CPC6128



Benutzerhandbuch

AMSTRAD

Bescheinigung des Herstellers/Importeurs

Hiermit wird bescheinigt, daß der

CPC 6128

(Gerät, Typ, Bezeichnung)

in Übereinstimmung mit den Bestimmungen der AmtsblVfg 1046/1984

funkentstört ist.

Der Deutschen Bundespost wurde das Inverkehrbringen dieses Gerätes angezeigt und die Berechtigung zur Überprüfung der Serie auf Einhaltung der Bestimmungen eingeräumt.

AMSTRAD GmbH, Robert-Koch-Str. 5, 6078 Neu-Isenburg

Name des Herstellers/Importeurs

AMSTRAD CPC 6128

Computer mit integriertem Diskettenlaufwerk

Die logische Fortsetzung

Der riesige Erfolg der AMSTRAD-Computer CPC464 und CPC664 bildete eine solide Grundlage für die Weiterentwicklung auf diesem Gebiet, so daß wir jetzt voller Stolz den CPC6128 vorstellen können.

Gemäß unserem Grundsatz der Kompatibilität kann der CPC6128 mit der Software der CPC464-und CPC664-Geräte betrieben werden. Er wurde zusätzlich mit weiteren 64K-Byte RAM für das CP/M-Betriebssystem und andere Anwendersoftware ausgerüstet.

CP/M Plus

CP/M Plus (auch unter dem Namen CP/M 3.1 bekannt) bietet leichten Zugang zur CP/M 80 Software-Bibliothek. Die 61K TPA bedeuten, daß jedes CP/M 80 Programm genug Platz findet, und daß eine Unmenge von Daten aufgenommen werden kann. CP/M Plus ist mit CP/M 2.2 kompatibel. Trotzdem haben wir beide Versionen angeboten, um sicherzugehen, daß Anwenderprogramme, die mit Firmware-Eigenschaften der AMSTRAD CP/M 2.2-Systeme entwickelt wurden, ohne Abänderungen betrieben werden können.

CP/M Plus bietet viele Verbesserungen, mit der die Leistung des Systems erhöht wurde, und die CP/M 2.2 Software, die heute auf dem Markt ist, kann diese Möglichkeiten problemlos ausschöpfen. Das CP/M Plus System besitzt die besondere Eigenschaft, den Bildschirm auf die Eigenschaften von VT52 sowie Zenith Z/19 und Z/29 Terminals einstellen zu können, so daß die Software dieser Geräte ohne weiteres auf dem CPC6128 betrieben werden kann.

GSX

Diese Version von CP/M Plus beinhaltet ein erweitertes Graphik-Betriebssystem, das sogenannte GSX (Graphics Extension System). Dadurch können Drucker bzw. Plotter und der Graphik-Bildschirm mit Standardanweisungen aus den Anwenderprogrammen direkt angesprochen werden. Das Prinzip der CP/M-Programmverträglichkeit geht also über die Text-Bildschirmkompatibilität hinaus. Software, die sich die Möglichkeiten von GSX zunutze macht, kann an diverse Zeichengeräte angeschlossen werden, um Kurven, Diagramme und ähnliches auszudrucken.

Dr. LOGO

Die Programmiersprache LOGO von Digital Research spielt eine immer größere Rolle im Programmierunterricht an den Schulen. Der größere Speicher des 6128 ermöglichte es, das erstmals im CP/M 2.2 verwendete Dr. LOGO zu erweitern. Natürlich werden alle Programme ab AMSTRAD CP/M 2.2 aufwärts kompatibel sein.

Disketten

Der CPC664 mit seinem eingebauten Diskettenlaufwerk war ein Richtungsweiser für die Zukunft. Auch der CPC6128 bringt dem zunehmend anspruchsvollen Benutzer, der Personal-Computer Leistung zum Home-Computerpreis wünscht, den preiswerten Diskettenbetrieb. Die Disketten aller AMSTRAD-Systeme sind austauschbar, wenngleich natürlich Programme, die mit den neuen Möglichkeiten von CP/M Plus und GSX arbeiten, nicht auf Geräten betrieben werden können, denen diese Möglichkeiten fehlen.

Software

Mit dem CPC6128 kann die gesamte CPC664 und CPC464/DDI1 Diskettensoftware betrieben werden, sowie nahezu jede CPC464 Software (vorausgesetzt, daß an den CPC6128 ein Kassettenlaufwerk angeschlossen wird). Auf diese Weise hat der Anwender sofort eine beneidenswerte Auswahl aus AMSTRADs großem Softwareangebot, sowie die Produkte zahlreicher unabhängiger Softwareanbieter zur Verfügung.

AMSTRAD freut sich über alle Kommentare und Vorschläge zum Produkt selbst oder zu diesem Handbuch.
Schreiben Sie an:

AMSTRAD GmbH
Robert-Koch-Str. 5
D-6078 Neu-Isenburg

Alle Wartungs- und Servicearbeiten müssen von AMSTRAD-autorisierten Händlern durchgeführt werden. AMSTRAD trägt keine Verantwortung für Schäden, die durch unsachgemäße Wartung bzw. Service durch unbefugte Personen entstanden sind. Diese Anleitung dient nur dazu, dem Anwender bei der Benutzung des Produktes zu helfen. AMSTRAD übernimmt keine Verantwortung für Schäden, die durch die Anwendung von falschen Informationen, oder Fehlern bzw. fehlenden Informationen in dieser Anleitung, oder durch eine falsche Anwendung des Produktes verursacht wurden.

Dr.LOGO, CP/M, CP/M Plus, GSX und Dr.Graph sind Warenzeichen der Digital Research Inc.
IBM und IBM PC sind Warenzeichen der International Business Machines Inc.
Z19, Z29 und H89 sind Warenzeichen der Zenith Data Systems Inc.
VT52 ist ein Warenzeichen der Digital Equipment Corp.
AMSDOS, CPC6128, CPC664 und CPC464 sind Warenzeichen von AMSTRAD.

Zweite Ausgabe 1988
Herausgegeben von AMSTRAD

Originalausgabe in Englisch

Original Copyright © 1985 by AMSOFT, AMSTRAD Consumer Electronics plc and Locomotive Software Limited

Zusammengestellt von Ivor Spital

Autoren: Ivor Spital, Roland Perry, William Poel, Cliff Lawson mit Genehmigung von Locomotive Software Ltd.

Mitarbeit: Alexander Martin, David Radisic, Ken Clark

Deutsche Übersetzung: Metta Voigts

Photosatz: KAMSET Typesetting Graphics Brentwood



Vervielfältigung und Weitergabe von Informationen aus diesem Benutzerhandbuch - auch auszugsweise- bedürfen der vorherigen schriftlichen Zustimmung von AMSTRAD.

Alle technischen Daten, Informationen sowie Eigenschaften des in diesem Benutzerhandbuch beschriebenen Produktes wurden nach bestem Wissen zusammengestellt und entsprechen dem Stand bei Drucklegung.

Änderungen und Verbesserungen des Produktes aufgrund technischer Neuentwicklungen sind möglich.

Wichtige Hinweise

Vor der Inbetriebnahme sind folgende Anweisungen unbedingt zu beachten.

Hinweise zur Inbetriebnahme:

1. Netzstecker immer an einer geerdeten Steckdose anschließen.

2. Die Geräte nur an eine Stromversorgung mit 220-240V ~ 50Hz anschließen.

3. Geräte nicht öffnen. Sie enthalten keine vom Benutzer reparierbaren Teile. Servicearbeiten nur von hierzu autorisierten Technikern durchführen lassen.

4. Um die Augen nicht unnötig zu belasten, den Monitor so weit wie möglich von der Tastatur entfernt aufstellen. Der Raum sollte gut beleuchtet sein. Der Helligkeitsregler (mit **BRIGHTNESS** bezeichnet) am Monitor sollte so niedrig wie möglich eingestellt sein.

5. Den Computer frontal vor dem Monitor, aber so weit wie möglich von ihm entfernt aufstellen. Um eine größtmögliche Datensicherheit zu gewährleisten, solite das Diskettenlaufwerk nicht direkt vor dem Bildschirm stehen, sondern rechts davon. Stellen Sie den Computer nicht in die Nähe von elektrischen Störfeldern.

6. Diskettenlaufwerk und Disketten stets außerhalb starker magnetischer Felder halten.

7. Arbeiten Sie mit einem Disketten-Doppellaufwerk, achten Sie darauf, daß das verbindende Flachbandkabel nicht in der Nähe des Netzkabels verläuft.

8. Lüftungsschlitze dürfen nicht zugedeckt werden.

9. Setzen Sie die Geräte nicht extremer Hitze, Kälte, Feuchtigkeit oder Staub aus.

Hinweise zum Betrieb:

(Lassen Sie sich durch die vielen Fachausdrücke in diesem Abschnitt nicht verwirren. Die Bedeutung dieser Hinweise wird mit dem Durcharbeiten des Handbuches klarer.)

1. Schalten Sie die Geräte niemals an oder ab, solange sich eine Diskette im Laufwerk befindet, da andernfalls wertvolle Programme oder Daten auf der Diskette zerstört werden könnten.

2. Machen Sie immer Reservekopien von Disketten mit wertvollem Inhalt. Es ist besonders wichtig, Kopien von den mitgelieferten CP/M Systemdisketten des CPC6128 zu machen. Der Verlust bzw. die Beschädigung dieser Disketten könnte mit hohen Kosten verbunden sein.

3. Um zu verhindern, daß Sie Ihre CP/M Systemdisketten aus Versehen überschreiben, vergewissern Sie sich, daß die Schreibschutzlöcher auf den Disketten immer offen sind.

4. Wenn Sie mit einem Disketten-Doppellaufwerk arbeiten, d.h. wenn Sie sich zusätzlich ein AMSTRAD FD1 angeschafft haben, schalten Sie das zweite Laufwerk immer an, bevor Sie den Computer einschalten.

5. Berühren Sie niemals die Oberfläche der Diskette, die sich im Plastikgehäuse befindet.

6. Nehmen Sie niemals eine Diskette aus dem Laufwerk, während etwas auf sie geschrieben oder von ihr gelesen wird.

7. Bedenken Sie immer, daß durch das Formatieren einer Diskette ihr Inhalt gelöscht wird.

8. Das eingebaute Disketten-Interface beansprucht einen kleinen Teil des Speichers, der möglicherweise für Kassettenprogramme benötigt wird, die für den CPC464 geschrieben wurden. Diese Kassetten laufen auf dem CPC6128 mit angeschlossenem Kassettenlaufwerk nicht einwandfrei. In der Regel läuft jedoch AMSTRAD-Software für den CPC464 auch auf dem CPC6128.

9. Die Lizenz für Ihre CP/M Systemdisketten (die mit einer elektronisch codierten Seriennummer versehen wurden) erlaubt ihren Betrieb nur auf einem bestimmten Computersystem. Insbesondere dürfen Sie keiner anderen Person eine Kopie Ihrer serienmäßig codierten CP/M-Diskette überlassen. Lesen Sie bitte die Lizenzbestimmungen im 1.Teil des Anhangs aufmerksam durch.

INHALT

Die Zahlen geben die Seiten der jeweiligen Kapitel an.

Kapitel 1
Grundlagenkurs

Inbetriebnahme des Computers 1
Anschluß der Peripheriegeräte 7
Disketten 11
Die Tastatur 15
Das Laden von Software 21
Einführung in das AMSTRAD-BASIC 24
Einführung in den Diskettenbetrieb 41
Bildschirm-Modus, Farben und Graphik 51
Tonerzeugung 73
Einführung in AMSDOS und CP/M 81
Das Bank Manager-Programm 90

Kapitel 2
Das erste Programm

Vorüberlegungen
Wie man ein einfaches Programm schreibt
Arrays 3
Zusammenstellung eines Menüs 6
Speichern von Variablen 9
Editieren und automatische Numerierung 11
Laden von Variablen 11
Übersichtliche Gestaltung 12

Kapitel 3
Liste aller Befehle des AMSTRAD CPC6128 BASIC

Erläuterungen zur Notation 1
Sonderzeichen 2
Datentypen 2
Aufbau der Befehlsbeschreibungen 3
Befehle 4

Kapitel 4
Das Arbeiten mit Disketten und Kassetten

Reservekopien von Systemdisketten 2
Start mit CP/M Plus 2
Das PROFILE-Programm 3
Das HELP-Programm 3
Betrieb mit einem oder zwei Laufwerken 4
Kopieren von Disketten-Dateien mit PIP 4
Disketten mit reinem BASIC-Inhalt 5
Schlüsselfertige AMSTRAD BASIC-Disketten 5
Schlüsselfertige CP/M-Disketten 5
Konfigurieren eines CP/M-Programms 7
Start eines schlüsselfertigen CP/M-Pakets 9
Start mit GSX 9
Betrieb mit CP/M 2.2 11
Kassettenbetrieb 12
Speichern auf Kassette 17

Kapitel 5
AMSDOS und CP/M

AMSDOS:

Einführung in AMSDOS 1
Disketten-Inhaltsverzeichnis 2
Diskettenwechsel 2
Dateinamen und -typen 2
AMSDOS Datei-Vorspann 3
Dateinamen beim Betrieb mit zwei Diskettenlaufwerken 4
Universalzeichen 5
Liste der AMSDOS-Befehle 8
Kopieren von Dateien 12
Liste der AMSDOS-Fehlermeldungen 15

CP/M:

Einführung in CP/M 17
"Booten" von CP/M Plus 18
Direkter Konsolmodus 19
Peripherie-Verwaltung 24
Transiente Programme 30
CP/M 2.2 33

Kapitel 6
Einführung in LOGO

Was ist LOGO? 1
Dr. LOGO Prozeduren 3
Editieren von Programmen und Prozeduren 6
Bedienungshinweise 7
Verzeichnis der Dr. LOGO Grundprozeduren 7
Wörter- und Listenverarbeitung 8
Arithmetische Operationen 13
Logische Operationen 16
Variablen 17
Prozeduren 18
Editier-Befehle 19
Drucker-Funktionen 20
Text-Bildschirm 20
Graphik-Bildschirm 22
Schildkröten-Graphik 26
Speicherplatz-Verwaltung 30
Eigenschaftslisten 32
Disketten-Dateien 33
Tastatur und Joystick 36
Sound-Befehle 37
Ablauf-Steuerung 38
Ausnahmefälle 40
System-Primitive 42
System-Variablen 42
System-Eigenschaften 43

Kapitel 7
Übersicht

Cursorpositionen und Steuer-Codes 1
Unterbrechungsfunktionen 8
ASCII-Zeichensatz und graphische Zeichen 9
Tastenbezeichnungen 22
Noten und Tonperioden 25
Fehlermeldungen 28
BASIC-Schlüsselwörter 33
Entwurfsgitter 35
Anschlüsse 39
Drucker 42

Joysticks 43
Disketten-Organisation 44
Eingebaute System-Erweiterungen (RSX) 46
Speicher 47
CP/M Plus Bildschirmanpassung 49
CP/M Plus Zeichensatz 54

Kapitel 8
Näheres über den Bank Manager

Die zusätzlichen 64K 1
Speichern von Bildschirm-Inhalten 2
Bank-switching 2
Pseudo-Datei-Operation 4

Kapitel 9
Wenn Sie mal Zeit haben

Computer im allgemeinen:

Hardware und Software 1
Unterschiede zwischen Computern 1
Einige weit verbreitete Irrtümer 4
Die Sprache der Computer 7
Bits und Bytes 9
Das Hexadezimalsystem 11

Der CPC6128 im besonderen:

Zeichensatz 14
Variablen 16
Logik 18
Benutzer-definierte Zeichen 21
Druckformatierung 22
Windows 27
Unterbrechungen 30
Daten 32
Tonerzeugung 36
Graphik 49
Zusätzlicher Speicherplatz für Graphik 58

Anhang 1: Endabnehmer-Lizenzabkommen

Anhang 2: Fachwörterverzeichnis

Anhang 3: 6 Programme

Anhang 4: Index

Kapitel 1
GRUNDLAGENKURS

Teil 1: Inbetriebnahme des Computers

Der CPC6128 kann wahlweise mit folgenden Geräten betrieben werden:

1. AMSTRAD GT65 Grünmonitor
2. AMSTRAD CTM 644 Farbmonitor
3. AMSTRAD MP2 Modulator mit Netzteil und einem normalen Fernsehgerät (UHF)

Wenn der Computer nicht benutzt wird, Netzstecker ziehen.

Entfernen Sie keine Schrauben und versuchen Sie nicht, das Gehäuse des Computers, des Monitors oder des Modulators zu öffnen. Beachten Sie die Warnungen auf den Etiketten an der Unter- bzw. Rückseite der Geräte.

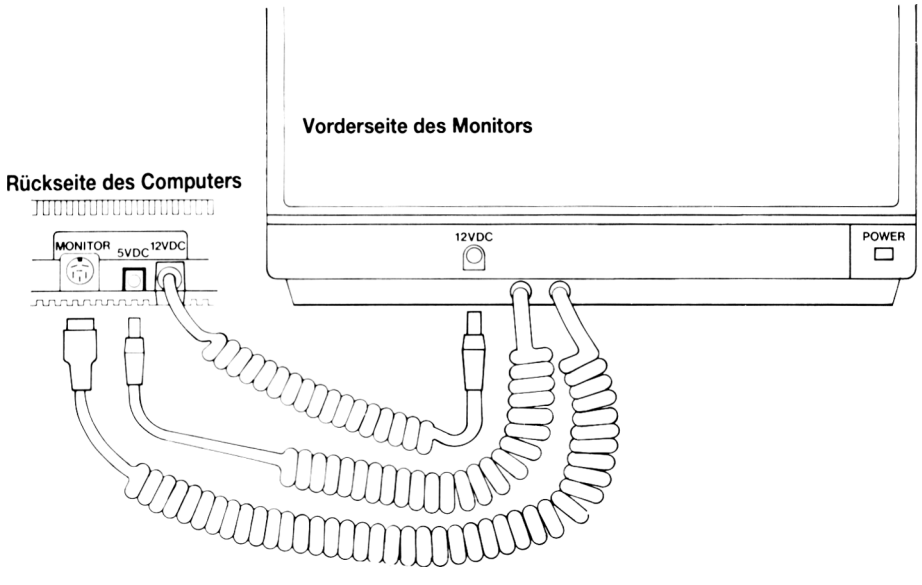
VORSICHT!

Innenteile unter Spannung. Keine Schrauben lösen.

Anschluß des Grün- oder Farbmonitors an den Computer

(Wenn Sie den CPC6128 mit einem MP2 Modulator mit Netzteil betreiben, gehen Sie gleich zum nächsten Abschnitt über).

1. Vergewissern Sie sich, daß der Monitor nicht an das Stromnetz angeschlossen ist.
2. Stecken Sie das Kabel mit dem größeren 6-poligen DIN-Stecker, das vorn aus dem Monitor kommt, in die mit **MONITOR** bezeichnete Buchse an der Rückseite des Computers.
3. Stecken sie das Kabel mit dem kleineren Stecker (5V Gleichspannung), das vorn aus dem Monitor kommt, in die mit **5V DC** bezeichnete Buchse an der Rückseite des Computers.
4. Stecken Sie das Kabel mit dem kleinen Stecker (12V Gleichspannung), das hinten aus dem Computer kommt, in die Buchse an der Vorderseite des Monitors.

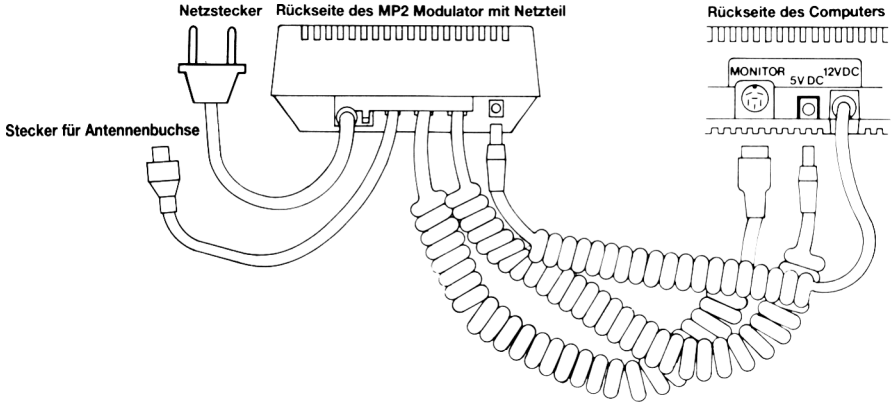


Anschluß des MP2 Modulator mit Netzteil an den Computer

Der MP2 ist ein Zusatzgerät, mit dem Sie den Computer an Ihren Farbfernseher anschließen können. Auf diese Weise brauchen Sie auf die vielfältigen farblichen Möglichkeiten des CPC6128 nicht zu verzichten, wenn Sie nur einen Grün-Monitor GT65 gekauft haben.

Der MP2 sollte rechts neben den CPC6128 gestellt werden.

1. Vergewissern Sie sich, daß der MP2 nicht an das Stromnetz angeschlossen ist.
2. Stecken Sie das Kabel mit dem größeren 6-poligen DIN-Stecker, das aus dem MP2 kommt, in die mit **MONITOR** bezeichnete Buchse an der Rückseite des Computers.
3. Stecken Sie das Kabel mit dem kleineren Stecker (5V Gleichspannung), das aus dem MP2 kommt, in die mit **5V DC** bezeichnete Buchse an der Rückseite des Computers.
4. Stecken Sie das Kabel mit dem Antennenstecker, das aus dem MP2 kommt, in die Antennenbuchse Ihres Fernsehgerätes.
5. Stecken Sie das Kabel mit dem kleinen Stecker (12V Gleichspannung), das hinten aus dem Computer kommt, in die Buchse an der Rückseite des MP2.



Das Einschalten des CPC6128 mit angeschlossenem GT65 oder CTM644

(Wenn Sie den CPC6128 mit Modulator/Netzteil betreiben, gehen Sie gleich zum nächsten Abschnitt über.)

Wenn Sie die Geräte wie vorstehend beschrieben verbunden haben, stecken Sie den Netzstecker in eine Steckdose. Drücken sie den **POWER**-Knopf rechts unten am Monitor. Wenn sich der Knopf in der Aus-Position befindet, ist die Stromzufuhr zum System unterbrochen.

Schalten Sie den Schiebeschalter (**POWER**) rechts seitlich am Computer auf **ON**.

Jetzt sollte über dem Tastenfeld ein rotes Lämpchen aufleuchten und auf dem Bildschirm folgendes Bild erscheinen:

```
AMSTRAD 128K Microcomputer    (v3)
© 1985 Amstrad Consumer Electronics plc
            and Locomotive Software Ltd.

BASIC 1.1

Ready
■
```

Um die Augen nicht unnötig zu belasten, stellen Sie den mit **BRIGHTNESS** bezeichneten Helligkeitsregler so ein, daß die Schrift auf dem Bildschirm nicht zu grell erscheint oder verschwimmt.

Den Helligkeitsregler finden Sie beim GT65 an der Vorderseite rechts unten, und beim CTM644 an der rechten Seitenwand.

Wenn Sie den GT65 Grünmonitor benutzen, müssen Sie möglicherweise auch den Kontrast und den Bildfang nachstellen. Die Regler dafür finden sie vorn unten am Monitor.

Der Kontrast (**CONTRAST**) sollte so schwach wie möglich eingestellt sein, um den Augen das Sehen angenehm zu machen.

Die Bildfangregelung (**V-HOLD**) sollte so eingestellt sein, daß sich das Bild in der Mitte des Bildschirms befindet und nicht zittert oder nach oben wegrollt.

Das Einschalten des CPC6128 mit angeschlossenem MP2 Modulator mit Netzteil

Wenn Sie die Geräte wie vorstehend beschrieben verbunden haben, stecken Sie den Netzstecker in eine Steckdose.

Schieben Sie jetzt den **POWER**-Schalter rechts seitlich am Computer auf **ON**, um den Computer anzuschalten.

Nun sollte über dem Tastenfeld ein rotes Lämpchen aufleuchten. Sie schalten jetzt das Fernsehgerät an und stellen es so ein, daß Sie ein Signal vom Computer erhalten.

Falls Sie einen Fernseher mit Programmtasten besitzen, drücken Sie eine freie bzw. unbenutzte Taste. Drehen Sie am zugehörigen Regler für die Feineinstellung, wie es in der Gebrauchsanleitung für Ihren Fernseher beschrieben ist, bis Sie - ungefähr bei Kanal 36 - folgendes Bild erhalten:

```
AMSTRAD 128K Microcomputer    (v3)
C 1985 Amstrad Consumer Electronics plc
            and Locomotive Software Ltd.

BASIC 1.1

Ready
■
```

Justieren Sie den Fernseher, bis Sie ein klares Bild haben. Die Schrift sollte goldgelb auf tiefblauem Hintergrund stehen.

Falls Ihr Fernseher einen Programmdrehwähler hat, drehen Sie den Knopf solange, bis das oben gezeigte Bild erscheint und absolut ruhig bleibt (wiederum ungefähr bei Kanal 36).

Anschluß weiterer Geräte

Einzelheiten über den Anschluß weiterer Geräte an das Standard-Computersystem wie:

Joystick(s)
Kassettenlaufwerk
Drucker
Zweites Diskettenlaufwerk
Externe Verstärker/Lautsprecher
Erweiterungsgeräte

...finden Sie im 2. Teil des Kapitels 'Grundlagenkurs'.

Überprüfen Sie zum Schluß noch einmal, ob Sie folgende Punkte unter 'WICHTIGE HINWEISE' am Beginn des Handbuches beachtet haben:

'Hinweise zur Inbetriebnahme': Punkt 1, 2, 4, 5, 6, 7 und 8
'Hinweise zum Betrieb': Punkt 1

Teil 2: Anschluß der Peripheriegeräte

Dieser Abschnitt erläutert den Anschluß des weiteren Zubehörs, der sogenannten Peripheriegeräte, an das CPC6128 System. Der Anwendung dieser Geräte sind jeweils eigene Abschnitte in diesem Handbuch gewidmet.

Der Joystick

Den AMSTRAD Joystick JY2 können Sie als Zubehör zu ihrem Computer kaufen. Er wird in Computerspielen zum Steuern oder Schießen eingesetzt. Der CPC6128 kann mit zwei Joysticks verbunden werden. Der AMSTRAD Joystick JY1 eignet sich ebenfalls für den Betrieb mit dem CPC6128.

Stecken Sie das Kabel des JY2 in die mit **JOYSTICK** bezeichnete Buchse am Computer. Wenn Sie einen zweiten Joystick anschließen möchten, stecken Sie den Stecker des zweiten Joysticks in die Buchse am ersten Joystick.

Weitere Einzelheiten zum Thema 'Joystick' finden Sie weiter hinten in diesem Handbuch.

Das Kassettenlaufwerk

Programme können statt auf Diskette auch auf Kassette gespeichert bzw. von Kassette gelesen werden. Die Befehle, mit denen Daten auf Diskette oder Kassette gespeichert und wieder abgerufen werden können, werden später behandelt.

Um das Kassettenlaufwerk an den CPC6128 anschließen zu können, benötigen Sie ein Standard-Verbindungskabel für Kassettenrecorder.

Stecken Sie den größeren 5-poligen DIN-Stecker des Kabels in die mit **TAPE** bezeichnete Buchse am Computer.

Stecken Sie das Ende des blauen Kabels in die mit **REMOTE** oder **REM** bezeichnete Buchse des Kassettengeräts.

Das Ende des roten Kabels stecken Sie in die mit **MIC**, **COMPUTER IN**, oder **INPUT** bezeichnete Buchse des Kassettengeräts.

Zum Schluß stecken Sie das Ende des weißen Kabels in die mit **EAR**, **COMPUTER OUT**, oder **OUTPUT** bezeichnete Buchse des Kassettengeräts.

Besonderer Hinweis: Eine einwandfreie Datenübertragung zwischen dem CPC6128 und der Kassette hängt im wesentlichen von der richtigen Einstellung der Lautstärke (VOLUME) ab. Wenn Sie den Eindruck haben, daß das Speichern oder Laden von Programmen nicht richtig funktioniert, experimentieren Sie mit dem Lautstärkeregler, bis Sie die optimale Einstellung gefunden haben.

Der Drucker

Der CPC6128 kann an jeden Drucker angeschlossen werden, der mit einer standardisierten Centronics-Schnittstelle ausgerüstet ist. Wenn Sie einen AMSTRAD Drucker anschließen wollen, verwenden Sie einfach das zugehörige Anschlußkabel.

Wenn Sie einen anderen Centronics-kompatiblen Drucker benutzen möchten, brauchen Sie ein passendes Verbindungskabel für den Drucker.

Stecken Sie den Platinenstecker des Verbindungskabels in die mit **PRINTER** bezeichnete Buchse an der Rückseite des Computers.

Stecken Sie das andere Ende des Kabels mit dem Centronics-Stecker in die Buchse an der Rückseite des Druckers. Wenn die Buchse des Druckers mit Sicherheitsklammern versehen ist, drücken Sie diese in die Aussparungen des Steckers.

Einzelheiten zum Betrieb des Druckers werden weiter hinten im Handbuch behandelt.

Das zweite Diskettenlaufwerk (AMSTRAD FD1)

Das AMSTRAD FD1 kann als zweites Diskettenlaufwerk an das System angeschlossen werden. Die Vorteile des doppelten Diskettenbetriebs werden besonders dem regelmäßigen CP/M Benutzer zugute kommen. Viele Programme sind so angelegt, daß sich die Bibliotheksdiskette in einem Laufwerk befindet, während mit den Dateien auf der Diskette im zweiten Laufwerk gearbeitet wird.

Der Betrieb mit CP/M bedeutet immer, daß ein Programm von Diskette geladen werden muß (der Benutzer hat keinen Zugang zum ROM-BASIC). Da CP/M durch eine spezielle Überlagerungstechnik den Mehrprogrammbetrieb ermöglicht, und Programme zuläßt, die größer sind als der RAM-Speicher, kann die eigentliche Bibliotheksdiskette so viele Programme enthalten, daß für die Daten wenig Platz bleibt.

Dank der vielseitigen Möglichkeiten Ihrer CPC6128 System-Diskette können Sie alle notwendigen Arbeiten an Ihren Dateien, wie kopieren, löschen usw., mit einem einzigen Diskettenlaufwerk bewältigen. Mit einem zweiten Diskettenlaufwerk können Sie diese Vorgänge jedoch beschleunigen und Mißgeschicke vermeiden.

Für den Anschluß des FD1 benötigen sie das AMSTRAD DI2 Verbindungskabel für Diskettenlaufwerke.

Stecken Sie das Kabelende mit dem größeren Platinenstecker in die mit **DISC DRIVE 2** bezeichnete Buchse an der Rückseite des Computers.

Stecken Sie das andere Ende mit dem kleineren Stecker in die Buchse hinten am FD1 Diskettenlaufwerk.

Achtung! Bevor Sie das zweite Diskettenlaufwerk anschließen bzw. abmontieren, vergewissern Sie sich, daß sich keine Diskette in den beiden Laufwerken befindet und daß das System abgeschaltet ist. Wenn Sie Geräte anschließen oder entfernen, ohne das System vorher abzuschalten, besteht die Gefahr, daß das Programm im Computerspeicher beschädigt wird. Speichern Sie daher wertvolle Programme, bevor Sie mit den Anschlußkabeln herumhantieren!

Wenn Sie das FD1 an den 6128 angeschlossen haben, schalten Sie ZUERST das Diskettenlaufwerk an (Schiebeschalter an der Rückseite des FD1), DANACH schalten Sie den Computer an (Schiebeschalter an der rechten Seite des Computers). Jetzt sollten das grüne und das rote Lämpchen an der Vorderseite des Diskettenlaufwerks leuchten. Das doppelte Diskettenlaufwerk ist somit betriebsbereit.

Einzelheiten über den Betrieb mit zwei Diskettenlaufwerken finden sie weiter hinten im Handbuch.

Externe Verstärker/Lautsprecher

Wenn Sie an Ihren 6128 einen Stereo-Verstärker mit Lautsprechern anschließen, kommen die 3-Kanal-Tonqualitäten des Computers voll zum tragen.

An der Zuleitung Ihres Verstärkers sollte sich ein 3,5mm Stereo-Stecker befinden, den Sie in die mit **STEREO** bezeichnete Buchse des Computers stecken.

Die Verbindungsstellen des Steckers sind folgende:

Steckerspitze: Linker Kanal
Innenring des Steckers: Rechter Kanal
Hintere Steckerhülse: Erde (gemeinsam)

Der 6128 sendet ein Signal mit fester Spannung aus der **STEREO**-Buchse aus. Deshalb sollten Sie Lautstärke, Balance und Klang mit den Reglern am Verstärker einstellen.

Kopfhörer mit hoher Impedanz können ebenfalls an den 6128 angeschlossen werden, allerdings ist die Lautstärke nicht am Computer zu regulieren. Lautsprecher mit niedriger Impedanz, wie sie für Stereoanlagen verwendet werden, funktionieren nicht, wenn sie direkt an den Computer angeschlossen werden.

Einzelheiten darüber, wie der Ton über den gewünschten Kanal ausgegeben wird, finden Sie weiter hinten im Handbuch.

Erweiterungsgeräte

Erweiterungsgeräte, wie Schnittstellen für serielle Datenübertragung, Modems, Lichtgriffel, ROMs usw. können mit dem 6128 über die Buchse mit der Bezeichnung **EXPANSION** verbunden werden.

Weitere Einzelheiten über Anschlüsse an die **EXPANSION**-Buchse finden Sie im Kapitel 'Übersicht'.

Zum Schluß überprüfen Sie noch einmal, ob Sie folgende Punkte unter 'WICHTIGE HINWEISE' am Beginn des Handbuches beachtet haben:

'Hinweise zur Inbetriebnahme': Punkt 6 und 7

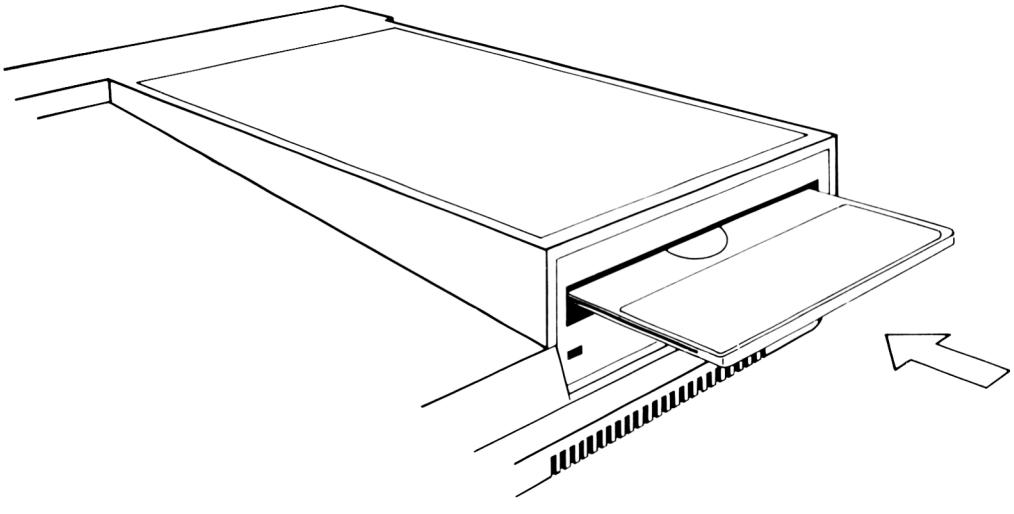
'Hinweise zum Betrieb': Punkt 4 und 8

Teil 3: Disketten

Für den AMSTRAD CPC6128 werden 3-Zoll Kompakt-Disketten verwendet. Wir möchten Ihnen dringend empfehlen, ausschließlich AMSTRAD CF2 Kompakt-Disketten zu verwenden, um eine zuverlässige Datenübertragung zwischen Computer und Diskette zu gewährleisten. Es können aber auch Disketten führender Hersteller verwendet werden.

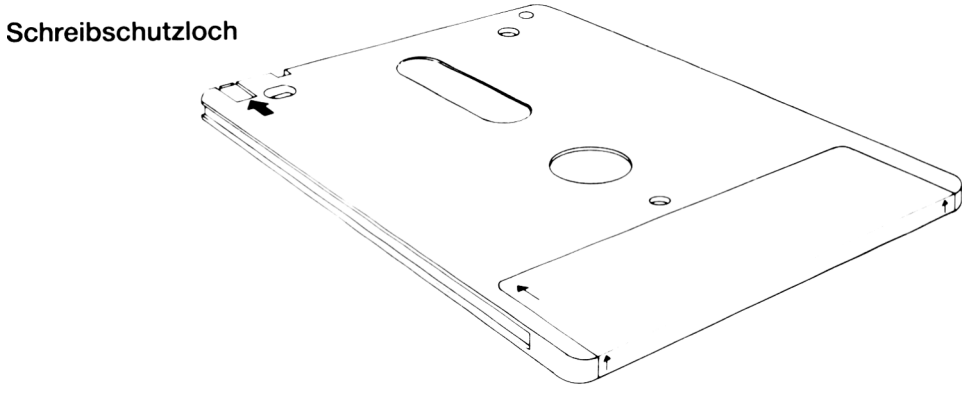
Einlegen der Diskette

Die Disketten sind beidseitig verwendbar. Sie schieben die Diskette so in das Gerät, daß das Etikett nach außen weist und die Seite, die Sie benutzen möchten, oben liegt:



Schreibschutz

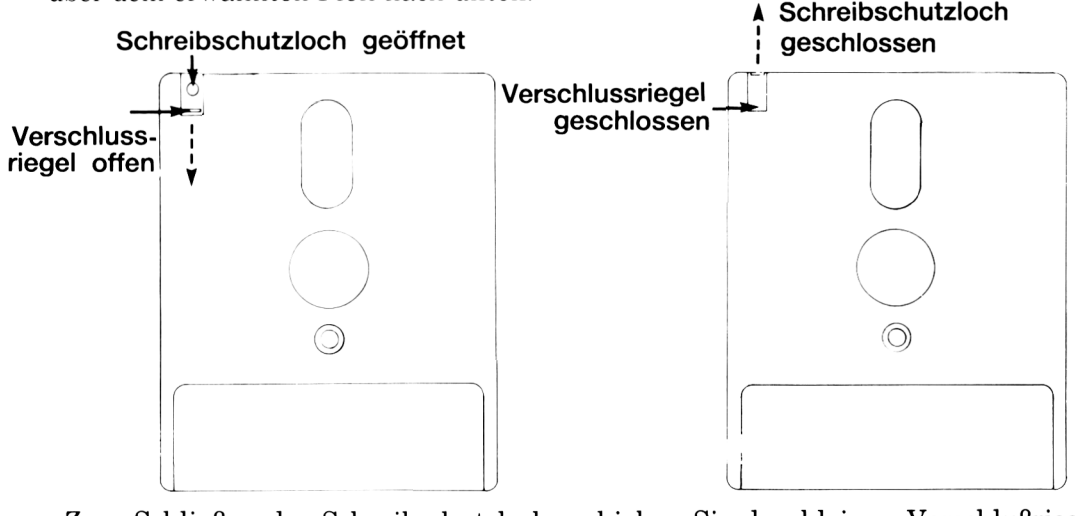
Auf jeder Seite einer unbespielten Diskette sehen Sie in der linken Ecke einen Pfeil, der auf ein kleines verschließbares Loch weist. Dieses sogenannte Schreibschutzloch verhindert das Löschen und Überschreiben der Diskette:



Wenn das Loch geschlossen ist, können Daten vom Computer auf die Diskette 'geschrieben' werden. Ist das Loch jedoch geöffnet, wird die Diskette keine Daten aufnehmen, damit Sie nicht aus Versehen ein wertvolles Programm löschen.

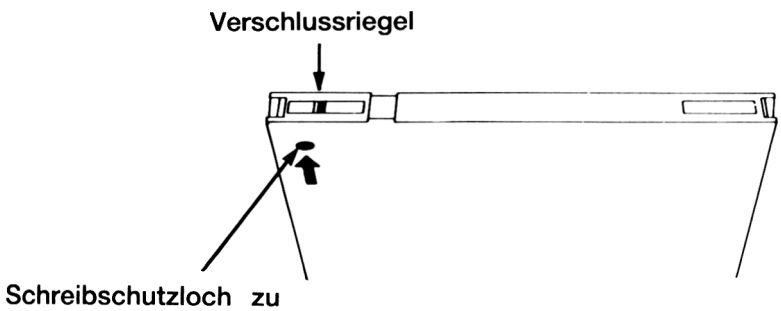
Die Diskettenhersteller haben unterschiedliche Mechanismen zum Öffnen und Schließen des Schreibschutzlochs entwickelt. Beim AMSTRAD CF2 funktioniert er folgendermaßen:

Zum Öffnen des Schreibschutzlochs schieben Sie den kleinen Verschlußriegel über dem erwähnten Pfeil nach unten:

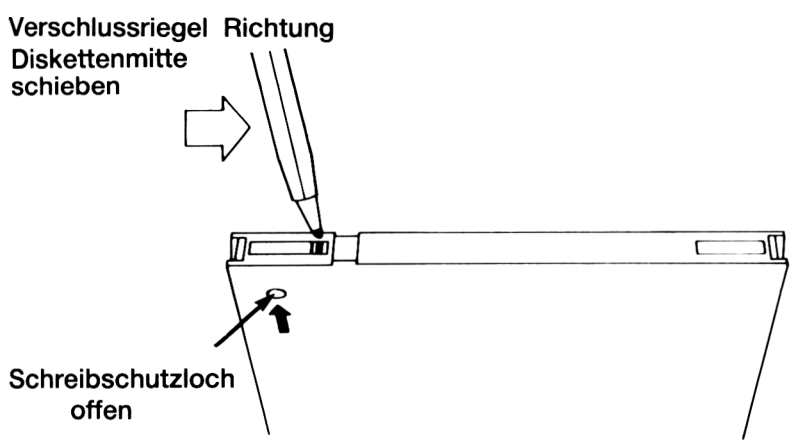


Zum Schließen des Schreibschutzlochs schieben Sie den kleinen Verschlußriegel in Pfeilrichtung nach oben:

Bei einigen anderen Disketten befindet sich der Riegel über dem Loch in einem Schlitz an der Schmalseite des Diskettengehäuses:



Um das Schreibschutzloch bei diesen Disketten zu öffnen, wird der Riegel mit einem spitzen Gegenstand zur Mitte der Diskette hingeschoben:



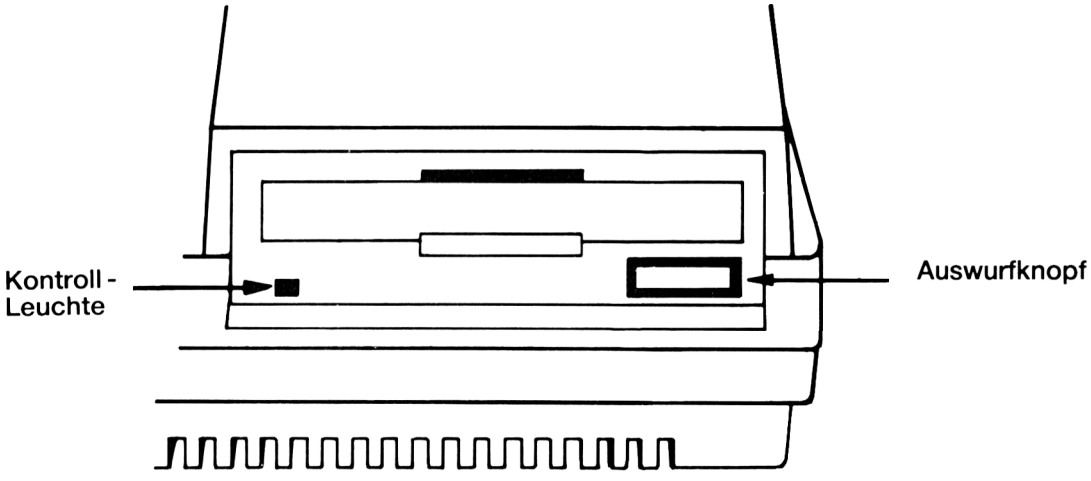
Denken Sie daran, daß das Öffnen des Schreibschutzloches in jedem Fall ein Überschreiben der Diskette verhindert, gleichgültig, welchen Mechanismus die Diskette aufweist.

ACHTUNG!

Achten Sie darauf, daß die Schreibschutzlöcher auf Ihren Systemdisketten immer geöffnet sind.

Wenn sich die Diskette im Laufwerk befindet

Vorn am eingebauten Diskettenlaufwerk sehen Sie ein rotes Kontrollämpchen und einen Druckknopf zum Auswerfen der Diskette



Das Kontrollämpchen

...zeigt an, daß Daten von Diskette gelesen oder auf Diskette geschrieben werden.

Wenn ein zweites Diskettenlaufwerk (Laufwerk B) angeschlossen ist, leuchtet dessen Kontrollämpchen ständig. Es erlischt, wenn Daten auf das eingebaute Diskettenlaufwerk (Laufwerk A) geschrieben, bzw. von ihm gelesen werden.

Der Auswurfknopf

...wird betätigt, um die Diskette herauszunehmen.

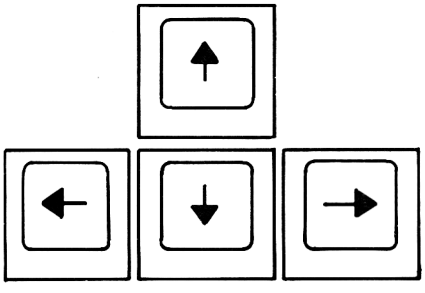
Zum Schluß überprüfen sie noch einmal, ob Sie folgende Punkte unter 'WICHTIGE HINWEISE' am Beginn des Handbuchs beachtet haben:

'Hinweise zum Betrieb': Punkt 1, 3, 4, 5 und 6

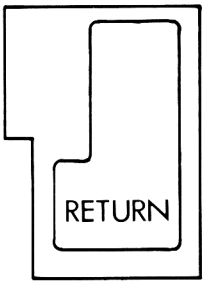
Teil 4: Startvorbereitungen

Bevor wir Ihnen erklären, wie die Programme geladen bzw. auf Diskette gespeichert werden, möchten wir Sie noch mit der Funktion einiger Tasten vertraut machen. Falls Sie schon mit Computern gearbeitet haben, können Sie diesen Abschnitt überspringen.

Sie haben den Computer also eingeschaltet, sehen die 'Einschalt'-Meldung auf dem Bildschirm und möchten nun wissen, was man mit den einzelnen Tasten anfängt.

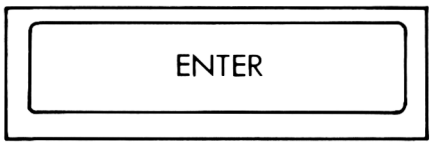


Rechts unten auf dem Tastenfeld sehen Sie die Cursor-Tasten...
Mit ihnen läßt sich der Cursor (der kleine kompakte Block auf dem Bildschirm) in die jeweilige Pfeilrichtung bewegen. Drücken Sie eine Cursor-Taste nach der anderen, und Sie sehen, wie der Cursor seine Positionen ändert.

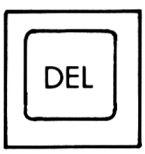


Die **[RETURN]**-Taste (return=zurück) dient dazu, die eingetippte Information dem Computerspeicher zu übergeben. Wenn Sie die **[RETURN]**-Taste gedrückt haben, wird auf dem Bildschirm eine neue Zeile angefangen. Nach jeder Anweisung, die Sie eingetippt haben, müssen Sie die **[RETURN]**-Taste drücken.

Bis auf weiteres schreiben wir im folgenden **[RETURN]** nach jeder Anweisung oder Programmzeile, um anzudeuten, daß die **[RETURN]**-Taste gedrückt werden soll.

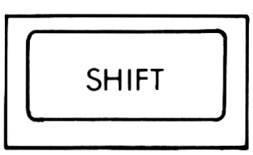


Die **[ENTER]**-Taste (enter=eingeben) hat unter Standardbedingungen dieselbe Funktion wie **[RETURN]** und kann genauso angewendet werden. Allerdings kann die **[ENTER]**-Taste ebenso wie die Funktionstasten auf der Zahlentastatur für andere Zwecke umdefiniert werden. Doch darauf gehen wir später noch ein.



Mit der **[DEL]**-Taste (delete=löschen) können Sie das Zeichen, das sich links vom Cursor befindet (z.B. eine Zahl oder einen Buchstaben), löschen, wenn Sie es nicht brauchen.

Tippen Sie `abcd` ein, und Sie sehen, daß sich der Buchstabe d jetzt links vom Cursor befindet. Wenn Ihnen das d überflüssig erscheint, drücken Sie die **[DEL]**-Taste, und schon ist es verschwunden. Wenn Sie **[DEL]** gedrückt halten, werden nacheinander auch die Buchstaben c,b und a gelöscht.



Auf der Tastatur befinden sich zwei **[SHIFT]**-Tasten (Umschalttasten). Wenn Sie eine von beiden drücken und unten halten, während Sie eine andere Taste anschlagen, erscheint ein Großbuchstabe bzw. das obere Symbol der Taste auf dem Bildschirm.

Schreiben Sie den Buchstaben e, drücken Sie dann die **[SHIFT]**-Taste und schlagen Sie die E-Taste nochmal an. Nun sehen Sie auf dem Bildschirm:

```
eE
```

Jetzt tippen Sie einige Leerstellen, indem Sie die Leertaste gedrückt halten. Schreiben Sie die Ziffer 2 (auf der obersten Tastenreihe), dann drücken Sie die **[SHIFT]**-Taste und schlagen nochmal die Taste mit der 2 an. Nun sehen Sie auf dem Bildschirm:

2"

Sie wissen jetzt also, was geschieht, wenn **[SHIFT]** gedrückt ist und gleichzeitig eine andere Taste angeschlagen wird. Probieren Sie dies auch mit anderen Tasten aus.



Die **[CAPS LOCK]**-Taste (Umschaltfeststeller) hat eine ähnliche Funktion wie **[SHIFT]**, mit dem Unterschied, daß sie nur einmal gedrückt wird. Ab dann erscheinen alle eingetippten Buchstaben auf dem Bildschirm als Großbuchstaben. Wenn Sie die Zahlentasten drücken, erscheinen jedoch nicht die oberen Symbole.

Drücken Sie einmal auf die **[CAPS LOCK]**-Taste und tippen Sie dann ein:

```
abcdef123456
```

Auf dem Bildschirm sehen Sie:

```
ABCDEF123456
```

Sie haben bemerkt, daß die Buchstaben zwar in Großschreibweise erscheinen, die Zahlen jedoch unverändert bleiben. Wenn Sie nun eines der über den Zahlen abgebildeten Symbole schreiben wollen, nachdem Sie **[CAPS LOCK]** gedrückt haben, drücken Sie einfach **[SHIFT]**, bevor Sie die Taste mit dem Symbol anschlagen. Drücken Sie die **[SHIFT]**-Taste, während Sie folgendes schreiben:

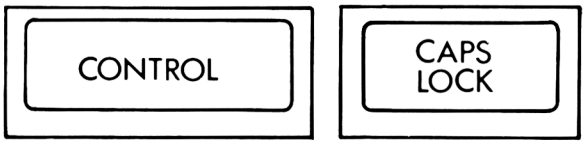
```
abcdef123456
```

Auf dem Bildschirm erscheint:

```
ABCDEF!"#$%&
```

Wenn Sie wieder kleine Buchstaben schreiben wollen, drücken Sie einfach noch einmal die **[CAPS LOCK]**-Taste.

