Spiels Ihrer Phantasie überlassen bleiben muß.

Nur ein paar grundlegende Informationen sollen an dieser Stelle folgen:

Es hat sich mittlerweile bei Videospielen durchgesetzt, daß parallel zur Handlung eine Melodie läuft, die charakteristisch für das Spielgeschehen ist. Wenn Objekte eines Spiels kollidieren, kommt es im allgemeinen zur Zerstörung oder Auslöschung mindestens eines dieser Elemente. Diese Aktion wird ebenfalls von einem speziellen Ton, Geräusch oder einer anderen Melodie begleitet.

Die parallele Darstellung von Bild und Ton fällt mit dem CPC 464 leicht. Der Befehl ON SQ(*kanal*) GOSUB springt nur dann ein Unterprogramm an, wenn die Tonwarteschlange von Kanal *kanal* vollständig geleert ist. So ist es möglich, immer rechtzeitig ein paar Töne nachzuschieben, damit die Melodie während grafischer Operationen nicht abbricht. Das Nachladen geschieht dabei so schnell, daß der Spieler nichts davon merkt.

Eine laufende Melodie kann nur mit dem Befehl CLEAR abgebrochen werden. Das ist in einem Programm zweifellos eine sehr ungeeignete Methode, weil alle anderen Daten mit gelöscht werden würden. Als einzige Alternative bietet sich an, über den entsprechenden Kanal einen Ton der Lautstärke null mit gesetztem Flush-Bit auszugehen. Das gesetzte Flush-Bit läßt den CPC 464 die laufende Tonausgabe unterbrechen und den aktuellen Ton ausgeben. Weil dieser wegen der angegebenen Lautstärke nicht hörbar ist, entspricht das Verfahren im Prinzip einem Melodiestopp.

Meistens folgt jedoch auf das Ende einer Melodie direkt ein anderes Geräusch, so daß der folgende SOUND-Befehl den Ton nur über den gleichen Kanal wie der vorhergehende SOUND-Befehl mit gesetztem Flush-Bit ausgeben muß, um die alte Melodie abzulösen.

3.11.6 Ende des Spiels

Ihr Spiel muß von einem interessanten Ausklang beendet werden, denn auch das Ende eines Spiels sollte den Benutzer fesseln. Jeder effektvolle Schluß ist erlaubt, nur das Einblenden eines lapidaren *GAME OVER* sollten Sie sich verkneifen, denn davon wird der Spieler oft genug gelangweilt.

Eine Bewertung der spielerischen Leistung in Form eines kurzen Kommentars sollte in ihrem Spielende nicht fehlen. Das erfreut den Spieler und bei passender Wortwahl – besonders im Fall eines schlechten Abschneidens – auch die Mitspieler!

Eine Anzeige des Endpunktstands ist ebenso selbstverständlich wie das obligatorische Auflisten der 10 bis 20 besten Spieler. Dazu kommt die Möglichkeit, seinen Namen in diese Liste eintragen zu können, sofern im letzten Spiel die entsprechende Leistung erbracht wurde. (Jedes Spiel ist ein Konkurrenzkampf gegen Computer oder Mitspieler, und jeder Spieler möchte bei einem guten Ergebnis auch eine gewisse Anerkennung erfahren!)

Ihre eigene absolute Höchstpunktzahl muß natürlich fest im Programm installiert und angezeigt werden, denn nichts ist erhebender, als den besten Punktestand des Programmierers eines Spiels zu überbieten. Hier gilt nämlich das Motto: Wenn man schon nicht besser als der Programmierer Spiele schreiben kann, dann will man sie wenigstens besser spielen können!

Das Allerletzte Ihres Spiels ist die Aufforderung an den Benutzer, einen erneuten Versuch zu starten, möglichst mit einem Kommentar wie: «Einmal muß es ja klappen!», um seinen Ergeiz anzuspornen.