Wenn Sie Großbuchstaben und Symbole schreiben möchten, ohne ständig **[SHIFT]** drücken zu müssen, können Sie folgendes tun:



Halten Sie die **[CONTROL]**-Taste gedrückt und drücken Sie einmal **[CAPS LOCK]**. So erhalten Sie die sogenannte 'SHIFT LOCK'-Funktion. Tippen Sie folgendes ein:

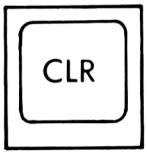
```
abcdef123456
```

Auf dem Bildschirm erscheint:

```
ABCDEF!"#$%&
```

Wenn Sie die SHIFT LOCK-Funktion eingeschaltet haben, können Sie dennoch Zahlen schreiben, indem sie die numerische Tastatur (*f*Ø bis *f*9) auf der rechten Seite des Tastenfeldes benutzen.

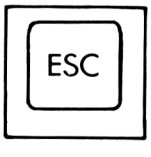
Wenn Sie nochmals **[CTRL]** und **[CAPS LOCK]** drücken, kommen Sie wieder zu der Funktion zurück, die Sie vorher eingestellt hatten, d.h. entweder **[CAPS LOCK]** oder Kleinbuchstaben. Hatten Sie vorher **[CAPS LOCK]** gedrückt, drücken Sie einfach noch einmal **[CAPS LOCK]**, um zur Kleinschreibweise zurückzukehren.



Mit der **[CLR]**-Taste (clear=räumen) wird das Zeichen gelöscht, auf dem sich der Cursor befindet.

Tippen Sie `ABCDEFGH` ein. Der Cursor befindet sich jetzt rechts vom zuletzt eingegebenen Buchstaben (`H`). Drücken Sie nun viermal auf die Cursor-Links-Taste. Der Cursor hat sich nun vier Stellen nach links bewegt und liegt auf dem Buchstaben `E`.

Beachten Sie, daß das E noch innerhalb des Cursors sichtbar ist. Wenn Sie jetzt einmal **[CLR]** drücken, verschwindet das `E`, während `FGH` jeweils eine Stelle nach links rücken. Der Cursor liegt nun auf dem `F`. Drücken Sie noch einmal **[CLR]** und halten Sie die Taste unten. Sie sehen, wie auch `F`, `G` und `H` nacheinander gelöscht werden.



Die **[ESC]**-Taste (escape=ausbrechen) wird benutzt, um aus einer Funktion auszubrechen. Drücken Sie einmal **[ESC]**, und der Computer unterbricht seine laufende Funktion. Er arbeitet jedoch weiter, sobald irgendeine andere Taste gedrückt wird.

Zweimaliges Drücken der **[ESC]**-Taste bewirkt, daß der Computer die Funktion, die er gerade ausübt, völlig abbricht. Er wartet nun auf neue Anweisungen.

Wichtig

Wenn Sie den rechten Bildschirmrand erreicht haben, nachdem Sie 40 Zeichen eingetippt haben, erscheint das nächste Zeichen auf der nächsten Zeile am linken Rand. Sie dürfen **NICHT** auf die **[RETURN]**-Taste drücken, wie Sie auf einer Schreibmaschine den Zeilenvorschub betätigen würden.

Der Computer erledigt dies automatisch für Sie, und wird auf ein unerwartetes **[RETURN]** mit einer Fehlermeldung - normalerweise `Syntax error` - entweder sofort oder beim Programmablauf reagieren.

Syntax Error

Die Meldung '`Syntax error`' erscheint auf dem Bildschirm, wenn der Computer nicht versteht, was Sie eingetippt haben. Schreiben Sie als Beispiel:

```
printt [RETURN]
```

Auf dem Bildschirm erscheint:

```
Syntax error
```

Der Computer gibt diese Meldung aus, weil er die Anweisung '`printt`' nicht versteht.

Wenn Sie einen Tippfehler in einer Programmzeile machen, z.B.:

```
10 printt "abc" [RETURN]
```

...erscheint die Meldung '`Syntax error`' erst, wenn der Computer diese Anweisung verarbeitet, also dann, wenn das Programm läuft.

Schreiben Sie:

```
run [RETURN]
```

(Dieser Befehl weist den Computer an, das Programm, das Sie gerade in seinen Speicher getippt haben, auszuführen.)

Nun erscheint folgendes auf dem Bildschirm:

```
Syntax error in 10
10 printt "abc"
```

Die Meldung sagt Ihnen, in welcher Zeile der Fehler aufgetreten ist, und liefert Ihnen gleich die Programmzeile mit Korrektur-Cursor, so daß Sie den Fehler korrigieren können.

Betätigen Sie die Cursor-Rechts-Taste, bis sich der Cursor über einem `t` in '`printt`' befindet. Drücken Sie **[CLR]**, um das unerwünschte `t` zu entfernen und geben Sie die korrigerte Zeile mit **[RETURN]** in den Computer ein.

Nun schreiben Sie noch einmal:

```
run [RETURN]
```

...und sehen, daß der Computer Ihre Anweisung diesmal verstanden und ausgeführt hat. Auf dem Bildschirm erscheint:

```
abc
```

Zum Schluß überprüfen Sie noch einmal, ob Sie folgende Punkte unter 'WICHTIGE HINWEISE' am Beginn des Handbuches beachtet haben:

'Hinweise zur Inbetriebnahme': Punkt 4 und 5
'Hinweise zum Betrieb': Punkt 1

Teil 5: Das Laden von Programmen und Spielen

An dieser Stelle begrüßen wir wieder diejenigen Leser, die den vierten Teil überspringen konnten.

Um kurz zu demonstrieren, mit welcher Geschwindigkeit Diskettensoftware geladen werden kann, schalten Sie das Computersystem ein, und legen die Seite 4 Ihres CP/M Systemdiskettensatzes ein. Legen Sie die Diskette so ein, daß die Seite 4 oben liegt.

Tippen Sie:

```
run "rointime.dem" [RETURN]
```

Innerhalb von wenigen Sekunden wird das Programm in den Computerspeicher geladen. Sie werden jetzt gefragt, ob Sie einen Grünmonitor verwenden und müssen `Y` (yes) für ja und `N` (no) für nein eintippen. Nun wird Ihnen eine pausenlose Vorführung des Spiels 'Roland in time' (Roland im Lauf der Zeiten) gezeigt. Vielleicht sind Sie so davon begeistert, daß Sie es sich gleich kaufen möchten!

Wenn Sie sich das Vorführprogramm fertig angeschaut haben, können Sie es abbrechen, indem Sie **[CONTROL]** und **[SHIFT]** drücken und unten halten, während Sie die **[ESC]**-Taste drücken. Dadurch wird der Computer ganz zurückgesetzt, d.h. in seinen Ausgangszustand versetzt. Diese drei Tasten können Sie immer dann betätigen, wenn Sie von vorn beginnen möchten. (Um den Computer zurückzusetzen, brauchen Sie vorher nicht die Disketten herauszunehmen.)

Falls das Programm nicht geladen wurde, sehen Sie sich die Fehlermeldung auf dem Bildschirm an, um herauszufinden, was Sie falsch gemacht haben. Erscheint:

```
Drive A: disc missing
Retry, Ignore or Cancel?
```

...so haben Sie entweder die Diskette nicht richtig hineingeschoben, oder Sie haben Sie in das Laufwerk B eingelegt, falls Sie mit zwei Diskettenlaufwerken arbeiten.

```
ROINTIME.DEM not found
```

...bedeutet, daß Sie entweder die falsche Diskette eingelegt haben, bzw. die falsche Seite oben liegt, oder daß Sie den Namen des Programms, `ROINTIME.DEM`, nicht richtig geschrieben haben.

```
Bad command
```

...bedeutet wahrscheinlich, daß Sie `ROINTIME.DEM` nicht so wie angegeben geschrieben haben, sondern eine unerwünschte Leerstelle oder ein falsches Satzzeichen eingebaut haben.

```
Type mismatch
```

...bedeutet, daß Sie die Anführungszeichen ausgelassen haben.

```
Syntax error
```

...bedeutet, daß das Wort '`run`' falsch geschrieben wurde.

```
Drive A: read fail
Retry, Ignore or Cancel?
```

...bedeutet, daß der Computer die Daten auf Ihrer Diskette nicht lesen konnte. Überprüfen Sie, ob Sie die richtige Diskette eingelegt haben und drücken Sie `R` für 'Retry' ('neuer Versuch'). Diese Meldung wird immer dann erscheinen, wenn Ihre Diskette beschädigt wurde, weil sie sich noch im Laufwerk befand, als das Computersystem abgeschaltet wurde.

Sobald Sie wissen, wie man Reservekopien von Disketten macht, versäumen Sie nicht, Ihre System-Disketten zu kopieren, und machen Sie immer eine Kopie von wertvollen Programmen.

Wie die AMSTRAD-Software geladen wird

Nachdem wir Ihnen hoffentlich die unterhaltende Seite der Computerarbeit schmackhaft gemacht haben, wollen wir jetzt ein Spiel laden.

Schieben Sie ihre Software-Diskette in das Laufwerk und tippen Sie:

```
run "disc" [RETURN]
```

Nach wenigen Sekunden ist Ihre Diskette geladen, und das Spiel kann beginnen.

Versuchen Sie, die Seite 4 Ihres CP/M Systemdiskettenpakets mit dem Befehl `run "disc"` zu laden. Sie sehen und hören dann die endlose Vorführung des Demonstrationsprogramms.

Wenn Sie sich genug angeschaut haben, setzen Sie den Computer mit Hilfe der **[CONTROL]**-, **[SHIFT]**- und **[ESC]**-Tasten zurück.

Mit der Anweisung (`run "disc"`) können Sie die meisten Disketten der AMSTRAD-Software laden, mit Ausnahme weniger Fälle, in denen Sie etwas anderes eintippen müssen. Befolgen Sie deshalb immer die Ladeanweisung auf dem Etikett jeder Diskette.

Zum Schluß überprüfen Sie noch einmal, ob Sie folgende Punkte unter 'WICHTIGE HINWEISE' am Beginn des Handbuches beachtet haben:

'Hinweise zur Inbetriebnahme': Punkt 6
'Hinweise zum Betrieb': Punkt 1, 5 und 6

Teil 6: Einführung in AMSTRAD BASIC

Bis jetzt wissen Sie, wie man den Computer behandeln sollte, wie man ihn aufstellt und wie man das Zubehör anschließt. Sie kennen die Funktion einiger Tasten und können die Software laden. Im folgenden Abschnitt machen wir Sie mit einigen Anweisungen bekannt, die Sie den Computer ausführen lassen können.

Wie Sie und ich, kann der Computer nur Anweisungen in einer Sprache verstehen, die er kennt. Diese Computersprache heißt BASIC (Abkürzung für 'Beginners' All-purpose Symbolic Instruction Code'). Die BASIC-Vokabeln werden Schlüsselwörter genannt. Jedes Schlüsselwort gibt dem Computer den Auftrag, eine bestimmte Funktion zu erfüllen. Alle Sprachen haben grammatische Regeln, und auch BASIC ist in dieser Hinsicht keine Ausnahme. Grammatik wird hier als 'Syntax' bezeichnet. Der Computer sagt Ihnen freundlicherweise immer, wenn Sie einen Syntaxfehler (engl. `Syntax error`) gemacht haben.

Die wichtigsten BASIC-Befehle

Im 3. Kapitel finden Sie eine Zusammenstellung und Beschreibung aller Befehle des AMSTRAD CPC6128 BASIC. In diesem Teil stellen wir Ihnen die gebräuchlichsten BASIC-Schlüsselwörter vor.

CLS

(clear screen=Bildschirm löschen)

Um den Bildschirm zu löschen, tippen Sie:

```
cls [RETURN]
```

Wie Sie sehen, verschwindet die Schrift vom Bildschirm und oben links erscheint das Wort `Ready` mit dem Cursor ■.

Beachten Sie, daß Sie die BASIC-Schlüsselwörter sowohl in großen als auch in kleinen Buchstaben eingeben können.

PRINT

(print=drucken)

Diesen Befehl geben Sie immer dann ein, wenn Zeichen, Worte oder Zahlen auf dem Bildschirm erscheinen sollen. Schreiben Sie folgende Befehlszeile:

```
print "hallo" [RETURN]
```

Nun erscheint auf dem Bildschirm:

```
hallo
```

Die Anführungszeichen "" sagen dem Computer, was gedruckt werden soll. `hallo` wurde ausgegeben, sobald die **[RETURN]**-Taste gedrückt wurde. Schreiben Sie jetzt:

```
cls [RETURN]
```

...um den Bildschirm zu löschen.

RUN

(run=laufen)

Das vorige Beispiel enthielt eine einzelne Befehlszeile. Sobald die **[RETURN]**-Taste gedrückt und der Befehl ausgeführt war, hatte der Computer den Befehl vergessen. Man kann aber eine Reihe von Befehlen, die in einer bestimmten Reihenfolge ausgeführt werden sollen, speichern. Das heißt, man schreibt ein Programm. Die BASIC-Anweisungen in einem Programm sehen aus wie im obigen Beispiel, mit dem Unterschied, daß jede Befehlszeile mit einer Nummer versehen ist. Wenn das Programm aus mehr als einer Zeile besteht, sagen die Zeilennummern dem Computer, in welcher Reihenfolge die Anweisungen ausgeführt werden sollen. Mit **[RETURN]** wird die numerierte Programmzeile gespeichert, bis die darin enthaltene Anweisung ausgeführt wird, d.h. bis das Programm läuft. Tippen Sie jetzt folgendes ein:

```
10 print "hallo" [RETURN]
```

Sie haben bemerkt, daß diesmal **nicht** `hallo` auf dem Bildschirm erschien, als Sie die **[RETURN]**-Taste drückten. Stattdessen wurde der Befehl als einzeiliges Programm im Computer gespeichert. Um das Programm laufen zu lassen, brauchen Sie den Befehl `run`. Schreiben Sie:

```
run [RETURN]
```

Jetzt erscheint das Wort `hallo` auf dem Bildschirm.

Übrigens können Sie für das Wort print auch das **?** Fragezeichen eintippen, also:

```
10 ? "hallo" [RETURN]
```

LIST

(list=auflisten)

Wenn sich ein Programm im Speicher befindet, können Sie es überprüfen, indem Sie es auflisten lassen. Schreiben Sie:

```
list [RETURN]
```

Auf dem Bildschirm erscheint jetzt Ihr Programm:

```
10 PRINT "hallo"
```

Wie Sie sehen, ist das Wort `PRINT` großgeschrieben. Das heißt, daß der Computer `PRINT` als BASIC-Schlüsselwort erkannt hat.

Tippen Sie nun `cls` **[RETURN]** ein, um den Bildschirm frei zu machen. Obwohl Ihr Programm jetzt nicht mehr zu sehen ist, befindet es sich immer noch im Computerspeicher.

GOTO

(go to = gehe nach)

Das Schlüsselwort `GOTO` weist den Computer an, zu einer anderen Zeile zu springen, entweder, um einen Programmteil auszulassen, oder um eine sogenannte Schleife zu bilden. Schreiben Sie:

```
10 print "hallo" [RETURN]
20 goto 10 [RETURN]
```

Jetzt schreiben Sie:

```
run [RETURN]
```

...und sehen, wie links lauter `hallo`'s untereinander gedruckt werden. Das liegt daran, daß Zeile `20` dem Computer sagt, zu Zeile `10` zurückzukehren und das Programm von da ab durchzuführen.

Um den Programmablauf zu stoppen, drücken Sie einmal auf die **[ESC]**-Taste. Wenn Sie jetzt eine beliebige andere Taste drücken, fährt das Programm fort. Damit sie weitere Anweisungen eintippen können, müssen Sie zweimal **[ESC]** drücken.

Löschen Sie nun den Bildschirm mit:

```
cls [RETURN]
```

Wenn Sie möchten, daß das Wort `hallo` jede Zeile auf dem Bildschirm ausfüllt, tippen Sie das vorige Programm ein, und setzen dabei hinter das zweite Anführungszeichen ein **;** Semikolon:

```
10 print "hallo"; [RETURN]
20 goto 10 [RETURN]
run [RETURN]
```

Das Semikolon sagt dem Computer also, daß die folgende Zeichengruppe direkt hinter die vorhergehende Zeichengruppe gesetzt werden soll, vorausgesetzt, daß sie noch in die Zeile paßt.

Drücken Sie jetzt zweimal **[ESC]**, um das Programm zu verlassen. Tippen Sie wieder Programmzeile **10** ein, wobei Sie das Semikolon durch ein Komma ersetzen:

```
10 print "hallo", [RETURN]
run [RETURN]
```

Das Komma bewirkt also, daß jede Zeichengruppe 13 Spalten hinter die vorangehende Zeichengruppe gesetzt wird. Diese Möglichkeit ist nützlich, wenn man Informationen in einzelnen Spalten darstellen möchte. Geht die Anzahl der Zeichen in einer Gruppe jedoch über 12 hinaus, wird die folgende Zeichengruppe eine Spalte weiter gesetzt, so daß zwischen den Spalten stets ein Zwischenraum liegt.

Die Breite der Spalten kann mit dem Befehl `ZONE` verändert werden, der weiter hinten im Handbuch erklärt wird.

Verlassen Sie das Programm wieder, indem Sie zweimal **[ESC]** drücken. Um den Computerspeicher vollständig zu löschen, drücken Sie der Reihe nach **[CONTROL]**, **[SHIFT]** und **[ESC]** und halten diese Tasten, bis der Computer zurückgesetzt ist.

INPUT

Mit diesem Befehl wird dem Computer mitgeteilt, daß er auf eine Information warten soll, z.B. auf die Antwort auf eine Frage.

Schreiben Sie:

```
10 input "Wie alt bist du";Alter [RETURN]
20 print "Du siehst juenger als";Alter;"Jahre
   aus." [RETURN]
run [RETURN]
```

Nun erscheint auf dem Bildschirm die Frage:

```
Wie alt bist du?
```

Geben Sie Ihr Alter ein und drücken Sie **[RETURN]**. Wenn Sie z.B. 18 eintippen, sagt Ihnen der Computer:

```
Du siehst juenger als 18 Jahre aus.
```

Das Beispiel zeigt Ihnen den Gebrauch des `INPUT`-Befehls und einer Zahlenvariable. Das Wort Alter am Ende der Zeile `10` speichert der Computer als Zahlenvariable, d.h. er assoziiert es mit jeder Zahl, die eingegeben wird, wenn das Programm läuft. Er setzt dann in Zeile `20` die eingegebene Zahl für das Wort Alter ein. Wir haben als Variable das Wort Alter gewählt; wir hätten ebensogut einen Buchstaben, z.B. `b`, nehmen können.

Setzen Sie den Computer wieder mit **[CONTROL]**, **[SHIFT]** und **[ESC]** zurück, um den Speicher zu löschen. Wenn statt einer Zahl Buchstaben oder Buchstaben mit Zahlen eingegeben werden sollen, muß die Variable hinten mit einem Dollarzeichen `$` versehen werden. Solch einen Variablentyp nennt man 'Stringvariable'.

Tippen Sie nun das folgende Programm ein (wobei Sie darauf achten, daß in Zeile `20` nach dem `o` in `hallo` und vor dem `i` in `ich` eine Leerstelle bleibt):

```
10 input "Wie heisst du";Name$ [RETURN]
20 print "hallo ";Name$;" ich heisse Roland" [RETURN]
run [RETURN]
```

Der Computer fragt Sie jetzt:

```
Wie heisst du?
```

Tippen Sie Ihren Namen ein und drücken Sie **[RETURN]**.

Wenn Sie z.B. den Namen `Fritz` eingegeben haben, erscheint nun auf dem Bildschirm:

```
Hallo Fritz ich heisse Roland
```

Wir haben in diesem Beispiel für die Stringvariable den Ausdruck `Name$` gewählt; wir hätten ebensogut einen Buchstaben mit `$` , z.B. `a$` nehmen können. Wir wollen jetzt die beiden obigen Beispiele in einem Programm kombinieren.

Setzen Sie den Computer mit **[CONTROL] [SHIFT] [ESC]** zurück und tippen Sie folgendes ein:

```
5 cls [RETURN]
10 input "Wie heisst du";a$ [RETURN]
20 input "Wie alt bist du";b [RETURN]
30 print "Ich muss sagen ";a$;" du siehst nicht wie";b;
   "Jahre aus"
run [RETURN]
```

Wir haben in diesem Programm zwei Variablen verwendet, `a$` für den Namen und `b` für das Alter. Der Computer fragt Sie jetzt:

```
Wie heisst du?
```

Sie tippen jetzt Ihren Namen ein (z.B. `Fritz`), und drücken **[RETURN]**.

Nun werden Sie gefragt:

```
Wie alt bist du?
```

Sie geben Ihr Alter an (z.B. `18`) und drücken **[RETURN]**.

Wenn Sie Fritz heißen und 18 sind, lesen Sie jetzt auf dem Bildschirm:

```
Ich muss sagen Fritz du siehst nicht wie 18 Jahre aus
```

Wie man ein Programm verändert

Wenn Sie Fehler im Programm gemacht haben, und die Meldung `syntax error` oder eine andere Fehlermeldung auftaucht, brauchen Sie die fehlerhafte Zeile nicht neu zu schreiben, sondern können sie korrigieren (editieren). Um dies zu demonstrieren, tippen Sie folgendes Programm mit den Fehlern ein:

```
5 clss [RETURN]
10 input "Wie heisst du";a$ [RETURN]
20 input "Wie alt bis du";b [RETURN]
30 print "Ich muss sagen";a$;" du siehst
   nicht wie";b;"Jahre aus" [RETURN]
```

Das Programm weist drei Fehler auf:

In Zeile 5 haben wir `clss` statt `cls` geschrieben.
In Zeile 10 haben wir `bis` statt `bist` geschrieben.
In Zeile 30 haben wir eine Leerstelle zwischen sagen und den Anführungszeichen vergessen.

Es gibt im wesentlichen drei Methoden, ein Programm zu korrigieren. Erstens kann die fehlerhafte Zeile neu geschrieben werden. Wenn Sie eine Zeile neu geschrieben und die **[RETURN]**-Taste gedrückt haben, ersetzt diese Zeile die im Speicher vorhandene Zeile mit derselben Nummer.

Zweitens gibt es die EDIT-Methode.

Drittens gibt es die COPY-Cursor-Methode.

Die EDIT-Methode

Um den Fehler in Zeile 5 zu korrigieren, tippen Sie:

```
edit 5 [RETURN]
```

Nun erscheint Zeile 5 mit dem Cursor auf dem `c` in `clss`.

Um das überflüssige s in clss zu streichen, betätigen Sie die Cursor-Rechts-Taste, bis der Cursor auf dem letzten s liegt, und drücken die **[CLR]**-Taste. Schon ist das `s` verschwunden. Wenn Sie jetzt **[RETURN]** drücken, wird die korrigierte Version gespeichert. Überprüfen Sie mit:

```
list [RETURN]
```

...ob Zeile 5 nun richtig ist.

Mit dem `AUTO`-Befehl, der später noch erläutert wird, können mehrere aufeinanderfolgende Zeilen nach der `EDIT`-Methode geändert werden.

Die COPY-Cursor-Methode

Der COPY-Cursor ist ein zweiter Cursor, der auf dem Bildschirm erscheint, wenn Sie **[SHIFT]** und gleichzeitig eine der vier Cursor-Tasten drücken. Er läßt sich in alle vier Richtungen bewegen, während der Haupt-Cursor an seinem Platz bleibt.

Um die Fehler in Zeile 10 und 30 zu berichtigen, drücken Sie die **[SHIFT]**-Taste und dann die Cursor-Oben-Taste, bis sich der COPY-Cursor am Anfang von Zeile 10 befindet. Sie sehen, daß der Haupt-Cursor sich nicht bewegt hat, so daß Sie nun zwei Cursor auf dem Bildschirm haben. Jetzt drücken Sie die **[COPY]**-Taste, bis der Cursor auf der Leerstelle zwischen bis und du steht. Sie werden bemerken, daß Zeile 10 unten noch einmal geschrieben wird, und daß der Haupt-Cursor an der gleichen Stelle steht wie der COPY-Cursor. Jetzt tippen Sie das fehlende t ein. Es erscheint nur in der untersten Zeile.

Der Haupt-Cursor hat sich bewegt, während der COPY-Cursor an seinem Platz blieb. Drücken Sie jetzt die **[COPY]**-Taste, bis die gesamte Zeile kopiert ist. Sobald Sie **[RETURN]** drücken, wird die neue Zeile 10 gespeichert. Der COPY-Cursor verschwindet und der Haupt-Cursor erscheint unter der neuen Zeile 10. Um den zweiten Fehler zu korrigieren, drücken Sie **[SHIFT]** und die Cursor-Oben-Taste, bis der COPY-Cursor am Anfang der Zeile 30 steht.

Drücken Sie **[COPY]**, bis der COPY-Cursor auf dem Anführungszeichen neben sagen steht. Schlagen Sie einmal die Leertaste an. In der untersten Zeile wird nun eine Leerstelle eingeschoben. Halten Sie die **[COPY]**-Taste solange unten, bis die gesamte Zeile 30 kopiert ist, und drücken Sie **[RETURN]**.

Um zu überprüfen, ob die korrigierte Version des Programms gespeichert wurde, können das Programm auflisten lassen mit dem Befehl:

```
list [RETURN]
```

Hinweis: Um den Cursor beim Korrigieren schnell an den Anfang oder das Ende der Zeile zu bewegen, drücken Sie **[CTRL]** und einmal die Cursor-Links- bzw. Cursor-Rechts-Taste.

Setzen Sie den Computer nun mit **[CONTROL] [SHIFT] [ESC]** zurück.

IF

Mit `IF` (wenn) wird der Computer gefragt, ob eine bestimmte Bedingung vorliegt. Wenn ja, wird er mit `THEN` (dann) angewiesen, einen bestimmten Auftrag zu erfüllen. In der Befehlszeile:

```
if 1+1=2 then print "richtig" [RETURN]
```

z.B. wird der Computer prüfen, ob 1+1=2 ist und dann entsprechend reagieren.

Der Befehl `ELSE` kann zusammen mit `IF` und `THEN` verwendet werden, um dem Computer zu sagen, was er tun soll, wenn die Bedingung nicht erfüllt ist, z.B.:

```
if 1+1=0 then print "richtig" else print "falsch" [RETURN]
```

Wir werden die Befehle `IF` und `THEN` jetzt in unser Programm einbauen.

Beachten Sie dabei, daß wir zwei neue Symbole eingeführt haben: `<` bedeutet kleiner als (neben der **M**-Taste) und `>` bedeutet größer als (neben der `<`-Taste). Schreiben Sie also:

```
5 CLS
10 INPUT "Wie heisst du";a$
20 INPUT "Wie alt bist du";Alter
30 IF Alter <13 THEN 60
40 IF Alter <20 THEN 70
50 IF Alter >19 THEN 80
60 PRINT "Na ";a$;", mit";Alter;"Jahren bist du noch k
   ein Teenager":END
70 PRINT "So,so ";a$;", mit";Alter;"Jahren bist du ja
   ein Teenager":END
80 PRINT "Na ";a$;", mit";Alter;"Jahren bist du kein T
   eenager mehr"
```

Überprüfen Sie, ob Sie das Programm richtig eingegeben haben, mit:

`list` **[RETURN]**

Jetzt tippen Sie:

`run` **[RETURN]**

Beantworten Sie die Fragen, die der Computer Ihnen stellt und warten Sie ab, was passiert.

Sie haben also gesehen, was durch die Befehle `IF` und `THEN` im Programm bewirkt wird. Am Ende der Zeilen `60` und `70` haben wir das Wörtchen `END` angehängt. `END` ist ein Schlüsselwort, mit dem das Programm beendet wird. Hätten wir es in Zeile `60` ausgelassen, wären auch die Zeilen `70` und `80` ausgeführt worden. Und hätten wir es in Zeile `70` ausgelassen, wäre auch Zeile `80` ausgeführt worden.

Der Doppelpunkt `:` vor `END` trennt den Befehl von der vorangehenden Anweisung. Doppelpunkte werden verwendet, um zwei oder mehr Befehle in einer Programmzeile unterzubringen. Außerdem haben wir das Programm um die Zeile `5` ergänzt, um den Bildschirm am Anfang des Programms frei zu machen. Wir werden dies von jetzt an in den folgenden Programmen immer tun, damit unsere Arbeit sauberer aussieht.

Setzen Sie den Computer mit **[CONTROL] [SHIFT] [ESC]** zurück.

FOR und NEXT

(for...to = für...bis; next=nächstes)

Die Befehle FOR und NEXT werden verwendet, wenn man eine bestimmte Operation mehrere Male durchführen lassen will. Die Anweisungen für diese Operation werden in eine FOR NEXT-Schleife eingeschlossen.

Schreiben Sie:

```
5 cls
10 for a=1 to 10 [RETURN]
20 print "Operationsnummer";a [RETURN]
30 next a [RETURN]
run [RETURN]
```

Sie sehen, daß die Anweisung in Zeile 20 zehnmal durchgeführt wurde, wie im FOR-Befehl in Zeile 10 angegeben. Der Wert der Variablen wurde dabei jedesmal um 1 erhöht.

Mit dem Schlüsselwort STEP (Schritt) können Sie bestimmen, um welchen Wert die Variable in der FOR NEXT-Anweisung bei jedem Durchgang erhöht werden soll. Ändern Sie z.B. Zeile 10 um in:

```
10 for a=10 to 50 step 5 [RETURN]
run [RETURN]
```

Sie können Rückwärtsschritte machen, z.B.:

```
10 for a=100 to 0 step -10 [RETURN]
run [RETURN]
```

REM

REM ist die Abkürzung für das englische Wort 'remark' (= Anmerkung). Alles, was hinter diesem Schlüsselwort in derselben Zeile steht, wird vom Computer ignoriert und stellt nur eine Gedächtnisstütze für Sie dar. Es kann sich z.B. um einen Programmtitel oder um die Bedeutung einer Variablen handeln:

```
10 REM Zap the invaders [RETURN]
20 L=5 :REM Anzahl der Menschenleben [RETURN]
```

Statt :REM können Sie auch ein einfaches Anführungszeichen ' (drücken Sie **[SHIFT]** und die **7**-Taste) einsetzen:

```
10 'Zap the invaders [RETURN]
20 L=5 'Anzahl der Menschenleben [RETURN]
```

GOSUB

(go subroutine = gehe zum Unterprogramm)

Wenn Sie ein Programm schreiben, in dem eine oder mehrere Anweisungen öfter durchgeführt werden sollen, brauchen Sie die entsprechenden Zeilen nicht an all diesen Stellen im Programm einzutippen. Stattdessen können Sie die Anweisungen zu einem Unterprogramm (auch Subroutine genannt) zusammenfassen, das bei Bedarf mit dem Befehl GOSUB, gefolgt von der ersten Zeilennummer des Unterprogramms, aufgerufen wird. Das Ende des Unterprogramms wird durch den Befehl RETURN gekennzeichnet. An diesem Punkt angelangt, wird der Computer zu der Anweisung zurückkehren, die dem Aufruf des gerade absolvierten Unterprogramms folgt.

Die folgenden beiden Programme sollen Ihnen die Funktion des GOSUB-Befehls verdeutlichen. Da in diesen Programmen nichts weiter 'passiert', sondern lediglich ein Text auf den Bildschirm geschrieben wird, brauchen Sie sich nicht die Mühe zu machen, sie einzutippen:

```
10 MODE 2 [RETURN]
20 PRINT "This old man he played one" [RETURN]
30 PRINT "He played knick-knack on my drum" [RETURN]
40 PRINT "With a knick-knack paddy wack" [RETURN]
50 PRINT "Give a dog a bone" [RETURN]
60 PRINT "This old man came rolling home" [RETURN]
70 PRINT [RETURN]
80 PRINT "This old man he played two" [RETURN]
90 PRINT "He played knick-knack on my shoe" [RETURN]
100 PRINT "With a knick-knack paddy wack" [RETURN]
110 PRINT "Give a dog a bone" [RETURN]
120 PRINT "This old man came rolling home" [RETURN]
130 PRINT [RETURN]
140 PRINT "This old man he played three" [RETURN]
150 PRINT "He played knick-knack on my knee" [RETURN]
180 PRINT "With a knick-knack paddy wack" [RETURN]
190 PRINT "Give a dog a bone" [RETURN]
200 PRINT "This old man came rolling home" [RETURN]
210 PRINT [RETURN]
run [RETURN]
```

Sie sehen, daß in diesem Programm drei Zeilen mehrere Male, z.B. zwischen Zeile 180 und 200, wiederholt werden. Diesen Refrain können wir zu einem Unterprogramm machen, das wir mit dem Befehl RETURN abschließen. Das Unterprogramm können wir nun mit der Anweisung GOSUB 180 jederzeit (am Ende jeder Strophe) aufrufen. Dann sieht das Programm so aus:

```
10 MODE 2 [RETURN]
20 PRINT "This old man he played one" [RETURN]
30 PRINT "He played knick-knack on my drum" [RETURN]
40 GOSUB 180 [RETURN]
80 PRINT "This old man he played two" [RETURN]
90 PRINT "He played knick-knack on my shoe" [RETURN]
100 GOSUB 180 [RETURN]
140 PRINT "This old man he played three" [RETURN]
150 PRINT "He played knick-knack on my knee" [RETURN]
160 GOSUB 180 [RETURN]
170 END [RETURN]
180 PRINT "With a knick-knack paddy wack" [RETURN]
190 PRINT "Give a dog a bone" [RETURN]
200 PRINT "This old man came rolling home" [RETURN]
210 PRINT [RETURN]
220 RETURN [RETURN]
run [RETURN]
```

Sehen Sie, wieviel mühevolle Schreibarbeit wir uns dadurch hätten sparen können? In einem wohlstrukturierten Programm sind Unterprogramme unerläßlich. Sie gehören zu einem guten Programmierstil.

Wenn Sie Unterprogramme schreiben, denken Sie daran, daß Sie nicht unbedingt jedes Mal dieselbe Stelle des Unterprogramms aufrufen müssen, d.h. seine erste Zeile. Ein Unterprogramm zwischen Zeile 500 und 800 kann z.B. mit folgenden Anweisungen aufgerufen werden: GOSUB 500, oder GOSUB 640, oder GOSUB 790.

Beachten Sie im obigen Programm den Befehl END in Zeile 170. Würde er fehlen, so würde das Programm nach Zeile 160 weiterlaufen und die Anweisung in Zeile 180 ausführen. Dies ist jedoch nur wünschenswert, wenn die Zeile mit einem GOSUB-Befehl aufgerufen wird.

Einfache Arithmetik

Ihr Computer kann sehr einfach als Rechner eingesetzt werden. Die folgenden Beispiele erläutern das näher. Wir verwenden in diesem Abschnitt das Fragezeichen für den Befehl `PRINT`. Das Ergebnis erscheint auf dem Bildschirm, sobald die **[RETURN]**-Taste gedrückt wird.

Addition

(drücken Sie für 'plus' die **[SHIFT]**- und **;** Tasten)

Tippen Sie:

```
?3+3 [RETURN]
 6
```

(Das Gleichheitszeichen = wird NICHT eingegeben)

Tippen Sie:

```
?8+4 [RETURN]
 12
```

Subtraktion

(drücken Sie für 'minus' die = Taste ohne **[SHIFT]**)

Tippen Sie:

```
?4-3 [RETURN]
 1
```

Tippen Sie:

```
?8-4 [RETURN]
 4
```

Multiplikation

Benutzen Sie das Sternchen * als Multiplikationszeichen. (**:** Taste mit **[SHIFT]**)
Tippen Sie:

```
?3*3 [RETURN]
 9
```

Tippen Sie:

```
?8*4 [RETURN]
 32
```

Division

Benutzen Sie den Schrägstrich / als Divisionszeichen. (? Taste ohne **[SHiFT]**)

Tippen Sie:

```
?3/3 [RETURN]
 1
```

Tippen Sie:

```
?8/4 [RETURN]
 2
```

Ganzzahlige Division

Benutzen Sie diesen Schrägstrich \ für die Division ohne Rest.

Tippen Sie:

```
?10\6 [RETURN]
 1
```

Tippen Sie:

```
?20\3 [RETURN]
 6
```

Modulo

Modulo nennt man den Rest, der nach einer ganzzahligen Division bleibt.
Benutzen Sie `MOD`, um die Restzahl zu erhalten.
Tippen Sie:

```
?10 MOD 4 [RETURN]
 2
```

Tippen Sie:

```
?9 MOD 3 [RETURN]
 0
```

Quadratwurzel

Benutzen Sie `sqr()`, um die Quadratwurzel zu erhalten. Die Zahl, aus der die Wurzel gezogen werden soll, muß in Klammern stehen.

Tippen Sie:

```
?sqr(16) [RETURN]  (Dies bedeutet √16)
 4
```

Tippen Sie:

```
?sqr (100) [RETURN]
 10
```

Potenzen

Potenzieren nennt man die Rechnungsart, bei der eine Zahl so oft mit sich selbst multipliziert wird, wie der Exponent (die Hochzahl) angibt.

Benutzen Sie den Pfeil ↑ zum Potenzieren. (↑ Taste ohne **[SHIFT]**)

Tippen Sie:

```
?3↑3 [RETURN]  (Dies bedeutet 3^3)
 27
```

Tippen Sie:

```
?8↑4 [RETURN]  (Dies bedeutet 8^4)
 4096
```

Kubikwurzel

Zum Berechnen der Kubikwurzel verwenden Sie eine ähnliche Methode wie beim Potenzieren.

Um die Kubikwurzel von 27 ($\sqrt[3]{27}$) zu finden, tippen Sie:

```
?27↑(1/3) [RETURN]
 3
```

Tippen Sie:

```
?125↑(1/3) [RETURN]
 5
```

Gemischte Rechenoperationen

Der Computer führt auch gemischte Rechenoperationen durch, jedoch müssen bestimmte Prioritäten beachtet werden.

Höchste Priorität hat die Multiplikation, danach folgen Division, Addition und Subtraktion. Diese Prioritätenliste gilt jedoch nur, solange es sich ausschließlich um diese vier Rechenarten handelt.

Nehmen wir folgende Aufgabe:

```
3+7-2*7/4
```

Vielleicht denken Sie, daß sie so gerechnet wird:

```
3+7-2 *7 /4
=   8 *7 /4
=     56 /4
=        14
```

Doch der Computer rechnet folgendermaßen:

```
3+7-2*7/4
=3+7-14/4
=3+7-3.5
=10-3.5
=6.5
```

Überprüfen Sie dies, indem Sie die Aufgabe wie oben angegeben eintippen:

```
?3+7-2*7/4 [RETURN]
 6.5
```

Die Reihenfolge der Berechnung kann verändert werden, wenn Sie Klammern verwenden. Der Ausdruck in Klammern hat Vorrang vor der Multiplikation und den anderen Rechenarten außerhalb der Klammern. Überzeugen Sie sich, indem Sie die Aufgabe so schreiben:

```
?(3+7-2)*7/4 [RETURN]
 14
```

Die Prioritätenliste aller mathematischen Operatoren ist:

↑	Potenzierung
MOD	Modulo
-	unäres Minus (es bezeichnet eine Zahl als negativ)
* und /	Multiplikation und Division
\	ganzzahlige Division
+ und -	Addition und Subtraktion

Andere Exponenten

Falls Sie mit sehr großen oder sehr kleinen Zahlen rechnen, ist eine andere Schreibweise vorteilhafter. Der Buchstabe E wird als Potenz zur Basis 10 verstanden. Es ist hier gleichgültig, ob sie ein großes E oder ein kleines e schreiben.

Die Zahl 300 z.B. kann auch als $3x10^2$ geschrieben werden. In der wissenschaftlichen Schreibweise heißt es 3E2. Ähnlich verhält es sich mit 0.03 (auch $3x10^{-2}$): Die wissenschaftliche Schreibweise ist 3E-2. Probieren Sie folgende Beispiele aus:

Sie können eintippen:

```
?30*10 [RETURN]
 300
```

...oder Sie können eintippen:

```
?3E1*1E1 [RETURN]
 300

?3000*1000 [RETURN] ... oder: ?3E3*1E3 [RETURN]
 3000000

?3000*0.001 [RETURN] ... oder: ?3E3*3E-3 [RETURN]
 3
```

Teil 7: Speichern

Nachdem Sie nun mit dem Eintippen einiger Befehle die ersten Fingerübungen absolviert haben, möchten Sie wahrscheinlich wissen, wie man ein Programm auf Diskette speichert, und wie man es von der Diskette wieder in den Computer lädt.

Auch wenn Sie das Speichern und Laden schon vom Kassettenbetrieb her kennen, sollten Sie einige wichtige Besonderheiten beachten, bevor Sie Programme auf Diskette abspeichern.

Es gibt zwei wesentliche Unterschiede zwischen dem Diskettenbetrieb und Kassettenbetrieb. Erstens kann man eine neue, unbespielte Diskette nicht einfach auspacken und bespielen wie eine Kassette. Eine neue Diskette muß erst 'formatiert' werden, was wir gleich näher erläutern werden.

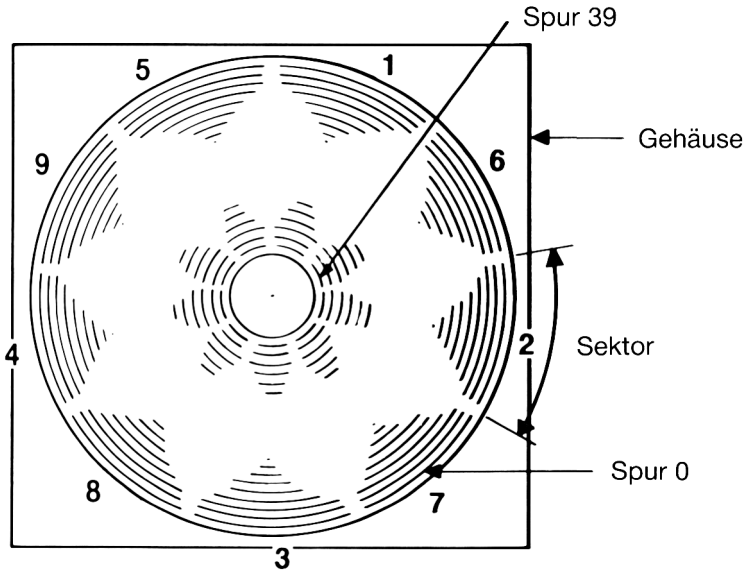
Zweitens müssen Disketten-Dateien korrekt benannt werden. Es gibt nicht viele Regeln für die Benennung von Dateien, die auf Kassette gespeichert werden; die Dateinamen können sehr verschieden lang sein oder auch ausgelassen werden. Nicht so bei Disketten-Dateien. Dateinamen für die Diskettenspeicherung müssen genau der CP/M Norm entsprechen, wie wir noch weiter ausführen werden.

Was heißt formatieren?

Bevor Sie irgendwelche Daten auf eine neue, unbespielte Diskette schreiben, muß die Diskette formatiert werden.

Sie können sich das so vorstellen, daß die Diskette erst mit einem Regalsystem ausgestattet werden muß, auf denen die Information dann abgelegt werden kann. Mit anderen Worten, die Diskette muß erst in systematische Teile gegliedert werden, denen die Daten dann zugeordnet werden können.

Beim Formatieren wird die Diskette in 360 Teile gegliedert:



Die Diskette hat ähnlich wie eine Schallplatte 40 Spuren, von Spur 0 (ganz außen) bis Spur 39 (ganz innen), außerdem wird sie wie ein Kuchen in 9 Sektoren eingeteilt.

Jede Spur eines Sektors kann 512 Bytes Daten speichern. Insgesamt beträgt der Speicherplatz pro Diskettenseite also 180 KBytes.

Das Formatieren mit den CP/M Systemdisketten

Um also eine neue, unbespielte Diskette gebrauchsfertig zu machen, müssen Sie sie formatieren. Dazu verwenden Sie die Seite 1 Ihres Systemdiskettensatzes, der Ihnen beim Kauf des Computers übergeben wurde.

Schalten Sie das Computersystem an und schieben Sie die Seite 1 Ihres Systemdiskettensatzes in das Laufwerk.

Wenn Sie mit zwei Diskettenlaufwerken arbeiten, schieben Sie Seite 1 immer in das eingebaute Hauptlaufwerk (Laufwerk A).

Tippen Sie:

```
|cpm [RETURN]
```

(Den Balken | finden Sie, indem Sie **[SHIFT]** und die @ Taste drücken.)

Nach wenigen Sekunden erscheint auf dem Bildschirm folgende Meldung:

```
CP/M Plus      Amstrad Comsumer Electronics plc
```

Diese Meldung sagt Ihnen, daß der Computer jetzt unter der Kontrolle des CP/M Plus -Betriebssystems steht. Außerdem sehen Sie den Buchstaben A› mit dem Cursor. Dies ist ein 'Bereitschaftszeichen', ähnlich wie `Ready` beim normalen BASIC-Betrieb. Es sagt Ihnen, daß der Computer auf weitere Anweisungen von Ihnen wartet.

Wenn Sie mit CP/M arbeiten, können Sie dem Computer keine BASIC-Befehle geben, weil er sie dann nicht versteht.

Wenn Sie z.B.:

```
cls [RETURN]
```

...eintippen, wird der Computer Ihren Befehl mit einem Fragezeichen wiederholen:

```
CLS?
```

...um Ihnen kund zu tun, daß er diesen Befehl nicht versteht.

Sehen wir uns also einige CP/M-Befehle an. Schreiben Sie:

```
dir [RETURN]
```

Auf dem Bildschirm sehen Sie jetzt ein Inhaltsverzeichnis (directory) mit den Dienstprogrammen (utility COMmands) von CP/M. Eines davon heißt `DISCKIT3`. Schreiben Sie:

```
disckit3 [RETURN]
```

Nach wenigen Augenblicken sehen Sie oben auf dem Bildschirm die DISC KIT Eröffnungsmeldung, gefolgt von:

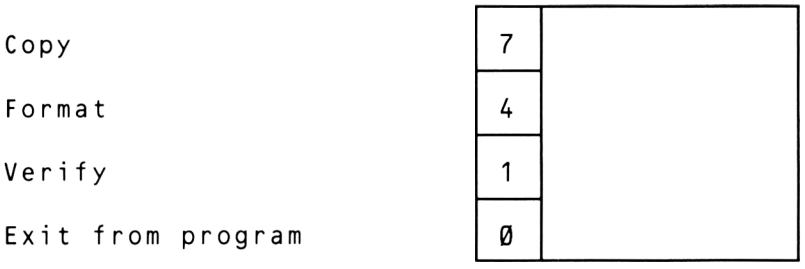
```
One drive found
```

Dies bedeutet, daß Sie das DISC KIT- Dienstprogramm laufen haben, und daß Sie nur ein Diskettenlaufwerk - das eingebaute - in Betrieb haben.

Haben Sie ein zusätzliches Diskettenlaufwerk angeschlossen, so lautet die Meldung:

```
Two drives found
```

Weiter unten auf dem Bildschirm sehen Sie folgendes Bild:



Dies ist das sogenannte DISC KIT Hauptmenü. Die Nummern im Kasten beziehen sich auf die Funktionstasten rechts auf der Tastatur (**fØ**, **f1**, **f4** und **f7**). Wenn Sie eine dieser Tasten drücken, wird die zugehörige Anweisung ausgeführt.

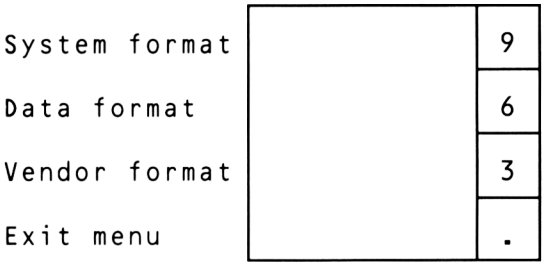
Wenn Sie zu diesem Zeitpunkt Taste **fØ** drücken, verlassen Sie das DISC KIT-Programm und landen wieder beim Direktmodus der CP/M Bedienkonsole, dem Bereitschaftszeichen `A>`.

Wir möchten die Diskette formatieren und müssen daher Taste **f4** drücken.

ACHTUNG!

Das Formatieren beschriebener Disketten löscht ihrenInhalt.

Nun erscheint ein neues Menü, das verschiedene Formate zur Auswahl anbietet:



Sie drücken auf die entsprechende Taste (**f3**, **f6** oder **f9**) für das gewünschte Format. Die verschiedenen Formate werden weiter hinten im Handbuch erklärt. Wir wählen vorläufig das Data-Format, indem wir die Taste **f6** drücken.

Wenn Sie an dieser Stelle die **.** Taste drücken, verlassen Sie den Format-Modus und kehren zum DISC KIT-Menü zurück.

Haben Sie also die Taste **f6** gedrückt, erscheint - vorausgesetzt, Sie haben kein zusätzliches Diskettenlaufwerk angeschlossen - folgende Meldung auf dem Bildschirm:




```
Format as Data
    Any other key to exit menu
```

Jetzt müssen Sie die CP/M Systemdiskette herausnehmen und die Diskette einlegen, die Sie formatieren wollen. Die Seite, die formatiert werden soll, muß oben liegen.

Jetzt drücken Sie die **Y**-Taste (`Y` steht für 'ja, die Diskette soll formatiert werden'.)

Die Diskette wird von Spur 0 bis Spur 39 formatiert, wobei oben links auf dem Bildschirm die Nummer der Spur angezeigt wird, die gerade bearbeitet wird.

Disketten mit offenem Schreibschutzloch können nicht formatiert werden. Falls Sie das versuchen, erscheint folgende Meldung:

```
Disc write-protected
Insert disc to format
R-etry or C-ancel
```

Daraufhin sollten Sie Ihre Anweisung zurücknehmen, indem Sie `C` für Cancel eintippen. Dann legen Sie die richtige Diskette mit geschlossenem Schreibschutzloch zum Formatieren ein.

Sorgen Sie dafür, daß das Schreibschutzloch von Disketten mit Programmen, die Sie behalten wollen, immer offen ist, und schließen Sie NIEMALS die Schreibschutzlöcher Ihrer CP/M-Systemdisketten!

Nach Beendigung des Formatierens sehen Sie folgende Mitteilung:

```
Format completed
Remove disc
Press any key to continue
```

Sie sagt Ihnen, daß das Formatieren beendet ist und Sie die Diskette herausnehmen sollen. Danach müssen Sie irgendeine Taste drücken.

Wenn Sie möchten, können Sie nun eine andere Diskette zum Formatieren einlegen und wieder `Y` drücken. Sie können den Vorgang beliebig oft wiederholen, bis Sie alle Disketten formatiert haben, die dieses Format erhalten sollen.

Sind Sie mit dem Formatieren fertig, drücken Sie irgendeine Taste (außer `Y`) und kehren zum DISC KIT-Haupt-Menü zurück.

Die beiden anderen Wahlmöglichkeiten des Menüs, `Copy` und `Verify`, werden wir später besprechen. Für's erste reicht es, was Sie jetzt über das Formatieren gelernt haben. Setzen Sie den Computer nun mit **[CONTROL] [SHIFT] [ESC]** zurück.

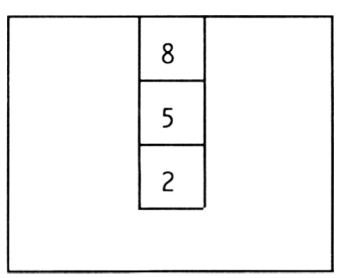
Bewahren Sie Ihre CP/M-Systemdisketten immer an einem sicheren Ort auf, denn sie sind im wahrsten Sinne des Wortes der Schlüssel zu Ihrem System. Weiter hinten im Handbuch sagen wir Ihnen, wie Sie 'Arbeitskopien' von Ihren Systemdisketten machen können, so daß Sie Original-Systemdisketten gut wegschließen können.

Formatieren auf einem Disketten-Doppellaufwerk

Gehen Sie wie oben beschrieben vor: Wählen Sie `Format` aus Ihrem DISC KIT-Menü, indem Sie die Taste **f4** drücken, und wählen Sie anschließend `Data format`, indem Sie Taste **f6** drücken.