Schlußwort

Mit diesem Kapitel wird nicht nur das Spieleprogrammieren, sondern auch dieses Buch beendet. Deshalb zum Schluß folgender Tip:

Jeder fleißige Spieler weiß, wie vergnüglich und unterhaltsam witzige Kommentare in einem Spielprogramm sein können, besonders dann, wenn dem Programm verschiedene, der Situation angepaßte Bemerkungen zur Auswahl stehen. Diese Kommentare zum Spielgeschehen sind für ein Spiel das Salz in der Suppe! Vor allem bei der Bewertung einzelner Höchst- und Niedrigstleistungen während des Spielablaufs und bei der abschließenden Bewertung am Spielende sind flotte Sprüche beliebt und ideal einzusetzen.

Setzen Sie deswegen öfters lustige Bemerkungen in Ihrem Spiel ein, um dessen Ablauf aufzulockern. Einige Beispiele:

«Sie hätten besser Kamikazeflieger statt Pilot werden sollen!»
«Na ja, ein blindes Huhn findet auch mal ein Korn.»
«Wenn Sie mit geschlossenen Augen spielen würden, könnten Sie auch nicht schlechter werden!»
«Haben Sie sich schon mal beim Zirkus als Clown beworben?»
«Bald sind Sie besser als ich!»

Mit etwas Fantasie und Geschick können sie an geeigneter Stelle auch grafische Effekte und Gags in Ihre Programme einbauen, z.B. ein für Hamburger werbendes Ungeheuer oder einen Teufel, der die Seele des verunglückten Spielers holt.

Die Hauptsache ist, daß es Ihnen immer wieder gelingt, Witz in Ihr Spiel zu bringen, damit es für den Benutzer nicht langweilig wird. Natürlich erfordert dieser Programmierstil mehr Zeit und Speicherplatz als die übliche 08/15-Programmerstellung.

Er wird sich jedoch schnell durch Beliebtheit und häufigen Gebrauch Ihres Spiels auszahlen.

Wenn man sich von Ihrem Spiel nicht mehr losreißen kann und deswegen Tag und Nacht vor dem Computer verbringt, dann wissen Sie, daß Sie einen Bestseller geschrieben haben!

Zum kreativen Umgang mit der Hardware des CPC 464 lädt der vom selben Autor geplante Folgeband ein:

Hardware-Erweiterungen für den CPC 464 selbstgebaut

Stichwortverzeichnis

VOGEL-BUCHVERLAG WÜRZBURG

Peschetz, Johann/
Peschetz, Alma J.

Was der Atari alles kann

Reihe HC –
Mein Home-Computer

Bd. 1: 236 S., 52 Abb.
35,– DM, 1985
ISBN 3-8023-**0795**-X

Bd. 2: 240 S., 47 Abb.
35,– DM, 1985
ISBN 3-8023-**0796**-8

Wer Atari-BASIC kennt, findet in diesen Büchern eine Brücke zwischen hoher Theorie und praxisbezogener Anwendung. So wird denn auch nichts ausgelassen: Einstieg mit Musik, Mathematische Grundlagen, Grafikmöglichkeiten des Atari, Utilities (Hilfsprogramme), viele Spiele und Organisationshinweise machen diese Bücher beim täglichen Umgang mit dem Atari so wertvoll.

Hettinger, Andreas
Heinz, Andreas

Start mit Atari-BASIC

Reihe HC –
Mein Home-Computer

184 Seiten,
10 Abbildungen,
30,– DM, 1985
ISBN 3-8023-**0827**-1

Durch handliche Programme und Übungen erlernen Sie die nur scheinbar so komplizierte Programmiersprache Atari-BASIC gewissermaßen spielend und werden — nach intensiver Beschäftigung mit dem Inhalt des Buches — in der Lage sein, selbst Programme zu schreiben. Als Anregung für kreatives Denken finden Sie eine Anzahl lauffähiger Programme für alle Atari 400, 600 XL, 800 und 800 XL.