An dieser Stelle wird Ihnen ein drittes Menü gezeigt, das Ihnen die beiden Laufwerke, auf denen Sie formatieren können, zur Auswahl anbietet:

```
Format A:
Format B:
Exit menu
```



Wenn Sie sich für `Format B:` (Taste **f5**) entscheiden, können Sie Ihre Systemdiskette (Seite 1) im Laufwerk A lassen, während Sie die Diskette, die Sie formatieren wollen, ins Laufwerk B einlegen.

Nachdem Sie also Taste **f5** gedrückt haben, können Sie entweder `Y` drücken, um den Formatiervorgang einzuleiten, oder eine andere Taste anschlagen, um zum DISC KIT-Menü zurückzukehren.

Haben Sie `Format A:` (Taste **f8**) gewählt, müssen Sie UNBEDINGT daran denken, Ihre Systemdiskette aus dem Laufwerk A herauszunehmen, und dafür die Diskette einzulegen, die formatiert werden soll.

Denken Sie daran: GEHEN SIE NIEMALS DAS RISIKO EIN, IHRE CP/M SYSTEMDISKETTEN ZU ÜBERSCHREIBEN!

Wir haben jetzt also eine formatierte, unbeschriebene Diskette (oder zwei), auf die wir BASIC-Programme übertragen können, um sie später wieder abzurufen.

Speichern von Programmen auf Diskette

Wenn Sie ein Programm in den Computerspeicher geschrieben haben, können Sie es auf Diskette speichern mit dem Befehl:

```
save "datei" [RETURN]
```

Beachten Sie, daß das Programm benannt werden muß.

Der Dateiname für die Diskettenspeicherung besteht aus zwei Teilen (Feldern). Im ersten Feld muß eine Bezeichnung eingetragen werden, die aus bis zu acht Buchstaben und/oder Ziffern bestehen kann. Es dürfen keine Leerstellen oder Satzzeichen darin vorkommen. Das erste Feld enthält gewöhnlich den Namen des Programms.

Es steht Ihnen frei, im zweiten Feld etwas einzutragen. Die Bezeichnung im zweiten Feld kann aus drei Buchstaben und/oder Ziffern bestehen, wiederum ohne Leerstellen oder Satzzeichen. Die beiden Felder werden durch einen Punkt . getrennt.

Wenn Sie im zweiten Feld nichts angeben, wird das System automatisch seine eigene Bezeichnung einsetzen, z.B. `.BAS` für BASIC-Datei oder `.BIN` für Binär-(Maschinencode-) Datei.

Um auszuprobieren, wie etwas auf Diskette gespeichert wird, schreiben Sie ein kurzes Programm, legen eine formatierte Diskette ins Diskettenlaufwerk und tippen:

```
save "Beispiel" [RETURN]
```

Nach wenigen Sekunden erscheint die Bereitschaftsmeldung `Ready` auf dem Bildschirm, d.h. das Programm ist auf Diskette gespeichert worden. (Wenn nicht, sehen Sie sich die Fehlermeldung auf dem Bildschirm an, um festzustellen, ob Sie vielleicht die Diskette nicht in das richtige Laufwerk geschoben haben, oder ob Sie vergessen haben, das Schreibschutzloch zu schließen, oder ob Sie den Befehl fehlerhaft eingetippt haben.)

Das Inhaltsverzeichnis

Nachdem Sie das Programm gespeichert haben, können Sie sich das Inhaltsverzeichnis (catalog) der Diskette ansehen, indem Sie eintippen:

```
cat [RETURN]
```

Auf dem Bildschirm erscheint jetzt der Name der Datei, einschließlich der näheren Bezeichnung im zweiten Feld. Außerdem wird die Länge der Datei (zum nächsthöheren K-Byte aufgerundet) und der Umfang des verbliebenen Speicherplatzes auf der Diskette angezeigt:

```
Drive A: user 0

BEISPIEL.BAS 1K

177K free
```

Laden von Diskette

Programme können mit folgenden Befehlen von Diskette geladen und anschließend gestartet werden:

```
load "programm" [RETURN]
run [RETURN]
```

...oder sie können direkt gestartet werden mit:

```
run "programm" [RETURN]
```

Hinweis: Geschützte Programme können nur direkt gestartet werden.

|A UND |B

Wenn Sie mit einem zusätzlichen Diskettenlaufwerk arbeiten, können Sie bestimmen, welches Laufwerk (A oder B) Sie ansprechen, indem Sie den Befehlen SAVE, CAT oder LOAD

```
|a [RETURN]
```

oder

```
|b [RETURN]
```

voranstellen.

Kopieren von Programmen von Diskette auf Diskette

Mit den in diesem Abschnitt vorgestellten Befehlen läßt sich ein Programm mühelos von einer Diskette auf eine andere kopieren. Das Programm wird von der Originaldiskette in den Computerspeicher geladen, danach wird die Originaldiskette herausgenommen und das Programm vom Speicher auf die neue Diskette geschrieben.

Wenn Sie zwei Diskettenlaufwerke betreiben, um ein Programm von einer Diskette auf die andere zu kopieren, können Sie die Originaldiskette z.B. in das Laufwerk B, und die Kopierdiskette in das Laufwerk A einlegen. Dann schreiben Sie:

```
|b [RETURN]
load "programm" [RETURN]
|a [RETURN]
save "programm" [RETURN]
```

Mit dem CPC6128 können Dateien auf vier verschiedene Arten gespeichert werden. Neben dem gewöhnlichen Speicherverfahren von BASIC-Dateien mit:

```
save "programm" [RETURN]
```

...haben Sie drei weitere Methoden für spezielle Zwecke zur Verfügung:

ASCII Dateien

```
save "programm",a [RETURN]
```

Der nachgestellte Ausdruck `,a` weist den Computer an, die Datei in Form einer ASCII Textdatei zu speichern. Diese Art der Datenspeicherung wird normalerweise in der Textverarbeitung und bei anderen Anwenderprogrammen benutzt. Eine detaillierte Beschreibung folgt, wenn diese Programme erklärt werden.

Geschützte Dateien

```
save "programm",p [RETURN]
```

Mit dem nachgestellten Ausdruck `,p` wird der Computer angewiesen, die Daten in codierter Form zu speichern. Ein auf diese Weise gespeichertes Programm kann nach dem Laden nicht aufgeLISTet werden, und wenn man es bei der Durchführung mit **[ESC]** stoppt, wird es gelöscht.

Mit ,p gespeicherte Programme können nur mit den Befehlen:

```
run "programm" [RETURN]
```

...und:

```
chain "programm" [RETURN]
```

...direkt gestartet werden.
Bevor Sie diese Form der Speicherung wählen, sollten Sie sich eine ungeschützte Aufzeichnung machen, um sich die Möglichkeit von Änderungen offenzuhalten.

Binärdateien

```
save "programm",b, ‹Startadresse›,‹Dateilänge›
                                    [,‹Anfangspunkt›] [RETURN]
```

Damit können Sie das Programm, so wie es im RAM des Computers gespeichert ist, in binärer Form auf Diskette übertragen. Dazu muß der Computer wissen, an welcher Stelle (Adresse) im Speicher sich die aufzunehmenden Daten befinden, wie lang die Datensammlung ist, und, wenn nötig, an welchem Punkt die Programmausführung beginnen soll.

Bildschirmabdruck (screen dump)

Mit dieser Methode der binären Speicherung können Bildschirmanzeigen direkt auf Diskette aufgenommen werden. Der Befehl:

```
save "scrndump",b,49152,16384 [RETURN]
```

speichert das Bild so, wie Sie es sehen. Es fängt bei der Adresse 49152 im Bildschirmspeicher an, und die Länge des Bildschirmspeicherbereichs, der übertragen werden soll, beträgt 16384. Mit dem Befehl:

```
load "scrndump" [RETURN]
```

...holen Sie das Bild wieder auf den Bildschirm.

Weitere Informationen über das Übertragen von Programmdateien auf Disketten (und Kassetten) finden Sie weiter hinten im Handbuch.

Zum Schluß überprüfen Sie noch einmal, ob Sie folgende Punkte unter 'WICHTIGE HINWEISE' am Beginn des Handbuches beachtet haben:

'Hinweise zur Inbetriebnahme': Punkt 5, 6 und 7
'Hinweise zum Betrieb': Punkt 1, 2, 3, 4, 5, 6, 7 und 9

Teil 8: Bildschirm-Modus, Farben und Graphik

Auf dem AMSTRAD CPC6128 Colour Personal-Computer sind drei verschiedene Darstellungsarten (Modi) möglich: Modus 0, Modus 1 und Modus 2.

Nach dem Einschalten des Computers ist automatisch Modus 1 eingestellt.

Um den Unterschied zwischen den drei Modi zu sehen, schalten Sie den Computer ein und drücken die Taste mit der 1. Halten Sie die Taste, bis zwei Zeilen mit Einsern vollgeschrieben sind. Wenn Sie jetzt zählen, sehen Sie, daß in jeder Zeile 40 Ziffern stehen. Das heißt, daß im Modus 1 vierzig Spalten zur Verfügung stehen. Drücken Sie jetzt **[RETURN]** - Sie bekommen zwar einen Syntaxfehler, aber der stört uns nicht. Wir kommen auf diese Weise schneller auf `Ready` zurück, so daß wir die nächste Anweisung eingeben können.

Tippen Sie:

```
mode 0 [RETURN]
```

Sie sehen, daß die Schrift auf dem Bildschirm jetzt größer ist. Schreiben Sie wieder zwei Zeilen mit Einsern voll. Jetzt zählen Sie nur noch 20 Ziffern pro Zeile. Das heißt, in Mode 0 haben wir 20 Spalten zur Verfügung. Drücken Sie wieder **[RETURN]**.

Tippen Sie nun:

```
mode 2 [RETURN]
```

Dies ist der kleinste Modus, und wenn Sie eine Reihe mit Einsern vollgetippt haben und zählen, kommen Sie auf 80. In Modus 2 haben wir also 80 Spalten.

Fassen wir zusammen:

Modus 0 = 20 Spalten
Modus 1 = 40 Spalten
Modus 2 = 80 Spalten

Drücken Sie noch einmal **[RETURN]**.

Farben

Sie haben 27 Farben zur Auswahl. Diese sind auf dem grünen Bildschirm (GT65) als unterschiedlich helle Grünstufen dargestellt. Wenn Sie einen GT65 haben, können Sie einen AMSTRAD MP2 Modulator mit Netzteil dazukaufen, um die Farbmöglichkeiten des Computers mit Ihrem eigenen Farbfernseher auszuschöpfen.

In Modus 0 können bis zu 16 der 27 Farben gleichzeitig verwendet werden.
In Modus 1 können bis zu 4 der 27 Farben gleichzeitig verwendet werden.
In Modus 2 können bis zu 2 der 27 Farben gleichzeitig verwendet werden.

Sie können Sie Farben des Bildrandes (`BORDER`), des Schriftuntergrundes odes 'Papiers' (`PAPER`) und der Schrift oder des Schreibstifts (`PEN`) unabhängig voneinander bestimmen.

Die Palette der 27 möglichen Farben ist mit der jeweils zugehörigen `INK`-Nummer in Tabelle 1 aufgeführt.

Damit Sie sie immer gleich zur Hand haben, ist sie außerdem auf dem Computer, rechts vom Tastenfeld, aufgedruckt.

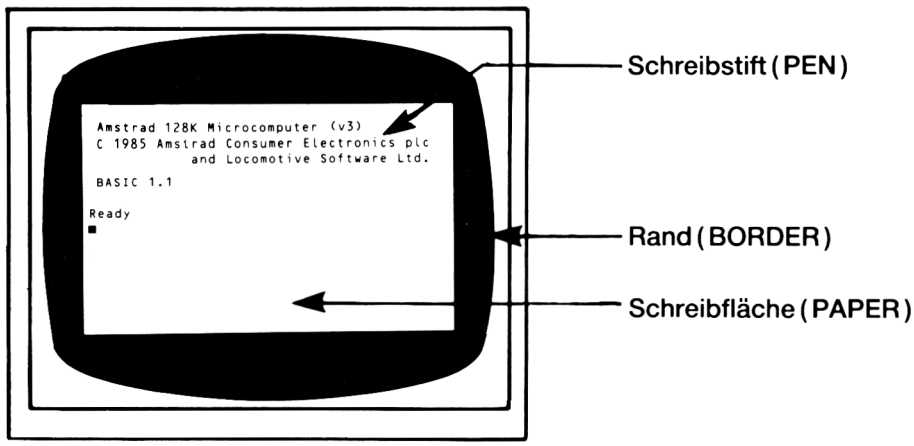
Farbtabelle

INK Nr.	Farbe	INK Nr.	Farbe	INK Nr.	Farbe
0	schwarz	9	grün	18	leuchtend grün
1	blau	10	blaugrün	19	seegrün
2	leuchtend blau	11	himmelblau	20	leuchtend blaugrün
3	rot	12	gelb	21	limonengrün
4	magenta	13	weiß	22	pastellgrün
5	mauve	14	pastellblau	23	pastell-blaugrün
6	leuchtend rot	15	orange	24	leuchtend gelb
7	purpur	16	rosa	25	pastellgelb
8	leuchtend magenta	17	pastellmagenta	26	leuchtend weiß

Tabelle 1: Die INK-Nummern und ihre Farben

Wie schon erwähnt, steht der Computer beim Einschalten auf Modus 1. Um von einem anderen Modus auf Modus 1 zurückzukommen, tippen Sie:

```
mode 1 [RETURN]
```



Der Rand (BORDER) ist die Zone um die eigentliche Schreibfläche (PAPER). Beim Einschalten sind sowohl BORDER als auch PAPER blau. Die Schrift kann nur auf der Schreibfläche, innerhalb des Randes, stehen. PAPER ist der Untergrund, auf dem die Schriftzeichen stehen, PEN (Schreibstift) bezeichnet den Schreibstift, mit dem die Zeichen geschrieben werden.

Wir zeigen jetzt, wie die Farben auf dem Bildschirm ausgewählt werden, und wie Sie sie verändern können.

Beim Einschalten, oder nach dem Zurücksetzen des Computers ist BORDER (der Rand) immer in Farbe 1 dargestellt. Wie Sie aus Ihrer Farbtabelle entnehmen können, ist Farbe 1 blau. Wenn Sie den Rand gern in einer anderen Farbe hätten, geben Sie den Befehl BORDER mit der Nummer der gewünschten Farbe ein. Soll der Rand z.B. weiß erscheinen, tippen Sie:

```
border 13 [RETURN]
```

So weit, so gut. Nun kommen wir zum schwierigeren Teil.

Beim Einschalten, oder nach dem Zurücksetzen des Computers ist PAPER automatisch auf 0, und PEN auf 1 gestellt. Allerdings stimmen 0 und 1 NICHT mit den Nummern auf der Farbtabelle überein!

Sie müssen sich merken, daß 0 und 1 PAPER- und PEN-Nummern sind. Sie sind KEINE INK-Farbnummern. Um sich dies zu verdeutlichen, stellen Sie sich vor, Sie haben vier Füllfederhalter, Nr. 0, 1, 2 und 3. Jeden dieser Füller können Sie mit einer Farbe aus 27 verschiedenen Tintenfässern, die von 0 bis 26 numeriert sind, füllen. Es folgt also, daß Füller Nr.1 nicht unbedingt immer mit derselben Farbe schreibt, da er mit verschiedenen Tinten gefüllt werden kann. Wenn Sie wollten, könnten Sie sogar jeden Füller mit derselben Tinte füllen.

Genauso ist es mit dem Computer. Mit dem Befehl PEN wählen Sie einen der vier Füller, und mit dem Befehl INK die Tintenfarbe für diesen Füller.

Sie erinnern sich, daß wir den Standardmodus (Modus 1 = 40 Spalten) eingestellt haben. Wenn Sie sich die erste und dritte Spalte in der Tabelle 2 unten ansehen, stellen Sie fest, daß der PEN-Nummer 1 die INK-Farbnummer 24 zugeordnet ist. Sehen Sie in ihrer Farbtabelle nach, welche Farbe unter Nummer 24 aufgeführt ist: es ist leuchtend gelb, die Schriftfarbe, die erscheint, wenn Sie den Computer einschalten.

STANDARDFARBEN

PAPER/PEN	INK-Farbe Modus 0	INK-Farbe Modus 1	INK-Farbe Modus 2
0	1	1	1
1	24	24	24
2	20	20	1
3	6	6	24
4	26	1	1
5	0	24	24
6	2	20	1
7	8	6	24
8	10	1	1
9	12	24	24
10	14	20	1
11	16	6	24
12	18	1	1
13	22	24	24
14	blinkend 1,24	20	1
15	blinkend 16,11	6	24

Tabelle 2: PAPER/PEN/MODE/INK Zuordnungstabelle

Die Zuordnung zwischen PAPER, PEN und INK ist jedoch nicht unabänderlich. Es ist lediglich die Standardeinstellung, die Sie vorfinden, wenn Sie den Computer einschalten oder zurücksetzen. Mit dem INK-Befehl können Sie die Farben des Schriftuntergrunds (PAPER) und der Schrift (PEN) verändern. Die Anweisung enthält zwei Teile, sogenannte Parameter. Erstens muß die Nummer des PENs angegeben werden, mit dem geschrieben werden soll, bzw. die Nummer des PAPERs, auf dem geschrieben werden soll, und zweitens muß die Nummer der INK-Farbe angegeben werden. Diese beiden Teile der Anweisung werden durch Komma getrennt.

Wir wissen, daß wir z.Z. mit PEN Nummer 1 schreiben (Standardeinstellung). Um die INK-Farbe des PENs in orange umzuwandeln, müssen wir schreiben:

```
ink 1,15 [RETURN]
```

Sie sehen, daß die Schrift jetzt orange wird.

Entsprechend können Sie mit dem INK-Befehl die Farbe der Schreibfläche (PAPER) ändern. Wir wissen, daß PAPER beim Einschalten die Nummer 0 hat. Um die Schreibfläche grün einzufärben, geben wir als zweiten Parameter im INK-Befehl die Nummer 9 an:

```
ink 0,9 [RETURN]
```

Jetzt wollen wir einen anderen Füller (PEN) verwenden. Schreiben Sie:

```
pen 3 [RETURN]
```

Beachten Sie, daß jetzt nur die Schrift nach dem Befehl ihre Farbe ändert. Wir schreiben also mit PEN Nummer 3, und aus der Tabelle 2 können Sie entnehmen, daß PEN 3 automatisch mit INK-Farbe Nr. 6 (leuchtend rot) schreibt. Wir können die Farbe jetzt in rosa verwandeln, indem wir schreiben:

```
ink 3,16 [RETURN]
```

Die 3 in dieser Anweisung steht für die Nummer des PENs, den wir vorhin gewählt hatten, und 16 ist die Nummer für die Farbe rosa.

Nun wollen wir einen anderen Schriftuntergrund (PAPER) wählen. Wenn PAPER geändert wird, ändert sich der Untergrund des bisher geschriebenen nicht. Dies wird deutlich, wenn Sie eintippen:

```
paper 2 [RETURN]
```

Wieder können Sie anhand der Tabellen 1 und 2 feststellen, weshalb der Schriftuntergrund jetzt leuchtend blaugrün wird. Mit folgendem INK-Befehl verwandeln Sie ihn in schwarz:

```
ink 2,0 [RETURN]
```

Auf dem Bildschirm haben wir jetzt Schrift von PEN 1 und PEN 3 auf PAPER 0 und PAPER 2. Sie können auch die Farbe von einem PEN oder PAPER verändern, die Sie im Moment nicht benutzen. Mit dem Befehl:

```
ink 1,2 [RETURN]
```

...z.B. ändern Sie die Farbe der Schrift, die Sie vorhin mit PEN 1 geschrieben haben.

Löschen Sie nun die Schreibfläche mit:

```
cls [RETURN]
```

Nun müßten Sie eigentlich in der Lage sein, die ursprünglichen Farben auf dem Bildschirm (blauer Rand, blaue Schreibfläche, gelbe Schrift) mit den BORDER-, PAPER-, PEN- und INK-Befehlen wiederherzustellen. Versuchen Sie es. Gelingt es Ihnen nicht, setzen Sie den Computer mit **[CONTROL] [SHIFT] [ESC]** zurück.

Blinkende Farben

Sie können die Schrift in zwei Farben abwechselnd blinken lassen. Dazu müssen Sie im INK-Befehl, mit dem Sie die PEN-Farbe festlegen, eine zweite Farbnummer angeben. Wenn Sie die Schrift abwechselnd in leuchtend weiß und leuchtend rot blinken lassen möchten, schreiben Sie (nachdem Sie den Computer zurückgesetzt haben):

```
ink 1,26,6 [RETURN]
```

1 ist die PEN-Nummer, 26 steht für die Farbe leuchtend weiß und 6 ist leuchtend rot.

Genauso können Sie den Schriftuntergrund in zwei Farben blinken lassen. Dazu geben Sie im INK-Befehl, mit dem Sie die PAPER-Farbe festlegen, eine zweite Farbnummer an. Wenn der Schriftuntergrund abwechselnd grün und leuchtend gelb blinken soll, schreiben Sie:

```
ink 0,9,24 [RETURN]
```

Ø ist die PAPER-Nummer, 9 ist die Farbe grün und 24 ist leuchtend gelb.

Setzen Sie den Computer nun mit **[CONTROL] [SHIFT] [ESC]** zurück.

In Tabelle 2 sehen Sie, daß im Modus 0 für PEN 14 und 15 bzw. PAPER 14 und 15 schon zwei blinkende Farben vorgesehen sind. Sie sind also mit einem zusätzlichen INK-Parameter vorprogrammiert worden.

Tippen Sie folgendes ein:

```
mode Ø [RETURN]
pen 15 [RETURN]
```

Jetzt sehen Sie das Wort Ready auf Ihrem Bildschirm abwechselnd in himmelblau und rosa blinken.

Tippen Sie:

```
paper 14 [RETURN]
cls [RETURN]
```

Jetzt blinkt auch der Schriftuntergrund, und zwar in gelb und blau.

Die Farben 14 und 15 für PEN und PAPER können Sie mit dem INK-Befehl in eine andere Farbe oder zwei andere blinkende Farben umwandeln.

Auch den Rand können Sie in zwei Farben blinken lassen. Dazu geben Sie im BORDER-Befehl zwei Farbnummern an. Schreiben Sie:

```
border 6,9 [RETURN]
```

Jetzt sehen Sie den Rand in leuchtend rot und grün blinken. Beachten Sie, daß für die Wahl der Randfarbe(n) die gesamte Farbpalette (27 Farben) zur Verfügung steht, gleichgültig, ob Sie Modus 0, 1 oder 2 eingestellt haben.

Setzen Sie den Computer mit **[CONTROL], [SHIFT]** und **[ESC]** zurück.

Zur Demonstration der farblichen Möglichkeiten Ihres Computers tippen Sie folgendes Programm ein und lassen es laufen:

```
10 MODE 0 [RETURN]
20 rate=600: REM bestimmt tempo des programms [RETURN]
30 FOR b=0 TO 26 [RETURN]
40 LOCATE 3,12 [RETURN]
50 BORDER b [RETURN]
60 PRINT "Randfarbe";b [RETURN]
70 FOR t=1 TO rate [RETURN]
80 NEXT t,b [RETURN]
90 CLG [RETURN]
100 FOR p=0 TO 15 [RETURN]
110 PAPER p [RETURN]
120 PRINT "paper";p [RETURN]
130 FOR n=0 TO 15 [RETURN]
140 PEN n [RETURN]
150 PRINT "pen";n [RETURN]
160 NEXT n [RETURN]
170 FOR t=1 TO rate*2 [RETURN]
180 NEXT t,p [RETURN]
190 MODE 1 [RETURN]
200 BORDER 1 [RETURN]
210 PAPER 0 [RETURN]
220 PEN 1 [RETURN]
230 INK 0,1 [RETURN]
240 INK 1,24 [RETURN]
run [RETURN]
```

HINWEIS

In obigem Programm, wie auch in den Beispielprogrammen der weiteren Kapitel, sind die BASIC-Schlüsselwörter in GROSSBUCHSTABEN dargestellt. So erscheinen die Schlüsselwörter auch auf dem Bildschirm, wenn der Computer ein Programm auf`LIST`et. Im allgemeinen ist es besser, wenn Sie die Befehle mit kleinen Buchstaben eintippen, da Sie dann beim Auf`LIST`en eventuelle Tippfehler leichter finden, weil der Computer ein falsch geschriebenes BASIC-Schlüsselwort nicht in Großschreibweise wiedergibt.

In diesem Kapitel werden von jetzt ab die Schlüsselwörter in den Programmen sowohl groß- als auch kleingeschrieben, so daß Sie sich an die neue Schreibweise gewöhnen können.

Variablennamen wie `x` oder `a$` werden beim Auflisten des Programms nicht in Großbuchstaben dargestellt, obwohl der Computer sie in jedem Fall erkennt, ob sie im Programm nun groß- oder kleingeschrieben wurden.

Achtung!

Von jetzt ab werden wir nicht mehr nach jeder Programmzeile **[RETURN]** schreiben. Wir gehen davon aus, daß Sie die Taste mittlerweile automatisch drücken.

Graphik

Der Computer hat eine ganze Reihe von Symbolen gespeichert. Mit dem Schlüsselwort:

```
chr$( )
```

...können Sie diese Symbole ausgeben lassen. In Klammern sollte die Nummer des Symbols, eine Zahl zwischen 32 und 255, stehen.

Setzen Sie den Computer mit **[CONTROL]**, **[SHIFT]** und **[ESC]** zurück und tippen Sie:

```
print chr$(250)
```

Vergessen Sie nicht, **[RETURN]** zu drücken. Auf dem Bildschirm erscheint Symbol 250, ein Männchen, das nach rechts läuft.

Um alle Zeichen und Symbole mit ihren Nummern aufführen zu lassen, schreiben Sie folgendes Programm, wobei Sie daran denken, nach jeder Zeile **[RETURN]** zu drücken:

```
10 for n=32 to 255
20 print n;chr$(n)
30 next n
```

Im Kapitel 7 ('Übersicht') sind alle Zeichen und Symbole mit ihren Nummern aufgeführt.

LOCATE

Mit diesem Befehl kann der Cursor an eine gewünschte Stelle auf dem Bildschirm gesetzt werden. Ohne LOCATE-Befehl beginnt der Cursor in der linken, oberen Ecke des Bildschirms. Dies ist der Koordinatenpunkt 1,1 (dies bedeutet, erste Position auf der horizontalen x-Achse und erste Position auf der vertikalen y-Achse). In Modus 1 haben wir 40 Spalten und 25 Zeilen. Um den Cursor in Modus 1 in die Mitte der ersten Zeile zu rücken, geben wir als Koordinatenpunkt 20,1 an.

Um sich das zu verdeutlichen, schreiben Sie (mit **[RETURN]** nach jeder Zeile):

```
mode 1
```

Der Bildschirm wird freigemacht, und der Cursor rückt in die linke, obere Ecke.

```
10 locate 20,1
20 print chr$(250)
run
```

Um zu zeigen, daß die Figur tatsächlich in der obersten Zeile steht, schreiben Sie:

```
border 0
```

Der Rand ist jetzt schwarz, und Sie sehen das Männchen in der Mitte der obersten Zeile der Schreibfläche.

Im Modus 0 haben wir nur 20 Spalten, aber wieder 25 Zeilen zur Verfügung. Schreiben Sie:

```
mode 0
run
```

Jetzt erscheint das Männchen oben rechts auf der Bildfläche, weil in diesem Modus Punkt 20 der x-Koordinate auf der letzten Spalte liegt.

Im Modus 2 haben wir 80 Spalten und 25 Zeilen. Wenn Sie das Programm unverändert lassen, können Sie wahrscheinlich erraten, wo das Männchen dieses Mal auftaucht. Schreiben Sie:

```
mode 2
run
```

Um zu Modus 1 zurückzukehren, schreiben Sie:

```
mode 1
```

Probieren Sie jetzt selbst aus, mit den Befehlen `locate` und `chr$( )` verschiedene Symbole auf unterschiedliche Bildschirmpositionen zu setzen. Hier ein weiteres Beispiel:

```
locate 20,12:print chr$(240)
```

Sie sehen jetzt einen Pfeil in der Bildschirmmitte. In dieser Anweisung steht die Zahl:

20 für den Punkt auf der horizontalen x-Koordinatenachse (Bereich 1 bis 40)
12 für den Punkt auf der vertikalen y-Koordinatenachse (Bereich 1 bis 25)
240 für die Nummer des Symbols

Durch folgendes Programm wird das Symbol wiederholt über den ganzen Bildschirm geschrieben:

```
10 CLS
20 FOR x=1 TO 39
30 LOCATE x,20
50 PRINT CHR$(250)
60 NEXT x
70 GOTO 10
run
```

Drücken Sie zweimal die **[ESC]**-Taste, um das Programm abzubrechen.

Wenn das Symbol auf dem Bildschirm gelöscht werden soll, bevor ein neues erscheint, schreiben Sie:

```
50 print " ";chr$(250)
```

Diese Zeile ersetzt automatisch die alte Zeile 50 im Programm.

Nun tippen Sie:

```
run
```

FRAME

Fügen Sie nun folgende Anweisung hinzu, um die Bewegung der Symbole auf dem Bildschirm gleichmäßiger zu gestalten:

```
40 frame
```

Der FRAME-Befehl synchronisiert die Bewegung von Zeichen auf dem Bildschirm mit dem Strahlrücklauf. Wenn Ihnen diese Erklärung zu technisch ist, merken Sie sich einfach, daß der Befehl dann eingesetzt wird, wenn sich Symbole oder Graphiken gleitend über den Bildschirm bewegen sollen.

Dieses Programm läßt sich noch weiter verbessern, indem man noch einige Verzögerungsschleifen einbaut und ein anderes Symbol in die Gegenrichtung laufen läßt.

Schreiben Sie:

```
list
```

...und ergänzen Sie das Programm mit folgenden Zeilen:

```
10 CLS
20 FOR x=1 TO 39
30 LOCATE x,20
40 FRAME
50 PRINT " ";CHR$(250)
60 NEXT x
70 FOR n=1 TO 300:NEXT n
80 FOR x=39 TO 1 STEP -1
90 LOCATE x,20
100 FRAME
110 PRINT CHR$(251);" "
120 NEXT x
130 FOR n=1 TO 300:NEXT n
140 GOTO 20
run
```

PLOT

Im Gegensatz zum LOCATE-Befehl benutzen wir den PLOT-Befehl, um die Position des Graphik-Cursors zu bestimmen, dieses Mal mit Pixel-Koordinaten. (Ein Pixel ist ein Bildpunkt, oder die kleinste darstellbare Einheit auf dem Bildschirm.)

Beachten Sie, daß der Graphik-Cursor unsichtbar, und nicht identisch mit dem Text-Cursor ist.

In horizontaler Richtung hat der Bildschirm 640 Pixel, in vertikaler Richtung 400 Pixel. Der Ursprung des Koordinatenkreuzes mit dem Punkt 0,0 liegt in der Ecke unten links auf dem Bildschirm. Die Pixel-Koordinaten bleiben im Modus 0, 1 und 2 immer gleich.

Um dies zu testen, setzen Sie den Computer mit **[CONTROL] [SHIFT] [ESC]** zurück und schreiben:

```
plot 320,200
```

In der Mitte des Bildschirms können Sie nun einen winzigen Punkt erkennen.

Ändern Sie jetzt den Modus mit:

```
mode 0
plot 320,200
```

Sie sehen, daß der Punkt sich noch an derselben Stelle befindet, jedoch etwas größer geworden ist. Ändern Sie wieder den Modus und geben dieselben Koordinaten an:

```
mode 2
plot 320,200
```

Der Punkt ist immer noch in der Mitte, aber viel kleiner.

Setzen Sie mehrere Punkte in unterschiedlichen Modi an verschiedene Stellen des Bildschirms, um sich mit dem PLOT-Befehl vertraut zu machen. Wenn Sie damit fertig sind, kehren Sie zum Modus 1 zurück und löschen gleichzeitig den Bildschirm mit:

```
mode 1
```

DRAW

Setzen Sie den Computer zunächst mit **[CONTROL] [SHIFT] [ESC]** zurück. Mit dem Befehl DRAW ziehen Sie eine Linie, ausgehend von der gegenwärtigen Position des Graphik-Cursors. Um dies zu verdeutlichen, zeichnen Sie mit folgendem Programm ein Rechteck auf den Bildschirm.

Dabei bringen wir den Graphik-Cursor mit dem PLOT-Befehl an die Ausgangsposition, ziehen dann mit DRAW eine Linie zur linken oberen Ecke, von dort zur rechten oberen Ecke usw. Schreiben Sie:

```
5 cls
10 PLOT 10,10
20 DRAW 10,390
30 DRAW 630,390
40 DRAW 630,10
50 DRAW 10,10
60 GOTO 60
run
```

Drücken Sie zweimal **[ESC]**, um das Programm abzubrechen.

(Sehen Sie sich Zeile 60 einmal genauer an. Der Computer hat die Anweisung, solange eine Schleife um Zeile 60 zu drehen, bis Sie aus dem Programm 'ausbrechen', indem Sie zweimal die **[ESC]**-Taste drücken. Eine derartige Anweisung empfiehlt sich, wenn Sie verhindern wollen, daß der Computer gleich nach Ausführung des Programms abbricht, und sich mit `Ready` wiedermeldet.)

Fügen Sie jetzt die folgenden Programmzeilen hinzu, mit denen ein zweites Rechteck innerhalb des ersten gezeichnet wird:

```
10 PLOT 10,10
20 DRAW 10,390
30 DRAW 630,390
40 DRAW 630,10
50 DRAW 10,10
60 PLOT 20,20
70 DRAW 20,380
80 DRAW 620,380
90 DRAW 620,20
100 DRAW 20,20
110 GOTO 110
run
```

Drücken Sie wieder zweimal die **[ESC]**-Taste, um das Programm abzubrechen.

MOVE

Wie mit `PLOT` wird der Graphik-Cursor mit dem Befehl `MOVE` an einen gewünschten Koordinatenpunkt gesetzt. Allerdings wird an der neuen Cursorposition kein Pixel (Bildpunkt) abgebildet.

Schreiben Sie:

```
cls
move 639,399
```

Der Graphik-Cursor befindet sich nun in der oberen rechten Ecke, obwohl wir nirgends eine Spur von ihm entdecken können.

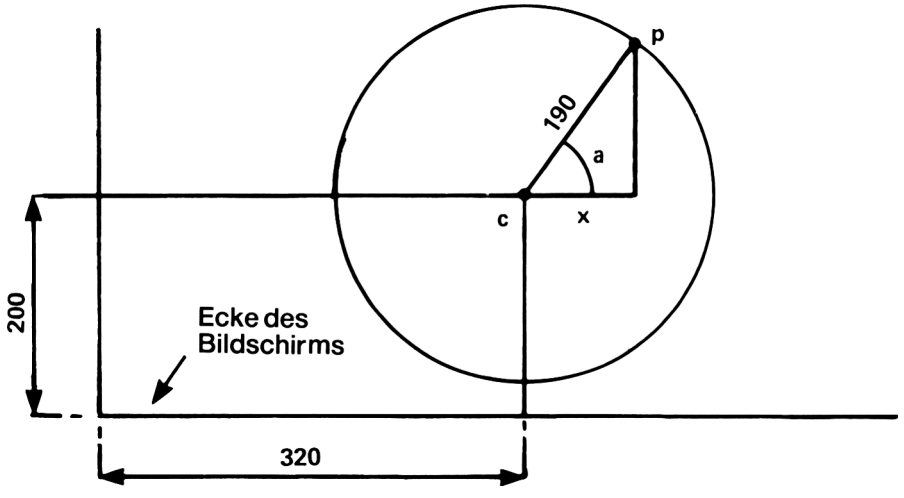
Wir können dies überprüfen, indem wir von seiner Position eine Linie zur Bildschirmmitte ziehen. Schreiben Sie:

```
draw 320,200
```

Kreise

Kreise können entweder geplottet oder gezeichnet werden. Eine Möglichkeit, einen Kreis zu bilden, besteht darin, die Koordinatenpunkte jedes Punktes auf der Kreislinie zu plotten. Wenn Sie sich die Abbildung unten ansehen, erkennen Sie, daß der Punkt p auf der Kreislinie mit folgenden x- und y-Koordinaten geplottet werden kann:

```
x=190*cos(a)
y=190*sin(a)
```



Das Plotten von Kreispunkten

In den vorigen Programmen haben wir die Koordinatenpunkte auf die linke untere Ecke des Bildschirms bezogen. Wenn wir einen Kreis genau in die Mitte des Bildschirms plazieren wollten, mußten wir den Kreismittelpunkt auf den Koordinatenpunkt 320,200 plotten und diese Koordinaten dann zu allen Kreispunkten hinzuaddieren.

Ein Kreis nach dieser Methode wird mit folgendem Programm gezeichnet:

```
new
10 CLS
20 DEG
30 FOR a=1 TO 360
40 MOVE 320,200
50 PLOT 320+190*COS(a),200+190*SIN(a)
60 NEXT
run
```

Beachten Sie das Schlüsselwort NEW vor dem eigentlichen Programm. NEW sagt dem Computer, daß der Speicher gelöscht werden soll. Es hat also eine ähnliche Funktion wie **[CONTROL] [SHIFT] [ESC]**, mit dem Unterschied, daß der Bildschirm dabei nicht gelöscht wird.

Der Kreisradius kann verkleinert werden, indem man die Anzahl der Kreispunkte (in unserem Beispiel 190) reduziert.

Der Kreis wird anders geplottet (im Bogenmaß), wenn man die Zeile 20 entfernt. Löschen Sie Zeile 20, indem Sie einfach:

```
20
```

eintippen.

Einen ausgefüllten Kreis bekommen Sie, wenn Sie das Schlüsselwort PLOT in Zeile 50 durch DRAW ersetzen:

```
50 draw 320+190*cos(a),200+190*sin(a)
```

Probieren Sie diese Version mit und ohne Zeile 20 aus.

Sie haben vielleicht bemerkt, daß wir in Zeile 60 statt NEXT a einfach NEXT geschrieben haben. Dies ist zulässig, da der Computer automatisch die richtige FOR-Anweisung zum angegebenen NEXT ermittelt. Für Programme, in denen viele FOR...NEXT-Schleifen vorkommen, ist es jedoch der Übersichtlichkeit halber zu empfehlen, den Namen der Variablen in der NEXT-Anweisung anzugeben.

ORIGIN

Im vorigen Programm haben wir mit MOVE den Kreismittelpunkt angegeben und dann die Mittelpunktkoordinaten zu den Kreislinienkoordinaten addiert. Mit dem Befehl ORIGIN können wir uns diese Prozedur sparen und den Koordinatenursprung einfach in die Kreismitte verlegen. Dann sieht das Programm so aus:

```
new
10 CLS
20 FOR a=1 TO 360
30 ORIGIN 320,200
40 PLOT 190*COS(a),190*SIN(a)
50 NEXT
run
```

Plotten Sie nun vier kleinere Kreise mit folgendem Programm:

```
new
10 CLS
20 FOR a=1 TO 360
30 ORIGIN 196,282
40 PLOT 50*COS(a),50*SIN(a)
50 ORIGIN 442,282
60 PLOT 50*COS(a),50*SIN(a)
70 ORIGIN 196,116
80 PLOT 50*COS(a),50*SIN(a)
90 ORIGIN 442,116
100 PLOT 50*COS(a),50*SIN(a)
110 NEXT
run
```

Das folgende Programm zeigt Ihnen, wie ein Kreis auf andere Art erstellt wird:

```
new
10 MODE 1
20 ORIGIN 320,200
30 DEG
40 MOVE 0,190
50 FOR a=0 TO 360 STEP 10
60 DRAW 190*SIN(a),190*COS(a)
70 NEXT
run
```

Dieses Mal wird die Kreislinie durch den Befehl `DRAW` von Koordinatenpunkt zu Koordinatenpunkt gezogen. Das geht erheblich schneller als mit `PLOT`.
Sehen Sie sich auch hier an, was geschieht, wenn der Befehl `DEG` in Zeile `30` entfernt wird.

FILL

Der Befehl `FILL` bewirkt, daß ein Bildschirmbereich, der durch Linien oder den Bildschirmrand bzw. den Rand eines Bildschirmteilbereichs (Window) begrenzt ist, ausgefüllt wird.

Setzen Sie den Computer mit **[CONTROL] [SHIFT] [ESC]** zurück und tippen Sie folgendes Programm ein:

```
new
10 CLS
20 MOVE 20,20
30 DRAW 620,20
40 DRAW 310,380
50 DRAW 20,20
run
```

Auf dem Bildschirm erscheint nun ein Dreieck. Setzen Sie den Graphik-Cursor in die Mitte mit der Anweisung:

```
move 320,200
```

Wenn wir jetzt das Schlüsselwort **FILL** mit einer **PEN**-Nummer, z.B. **3**, eingeben, wird dieser **PEN**, von der Position des Graphik-Cursors ausgehend (in diesem Fall von der Mitte), den Bereich bis zu den gezogenen Linien ausfüllen. Schreiben Sie:

```
fill 3
```

Setzen Sie nun den Graphik-Cursor außerhalb des Dreiecks:

```
move 0,0
```

...und sehen Sie, was passiert, wenn Sie jetzt

```
fill 2
```

eintippen.

Der Computer hat mit **PEN** Nummer **2** die gesamte Fläche zwischen den gezogenen Linien und dem Bildschirmrand ausgefüllt.

Ändern Sie nun das Programm, indem Sie die folgenden Zeilen eintippen, und sehen Sie sich das Resultat an:

```
new
10 CLS
20 MOVE 20,20
30 DRAW 620,20
40 DRAW 310,380
50 DRAW 50,50
60 MOVE 320,200
70 FILL 3
run
```

Sie sehen, daß Lücken in den Grenzlinien die Farbe durchsickern läßt!

Dies zeigt sich auch deutlich, wenn Sie einen geplotteten Kreis ausfüllen wollen. Schreiben Sie:

```
new
10 CLS
20 FOR a=1 TO 360
30 ORIGIN 320,200
40 PLOT 190*COS(a),190*SIN(a)
50 NEXT
60 MOVE -188,0
70 FILL 3
run
```

Vergleichen Sie, was passiert, wenn Sie einen mit dem Befehl **DRAW** gezeichneten Kreis ausfüllen:

```
new
10 MODE 1
20 ORIGIN 320,200
30 DEG
40 MOVE 0,190
50 FOR d=0 TO 360 STEP 10
60 DRAW 190*SIN(d),190*COS(d)
70 NEXT
80 MOVE -188,0
90 FILL 3
run
```

Wir können die Kreislinie auch unsichtbar machen, indem wir dem **PEN** die Farbe (**INK**) des **PAPER**s zuordnen. Ergänzen Sie das Programm mit:

```
45 GRAPHICS PEN 2:INK 2,1
run
```

Mit dem Schlüsselwort **GRAPHICS PEN** wählen Sie den **PEN**, mit dem Sie etwas auf den Graphik-Bildschirm zeichnen wollen. Mit dem Befehl **INK** teilen Sie dem **PEN** die gewünschte Farbe zu (in diesem Fall Farbe Nummer **1**, die **PAPER**-Farbe).

Schreiben Sie zum Schluß noch folgendes Demonstrationsprogramm:

```
new
10 MODE 0:BORDER 13
20 MOVE 0,200:DRAW 640,200
30 FOR x=80 TO 560 STEP 80
40 MOVE x,0:DRAW x,400
50 NEXT:MOVE -40,300
60 FOR c=0 TO 7
70 MOVER 80,0:FILL c
80 MOVER 0,-200:FILL c+8
90 MOVER 0,200:NEXT
100 GOTO 100
run
```

Mit folgender Ergänzung können Sie die Farben der ausgefüllten Bereiche variieren. Schreiben Sie:

```
new
10 MODE 0:BORDER 13
20 MOVE 0,200:DRAW 640,200
30 FOR x=80 TO 560 STEP 80
40 MOVE x,0:DRAW x,400
50 NEXT:MOVE -40,300
60 FOR c=0 TO 7
70 MOVER 80,0:FILL c
80 MOVER 0,-200:FILL c+8
90 MOVER 0,200:NEXT
100 SPEED INK 30,30
110 BORDER RND*26,RND*26
120 INK RND*15,RND*26,RND*26
130 FOR t=1 TO 500:NEXT:GOTO 110
run
```

Eine detailliertere Beschreibung der graphischen Möglichkeiten des CPC6128 finden Sie im Abschnitt über Graphik im Kapitel 'Wenn Sie gerade Zeit haben'.

Zum Abschluß dieses Abschnitts haben wir einige Demonstrationsprogramme zur Graphik aufgeführt, die viele Befehle und Schlüsselwörter enthalten, die Sie gerade kennengelernt haben. Jedes Programm läuft endlos.

```
new
10 BORDER 0:GRAPHICS PEN 1
20 m=CINT(RND*2):MODE m
```

```
30 i1=RND*26:i2=RND*26
40 IF ABS(i1-i2)<10 THEN 30
50 INK 0,i1:INK 1,i2
60 s=RND*5+3:ORIGIN 320,-100
70 FOR x= -1000 TO 0 STEP s
80 MOVE 0,0:DRAW x,300:DRAW 0,600
90 MOVE 0,0:DRAW -x,300:DRAW 0,600
100 NEXT:FOR t=1 TO 2000:NEXT:GOTO 20
run

10 MODE 1:BORDER 0:PAPER 0
20 GRAPHICS PEN 2:INK 0,0:i=14
30 EVERY 2200 GOSUB 150
40 flag=0:CLG
50 INK 2,14+RND*12
60 b%=RND*5+1
70 c%=RND*5+1
80 ORIGIN 320,200
90 FOR a=0 TO 1000 STEP PI/30
100 x%=100*COS(a)
110 MOVE x%,y%
120 DRAW 200*COS(a/b%),200*SIN(a/c%)
130 IF flag=1 THEN 40
140 NEXT
150 flag=1:RETURN
run

10 MODE 1:BORDER 0:DEG
20 PRINT "Bitte warten"
30 FOR n=1 TO 3
40 INK 0,0:INK 1,26:INK 2,6:INK 3,18
50 IF n=1 THEN sa=120
60 IF n=2 THEN sa=135
70 IF n=3 THEN sa=150
80 IF n=1 THEN ORIGIN 0,-50,0,640,0,400 ELSE ORIGIN 0,0
   ,0,640,0,400
90 DIM cx(5),cy(5),r(5),lc(5)
100 DIM np(5)
110 DIM px%(5,81),py%(5,81)
120 st=1:cx(1)=320:cy(1)=200:r(1)=80
130 FOR st=1 TO 4
140 r(st+1)=r(st)/2
150 NEXT st
160 FOR st=1 TO 5
170 lc(st)=0:np(st)=0
```

```
180 np(st)=np(st)+1
190 px%(st,np(st))=r(st)*SIN(lc(st))
200 py%(st,np(st))=r(st)*COS(lc(st))
210 lc(st)=lc(st)+360/r(st)
220 IF lc(st) < 360 THEN 180
230 px%(st,np(st)+1)=px%(st,1)
240 py%(st,np(st)+1)=py%(st,1)
250 NEXT st
260 CLS:cj=REMAIN(1):cj=REMAIN(2)
270 cj=REMAIN(3):INK 1,2:st=1
280 GOSUB 350
290 LOCATE 1,1
300 EVERY 25,1 GOSUB 510
310 EVERY 15,2 GOSUB 550
320 EVERY 5,3 GOSUB 590
330 ERASE cx,cy,r,lc,np,px%,py%:NEXT
340 GOTO 340
350 cx%=cx(st):cy%=cy(st):lc(st)=0
360 FOR x%=1 TO np(st)
370 MOVE cx%,cy%
380 DRAW cx%+px%(st,x%),cy%+py%(st,x%),1+(st MOD 3)
390 DRAW cx%+px%(st,x%+1),cy%+py%(st,x%+1),1+(st MOD 3)
400 NEXT x%
410 IF st=5 THEN RETURN
420 lc(st)=0
430 cx(st+1)=cx(st)+1.5*r(st)*SIN(sa+lc(st))
440 cy(st+1)=cy(st)+1.5*r(st)*COS(sa+lc(st))
450 st=st+1
460 GOSUB 350
470 st=st-1
480 lc(st)=lc(st)+2*sa
490 IF (lc(st) MOD 360)<>0 THEN 430
500 RETURN
510 ik(1)=1+RND*25
520 IF ik(1)=ik(2) OR ik(1)=ik(3) THEN 510
530 INK 1,ik(1)
540 RETURN
550 ik(2)=1+RND*25
560 IF ik(2)=ik(1) OR ik(2)=ik(3) THEN 550
570 INK 2,ik(2)
580 RETURN
590 ik(3)=1+RND*25
600 IF ik(3)=ik(1) OR ik(3)=ik(2) THEN 590
610 INK 3,ik(3)
620 RETURN
```

Teil 9: Tonerzeugung

Klänge gibt der Computer über einen eingebauten Lautsprecher aus. Wenn Sie mit einem MP2 Modulator/Netzteil und Ihrem Fernseher arbeiten, sollten Sie den Ton des Fernsehgerätes abstellen.

Die Lautstärke können Sie mit dem **VOLUME**-Regler rechts am Computer einstellen. Sie können den Ton auch über die **STEREO**-Buchse des Computers auf Ihre Stereoanlage übertragen, so daß Sie die Musik des Computers in Stereo - über die Lautsprecher oder Kopfhörer - genießen können. Im 2.Teil dieses Kapitels erfahren Sie, wie die Stereoanlage an den Computer anzuschließen ist.

Der SOUND-Befehl

Der Befehl `SOUND` besteht aus sieben Teilen oder Parametern. Die ersten beiden Parameter müssen angegeben werden, die anderen können nach Wunsch eingesetzt werden. Der Befehl hat folgende Form:

`SOUND` ‹Kanalstatus›`,`‹Tonperiode›`,`‹Dauer›`,`‹Lautstärke›`,`
‹Lautstärken-Hüllkurve›`,`‹Ton-Hüllkurve›`,`‹Rauschen›

Das sieht ganz schön kompliziert aus, aber wenn wir jeden Parameter analysieren, bekommen wir die Sache bald in den Griff. Sehen wir uns also die einzelnen Parameter der Reihe nach an:

Kanalstatus

Der Einfachheit halber betrachten wir den ‹Kanalstatus› vorläufig als Größe, die angibt, welcher Tonkanal benutzt wird. Es gibt drei Tonkanäle, und wir benutzen jetzt ‹Kanalstatus› Nr. `1`.

Tonperiode

‹Tonperiode› ist der technische Ausdruck für die Höhe eines Tones, für die Note, die gespielt werden soll. Jede Note hat eine Nummer, die für den Parameter ‹Tonperiode› eingesetzt wird. Wenn Sie im Kapitel 'Übersicht' nachsehen, stellen Sie fest, daß das eingestrichene C die Tonperiode `239` hat.

Setzen Sie den Computer mit **[CONTROL] [SHIFT] [ESC]** zurück und schreiben Sie:

```
10 sound 1,239
run
```

Für 0,2 Sekunden hören Sie das eingestrichene C.

Wenn Sie nichts gehört haben, prüfen Sie, ob der Lautstärkeregler am Computer vielleicht auf 0 stand. Drehen Sie auf und tippen Sie noch einmal `RUN` ein.

Dauer

Dieser Parameter setzt die Länge eines Tones fest. Die Einheit für die Dauer ist 0,01 Sekunden (eine Hundertstelsekunde). Wenn Sie die Dauer nicht angeben, wird automatisch der Wert `20` angenommen. Deshalb dauerte der eben gehörte Ton 0,2 Sekunden (= 0,01 x 20).

Für eine Tondauer von einer Sekunde muß der Wert 100 angegeben werden, und soll der Ton zwei Sekunden dauern, muß 200 angegeben werden. Tippen Sie:

```
10 sound 1,239,200
run
```

Jetzt hören Sie das eingestrichene C zwei Sekunden lang.

Lautstärke

Dieser Parameter bestimmt die Lautstärke am Anfang eines Tones. Es können Werte zwischen `0` und `15` angegeben werden (Bei `0` ist nichts zu hören, `15` ist die maximale Lautstärke.) Wird für ‹Lautstärke› nichts angegeben, nimmt der Computer automatisch `12` an. Schreiben Sie:

```
10 sound 1,239,200,5
run
```

Merken Sie sich die Lautstärke dieses Tones, und geben Sie dann denselben Befehl mit einem größeren Lautstärkewert ein:

```
10 sound 1,239,200,15
run
```

Sicher hören Sie den Unterschied.

Lautstärken-Hüllkurve

Die Lautstärken-Hüllkurve kann mit dem Befehl ENV bestimmt werden. Mit ihr läßt sich die Lautstärke eines Tones während seiner Dauer variieren. Sie können mehrere verschiedene Lautstärken-Hüllkurven definieren, die jeweils mit einer Nummer versehen werden. Wenn Sie mit ENV eine Lautstärken-Hüllkurve mit der Nummer 1 bestimmt haben, die Sie im SOUND-Befehl einsetzen wollen, geben Sie für den Parameter ‹Lautstärken-Hüllkurve› 1 an. In Kürze werden wir erläutern, wie eine Lautstärken-Hüllkurve definiert wird.

Ton-Hüllkurve

Die Ton-Hüllkurve kann mit dem Befehl ENT bestimmt werden. Mit ihr läßt sich die Höhe eines Tones während seiner Dauer variieren, so daß ein Vibrato-ähnlicher Effekt entsteht. Sie können mehrere verschiedene Ton-Hüllkurven definieren, die jeweils mit einer Nummer versehen werden. Wenn Sie mit ENT eine Ton-Hüllkurve mit der Nummer 1 bestimmt haben, die Sie im SOUND-Befehl einsetzen wollen, geben Sie für den Parameter ‹Tonhüllkurve› 1 an. In Kürze werden wir erläutern, wie eine Ton-Hüllkurve definiert wird.

Rauschen

‹Rauschen› ist der letzte Parameter des SOUND-Befehls. Der Computer bietet 31 Arten von Rauschen zur Begleitung des Tones an, die durch Angabe eines Wertes zwischen 1 und 31 gewählt werden können. Fügen Sie am Ende des SOUND-Befehls den Wert 2 an und hören Sie den Effekt. Setzen Sie dann für ‹Rauschen› den Wert 27 ein und achten Sie auf den Unterschied. Schreiben Sie:

```
10 sound 1,239,200,15,,,2
```

Beachten Sie die beiden 'Leer'-Parameter (,,,) vor dem ‹Rauschen›-Parameter 2. Diese mußten wir einfügen, weil wir keine ‹Lautstärken-Hüllkurve› und keine ‹Tonhüllkurve› definiert haben.

Definition einer Lautstärken-Hüllkurve

Das Schlüsselwort zur Definition einer Lautstärken-Hüllkurve (engl. Volume ENVelope) heißt ENV. In seiner einfachsten Form besteht der Befehl aus vier Parametern und sieht folgendermaßen aus:

ENV ‹Hüllkurvennummer›, ‹Schrittanzahl›,‹Schrittweite›,‹Dauer pro Schritt›

Wieder wollen wir uns die Parameter der Reihe nach genauer ansehen:

Hüllkurvennummer

Dies ist die Nummer (zwischen 1 und 15), die einer bestimmten Lautstärken-Hüllkurve gegeben wird, damit sie im SOUND-Befehl aufgerufen werden kann.

Schrittanzahl

Dieser Parameter bestimmt, wieviele Lautstärkeveränderungen ein Ton während seiner Dauer durchlaufen soll. Wenn Sie z.B. in einen 10 Sekunden dauernden Ton 10 verschiedene Lautstärkeschritte von jeweils einer Sekunde Dauer einbauen wollen, müssen Sie für den Parameter ‹Schrittanzahl› den Wert 10 angeben.

Für die ‹Schrittanzahl› können die Werte 0 bis 127 angegeben werden.

Schrittgrösse

Die Differenz zwischen den einzelnen Lautstärkeschritten kann zwischen 0 und 15 betragen, da im SOUND-Befehl 16 verschiedene Lautstärkegrade möglich sind. Als ‹Schrittgröße›-Parameter können jedoch Werte zwischen -128 und 127 angegeben werden. Die Lautstärke geht dabei jedesmal auf 0 zurück, wenn das Maximum erreicht ist.

Schrittzeit

Dieser Parameter bestimmt die Zeit zwischen den einzelnen Lautstärkeschritten in Einheiten zu Hundertstelsekunden (0,01 Sekunden). Es können Werte von 0 bis 255 eingesetzt werden, d.h. ein Lautstärkeschritt kann höchstens 2,56 Sekunden lang sein (0 wird als 256 behandelt).

Beachten Sie, daß die Dauer aller Lautstärkeschritte zusammengenommen nicht länger sein sollte als die Dauer des Tones, die Sie im SOUND-Befehl angeben. Sonst hört der Ton auf, bevor alle Lautstärkeschritte durchlaufen sind. (In solch einem Fall verfällt der Rest der Lautstärken-Hüllkurve.)

Ist dagegen der ‹Schrittzeit›-Parameter im SOUND-Befehl größer als die Dauer aller Lautstärkeschritte zusammen, wird der Ton für den Rest seiner Dauer auf der letzten Lautstärkestufe verharren.

Experimentieren Sie nun mit der Lautstärken-Hüllkurve in folgendem Programm. Tippen Sie:

```
10 env 1,10,1,100
20 sound 1,142,1000,1,1
run
```

Zeile 20 definiert eine Note mit der Tonperiode 142 (Kammerton a), die 10 Sekunden dauert, eine Anfangslautstärke von 1 hat und die Lautstärken-Hüllkurve 1 benutzt. Die Lautstärken-Hüllkurve 1 wird in Zeile 10 definiert. Sie besteht aus 10 Schritten, in denen die Lautstärke jeweils um 1 erhöht wird, und zwar nach je einer Sekunde (100 x 0,01 Sekunden).

Ändern Sie Zeile 10 des Programms wie folgt, um die unterschiedlichen Effekte der Hüllkurven zu hören:

```
10 env 1,100,1,10
10 env 1,100,2,10
10 env 1,100,4,10
10 env 1,50,20,20
10 env 1,50,2,20
10 env 1,50,15,30
```

Versuchen Sie zum Schluß noch diese Variation:

```
10 env 1,50,2,10
```

Sie merken hierbei, daß die Lautstärke nach der Hälfte der Zeit konstant bleibt. Das liegt daran, daß wir 50 Schritte von jeweils 0.1 Sekunden hatten. Die Zeit, in der die Lautstärke verändert wurde, betrug also insgesamt 5 Sekunden, während die Dauer des Tones im SOUND-Befehl mit 1000, also 10 Sekunden, angegeben war.

Probieren Sie jetzt selbst, verschiedene Töne zu erzeugen.

Wenn Sie eine kompliziertere Lautstärken-Hüllkurve bilden wollen, können Sie die Parameter ‹Schrittanzahl›, ‹Schrittgröße› und ‹Schrittzeit› in der ENV-Anweisung bis zu viermal wiederholen. Sie haben dann mehrere Abschnitte in derselben Hüllkurve.

Definition einer Ton-Hüllkurve

Das Schlüsselwort zur Definition einer Ton-Hüllkurve (engl. Tone ENvelope) heißt ENT. In seiner einfachsten Form besteht der Befehl aus vier Parametern und sieht folgendermaßen aus:

ENT ‹Hüllkurvennummer›,‹Schrittanzahl›,‹Tonperiode des Schrittes›,‹Schrittzeit›

Sehen wir uns die Parameter wieder einzeln an:

Hüllkurvennummer

Dies ist die Nummer (Ø bis 15), die einer bestimmten Hüllkurve gegeben wird, damit sie im SOUND-Befehl aufgerufen werden kann.

Schrittanzahl

Dieser Parameter bestimmt, wieviele Tonhöhen-Variationen der Ton während seiner Dauer durchlaufen soll. Wenn Sie z.B. in einen 10 Sekunden dauernden Ton 10 verschiedene Tonhöhen-Schritte von jeweils einer Sekunde Dauer einbauen wollen, müssen Sie für den Parameter ‹Schrittanzahl› den Wert 10 angeben.

Für die Schrittanzahl stehen die Werte Ø bis 239 zur Verfügung.

Tonperiode des Schrittes

Die Differenz zwischen den einzelnen Tonhöhen-Schritten kann zwischen -128 und 127 betragen. Mit negativen Schritten nimmt die Tonhöhe zu, mit positiven Schritten nimmt sie ab. Die kürzeste Tonperiode ist Ø. Dies darf beim Definieren der Ton-Hüllkurve nicht vergessen werden. Im Kapitel 'Übersicht' finden Sie eine Aufstellung aller Tonperioden.

Schrittzeit

Dieser Parameter bestimmt die Zeit zwischen den einzelnen Tonhöhen-Schritten in Einheiten zu Hundertstelsekunden (0,01 Sekunden). Es können Werte von Ø bis 255 eingesetzt werden, d.h. ein Schritt kann höchstens 2,56 Sekunden lang sein (Ø wird als 256 behandelt).

Beachten Sie, daß die Dauer aller Schritte zusammengenommen nicht länger sein sollte als die Dauer des Tones, die Sie im SOUND-Befehl angeben. Sonst hört der Ton auf, bevor alle Tonhöhen-Variationen durchlaufen sind. (In solch einem Fall verfällt der Rest der Hüllkurve.)

Ist dagegen der ‹Dauer›-Parameter im SOUND-Befehl größer als die Dauer aller Tonhöhen-Schritte zusammen, wird der Ton für den Rest seiner Dauer auf der letzten Tonhöhen-Stufe verharren.

Experimentieren Sie nun mit der Ton-Hüllkurve in folgendem Programm. Tippen Sie:

```
10 ent 1,100,2,2
20 sound 1,142,200,15,,1
run
```

Zeile 20 definiert einen Ton mit der Tonperiode 142 (Kammerton a), der 10 Sekunden dauert und eine Anfangslautstärke von 15 (maximale Laustärke) hat. Er hat keine Lautstärken-Hüllkurve - dies wird durch den Leer-Parameter ,, signalisiert - und benutzt die Ton-Hüllkurve 1. Die Tonhüllkurve 1 wird in Zeile 10 definiert. Sie besteht aus 100 Schritten, in denen die Tonperiode jeweils um 2 erhöht wird (d.h. die Tonhöhe nimmt ab), und zwar nach je 0,02 Sekunden (2 x 0,01 Sekunden).

Ändern Sie Zeile 10 des Programms nacheinander wie folgt, um die unterschiedlichen Effekte der Hüllkurven zu hören:

```
10 ent 1,100,-2,2
10 ent 1,10,4,20
10 ent 1,10,-4,20
```

Ersetzen Sie die Tonhüllkurve und den SOUND-Befehl jetzt durch folgendes Programm:

```
10 ent 1,2,17,70
20 sound 1,71,140,15,,1
30 goto 10
run
```

Drücken Sie zweimal **[ESC]**, um das Programm abzubrechen.

Sie können nun die Lautstärken-Hüllkurve, die Ton-Hüllkurve und den SOUND-Befehl in einem Programm kombinieren, um verschiedenartige Klänge zu erzeugen. Beginnen Sie mit:

```
10 env 1,100,1,3
20 ent 1,100,5,3
30 sound 1,142,300,1,1,1
run
```

Ersetzen Sie dann Zeile 20 mit:

```
20 ent 1,100,-2,3
run
```

Und ändern Sie schließlich alle Zeilen um:

```
10 env 1,100,2,2
20 ent 1,100,-2,2
30 sound 1,142,200,1,1,1
run
```

Wenn Sie eine kompliziertere Ton-Hüllkurve bilden wollen, können Sie die Parameter ‹Schrittanzahl›, ‹Tonperiode des Schrittes› und ‹Schrittzeit› in der ENT-Anweisung bis zu viermal wiederholen. Sie haben dann mehrere Abschnitte in derselben Hüllkurve.

Probieren Sie selbst einige Variationen aus. Ergänzen Sie den SOUND-Befehl durch ein Rauschen und erweitern Sie die Lautstärken- und Ton-Hüllkurve durch neue Abschnitte.

Im Kapitel 3 'Liste aller Befehle des AMSTRAD CPC6128 BASIC' finden Sie alle Details zu den einzelnen Befehlen für die Tonerzeugung. Wenn Sie sich mehr für den melodischen Aspekt der Tonerzeugung interessieren, lesen Sie den Abschnitt 'Der Ton macht die Musik' im Kapitel 9.

Teil 10: Kurze Einführung in AMSDOS und CPM

Was ist AMSDOS?

Wenn der Computer eingeschaltet oder zurückgesetzt wird, arbeitet er automatisch mit AMSDOS. AMSDOS ist die Abkürzung für Amstrad Disc Operating System (Amstrad Disketten-Betriebssystem). Es stellt folgende Befehle für die Dateiverwaltung zur Verfügung:

```
LOAD "Datei"
RUN "Datei"
SAVE "Datei"
CHAIN "Datei"
MERGE "Datei"
CHAIN MERGE "Datei"
OPENIN "Datei"
OPENOUT "Datei"
CLOSEIN
CLOSEOUT
CAT
EOF
INPUT #9
LINE INPUT #9
LIST #9
PRINT #9
WRITE #9
```

Darüberhinaus kennt AMSDOS noch eine Reihe von speziellen Steuerungs-Befehlen für die Disketten-Organisation.

Diese Befehle werden externe Befehle genannt. Ihnen wird der Balken | vorangestellt (@ Taste mit **[SHIFT]**).

Dies sind einige der gebräuchlicheren externen Befehle:

```
|a
|b
|tape (kann in |tape.in und |tape.out unterteilt werden)
|disc (kann in |disc.in und |disc.out unterteilt werden)
```

`|a` und `|b` werden beim Betrieb von zwei Diskettenlaufwerken verwendet, um anzuzeigen, für welches Laufwerk der dann folgende Befehl gilt.

Die Anweisung:

```
|a
load "Datei"
```

...z.B. sagt dem Computer, das genannte Programm von der im Laufwerk A befindlichen Diskette in den Speicher zu laden.

Wird kein Laufwerk mit `|a` oder `|b` angesprochen, oder wurde der Computer zurückgesetzt, arbeitet das System automatisch mit dem Laufwerk A.

Falls Sie nur das eingebaute Diskettenlaufwerk benutzen, können Sie es als Laufwerk A betrachten, und `|a` oder `|b` braucht nicht angegeben zu werden. Geben Sie `|b` ein, obwohl kein zusätzliches Laufwerk angeschlossen ist, erscheint folgende Meldung auf dem Bildschirm:

```
Drive B: disc missing
Retry, Ignore or Cancel
```

Sie sollten dann `C` für Cancel (= Befehl wiederrufen) eintippen.

Wenn sie eine Kassette benutzen möchten

Der Befehl `|tape` weist den Computer an, alle Lade- und Speicherbefehle usw. auf das externe Kassettenlaufwerk und nicht auf die Diskette zu beziehen. Ohne Angabe von `|tape` richtet der Computer nach dem Einschalten oder Zurücksetzen alle Befehle automatisch an das Diskettenlaufwerk.

Wollen Sie vom Kassettenbetrieb wieder auf Diskettenbetrieb umstellen, tippen Sie:

```
|disc
```

Es besteht auch die Möglichkeit, von Kassette zu laden und auf Diskette zu speichern. Dann schreiben Sie:

```
|tape.in
```

Dieser Befehl weist den Computer an, Daten von der Kassette einzulesen, aber im übrigen die Standardfunktion beizubehalten, d.h. auf Diskette zu schreiben.

Wenn Sie umgekehrt Daten von Diskette einlesen und auf Kassette speichern wollen, müssen Sie erst `|disc.in` eingeben, um den vorherigen Befehl `|tape.in` (s.o.) rückgängig zu machen. Dann weisen Sie den Computer mit `|tape.out` an, Daten auf Kassette zu speichern.

Die Befehle `|tape.in` und `|disc.in` bzw. `|tape.out` und `|disc.out` heben sich also gegenseitig auf.

Weitere Informationen zur Übertragung von Daten auf Diskette bzw. Kassette finden Sie im Kapitel 4 (Das Arbeiten mit Disketten und Kassetten) und Kapitel 5 (AMSDOS und CP/M).

Datentransfer zwischen Disketten bzw. Diskette und Kassette

In Teil 7 dieses Kapitels haben Sie erfahren, wie eine neue Diskette mit dem `DISCKIT3`-Programm auf Seite 1 Ihres CP/M-Systemdiskettensatzes formatiert wird und wie ein Programm von einer Diskette auf eine andere kopiert wird. Mit folgenden Steuerbefehlen:

```
|tape |disc |tape.in |tape.out |disc.in |disc.out |a |b
```

sind Sie nun in der Lage, Programme auf die Diskette in Laufwerk A, Laufwerk B (sofern angeschlossen) oder auf Kassette (sofern angeschlossen) zu speichern, bzw. von diesen Geräten zu laden.

Weitere externe Steuerungsbefehle:

```
|dir |drive |era |ren |user
```

lernen Sie im Kapitel 5 ('AMSDOS und CP/M') kennen.

Kopieren einer ganzen Diskette

Mit Hilfe des `DISCKIT3`-Programms auf Seite 1 Ihrer Systemdisketten können Sie den Inhalt einer Diskette auf eine andere übertragen.

Mit dieser Methode können Sie Reservekopien von den System-Disketten selbst machen.

Legen Sie Seite 1 Ihrer System-Disketten in das eingebaute Diskettenlaufwerk und schreiben Sie:

```
|cpm
```

Wenn das Bereitschaftszeichen A> erscheint, schreiben Sie:

```
disckit3
```

Nach wenigen Sekunden sehen Sie die DISC KIT-Eröffnungsmeldung oben auf dem Bildschirm, sowie:

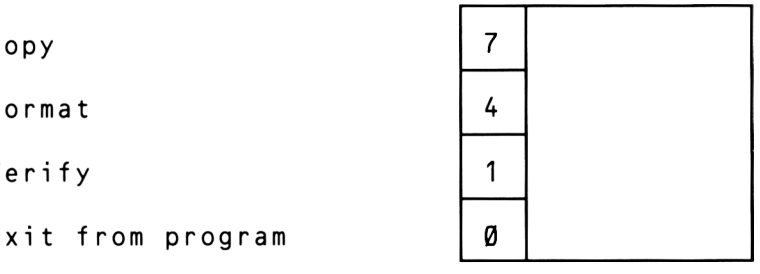
```
One drive found
```

Der Computer hat also verstanden, daß Sie mit dem DISC KIT-Dienstprogramm und nur einem Diskettenlaufwerk (dem eingebauten) arbeiten.

Haben Sie ein zweites Diskettenlaufwerk angeschlossen, lautet die Meldung:

```
Two drives found
```

Weiter unten auf dem Bildschirm sehen Sie folgendes Bild:



Dies ist das sogenannte DISC KIT-Hauptmenü. Die Nummern im Kasten beziehen sich auf die Funktionstasten auf der rechten Seite des Tastenfeldes (**fØ**, **f1**, **f4** und **f7**). Wenn Sie eine der Tasten drücken, läuft das zugehörige Programm ab.

Falls Sie an dieser Stelle **Ø** drücken, verlassen Sie das DISC KIT-Programm und kehren zum direkten Konsolmodus CP/M (Bereitschaftszeichen A>) zurück.

Da wir jetzt eine Diskette kopieren wollen, drücken wir Taste **f7**.

ACHTUNG!

Das Kopieren auf eine bespielte Diskette löscht ihren Inhalt.

Kopieren mit einem Diskettenlaufwerk

Wenn Sie nur mit dem eingebauten Laufwerk arbeiten, also kein weiteres Diskettenlaufwerk angeschlossen haben, erscheint nun die Meldung:




```
Copy
        Any other key to exit menu
```

Daraufhin nehmen Sie die CP/M-Systemdiskette heraus und legen die Diskette ein, deren Daten kopiert werden sollen. Möchten Sie die CP/M-Systemdiskette selbst kopieren, lassen Sie sie drin.

Befindet sich die Diskette, die kopiert werden soll (Quellendiskette) im Laufwerk, drücken Sie die **Y**-Taste für yes (d.h. ja, diese Diskette soll kopiert werden).

Der Computer untersucht das Format der Diskette und gibt es oben auf dem Bildschirm an. Nach einer Weile kommt die Meldung:

```
Insert disc to WRITE
Press any key to continue
```

Daraufhin nehmen Sie die Quellendiskette aus dem Laufwerk heraus und schieben die Diskette, auf die die Daten übertragen werden sollen (Zieldiskette), hinein. Dann drücken Sie eine beliebige Taste.

Oben auf dem Bildschirm wird nun das Format der Zieldiskette angezeigt (auch wenn es sich um eine neue, unformatierte Diskette handelt).

Ist die Zieldiskette nicht oder unzureichend formatiert, wird dies beim Kopiervorgang berücksichtigt. Sie sehen dann eine der folgenden Meldungen:

```
Disc isn't formatted (or faulty)
Going to format while copying
Disc will be system format
```

...oder eine ähnliche Mitteilung, je nachdem, welche Diskette kopiert wird.

Hat der Computer den Teil der Diskette übertragen, kommt folgende Aufforderung:

```
Insert disc to READ
press any key to continue
```

...woraufhin Sie wieder die Quellendiskette einlegen und eine beliebige Taste drücken.

Dieser Prozeß wiederholt sich einige Male, bis der gesamte Inhalt der Quellendiskette auf die Zieldiskette kopiert worden ist. Dann erscheint die Meldung:

```
Copy completed
Remove disc
Press any key to continue
```

Sie nehmen die Diskette heraus und drücken wieder eine beliebige Taste. Dann entscheiden Sie, ob Sie eine weitere Diskette kopieren wollen (**Y**-Taste drücken) oder zum DISC KIT-Menü zurückkehren möchten.

Schreibschutz

Denken Sie daran, daß Sie nichts auf eine Diskette mit offenem Schreibschutzloch kopieren können. Falls Sie dies versuchen, antwortet der Computer mit:

```
Disc write-protected
Insert disc to WRITE
R-etry or C-ancel
```

Sie tippen dann `C` für Cancel (Befehl wiederrufen) ein, nehmen die Diskette heraus und schieben die richtige Diskette mit geschlossenem Schreibschutzloch ins Laufwerk.

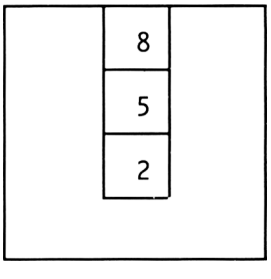
Lassen Sie das Schreibschutzloch einer Diskette, deren Inhalt Sie behalten wollen, immer geöffnet. Vor allem schließen Sie NIEMALS die Schreibschutzlöcher Ihrer CP/M-Systemdisketten!

Kopieren mit einem Doppellaufwerk

Gehen Sie wie oben beschrieben vor: Laden Sie das `DISCKIT3`-Programm von Seite 1 und wählen Sie `Copy` aus dem DISCKIT-Menü (Taste **f7** drücken).

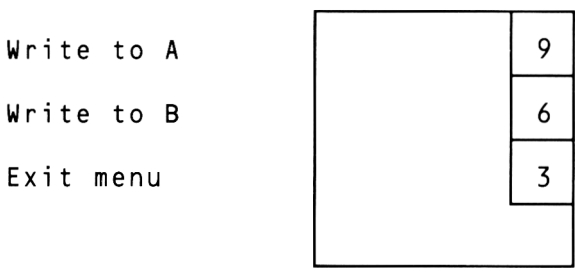
Danach steht Ihnen ein weiteres Menü zur Auswahl:

```
Read from A:
Read from B:
Exit menu
```



Wenn Sie mit einem Doppellaufwerk kopieren, brauchen Sie nicht ständig die Disketten auszuwechseln. Mit dem obigen Menü können Sie wählen, in welches Laufwerk Sie die Quellendiskette einlegen wollen. Drücken Sie die Taste **f8**, wenn die Diskette in Laufwerk A kopiert werden soll.

Nun erscheint das dritte Menü:



Hier haben Sie zu entscheiden, in welches Laufwerk Sie die Zieldiskette schieben wollen. Sie können zur Aufnahme dasselbe Laufwerk benutzen, in das Sie auch die Quellendiskette legen. Dies würde jedoch bedeuten, daß Sie die Disketten mehrere Male austauschen müßten. Sie hätten also keinen Vorteil von Ihrem Doppellaufwerk.

Das Kopieren geht einfacher und schneller, wenn Sie auf die Diskette in Laufwerk B kopieren und die Taste **f6** drücken.

Schieben Sie nun die beiden Disketten in die zugehörigen Laufwerke und tippen Sie `Y`, um den Kopiervorgang einzuleiten.

Wieder wird das Format der beiden Disketten angezeigt, und wenn die Zieldiskette nicht oder unzureichend formatiert ist, wird dies beim Kopieren berücksichtigt.

Ist das Kopieren beendet, meldet der Computer:

```
Copy completed
Remove both discs
Press any key to continue
```

Sie müssen dann BEIDE Disketten herausnehmen, bevor Sie weitere Operationen vornehmen können. Daraufhin können Sie eine weitere Diskette kopieren (**Y**-Taste drücken) oder zum DISCKIT-Hauptmenü zurückkehren (eine beliebige Taste drücken).

Prüfung der Disketten

Das DISKIT3-Programm bietet noch die Möglichkeit, eine Diskettenkopie auf ihre Richtigkeit hin zu überprüfen.

Das Format der Diskette wird angezeigt, und alle Dateien der Diskette werden gelesen. Etwaige Fehler in den Dateien erscheinen auf dem Bildschirm.

Zum Überprüfen einer Diskette legen Sie Seite 1 der Systemdisketten ein und tippen:

```
|cpm
```

Nach dem Bereitschaftszeichen A> schreiben Sie:

```
disckit3
```

Wählen Sie jetzt die Option Verify aus dem DISC KIT-Menü (Taste **f1**) und folgen Sie den Anweisungen auf dem Bildschirm.

Wenn Sie mit einem Doppellaufwerk arbeiten, haben Sie die Möglichkeit, die Diskette in Laufwerk A oder in Laufwerk B zu überprüfen.

Haben Sie die Diskette, deren Inhalt geprüft werden soll, in das entsprechende Laufwerk eingelegt, tippen Sie Y, um den Prüfvorgang einzuleiten.

Ist der Computer mit dem Prüfen fertig, bringt er folgende Meldung:

```
Verify completed
Remove disc
Press any key to continue
```

Falls Sie mit einem Doppellaufwerk arbeiten, müssen Sie BEIDE Disketten herausnehmen, bevor Sie weitere Operationen vornehmen können. Mit Y können Sie den Prüfvorgang wiederholen, oder Sie kehren zum DISC KIT-Menü zurück, indem Sie irgendeine andere Taste drücken.

Kopieren mit CP/M 2.2

Auf Seite 4 Ihres System-Diskettensatzes finden Sie ein weiteres DISC KIT Programm. Damit können Sie Disketten formatieren, kopieren oder überprüfen, die mit dem oder für den CPC664 oder den CPC464 + DDI1 - beide arbeiten mit CP/M 2.2 - geschrieben wurden. Dieses Programm heißt DISCKIT2, und es funktioniert genauso wie das eben beschriebene DISCKIT3-Programm.

Wenn Sie mit DISCKIT2 arbeiten wollen, legen Sie Seite 4 Ihrer Systemdisketten in das Laufwerk und schreiben:

```
Icpm
```

Nachdem Sie die CP/M 2.2 Eröffnungsmeldung und das Bereitschaftszeichen A> erhalten haben, tippen Sie:

```
disckit2
```

Nun wählen Sie aus den Menüs, was Sie brauchen und folgen den Anweisungen auf dem Bildschirm, wie für DISCKIT3 beschrieben.

Anmerkung: Beim Kopieren benutzt das DISCKIT2 -Programm den Bildschirmspeicher für die Daten. Deshalb sehen Sie während des Kopiervorganges wahllose Anzeigen auf dem Bildschirm.

Weitere Information

Informationen über die anderen Programme auf Ihrem CP/M Systemdiskettensatz finden Sie in den Kapiteln 4 bis 9.

Zum Schluß überprüfen Sie noch einmal, ob Sie die am Anfang des Handbuches genannten Vorsichtsmaßnahmen mit dem Titel 'WICHTIGE HINWEISE' beachtet haben:

'Hinweise zur Inbetriebnahme': Punkt 5, 6 und 7
'Hinweise zum Betrieb': Punkt 1, 2, 3, 4, 5, 6, 7 und 9

Teil 11: Das BANK MANAGER - Programm

Die zusätzlichen 64K

Der Speicher des CPC6128 enthält 128K RAM (Random Access Memory) in zwei Einheiten zu je 64K. CP/M Plus arbeitet die ganze Zeit mit dem gesamten Speicherplatz von 128K, während BASIC normalerweise nur die ersten 64K, nicht aber die zweiten 64K benutzt. Es wäre schade, wenn man die zusätzlichen 64K beim Programmieren in BASIC völlig ungenutzt lassen müßte. Deshalb wurde ein Programm geschrieben, mit dem der erweiterte Speicherbereich auch für BASIC zugänglich gemacht werden kann. Durch spezielle Befehle in diesem Programm können die zweiten 64K RAM als Speicherplatz für Bildschirmanzeigen oder für Zeichenfolgen ausgewiesen werden.

Das Programm mit diesen speziellen Befehlen heißt 'BANK MANAGER'. Mit 'Bank' bezeichnet man einen Teil des Speichers.

Der BANK MANAGER für Bildschirmanzeigen

Beim 6128 ist jederzeit irgendetwas auf dem Bildschirm zu sehen. Er benötigt 16K Speicherplatz, um die Information über die Farbe und Helligkeit jedes Bildpunktes (Pixel genannt) einer Bildschirmanzeige zu speichern. Der Speicher des 6128 kann bis zu sechs Bildschirmanzeigen - jede in einem 16K-Block - gleichzeitig speichern. Das BANK MANAGER-Programm gibt Ihnen die Möglichkeit, mit BASIC bis zu fünf der sechs Bildschirmanzeigen auf den Bildschirm zu holen und sie auszuwechseln.

Wenn Sie den Computer einschalten, sehen Sie ein Bild aus einem 16K-Speicherbereich (wir nennen ihn Block 1) der ersten 64K. Die anderen vier Bilder, Block 2, Block 3, Block 4 und Block 5 genannt, befinden sich im zweiten 64K-Speicher.

Nur ein Bild aus Block 1 (der ersten 64K) kann auf den Bildschirm gebracht werden. Um also ein Bild aus den zweiten 64K (Block 2 bis 5) tatsächlich auf dem Bildschirm zu sehen, muß es erst nach Block 1 tranferiert werden. Der BANK MANAGER hat alle nötigen Befehle, um die Bildschirmanzeigen zwischen den Blöcken hin- und herzuschieben. Mit `|SCREENCOPY` (Bildkopie) kann z.B. ein Bild von einem Block in einen anderen übertragen werden, wobei das im letzteren Block befindliche Bild gelöscht wird. Mit `|SCREENSWAP` (Bildtausch) kann der Inhalt zweier Blöcke ausgetauscht werden.

Wie die gerade behandelten AMSDOS-Befehle kennt auch der BANK MANAGER 'externe Befehle', die mit dem Balken | (@Zeichen mit **[SHIFT]**) beginnen.

Setzen Sie den Computer mit **[CONTROL] [SHIFT] [ESC]** zurück, legen Sie Seite 1 Ihres Systemdiskettensatzes ein und schreiben Sie:

```
RUN "BANKMAN"
```

Das Laden des Programms ist ausführlich im Kapitel 7 unter RSX (Resident System eXtensions) beschrieben. Es ist ratsam, sich Kenntnisse über RSX und über die Reservierung von Speicherplatz anzueignen, bevor Sie diese Teile in Ihr Programm einbauen. Für die folgenden Beispiele sind jedoch Kenntnisse über das Laden nicht erforderlich.

Tippen Sie:

```
MODE 1

PRINT "STANDARDBILD"
|SCREENSWAP,1,2
```

Der Text ist nun verschwunden. Stattdessen sehen Sie das, was als Bild 2 in Block 2 gespeichert wurde. Wenn der Computer gerade angeschaltet wurde, wird dies ein zufälliges Muster sein. Um das Muster zu löschen, schreiben Sie:

```
MODE 1
```

...und tippen dann ein:

```
PRINT "BILD 2"
|SCREENSWAP,1,2
```

Nun taucht Ihr erster Text wieder auf. Wenn Sie den Befehl `|SCREENSWAP,1,2` wiederholen, werden die beiden Bildschirmanzeigen wieder ausgetauscht. Sie können alle fünf Bilder gegeneinander austauschen. Beachten Sie aber dabei, daß sich das auf dem Bildschirm gezeigte Bild nur ändert, wenn die 1 im `|SCREENSWAP`-Befehl vorkommt.

Der zweite Befehl in diesem Zusammenhang heißt `|SCREENCOPY`. Damit können Sie den Inhalt eines Blocks in einen anderen kopieren, wobei der Inhalt des anderen Blocks überschrieben wird.

Tippen Sie:

```
MODE 1

PRINT "DIESES BILD SOLL KOPIERT WERDEN"
|SCREENCOPY,2,1
```

Der Inhalt von Block 1 wird in Block 2 geschrieben. Wenn Sie die Parameter umkehren und schreiben:

```
MODE 1

|SCREENCOPY,1,2
```

...wird das gegenwärtig gezeigte Bild von Bild 2 überschrieben.

Der erste Parameter gibt also an, in welchen Block das Bild kopiert werden soll, und der zweite Parameter bestimmt, aus welchem Block kopiert werden soll.

Beachten Sie, wenn Sie mit `|SCREENCOPY` arbeiten, daß ein unzusammenhängendes Bild auf dem Bildschirm erscheint, wenn die Modi der Bilder unterschiedlich sind, oder wenn das Bild nach dem letzten `MODE`-Befehl nach oben durchgerollt ist. Die SCREEN-Befehle des BANK MANAGER-Programms sind weniger für Bildschirmtexte als für graphische Darstellungen gedacht, bei denen das Wegrollen des Bildes selten vorkommt.

Der BANK MANAGER zum Speichern von Textfolgen

Der BANK MANAGER bietet außerdem vier Befehle, mit denen die zusätzlichen 64K als Speicherraum für Textfolgen (Strings) reserviert werden können.

Die meisten Programme bestehen aus zwei Teilen: Erstens aus den Anweisungen und zweitens aus den Daten, die mit dem Programm verarbeitet werden. Ein gutes Beispiel für ein solches Programm ist ein Adressbuch-Programm, in dem Namen, Adressen usw. in Tabellenform gespeichert werden. Solch ein Programm nennt man Datenbank.

Textfolgen (Strings) können im zweiten 64K Speicher hintereinander lückenlos gespeichert werden. Der Speicher, in dem die Strings gespeichert werden, läßt sich in gleich große Bereiche, sogenannte Sätze, unterteilen. Ein Satz hat eine festgesetzte Länge von 2 bis 255 Zeichen, während ein String in BASIC je nach Inhalt unterschiedlich lang sein kann. Die unübersichtliche String-Information wird in die Sätze eingeordnet. Nach jeder Operation, bei der Daten in einen Satz gespeichert oder von ihm abgerufen werden, wird zum nächsten Satz übergegangen, damit die nächste Operation an ihm ausgeführt wird. Der Satz, der für die nächste Operation bereit steht, wird 'aktueller Datensatz' genannt. Mit ihm wird automatisch gearbeitet, wenn kein anderer Datensatz angegeben wurde.

Diese Art der Speicherplatzverwaltung nennen wir 'RAMdisc', weil dieses System ähnlich wie Random Access-Diskettensysteme funktioniert, nur stattdessen mit RAM (Random Access Memory) arbeitet.

Lesen Sie die Beschreibungen zu den folgenden Befehlen, damit Sie ihre Funktion verstehen, wenn Sie sie auch nicht selber anwenden können, und probieren Sie dann die angegebenen Beispiele durch.

Der erste RAMdisc-Befehl heißt **!BANKOPEN**. Dieser Befehl gibt an, wieviele Zeichen jeder Satz maximal enthalten kann. Die Schreibweise des Befehls sieht so aus:

```
!BANKOPEN,n
```

n gibt dabei die Anzahl der Zeichen pro Satz an. Für n können Werte zwischen 0 und 255 eingesetzt werden, wobei die Werte 0 und 1 in der Praxis untauglich sind.

Mit **!BANKWRITE** kann ein String in einen aktuellen Datensatz gespeichert werden. Danach wird der nächste Datensatz 'aktuell' und steht für die nächste Operation bereit. Es gibt folgende Schreibweisen für diesen Befehl:

```
!BANKWRITE,!r%,a$
```

oder:

```
!BANKWRITE,!r%,a$,n
```

r% ist eine ganzzahlige Variable, ein Code, der etwas über die Operation aussagt. a$ ist eine Stringvariable. Sie enthält die Zeichen, die in den Satz geschrieben werden sollen. In der ersten Schreibweise wird der aktuelle Datensatz angesprochen. In der zweiten Schreibweise bezeichnet der Parameter n den Datensatz, in den geschrieben werden soll.

Mit **!BANKREAD** wird der Inhalt des aktuellen Satzes als String ausgegeben. Danach wird der folgende Datensatz 'aktuell' und steht für die nächste Operation bereit. Beim Lesen ändert sich der Inhalt eines Satzes nicht, so daß er beliebig oft gelesen werden kann. Für **!BANKREAD** gibt es folgende Schreibweisen:

```
! BANKREAD,!r%,a$
```

oder:

```
!BANKREAD,!r%,a$,n
```

r% ist eine ganzzahlige Variable, ein Code, der etwas über die Operation aussagt. a$ ist eine Stringvariable, in die der Inhalt des Satzes gelesen werden soll. In der ersten Schreibweise wird der aktuelle Datensatz angesprochen. In der zweiten Schreibweise bezeichnet der Parameter n den Datensatz, der gelesen werden soll.

Der letzte Befehl heißt **!BANKFIND**. Er wird benutzt, um die Datensätze nach einem bestimmten String abzusuchen. Ist die Suche erfolgreich, so wird die Nummer des Satzes ausgegeben, in dem der String sich befindet. **!BANKFIND** hat die Schreibweisen:

```
! BANKFIND,@r%,a$
```

oder

```
!BANKFIND,@r%,a$,n
```

oder

```
!BANKFIND,@r%,a$,n,m
```

r% ist eine ganzzahlige Variable. Sie enthält entweder die Nummer des Satzes, in dem der String gefunden wurde, oder einen Code, der anzeigt, daß der String nicht gefunden wurde. **a$** steht für den gesuchten String. Der Wahlparameter n gibt an, ab welchem Satz die Suche beginnen soll. Wird n nicht angegeben, wird ab dem aktuellen Datensatz gesucht. Der Wahlparameter m gibt an, bei welchem Satz die Suche eingestellt werden soll, auch wenn der String bis dahin nicht gefunden wurde. Wird m nicht angegeben, so werden die ganzen 64K durchsucht, obwohl das Ende der Datei möglicherweise schon viel früher erreicht ist.

Probieren Sie die BANK-Befehle nun aus. Wenn Sie schon mit dem BANK MANAGER gearbeitet haben, um die SCREEN-Befehle zu testen, und wenn Sie den Computer seither nicht zurückgesetzt haben, sind die Zusatzbefehle im Speicher vorhanden. Wenn nicht, müssen Sie Seite 1 Ihres Systemdiskettensatzes einschieben und

```
RUN "BANKMAN"
```

...eintippen. Dann schreiben Sie:

```
! BANKOPEN,20
```

Dadurch wird die Satzlänge auf 20 Zeichen festgelegt und 0 als aktueller Satz bestimmt. Nun schreiben Sie:

```
a$="ERSTER EINTRAG"+SPACE$(6)
```

Der String **a$** ist also genau 20 Zeichen lang.

Jetzt schreiben Sie:

```
r%=0
```

um den Beginn von **r%** bei **0** festzusetzen.

Jetzt schreiben Sie:

```
!BANKWRITE,@r%,a$
```

Damit wird a$ in den ersten Datensatz (Satz 0) geschrieben. Tippen Sie nun:

```
d$=SPACE$(20)
!BANKREAD,@r%,d$,0
PRINT d$
```

Die erste Anweisung setzt die Länge von d$ auf 20 Stellen fest. Damit ist genug Platz vorhanden, wenn der Satz gelesen wird. Mit der zweiten Anweisung wird Satz 0 gelesen und das Ergebnis in d$ eingesetzt. Da nach der vorherigen !BANKWRITE-Operation der aktuelle Satz der Satz 1 wäre, müssen wir 0 angeben, damit Satz 0 gelesen wird. (Sie erinnern sich: Wenn keine Satznummer angegeben wird, wird die Operation mit dem aktuellen Satz durchgeführt.) Schließlich wird das Ergebnis der Leseoperation auf dem Bildschirm wiedergegeben. Für d$ müßte 'ERSTER EINTRAG' und 6 Leerstellen erscheinen.

Schreiben Sie nun folgendes:

```
b$="ZWEI"+SPACE$(16)
c$="DREI"+SPACE$(16)
!BANKWRITE,@r%,b$,1
!BANKWRITE,@r%,c$,
```

Hiermit werden b$ und c$ in Satz 1 und 2 gesetzt. Wir haben im ersten !BANKWRITE-Befehl wieder den Wahlparameter benutzt, um Satz 1 als den Satz zu bestimmen, in den b$ gesetzt werden soll. Der folgende Satz wird danach automatisch zum aktuellen Satz, so daß wir nicht anzugeben brauchen, wo c$ hingesetzt werden soll. Er wird automatisch in Satz 2 plaziert.

Tippen Sie:

```
PRINT r%
```

Als Ergebnis müßte jetzt für r% die Zahl 2 auf dem Bildschirm erscheinen. Diese Nummer zeigt den Satz an, mit dem zuletzt operiert wurde, bzw. den aktuellen Satz minus 1. Im obigen Beispiel wurde zuletzt mit Satz 2 operiert, und der nächste Satz ist 3.

Mit diesem 'Rückkehr-Code' kann also in Erfahrung gebracht werden, welche Operation gerade ausgeführt wurde. Bei einer erfolgreichen Operation wird eine positive Zahl angegeben, die für die Satznummer steht. Bei einer erfolglosen Operation wird eine negative Zahl angegeben, die für einen Fehlercode steht. Bei !BANKWRITE und !BANKREAD können zwei mögliche Fehler ausgegeben werden:

-1 zeigt an, daß das Ende der Datei erreicht ist. Entweder wurden alle Sätze durchgegangen, oder es wurde ein Satz angegeben, der nicht existiert.

-2 zeigt Umschaltfehler bei der Bankauswahl an. Das dürfte eigentlich nicht vorkommen.

Probieren Sie einige weitere Beispiele aus:

```
d$=STRING$(20,"X")
!BANKOPEN,20
FOR n=1 to 3:!BANKWRITE,@r%,d$:NEXT
```

d$ besteht hier aus 20 'X'en. !BANKOPEN bestimmt 0 als aktuellen Satz, so daß !BANKWRITE den Inhalt der Sätze 0, 1 und 2 mit d$ überschreibt.

Schreiben Sie jetzt:

```
a$="EINS"
!BANKWRITE,@r%,a$,0
```

Auf diese Weise wird das Wort 'EINS' in den Satz 0 geschrieben und überschreibt dabei vier 'X'e. Füllen Sie d$ nun wieder mit Leerstellen aus:

```
d$=SPACE$(20)
```

Anschließend tippen Sie:

```
!BANKREAD,@r%,d$,0
```

Gehen wir die Schritte bis hierher noch einmal durch:

Alle drei Sätze enthalten Xe. Satz 0 enhält außerdem das Wort 'EINS'. d$ wurde dann mit 20 Leerstellen ausgefüllt. Und schließlich wurde Satz 0 wieder in d$ hineingelesen. Schreiben Sie nun:

```
PRINT d$
```

Als Ergebnis sollte nun 'EINSXXXXXXXXXXXXXXX' auf dem Bildschirm stehen. Hieran wird deutlich, was bei der Benutzung dieser Befehle zu berücksichtigen ist. Wenn ein String einen Satz nicht ganz ausfüllt, bleiben die alten Zeichen im Satz erhalten, soweit sie nicht überschrieben wurden. Deshalb empfiehlt es sich, den Satz mit Leerstellen oder CHR$(32)'s auszufüllen, bevor man die neue Information darin abspeichert. Damit erspart man sich das Lesen des Strings, um festzustellen, ob er noch Zeichen enthält, die dort nichts mehr zu suchen haben. Das gleiche gilt für den String, in den der Satz hineingelesen werden soll. Ist der String länger als der Inhalt des Satzes, bleiben am Ende des Strings Zeichen erhalten, die nicht dazugehören. Darum wurde d$ gelöscht (mit Leerstellen gefüllt), bevor Satz 0 in ihn hineingelesen wurde.

Es kann vorkommen, daß man einen String in einen Satz schreiben will, der nicht genug Platz bietet. In dem Fall werden die Zeichen, die nicht mehr hineinpassen, nicht aufgenommen.

Genauso kann es vorkommen, daß der Inhalt eines Satzes zu lang für den String ist, in den er gelesen werden soll. Auch hier werden die Zeichen, die nicht mehr hineinpassen, nicht berücksichtigt. In gewöhnlichen Operationen mit Strings würde BASIC den String automatisch auf die erforderliche Länge erweitern, doch mit einem externen Befehl ist das nicht möglich.

Kommen wir schließlich noch zum Befehl **|BANKFIND**. Mit ihm werden alle Sätze auf einen bestimmten Eintrag hin abgesucht. Wenn Satz 24 beispielsweise mit dem Wort **"KARL"** beginnt, könnte er mit folgendem Befehl aufgespürt werden:

```
| BANKFIND,@r%,"KARL"
```

Diese Möglichkeit ist besonders nützlich in Datenbank-Programmen, wenn z.B. ein Name oder eine Adresse gesucht wird.

|BANKFIND beginnt seine Suche im aktuellen Satz und kämmt von da aus alle Sätze durch, bis es den gesuchten String gefunden hat, oder bis das Ende der zweiten 64K, in denen die Sätze gespeichert sind, erreicht ist.

Man kann auch bestimmen, ab welchem Satz die Suche beginnen soll, indem man am Ende der Anweisung die betreffende Satznummer anfügt.

Durch Angabe einer zweiten Satznummer kann man bestimmen, wo die Suche abgebrochen werden soll.

|BANKFIND eignet sich auch für die Suche nach einem String, der nicht am Anfang eines Satzes steht. In dem Fall füllt man die Stellen vor dem gesuchten String mit **CHR$(Ø)**'s aus. Die **CHR$(Ø)**'s werden als Universalzeichen (wie **?** in CP/M Dateinamen) betrachtet, d.h. sie stehen für jedes beliebige Zeichen. Die Anweisung sieht dann z.B. so aus:

```
a$=STRING$ (1Ø,Ø)+"KARL"
|BANKFIND,@r%,a$,Ø
```

Jetzt würde untersucht, wo der String **"KARL"** zum ersten Mal auftritt. Er müßte zwischen der 11. und 14. Stelle eines Satzes stehen. Die ersten 10 Zeichen könnten eine Telefonnummer oder irgendeine andere Information enthalten, die von **|BANKFIND** ignoriert wird.

Die Nummer des Satzes, in dem der String gefunden wurde, wird für die Variable **r%** eingesetzt, die den Abfrage-Code darstellt (wenn **"KARL"** gefunden wird). Ansonsten wird der Abfrage-Code den Wert -3 enthalten.

Weitere Einzelheiten

Nähere Einzelheiten über den BANK MANAGER finden Sie im Kapitel 8. Darüberhinaus sollten Sie sich im Kapitel 7 (Teil 13 und 14) über RSX informieren.

Überprüfen Sie zum Schluß noch einmal, ob Sie folgende Punkte unter 'WICHTIGE HINWEISE' am Anfang des Handbuchs beachtet haben:

Punkt 1, 2, 3 und 9 aus dem Abschnitt 'Hinweise zum Betrieb'.

Damit ist nun unser elfteiliger Einführungskurs zum CPC6128 abgeschlossen. Sie kennen jetzt die Funktion der meisten Tasten auf der Tastatur, Sie können einige der einfacheren BASIC-Befehle einsetzen, Sie können eine nagelneue Diskette formatieren und somit gebrauchsfertig machen und Sie wissen über die wichtigsten Diskettenfunktionen, wie Laden und Speichern, Bescheid. Außerdem kennen Sie ein paar AMSDOS-, CP/M- und BANK MANAGER-Befehle.

In den folgenden Kapiteln werden die Funktionen des Computers und das AMSTRAD-BASIC genauer beschrieben. Sie werden Ihre Kenntnisse über das eingebaute Diskettenlaufwerk, sowie über AMSDOS und CP/M vertiefen können, und Sie werden eine neue Sprache kennenlernen: Dr. LOGO von Digital Research.

Viel Spaß beim Lesen und gutes Gelingen!

Kapitel 2
Das erste Programm

Sie haben jetzt die Einführung gelesen und der Computer steht eingeschaltet vor Ihnen. Sie wissen, wie man mit `FOR` und `NEXT` eine Schleife bildet, und wie man `IF` und `THEN` anwendet, um den Computer eine Anweisung ausführen zu lassen, wenn eine Bedingung erfüllt ist.

Sie werden es bald überdrüssig sein, auf dem ganzen Bildschirm nichts als Ihren Namen zu lesen. Sie werden ihn jetzt zu nützlicheren Aufgaben oder zur Unterhaltung verwenden wollen. Im nachfolgenden Kapitel sind alle AMSTRAD BASIC-Befehle samt einer Beschreibung ihrer 'Syntax' und ihrer Anwendung aufgeführt.

Mit dieser Liste sind Sie in der Lage, den Computer zu allen möglichen Aufgaben zu bewegen - soweit es Ihre Phantasie erlaubt.

Wenn Sie noch nie einen Computer benutzt haben, mag die Vorstellung, tatsächlich zu programmieren, etwas Furchteinflößendes an sich haben. Haben Sie keine Angst! Es ist bei weitem einfacher, als Sie denken, und ganz gewiß einfacher, als Sie von der Technologie und der Fachsprache her erwarten würden. Stellen Sie sich BASIC nicht als neue Sprache vor, sondern als eine Abwandlung des Englischen, die mit vielen Abkürzungen arbeitet, um Zeit zu sparen. Mit anderen Worten, sehen Sie `CLS` nicht als magischen Code an, sondern als Kurzform für `C`lear `S`creen.

Versuchen Sie, keine Angst vor BASIC zu haben. So werden Sie bald Vergnügen am Programmieren finden und die Früchte Ihrer Bemühungen ernten. Programmieren kann eine sehr lohnende Betätigung sein, besonders, wenn Sie Anfänger sind und mit dem Gerät und der Sprache experimentieren. Denken Sie immer daran, daß nichts, was Sie eintippen, Ihrem Computer schaden kann, es sei denn, sie schreiben aus Versehen auf Ihre CP/M Systemdiskette. Es lohnt sich immer, etwas Neues auszuprobieren.

Vorüberlegungen

Der Anfang ist für den Anfänger oft der schwierigste Teil des Programms. Sie sollten es allerdings vermeiden, sich gleich hineinzustürzen und in die Tasten zu hauen, ohne sich vorher Gedanken zu machen.

Als erstes sollten Sie überlegen, welche Möglichkeiten das Programm bieten soll, und wie die Ergebnisse auf dem Bildschirm dargestellt werden sollen.

Wenn Sie sich darüber klar geworden sind, können Sie anfangen, ein Programm zu schreiben, das Ihre Anforderungen erfüllt. Dabei sollten Sie immer daran denken, daß das Programm von Anfang bis Ende möglichst reibungslos laufen sollte. Es sollte möglichst wenig gesprungen, d.h. mit dem Befehl `GOTO` gearbeitet werden. Ein gutes Programm wird übersichtlich aufgebaut sein und Sie nicht in ein heilloses Durcheinander stürzen, wenn Sie versuchen, einen Fehler zu finden.