James/Gee/
Ewbank

Das Atari-Spielebuch für 600 XL/800 XL

Reihe HC –
Mein Home-Computer

184 Seiten,
21 Abbildungen,
21 Spielprogramme,
30,– DM, 1984
ISBN 3-8023-**0788**-7

21 Spiele voller Spannung, Action und bewegter Grafik — speziell für den Atari 600/800 XL geschrieben — warten nur darauf, gestartet zu werden. Anhänger bewegter Grafik — Anfänger wie Fortgeschrittene — kommen voll auf ihre Kosten: Jeder kann diese Programme analysieren und verstehen —, sieht, welche raffinierten Programmiertechniken die außergewöhnlichen Fähigkeiten des Atari ausnützen.

Schneller erfolgreich durch Computer-Bücher

VOGEL-BUCHVERLAG WÜRZBURG

Logo ist die Programmiersprache für Home- und Personal-Computer. Benutzerfreundlichkeit, Leistungsfähigkeit und Vielseitigkeit machen sie vor allem bei Anfängern sehr beliebt.

Senftleben, D.
Start mit Atari-Logo
Reihe HC – Mein Home-Computer
220 Seiten, 70 Abbildungen, 30,– DM, 1984
ISBN 3-8023-**0794**-1

In dieser Einführung wird mit Grafik, Text und Musik gespielt, gearbeitet, experimentiert. Mittels Schildkrötengrafik wird das kleine Logo-Einmaleins in 12 Lektionen entwickelt. Große Bildschirmfotos machen die Lektionen anschaulich und regen zur Mitarbeit an. Dank des bausteinorientierten Konzepts kann jeder seine eigenen Teilbausteine erzeugen und sie zu neuen größeren Blöcken zusammenfügen.

Tauber, Michael J.
Spiel und Aktion mit Atari-Logo
Reihe HC – Mein Home-Computer
ca. 160 Seiten, ca. 28,– DM, 1985
ISBN 3-8023-**0842**-5

erscheint voraussichtlich im April 1985

Dieses Buch baut auf den Grundlagen von „Senftleben, Start mit Atari-Logo" auf. Es wird die Programmierung einfacher Spielabläufe gezeigt. Etwa, wie man die Turtle in eine andauernde Bewegung versetzt und die Geschwindigkeit und die Richtung der Turtle mit Tasten, Joystick oder Paddle steuert. Das sind Ausgangspunkte für Geschicklichkeitsspiele, wie sie hier u.a. geboten werden.

Moll, Gerhard
Informatik mit Logo für junge Leute
Reihe HC – Mein Home-Computer
ca. 120 Seiten, zahlreiche Listings, ca. 25,– DM, 1985
ISBN 3-8023-**0807**-7