Zum Glück ist BASIC eine sehr nachsichtige und hilfsbereite Sprache. Wenn Sie z.B. etwas falsch gemacht haben, erscheinen häufig Fehlermeldungen auf dem Bildschirm. BASIC gestattet Ihnen auch, nachträglich etwas zu ergänzen. Neue Programmteile können mit minimalem Aufwand in das bestehende Programm eingebaut werden.

Wie man ein einfaches Programm schreibt

Unser erstes Programm soll die Namen und Telefonnummern unserer Freunde speichern. Wir nennen dieses Programm 'Telefonbuch'. Zunächst stellen wir uns die eben erwähnten Fragen: 'Welche Möglichkeiten soll das Programm bieten?' und 'Wie sollen die Ergebnisse dargestellt werden?'

Wir nehmen einmal an, das Programm soll bis zu 100 Namen und Telefonnummern speichern. Es soll so beschaffen sein, daß Sie eine gesuchte Telefonnummer erhalten, wenn Sie den zugehörigen Namen eintippen. Außerdem wollen Sie die Möglichkeit haben, eine vollständige Liste der gesamten Information auf dem Bildschirm zu sehen. Haben Sie gemerkt, daß wir uns schon automatisch Gedanken über die Darstellungsform der Ergebnisse auf dem Bildschirm gemacht haben?

So, jetzt können Sie die Finger auf die Tasten legen. Schreiben Sie als erstes den Titel:

```
10 REM Telefonbuch
```

Es ist nicht unbedingt notwendig, dem Programm einen Titel zu geben, aber wenn sich ein paar Programme angesammelt haben, helfen Ihnen die Titel, die Programme auf einen Blick zu unterscheiden.

Wir wissen, daß wir eine Zeichenfolge (den Namen einer Person) für eine Variable eingeben wollen (`INPUT`). Diese Variable nennen wir `NAME$`. Die Variable für die Telefonnummer nennen wir entsprechend `TEL$`.

Erinnern Sie sich an die Beispielprogramme aus dem Einführungskapitel? Dort wurde der INPUT-Befehl benutzt, um den Wert einer Variablen einzugeben. Wenn Sie also schreiben:

```
20 INPUT "Geben Sie den Namen an";NAME$
30 INPUT "Geben Sie die Telefonnummer an";TEL$
run
```

können Sie einen Namen eintippen, z.B. Johann, und danach seine Telefonnummer, z.B. 08245 510.

Das Programm hat diese Information nun gespeichert , aber es hat noch keine Ergebnisse auf dem Bildschirm gezeigt. Daher brauchen wir einen Programmteil, um die Information abfragen und auf dem Bildschirm sehen zu können. Im Moment müßten wir, um die Werte von NAME$ und TEL$ zu erhalten, Befehle verwenden wie:

```
PRINT NAME$.....und.....PRINT TEL$
```

Aber warten Sie! Wir sagten, daß wir bis zu 100 Namen und 100 Telefonnummern in dem Programm speichern wollen. Wir müssen doch hoffentlich nicht ein Programm mit 200 INPUT-Befehlen schreiben, jeden mit einem anderen Variablennamen, und dann 200 PRINT-Befehle, um eine Liste der Namen und Telefonnummern auf dem Bildschirm zu erhalten!? Keine Angst, für solche Fälle kann man im Computer Tabellen (Arrays genannt) einrichten. Ein Array ermöglicht Ihnen, für eine Gruppe von Datenelementen eine einzige Variable (z.B. NAME$) zu verwenden. Um die Datenelemente zu unterscheiden, werden sie mit einer Nummer gekennzeichnet, die hinter der Variablen in Klammern angegeben wird. Diese Nummer bezeichnet man als Index, und einen Ausdruck wie NAME$(27) nennt man 'indizierte Variable'. Wenn wir jetzt eine numerische Variable als Index verwenden, beispielsweise NAME$(x), können wir die ganze Liste von Variablen von 1 bis 100 verarbeiten, indem wir den Wert für x mit Hilfe einer Schleife (FOR x=1 TO 100) verändern. Da der Wert für x bei jedem Durchgang um 1 zunimmt, ändert sich der Index und bezieht sich jedesmal auf ein anderes Datenelement, d.h. auf einen anderen Namen des NAME$-Arrays.

Wir brauchen zwei Arrays, eines für NAME$ und eines für TEL$, mit einer 'Dimension' von jeweils 100 Elementen. Bevor wir ein Array einrichten, müssen wir seine DIMension festlegen. Wir überschreiben die Zeilen 20 und 30 daher mit folgenden Angaben:

```
20 DIM NAME$(100)
30 DIM TEL$(100)
```

Wir haben jetzt unsere Variablen festgelegt. Nun wollen wir ein Programm schreiben, das uns in die Lage versetzt, Namen und Telefonnummern in die Arrays einzutragen, damit wir sie später abfragen können. Wir fügen hinzu:

```
40 FOR x=1 TO 100
50 INPUT;" Name";NAME$(x)
60 INPUT " Telefon";TEL$(x)
70 NEXT
run
```

Schön und gut, aber vielleicht möchten wir nicht alle 100 Namen auf einmal eintragen. Außerdem ist die Darstellung des Programms auf dem Bildschirm höchst unbefriedigend. Es müßte ein wenig mehr Ordnung ins Bild gebracht werden. Zunächst einmal soll der Bildschirm von allem überflüssigen Text befreit werden, bevor wir einen neuen Namen mit Telefonnummer eingeben. Damit der Bildschirm nach jeder Eingabe gelöscht wird, schreiben wir in unser Programm:

```
45 CLS
```

Wie sagen wir dem Computer, daß wir mit der Eingabe von Informationen vorläufig fertig sind? Wenn wir (**[ESC]**) drücken, wird das Programm zwar gestoppt, aber sobald wir dann wieder **RUN** eintippen, verlieren wir all die Werte unserer sorgfältig eingegebenen Variablen!

Stattdessen können wir folgendes tun: Bevor das Programm einen neuen Namen aufnimmt, lassen wir den Computer prüfen, ob etwas für den Wert von **NAME$(x)** eingetippt wurde, mit anderen Worten, ob der Wert für **NAME$(x)** ein Leerstring ist oder nicht. Wenn (**IF**) das der Fall ist, dann (**THEN**) veranlassen wir das Programm, keine weiteren Informationen aufzunehmen. Haben Sie begriffen, wie wir das bewerkstelligen können? Fügen Sie hinzu:

```
55 IF NAME$(x)="" THEN 80
80 PRINT "Keine weiteren Eingaben"
```

Das Programm sagt Ihnen, wie die Eingabe beendet werden kann, wenn Sie hinzufügen:

```
47 PRINT "Druecke [RETURN] um Eingabe zu beenden"
```

Der nächste Teil des Programms soll bewirken, daß die gespeicherte Information in Form einer Liste auf dem Bildschirm gezeigt wird. Fügen Sie hinzu:

```
90 FOR x=1 TO 100
100 PRINT NAME$ (x);" ";TEL$(x)
110 NEXT
```

Wieder weiß das Programm nicht, wie es aufhören soll, bevor es das hundertste Element erreicht hat. Also fügen wir hinzu:

```
95 IF NAME$(x)="" THEN 120
120 PRINT "Ende der Liste"
```

In Zeile 95 wird festgestellt, ob NAME$(x) ein Leerstring ist, und wenn (IF) dies der Fall ist, dann (THEN) wird die Ausgabe beendet, indem die Zeilen 100 und 110 übergangen werden.

Das Programm sollte noch mehr Möglichkeiten bieten, wie wir anfangs überlegt hatten. Wir wollen den Computer jetzt veranlassen, einen bestimmten Namen ausfindig zu machen, den wir eingegeben haben. Fügen Sie hinzu:

```
130 INPUT "suche";SUCH$
140 FOR x=1 TO 100
150 IF INSTR(NAME$(x),SUCH$)=0 THEN 180
160 PRINT NAME$(x);" ";TEL$(x)
170 END
180 NEXT
190 PRINT "Name nicht gefunden"
run
```

In Zeile 150 sehen Sie einen neuen Befehl: INSTR. Er trägt dem Computer auf, den ersten STRing zu befragen (engl. INterrogate), wo der zweite String zum ersten Mal auftaucht. Er bewirkt, mit anderen Worten, daß der Wert für NAME$ mit dem Wert für SUCH$ (dies ist die Variable für den gesuchten Namen, die Sie in Zeile 130 eingegeben haben) verglichen wird. Findet INSTR den Namen (oder einen Teil davon) nicht, gibt es den Wert 0 aus. In diesem Fall geht das Programm weiter zu Zeile 180 und versucht es mit dem nächsten Wert von x. Ist das Programm alle 100 Werte von x durchgegangen, geht es zu Zeile 190 und sagt Ihnen, daß es den Namen nicht gefunden hat. Hat es den Namen hingegen ausfindig gemacht, ergibt INSTR nicht den Wert 0. Dann rückt das Programm von Zeile 150 zu 160 und zeigt den Namen und die Telefonnummer an. Mit Zeile 170 ENDet das Programm.

Sie sehen, daß sich unser Programm jetzt rasch entwickelt, aber es gibt noch viel zu tun. Wir wollen uns jetzt einen Moment lang zurücklehnen und einige Nachteile dieses Programms betrachten. Beginnen wir mit der Abfolge des Programms: Erst tippen Sie die Information ein, dann erscheint eine Darstellung der Information auf dem Bildschirm, dann suchen Sie einen bestimmten Namen.

Was ist, wenn...?

Und was ist, wenn Sie das Programm nicht in dieser Reihenfolge haben wollen? Was machen Sie, wenn Sie zuerst einen Namen suchen wollen, den Sie gestern abgespeichert haben? Und was machen Sie, wenn Sie weitere Namen und Telefonnummern hinzufügen wollen? Über all diese Fragen müssen Sie nachdenken. Die Lösung solcher Probleme ist die eigentliche Aufgabe des Programmierers. Wie wir schon erwähnt haben, ist BASIC eine Sprache, die es Ihnen freundlicherweise gestattet, nachträglich etwas ins Programm einzufügen, aber ein guter Programmierer wird derartige Schwierigkeiten vorhersehen.

Ein weiterer großer Nachteil des Programms besteht darin, daß die Werte der Variablen in den Arrays alle in einem Teil des Computers gespeichert sind, der gelöscht ist, wenn Sie das Programm laufen lassen. Sie möchten natürlich nicht jedesmal die gesamte Information eintippen, wenn Sie das Programm benutzen.

Sie müssen also die Möglichkeit haben, die Werte aller `NAME$`-und `TEL$`-Variablen zu speichern, bevor Sie den Computer abschalten, und umgekehrt müssen Sie die Möglichkeit haben, den Computer mit den Werten der Variablen zu laden, wenn Sie das Programm starten wollen.

Lösungen

Um das erstgenannte Problem zu lösen (die Reihenfolge), müssen Sie das Programm so schreiben, daß es die verschiedenen Funktionen nach Ihrer Wahl ausführt. Diese Art von Programm wird mit dem etwas irreführenden Wort 'Menü' bezeichnet. (Das englische Wort 'menu' bedeutet 'Speisekarte'. Ein Menü wird also wie eine Speisekarte auf dem Bildschirm präsentiert, aus der Sie das Gewünschte auswählen.) Wenn Sie schon einmal einen Geldautomaten an einem Bankgebäude bedient haben, haben Sie bereits mit einem menügesteuerten Programm gearbeitet! Wir bauen jetzt ein Menü in unser Programm ein:

```
32 PRINT "1. Information eingeben"
33 PRINT "2. Information auflisten"
34 PRINT "3. Suchen"
35 PRINT "4. Information speichern"
36 PRINT "5. Information laden"
37 INPUT "Menu-Auswahl";ms
38 ON ms GOSUB 40,90,130

85 RETURN
125 RETURN
170 RETURN
200 RETURN
```

Wir haben also zunächst die Wahlmöglichkeiten aufführen lassen, dann die Auswahl eingegeben (INPUT) und sie in der Variablen ms abgelegt. Das Kommando ON ms GOSUB in Zeile 38 sagt dem Programm, daß es zur ersten Zeilennummer des Unterprogramms (sog. Subroutine), in diesem Fall zu Zeile 40 gehen soll, wenn ms gleich 1 ist. Wenn ms gleich 2 ist, soll es zur zweiten Zeilennummer der Subroutine gehen usw.

Da nun jede der einzelnen Programmfunktionen eine Subroutine darstellt, die durch das Kommando ON ms GOSUB aufgerufen werden kann, muß hinter jeder Funktion das Kommando RETURN stehen. Die RETURN-Befehle wurden folglich oben ergänzt.

Erinnern Sie sich noch, was nach dem Befehl RETURN geschieht? Das Programm kehrt von der Subroutine zu der Stelle zurück, die auf das zuletzt befolgte GOSUB-Kommando folgt. In vorliegenden Fall kehrt es zu der Anweisung zurück, die auf Zeile 38 folgt, d.h. es geht zu Zeile 40, dem Punkt, an dem die Information eingegeben wird. Damit sind wir jedoch nicht einverstanden, also fügen wir hinzu:

```
39 GOTO 32
```

Das Programm macht nun eine Schleife und zeigt noch einmal das Menü an. Lassen Sie das Programm jetzt noch einmal durchlaufen (Kommando RUN), um zu sehen, wie weit wir vorangekommen sind.

Sehen wir uns also einen Moment lang das Listing des Programms an. (Wenn das Programm noch läuft, drücken Sie zweimal die **[ESC]**-Taste.) Schreiben Sie nun:

```
LIST
```

Sie sollten folgendes Bild vor sich haben:

```
10 REM Telefonbuch
20 DIM NAME$(100)
30 DIM TEL$(100)
32 PRINT "1. Information eingeben"
33 PRINT "2. Information auflisten"
34 PRINT "3. Suchen"
35 PRINT "4. Information speichern"
36 PRINT "5. Information laden"
37 INPUT "Menu-Auswahl";ms
38 ON ms GOSUB 40,90,130
39 GOTO 32
40 FOR x=1 TO 100
45 CLS
47 PRINT "Druecke RETURN um Eingabe zu beenden"
```

```
50 INPUT ;" Name";NAME$(x)
55 IF NAME$(x)="" THEN 80
60 INPUT " Telefon";TEL$(x)
70 NEXT
80 PRINT "keine weiteren Eingaben"
85 RETURN
90 FOR x=1 TO 100
95 IF NAME$(x)="" THEN 120
100 PRINT NAME$(x);" ";TEL$(x)
110 NEXT
120 PRINT "Ende der Liste"
125 RETURN
130 INPUT "Suche";SUCH$
140 FOR x=1 TO 100
150 IF INSTR(NAME$(x),SUCH$)=0 THEN 180
160 PRINT NAME$(x);" ";TEL$(x)
170 RETURN
180 NEXT
190 PRINT "Name nicht gefunden"
200 RETURN
```

Sie werden bemerkt haben, daß in manchen Teilen des Programms der Platz knapp wird, um weitere Anweisungen einzufügen. Wir schaffen uns einfach mehr Platz und Ordnung, indem wir die Zeilen neu numerieren. Schreiben Sie:

```
RENUM
LIST
```

Jetzt müßte die Numerierung folgendermaßen aussehen:

```
10 REM Telefonbuch
20 DIM NAME$(100)
30 DIM TEL$(100)
40 PRINT 1. "Information eingeben"
50 PRINT 2. "Information auflisten"
60 PRINT 3. "Suchen"
70 PRINT 4. "Information speichern"
80 PRINT 5. "Information laden"
90 INPUT "Menu-Auswahl";ms
100 ON ms GOSUB 120,210,270
110 GOTO 40
120 FOR x=1 TO 100
130 CLS
140 PRINT "Druecke [RETURN] um Eingabe zu beenden"
150 INPUT;" Name";NAME$(x)
160 IF NAME$(x)="" THEN 190
```

```
170 INPUT " Telefon";TEL$(x)
180 NEXT
190 PRINT "keine weiteren Eingaben"
200 RETURN
210 FOR x=1 TO 100
220 IF NAME$(x)="" THEN 250
230 PRINT NAME$(x);" ";TEL$(x)
240 NEXT
250 PRINT "Ende der Liste"
260 RETURN
270 INPUT "Suche";SUCH$
280 FOR x=1 TO 100
290 IF INSTR(NAME$(x),SUCH$)=0 THEN 320
300 PRINT NAME$(x);" ";TEL$(x)
310 RETURN
320 NEXT
330 PRINT "Name nicht gefunden"
340 RETURN
```

Das sieht schon besser aus. Doch nun weiter im Programm. Als nächstes fügen wir eine Anweisung hinzu, die bewirkt, daß neue Namen und Telefonnummern, die Sie ergänzen wollen, in das erste freie Element des Arrays eingeordnet werden. Dieses Mal verwenden wir das neue Kommando LEN (engl. length), um die Länge einer Zeichenfolge zu bestimmen. Wir wollen also folgendes ausdrücken:

Wenn (IF) die Länge (LEN) des Namens (NAME$(x)) größer als 0 ist, mit anderen Worten, wenn schon ein Name in diesem Element des Arrays eingetragen ist, dann (THEN) springe zu Zeile 180 (die zur nächsten (NEXT) Zeile im Array führt).

Wieder sehen Sie, wie nah sich BASIC am Englischen orientiert. Es ist wahrhaftig keine neue Fremdsprache! Die Anweisung lautet:

```
135 IF LEN(NAME$(x))>0 THEN 180
```

So einfach ist die Lösung! Mit Ihrem BASIC-Wortschatz und ein wenig Überlegung können Sie solche Probleme immer schnell lösen. Es ist fast immer mindestens ein Befehl vorhanden, mit dem Sie Schwierigkeiten beim Programmieren meistern können. Und je mehr Sie programmieren, um so leichter werden Ihnen Lösungen einfallen.

Unser nächstes Anliegen war, den Inhalt der Variablen zu sichern, während das Programm läuft. In Teil 7 der Einführung wurde erklärt, wie man das Programm selbst, unter Verwendung des Kommandos SAVE, speichert. Aber das Programm stellt nur den Rahmen dar, mit dessen Hilfe die Variablenwerte eingetippt und über den Bildschirm abgerufen werden können. Wenn man das Programm speichert, speichert man nur den Rahmen, nicht aber die Werte der Variablen.

Deshalb müssen wir einen Programmteil schreiben, der die Werte der Variablen auf einer Diskette speichert. Dazu schaffen wir eine spezielle Datei. Zunächst eröffnen wir die Datei (Kommando: OPENOUT) und geben ihr einen Namen, sagen wir 'Datei'. Dann schreiben (WRITE) wir die Werte der Variablen NAME$(x) und TEL$(x) von 1 bis 100 in die Datei. Danach schließen wir die Datei mit dem Kommando CLOSEOUT und kehren mit **[RETURN]** zum Menü zurück. Diesen Programmteil wollen wir jetzt ab Zeile 350 ergänzen. Damit wir nicht in jeder neuen Zeile die Nummer eintippen müssen, arbeiten wir mit dem Befehl

```
AUTO 350
```

So werden die Zeilen ab 350 AUTOmatisch numeriert:

```
350 OPENOUT "Datei"
360 FOR x=1 TO 100
370 WRITE #9,NAME$(x),TEL$(x)
380 NEXT
390 CLOSEOUT
400 PRINT "Datei gespeichert"
410 RETURN
```

Wenn Sie die Zeile 410 getippt und die **[RETURN]**-Taste gedrückt haben, drücken Sie **[ESC]**, um die automatische Numerierung abzubrechen.

Wir haben jetzt unser Menü um eine weitere Auswahlmöglichkeit bereichert und müssen daher die Zahlenfolge in dem Befehl ON ms GOSUB in Zeile 100 ergänzen. Sie EDITieren also Zeile 100 und erhalten folgendes Bild:

```
100 ON ms GOSUB 120,210,270,350
```

Jetzt wird jedesmal, wenn Sie die vierte Möglichkeit aus dem Menü wählen, die Information, die Sie eingegeben haben, auf Diskette gespeichert.

Sie haben bemerkt, daß in Zeile 370, wo das Programm die Werte der Variablen NAME$ und TEL$ auf die Diskette schreibt (WRITE), nach dem Wort WRITE der Ausdruck #9 erscheint. Das Zeichen # sagt dem Computer, welchem 'Stream' er die Daten zuordnen soll. Der Computer hat 10 Streams.

Wenn er die Daten zu den Streams 0 bis 7 (#0 bis #7) schickt, erscheinen sie auf dem Bildschirm, da die Streams #0 bis #7 Bildschirm-Streams (auch Fenster oder WINDOWs genannt) darstellen.

Über Stream #8 werden die Daten zum Drucker geschickt, falls einer angeschlossen ist.

Daten, die zum Stream #9 gelenkt werden, wie wir es in Zeile 370 getan haben, landen im Diskettenlaufwerk.

Kleine Randbemerkung

An dieser Stelle soll noch etwas zum Kommando AUTO gesagt werden, das wir eben verwendet haben. Wenn Sie die Zeilennummer auslassen und lediglich schreiben:

```
AUTO
```

wird der Computer die Zeilenumerierung mit 10 beginnen und nach jedem **[RETURN]** 10 Nummern weitergehen. Wenn Sie vorher schon die Nummern 10, 20, 30 usw. benutzt haben, erscheint nach jedem **[RETURN]** automatisch der Inhalt der Zeilen 10, 20, 30 usw. auf dem Bildschirm und kann editiert werden. Auf diese Weise wird das Editieren von aufeinanderfolgenden Zeilen mit gleichem Zeilennummernabstand erleichtert.

Zurück zum Programm

Wir haben also Anweisungen gegeben, damit die Information auf Diskette gespeichert werden kann. Mit dem letzten Hauptteil des Programms sollen die Daten von der Diskette wieder zum Gebrauch bereitgestellt (geladen) werden. Wir müssen also die Menü-Auswahl in Zeile 100 wieder um eine Zahl erweitern. Editieren Sie Zeile 100 wie folgt:

```
100 ON ms GOSUB 120,210,270,350,420
```

Wie sieht nun die Anweisung zum Laden der Information aus? Zunächst eröffnen wir die Datei auf der Diskette, die wir 'Datei' genannt haben, mit dem Befehl OPENIN. Dann laden wir alle Werte für die Variablen NAME$(x) und TEL$(x) von 1 bis 100 von der Diskette (Stream #9) in den Computer (INPUT). Danach schließen wir die Datei (CLOSEIN) und kehren zum Menü zurück **[RETURN]**. Geben Sie also folgende Answeisungen ein:

```
420 OPENIN "Datei"
430 FOR x=1 TO 100
440 INPUT #9,NAME$(x),TEL$(x)
450 NEXT
460 CLOSEIN
470 PRINT "Datei geladen"
480 RETURN
```

Das Ende vom Anfang

Jetzt haben wir ein Programm geschrieben, das all die Funktionen erfüllt, die wir zur Verfügung haben wollten, als wir überlegten, 'welche Möglichkeiten das Programm bieten soll'. Bleibt uns nur noch, die 'Darstellung der Ergebnisse auf dem Bildschirm' zu verbessern.

Der Anfang vom Ende

Fügen wir also einige Anweisungen hinzu, um die Darstellung des Programms zu verschönern.

```
34 MODE 1
```

So wird der Bildschirmmodus festgelegt und der Bildschirm am Anfang des Programms gelöscht. Fügen Sie hinzu:

```
36 WINDOW #1,9,34,10,14
```

Lassen Sie sich durch diese scheinbar schwierige Anweisung nicht verwirren. Wir schaffen dadurch lediglich ein kleines Fenster auf dem Bildschirm für unser Menü. Nach dem Wort 'WINDOW' geben wir an, welcher Stream-Nummer das Fenster zugeordnet werden soll. (Sie erinnern sich, daß wir 8 Fenster von #0 bis #7 zur Verfügung haben.) Da wir wissen, daß alle Angaben auf dem Bildschirm über Stream #0 laufen, wenn nichts anderes angegeben ist, werden wir für unser kleines Fenster nicht Stream #0 in Anspruch nehmen, sonst wird alles, was das Programm auf den Bildschirm bringt, in unserem Fenster erscheinen. Deshalb geben wir einen anderen Stream zwischen #1 und #7 an. Wir haben Stream #1 gewählt. Die vier Zahlen nach #1 teilen dem Computer die gewünschte Größe des Fensters mit. Die Nummern bestimmen die Begrenzung des Fensters nach links, rechts, oben und unten. Sie beziehen sich auf Textspalten und Zeilennummern (genauso wie beim LOCATE-Befehl). In unserem Beispiel bestimmen wir also (nachdem wir angegeben haben, daß wir Stream #1 benutzen wollen), daß das Fenster von Spalte 13 bis Spalte 30 reicht, und daß es oben von Zeile 10 und unten von Zeile 14 begrenzt wird.

Damit nun alle Wahlmöglichkeiten des Menüs auf dem Bildschirm in Fenster #1 erscheinen, müssen wir die Zeilen 40 bis 80 wie folgt editieren:

```
40 PRINT #1,"1. Information eingeben"
50 PRINT #1,"2. Information auflisten"
60 PRINT #1,"3. Suchen"
70 PRINT #1,"4. Information speichern"
80 PRINT #1,"5. Information laden"
```

Nun fügen wir hinzu:

```
85 LOCATE 7,25
```

Auf diese Weise wird die INPUT-Anweisung des Menüs in Zeile 90 so positioniert, daß es ordentlicher aussieht.

Damit der Bildschirm immer frei ist, wenn Sie etwas aus dem Menü wählen, fügen Sie hinzu:

```
95 CLS
```

Zu guter Letzt hängen wir noch die folgenden 3 Zeilen an, damit das Programm eine Pause macht, bevor es zum Menü zurückkehrt:

```
103 LOCATE 9,25
105 PRINT "Mit beliebiger Taste ins Menu"
107 IF INKEY$="" THEN 107
```

Zeile 103 sagt dem Computer, an welche Stelle er die Mitteilung aus Zeile 105 setzen soll. Zeile 107 fragt die Tastatur, welche Zeichenfolge eingegeben wird (indem eine Taste gedrückt wird). Wenn es entdeckt, daß ein Leerstring eingegeben wurde (weil keine Taste gedrückt wurde), kehrt das Programm in einer Schleife solange zur selben Anweisung zurück, bis INKEY$ einen String findet, d.h. bis eine Taste gedrückt wurde. Dies ist eine wirkungsvolle Art, eine Pause ins Programm einzubauen. BASIC wird jetzt erst zur nächsten Zeile vorgehen, wenn irgendeine Taste gedrückt wird.

Nun haben wir das fertige Programm also vor uns. Oder fehlt noch etwas? Nun, Sie könnten es noch mit verschiedenen anderen Fähigkeiten ausstatten: Namen und Telefonnummern zu ändern oder zu löschen, die Liste in alphabetische Reihenfolge zu bringen oder sie auszudrucken. All diese Erweiterungen des Programms sind machbar. Eigentlich können Sie ein Programm immer weiter verbessern und straffen, besonders wenn Sie einen Computer mit solch vielfältigen Möglichkeiten wie dem 6128 besitzen. Wir müssen notgedrungen irgendwo eine Grenze setzen, und deshalb verlassen wir jetzt unser 'Telefonbuch' in der Hoffnung, daß wir Ihnen das eine oder andere über die Kunst, ein Programm aufzubauen, nahebringen konnten. Gestalten Sie die Numerierung noch etwas übersichtlicher, indem Sie

```
RENUM
```

eintippen und dann speichern Sie das Programm auf Diskette oder löschen es. Doch nicht zu voreilig - es könnte Ihnen noch gute Dienste leisten, wenn Sie die Namen und Telefonnummern Ihrer Freunde speichern wollen.

```
10 REM Telefonbuch
20 DIM NAME$(100)
30 DIM TEL$(100)
40 MODE 1
50 WINDOW #1,9,34,10,14
60 PRINT #1,"1. Information eingeben"
70 PRINT #1,"2. Information auflisten"
80 PRINT #1,"3. Suche"
90 PRINT #1,"4. Information speichern"
100 PRINT #1,"5. Information laden"
110 LOCATE 7,25
120 INPUT "Menu-Auswahl";ms
130 CLS
140 ON ms GOSUB 190,290,350,430,500
150 LOCATE 9,25
160 PRINT "Mit beliebiger Taste ins Menu"
170 IF INKEY$="" THEN 170
180 GOTO 40
190 FOR x=1 TO 100
200 CLS
210 IF LEN(NAME$(x))>0 THEN 260
220 PRINT "druecke [RETURN] um Eingabe zu beenden"
230 INPUT;" Name";NAME$(x)
240 IF NAME$(x)="" THEN 270
250 INPUT " Telefon";TEL$(x)
260 NEXT
270 PRINT "keine weiteren Eingaben"
280 RETURN
290 FOR x=1 TO 100
300 IF NAME$(x)="" THEN 330
310 PRINT NAME$(x);" ";TEL$(x)
320 NEXT
330 PRINT "Ende der Liste"
340 RETURN
350 INPUT "Suche";SUCH$
360 FOR x=1 TO 100
370 IF INSTR(NAME$(x),SUCH$)=0 THEN 400
380 PRINT NAME$(x);" ";TEL$(x)
390 RETURN
400 NEXT
410 PRINT "Name nicht gefunden"
420 RETURN
430 OPENOUT "Datei"
440 FOR x=1 TO 100
450 WRITE #9,NAME$(x),TEL$(x)
```

Kapitel 3 Liste aller Befehle des AMSTRAD CPC 6128 BASIC

Wichtiger Hinweis

Es ist unbedingt erforderlich, daß Sie die in diesem Kapitel verwendete Terminologie und Notation kennen. Bei der Erklärung der Schreibweise der einzelnen Befehle tauchen verschiedene Arten von Klammern auf, die jeweils ihre eigene Bedeutung haben. Achten Sie also darauf, welche Klammer verwendet wird!

Jeder Teil eines Befehls, der nicht in Klammern steht, muß wie angegeben eingetippt werden. Der Befehl `END` z.B. sieht folgendermaßen aus:

```
END
```

Sie müssen also das Wort `END` Buchstabe für Buchstabe eintippen.

Wenn ein Ausdruck in spitzen Klammern ‹› steht, z.B.:

‹Zeilennummer›

...brauchen Sie weder die spitzen Klammern, noch das, was in den Klammern steht, einzutippen. Im obigen Beispiel wird lediglich angezeigt, welcher Datentyp in dem Befehl eingesetzt werden soll. Für:

`EDIT` ‹Zeilennummer›

...tippen Sie also z.B.:

```
EDIT 100
```

...ein.

Runde Klammern `()` müssen mit angegeben werden. Heißt es z.B.:

`COS (`‹numerischer Ausdruck›`)`

... so muß die Zahl, deren COSinus gefragt ist, in runden Klammern stehen, etwa so:

```
PRINT COS(45)
```

Eckige Klammern [] besagen, daß die darin eingeschlossene Angabe dem Befehl wahlweise hinzugefügt werden kann.

```
RUN [‹Zeilennummer›]
```

z.B. bedeutet, daß der Befehl RUN nicht durch einen Parameter ergänzt werden muß, daß Sie jedoch die Möglichkeit haben, ihn durch den Parameter ‹Zeilennummer› zu erweitern. Der Befehl könnte entweder als:

```
RUN ...oder als: RUN 100
```

...eingegeben werden.

Sonderzeichen

& oder &H	kennzeichnet die nachfolgende Konstante als hexadezimal (sedezimal)
&X	kennzeichnet die nachfolgende Konstante als binär (dual)
#	kennzeichnet die Ein-/Ausgabeeinheit (Stream)

Datentypen

Zeichenfolgen (Strings) können 0 bis 255 Zeichen lang sein. Ein ‹Textausdruck› ist ein Ausdruck, der den Wert einer Zeichenfolge wiedergibt. Zeichenfolgen können mit einem '+'-Zeichen aneinandergehängt werden. Dabei darf ihre Gesamtlänge 255 Zeichen nicht überschreiten.

Ein ‹numerischer Ausdruck› kann entweder ganzzahlig oder reell sein, d.h. die Zahlenwerte werden ohne bzw. mit Dezimalpunkt dargestellt. Ein ‹ganzzahliger Ausdruck› kann die Werte von –32768 bis –32767 annehmen. Ein ‹reeller Ausdruck› kann im Bereich von –1.7E+38 bis +1.7E+38 liegen (E bedeutet 'mal 10 hoch'). Reelle Größen haben eine Genauigkeit von etwas über neun Stellen. Die kleinste nahe 0 darstellbare Zahl ist 2.9E–39.

Ein ‹numerischer Ausdruck› hat immer einen Zahlenwert als Ergebnis. Dabei kann es sich um eine echte Zahlenangabe, eine Variable (ganzzahlig oder reell) oder einen mathematischen Ausdruck handeln. Kurz, alles was keinen ‹Textausdruck› darstellt, fällt in diese Kategorie.

Die ‹Ein-/Ausgabeeinheit› (‹Stream-Ausdruck›) stellt einen ‹numerischen Ausdruck› dar, der das Textfenster, den Drucker oder die Diskette bezeichnet.

‹Liste von:‹Einheit› bezeichnet einen Parameter, der eine Liste von Dateneinheiten enthält, die durch Kommata getrennt werden. Die Liste kann eine oder beliebig viele Einheiten umfassen, soweit es die Zeilenlänge (= 255 Zeichen) erlaubt.

Durch ein Typenkennzeichen hinter dem Variablennamen wird die Art des Ausdrucks gekennzeichnet. Dabei bedeutet:

%	ganzzahlig
!	reell (alle Variablen werden als reelle Zahlen angesehen, wenn kein Typenzeichen angegeben ist.)
$	String (Zeichenfolge)

Bitte beachten Sie, daß alle AMSTRAD CPC 6128 BASIC-Befehlsbeschreibungen folgendermaßen aufgebaut sind:

BEFEHL

Befehlsschreibweise (Syntax)

Beispiel

Beschreibung

Verwandte Befehle

Befehle sind entweder:

KOMMANDOS, die sofort ausgeführt werden, oder
FUNKTIONEN, die den Wert eines Ausdrucks berechnen, oder
OPERATOREN, die logische Verknüpfungen herstellen.

BASIC wandelt alle mit kleinen Buchstaben eingegebenen Befehle in Großbuchstaben um, wenn das Programm aufge`LIST`et wird. Wenn Sie die Befehle mit kleinen Buchstaben eingeben, können Sie eventuelle Tippfehler leichter finden, da dann alle falschgeschriebenen Befehle bei der Auflistung weiterhin in Kleinschreibung erscheinen. In den verwendeten Beispielen sind alle Befehle mit Großbuchstaben geschrieben.

BEFEHLE:

ABS

ABS (‹numerischer Ausdruck›)

```
PRINT ABS(-67.98)
 67.98
```

FUNKTION: Gibt den ABSoluten Wert des angegebenen Ausdrucks zurück, d.h. negative Zahlen werden in positive verwandelt.

Verwandte Befehle: SGN

AFTER

AFTER ‹Intervall› [,‹Zeitgebernummer›] GOSUB ‹Zeilennummer›

```
10 AFTER 250 GOSUB 60:CLS
20 PRINT "Rate einen Buchstaben in 5 Sekunden"
30 a$=INKEY$:IF Signal=1 THEN END
40 IF a$<>CHR$(INT(RND*26+97)) THEN 30
50 PRINT a$;" ist richtig. Du hast gewonnen"
55 SOUND 1,478:SOUND 1,358:END
60 PRINT "Zu spaet. Ich habe gewonnen"
70 SOUND 1,2000:Signal=1:RETURN
run
```

KOMMANDO: Ruft nach einem bestimmten Zeitabschnitt ein BASIC-Unterprogramm auf. Der ‹Intervall› Parameter gibt das Zeitintervall in Einheiten zu 0,02 Sekunden (Fünfzigstelsekunden) an.

Die ‹Zeitgebernummer› bestimmt, welcher der vier möglichen Zeitgeber (0, 1, 2 oder 3) verwendet werden soll. Zeitgeber 3 hat die höchste, Zeitgeber 0 die niedrigste Priorität.

Jeder Zeitgeber kann mit einem anderen Unterprogramm verbunden sein.

Weitere Informationen über Unterbrechungen finden Sie im 2.Teil von Kapitel 9.

Verwandte Befehle: EVERY, REMAIN, RETURN

AND

‹Aussage› **AND** ‹Aussage›

```
IF "Anna"<"Berta" AND "Hund">"Katze" THEN PRINT "richt
ig" ELSE PRINT "falsch"
richtig
IF "Berta"<"Anna" AND "Katze">"Hund' THEN PRINT "richt
ig" ELSE PRINT "falsch"
falsch
IF "Anna"<"Berta" AND "Katze">"Hund' THEN PRINT "richt
ig" ELSE PRINT "falsch"
falsch

....

PRINT 1 AND 1
 1
PRINT Ø AND Ø
 Ø
PRINT 1 AND Ø
 Ø
```

OPERATOR: Führt Bit für Bit Boolesche Vergleichsoperationen mit ganzen Zahlen durch. Das Ergebnis ist **Ø**, es sei denn, beide Aussagen entsprechen **1**.

Weitere Informationen über Logik finden Sie im 2.Teil von Kapitel 9.

Verwandte Befehle: **OR**, **NOT**, **XOR**

ASC

ASC (‹Textausdruck›)

```
PRINT ASC("x")
 12Ø
```

FUNKTION: Gibt den numerischen Wert des ersten Zeichens im ‹Textausdruck› wieder.

Verwandte Befehle: **CHR$**

ATN

ATN (‹numerischer Ausdruck›)

```
PRINT ATN(1)
 Ø.785398163
```

FUNKTION: Berechnet den ArcusTaNgens des ‹numerischen Ausdrucks›.

Mit den Befehlen DEG und RAD kann das Ergebnis der obigen Berechnung in Winkelgraden bzw. im Bogenmaß ausgedrückt werden.

Verwandte Befehle: COS, DEG, RAD, SIN, TAN

AUTO

AUTO [‹Zeilennummer› || , ‹Schrittweite›]

```
AUTO 1ØØ,5Ø
```

KOMMANDO: Numeriert die Zeilen AUTOmatisch. Der wahlweise anzugebende ‹Zeilennummer›-Parameter nennt die erste Zeilennummer, mit der begonnen wird, falls die Numerierung erst ab einem bestimmten Punkt im Programm gewünscht wird. Wird der Parameter nicht angegeben, beginnen die Zeilennummern mit 1Ø.

Die ‹Schrittweite› bestimmt die Differenz zur nächsten Zeilennummer. Wird sie nicht angegeben, so wird als Standardwert 1Ø angenommen.

Falls eine Zeile mit derselben Nummer schon vorhanden ist, erscheint der Inhalt dieser Zeile auf dem Bildschirm und kann bei Bedarf geändert werden. Mit dem Drücken der **[RETURN]**-Taste wird die gezeigte (und eventuell geänderte) Zeile gespeichert.

Ein Druck auf die **[ESC]**-Taste beendet die automatische Numerierung.

Verwandte Befehle: **keine**

BIN$

BIN$ (‹vorzeichenloser ganzzahliger Ausdruck›[,‹Feldbreite›])

```
PRINT BIN$(64,8)
01000000
```

FUNKTION: Gibt den Wert des ‹vorzeichenlosen ganzzahligen Ausdrucks› in BINärer Form wieder. Die ‹Feldbreite› (zwischen 0 und 16) gibt an, wieviele Binärstellen verwendet werden sollen. Ist die Anzahl der tatsächlich benötigten Stellen kleiner als vorgeschrieben, so werden dem Ergebnis Nullen vorangestellt. Ist sie größer als vorgeschrieben, so wird das Ergebnis NICHT gekürzt, sondern in sovielen Ziffern ausgegeben, wie benötigt werden.

Der Wert des ‹vorzeichenlosen ganzzahligen Ausdrucks›, der in Binärform umgewandelt werden soll, muß zwischen –32768 und +65535 liegen.

Verwandte Befehle: DEC$, HEX$, STR$

BORDER

BORDER ‹Farbe› [,‹Farbe›]

```
10 REM 729 Randfarben-Kombinationen
20 SPEED INK 5,5
30 FOR a=0 TO 26
40 FOR b=0 TO 26
50 BORDER a,b:CLS:LOCATE 14,13
60 PRINT "Rand";a;",";b
70 FOR t=1 TO 500
80 NEXT t,b,a
run
```

KOMMANDO: Die nicht beschreibbare Randfläche des Bildschirms (engl. border) wird in der angegebenen ‹Farbe› dargestellt. Wird eine zweite ‹Farbe› angegeben, so wechseln sich die beiden Farben mit der im SPEED INK-Befehl angegebenen Geschwindigkeit ab. ‹Farbe› darf Werte von 0 bis 26 haben.

Verwandte Befehle: SPEED INK

BREAK

(Siehe ON BREAK CONT, ON BREAK GOSUB, ON BREAK STOP)

CALL

CALL ‹Adressausdruck› [,‹Liste von:‹Parameter›]

```
CALL Ø
```

KOMMANDO: Springt in ein Unterprogramm außerhalb des BASIC-Bereiches, das an der angegebenen Speicheradresse beginnt. In unserem Beispiel wird der Computer durch diesen Befehl ganz zurückgesetzt.

Dieser Befehl sollte mit Vorsicht angewendet werden.

Verwandte Befehle: UNT

CAT

CAT

```
CAT
```

KOMMANDO: Zeigt das Inhaltsverzeichnis (engl. catalogue) einer Diskette in alphanumerischer Reihenfolge auf dem Bildschirm. Dabei wird außer der Länge der einzelnen Dateien (zum nächsthöheren KByte aufgerundet) der Umfang des freien Speicherplatzes auf der Diskette angegeben. Über dem Verzeichnis wird das Laufwerk (Drive) und die Benutzer-Nummer (User) angegeben.

Der Befehl hat keine Auswirkungen auf das augenblicklich im Speicher befindliche Programm.

Verwandte Befehle: LOAD, RUN, SAVE

CHAIN

CHAIN ‹Dateiname› [,‹Zeilennummer›]

```
CHAIN "Testprog.bas",35Ø
```

KOMMANDO: Lädt ein Programm von der Diskette in den Speicher, wobei das gerade im Speicher befindliche Programm gelöscht wird. Das neue Programm wird gleich gestartet, entweder von vorn, oder ab der angegebenen ‹Zeilennummer›.

Der Befehl kann verwendet werden, um geschützte Programme, (die mit ,p gesichert wurden) zu laden und laufen zu lassen.

Verwandte Befehle: CHAIN MERGE, LOAD, MERGE

CHAIN MERGE

CHAIN MERGE ‹Dateiname› [,‹Zeilennummer›]
[,DELETE ‹Zeilenbereich›]

```
CHAIN MERGE "Neuprogr.bas",750,DELETE 400-680
```

KOMMANDO: Lädt ein Programm von der Diskette in den Speicher und fügt es in das gegenwärtig im Speicher befindliche Programm ein. Das auf diese Weise neu entstandene Programm wird gleich gestartet, entweder von vorn, oder ab der angegebenen ‹Zeilennummer›. Mit DELETE ‹Zeilenbereich› wird der angegebene Bereich des Original-Programms vor dem Laden gelöscht, falls erforderlich.

Beachten Sie, daß Zeilen des alten Programms von Zeilen des einzufügenden Programms überschrieben werden, wenn sie dieselbe Zeilennummer aufweisen.

Geschützte Programme, die durch ',P' gesichert wurden, können NICHT mit diesem Befehl geladen werden.

Verwandte Befehle: CHAIN, DELETE, LOAD, MERGE

CHR$

CHR$ (‹ganzzahliger Ausdruck›)

```
10 FOR x=32 TO 255
20 PRINT x;CHR$(x)
30 NEXT
run
```

FUNKTION: Wandelt einen ‹ganzzahligen Ausdruck› zwischen 0 und 255 entsprechend der AMSTRAD-Zeichentabelle (siehe Kapitel 7, Teil 3) in eine Zeichenfolge (engl. CHaRacter String) um.

Die Zahlen 0 bis 31 stellen Kontrollzeichen dar. Im obigen Beispiel steht daher für x eine Zahl zwischen 32 und 255.

Verwandte Befehle: ASC

CINT

CINT (‹numerischer Ausdruck›)

```
10 n=1.9999
20 PRINT CINT(n)
run
 2
```

FUNKTION: Rundet den ‹numerischen Ausdruck› im Wertebereich von –32768 bis +32767 zu einer ganzen Zahl (engl. Convert to INTeger).

Verwandte Befehle: CREAL, FIX, INT, ROUND, UNT

CLEAR

CLEAR

```
CLEAR
```

KOMMANDO: Setzt alle Variablen auf Null, vergißt alle offenen Dateien, Arrays und Anwenderfunktionen. BASIC wird auf Bogenmaß-Ausgabe gesetzt.

Verwandte Befehle: **keine**

CLEAR INPUT

CLEAR INPUT

```
10 CLS
20 PRINT "Gib Buchstaben ein!"
30 FOR t=1 TO 3000
40 NEXT
50 CLEAR INPUT
run
```

KOMMANDO: Entledigt sich aller vorher eingetippten INPUT-Informationen, die noch im Tastatur-Puffer liegen.

Die Wirkung dieses Befehls sehen Sie am besten, wenn Sie das oben angegebene Programm laufen lassen und Buchstaben eingeben, wenn der Computer Sie dazu auffordert. Streichen Sie dann Zeile 50 und lassen Sie das Programm nochmals laufen.

Verwandte Befehle: INKEY, INKEY$, JOY

CLG

CLG [‹Farbstift›]

```
LOCATE 1,20
CLG 3
```

KOMMANDO: Löscht den Graphik-Bildschirm und füllt ihn mit der dem ‹Farbstift› (INK) zugeordneten Farbe. Ist kein ‹Farbstift› angegeben, wird der im letzten CLG-Befehl bestimmte ‹Farbstift› verwendet.

Verwandte Befehle: CLS, GRAPHICS PAPER, INK, ORIGIN

CLOSEIN

CLOSEIN

```
CLOSEIN
```

KOMMANDO: Schließt Eingabe einer Datei von der Diskette.

Verwandte Befehle: EOF, OPENIN

CLOSEOUT

CLOSEOUT

```
CLOSEOUT
```

KOMMANDO: Schließt die Ausgabe einer Datei auf die Diskette (siehe OPENOUT).

Verwandte Befehle: OPENOUT

CLS

CLS [#‹Stream-Ausdruck›]

```
10 PAPER#2,3
20 CLS#2
run
```

KOMMANDO: Löscht den angegebenen Bildschirmbereich und füllt ihn mit der Farbe des im PAPER-Befehl festgelegten Farbstifts. Ist kein Stream angegeben, wird der ganze Bildschirm gelöscht.

Verwandte Befehle: CLG, INK, PAPER, WINDOW

CONT

CONT

```
CONT
```

KOMMANDO: Setzt ein Programm fort (engl. CONTinue), das entweder durch zweimaliges Drücken der **[ESC]**-Taste oder nach einem STOP-Befehl unterbrochen war. Der Befehl ist nur wirksam, wenn das Programm vorher nicht geändert wird, und wenn es nicht geschützt ist.

Kommandos dürfen zwischen Abbruch und Fortsetzung eingegeben werden.

Verwandte Befehle: STOP

COPYCHR$

COPYCHR$ (#‹Stream-Ausdruck›)

```
10 CLS
20 PRINT "obere Ecke"
30 LOCATE 1,1
40 a$=COPYCHR$(#0)
50 LOCATE 1,20
60 PRINT a$
run
```

FUNKTION: Kopiert ein Zeichen im angegebenen Bildschirmbereich (Stream), der unbedingt angegeben werden muß. Im Beispielprogramm wird der Buchstabe von Position 1,1 (oben links auf dem Bildschirm) in Position 1,20 wiederholt.

Wird das Zeichen nicht erkannt, wird ein Leerstring wiedergegeben.

Verwandte Befehle: LOCATE

COS

COS (‹numerischer Ausdruck›)

```
DEG
PRINT COS(45)
 0.707106781
```

FUNKTION: Berechnet den COSinus des ‹numerischen Ausdrucks›.

Beachten Sie, daß das Ergebnis der Berechnung mit den Befehlen DEG und RAD im Winkel- bzw. Bogenmaß ausgedrückt werden kann.

Verwandte Befehle: ATN, DEG, RAD, SIN

CREAL

CREAL (‹numerischer Ausdruck›)

```
10 a=PI
20 PRINT CINT(a)
30 PRINT CREAL(a)
run
 3
 3.14159265
```

Gibt den Ausdruck als reelle Zahl, d.h. als Dezimalzahl zurück (engl. Convert to REAL).

Verwandte Befehle: CINT

CURSOR

CURSOR [‹Systemschalter›][,‹Benutzerschalter›]

```
10 CURSOR 1
20 PRINT "Frage?";
30 a$=INKEY$:IF a$="" THEN 30
40 PRINT a$
50 CURSOR 0
run
```

KOMMANDO: Stellt den System- bzw. den Benutzerschalter für den Cursor auf Ein oder Aus. Die ‹Systemschalter›- und ‹Benutzerschalter›-Parameter müssen entweder 0 (Aus) oder 1 (Ein) sein. Im obigen INKEY$-Befehl, wo der Cursor normalerweise nicht sichtbar ist, wurde er eingeschaltet, indem der ‹Systemschalter› auf 1 gestellt wurde (Zeile 10).

Der Cursor ist sichtbar, wenn sowohl der ‹Systemschalter› als auch der ‹Benutzerschalter› auf 1 stehen. Beim Befehl INPUT schaltet sich der Cursor automatisch an, bei INKEY$ ab.

Es wird empfohlen, den Cursor abzuschalten, wenn ein Text über den Bildschirm ausgegeben wird.

Einer der beiden Parameter, jedoch nicht beide zugleich, können ausgelassen werden. Wird ein Parameter nicht angegeben, so ändert sich die Stellung des betreffenden Schalters nicht.

Verwandte Befehle: LOCATE

DATA

DATA ‹Liste von:›Konstante›

```
10 FOR x=1 TO 4
20 READ Vorname$,Nachname$
30 PRINT Vorname$;" ";Nachname$
40 NEXT
50 DATA Hilde,Meyer,Karl,Schmidt
60 DATA Rita,Faber,Rudi,Seiler
run
```

KOMMANDO: Stellt dem Programm Datenwerte zur Verfügung. Mit dem READ-Befehl können Daten in eine Variable eingelesen werden. Danach rückt der Zeiger zum nächsten Punkt auf der Datenliste. Mit dem RESTORE-Befehl kann der Zeiger an eine bestimmte Stelle der Datenliste gebracht werden.

Weitere Informationen über Daten finden Sie im 2.Teil des 9. Kapitels.

Verwandte Befehle: READ, RESTORE

DEC$

DEC$ (‹numerischer Ausdruck›,‹Formatschablone›)

```
PRINT DEC$(10↑7,"ff##########,.##")
 £10,000,000.00
```

FUNKTION: Gibt den ‹numerischen Ausdruck› als dezimalen (engl. decimal) String wieder. Dabei wird mit der angegebenen ‹Formatschablone› das Format des Strings bestimmt.

Die Formatschablone darf NUR mit folgenden Zeichen angegeben werden:

```
+ - £ $ * # , . ↑
```

Wie diese Formatzeichen angewendet werden, ist unter dem Befehl PRINT USING ausgeführt.

Verwandte Befehle: BIN$, HEX$, PRINT USING, STR$

DEF FN

DEF FN‹Name› [(‹Formalparameter›)]=‹allgemeiner Ausdruck›

```
10 t=TIME/300
20 DEF FNUhr=INT(TIME/300-t)
30 EVERY 100 GOSUB 50
40 GOTO 40
50 PRINT "Programm wurde vor";
60 PRINT FNUhr;"Sekunden gestartet"
70 RETURN
run
```

KOMMANDO: DEFiniert eine FuNktion. Mit BASIC kann man innerhalb eines Programms sogenannte Funktionen definieren, die einen Wert zurückgeben. Die mit DEF FN-Befehlen erzeugten Funktionen gelten innerhalb des Programms und arbeiten auf dieselbe Art und Weise wie die in BASIC eingebauten Funktionen (z.B. COS).
(Beachten Sie, daß im obigen Beispiel der Wert der Funktion FNUhr laufend aktualisiert wird, selbst wenn das Programm durch einmaliges Drücken der **[ESC]**-Taste unterbrochen wird, oder wenn es nach zweimaligem Drücken der **[ESC]**-Taste mit dem Befehl CONT wiederaufgenommen wird.)

Verwandte Befehle: **keine**

DEFINT

DEFINT Liste von:‹Buchstaben›

```
10 DEFINT n
20 Zahl=123.456
30 PRINT Zahl
run
 123
```

KOMMANDO: Legt fest, daß alle Variablen, die mit den angegebenen Buchstaben beginnen, ganzzahlig (engl. INTeger) sind. Wenn kein Typenkennzeichen (!, %, oder $) angegeben ist, wird der Standard-(engl. DEFault) Variablentyp angenommen. Es können einzelne Buchstaben aufgezählt werden, z.B.:

```
DEFINT a,b,c
```

...oder es kann ein Buchstabenbereich angegeben werden, z.B.:

```
DEFINT a-z
```

Verwandte Befehle: DEFREAL, DEFSTR

DEFREAL

DEFREAL Liste von: Buchstaben

```
DEFREAL x,a-f
```

KOMMANDO: Legt fest, daß alle Variablen, die mit den angegebenen Buchstaben beginnen, reelle (engl. real) Zahlen sind. Wenn kein Typenkennzeichen (!, %, oder $) angegeben ist, wird der Standard-Variablentyp angenommen. Es können einzelne Buchstaben aufgezählt werden, z.B.:

```
DEFREAL a,b,c
```

...oder es kann ein Buchstabenbereich angegeben werden, z.B.:

```
DEFREAL a-z
```

Verwandte Befehle: DEFINT, DEFSTR

DEFSTR

DEFSTR Liste von: Buchstaben

```
10 DEFSTR n
20 name="AMSTRAD"
30 PRINT name
run
AMSTRAD
```

KOMMANDO: Legt fest, daß alle Variablen, die mit den angegebenen Buchstaben beginnen, STRings sind. Wenn kein Typenkennzeichen (!, %, oder $) angegeben ist, wird der Standard-Variablentyp angenommen. Es können einzelne Buchstaben aufgezählt werden, z.B.:

```
DEFSTR a,b,c
```

...oder es kann ein Buchstabenbereich angegeben werden, z.B.:

```
DEFSTR a-z
```

Verwandte Befehle: DEFINT, DEFREAL

DEG

DEG

```
DEG
```

KOMMANDO: Schaltet auf Winkelgradmaß (engl. DEGree) um. Die Werte für die Funktionen SIN, COS, TAN und ATN werden standardmäßig als Bogenmaß ausgegeben. Mit DEG wird BASIC auf die Ausgabe von Winkelgradmaß umgestellt, bis der Befehl durch RAD, NEW, CLEAR, LOAD oder RUN wieder aufgehoben wird.

Verwandte Befehle: ATN, COS, RAD, SIN, TAN

DELETE

DELETE ‹Zeilenbereich›

```
DELETE 100-200
```

KOMMANDO: Löscht (engl. delete) alle Programmzeilen innerhalb des ‹Zeilenbereichs›.

Die erste oder letzte Nummer des ‹Zeilenbereichs› kann weggelassen werden, wenn das gesamte Programm bis zu einer bestimmten Zeile...

```
DELETE -200
```

...bzw. ab einer bestimmten Zeile...

```
DELETE 50-
```

...gelöscht werden soll. Mit...

```
DELETE
```

...wird das gesamte Programm gelöscht.

Verwandte Befehle: CHAIN MERGE, RENUM

DERR

```
DERR
```

```
LOAD "xyz.abc"

XYZ .ABC not found
Ready
PRINT DERR
 146
```

FUNKTION: Gibt den letzten Fehlercode an, der vom Diskettenspeichersystem ausgegeben wurde. Mit DERR kann festgestellt werden, welcher Fehler im Diskettensystem (engl. Disc ERRor) aufgetreten ist. Im Kapitel 7 finden Sie eine Liste der Fehlermeldungen.

Verwandte Befehle: ERL, ERR, ERROR, ON ERROR GOTO, RESUME

DI

```
DI
```

```
10 CLS:TAG:EVERY 10 GOSUB 90
20 X1=RND*320:X2=RND*320
30 Y=200+RND*200:C$=CHR$(RND*255)
40 FOR X=320-X1 TO 320+X2 STEP 4
50 DI
60 MOVE 320,0,1:MOVE X-2,Y:MOVE X,Y
70 PRINT " ";C$;:FRAME
80 EI:NEXT:GOTO 20
90 MOVE 320,0:DRAW X+8,Y-16,0:RETURN
run
```

KOMMANDO: Verhindert alle Unterbrechungen (engl. Disables Interrupts) außer durch **[ESC]**, bis sie durch das EI-Kommando wieder ermöglicht werden.

Beachten Sie, daß ein Unterbrechungs-Unterprogramm automatisch Unterbrechungen von gleicher oder geringerer Priorität verhindert.

Das Kommando wird benutzt, um ein Programm ohne Unterbrechung Befehl für Befehl ablaufen zu lassen, falls z.B. zwei Programmteile ein und dasselbe Mittel benutzen wollen. Im obigen Beispiel verwenden das Haupt- und das Unterprogramm jeweils den Graphik-Bildschirm.

Weitere Informationen zum Thema Unterbrechungen finden Sie im 2.Teil des 9.Kapitels.

Verwandte Befehle: AFTER, EI, EVERY, REMAIN

DIM

DIM ‹Liste von:‹indizierten Variablen›

```
10 CLS
20 DIM Freund$(5),Tel$(5)
30 FOR n=1 TO 5
40 PRINT "Freund Nummer";n
50 INPUT "Name";Freund$(n)
60 INPUT "Telefon";Tel$(n)
70 PRINT
80 NEXT
90 FOR n=1 TO 5
100 PRINT n;Freund$(n),Tel$(n)
110 NEXT
run
```

KOMMANDO: Legt die 'DIMension' eines Arrays fest. DIM stellt einen Speicherbereich für Arrays (Tabellen) bereit und legt den höchsten Indexwert fest. In BASIC muß die Indexobergrenze angegeben werden, sonst wird der Standardwert 10 angenommen.

Ein Array wird durch die ‹indizierte Variable› gekennzeichnet, die es ermöglicht, daß ein und dieselbe Variable für eine Gruppe von Elementen verwendet werden kann. Die einzelnen Elemente werden zur Unterscheidung mit einer Indexzahl versehen. Ein Array kann mit einer Schleife bearbeitet werden, wobei in jedem Durchgang das Element mit der nächsten Indexzahl behandelt wird.

Der niedrigste Indexwert ist 0, d.h. das erste Element in einem Array hat den Index 0.

Arrays können mehrdimensional sein, wobei jedes Element eine feste Stellung innerhalb der Matrix des Arrays einnimmt. Ist die Dimension eines Arrays z.B. mit:

```
DIM Position$(20,20,20)
```

...angegeben, könnte ein Element des Arrays z.B. über:

```
Position$(4,5,6)
```

aufgerufen werden.

Verwandte Befehle: ERASE

DRAW

DRAW ‹x-Koordinate› , ‹y-Koordinate›[,[‹Farbstift][,‹Farbstiftmodus›]]

```
10 MODE 0:BORDER 0:PAPER 0:INK 0,0
20 x=RND*640:y=RND*400:z=RND*15
30 DRAW x,y,z
40 GOTO 20
run
```

KOMMANDO: Zeichnet (engl. draw) eine Linie auf dem Graphik-Bildschirm von der augenblicklichen Cursorposition zum angegebenen absoluten Koordinatenpunkt.Es darf zwischen Farbstift 0 und 15 gewählt werden.

Der wahlweise anzugebende ‹Farbstiftmodus› bestimmt, wie die Farbe des zu benutzenden Farbstifts mit den anderen Farben auf dem Bildschirm interagiert. Die vier Farbmodi sind:

0: Normal
1: XOR (exklusives ODER)
2: AND
3: OR

Verwandte Befehle: DRAWR, GRAPHICS PEN, MASK

DRAWR

DRAWR ‹x-Versatz› , ‹y-Versatz›[,‹Farbstift›][,‹Farbmodus›]

```
10 CLS:PRINT "komm die Treppe rauf"
20 MOVE 0,350:FOR n=1 TO 8
30 DRAWR 50,0
40 DRAWR 0,-50
50 NEXT:MOVE 348,0:FILL 3
60 GOTO 60
run
```

KOMMANDO: Zeichnet eine Linie auf dem Graphik-Bildschirm von der augenblicklichen Cursorposition zur angegebenen Relativen (x,y)-Position, also um x und y Zeichenpositionen versetzt. Es darf zwischen Farbstift 0 und 15 gewählt werden.

Der wahlweise anzugebende ‹Farbmodus› bestimmt, wie die Farbe des zu benutzenden Farbstifts mit den anderen Farben auf dem Bildschirm interagiert. Die vier Farbmodi sind:

0: Normal
1: XOR (exklusives ODER)
2: AND
3: OR

Verwandte Befehle: DRAW, GRAPHICS PEN, MASK

EDIT

EDIT ‹Zeilennummer›

```
EDIT 20
```

KOMMANDO: Ruft eine Zeile mit Cursor zur Bearbeitung auf den Bildschirm.

Verwandte Befehle: AUTO, LIST

EI

EI

```
EI
```

KOMMANDO: Macht das Kommando DI rückgängig, d.h. ermöglicht wieder Unterbrechungen (engl. Enables Interrupts).

Sind die Unterbrechungsfunktionen in einem Unterbrechungs-Unterprogramm ausgeschaltet worden, so treten sie automatisch wieder in Kraft, wenn BASIC am Ende des Unterprogramms auf das RETURN- Kommando trifft.

Weitere Informationen über Unterbrechungen finden Sie im 2.Teil des 9.Kapitels.

Verwandte Befehle: AFTER, DI, EVERY, REMAIN

ELSE

(Siehe IF)

END

```
END
```

```
    END
```

KOMMANDO: Beendet die Durchführung eines Programms und schaltet zum Direkt-Modus um. In einem Programm können beliebig viele **END**-Befehle vorkommen. Mit der letzten Befehlszeile **END**et ein Programm automatisch.

Verwandte Befehle: **STOP**

ENT

ENT‹Hüllkurvennummer›[**,**‹Hüllkurvenabschnitt›][**,**‹Hüllkurvenabschnitt›]
[**,**‹Hüllkurvenabschnitt›][**,**‹Hüllkurvenabschnitt›]
[**,**‹Hüllkurvenabschnitt›]

```
10 ENT 1,10,-50,10,10,50,10
20 SOUND 1,500,200,10,,1
run
```

KOMMANDO: Definiert die Tonhüllkurve (engl. Tone **EN**velope) für den **SOUND**-Befehl. Ihre ‹Hüllkurvennummer› (Bereich 1 bis 15) wird im **SOUND**-Befehl angegeben. Ist sie negativ (im Bereich -1 bis -15), wiederholt sich die Hüllkurve bis zum Ende des **SOUND**-Befehls.

Für jeden ‹Hüllkurvenabschnitt› können 2 oder 3 Parameter angegeben werden.

Werden drei Parameter verwendet, so heißen diese:

‹Schrittanzahl› **,** ‹Schrittweite› **,** ‹Schrittzeit›

PARAMETER 1: ‹Schrittanzahl›

Dieser Parameter bestimmt, wieviele verschiedene Tonhöhen der Ton in einem Hüllkurvenabschnitt durchlaufen soll. Wenn Sie z.B. in einem 10 Sekunden dauernden Ton 10 Tonhöhenschritte von jeweils einer Sekunde Dauer haben wollen, muß für den Parameter ‹Schrittanzahl› der Wert 10 angegeben werden.

Für die ‹Schrittanzahl› steht der Wertebereich 0 bis 239 zur Verfügung.

PARAMETER 2: ‹Schrittweite›

Dieser Parameter muß im Bereich –128 bis +127 liegen. Mit negativen Schritten nimmt die Tonhöhe zu, mit positiven Schritten nimmt sie ab. Die kürzeste Tonperiode ist 0. Im Kapitel 7 finden Sie eine Aufstellung der gesamten Tonperioden.

PARAMETER 3: ‹Schrittzeit›

Dieser Parameter gibt die Zeit zwischen den einzelnen Schritten in Hundertstelsekunden (0,01 Sekunden) an. Für diesen Parameter können Sie die Zahlen 0 bis 255 (wobei 0 als 256 behandelt wird) einsetzen. Das längste Zeitintervall zwischen zwei Schritten beträgt also 2,56 Sekunden.

Werden zwei Parameter angegeben, so heißen diese:

‹Tonperiode› , ‹Schrittzeit›

PARAMETER 1: ‹Tonperiode›

Dieser Parameter gibt den neuen absoluten Wert für die Tonperiode an (Siehe Parameter 2 des SOUND-Befehls).

PARAMETER 2: ‹Schrittzeit›

Dieser Parameter gibt die Zeit zwischen den einzelnen Schritten in Hundertstelsekunden (0,01 Sekunden) an. Für diesen Parameter können Sie die Zahlen 0 bis 255 (wobei 0 als 256 behandelt wird) einsetzen. Das längste Zeitintervall zwischen zwei Schritten beträgt also bis zu 2,56 Sekunden.

ALLGEMEINES

Beachten Sie, daß die Gesamtlänge aller ‹Schrittzeiten› nicht größer als der ‹Dauer›-Parameter im SOUND-Befehl sein sollte, sonst hört der Ton auf, bevor er alle Tonhöhenschritte durchlaufen hat. (In solch einem Fall wird der restliche Inhalt der Tonhüllkurve gelöscht.)

Wenn umgekehrt der ‹Dauer›-Parameter im SOUND-Befehl größer ist als die Gesamtlänge der ‹Schrittzeiten›, wird der Ton, nachdem er alle Schritte durchlaufen hat, auf der letzten Note verharren.

Im ENT-Befehl dürfen bis zu fünf verschiedene ‹Hüllkurvenabschnitte› verwendet werden, die jeweils aus den oben beschriebenen zwei oder drei Parametern bestehen können.

Der erste Schritt einer Tonhüllkurve wird unverzüglich ausgeführt.

Wird eine neue Tonhüllkurve bestimmt, wird ihr vorheriger Wert gelöscht.

Wird ein ‹Hüllkurvenabschnitt› ohne ‹Hüllkurvennummer› angegeben, so werden die vorherigen Werte gelöscht.

Weitere Informationen über Tonerzeugung finden Sie im 2.Teil des 9. Kapitels.

Verwandte Befehle: ENV, SOUND

ENV

ENV‹Hüllkurvennummer›[,‹Hüllkurvenabschnitt›][,‹Hüllkurvenabschnitt›]
[,‹Hüllkurvenabschnitt›][,‹Hüllkurvenabschnitt›]
[,‹Hüllkurvenabschnitt›]

```
10 ENV 1,15,-1,10,15,1,10
20 SOUND 1,200,300,15,1
run
```

KOMMANDO: Definiert die Lautstärkenhüllkurve (engl. Volume ENvelope) für den SOUND-Befehl. Ihre ‹Hüllkurvennummer› (im Bereich 1 bis 15) wird im SOUND-Befehl angegeben.

Jeder Hüllkurvenabschnitt kann aus zwei oder drei Parametern bestehen.

Werden drei Parameter angegeben, so heißen diese:

‹Schrittanzahl›, ‹Schrittweite›, ‹Schrittzeit›

PARAMETER 1: ‹Schrittanzahl›

Dieser Parameter bestimmt, wie viele verschiedene Lautstärkeveränderungen der Ton in einem Hüllkurvenabschnitt durchlaufen soll. Wenn Sie z.B. in einem 10 Sekunden dauernden Tonabschnitt 10 Lautstärkeveränderungen von jeweils einer Sekunde haben wollen, müssen Sie für den Parameter ‹Schrittanzahl› den Wert 10 angeben.

Für die ‹Schrittanzahl› steht der Wertebereich 0 bis 127 zur Verfügung.

PARAMETER 2: ‹Schrittweite›

Die Differenz zwischen den jeweiligen Lautstärkeschritten kann zwischen 0 und 15 liegen. Die 16 verschiedenen Lautstärkegrade sind dieselben, die Sie im SOUND-Befehl hören. Der ‹Schrittweite› Parameter kann hingegen zwischen –128 und +127 liegen, wobei die Lautstärke nach jeweils 15 Schritten zu 0 zurückkehrt.

PARAMETER 3: ‹Schrittzeit›

Dieser Parameter bestimmt die Zeit zwischen den einzelnen Lautstärkeschritten in Hundertstelsekunden (0,01 Sekunden). Für diesen Parameter könne Sie die Zahlen 0 bis 255 (wobei 0 als 256 behandelt wird) einsetzen. Die längste Pause zwischen zwei Schritten beträgt also 2,56 Sekunden.

Werden zwei Parameter angegeben, so heißen diese:

‹Registerwert›, ‹Hüllkurvenperiode›

PARAMETER 1: ‹Registerwert›

Dieser Parameter gibt den Wert an, der an das Lautstärkeregister des Ton-Chips übermittelt wird.

PARAMETER 2: ‹Hüllkurvenperiode›

Dieser Parameter gibt den Wert an, der an das Hüllkurvenregister des Ton-Chips übermittelt wird.

Wer die Hardware-Register direkt setzt, sollte sich in der Hardware auskennen. Sind nicht genügend Kenntnisse über Hardware vorhanden, so wird empfohlen, eine Software-Hüllkurve mit einem passenden ‹Schrittzeit›-Parameter zu benutzen.

ALLGEMEINES

Beachten Sie, daß die Gesamtlänge aller ‹Schrittzeiten› nicht größer als der ‹Dauer› Parameter im SOUND-Befehl sein sollte, sonst hört der Ton auf, ehe alle Lautstärkeveränderungen durchlaufen sind. (In solch einem Fall wird der restliche Inhalt der Lautstärken-Hüllkurve gelöscht.)

Wenn umgekehrt der ‹Dauer›-Parameter im SOUND-Befehl größer ist als die Gesamtlänge der Schrittzeiten, wird der Ton, nachdem er alle Lautstärkeschritte durchlaufen hat, auf dem letzten Lautstärkegrad verharren.

Im ENV-Befehl dürfen bis zu fünf verschiedene Hüllkurvenabschnitte verwendet werden, die jeweils aus den oben beschriebenen zwei oder drei Parametern bestehen können.

Der erste Schritt einer Lautstärken-Hüllkurve wird unverzüglich ausgeführt.

Wird eine neue Lautstärken-Hüllkurve bestimmt, wird ihr vorheriger Wert gelöscht.

Wird eine Lautstärken-Hüllkurvennummer ohne Hüllkurvenabschnitt angegeben, so werden die vorherigen Werte gelöscht.

Weitere Informationen über Tonerzeugung finden Sie im 2.Teil des 9. Kapitels.

Verwandte Befehle: ENV, SOUND

EOF

EOF

```
10 OPENIN "ex1.bas"
20 WHILE NOT EOF
30 LINE INPUT #9,a$
40 PRINT a$
50 WEND
60 CLOSEIN
run
```

FUNKTION: Prüft, ob eine Eingabe von Diskette das Dateiende (engl. End Of File) erreicht hat. Gibt -1 (richtig) zurück, wenn keine Datei geöffnet ist, oder Datei zu Ende ist, ansonsten erscheint 0 (falsch).

Verwandte Befehle: OPENIN, CLOSEIN

ERASE

ERASE ‹Liste von:‹Variablenname›

```
DIM a(100),b$(100)
ERASE a,b$
```

KOMMANDO: Löscht (engl. erase) den Inhalt eines Arrays, das nicht länger benötigt wird. Der Speicherplatz kann nun anderweitig verwendet werden.

Verwandte Befehle: DIM

ERL

ERL

```
10 ON ERROR GOTO 30
20 GOTO 1000
30 PRINT "Fehler in Zeile";ERL
40 END
run
```

FUNKTION: Gibt die Zeilennummer (engl. Line number) an, in der der letzte Fehler (engl. ERror) aufgetreten ist. Im obigen Beispiel gibt ERL an, daß der Fehler in Zeile 20 liegt.

Verwandte Befehle: DERR, ERR, ERROR, ON ERROR GOTO, RESUME

ERR

ERR

```
GOTO 500
Line does not exist
Ready
PRINT ERR
 8
```

FUNKTION: Gibt die Nummer an, unter der die zuletzt aufgetretene Fehlermeldung in der Fehlerliste (siehe Kap.7) zu finden ist. Die Fehlermeldung 'Line does not exist' aus unserem Beispiel ist unter Punkt 8 in der Fehlerliste aufgeführt.

Verwandte Befehle: DERR, ERL, ERROR, ON ERROR GOTO, RESUME

ERROR

ERROR ‹ganzzahliger Ausdruck›

```
10 IF INKEY$="" THEN 10 ELSE ERROR 17
run
```

KOMMANDO: Ruft den im ‹ganzzahligen Ausdruck› angegebenen Fehler auf. Im 7. Kapitel finden Sie eine Liste der 32 Fehlermeldungen. BASIC wird den Fehler so behandeln als hätte es ihn selbst entdeckt, und wird zu einer Fehlerbehandlungsroutine springen. Außerdem werden durch den Befehl ERROR die zugehörigen Werte für ERR und ERL angegeben.

Wenn sie ERROR mit einem ‹ganzzahligen Ausdruck› zwischen 33 und 255 versehen, können Sie ihre eigenen Fehlermeldungen kreieren, wie das folgende Beispiel erläutert:

```
10 ON ERROR GOTO 100
20 LINE INPUT "Gib ein Zeichen ein ";a$
30 IF LEN(a$)<>1 THEN ERROR 100
40 GOTO 20
100 IF ERR=100 THEN 110 ELSE 120
110 PRINT CHR$(7)
120 PRINT "Ich sagte, EIN Zeichen!"
130 RESUME 20
run
```

Verwandte Befehle: ERL, ERR, ON ERROR GOTO, RESUME

EVERY

EVERY ‹Zeitintervall›[,‹Zeitgeber›] GOSUB ‹Zeilennummer›

```
10 EVERY 50,1 GOSUB 30
20 GOTO 20
30 SOUND 1,20
40 RETURN
run
```

KOMMANDO: Ruft in regelmäßigen Zeitabständen ein BASIC-Unterprogramm auf. Das ‹Zeitintervall› wird in Einheiten zu Fünfzigstelsekunden (0,02 Sekunden) angegeben.

Der ‹Zeitgeber› bestimmt, welcher der vier Zeitgeber (Bereich 0 bis 3) verwendet wird. Zeitgeber 3 hat die höchste, Zeitgeber 0 (Standardzeitgeber) die niedrigste Priorität.

Mit jedem Zeitgeber kann ein Unterprogramm verbunden werden.

Weitere Informationen über Unterbrechungen finden Sie im 2.Teil des 9. Kapitels.

Verwandte Befehle: AFTER, REMAIN

EXP

EXP (‹numerischer Ausdruck›)

```
PRINT EXP(6.876)
 968.743625
```

FUNKTION: Gibt den Wert von 'e',potenziert mit dem ‹numerischen Ausdruck›, zurück. 'e' ist der natürliche Logarithmus von 1, also angenähert 2,7182818.

Verwandte Befehle: LOG

FILL

FILL ‹Farbstift›

```
10 MODE 0
20 FOR n=1 TO 500
30 PRINT "0";
40 NEXT
50 Farbstift=2+RND*13
60 FILL Farbstift
70 GOTO 50
run
```

KOMMANDO: Füllt (engl. fill) einen beliebigen Bereich auf dem Graphik-Bildschirm mit dem angegebenen Farbstift (Bereich 0 bis 15) aus. Die begrenzenden Linien dieses Bereichs werden entweder mit dem im GRAPHICS PEN-Befehl festgelegten Farbstift oder mit dem Farbstift gezogen, der den Bereich ausfüllen soll.

Der Farbstift beginnt seine Arbeit an der Graphik-Cursor-Position. Liegt der Cursor am Rand des auszufüllenden Bereichs, passiert gar nichts.

Verwandte Befehle: GRAPHICS PEN

FIX

FIX (‹numerischer Ausdruck›)

```
PRINT FIX(9.99999)
 9
```

FUNKTION: Rundet den ‹numerischen Ausdruck› zu einer ganzen Zahl ab, d.h. die Zahlen hinter dem Dezimalpunkt werden abgeschnitten.

Verwandte Befehle: CINT, INT, ROUND

FN

(Siehe DEF FN)

FOR

FOR ‹einfache Variable›=‹Start› TO ‹Ende›[STEP‹Schrittweite›]

```
10 FOR n=2 TO 8 STEP 2
20 PRINT n;
30 NEXT n
run
```

KOMMANDO: Wiederholt den Programmteil zwischen den FOR und NEXT-Befehlen so oft, bis die Werte zwischen ‹Start› und ‹Ende› durchlaufen sind. Ist keine ‹Schrittweite› angegeben, so wird der Wert 1 angenommen.

Für die ‹Schrittweite› darf auch ein negativer ‹numerischer Ausdruck› angegeben werden. In diesem Fall muß der ‹Start›- Parameter größer sein als der ‹Ende›-Parameter, sonst schreitet das Programm nicht zum jeweils nächsten Wert.

FOR...NEXT-Schleifen können verschachtelt werden, d.h. in eine Schleife kann eine andere eingebaut sein, in die eine dritte Schleife eingebaut ist, usw.

Im NEXT-Kommando braucht der Variablenname nicht angegeben zu werden, da BASIC automatisch herausfindet, auf welches FOR sich eine anonymes NEXT bezieht.

Verwandte Befehle: NEXT, STEP, TO

FRAME

FRAME

```
10 MODE 0
20 PRINT "FRAME off"
30 TAG
40 MOVE 0,200
50 FOR x=0 TO 500 STEP 4
60 IF f=1 THEN FRAME
70 MOVE x,200
80 PRINT " ";CHR$(143);
90 NEXT
100 IF f=1 THEN RUN
110 CLS
120 TAGOFF
130 PRINT "FRAME on"
140 f=1
150 GOTO 30
run
```

KOMMANDO: Synchronisiert den Schreibvorgang auf dem Bildschirm mit dem Strahlrücklauf (engl. frame flyback). Dadurch wird eine weichere Bewegung von Zeichen oder Graphik auf dem Bildschirm bewirkt, ohne Flackern oder Flimmern.

Verwandte Befehle: TAG, TAGOFF

FRE

FRE (‹numerischer Ausdruck›)
FRE (‹String›)

```
PRINT FRE(0)
PRINT FRE("")
```

FUNKTION: Stellt den Umfang des von BASIC nicht in Anspruch genommenen Speicherplatzes fest. Mit FRE("") werden auch kleinste FREie Speicherplätze aufgespürt.

Beachten Sie, daß BASIC nur die ersten 64K des Speichers beansprucht.

Verwandte Befehle: HIMEM, MEMORY

GOSUB

GOSUB ‹Zeilennummer›

```
GOSUB 210
```

KOMMANDO: Springt zu einem BASIC-Unterprogramm (engl. SUB-routine) in der angegebenen ‹Zeilennummer›. Nach dem Kommando RETURN, mit dem jedes Unterprogramm endet, fährt das Programm mit der Anweisung fort, die dem Aufruf des Unterprogramms folgt.

Verwandte Befehle: RETURN

GOTO

GOTO ‹Zeilennummer›

```
GOTO 90
```

KOMMANDO: Springt zur angegebenen ‹Zeilennummer›.

Verwandte Befehle: **keine**

GRAPHICS PAPER

GRAPHICS PAPER ‹Farbstift›

```
10 MODE 0
20 MASK 15
30 GRAPHICS PAPER 3
40 DRAW 640,0
run
```

KOMMANDO: Legt fest, auf welchem Farbhintergrund (‹Farbstift›-Skala 0 bis 15) die Zeichnungen auf dem Graphik-Bildschirm erscheinen sollen. Beim Zeichnen durchgehender Linien ist die Hintergrundfarbe unsichtbar. Im obigen Beispiel wird mittels des MASK-Befehls eine durchbrochene Linie gezeichnet, so daß die Hintergrundfarbe sichtbar wird.

Die GRAPHICS PAPER-Farbe bildet den Hintergrund für Zeichen, die unter Anwendung des TAG-Befehls auf den Graphik-Bildschirm geschrieben werden. Sie stellt außerdem die Standardfarbe dar, wenn der Graphik-Bildschirmbereich mit dem Befehl CLG gelöscht wird.

Verwandte Befehle: CLG, GRAPHICS PEN, INK, MASK, TAG, TAGOFF

GRAPHICS PEN

GRAPHICS PEN [‹Farbstift›][,‹Hintergrundmodus›]

```
10 MODE 0
20 GRAPHICS PEN 15
30 MOVE 200,0
40 DRAW 200,400
50 MOVE 639,0
60 FILL 15
run
```

KOMMANDO: Bestimmt den ‹Farbstift› (Bereich 0 bis 15) zum Zeichnen von Linien und Punkten. Der ‹Hintergrundmodus› ist entweder

0: Nicht transparenter Hintergrund

oder

1: Transparenter Hintergrund

(Beim Transparent-Modus erscheint der Hintergrund der unter Anwendung des TAG-Befehls auf den Bildschirm geschriebenen Zeichen sowie die Zwischenräume in gestrichelten Linien in der GRAPHICS PAPER-Farbe.)

Einer der beiden Parameter, jedoch nicht beide, können ausgelassen werden. Wird ein Parameter ausgelassen, ändert sich die jeweilige Einstellung nicht.

Verwandte Befehle: GRAPHICS PAPER, INK, MASK, TAG, TAGOFF

HEX$

HEX$ (‹vorzeichenloser ganzzahliger Ausdruck›[,‹Feldbreite›])

```
PRINT HEX$(255,4)
00FF
```

FUNKTION: Gibt den Wert des ‹vorzeichenlosen ganzzahligen Ausdrucks› in HEXadezimaler Form wieder. Die ‹Feldbreite› gibt an, wieviele Hexadezimalziffern (zwischen 0 und 16) verwendet werden sollen. Ist die Anzahl der tatsächlich benötigten Ziffern kleiner als vorgeschrieben, so werden der Hexadezimalzahl Nullen vorangestellt. Ist sie größer als vorgeschrieben, so wird die Hexadezimalzahl NICHT gekürzt, sondern in sovielen Ziffern ausgegeben, wie benötigt werden.

Der Wert des ‹vorzeichenlosen ganzzahligen Ausdrucks›, der in Hexadezimalform umgewandelt werden soll, muß zwischen –32768 und +65535 liegen.

Verwandte Befehle: BIN$, DEC$, STR$, UNT

HIMEM

HIMEM

```
PRINT HIMEM
42619
```

FUNKTION: Gibt die höchste (engl. HIghest) von BASIC belegte Speicheradresse an, (die mit dem MEMORY-Befehl neu festgelegt werden kann).

Beachten Sie, daß BASIC nur die ersten 64K im Speicher beansprucht.

Verwandte Befehle: FRE, MEMORY, SYMBOL, SYMBOL AFTER

IF

IF ‹logischer Ausdruck› THEN ‹Option›[ELSE Option›]

```
10 MODE 1
20 x=CINT(RND*100)
30 PRINT "Rate meine Zahl (0 bis 100)"
40 INPUT n
50 IF n<x THEN PRINT n;"ist zu niedrig"
60 IF n>x THEN PRINT n;"ist zu hoch"
70 IF n=x THEN 80 ELSE c=c+1:GOTO 40
80 PRINT "Gut gemacht, du hast richtig geraten!";
90 PRINT c+1;"Versuche!"
run
```

KOMMANDO: Stellt fest, ob (engl. if) der ‹logische Ausdruck› zutrifft. Wenn ja, dann (engl. then) wird die erste Option ausgeführt. Wenn nicht, wird eine Option aus dem ELSE-Teil ausgeführt. Ist keine ELSE-Option angegeben, fährt BASIC mit der nächsten Programmzeile fort.

Es können beliebig viele IF-Anweisungen ineinander verschachtelt werden. Da der IF-Befehl erst mit der Zeile endet, dürfen Anweisungen, die nichts mit dem IF-Kommando zu tun haben, NICHT in derselben Zeile stehen.

Um anzugeben, daß je nach dem Ergebnis des ‹logischen Ausdruck› zu einer bestimmten Zeile gesprungen werden soll, bestehen folgende Möglichkeiten:

```
IF a=1 THEN 100

IF a=1 GOTO 100

IF a=1 THEN GOTO 100
```

Verwandte Befehle: ELSE, GOTO, THEN

INK

INK ‹Farbstift›,‹Farbe›[,‹Farbe›]

```
10 MODE 1:PAPER 0:PEN 1
20 FOR p=0 TO 1
30 FOR i=0 TO 26
40 INK p,i
50 LOCATE 16,12:PRINT "Farbstift";p;",";i
60 FOR t=1 TO 400:NEXT t,i,p
70 INK 0,1:INK 1,24:CLS
run
```

KOMMANDO: Ordnet einem Farbstift eine oder zwei Farben zu. Als ‹Farbstift›-Parameter wird ein ganzzahliger Ausdruck zwischen 0 und 15 angegeben, der auch in den zugehörigen PEN- und PAPER-Befehlen eingesetzt wird. Der erste ‹Farbe›-Parameter sollte einen ganzzahligen Ausdruck zwischen 0 und 26 für den Farbwert enthalten. Wird wahlweise eine zweite Farbe bestimmt, so wechseln sich die beiden Farben mit der im SPEED INK- Befehl angegebenen Geschwindigkeit ab.

Verwandte Befehle: GRAPHICS PAPER, GRAPHICS PEN, PAPER, PEN, SPEED INK

INKEY

INKEY (‹ganzzahliger Ausdruck›)

```
10 IF INKEY(55)<>32 THEN 10
20 PRINT "Du hast [SHIFT] und V gedrueckt"
30 CLEAR INPUT
run
```

FUNKTION: Fragt (engl. INterrogate) die Tastatur (engl. KEYboard), welche Taste gedrückt wird. Die Tastatur wird jede Fünfzigstelsekunde (alle 0,02 Sekunden) daraufhin untersucht.

Mit dieser Funktion läßt sich leicht feststellen, ob eine bestimmte Taste gedrückt wird oder nicht, denn unabhängig davon, in welcher Stellung sich die **[SHIFT]**-und **[CONTROL]**-Tasten befinden, erscheint der Wert -1, wenn keine andere Taste gedrückt wird.

Im obigen Beispiel stellt die Funktion fest, wenn **[SHIFT]** und **V** (Taste Nr.55) gleichzeitig gedrückt werden. Sobald das der Fall ist, wird das Programm beendet. Eine Liste der Tasten mit den zugehörigen Nummern finden Sie oben rechts neben dem Tastenfeld und im Kapitel 7.

Am Zahlenwert, den der Computer ausgibt, läßt sich ablesen, ob eine bestimmte Taste gedrückt wird, und in welcher Stellung sich die **[SHIFT]**- und **[CONTROL]**-Tasten dabei befinden:

Ausgegebener Wert	**[SHIFT]** gedrückt	**[CONTROL]** gedrückt	Taste gedrückt
-1	ja/nein	ja/nein	nein
0	nein	nein	ja
32	ja	nein	ja
128	nein	ja	ja
160	ja	ja	ja

Verwandte Befehle: `CLEAR INPUT`, `INKEY$`, `JOY`

INKEY$

`INKEY$`

```
10 CLS
20 PRINT "Waehle Ja oder Nein (J/N)"
30 a$=INKEY$
40 IF a$="" THEN 30
50 IF a$="j" OR a$="J" THEN 80
60 IF a$="n" OR a$="N" THEN 90
70 GOTO 30
80 PRINT "Du hast JA gewaehlt":END
90 PRINT "Du hast NEIN gewaehlt"
run
```

FUNKTION: Fragt die Tastatur ab, ob eine Taste gedrückt wurde, die für einen String steht. Wird keine Taste gedrückt, gibt der Computer einen Leerstring aus. Im obigen Beispiel wird das Programm in den Zeilen `40` und `70` gezwungen, in einer Schleife zu Zeile `30` zurückzukehren.

Verwandte Befehle: `CLEAR INPUT`

INP

INP (‹Schnittstellennummer›)

```
PRINT INP(&FF77)
 255
```

FUNKTION: Gibt den Eingabe- (engl. INPut) Wert der mit der ‹Schnittstellennummer› bezeichneten I/O Adresse wieder.

Verwandte Befehle: OUT, WAIT

INPUT

INPUT [#‹Stream›,][;][‹Ausgabetext›‹Trennzeichen›]‹Liste von:‹Variablen›

```
10 MODE 1
20 INPUT "Gib zwei durch Komma getrennte Zahlen an, die
 miteinander multipliziert werden sollen";a,b
30 PRINT a;"mal";b;"ist";a*b
40 GOTO 20
run
```

FUNKTION: Nimmt Daten vom angegebenen ‹#Stream› auf. (Wenn nichts angegeben ist, Daten von Stream #0.)

Das erste, wahlweise anzugebende Semikolon (;) verhindert den Zeilenvorschub nach der Eingabe.

Das Trennzeichen muß entweder ein Komma oder ein Semikolon sein. Wird ein Semikolon angegeben, erscheint hinter dem Ausgabetext ein Fragezeichen.

Wird ein falscher Datentyp eingegeben, z.B. der Buchstabe O statt der Zahl 0 (Null), wenn eine numerische Variable gefragt ist, so reagiert BASIC mit...

```
?Redo from start
```

..und wiederholt den Ausgabetext, den Sie ins Programm geschrieben haben.

Alle Eingaben müssen mit **[RETURN]** abgeschlossen werden.

Verwandte Befehle: LINE INPUT

INSTR

INSTR ([‹Startposition›,]‹durchsuchter String›,‹gesuchter String›)

```
10 CLS:FOR n=1 TO 26
20 Alphabet$=Alphabet$+CHR$(n+64)
30 NEXT
40 INPUT "Gib einen Buchstaben an";a$
50 b$=UPPER$(a$)
60 PRINT b$;" ist Nummer";
70 PRINT INSTR(Alphabet$,b$);
80 PRINT "im Alphabet.":PRINT
90 GOTO 40
run
```

FUNKTION: Forscht im ‹durchsuchten String› nach dem ‹gesuchten String› und gibt an, an welcher Stelle der gesuchte String zum ersten Mal im durchsuchten String auftritt. Bleibt die Suche erfolglos, wird der Wert 0 ausgegeben.

Im ‹Startpositions›-Parameter kann eine ganze Zahl zwischen 1 und 255 angegeben werden, um zu bestimmen, wo die Suche beginnen soll.

Verwandte Befehle: **keine**

INT

INT (‹numerischer Ausdruck›)

```
PRINT INT(-1.995)
-2
```

FUNKTION: Rundet die angegebene Dezimalzahl zur nächstkleineren ganzen Zahl (engl. INTeger) ab. Für positive Zahlen ergeben sich dieselben Werte wie bei der FIX-Funktion, für negative Zahlen liegen die ausgegebenen INT-Werte hingegen um 1 unter denen der FIX-Werte.

Verwandte Befehle: CINT, FIX, ROUND

JOY

JOY (‹ganzzahliger Ausdruck›)

```
10 PRINT "Um das Programm zu beenden - ";
20 PRINT "benutze den Joystick"
30 IF JOY(0)<>0 THEN END
40 GOTO 10
run
```

FUNKTION: Liest einen Bit-signifikanten Wert, der den Bewegungen des im ‹ganzzahligen Ausdruck› bezeichneten Joysticks (entweder 0 oder 1) entspricht.

Bit	Bedeutung	Dezimalwert
0:	aufwärts	1
1:	abwärts	2
2:	links	4
3:	rechts	8
4:	Feuer 2	16
5:	Feuer 1	32

Wenn also der Haupt-'Feuer'-Knopf (Feuer 2) auf dem ersten Joystick gedrückt wird, während der Joystick-Griff nach links bewegt wird, gibt die Funktion JOY(0) den Dezimalwert 20 wieder, der sich aus 16 (Feuer 2) und 4 (links) zusammensetzt.

Weitere Informationen über Joysticks finden Sie im Kapitel 7.

Verwandte Befehle: CLEAR INPUT, INKEY

KEY

KEY ‹Erweiterungszeichennummer›,‹String›

```
KEY 11,"border 13:paper 0:pen 1:ink
0,13:ink 1,0:mode 2:list"+CHR$(13)
```

...jetzt **[ENTER]** drücken.

KOMMANDO: Ordnet den ‹String› der mit der ‹Erweiterungszeichennummer› bezeichneten Taste zu. Es gibt 32 Erweiterungszeichen (0 bis 31), die die Tastenwerte 128 bis 159 belegen. Tasten 128 (0 auf der Zahlentastatur) bis 140 (**[CONTROL] [ENTER]**) sind standardmäßig auf die Zahlen 0 bis 9 , sowie auf den Punkt, auf **[RETURN]** und RUN"**[RETURN]** - (für die Bedienung des Kassettenrecorders) festgelegt, doch können ihnen bei Bedarf auch andere ‹Strings› zugeordnet werden. Die Erweiterungszeichen 13 bis 31 (Tastenwerte 141 bis 159) sind standardmäßig Leerstrings, dürfen jedoch unter Verwendung des KEY DEF-Befehls erweitert und Tasten zugeordnet werden.

Als ‹Erweiterungszeichennummer› zur Bezeichnung der Taste kann eine Zahl zwischen 0 und 31 oder zwischen 128 und 159 gewählt werden.(Vergleiche Abbildung der Tastatur im Kapitel 7).

Insgesamt können den Tasten 120 Zeichen zugeordnet werden. Versucht man, diese Grenze zu überschreiten, antwortet der Computer mit der Fehlermeldung 5: 'Improper Argument'.

Verwandte Befehle: KEY DEF

KEY DEF

KEY DEF‹Tastennummer›,‹Dauerfunktion›[,‹Normalzeichen›
[,‹Shift›[,‹Control›]]]

```
KEY 159,"Dies ist die TAB-Taste"
KEY DEF 68,1,159
```

...jetzt die **[TAB]**-Taste drücken.

KOMMANDO: DEFiniert die Tasten-Werte, die durch Drücken der Taste mit der angegebenen ‹Tastennummer› ausgegeben werden. (Die Tastennummern - 0 bis 79 - finden Sie auf Ihrem Computer rechts oben neben dem Tastenfeld oder im Kapitel 7). Die ‹Normal›-, ‹Shift›- und ‹Control›-Parameter enthalten die Werte, die ausgegeben werden sollen, wenn die Taste allein, mit der (Shift)-Taste oder mit der (Control)-Taste gedrückt wird. Diese drei Parameter können wahlweise angegeben werden.

Mit dem ‹Dauerfunktions›-Parameter (1 oder 0) können Sie die Dauerfunktion einer Taste an- oder abstellen. Die Geschwindigkeit, mit der ein Zeichen in Dauerfunktion wiederholt wird, läßt sich mit dem SPEED KEY-Kommando variieren.

Im obigen Beispiel wird Taste 159 (sie entspricht dem Erweiterungszeichen 31) erst einem Erweiterungsstring zugeordnet. Der KEY DEF-Befehl bewirkt, daß Taste 68 (die **[TAB]**-Taste) ihre Dauerfunktion ausübt und den ‹normalen› Wert 159 ausgibt, wenn sie allein gedrückt wird.

Die ursprüngliche Funktion der Taste wird folgendermaßen wiederhergestellt:

```
KEY DEF 68,0,9
```

9 ist der normale ASCII-Wert für **[TAB]**.

Verwandte Befehle: KEY, SPEED KEY

LEFT$

LEFT$ (‹String›,‹Länge›)

```
10 CLS
20 a$="AMSTRAD"
30 FOR n=1 TO 9
40 PRINT LEFT$(a$,n)
50 NEXT
run
```

FUNKTION: Gibt den ‹String› in der angegebenen ‹Länge› (Zeichenanzahl von 0 bis 255) wieder. Dabei wird er von links (engl. left) beginnend in jeder Zeile um ein Zeichen verlängert. Ist er kürzer als die Länge angibt, wird der ganze String entsprechend oft wiederholt.

Verwandte Befehle: MID$, RIGHT$

LEN

LEN (‹String›)

```
10 LINE INPUT "Gib ein Wort ein";a$
20 PRINT "Das Wort ist";
30 PRINT LEN(a$);"Buchstaben lang."
run
```

FUNKTION: Gibt an, wieviele Zeichen (einschließlich Leerstellen) der ‹String› umfaßt. (engl. LENgth)

Verwandte Befehle: **keine**

LET

LET ‹Variable›=‹Ausdruck›

```
LET x=100
```

KOMMANDO: Weist einer Variablen einen Wert zu. Veraltete Form aus früheren BASIC-Sprachen, Variablen einen Wert zuzuweisen. Das Kommando ist im AMSTRAD-BASIC überflüssig, wurde aber aus Gründen der Kompatibilität mit Programmen aus älteren BASIC-Handbüchern aufgenommen. Das obige Beispiel kann einfach in folgender Form geschrieben werden:

```
x=100
```

Verwandte Befehle: **keine**

LINE INPUT

LINE INPUT [#‹Stream›,][;][‹Ausgabetext›‹Trennzeichen›]‹Stringvariable›

```
10 LINE INPUT "Schreibe eine Zeile Text";a$
20 CLS
30 PRINT "Die Variable a$ entspricht jetzt:-"
40 PRINT a$
run
```

KOMMANDO: Liest eine Zeile (engl. line) Text vom angegebenen ‹#Stream›. (Stream #0 wird angenommen, wenn der Parameter nicht angegeben wird.) Das erste, wahlweise anzugebende Semikolon (;) verhindert den Zeilenvorschub nach der Eingabe.

Das Trennzeichen kann entweder ein Komma oder ein Semikolon sein. Ein Semikolon läßt hinter dem Ausgabetext ein Fragezeichen erscheinen.

Die Eingabe über die Tastatur wird durch Drücken der **[RETURN]**-Taste abgeschlossen.

Die Eingabe von Diskette oder Kassette (Stream #9) wird beendet, wenn der Zeilenvorschub erfolgt, oder wenn das Limit von 255 Zeichen für die Stringvariable erreicht ist.

Verwandte Befehle: INPUT

LIST

LIST [‹Zeilenbereich›][,#‹Stream›]

```
LIST 100-1000,#1
```

KOMMANDO: Listet Programmzeilen im angegebenen Bildschirmbereich (Stream) auf. Wird der Parameter nicht angegeben, wird Stream #0 angenommen. Stream #8 ist der Drucker. Durch einmaliges Drücken der **[ESC]**-Taste kann das LISTen unterbrochen und durch die Leertaste wieder fortgesetzt werden. Zweimaliges Drücken von **[ESC]** bricht die Auflistung ab und schaltet auf Direktmodus um.

Die erste oder letzte Nummer des ‹Zeilenbereichs› kann weggelassen werden, wenn das gesamte Programm bis zu einer bestimmten Zeile:

```
LIST -200
```

...bzw. ab einer bestimmten Zeile:

```
LIST 50 -
```

...aufgelistet werden soll. Mit:

```
LIST
```

...wird das gesamte Programm aufgelistet.

Verwandte Befehle: **keine**

LOAD

`LOAD` ‹Dateiname›[`,`‹Adresse›]

```
LOAD "Programm.xyz",&2AF8
```

KOMMANDO: Lädt (engl. load) ein BASIC-Programm in den Speicher und löscht dabei ein eventuell im Speicher befindliches Programm. Wird außerdem eine Adresse angegeben, kann eine binäre Datei an die entsprechende Stelle des Speichers geladen werden, anstatt an die Adresse, von der es gespeichert wurde.

Ein geschütztes Programm kann NICHT mit `LOAD` geladen werden, da es sonst sofort aus dem Speicher gelöscht würde. Für geschützte Programme werden stattdessen die Befehle `RUN` und `CHAIN` verwendet.

Verwandte Befehle: `CHAIN`, `CHAIN MERGE`, `MERGE`, `RUN`, `SAVE`

LOCATE

`LOCATE` [#Stream›`,`]‹x-Koordinate›`,`‹y-Koordinate›

```
10 MODE 1
20 FOR n=1 TO 20
30 LOCATE n,n
40 PRINT CHR$(143);"Position";
50 PRINT n;",";n
60 NEXT
run
```

KOMMANDO: Setzt (engl. locate) den Text-Cursor im angegebenen Bildschirmbereich (Stream) auf die durch die x- und y-Koordinaten definierte Stelle. Position 1,1 befindet sich in der linken oberen Ecke des Bildschirmbereichs. Stream `#0` wird als Standardparameter angenommen, wenn nichts anderes angegeben ist.

Verwandte Befehle: `WINDOW`

LOG

LOG (‹numerischer Ausdruck›)

```
PRINT LOG(9999)
 9.21Ø24Ø37
```

FUNKTION: Berechnet den natürlichen LOGarithmus des ‹numerischen Ausdrucks›, der größer als 0 sein muß.

Verwandte Befehle: EXP, LOG1Ø

LOG1Ø

LOG1Ø (‹numerischer Ausdruck›)

```
PRINT LOG1Ø(9999)
 3.99995657
```

FUNKTION: Berechnet den Logarithmus zur Basis 1Ø des ‹numerischen Ausdrucks›, der größer als 0 sein muß.

Verwandte Befehle: EXP, LOG

LOWER$

LOWER$ (‹String›)

```
1Ø a$="JETZT ERSCHEINT ALLES IN "
2Ø PRINT LOWER$(a$+"KLEINBUCHSTABEN")
run
```

FUNKTION: Alle Buchstaben des angegebenen ‹Strings› werden in Kleinschreibung wiedergegeben. Ein nützliches Hilfsmittel, um Eingaben auszuwerten, die in gemischter Schreibweise eingegeben wurden.

Verwandte Befehle: UPPER$

MASK

MASK [‹ganzzahliger Ausdruck›][,‹erster Punkt›]

```
10 MODE 0:INK 5,21:INK 8,16
20 MOVE -100*RND,400*RND
30 WHILE XPOS<640
40 FOR x=1 TO 8
50 mask 2↑(8-x)
60 DRAWR 32,0,x,1:MOVER -32,0
70 NEXT
80 MOVER 34,0
90 WEND:GOTO 20
run
```

KOMMANDO: Legt fest, welche Schablone (engl. mask) beim Zeichnen von Linien benutzt werden soll. Der Dualwert des ‹ganzzahligen Ausdrucks› im Bereich 0 bis 255 bestimmt, welche Bildpunkte in jeder der aus acht Bildpunkten zusammengesetzten Zeichen gesetzt (1) oder nicht gesetzt (0) werden sollen.

Im Parameter ‹erster Punkt› wird entschieden, ob der erste Punkt der Linie gezeichnet werden soll (1) oder nicht (0).

Einer der beiden Parameter kann ausgelassen werden, jedoch nicht beide zugleich. Wird ein Parameter ausgelassen, so ändert sich die jeweilige Einstellung nicht.

Verwandte Befehle: **DRAW**, **DRAWR**, **GRAPHICS PAPER**, **GRAPHICS PEN**

MAX

MAX (‹Liste von: ‹numerischer Ausdruck›)

```
10 n=66
20 PRINT MAX (1,n,3,6,4,3)
run
 66
```

FUNKTION: Gibt den größten Wert aus der Liste der ‹numerischen Ausdrücke› an.

Verwandte Befehle: **MIN**

MEMORY

MEMORY ‹Adresse›

```
MEMORY &2ØAA
```

KOMMANDO: Setzt den BASIC zur Verfügung stehenden Speicherbereich (engl. memory) fest, indem es die höchste Byte-Adresse angibt.

Beachten Sie, daß BASIC nur die ersten 64K des Speichers beansprucht.