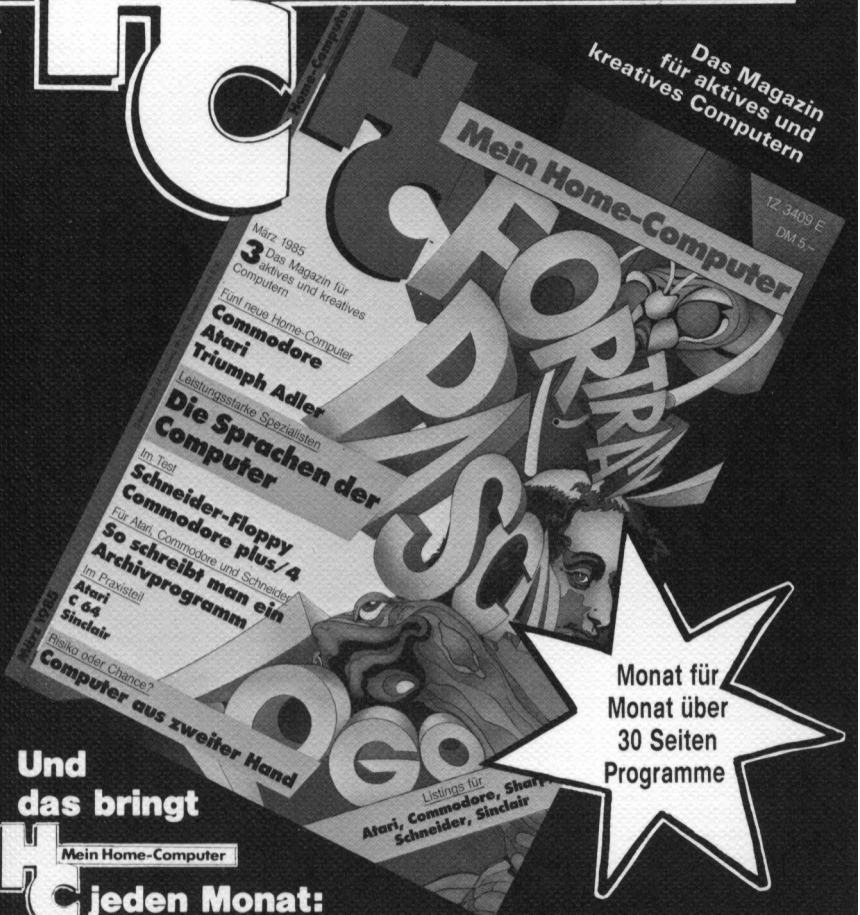
erscheint voraussichtlich im April 1985

Dieses Buch wendet sich an alle, die Logo laden können. Hier wird auf die Darstellung von zwei Logoversionen eingegangen: auf Commodore-Logo und LCSI-Logo, das auf dem Apple IIe zur Verfügung steht. Diese Einführung in die Informatik mit Logo befähigt den Anwender, gegebene Ansätze weiterzuentwickeln. Darüber hinaus kommt es darauf an, selbständig neue Ideen zu verwirklichen.

Schneller erfolgreich durch Computer-Bücher

Mein Home-Computer

Das Magazin für aktives und kreatives Computern



Monat für Monat über 30 Seiten Programme

Und das bringt

Mein Home-Computer

jeden Monat:

- ✻ Programme für alle gängigen Home-Computer
- ✻ Anwendungsbeispiele aus der Praxis
- ✻ Marktübersicht, Tests und Kaufberatung für Zusatzgeräte und Home-Computer
- ✻ Schnellkurse für Einsteiger zum Sammeln
- ✻ Tips und Tricks
- ✻ Interessantes, Aktuelles und Unterhaltsames aus der Home-Computer-Szene
- ✻ News, Clubnachrichten

Holen Sie sich die neueste Ausgabe bei Ihrem Zeitschriftenhändler oder fordern Sie ein Kennenlernheft direkt beim Vogel-Verlag, Leserservice HC, Postfach 67 40, 8700 Würzburg, an.

aktiv computern

Mit dem CPC 464 ist ein Home-Computer auf den Markt gekommen, der bei vielen Computerfreunden in kurzer Zeit begeisterte Aufnahme und Anerkennung gefunden hat. Die Gründe dafür liegen in seinem günstigen Preis-Leistungs-Verhältnis und in einer Reihe von bemerkenswerten Eigenschaften.

Dieses Buch gibt detaillierten Einblick in Hardware und Firmware des CPC 464, wie ihn ein Bedienungshandbuch nicht vermitteln kann. Besonderes Gewicht wird darauf gelegt, den Leser an die optimale Nutzung der vielfältigen und herausragenden Grafik-, Sound-, Interrupt- und Window-Möglichkeiten heranzuführen.

Jede Menge Tips und Tricks, eine Reihe von ausgetesteten Programmen und Routinen sowie nützliche Hinweise zum selbständigen Programmieren von Spielen werden helfen, Ihren CPC 464 besser zu verstehen und effektiver einzusetzen.

Mit diesem Buch haben Sie die idealen Grundlagen, unabhängig von vorgefertigten Programmen Ihre eigenen Wege zu gehen, Ihrer Phantasie freien Lauf zu lassen und kreativ zu computern.

VOGEL-BUCHVERLAG
WÜRZBURG

ISBN 3-8023-0841-7

Martin Aschoff

Was der CPC 464 alles kann

AMSTRAD CPC

MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA

https://acpc.me/