Verwandte Befehle: FRE, HIMEM, SYMBOL, SYMBOL AFTER

MERGE

MERGE ‹Dateiname›

```
MERGE "Neuprog.bas"
```

KOMMANDO: Lädt ein Programm von der Diskette in den Speicher und mischt (engl. merge) es in das bereits im Speicher befindlichen Programm.

Beachten Sie, daß Zeilen des alten Programms von Zeilen des einzufügenden Programms überschrieben werden, wenn sie dieselbe Zeilennummer aufweisen.

Geschützte Dateien, die mit ,P gespeichert wurden, können nicht in das laufende Programm integriert werden.

Verwandte Befehle: CHAIN, CHAIN MERGE, LOAD

MID$

MID$ (‹String›,‹Startposition›[,‹Länge des Unterstrings›])

```
10 MODE 1:ZONE 3
20 a$="ENZYKLOPAEDIE"
30 PRINT "Wie buchstabiert man ";a$
40 PRINT "OK....":PRINT
50 FOR n=1 TO LEN(a$)
60 PRINT MID$(a$,n,1),
70 FOR t=1 TO 700:NEXT t,n
80 PRINT:PRINT
90 INPUT "Gib ein neues Wort an";a$
100 GOTO 50
run
```

FUNKTION: Gibt einen neuen Unterstring wieder, der in der ‹Startposition› des ‹Strings› beginnt und die Anzahl von Zeichen aufweist, die in der ‹Länge des Unterstrings› angegeben wurde. Wird die Anzahl der Buchstaben nicht angegeben, wird alles, was auf die ‹Startposition› des ‹Strings› folgt, wiedergegeben.

Wird für ‹Startposition› eine Zahl angegeben, die die Anzahl der Zeichen im ‹String› übersteigt, wird ein Leerstring ausgegeben. Der Wertebereich für ‹Startposition› liegt zwischen 0 und 255. Die ‹Länge des Unterstrings› kann ebenfalls zwischen 0 und 255 Zeichen liegen.

Verwandte Befehle: `LEFT$`, `RIGHT$`

MID$

`MID$` (‹Stringvariable›`,`‹Einfügestelle›[`,`‹neue Stringlänge›]) = ‹neuer String›

```
10 a$="hallo"
20 MID$(a$,3,2)="XX"
30 PRINT a$
run
haXXo
```

KOMMANDO: Fügt den ‹neuen String› in den String ein, der durch ‹Stringvariable› definiert ist. Der neue String überschreibt den bisherigen Text ab der durch die ‹Einfügestelle› bezeichneten Stelle und ist so lang wie im Parameter ‹neue Stringlänge› angegeben.

Wenn Sie den Befehl `MID$` als Kommando verwenden, muß die ‹Stringvariable›, z.B. `a$` angegeben werden, und NICHT die String-Konstante, z.B. '`hallo`'.

Verwandte Befehle: `LEFT$`, `RIGHT$`

MIN

`MIN` (‹Liste von:‹numerischer Ausdruck›)

```
PRINT MIN(3,6,2.999,8,9)
 2.999
```

FUNKTION: Gibt den kleinsten Wert aus der Liste der ‹numerischen Ausdrücke› an.

Verwandte Befehle: `MAX`

MOD

‹Dividend› MOD ‹Divisor›

```
PRINT 10 MOD 3
 1
PRINT 10 MOD 5
 0
```

OPERATOR: Gibt den (gerundeten) Rest (Modulo) an, nachdem der ‹Dividend› durch den ‹Divisor› geteilt wurde.

Verwandte Befehle: keine

MODE

MODE ‹ganzzahliger Ausdruck›

```
10 m=m+1:IF m>2 THEN m=0
20 MODE m
30 PRINT "Dies ist Modus";m
40 PRINT "Druecke eine beliebige Taste"
50 IF INKEY$="" THEN 50 ELSE 10
run
```

KOMMANDO: Ändert den Bildschirm-Modus (0,1 oder 2) und setzt den Bildschirm auf Farbstift 0 (falls ein anderer Farbstift für den Untergrund verwendet wurde). Alle Text- und Graphik-Fenster werden auf den gesamten Bildschirm zurückgesetzt, Text- und Graphik-Cursor werden an ihre ursprünglichen Positionen gesetzt.

Verwandte Befehle: ORIGIN, WINDOW

MOVE

MOVE ‹x-Koordinate›,‹y-Koordinate›[,[‹Farbstift›][,‹Farbmodus›]]

```
10 MODE 1:TAG
20 x=RND*800-100:y=RND*430
30 MOVE x,y
40 PRINT "Hier bin ich";
50 GOTO 20
run
```

KOMMANDO: Bewegt (engl. move) den Graphik-Cursor zum absoluten, durch ‹x-Koordinate› und ‹y-Koordinate› definierten Punkt. Der ‹Farbstift›-Parameter kann zusätzlich angegeben werden, um den Graphik-Farbstift zu ändern (Bereich 0 bis 15).

Außerdem kann ein ‹Farbmodus› angegeben werden, um zu bestimmen, wie der Farbstift, mit dem geschrieben werden soll, mit den anderen Farben auf dem Graphik-Schirm interagiert. Die vier Farbmodi sind:

0:Normal
1:XOR (exklusives ODER)
2:AND
3:OR

Verwandte Befehle: `MOVER`, `ORIGIN`, `XPOS`, `YPOS`

MOVER

`MOVER` ‹x-Versatz› `,` ‹y-Veratz› [`,` [‹Farbstift›] [`,` ‹Farbmodus›]]

```
10 MODE 1:TAG:MOVE 0,16
20 PRINT "im Leben geht es";
30 FOR n=1 TO 10
40 MOVER -32,16
50 PRINT "auf";:NEXT:PRINT " und";
60 FOR n=1 TO 10
70 MOVER -64,-16
80 PRINT "ab";:NEXT
run
```

KOMMANDO: Bewegt (engl. move) den Graphik-Cursor zu einem Punkt, der von seiner gegenwärtigen Position aus bestimmt wird. Dieser Relative Koordinatenpunkt wird durch die Parameter ‹x-Versatz› und ‹y-Versatz› definiert. Mit dem ‹Farbstift›-Parameter (Bereich 0 bis 15) kann der Graphik-Farbstift geändert werden.

Der (wahlweise anzugebende) Farbmodus bestimmt, wie der ‹Farbstift›, mit dem gezeichnet werden soll, mit den anderen Farben auf dem Bildschirm interagiert. Die vier Farbmodi sind:

0:Normal
1:XOR (exklusives ODER)
2:AND
3:OR

Verwandte Befehle: `MOVE`, `ORIGIN`, `XPOS`, `YPOS`

NEW

NEW

```
NEW
```

KOMMANDO: Löscht das im Speicher befindliche Programm und seine Variablen. Die Tastendefinitionen bleiben erhalten, ebenso die Einstellungen der Bildschirmanzeige, wie MODE, PEN, PAPER, INK usw. Der Bildschirm wird nicht gelöscht.

Verwandte Befehle: **keine**

NEXT

NEXT [‹Liste von:‹Variablen›]

```
10 FOR a=1 TO 3
20 FOR b=0 TO 26
30 MODE 1
40 PEN a:BORDER b
50 PRINT "Schrift";a;"Rand";b
60 FOR c=1 TO 500
70 NEXT c,b,a
run
```

KOMMANDO: Bildet das Ende einer Schleife, die mit FOR beginnt. Das NEXT-Kommando kann ohne oder mit dem Variablennamen angegeben werden, der auf das entsprechende FOR-Kommando verweist. Im obigen Beispiel muß die Liste der Variablen in umgekehrter Reihenfolge zu den entsprechenden FOR-Kommandos aufgeführt werden, damit sich die ineinanderliegenden Schleifen nicht überlappen.

Verwandte Befehle: FOR, STEP, TO

NOT

NOT ‹Aussage›

```
IF NOT "Anna"<"Berta" THEN PRINT "richtig" ELSE PRINT
"falsch"
falsch
IF NOT "Katze">Hund THEN PRINT "richtig" ELSE PRINT
"falsch"
richtig
 ....
PRINT NOT -1
0
PRINT NOT 0
-1
```

OPERATOR: Führt Bit für Bit Operationen mit ganzen Zahlen durch, wobei jedes Bit seinen Umkehrwert erhält.

Weitere Informationen über Logik finden Sie im 2.Teil von Kapitel 9.

Verwandte Befehle: AND, OR, XOR

ON BREAK CONT

ON BREAK CONT

```
10 ON BREAK CONT
20 PRINT "Dieses Programm wird auch dann weiterlaufen,
wenn du versuchst, es mit [ESC] zu unterbrechen":PRINT
30 FOR t=1 TO 1000:NEXT:GOTO 20
run
```

KOMMANDO: Unterbindet die Möglichkeit, das Programm durch Drücken der **[ESC]**-Taste zu unterbrechen. Dieser Befehl sollte mit Vorsicht eingesetzt werden, da das Programm weiterläuft, bis der Computer ganz zurückgesetzt wird. Deshalb sollten Sie ein Programm, das diesen Befehl enthält, abspeichern, bevor Sie es laufen lassen.

ON BREAK CONT kann mit dem Befehl ON BREAK STOP wieder außer Kraft gesetzt werden.

Verwandte Befehle: ON BREAK GOSUB, ON BREAK STOP

ON BREAK GOSUB

ON BREAK GOSUB ‹Zeilennummer›

```
10 ON BREAK GOSUB 40
20 PRINT "Programm laeuft"
30 GOTO 20
40 CLS:PRINT "Wenn du zweimal [ESC] drueckst, ";
50 PRINT "wird die GOSUB-Routine aufgerufen"
60 FOR t=1 TO 2000:NEXT
70 RETURN
run
```

KOMMANDO: Springt in das Unterprogramm in der angegebenen Zeile, wenn die **[ESC]**-Taste zweimal gedrückt wird.

Verwandte Befehle: ON BREAK CONT, ON BREAK STOP, RETURN

ON BREAK STOP

ON BREAK STOP

```
10 ON BREAK GOSUB 40
20 PRINT "Programm laeuft"
30 GOTO 20
40 CLS:PRINT "Wenn du zweimal [ESC] ";
50 PRINT "drueckst, wird die GOSUB-Routine aufgerufen"
60 FOR t=1 TO 2000:NEXT
65 ON BREAK STOP
70 RETURN
run
```

KOMMANDO: Setzt die Befehle ON BREAK CONT und ON BREAK GOSUB außer Kraft, d.h., das Programm wird gestoppt, wenn die **[ESC]**-Taste nochmals betätigt wird. Im obigen Beispiel ist das ON BREAK GOSUB-Kommando nur einmal wirksam, weil das ON BREAK-Unterprogramm in Zeile 65 unterbrochen wird.

Verwandte Befehle: ON BREAK CONT, ON BREAK GOSUB

ON ERROR GOTO

ON ERROR GOTO ‹Zeilennummer›

```
10 ON ERROR GOTO 60
20 CLS:PRINT "Wenn ein Fehler auftritt, ";
30 PRINT "liste das Programm auf"
40 FOR t=1 TO 4000:NEXT
50 GOTO 100
60 PRINT "Fehler in Zeile";
70 PRINT ERL:PRINT:LIST
run
```

KOMMANDO: Springt beim Auftreten eines Fehlers zur angegebenen Zeile.

Mit ON ERROR GOTO 0 wird die Fehler-Falle abgestellt, und BASIC gibt die üblichen Fehlermeldungen aus.

Siehe auch Kommando RESUME.

Verwandte Befehle: DERR, ERL, ERR, ERROR, RESUME

ON ‹Ausdruck› GOSUB

ON ‹Selektor› GOSUB ‹Liste von:‹Zeilennummer›

```
10 PAPER 0: PEN 1:INK 0,1
20 CLS:PRINT "Auswahlmoeglichkeiten":PRINT
30 PRINT "1 - Randfarbe aendern":PRINT
40 PRINT "2 - Schriftfarbe aendern":PRINT
50 PRINT "3 - Modus aendern":PRINT
60 INPUT "Welche Moeglichkeit waehlst du";x
70 ON x GOSUB 90,110,130
80 GOTO 20
90 b=b-1:IF b<0 THEN b=26
100 BORDER b:RETURN
110 p=p-1:IF p<2 THEN p=26
120 INK 1,p:RETURN
130 m=m-1:IF m<0 THEN m=2
140 MODE m:RETURN
run
```

KOMMANDO: Springt in das Unterprogramm in der angegebenen ‹Zeilennummer›, das mit dem ‹Selektor› aufgerufen wird. Der Selektor ist eine positive ganze Zahl zwischen 0 und 255. Der jeweilige Selektorenwert bestimmt, die wievielte Zeilennummer aus der Liste der Zeilennummern aufgerufen wird. Im obigen Beispiel springt BASIC zu Zeile 90, wenn ‹Selektor› 1 gewählt wird, zu Zeile 110, wenn 2 gewählt wird, und zu Zeile 130, wenn 3 gewählt wird.

Ist der Selektorenwert 0, oder höher als die Anzahl der Zeilennummern in der ‹Liste von:‹Zeilennummer›, wird kein Unterprogramm angesprochen.

Verwandte Befehle: RETURN

ON ‹Ausdruck› GOTO

ON ‹Selektor› GOTO ‹Liste von:‹Zeilennummer›

```
9 REM ONXGOTO
10 CLS:PRINT "Auswahlmoeglichkeiten":PRINT
20 PRINT "1 - Programm auflisten":PRINT
30 PRINT "2 - aendern und hinzufuegen":PRINT
40 PRINT "3 - Inhaltsverzeichnis der Diskette":PRINT
50 INPUT "Was waehlst du";n
60 ON n GOTO 80,90,100
70 GOTO 10
80 LIST
90 AUTO
100 CAT
run
```

KOMMANDO: Springt zur angegebenen ‹Zeilennummer›, die mit dem ‹Selektor› aufgerufen wird. Der Selektor ist eine positive ganze Zahl zwischen 0 und 255. Der jeweilige Selektorenwert bestimmt, die wievielte Zeilennummer aus der ‹Liste von:‹Zeilennummer›n aufgerufen wird. Im obigen Beispiel springt BASIC zu Zeile 80, wenn (Selektor) 1 gewählt wird, zu Zeile 90, wenn 2 gewählt wird, und zu Zeile 100, wenn 3 gewählt wird.

Ist der Selektorenwert 0, oder höher als die Anzahl der Zeilennummern in der ‹Liste von:‹Zeilennummer›n, wird keine Zeile angesprochen.

Verwandte Befehle: **keine**

ON SQ GOSUB

ON SQ (‹Kanal›) GOSUB ‹Zeilennummer›

```
10 ENV 1,15,-1,1
20 ON SQ(1) GOSUB 60
30 MODE 0:ORIGIN 0,0,200,440,100,300
40 FOR x=1 TO 13:FRAME:MOVE 330,200,x
50 FILL x:NEXT:GOTO 40
60 READ s:IF s=0 THEN RESTORE:GOTO 60
70 SOUND 1,s,25,15,1
80 ON SQ(1) GOSUB 60:RETURN
90 DATA 50,60,90,100,35,200,24,500,0
run
```

KOMMANDO: Springt zum Unterprogramm in der angegebenen Zeile, wenn bei der Tonausgabe in der Warteschlange des angegebenen ‹Kanals› Platz frei ist. ‹Kanal› ist ein ‹ganzzahliger Ausdruck› mit folgenden Werten:

1 für Kanal A
2 für Kanal B
4 für Kanal C

Weitere Informationen über Tonerzeugung finden Sie im 2.Teil des 9. Kapitels.

Verwandte Befehle: `RETURN`, `SOUND`, `SQ`

OPENIN

`OPENIN` ‹Dateiname›

```
10 REM Oeffne eine Eingabedatei von der Diskette
20 OPENIN "Datei":INPUT #9,a,a$
30 CLOSEIN:PRINT "Die beiden Daten heissen:"
40 PRINT:PRINT a,a$
run
```

KOMMANDO: Eröffnet eine Eingabe-Datei von der Diskette, um sie im laufenden Programm zu verwenden. Es muß sich dabei um eine ASCII-Datei handeln.

Das obige Beispielprogramm wird nur laufen, wenn Sie die Datei aus dem nächsten Beispiel (unter `OPENOUT`) auf Diskette gespeichert haben.

Verwandte Befehle: `CLOSEIN`, `EOF`

OPENOUT

`OPENOUT` ‹Dateiname›

```
10 REM Eroeffne eine Ausgabedatei auf der Diskette
20 INPUT "Gib eine Zahlenvariable an";a
30 INPUT "Gib eine Stringvariable an";a$
40 OPENOUT "Datei"
50 WRITE #9,a,a$
60 CLOSEOUT:PRINT "Daten auf Diskette gespeichert"
run
```

KOMMANDO: Eröffnet eine Ausgabe-Datei auf Diskette.

Verwandte Befehle: `CLOSEOUT`

OR

‹Aussage› OR ‹Aussage›

```
IF "Anna"<"Berta" OR "Hund">"Katze" THEN PRINT "richt
ig" ELSE PRINT "falsch"
richtig
IF "Berta"<"Anna" OR "Katze">"Hund" THEN PRINT "richt
ig" ELSE PRINT "falsch"
falsch
IF "Anna"<"Berta" OR "Katze">"Hund" THEN PRINT "richt
ig" ELSE PRINT "falsch"
richtig

 .....

PRINT 1 OR 1
 1
PRINT Ø OR Ø
 Ø
PRINT 1 OR Ø
 1
```

OPERATOR: Führt Bit für Bit Operationen mit ganzen Zahlen durch. Das Ergebnis ist 1, es sei denn, beide Argumente entsprechen 0.

Weitere Informationen über Logik finden Sie im 2.Teil des 9. Kapitels.

Verwandte Befehle: AND, NOT, XOR

ORIGIN

ORIGIN ‹x›,‹y›[,links,‹rechts›,‹oben›,‹unten›]

```
10 MODE 1:BORDER 13:TAG
20 ORIGIN 0,0,100,540,300,100
30 GRAPHICS PAPER 3:CLG
40 FOR x=550 TO -310 STEP -10
50 MOVE x,206
60 PRINT "Dies ist ein Graphikfenster ";
70 FRAME:NEXT:GOTO 40
run
```

KOMMANDO: Setzt den Koordinaten-Ursprung (engl. origin) an den durch die ‹x› und ‹y›-Parameter bestimmten Punkt.

Durch Angabe der vier Wahl-Parameter kann ein Graphik-Bildschirmbereich bestimmt werden. Liegen die Koordinaten hierfür außerhalb des Bildschirmbereichs, so wird der Bildschirm als Begrenzung gewählt.

Verwandte Befehle: WINDOW

OUT

OUT ‹Interfaceadresse›,‹ganzzahliger Ausdruck›

```
OUT &F8F4,&FF
```

KOMMANDO: Schickt den Wert des ‹ganzzahligen Ausdrucks›, der zwischen 0 und 255 liegen muß, zur Ausgabe an das mit seiner Adresse angesprochene Interface.

Mit Vorsicht anzuwenden!

Verwandte Befehle: INP, WAIT

PAPER

PAPER [#‹Stream›,]‹Farbstift›

```
9 REM paper
10 MODE 0:PEN 0:INK 0,13
20 FOR p=1 TO 15
30 PAPER p:CLS
40 LOCATE 7,12:PRINT "Papier";p
50 FOR t=1 TO 500:NEXT t,p
run
```

KOMMANDO: Bestimmt die Farbe des Hintergrunds, das 'Papier', auf dem die Schrift erscheint. Für jedes Zeichen, das auf den Bildschirm geschrieben wird, füllt sich der zugehörige Hintergrund mit der dem ‹Farbstift› (ink) zugeordneten Farbe des PAPER-Kommandos, falls nicht der Transparentmodus gewählt wurde.

Wird der Stream-Parameter ausgelassen, so wird Stream #0 angenommen.

Die Anzahl der zur Verfügung stehenden ‹Farbstifte› im PAPER-Befehl hängt vom Modus ab.

Verwandte Befehle: GRAPHICS PAPER, INK, PEN

PEEK

PEEK (‹Adresse›)

```
10 MODE 1:ZONE 7
20 WINDOW 1,40,1,2:WINDOW #1,1,40,3,25
30 PRINT "Speicheradresse"
40 LOCATE 20,1:PRINT "Speicherinhalt"
50 FOR n=0 TO 65535
60 p=PEEK(n)
70 PRINT #1,n,"(&";HEX$(n);")";
80 PRINT #1,TAB(20);p,"(&";HEX$(p);")"
90 NEXT
run
```

FUNKTION: Gibt den Inhalt der angegebenen Z80 Speicheradresse an, die im Bereich &0000 bis &FFFF (0 bis 65535) liegen sollte. PEEK gibt an, welcher Wert (im Bereich &00 bis &FF oder 0 bis 255) an der angegebenen RAM-Adresse (nicht an der ROM-Adresse) gespeichert ist.

Verwandte Befehle: POKE

PEN

PEN [#‹Stream›,][‹Farbstift›][,‹Hintergrundmodus›]

```
10 MODE 0:PAPER 0:INK 0,13
20 FOR p=1 TO 15
30 PEN p:PRINT SPACE$(47);"Schrift";p
40 FOR t=1 TO 500:NEXT t,p:GOTO 20
run
```

KOMMANDO: Ordnet den angegebenen ‹Farbstift› zu (Bereich 0 bis 15), der in einem Bildschirmbereich (auf dem gesamten Bildschirm, wenn kein Stream angegeben ist) verwendet werden soll. Als Hintergrundmodus kann transparent (1) oder nicht transparent (0) gewählt werden.

Einer der beiden letzten Parameter kann ausgelassen werden, jedoch nicht beide gleichzeitig. Wird ein Parameter ausgelassen, ändert sich seine letzte Einstellung nicht.

Verwandte Befehle: PAPER

PI

PI

```
PRINT PI
 3.14159265
```

FUNKTION: Gibt den Wert Pi an, d.h. das Verhältnis zwischen Umfang und Durchmesser eines Kreises.

Verwandte Befehle: DEG, RAD

PLOT

PLOT ‹x-Koordinate›,‹y-Koordinate›[,[‹Farbstift›][,‹Farbmodus›]]

```
10 MODE 1:BORDER 0:PAPER 0:PEN 1
20 INK 0,0:INK 1,26:INK 2,13,26:DEG
30 FOR x=1 TO 360:ORIGIN 320,200
40 DRAW 50*COS(x),50*SIN(x),1
50 PLOT 100*COS(x),25*SIN(x):NEXT
60 ORIGIN 0,0:t=TIME+700:WHILE TIME<t
70 PLOT RND*640,RND*400:WEND
80 PLOT RND*640,RND*400,2
90 GOTO 90
run
```

KOMMANDO: Setzt einen Punkt (engl. plot) auf dem Graphik-Bildschirm an die absolute, durch die x- und y-Koordinate bestimmte Stelle. Bei Bedarf kann der ‹Farbstift›, mit dem der Punkt gezeichnet werden soll, aus dem Bereich 0 bis 15 gewählt werden.

Der Wahl-Parameter ‹Farbmodus› bestimmt, wie der ‹Farbstift› mit den übrigen Farben auf dem Bildschirm interagiert. Die vier Farbmodi sind:

0:Normal
1:XOR (exklusives ODER)
2:AND
3:OR

Verwandte Befehle: GRAPHICS PEN, PLOTR

PLOTR

PLOTR ‹x-Versatz›,‹y-Versatz›[,[‹Farbstift›][,‹Farbmodus›]]

```
10 REM Zeichne Linien mit dem Cursortasten
20 BORDER 0:GRAPHICS PEN 1
30 MODE 1:PLOT 320,200
40 IF INKEY(0)=0 THEN PLOTR 0,1
50 IF INKEY(1)=0 THEN PLOTR 1,0
60 IF INKEY(2)=0 THEN PLOTR 0,-1
70 IF INKEY(8)=0 THEN PLOTR -1,0
80 IF INKEY(9)=0 THEN 30:REM copy=loeschen
90 GOTO 40
run
```

KOMMANDO: Setzt einen Punkt (engl. plot) auf dem Graphik-Bildschirm, wobei die x- und y-Koordinaten Relativ zur gegenwärtigen Cursorposition liegen. Wahlweise kann ein ‹Farbstift› zwischen 0 und 15 angegeben werden.

Der ebenfalls wahlweise anzugebende ‹Farbmodus› bestimmt, wie der‹Farbstift›, mit dem gezeichnet werden soll, mit den übrigen Farben auf dem Bildschirm interagiert. Die vier Farbmodi sind:

0:Normal
1:XOR (exklusives ODER)
2:AND
3:OR

Verwandte Befehle: GRAPHICS PEN, PLOT

POKE

POKE ‹Adresse›, ‹ganzzahliger Ausdruck›

```
10 FOR m=49152 TO 65535
20 POKE m,100
30 NEXT
run
```

KOMMANDO: Schreibt den Wert des ‹ganzzahligen Ausdrucks› (Bereich 0 bis 255) direkt in den Z80 Speicher (RAM) an der angegebenen Adresse.

Mit Vorsicht anzuwenden!

Verwandte Befehle: PEEK

POS

POS (#Stream)

```
10 MODE 1:BORDER 0:LOCATE 8,2
20 PRINT "Benutze Cursor-Tasten rechts und links"
30 WINDOW 1,40,12,12:CURSOR 1,1
40 FOR n=1 TO 19:PRINT CHR$(9);:NEXT
50 IF INKEY(1)<>-1 THEN PRINT CHR$(9);
60 IF INKEY(8)<>-1 THEN PRINT CHR$(8);
70 LOCATE #1,2,24
80 PRINT #1,"Text-Cursor ";
90 PRINT #1,"horizontale Position =";
100 PRINT #1,POS(#0):GOTO 50
run
```

FUNKTION: Gibt an, auf welchem Punkt der x-Achse, bezogen auf den linken Rand des Bildschirmbereichs, sich der Text-Cursor gegenwärtig befindet. Der ‹Stream›-Parameter muß unbedingt angegeben werden, da sonst nicht automatisch Stream ·0 angenommen wird.

POS(#8) gibt die Position des Schreibkopfes im Drucker an, wobei 1 die erste Stelle innerhalb der Zeile bedeutet.

POS(#9) Meldet die Anzahl der Zeichen, die seit dem letzten **[RETURN]** auf die Diskette geschrieben worden sind.

Verwandte Befehle: VPOS, WINDOW

PRINT

PRINT [#‹Stream›,][‹Ausgabeliste›]

```
10 a$="klein"
20 b$="Dies ist ein laengerer Text"
30 PRINT a$;a$
40 PRINT a$,a$
50 PRINT
60 PRINT b$;b$
70 PRINT b$,b$
run
```

KOMMANDO: Gibt die ‹Ausgabeliste› auf dem angegebenen Stream aus (auf Stream #0, wenn nicht anders angegeben).

Ein Semikolon zwischen den einzelnen Ausdrücken der Ausgabeliste zeigt an, wo der nächste Ausdruck beginnt. BASIC prüft, ob der nächste Ausdruck noch in die Zeile paßt. Ist dies nicht der Fall, wird er in die nächste Zeile gesetzt.

Ein Komma als Trennzeichen bedeutet, daß der nächste Ausdruck in der nächsten Druckzone beginnt. Hat der vorhergehende Ausdruck seine Druckzone überschritten, so beginnt der folgende Ausdruck in der nächsten freien Zone.

PRINT SPC
PRINT TAB

PRINT [#‹Stream›,][‹Ausgabeliste›][;]
[SPC(‹ganzzahliger Ausdruck›)][‹Ausgabeliste›]

PRINT [#‹Stream›,][‹Ausgabeliste›][;]
[TAB(‹ganzzahliger Ausdruck›)][‹Ausgabeliste›]

```
10 PRINT "Dies ist die SPC-Funktion"
20 FOR x=6 TO 15
30 PRINT SPC(5)"a";SPC(x)"b"
40 NEXT
50 PRINT "Dies ist die TAB-Funktion"
60 FOR x=6 TO 15
70 PRINT TAB(5)"a";TAB(x)"b"
80 NEXT
run
```

SPC druckt die im ‹ganzzahligen Ausdruck› angegebene Anzahl von Leerstellen und setzt den folgenden Ausdruck unmittelbar dahinter, vorausgesetzt, daß er noch in die Zeile paßt. Ein Semikolon hinter SPC erübrigt sich daher.

TAB druckt, vom linken Rand des Textfensters aus beginnend, die angegebene Zahl von Leerstellen und setzt den folgenden Ausdruck unmittelbar dahinter, vorausgesetzt, daß er noch in die Zeile paßt. Ein Semikolon hinter TAB erübrigt sich daher. Liegt der Cursor hinter der gewünschten Position, so wird der nächste Ausdruck um eine Zeile nach unten versetzt.

PRINT USING

PRINT [#‹Stream›][‹Ausgabeliste›][;]
[USING ‹Formatschablone›][‹Trennzeichen››‹Ausdruck›]

```
10 FOR X=1 TO 10
20 n=100000*(RND↑5)
30 PRINT "Waren";USING "########,.##";n
40 NEXT
run
```

Mit diesem Befehl können Sie das Format des Ausdrucks, den Sie durch den Print-Befehl erhalten, bestimmen. Der Ausdruck erscheint dann in der angegebenen ‹Formatschablone›. Trennzeichen sind Komma und Semikolon. Die ‹Formatschablone› kann mit den folgenden Formatzeichen gebildet werden:

Formate für numerische Ausgaben

Innerhalb der Zahl:

`#` Jedes # steht für eine Stelle.
Beispiel: `######`

`.` Bestimmt die Position des Dezimalpunkts
Beispiel: `######.##`

`,` (Steht für eine Stelle.) Kann nur VOR dem Dezimalpunkt erscheinen.Gibt an, daß die Ziffern vor dem Dezimalpunkt in Dreiergruppen (zu Tausendern) eingeteilt werden sollen.
Beispiel: `######,.##`

Außerhalb der Zahl:

`ff` (Steht für zwei Stellen.) Gibt an, daß unmittelbar vor der ersten Ziffer oder dem Dezimalpunkt, aber nach etwaigen, der Zahl vorangehenden Zeichen, ein `f`-Zeichen erscheinen soll. Beachten Sie, daß `f` eine Stelle in der Ziffernfolge einnimmt.
Beispiel: `ff#######,.##`

`**` (Steht für zwei Stellen.) Gibt an, daß vorangehende Leerstellen durch `**` angezeigt werden sollen.
Beispiel: `**######,.##`

**£ (Steht für drei Stellen.) Steht für eine Kombination aus ** und ££, d.h. Sternchen ** und Pfund-Zeichen £ werden der Zahl vorangestellt.
Beispiel: **£######,.##

$$ (Steht für zwei Stellen.) Gibt an, daß unmittelbar vor der ersten Ziffer oder dem Dezimalpunkt, aber nach etwaigen, der Zahl vorangehenden Zeichen, ein $-Zeichen erscheinen soll.
Beispiel: $$######,.##

**$ (Steht für drei Stellen) Kombiniert die ** und $$ Anweisungen.
Beispiel: **$######,.##

+ Je nach dem Wert der Zahl soll ein + oder ein - Zeichen vorangestellt werden. Erscheint + am Anfang der Formatschablone, wird das Zeichen vor die Zahl und eventuelle Währungszeichen gesetzt. Erscheint es am Ende der Formatschablone, wird + oder - hinter die Zahl und eventuelle Exponenten gesetzt.
Beispiel: +####.####

- Das Minuszeichen kann nur am Ende der Formatschablone erscheinen. Es bewirkt, daß das Zeichen hinter jede negative Zahl (und ihre Exponenten) gesetzt wird. Ist die Zahl positiv, wird eine Leerstelle ausgedruckt. Wird in der Formatschablone kein Minuszeichen angegeben, erscheint VOR jeder negativen Zahl automatisch ein Minuszeichen.Beispiel: ####.####-

↑↑↑↑ Gibt an, daß die Zahl in Exponentialschreibweise dargestellt werden soll. Dieses Zeichen sollte hinter den Zifferzeichen, jedoch vor angehängten + oder - Zeichen stehen.
Beispiel: #.####↑↑↑↑+

Die ‹Formatschablone› für Zahlen darf nicht mehr als 20 Zeichen enthalten.

Reicht die Formatschablone für einen Ausdruck nicht aus, z.B.:

```
PRINT USING "####";12345678
```

...so wird der Ausdruck nicht gekürzt, um in die Schablone zu passen, sondern in seiner ganzen Länge ausgedruckt, wobei ein % Zeichen vorangestellt wird, um anzudeuten, daß das Format nicht eingehalten wurde.

Formate für Strings

```
10 CLS:a$="abcdefghijklmnopqrst"
20 PRINT "eingegebener Ausdruck= ";a$
30 PRINT:PRINT "! Formatzeichen= ";
40 PRINT USING "!";a$
50 PRINT:PRINT "\Leerzeichen\ Formatzeichen= ";
60 PRINT USING "\          \";a$
70 PRINT:PRINT "& Formatzeichen= ";
80 PRINT USING "&";a$
90 GOTO 90
run
```

! Gibt an, daß nur das erste Zeichen des Strings gedruckt werden soll.
Beispiel: !

\‹Leerstellen›\
Gibt an, daß nur die ersten x Zeichen des Strings gedruckt werden sollen, wobei x einschließlich der Schrägstriche der Länge der Schablone entspricht.
Beispiel: \ \

& Gibt an, daß der gesamte String wie vorgegeben gedruckt werden soll.
Beispiel: &

Die ‹Formatschablone› darf nicht über 255 Stellen hinausgehen.

Formatschablonen sowohl für numerische Angaben als auch für Strings können durch Stringvariablen dargestellt werden, z.B.:

```
10 a$="########,.##"
20 b$="!"
30 PRINT USING a$;12345.6789;
40 PRINT USING b$;"Pfennig"
```

Weitere Informationen über Formatschablonen finden Sie im 2.Teil des 9. Kapitels.

Verwandte Befehle: SPC, TAB, USING, ZONE

RAD

RAD

```
RAD
```

KOMMANDO: Schaltet auf Ausgabe in Bogenmaß (engl. radians) um. BASIC gibt automatisch Bogenmaßwerte aus, wenn der Computer angeschaltet oder zurückgesetzt wird, oder nach Kommandos wie NEW, CLEAR und LOAD, RUN.

Verwandte Befehle: ATN, COS, DEG, SIN, TAN

RANDOMIZE

RANDOMIZE [‹numerischer Ausdruck›]

```
10 RANDOMIZE 123.456
20 PRINT RND
   0.258852139
```

KOMMANDO: Setzt den Anfangswert für den Zufallszahlengenerator. In BASIC werden Zufallszahlen in Abhängigkeit von einem gegebenen Startwert erzeugt. Die Zahlenfolge steht fest. Der Anfangswert kann durch den Parameter ‹numerischer Ausdruck› festgesetzt werden. Wird der Parameter ausgelassen, muß der Benutzer ihn angeben, wenn er vom Computer gefragt wird: 'Random number seed?'
Mit RANDOMIZE TIME wird eine nahezu unwiederholbare Zahlenfolge erzeugt.

Verwandte Befehle: RND

READ

READ ‹Liste von:‹Variablen›

```
10 FOR n=1 TO 8
20 READ a$,c
30 PRINT a$;" ";:SOUND 1,c:NEXT
40 DATA hier,478,sind,426,8,379,Noten
50 DATA 358,von,319,einer,284,Ton,253,leiter,239
run
```

KOMMANDO: Liest (engl. read) Daten aus DATA-Anweisungen und ordnet sie den angegebenen Variablen der Reihe nach zu. Das RESTORE-Kommando setzt den Zeiger wieder auf den ersten DATA-Wert.

Weitere Einzelheiten über Daten erfahren Sie im 2.Teil des 9. Kapitels.

Verwandte Befehle: DATA, RESTORE

RELEASE

RELEASE ‹Kanäle›

```
10 SOUND 65,1000,100
20 PRINT "Druecke R, um Ton zu erzeugen"
30 IF INKEY(50)=-1 THEN 30
40 RELEASE 1
run
```

KOMMANDO: Hebt den Wartezustand, in dem sich die Kanäle aufgrund des SOUND-Befehls befinden, auf.

Für den Parameter ‹Kanäle› können ganze Zahlen zwischen 1 und 7 angegeben werden, die folgende Bedeutung haben:

1: gibt Kanal A frei
2: gibt Kanal B frei
3: gibt Kanäle A und B frei
4: gibt Kanal C frei
5: gibt Kanäle A und C frei
6: gibt Kanäle B und C frei
7: gibt Kanäle A, B und C frei

Weitere Informationen über Tonerzeugung finden Sie im 2.Teil des 9. Kapitels.

Verwandte Befehle: SOUND

REM

REM ‹Rest der Zeile›

```
10 REM Intergalactic Hyperspace Mega-Monster
   Invaders Deathchase by AMSOFT
20 REM Copyright AMSOFT 1985
```

KOMMANDO: Fügt Kommentarzeilen (engl. REMarks) in das Programm ein. Alle Zeichen von REM bis zum Ende der Zeile werden von BASIC ignoriert, einschließlich des Doppelpunktes, der normalerweise Befehle trennt.

Anstelle von REM kann das einfache Anführungszeichen ' verwendet werden, außer in DATA-Anweisungen, wo ' als Teil eines Zitats behandelt wird.

Verwandte Befehle: **keine**

REMAIN

REMAIN ‹Zeitgeber›

```
10 AFTER 500,1 GOSUB 40
20 AFTER 100,2 GOSUB 50
30 PRINT "Programm laeuft":GOTO 30
40 REM Dieses GOSUB-Unterprogramm wird nicht aufgerufen
   , da es durch Zeile 80 ausser Kraft gesetzt wird.
50 PRINT:PRINT "Zeitgeber 1 ist jetzt ";
60 PRINT "durch REMAIN ausser Kraft gesetzt."
70 PRINT "Verbleibende Zeiteinheiten :";
80 PRINT REMAIN(1)
run
```

FUNKTION: Gibt die verbleibende (engl. REMAINing) Restzeit des ‹Zeitgebers› (Bereich 0 bis 3) an und setzt ihn außer Kraft.

Weitere Einzelheiten über Unterbrechungen finden Sie im 2.Teil des 9. Kapitels.

Verwandte Befehle: AFTER, DI, EI, EVERY

RENUM

RENUM [‹neue Zeilennummer.][,[‹alte Zeilennummer›][,‹Schrittweite›]]

```
10 CLS
20 REM dies wird Zeile 123
30 REM dies wird Zeile 124
40 REM dies wird Zeile 125
RENUM 123,20,1
LIST
```

KOMMANDO: Numeriert das Programm um (engl. RENUMber).

Die ‹alte Zeilennummer› gibt an, ab welcher Zeile neu numeriert werden soll. Wird dieser Parameter ausgelassen, beginnt die neue Numerierung in der ersten Programmzeile.

Die ‹neue Zeilennummer› gibt an, wie die erste umnumerierte Zeile heißen soll. Wird dieser Parameter ausgelassen, heißt die erste neue Zeilennummer 10.

Die ‹Schrittweite› bestimmt die Abstände zwischen den einzelnen Zeilennummern. Wird dieser Parameter ausgelassen, betragen die Abstände automatisch 10.

In allen GOSUB- und GOTO-Befehlen und anderen Zeilenverweisen werden die Zeilennummern geändert. Zeilenangaben innerhalb von Strings, wie z.B. in KEY-Befehlen, werden jedoch nicht geändert, ebensowenig wie in REM-Aussagen und in den ‹Zeilennummer›-Parametern in CHAIN- und CHAIN MERGE-Befehlen.

Die Zeilennummern können von 1 bis 65535 reichen.

Verwandte Befehle: DELETE, LIST

RESTORE

RESTORE [‹Zeilennummer›]

```
10 READ a$:PRINT a$;" ";
20 RESTORE 50
30 FOR t=1 TO 500:NEXT:GOTO 10
40 DATA Mit RESTORE kann man Daten noch einmal lesen
50 DATA und noch einmal
run
```

KOMMANDO: Setzt den Zeiger zurück auf die DATA-Anweisung in der angegebenen Zeile. Wird die ‹Zeilennummer› nicht angegeben, so geht der Zeiger auf die erste DATA-Anweisung im Programm zurück.

Weitere Informationen zu DATA-Anweisungen finden Sie im 2.Teil des 9. Kapitels.

Verwandte Befehle: DATA, READ

RESUME

RESUME [‹Zeilennummer›]

```
10 ON ERROR GOTO 60
20 FOR x=10 TO 0 STEP-1:PRINT 1/x:NEXT
30 END
40 PRINT "gehe hierher, wenn ein Fehler auftritt"
50 END
60 PRINT "Fehler Nr.";ERR;"in Zeile";ERL
70 RESUME 40
run
```

KOMMANDO: Veranlaßt das Programm, in der nächsten Zeile oder bei der angegebenen ‹Zeilennummer› weiterzumachen, wenn es aufgrund eines ON ERROR GOTO-Sprungs unterbrochen wurde. Wird die ‹Zeilennummer› nicht angegeben, macht das Programm in der Zeile weiter, wo der Fehler aufgetreten ist. Entfernen Sie probeweise den ‹Zeilennummer›-Parameter aus dem Beispielprogramm und lassen Sie das Programm erneut laufen.

```
70 RESUME
run
```

Verwandte Befehle: DERR, ERL, ERR, ERROR, ON ERROR GOTO, RESUME NEXT

RESUME NEXT

RESUME NEXT

```
10 ON ERROR GOTO 90
20 PRINT "druecke jedesmal [RETURN]"
30 INPUT "1";a
40 INPUT "2";a
50 inpot "3";a :REM syntax error!
60 INPUT "4";a
70 INPUT "5";a
80 END
90 PRINT "Fehler Nr.";ERR;"in Zeile";ERL
100 RESUME NEXT
run
```

KOMMANDO: Nimmt das Programm wieder auf (engl. resume), nachdem ein Fehler aufgetreten ist und die ON ERROR GOTO-Anweisung ausgeführt wurde.

Das Programm läuft ab der Zeile weiter, die der Zeile mit dem Fehler folgt.

Verwandte Befehle: DERR, ERL, ERR, ERROR, ON ERROR GOTO, RESUME

RETURN

RETURN

```
10 GOSUB 50:PRINT "nach dem GOSUB-":END
50 FOR n=1 TO 20
60 PRINT "Unterprogramm"
70 NEXT:PRINT
80 RETURN
run
```

KOMMANDO: Steht am Ende eines Unterprogramms. BASIC kehrt vom Unterprogramm zu der Anweisung zurück (engl. return), die dem Aufruf des Unterprogramms folgt.

Verwandte Befehle: GOSUB

RIGHT$

RIGHT$ (‹String›,‹Länge›)

```
10 MODE 1:a$="CPC6128 Computer"
20 FOR n=1 TO 16:LOCATE 41-n,n
30 PRINT RIGHT$(a$,n)
40 NEXT
run
```

FUNKTION: Gibt den ‹String› in der angegebenen ‹Länge› (Zeichenanzahl 0 bis 255) wieder. Dabei wird der $tring von rechts (engl. right) beginnend in jeder Zeile um ein Zeichen verlängert. Ist er kürzer als der ‹Längen›-Parameter angibt, wird der ganze String entsprechend oft wiederholt.

Verwandte Befehle: LEFT$, MID$

RND

RND [(‹numerischer Ausdruck›)]

```
10 RANDOMIZE
20 FOR x=1 TO -1 STEP -1
30 PRINT "rnd Parameter=";x
40 FOR n=1 TO 6
50 PRINT RND(x)
60 NEXT n,x
run
```

FUNKTION: Gibt die nächste Zufallszahl (engl. RaNDom number) in der betreffenden Folge wieder, wenn der ‹numerische Ausdruck› positiv ist oder nicht angegeben wird.

Wird 0 als Parameter gewählt, so wird die letzte Zufallszahl wiederholt.

Wird für den Parameter eine negative Zahl eingesetzt, so wird eine neue Folge von Zufallszahlen ausgegeben, deren erste Zahl sich immer wiederholt.

Verwandte Befehle: RANDOMIZE

ROUND

ROUND (‹numerischer Ausdruck›[,‹Dezimalstellen›])

```
10 FOR n=4 TO -4 STEP-1
20 PRINT ROUND (1234.5678,n),
30 PRINT "Dezimalstellen: ";n
40 NEXT
run
```

FUNKTION: Rundet (engl. round) den ‹numerischen Ausdruck› zu einer Zahl mit sovielen Dezimalstellen oder Zehnerpotenzen, wie der ‹Dezimalstellen›-Parameter angibt. Ist der ‹Dezimalstellen›-Parameter negativ, wird der ‹numerische Ausdruck› zu einer ganzen Zahl mit sovielen Nullen vor dem Komma gerundet, wie der ‹Dezimalstellen›-Parameter angibt.

Verwandte Befehle: ABS, CINT, FIX, INT

RUN

RUN ‹String›

```
RUN "disc"
```

KOMMANDO: Lädt ein BASIC-Programm von der Diskette in den Rechner und startet (engl. run) es. Hierbei wird jedes vorher geladene Programm aus dem Speicher gelöscht.

Geschützte BASIC-Programme können auf diese Weise direkt gestartet werden.

Verwandte Befehle: LOAD

RUN

RUN [‹Zeilennummer›]

```
RUN 200
```

KOMMANDO: Startet ein im Speicher befindliches Programm von vorn oder ab der angegebenen ‹Zeilennummer›. Die Werte aller im Programm befindlichen Variablen werden auf Null gesetzt.

Geschützte Programme können nach dem Laden NICHT auf diese Weise gestartet werden.

Verwandte Befehle: CONT, END, STOP

SAVE

SAVE ‹Dateiname›[,‹Dateityp›][,‹Binärdateiparameter›]

```
SAVE "Programm.xyz"
```

... speichert das Programm als normale, ungeschützte BASIC-Datei.

```
SAVE "Programm.xyz",P
```

... speichert das Programm als geschützte BASIC-Datei.

```
SAVE "Programm.xyz",A
```

... speichert das Programm als ASCII-Datei.

```
SAVE "Programm.xyz",B,8000,3000,8001
```

... speichert das Programm als Binärdatei. In diesem Beispiel wird der Speicherbereich des Computers ab der Speicheradresse 8000 gespeichert. Der Umfang der Datei beträgt 3000 Bytes und die Adresse des Anfangspunkts ist 8001.

KOMMANDO: Schreibt das im Speicher befindliche Programm auf Diskette. Eine Binärdatei ist ein auf Diskette übertragener Speicherbereich. Die Binärdatei-Parameter sind:

‹Startadresse›, ‹Dateilänge›[,‹Anfangspunkt›]

Die Bildschirmanzeige kann als Binärdatei gespeichert werden. Solch ein Bildschirmabdruck erfolgt durch den Befehl:

```
SAVE "Bild",B,&C000,&4000
```

...und mit dem Befehl:

```
LOAD "Bild"
```

...kann das Bild wieder auf den Bildschirm geholt werden.

Verwandte Befehle: CHAIN, CHAIN MERGE, LOAD, MERGE, RUN

SGN

SGN (‹numerischer Asudruck›)

```
10 FOR n=200 TO -200 STEP-20
20 PRINT "Fuer den Wert";n;
30 PRINT "gibt SGN";
40 PRINT SGN(n);
50 PRINT "an."
60 NEXT
run
```

FUNKTION: Gibt das Vorzeichen (engl. SiGN) des ‹numerischen Ausdrucks› an. Ist der ‹numerische Ausdruck› kleiner als 0, erscheint -1, ist er gleich 0, erscheint 0, und ist er größer als 0, erscheint 1.

Verwandte Befehle: ABS

SIN

SIN (‹numerischer Ausdruck›)

```
10 CLS:DEG:ORIGIN 0,200
20 FOR n=0 TO 720
30 y=SIN(n)
40 PLOT n*640/720,198*y:NEXT
50 GOTO 50
run
```

FUNKTION: Berechnet den Sinus des ‹numerischen Ausdrucks›.

Mit den Befehlen DEG und RAD kann das Ergebnis in Winkelgrade bzw. Bogengrade umgewandelt werden.

Verwandte Befehle: ATN, COS, DEG, RAD, TAN

SOUND

SOUND ‹Kanalstatus› , ‹Tonperiode› [, ‹Dauer› [, ‹Lautstärke›
[, ‹Lautstärkenhüllkurve› [, ‹Tonhüllkurve› [, ‹Geräuschperiode›]]]]]

```
10 FOR z=0 TO 4095
20 SOUND 1,z,1,12
30 NEXT
run
```

KOMMANDO: Erzeugt einen Ton. Der Befehl hat folgende Parameter:

Parameter 1: ‹Kanalstatus›

Für ‹Kanalstatus› muß eine ganze Zahl zwischen 1 und 255 angegeben werden. Der Parameter ist Bit-signifikant. Die einzelnen Bits aus dem Binärwert von ‹Kanalstatus› haben folgende Bedeutung:

Bit	Dezimalwert	Kommando
0	1	Ton wird zum Kanal A geschickt
1	2	Ton wird zum Kanal B geschickt
2	4	Ton wird zum Kanal C geschickt
3	8	Rendezvous mit Kanal A
4	16	Rendezvous mit Kanal B
5	32	Rendezvous mit Kanal C
6	64	Halte den Tonkanal
7	128	Leere den Tonkanal

Wird also z.B. der Wert 64 für ‹Kanalstatus› eingesetzt, bedeutet dies: Der Ton wird zum Kanal C (4) geschickt und dort gehalten (64).

Parameter 2: ‹Tonperiode›

Dieser Parameter bestimmt die Tonhöhe, d.h. die Note, die gespielt werden soll. Für jede Note ist eine Zahl festgelegt, die für die ‹Tonperiode› eingesetzt wird. Siehe Kapitel 7, Teil 5.

Parameter 3: ‹Dauer›

Dieser Parameter setzt die Dauer eines Tones fest. Der Wert bestimmt, wieviele Hundertstelsekunden der Ton dauert. Wird die ‹Dauer› nicht angegeben, wird als Standardwert 20 (= 1/5 Sekunde) angenommen.

Ist der Wert 0, so dauert der Ton bis zum Ende der Lautstärkenhüllkurve.

Ist der ‹Dauer›-Parameter negativ, wird die Lautstärkenhüllkurve dem positiven Wert der Zahl entsprechend oft wiederholt.

Parameter 4: ‹Lautstärke›

Dieser Parameter bestimmt die Lautstärke am Anfang des Tones. Der Wertebereich liegt zwischen 0 und 15 (0= keine Lautstärke, 15= maximale Lautstärke). Wird keine Zahl angegeben, wird als Standardwert 12 angenommen.

Parameter 5: ‹Lautstärkenhüllkurve›

Um die Lautstärke eines Tones während seiner Dauer zu variieren, kann mit Hilfe des Befehls ENV eine ‹Lautstärkenhüllkurve› definiert werden. Dazu stehen 15 verschiedene Hüllkurven zur Verfügung (Wertebereich 1 bis 15). Der ‹Lautstärkenhüllkurve›-Parameter im SOUND-Befehl entspricht der Lautstärkenhüllkurven-Nummer im ENV-Befehl.

Siehe auch Beschreibung des ENV-Befehls.

Parameter 6: ‹Tonhüllkurve›

Um die Tonhöhe eines Tones während seiner Dauer zu variieren, kann mit Hilfe des ENT-Befehls eine Tonhüllkurve definiert werden. Dazu stehen 15 verschiedene Tonhüllkurven zur Verfügung (Wertebereich 1 bis 15). Der ‹Tonhüllkurve›-Parameter im SOUND-Befehl entspricht der Tonhüllkurven-Nummer im ENT-Befehl. Haben Sie im ENT-Befehl eine negative Hüllkurven-Nummer angegeben, benutzen Sie den positiven Wert der Zahl im ‹Hüllkurven›-Parameter des SOUND-Befehls.

Siehe auch Beschreibung des ENT-Befehls.

Parameter 7: ‹Geräuschperiode›

Weißes Rauschen kann den Ton in verschiedenen Varianten begleiten oder abgeschaltet werden. Der Wertebereich für den ‹Geräuschperiode›-Parameter liegt zwischen 0 und 31.

Weitere Einzelheiten über die Tonerzeugung finden Sie im 2.Teil des 9. Kapitels.

Verwandte Befehle: ENT, ENV, ON SQ GOSUB, RELEASE, SQ

SPACE$

SPACE$ (‹ganzzahliger Ausdruck›)

```
10 MODE 1
20 PRINT "Zwischen hier";
30 PRINT SPACE$(9);
40 PRINT "und dort "
50 PRINT "liegen 9 Leerstellen"
run
```

FUNKTION: Gibt einen $tring aus, der aus der angegebenen Anzahl (0 bis 255) von Leerzeichen (engl. space) besteht.

Verwandte Befehle: SPC, STRING$, TAB

SPC

siehe PRINT SPC

SPEED INK

SPEED INK ‹Periode 1›,‹Periode 2›

```
10 BORDER 7,18
20 FOR i=30 TO 1 STEP -1
30 SPEED INK i,i
40 FOR t=1 TO 700:NEXT t,i
run
```

KOMMANDO: Setzt die Geschwindigkeit fest, mit der sich die beiden im INK-oder BORDER-Befehl angegebenen Farben abwechseln sollen. ‹Periode 1› setzt die Farbdauer für die erste Farbe, ‹Periode 2› die Farbdauer für die zweite Farbe fest. Die Zeitdauer wird in Einheiten zu 0.02 Sekunden, oder Fünfzigstelsekunden, gemessen.

Wählen Sie die Farben und ihre Farbdauer sorgfältig aus, um hypnotische Effekte zu vermeiden.

Verwandte Befehle: BORDER, INK

SPEED KEY

SPEED KEY ‹Startverzögerung›,‹Wiederholungsperiode›

```
10 CLS:FOR k=7 TO 1 STEP-2
20 PRINT "Schreibe deinen Namen und druecke die
   [RETURN] Taste"
30 SPEED KEY k,k
40 LINE INPUT a$:NEXT
50 PRINT "Komischer Name!"
run
```

KOMMANDO: Legt fest, in welchen Zeitabständen sich ein Zeichen in Dauerfunktion wiederholt. Der Wert für den ‹Startverzögerungs›-Parameter bestimmt, wieviel Fünfzigstelsekunden (0,02 Sekunden) bis zum Einsetzen der Dauerfunktion verstreichen sollen. Der ‹Wiederholungsperioden›-Parameter gibt an, in welchen Zeitabständen eine Taste in Dauerfunktion angeschlagen wird.

Der Befehl SPEED KEY läßt sich nur auf Tasten anwenden, die standardmäßig mit Dauerfunktion ausgestattet sind, oder auf Tasten, die durch KEY DEF auf Dauerfunktion geschaltet wurden.

Wenn Sie für ‹Startverzögerung› kleine Werte einsetzen möchten, ist es ratsam, vorher eine Zahlentaste so zu programmieren, daß die Tastatur zu ihrem SPEED KEY Standard-Tempo von 30,2 zurückkehrt. Durch den Befehl:

```
KEY 0,"SPEED KEY 30,2"+CHR$(13)
```

...kehrt die Dauerfunktion zum Standardtempo zurück, wenn die 0 auf der Zahlentastatur gedrückt wird.

Verwandte Befehle: KEY DEF

SPEED WRITE

SPEED WRITE ‹ganzzahliger Ausdruck›

```
SPEED WRITE 1
```

KOMMANDO: Bestimmt, mit welcher Geschwindigkeit Daten auf eine angeschlossene Kassette gespeichert oder geschrieben werden. Lautet der ‹ganzzahlige Ausdruck› 1, so wird die Kassette mit 2000 Baud (Bits pro Sekunde) beschrieben. Wird der Wert 0 angegeben, beträgt die Schreibgeschwindigkeit 1000 Baud (Standardgeschwindigkeit). Beim Laden einer Datei von der Kassette wählt der Computer automatisch die richtige Lesegeschwindigkeit.

Um eine bessere Datensicherheit zu gewährleisten, empfiehlt es sich, SPEED WRITE 0 (die Standardgeschwindigkeit) zu wählen.

Der SPEED WRITE-Befehl ist ohne Bedeutung für den Diskettenbetrieb.

Verwandte Befehle: OPENOUT, SAVE

SQ

SQ (‹Kanal›)

```
10 SOUND 65,100,100
20 PRINT SQ(1)
run
 67
```

FUNKTION: Gibt die Anzahl der freien Plätze in der Tonwarteschlange (engl. Sound Queue) des durch 1,2 oder 4 bezeichneten Kanals an, wobei

1 für Kanal A
2 für Kanal B und
4 für Kanal C steht.

Die SQ-Funktion gibt eine Bit-signifikante ganze Zahl an, wobei die Bits folgendes aussagen:

Bits 0,1 und 2:	die Anzahl der freien Plätze in der Warteschlange
Bits 3,4 und 5:	Rendezvous-Zustand am oberen Ende der Schlange
Bit 6:	das obere Ende der Schlange befindet sich im Haltezustand
Bit 7:	der Kanal ist gerade aktiv

Bit 0 hat dabei die niedrigste, Bit 7 die höchste Signifikanz.

Daraus ergibt sich, daß Bit 7 nicht gesetzt werden kann, wenn Bit 6 gesetzt ist, und umgekehrt. Ebensowenig können Bits 6 und 7 gesetzt werden, wenn die Bits 3,4 oder 5 gesetzt sind.

Nähere Einzelheiten über die Erzeugung von Tönen finden Sie im 2.Teil des 9. Kapitels.

Verwandte Befehle: ON SQ GOSUB, SOUND

SQR

SQR (‹numerischer Ausdruck›)

```
PRINT SQR(9)
 3
```

FUNKTION: Gibt die Quadratwurzel (engl. SQuare Root) des angegebenen ‹numerischen Ausdrucks› wieder.

Verwandte Befehle: **keine**

STEP

(Siehe FOR)

STOP

STOP

```
10 FOR n=1 TO 30:PRINT n:NEXT
20 STOP
30 FOR n=31 TO 60:PRINT n:NEXT
run
cont
```

KOMMANDO: Stoppt das Programm, beläßt BASIC jedoch in einem Unterbrechungszustand, so daß das Programm mit dem CONT-Befehl wieder fortgesetzt werden kann. Das Kommando kann dazu benutzt werden, ein Programm zu Testzwecken an einer bestimmten Stelle zu unterbrechen.

Verwandte Befehle: CONT, END

STR$

STR$ (‹numerischer Ausdruck›)

```
10 a=&FF :REM 255 hexadezimal
20 b=&X1111 :REM 15 dual
30 c$="***"
40 PRINT c$+STR$(a+b)+c$
run
*** 270***
```

FUNKTION: Verwandelt den ‹numerischen Ausdruck› in einen Dezimal-String.

Verwandte Befehle: BIN$, DEC$, HEX$, VAL

STRING$

STRING$ (‹Länge›,‹Textzeichen›)

```
PRINT STRING$(40,"*")
****************************************
```

FUNKTION: Gibt einen String aus, der das angegebene ‹Textzeichen› so oft (zwischen 0 und 255 mal) enthält, wie im ‹ganzzahligen Ausdruck› angegeben. Dasselbe Ergebnis wird erzielt, wenn, man folgendes eingibt:

```
PRINT STRING$(40,42)
****************************************
```

Hier entspricht das ‹Textzeichen› 42 dem ASCII-Wert des Zeichens *. Es ist eine andere Form des Befehls:

```
PRINT STRING$(40,CHR$(42)).
```

Verwandte Befehle: SPACE$

SWAP

(Siehe WINDOW SWAP)

SYMBOL

SYMBOL ‹Zeichennummer›,‹Liste von: ‹Zeilen›

```
10 MODE 1:SYMBOL AFTER 105
20 Zeile1=255:REM binary 11111111
30 Zeile2=129:REM binary 10000001
40 Zeile3=189:REM binary 10111101
50 Zeile4=153:REM binary 10011001
60 Zeile5=153:REM binary 10011001
70 Zeile6=189:REM binary 10111101
80 Zeile7=129:REM binary 10000001
90 Zeile8=255:REM binary 11111111
100 PRINT "Zeile 110 belegt den Buchstaben i (105)
    mit einem neuen Wert. Tippe ein paar i's und
    sieh, was herauskommt. Liste nun das Programm auf."
110 SYMBOL 105,Zeile1,Zeile2,Zeile3,Zeile4,Zeile5,
    Zeile6,Zeile7,Zeile8
run
```

KOMMANDO: Verändert die Form eines Zeichens auf dem Bildschirm. Jeder Parameter muß eine ganze Zahl zwischen 0 und 255 enthalten.

Um Platz im Speicher des CPC6128 für das neudefinierte Zeichen zu schaffen, muß erst der Befehl:

```
SYMBOL AFTER x
```

...erteilt werden, wobei x für die Nummer dieses Zeichens steht, oder kleiner ist als diese.

Erst dann kann der Befehl **SYMBOL** eingesetzt werden, dem immer als erstes die Zeichenummer folgen muß.

Unabhängig davon, ob der Wert für x ein Zeichen repräsentiert, das direkt eingetippt werden kann, ist es möglich, das umdefinierte Zeichen mit folgendem Befehl auf den Bildschirm zu bringen:

```
PRINT CHR$(x)
```

Nach **SYMBOL(x)** können bis zu acht Parameter angegeben werden, die von oben nach unten die acht Zeilen darstellen, aus denen jedes Zeichen zusammengesetzt ist. Jeder dieser Parameter kann Werte zwischen 0 und 255 aufweisen. Der binäre Ausdruck jedes Parameters bestimmt das Muster der betreffenden Reihe im neudefinierten Zeichen.

Lautet der erste der acht Parameter z.B. 1, so erhält die oberste Reihe des neuen Zeichens das Aussehen des binären Ausdrucks für 1, nämlich 00000001. Der Punkt, an dem die 1 des binären Ausdruck steht, erscheint in der Schrift-Farbe, während überall dort, wo der binäre Ausdruck eine 0 aufweist, das Zeichen unsichtbar bleibt, da es in der 'Papier'-Farbe gedruckt wird. Das neue Zeichen würde also in diesem Beispiel oben rechts einen Punkt aufweisen. Wir führen dieses Beispiel fort und geben für die anderen sieben Parameter die Zahlen **3,7,15,31,63,Ø,Ø** an. In binärer Schreibweise sehen die acht Parameter dann folgendermaßen aus:

‹Zeilen›-Parameter	Binärzahl	Dezimalzahl
1	00000001	1
2	00000011	3
3	00000111	7
4	00001111	15
5	00011111	31
6	00111111	63
7	00000000	0
8	00000000	0

Wenn man sich die binäre Schreibweise dieser acht Parameter ansieht, kann man bereits erkennen, wie das neue Zeichen aussehen wird. Wir ordnen jetzt die acht Parameter dem Zeichen Nummer 255 zu mit folgendem Befehl:

```
SYMBOL 255,1,3,7,15,31,63,0,0
```

Der Wert 0 für die beiden letzten Parameter kann weggelassen werden:

```
SYMBOL 255,1,3,7,15,31,63
```

Die Parameter können ebensogut in Binärform angegeben werden:

```
SYMBOL 255,&X00000001,&X00000011,&X00000111,&X00001111,
&X00011111,&X00111111
```

Dies erspart die Umwandlung des Bit-Musters für das neue Zeichen, das Sie sich ausgedacht haben, in Dezimalschreibweise.

Sehen Sie sich jetzt das neue Zeichen an:

```
PRINT CHR$(255)
```

Wird das neue Zeichen einem Zeichen mit einer eigenen Taste zugeordnet, so erscheint das neue Zeichen jedesmal, wenn diese Taste gedrückt wird, und überall dort, wo der Computer das vorherige Zeichen ausgegeben hätte. BASIC wird auf das neue Zeichen nicht mit Unverständnis reagieren, sondern es mit dem alten Zeichen gleichsetzen. Weitere Informationen über selbstdefinierte Zeichen finden Sie im 2.Teil des 9. Kapitels.

Verwandte Befehle: SYMBOL AFTER

SYMBOL AFTER

SYMBOL AFTER ‹ganzzahliger Ausdruck›

```
10 CLS
20 SYMBOL AFTER 115
30 PRINT "Zeile 40 definiert das s um ";
40 SYMBOL 115,0,56,64,64,48,8,8,112
50 PRINT "in s"
60 PRINT "mache diese Definition von s rueckgaengig,"
70 PRINT "indem du schreibst: SYMBOL AFTER 240"
run
```

KOMMANDO: Legt fest, wieviele Zeichen (0 bis 256) selbstdefiniert werden können. Der Standardwert ist 240, d.h. 16 Zeichen kann der Benutzer selbst definieren (von 240 bis 255). Lautet der ‹ganzzahlige Ausdruck› 32, können alle Zeichen von 32 bis 255 neu definiert werden. Folglich können nach dem Befehl SYMBOL AFTER 256 keine neuen Zeichen definiert werden.

Sobald der Befehl SYMBOL AFTER erscheint, erhalten alle selbstdefinierten Zeichen ihre ursprüngliche Bedeutung wieder.

Der Befehl funktioniert nicht (es sei denn, in der Form SYMBOL AFTER 256), wenn der Wert von HIMEM zuvor durch den MEMORY-Befehl oder durch Eröffnung eines Dateipuffers mit OPENIN oder OPENOUT geändert wurde. In diesem Fall erscheint Fehlermeldung 5 (Improper argument).

Weitere Informationen über selbstdefinierte Zeichen finden Sie im 2.Teil des 9. Kapitels.

Verwandte Befehle: HIMEM, MEMORY, SYMBOL

TAB

(Siehe PRINT TAB)

TAG

TAG [#‹Stream›]

```
10 INPUT "Gib deinen Namen an";a$:CLS
20 PRINT "Du kommst gewiss viel herum, ";a$
30 TAG
40 x=LEN(a$)*17:y=50+RND*300:MOVE -x,y
50 FOR f=-x TO 640 STEP RND*7+3
60 MOVE f,y:PRINT " ";a$;:FRAME:NEXT
70 FOR b=640 TO -x STEP-RND*7+3
80 MOVE b,y:PRINT a$;" ";:FRAME:NEXT
90 GOTO 40
run
```

KOMMANDO: Der dem ‹Stream› zugeordnete Text erscheint immer dort, wo sich der Graphik-Cursor befindet. Auf diese Weise können Text und Symbole mit der Graphik gemischt werden oder Bildpunkt für Bildpunkt statt in Abständen von ganzen Zeichen bewegt werden. Der Standardwert für ‹Stream› ist 0, wenn kein Parameter angegeben wird.

Die linke obere Ecke des Textzeichens liegt auf dem Graphik-Cursor. Nicht abdruckbare Kontrollzeichen, wie z.B. Wagenrücklauf und Zeilenvorschub werden als Sonderzeichen ausgegeben, wenn hinter dem PRINT-Kommando kein ; steht.

Ist der ‹Stream›-Parameter 0, wird TAG abgeschaltet, sobald BASIC zum Direktmodus zurückkehrt. (TAG steht für "Text At Graphics".)

Verwandte Befehle: TAGOFF

TAGOFF

TAGOFF [#‹STREAM›]

```
10 MODE 2:TAG :REM Text At Graphics-on
20 year=1984:FOR x=1 TO 640 STEP 70
30 MOVE x,400:DRAWR 0,-350
40 year=year+1:PRINT year;:NEXT
50 TAGOFF :REM Text At Graphics-OFF
60 LOCATE 34,25:PRINT "Jahreszahlen"
70 GOTO 70
run
```

KOMMANDO: Macht den TAG-Befehl für den angegebenen ‹Stream› (Stream #0, wenn nichts anderes angegeben wurde) rückgängig und setzt den Text wieder dorthin, wo sich der Text-Cursor vor dem TAG-Befehl befand.

Verwandte Befehle: TAG

TAN

TAN (‹numerischer Ausdruck›)

```
PRINT TAN(45)
 1.61977519
```

FUNKTION: Berechnet den TANgens des ‹numerischen Ausdrucks›, der zwischen –200000 und +200000 liegen muß.

Beachten Sie, daß das Ergebnis der obigen Berechnung mit Hilfe der Befehle DEG und RAD im Winkel- bzw. Bogenmaß ausgedrückt werden kann.

Verwandte Befehle: ATN, COS, DEG, RAD, SIN

TEST

TEST (‹x-Koordinate›,‹y-Koordinate›)

```
10 CLS
20 PRINT "Sie benutzen PEN-Nummer";
30 PRINT TEST(10,386)
40 PRINT "Aendern Sie PENs und MODEs";
50 PRINT "....dann wieder RUN."
run
```

FUNKTION: Bewegt den Graphik-Cursor an den angegebenen absoluten x/y-Koordinatenpunkt und gibt die Nummer des Farbstifts an, der an dieser Position verwendet wurde.

Verwandte Befehle: MOVE, MOVER, TESTR, XPOS, YPOS

TESTR

TESTR (‹x-Versatz›,‹y-Versatz›)

```
10 MODE 0:FOR x=1 TO 15:LOCATE 1,x
20 PEN x:PRINT STRING$(10,143);:NEXT
30 MOVE 200,400:PEN 1
40 FOR n=1 TO 23:LOCATE 12,n
50 PRINT "Schrift";TESTR(0,-16):NEXT
run
```

FUNKTION: Verschiebt den Graphik-Cursor von seiner gegenwärtigen Position um die angegebenen x- und y-Werte und gibt die Nummer des Farbstifts an, der an dieser Position verwendet wurde.

Verwandte Befehle: MOVE, MOVER, TEST, XPOS, YPOS

THEN

(Siehe IF)

TIME

TIME

```
10 CLS:REM Uhr
20 INPUT "Stunde";Stunde
30 INPUT "Minute";Minute
40 INPUT "Sekunde";Sekunde
50 CLS:Uhrzeit=INT(TIME/300)
60 WHILE Stunde<13
70 WHILE Minute<60
80 WHILE Tick<60
90 Tick=(INT(TIME/300)-Uhrzeit)+Sekunde
100 LOCATE 1,1
110 PRINT USING "## ";Stunde,Minute,Tick
120 WEND
130 Tick=0:Sekunde=0:Minute=Minute+1
140 GOTO 50
150 WEND
160 Minute=0:Stunde=Stunde+1
170 WEND
180 Stunde=1
190 GOTO 60
run
```

FUNKTION: Gibt die Zeit (engl. time) an, die seit dem Einschalten des Computers bzw. seit dem letzten Zurücksetzen vergangen ist. Dabei wird die Zeit für das Speichern auf Diskette bzw. Laden von Diskette abgezogen.

Eine Einheit entspricht einer Dreihundertstelsekunde.

Verwandte Befehle: AFTER, EVERY, WEND, WHILE

TO

(Siehe FOR)

TROFF
TRON

```
TROFF
TRON
```

```
10 TROFF:PRINT:PRINT "TRace-OFF"
20 FOR n=1 TO 8
30 PRINT "Programm laeuft":NEXT
40 IF f=1 THEN END
50 TRON:PRINT:PRINT "TRace-ON"
60 f=1:GOTO 20
run
```

KOMMANDO: Mit TRON (TRace ON) kann die Ausführung eines Programms genau verfolgt (engl. trace) werden, da vor jeder Zeile die zugehörige Programmzeile in eckigen Klammern [] erscheint. Mit TROFF (TRace OFF) wird das Kommando TRON aufgehoben.

TRON ist vor allem dann hilfreich, wenn bei der Ausführung eines Programms ein Fehler auftritt und man verfolgen möchte, wie das Programm vor dem Fehler ablief.

Verwandte Befehle: **keine**

UNT

UNT (‹Adressausdruck›)

```
PRINT UNT(&FF66)
-154
```

KOMMANDO: Gibt eine ganze Zahl (engl. inTeger) zwischen –32768 und +32767 wieder, die das entsprechende Zweierkomplement des vorzeichenlosen (engl. UNsigned) ‹Adressen›-Wertes darstellt.

Verwandte Befehle: CINT, FIX, INT, ROUND

UPPER$

UPPER$ (‹String›)

```
10 CLS:a$="Wie gross du geworden bist!"
20 PRINT UPPER$(a$)
run
```

FUNKTION: Alle Buchstaben des angegebenen Strings werden in Großschreibung wiedergegeben. Ein nützliches Hilfsmittel, um Eingaben auszuwerten, die in gemischter Schreibweise eingegeben wurden.

Verwandte Befehle: LOWER$

USING

(Siehe PRINT USING)

VAL

VAL (‹String›)

```
10 CLS:PRINT "Ich kenne das kleine Einmaleins!"
20 PRINT:PRINT "Druecke eine Taste (1 bis 9)"
30 a$=INKEY$:IF a$="" THEN 30
40 n=VAL(a$):IF n<1 OR n>9 THEN 30
50 FOR x=1 TO 10
60 PRINT x;"X";n;"=";x*n
70 NEXT:GOTO 20
run
```

FUNKTION: Gibt den numerischen Wert (engl. VALue), einschließlich negativem Vorzeichen und Dezimalpunkt, des Zeichens wieder, das am Anfang des ‹Strings› steht.

Ist das erste Zeichen im String keine Zahl, wird 0 ausgegeben. Ist das erste Zeichen ein negatives Vorzeichen oder ein Dezimalpunkt, ohne daß eine Zahl folgt, erscheint die Fehlermeldung 13: 'Type mismatch'.

Verwandte Befehle: STR$

VPOS

VPOS (#‹Stream›)

```
10 MODE 1:BORDER 0:LOCATE 8,2
20 PRINT "benutze die Cursor-auf/ab Tasten"
30 WINDOW 39,39,1,25:CURSOR 1,1
40 LOCATE 1,13
50 IF INKEY(0)<>-1 THEN PRINT CHR$(11);
60 IF INKEY(2)<>-1 THEN PRINT CHR$(10);
70 LOCATE #1,3,24
80 PRINT #1,"Text-Cursor ";
90 PRINT #1,"vertikale Position =";
100 PRINT #1,VPOS(#0):GOTO 50
run
```

FUNKTION: Gibt an, auf welchem Punkt der y-Achse, bezogen auf den oberen Rand des Bildschirmbereichs, sich der Text-Cursor gegenwärtig befindet. Der ‹Stream›-Parameter muß unbedingt angegeben werden, da sonst nicht automatisch Stream #0 angenommen wird. (VPOS steht für Vertical POSition)

Verwandte Befehle: POS, WINDOW

WAIT

WAIT ‹Interfaceadresse› , ‹Maske›[, ‹Inversion›]

```
WAIT &FF34,20,25
```

KOMMANDO: Wartet, bis über das angesprochene Interface (Ein-/Ausgabeschnittstelle) ein bestimmter Wert zwischen 0 und 255 eingelesen ist. Der eingelesene Wert wird mit dem ‹Inversions›-Wert entsprechend XOR (exklusives ODER) verglichen und, wenn das Ergebnis 0 ist, noch mit der Maske entsprechend AND, bis das Ergebnis nicht 0 ist.

BASIC wartet solange, bis der geforderte Zustand hergestellt ist.

Dieser Befehl sollte mit Vorsicht angewendet werden.

Verwandte Befehle: INP, OUT

WEND

WEND

```
WEND
```

KOMMANDO: Schließt eine WHILE-Schleife ab. WEND findet automatisch den zugehörigen WHILE-Befehl.

Verwandte Befehle: TIME, WHILE

WHILE

WHILE ‹logischer Ausdruck›

```
10 CLS:PRINT "Zehn-Sekunden-Stoppuhr":t=TIME
20 WHILE TIME<t+3000
30 SOUND 1,0,100,15
40 WEND:SOUND 129,40,30,15
run
```

KOMMANDO: Wiederholt einen Programmteil solange, wie der ‹logische Ausdruck› zutrifft. WHILE bildet den Anfang der Schleife und definiert die Bedingung.

Verwandte Befehle: TIME, WEND

WIDTH

WIDTH ‹ganzzahliger Ausdruck›

```
WIDTH 40
```

KOMMANDO: Gibt die Anzahl der Zeichen pro Zeile für die Druckerausgabe an. BASIC fügt bei längeren Zeilen automatisch einen Wagenrücklauf und Vorschub auf eine neue Zeile ein.

Wird WIDTH (dt. Breite) nicht angegeben, geht der Computer von einem Standardwert von 132 aus.

Beim Befehl WIDTH 255 wird der Zeilenabschluß allein vom Drucker bestimmt. Wagenrücklauf und Zeilenvorschub können in diesem Fall jedoch durch einen PRINT-Befehl ohne Semikolon oder Komma bewirkt werden.

Verwandte Befehle: POS

WINDOW

`WINDOW` [#‹Stream›,]‹links›,‹rechts›,‹oben›,‹unten›

```
10 MODE 0:BORDER 0:REM Testkarte
20 INK 0,0:INK 1,25:INK 2,23:INK 3,21
30 INK 4,17:INK 5,6:INK 6,2:INK 7,26
40 PAPER 0:CLS
50 PAPER 1:WINDOW 2,4,1,18:CLS
60 PAPER 2:WINDOW 5,7,1,18:CLS
70 PAPER 3:WINDOW 8,10,1,18:CLS
80 PAPER 4:WINDOW 11,13,1,18:CLS
90 PAPER 5:WINDOW 14,16,1,18:CLS
100 PAPER 6:WINDOW 17,19,1,18:CLS
110 PAPER 7:WINDOW 2,19,19,25:CLS
120 GOTO 120
run
```

KOMMANDO: Definiert Größe und Lage eines Bildschirmbereichs (Window). Die Werte für die Parameter ‹links›, ‹rechts›, ‹oben› und ‹unten› sollten so gewählt werden, daß sie zum verwendeten Modus passen.

Wird der Stream nicht angegeben, so wird Stream `#0` (der ganze Bildschirm) angenommen.

Weitere Informationen zu Bildschirmfenstern finden Sie im 2. Teil des 9. Kapitels.

Verwandte Befehle: `WINDOW SWAP`

WINDOW SWAP

`WINDOW SWAP` ‹Stream›,‹Stream›

```
10 MODE 1:INK 1,24:INK 2,9:INK 3,6
20 WINDOW 21,40,13,25:PAPER 3
30 WINDOW #1,1,20,1,12:PAPER #1,2
40 CLS:PRINT " Fenster Nummer 0"
50 CLS #1:PRINT #1," Fenster Nummer 1"
60 LOCATE 1,6
70 PRINT " rotes Fenster (0)";SPC(2)
80 LOCATE #1,1,6
90 PRINT #1," gruenes Fenster (1)"
100 FOR t=1 TO 1000:NEXT
110 WINDOW SWAP 0,1:GOTO 60
run
```

KOMMANDO: Tauscht (engl. swap) die in den beiden ‹Streams› definierten Bildschirmbereiche gegeneinander aus..

Die Nummern der beiden Streams müssen angegeben werden, jedoch sollte kein #-Zeichen vorangestellt werden.

Der Befehl kann verwendet werden, um vom BASIC ausgegebene Meldungen, die normalerweise immer zu Stream #Ø geschickt werden, umzuleiten.

Weitere Informationen über 'Windows' finden Sie im 2.Teil des 9.Kapitels.

Verwandte Befehle: WINDOW

WRITE

WRITE [#‹Stream›,][‹Ausgabeliste›]

```
10 REM Variablen auf Diskette schreiben
20 INPUT "Gib eine Zahlenvariable an";a
30 INPUT "Gib eine Textvariable an";a$
40 OPENOUT "Datei"
50 WRITE #9,a,a$
60 CLOSEOUT:PRINT "Daten auf Diskette gespeichert"
run
```

KOMMANDO: Schreibt (engl. write) die in der ‹Ausgabeliste› aufgeführten Werte auf den angegebenen ‹Stream›. Die einzelnen Werte werden durch Kommata getrennt, Strings werden in Doppelanführungszeichen eingeschlossen.

Im obigen Beispiel werden die Werte der Variablen, die Sie eingegeben haben, auf Stream #9, also auf Diskette, geschrieben.

Um die Werte wieder von der Diskette zurückzuholen, müßte ein Programm wie das nachstehende eingesetzt werden:

```
10 REM Variablen von Diskette abrufen
20 OPENIN "Datei":INPUT #9,a,a$
30 CLOSEIN:PRINT "Die beiden Werte heissen:"
40 PRINT:PRINT a,a$
run
```

Verwandte Befehle: INPUT, LINE INPUT

XOR

‹Aussage› XOR ‹Aussage›

```
IF "Anna"<"Berta" XOR "Hund">"Katze" THEN PRINT "richt
ig" ELSE PRINT "falsch"
falsch

IF "Berta"<"Anna" XOR "Katze">"Hund" THEN PRINT "richt
ig" ELSE PRINT "falsch"
falsch

IF "Anna"<"Berta" XOR "Katze">"Hund" THEN PRINT "richt
ig" ELSE PRINT "falsch"
richtig

.............

PRINT 1 XOR 1
Ø
PRINT Ø XOR Ø
Ø
PRINT 1 XOR Ø
1
```

OPERATOR: Führt Bit für Bit Boolsche Operationen mit ganzen Zahlen durch. Das Ergebnis ist 1, es sei denn, beide Argumente haben denselben Wert.

Weitere Informationen über Logik finden Sie im 2.Teil des 9. Kapitels.

Verwandte Befehle: AND, OR, NOT

XPOS

XPOS

```
1Ø MODE 1:DRAW 32Ø,2ØØ
2Ø PRINT "x-Koordinate des Graphik-Cursors =";
3Ø PRINT XPOS
run
```

FUNKTION: Gibt die x-Achse an, auf der sich der Graphik-Cursor gerade befindet.

Verwandte Befehle: MOVE, MOVER, ORIGIN, YPOS

YPOS

YPOS

```
10 MODE 1:DRAW 320,200
20 PRINT "y-Koordinate des Graphik-Cursors =";
30 PRINT YPOS
run
```

FUNKTION: Gibt die y-Achse an, auf der sich der Graphik-Cursor gerade befindet.

Verwandte Befehle: MOVE, MOVER, ORIGIN, XPOS

ZONE

ZONE ‹ganzzahliger Ausdruck›

```
10 CLS
20 FOR z=2 TO 20
30 ZONE z
40 PRINT "X","X zone =";z
50 NEXT
run
```

KOMMANDO: Ändert die Breite der Druckzonen, die für die einzelnen, durch Kommata getrennten Ausdrücke aus dem Print-Befehl bereitstehen. Die Breite der Druckzonen kann zwischen 1 und 255 Stellen gewählt werden. Die Standardbreite beträgt 13 Stellen.

Verwandte Befehle: PRINT

Kapitel 4
Das Arbeiten mit Disketten und Kassetten

Teil 1: Disketten

Das Anfertigen von Arbeitsdisketten

Dieser Teil beschreibt die Anfertigung von Arbeits-Disketten und zeigt einige Möglichkeiten von CP/M und seinen Dienstprogrammen auf.

Folgende Themen werden behandelt:

- ★ Kopieren der Systemdisketten
- ★ Start mit CP/M Plus
- ★ Das HELP-Programm
- ★ Betrieb mit einem oder zwei Laufwerken
- ★ Kopieren von Dateien mit PIP
- ★ Disketten mit reinem BASIC-Inhalt
- ★ Schlüsselfertige AMSTRAD BASIC-Software
- ★ Installation von schlüsselfertiger CP/M Plus-Software
- ★ Start mit GSX
- ★ Betrieb mit CP/M 2.2

In Teil 7 des Grundlagenkurses wurde beschrieben, wie eine leere System-Diskette formatiert wird, die sowohl für BASIC und Spiele als auch für CP/M verwendbar ist.

In Teil 10 des Grundlagenkurses wurde gezeigt, wie mit dem `DISCKIT3`-Programm (auf Seite 1 Ihres Systemdiskettensatzes) originalgetreue Kopien von Disketten gemacht werden können.

Dieser Teil beschreibt die Anfertigung von Disketten für Ihre Programme.

Reservekopien von Systemdisketten

Wir empfehlen Ihnen dringend, von den mitgelieferten Systemdisketten bzw. Dienstprogramm-Disketten eine Kopie anzufertigen und die Originale an einem sicheren Ort zu verwahren. Es wäre nämlich sehr kostspielig, eine beschädigte Diskette zu ersetzen! Jede Diskette hat zwei Seiten; Ihr Systemdiskettensatz besteht also aus vier Seiten. Übrigens ist jede Diskette beidseitig bespielbar.

Seite 1 ist am wichtigsten. Sie enthält das Original-CP/M Plus-Programm und einige Dienstprogramme zur Handhabung der Disketten. Seite 2 enhält Dateien für Assembler-Programmierer. Auf Seite 3 finden Sie Dr. LOGO, HELP-Programme und GSX (darauf kommen wir später noch zu sprechen). Seite 4 beinhaltet die CP/M 2.2- und Dr. LOGO-Programme, die für die Vorgängermodelle CPC664 und CPC464 + DDI1 konzipiert waren. Wir haben diese Programme aus Kompatibilitätsgründen mitgeliefert, für den Fall, daß Sie sie brauchen. Für den CPC6128 werden sie normalerweise nicht benötigt.

Die Kopien Ihrer Systemdisketten können Sie als Programm-Bibliothek betrachten. Normalerweise werden Sie das benötigte Programm wählen, indem Sie die Bibliotheks-Diskette, auf der es sich befindet, einschieben. Dies ist einfacher, als wenn Sie das Programm erst auf eine Leerdiskette übertragen und von dort laufen lassen.

Um es noch einmal in aller Deutlichkeit zu sagen: Die Bibliotheks-Disketten, mit denen Sie arbeiten, müssen KOPIEN der mitgelieferten Systemdisketten sein!

Wenn Sie auf eine neue, unbespielte Diskette kopieren wollen, denken Sie daran, daß das `DISCKIT3`-Programm (auf Seite 1) sowohl zum Formatieren als auch zum Kopieren dient.

Start mit CP/M Plus

Sie werden sich daran gewöhnt haben, daß der Computer auf AMSTRAD BASIC eingestellt ist, wenn Sie ihn eingeschaltet haben. Er arbeitet solange mit BASIC, bis ein AMSDOS- (bzw. Kassetten-) Binär-Programm die Kontrolle übernimmt oder CP/M Plus mit dem Befehl `|cpm` geladen wird.

Sobald CP/M Plus geladen wurde, braucht der CPC6128 die Seite 1 nicht mehr, es sei denn, Sie wollen eines der ebenfalls darauf enthaltenen Dienstprogramme laufen lassen. Nur die Startdiskette muß eine Systemdiskette sein, alle anderen Disketten können reine Datendisketten sein, die eine größere Speicherkapazität besitzen.

Ein Programm läßt man ganz einfach ablaufen, indem man die Diskette, auf der es sich befindet, ins Laufwerk schiebt, und den Programm-Namen eintippt. Die Daten, die das Programm verwendet, können entweder auf derselben, oder auf einer anderen Diskette stehen. Unter CP/M Plus kann der Benutzer die Disketten genauso auswechseln wie unter AMSDOS. Wenn Sie der Einfachheit halber mehrere Programme, eventuell mit ein paar Dienstprogrammen, auf derselben Diskette speichern wollen, benutzen Sie das PIP-Programm von Seite 1. Wir kommen darauf noch an anderer Stelle in diesem Kapitel und in Kapitel 5 zurück.

Das Profile-Programm

Auf der Systemdiskette gibt es ein Programm namens PROFILE.SUB. Es enthält eine Liste der Befehle, die automatisch ausgeführt werden, nachdem CP/M Plus gestartet wird. Sie können dann - sofern Sie es noch nicht getan haben - eine Kopie von Seite 1 Ihrer Systemdisketten einschieben. Sobald das Bereitschaftszeichen A> erscheint, schreiben Sie:

```
REN PROFILE.SUB=PROFILE.ENG
```

Dadurch wird das Programm PROFILE.ENG in PROFILE.SUB umbenannt. Dieses Programm wird verwendet, wenn CP/M das nächste Mal gestartet wird. Es enthält die Befehle:

```
SETKEYS KEYS.CCP
LANGUAGE 2
```

Damit werden die Cursor-Tasten auf CP/M-Funktion umgestellt, und der Zeichensatz wird von amerikanisch auf deutsch umgeschaltet, so daß z.B. anstelle der eckigen Klammer [ein Ä ausgegeben wird. Wenn die Tastatur mit dem Befehl SETKEYS KEYS.CCP eingestellt wurde, können die CP/M Befehlszeilen auf ähnliche Weise editiert werden wie BASIC-Zeilen. Eine ausführliche Beschreibung von SETKEYS finden Sie im 2. Teil des 5. Kapitels.

Das Help-Programm

Seite 3 des System-Diskettensatzes bietet spezielle Hilfestellung in Form des HELP-Programms. Es ist sozusagen ein elektronisches Bedienungshandbuch für die CP/M Plus-Dienstprogramme. Um diese Funktion einzuschalten, legen Sie Seite 3 ins Laufwerk und schreiben, sobald Sie das Bereitschaftszeichen A> sehen:

```
HELP
```

Das HELP-Programm stellt Ihnen Fragen, die Sie beantworten, so daß Sie schließlich die gewünschte Information erhalten.

Ein Laufwerk oder zwei?

Wenn CP/M geladen wird, stellt es die Anzahl der angeschlossenen Diskettenlaufwerke fest und zeigt sie in der Eröffnungsmeldung an. Beachten Sie, daß die Angabe falsch sein kann, wenn eine Diskette im zweiten Laufwerk nicht vollständig eingeschoben ist.

Alle Fehlermeldungen, die sich auf den Diskettenmechanismus beziehen, erscheinen automatisch in der 25. Zeile des Bildschirms. Die Programme selbst nehmen nur die ersten 24 Zeilen ein.

Wenn Sie nur mit dem eingebauten Diskettenlaufwerk arbeiten, sehen Sie auf der untersten Zeile die Meldung '`Drive is A:`' oder '`Drive ist B:`'. Dies bedeutet, daß CP/M Plus Ihnen gestattet, ein Laufwerk so zu benutzen, als hätten Sie zwei angeschlossen. Sie können dann abwechslend mit zwei Disketten arbeiten, wobei in der untersten Zeile angesagt wird, welche Diskette das Programm jeweils braucht. Mit diesem Operationsmodus können Sie auf ein weiteres Diskettenlaufwerk verzichten; allerdings macht er oftmals ein häufiges Auswechseln der Disketten erforderlich, was erstens zeitraubend ist und zweitens die Gefahr von Irrtümern erhöht.

Kopieren von Diskette auf Diskette mit PIP

Ein Standard-Dienstprogramm namens `PIP` (Peripheral Interchange Program) ermöglicht es, Dateien von einer Diskette auf eine andere zu kopieren.

Dazu laden Sie zunächst das PIP-Programm von Seite 1, indem Sie nach dem `A>` Zeichen

```
PIP
```

...eintippen. Ein neues Bereitschaftszeichen `*` zeigt an, daß `PIP` korrekt geladen wurde. Normalerweise kopieren Sie Dateien von einer Quellendiskette (in Laufwerk `A:`) auf eine Zieldiskette (in Laufwerk `B:`) Wie wir schon gesehen haben, ist beim Betrieb ohne zusätzliches Laufwerk das eingebaute Diskettenlaufwerk zugleich Drive `A:` und Drive `B:`.

Um eine Datei, z.B. `SUBMIT.COM`, zu kopieren, schreiben Sie nach dem `*` Sternchen:

```
B:=A:SUBMIT.COM
```

...und wenn Sie alle Dateien einer Diskette auf eine andere kopieren möchten, benutzen Sie den Befehl:

```
B:=*.*
```

Um PIP zu verlassen, drücken Sie nach dem Bereitschaftszeichen * die **[RETURN]**-Taste.

PIP ist ein sehr vielseitiges Programm, wie Sie im 5. Kapitel noch erfahren werden.

Disketten mit reinem BASIC-Inhalt

Wie wir bereits ausgeführt haben, dient eine Systemdiskette normalerweise nur dazu, auf CP/M-Betrieb umzuschalten. BASIC-Disketten können daher reine Datendisketten sein, die eine etwas größere Kapazität haben.

Die Diskette muß mit dem DISCKIT3-Programm formatiert werden. Um Programme auf diesen Diskettentyp zu kopieren, müssen Sie sich entweder des PIP-Programms von Seite 1 bedienen, oder mit den BASIC-Befehlen LOAD und SAVE arbeiten.

Schlüsselfertige AMSTRAD BASIC-Disketten

Wenn Sie ein für den CPC6128 geschriebenes Anwenderprogramm in AMSTRAD BASIC kaufen, ist es in der Regel beim Einschalten betriebsbereit. Es ist sozusagen 'schlüsselfertig'. (Der Begriff stammt aus der Zeit, als Kleincomputer noch mit abschließbaren Netzschaltern ausgerüstet waren.) Ähnlich wie bei den beiden zum CPC6128 gehörenden Systemdisketten ist es dringend zu empfehlen, daß Sie nur mit einer Kopie arbeiten und das Original sicher verwahren.

Schlüsselfertige CP/M-Disketten

Durch das CP/M-Betriebssystem können Sie auf Ihrem Computer eine umfangreiche Software-Bibliothek einsetzen, die für Personal-Computer mit CP/M geschrieben wurde. Diese Programme tragen die Grundvoraussetzungen für ihren Betrieb bereits in sich. Sie müssen jedoch möglicherweise auf eine geeignete Diskette überspielt und über die besondere Methode informiert werden, mit welcher der 6128 den Bildschirm ansteuert.

Die Programme einer Diskette, die einem bestimmten Anwendungszweck dienen, nennt man 'Programmpaket'. Diese Pakete sind nomalerweise so konzipiert, daß sie auf vielen verschiedenen Computern mit jeweils unterschiedlicher Bildschirmgröße und unterschiedlichen Cursor-Techniken eingesetzt werden können.

Der 6128 besitzt eine eingebaute 'Bildschirmanpassung' für CP/M-Programme, wobei sich die Steuercodes von denen des BASIC unterscheiden.

Manchmal sind die käuflichen Pakete bereits für den CPC6128 installiert, bzw. bieten in ihrem Menü eine passende Installation für ihn an. In diesem Fall befolgen Sie einfach die Instruktionen für einen Zenith Z19/Z29. Hat das Paket kein Installationsmenü oder keine spezielle CPC6128-Variante, finden Sie im übernächsten Abschnitt 'Konfigurieren eines CP/M-Programms' einige der Befehle, mit denen die für das Paket erforderlichen Bildschirmbedingungen erzeugt werden. Normalerweise müssen bei der Installation oder kundenspezifischen Anpassung die entsprechenden Codes angegeben werden. Folgen Sie dabei den Anweisungen auf dem Software-Paket.

Die erworbene Software muß sich auf einer für den CPC6128 geeigneten Diskette befinden. Fast jeder Computer-Hersteller verwendet eigene Disketten. Auch dasselbe Disketten-Format bedeutet noch nicht, daß die auf ihnen gespeicherten Informationen kompatibel sind. Fragen Sie Ihren Händler daher nach 3 Zoll-AMSTRAD-Disketten.

Erstellung einer schlüsselfertigen CP/M-Systemdiskette

Es ist häufig nützlich, neben dem Anwender-Programm selbst die Dienstprogramme `SETKEYS.COM` und eventuell `SUBMIT.COM` mit den zugehörigen Instruktionsdateien auf einer schlüsselfertigen CP/M-Diskette zu haben.

`PIP` kann verwendet werden, die `.COM`-Dateien zu übertragen und die Instruktionsdatei für `SUBMIT` herzustellen. In diesem letzteren Modus stellt PIP eigentlich nur eine Änderungsfunktion, mit der eine Zeile geändert werden kann. Die Datei `LOG03.SUB` auf Seite 3 z.B. hätte mit folgendem Verfahren hergestellt werden können:

Sie legen Seite 1 der System-Disketten ins Laufwerk `A:` und schreiben:

```
PIP
```

Dann nehmen Sie die System-Diskette heraus und legen die Zieldiskette ein. Nun schreiben Sie:

```
LOGO3.SUB=CON:
SETKEYS KEYS.DRL
[CONTROL]J LOGO3
[CONTORL]Z
```

Konfigurieren eines CP/M-Programms

Der CPC6128 unterstützt eine große Anzahl von Steuercodes, die der kundenspezifischen Anpassung eines Software-Pakets an den CP/M-Betrieb dient. Bei den meisten Datenverarbeitungs- und sonstigen Software-Paketen sollen Meldungen auf irgendeinen Bildschirmbereich ausgegeben, oder von irgendeinem Bildschirmbereich eingegeben werden. Sie arbeiten in der Regel mit Cursor-Steuerung.

Ist Ihr Paket schon an das Schneider-System angepaßt, so brauchen Sie sich nicht weiter damit zu befassen.

Konfigurieren der Ausgabe

Das Installationsverfahren für ein Paket läuft normalerweise über ein bestimmtes Programm, das oft `INSTAL` genannt wird. Wenn es kein Terminal vom Typ Z19/Z29 oder den CPC6128 unterstützt, stellt es einige Fragen über die Parameter des CPC6128-Bildschirms. Die Antworten können an der folgenden Tabelle abgelesen werden, die einen Auszug aus dem 15.Teil von Kapitel 7 darstellt.

Kontroll-Codes	Hex-Wert		Dezimal-wert		Funktion
[BEL]		&07		7	Pieps-Ton
[BS]		&08		8	Bewegt den Cursor eine Stelle zurück
[LF]		&0A		10	Bewegt den Cursor eine Zeile abwärts
[CR]		&0D		13	Bewegt den Cursor zum linken Bildschirmrand
[ESC]A	&1B	&41	27	65	Bewegt den Cursor eine Zeile aufwärts
[ESC]C	&1B	&43	27	67	Bewegt den Cursor eine Stelle vor
[ESC]E	&1B	&45	27	69	Löscht den Bildschirm
[ESC]H	&1B	&48	27	72	Bringt den Cursor in die linke obere Bildschirmecke
[ESC]J	&1B	&4A	27	74	Löscht alles von der momentanen Cursorposition bis zum Ende des Bildschirms

[ESC]K	&1B	&4B	27	75	Löscht alles von der momentanen Cursorposition bis zum rechten Bildschirmrand
[ESC]L	&1B	&4C	27	76	Fügt eine Zeile ein
[ESC]M	&1B	&4D	27	77	Löscht die gesamte Zeile
[ESC]N	&1B	&4E	27	78	Löscht den Buchstaben, auf dem sich der Cursor befindet
[ESC]Y	&1B	&59	27	89	Bewegt den Cursor zur angegebenen
	‹c›	‹r›	‹c›	‹r›	Stelle. ‹c› steht für Spaltennummer +32, ‹r› für Zeilennummer +32.
[ESC]d	&1B	&64	27	100	Löscht alles vom Bildschirmanfang bis zur gegenwärtigen Cursorposition einschließlich
[ESC]o	&1B	&6F	27	111	Löscht alles von linken Bildschirmrand bis zur Cursorposition einschließlich
[ESC]p	&1B	&70	27	112	Beginn der invertierten Darstellung
[ESC]q	&1B	&71	27	113	Ende der invertierten Darstellung

Konfigurieren der Eingabe

Die Programme des Software-Pakets können unter Umständen die Tastatur abfragen. Mit Ausnahme der Cursor-Tasten geben die Tasten des 6128 Standardwerte wieder. Mit dem SETKEYS-Dienstprogramm können die Codenummern umdefiniert werden. Allerdings ist es möglichst vorzuziehen, wenn alle unterschiedlichen Pakete so konfiguriert werden, daß sie Standardwerte annehmen.

Bedauerlicherweise gibt es für die verschiedenen Software-Programme keine einheitlichen Regeln dafür, durch welche Tasten die einzelnen Steuerfunktionen ausgelöst werden. Sichtbare Zeichen, sowie die Leertaste, **[TAB]** und **[RETURN]**, sind weitgehend genormt. Aber schon bei der Rückwärtstaste hört die Einheitlichkeit auf, und von da ab wird es nur schlimmer! Vergleichen Sie nur, wieviele verschiedene Codes es für die Operation 'Cursor zum Zeilenanfang bewegen' gibt:

Bei CP/M heißt der Befehl: **[CONTROL]B**
Bei Dr.LOGO drücken Sie: **[CONTROL]A**
Und bei einer typischen Textverarbeitung drücken Sie wahrscheinlich: **[CONTROL]QS**

Drei nützliche Tastencode-Sätze werden standardmäßig angeboten. Jede Konfiguration kann von Dateien auf Seite 1 Ihrer System-Disketten abgerufen werden:

```
SETKEYS KEYS.CCP
```

Dieser Befehl wurde schon als einer der Befehle beschrieben, die PROFILE.SUB automatisch ausgibt. Er stellt die Tastatur auf CP/M-Befehle ein.

Start eines schlüsselfertigen CP/M-Pakets

Normalerweise muß nach dem Erscheinen des Bereitschaftszeichens A> nur noch der Programm-Name des Software-Pakets eingegeben werden. Um beispielsweise ein Lohnprogramm mit dem Namen GEHALT.COM zu starten, schreiben Sie lediglich:

```
GEHALT
```

Wenn irgendwelche Konfigurationen festgelegt werden müssen, steht möglicherweise eine SUBMIT-Datei zur Verfügung. Ein Beispiel dafür bildet die Datei LOGO3.SUB auf Seite 3. Sie wird mit dem Befehl:

```
SUBMIT LOGO3
```

aufgerufen. Der Inhalt der Datei wird gezeigt, wenn Sie

```
TYPE LOGO3.SUB
```

schreiben, woraufhin Sie sehen:

```
SETKEYS KEYS.DRL    ...stellt die Tastatur ein
LOGO3               ...startet das Dr.LOGO-Programm
SETKEYS KEYS.CCP    ...versetzt die Tastatur in ihren alten Zustand
```

Autostart eines schlüsselfertigen CP/M-Pakets

Das Betriebssystem CP/M ist in der Lage, ein bestimmtes Programm mit der Start-up-System-Diskette automatisch zu starten. Dazu wird der Name des Programms am Ende der PROFILE.SUB-Datei auf dieser Diskette angehängt.

Start mit GSX

GSX ist die Abkürzung für Graphics System Extension (Graphisches Erweiterungssystem). Es gestattet einem CP/M-Programm, sowohl Graphik als auch Text auszugeben. Damit können z.B. Balken- und Kreisdiagramme gezeichnet, sowie Überschriften in verschiedenen Drucktypen und Größen dargestellt werden. Die Abbildung auf der nächsten Seite illustriert diese Möglichkeit. Mit GSX kann sowohl auf den Bildschirm, als auch auf einen Drucker oder einen Zeichenstift-Plotter ausgegeben werden.

Mit GSX selber können keine Zeichnungen angefertigt werden, genausowenig wie man CP/M zur Textverarbeitung benutzen kann. Dazu braucht man ein spezielles Anwenderprogramm. Die abgebildete Zeichnung wurde mit dem Digital Research-Programm 'DR Graph' erstellt. Mit GSX können jedoch Standard-Systemeinrichtungen für Bildschirme, Drucker und Plotter geschaffen werden, so daß die Anwenderprogramme mit minimalem Re-Installationsaufwand auf den verschiedenen Geräten laufen können.

Um von einer Diskette GSX-Programme laufen lassen zu können, kopieren Sie mit Hilfe von **PIP** die Dateien **GSX.SYS**, **ASSIGN.SYS**, sowie die erforderlichen Gerätesteuerprogramme und das Anwenderprogramm selber auf eine formatierte Leerdiskette. Die Datei **ASSIGN.SYS** enthält ein 'Rezept' für bis zu drei Ausgabe-Gerätesteuerprogramme. Sie heißen, vom kleinsten angefangen:

```
21a:ddfxlr7       ;Epson 7 Bit Drucker
11a:ddhp7470      ;Zeichenstift-Plotter
01a:ddmode2       ;Bildschirm in Modus 2
01a:ddmode1       ;Bildschirm in Modus 1
01a:ddmode0       ;Bildschirm in Modus 0
```

Die erste Zahl gibt an, für welches Gerät das Steuerprogramm ist - für Drucker, Plotter oder Bildschirm. In ein und denselben Speicherbereich kann nur ein Gerätesteuerprogramm zur gleichen Zeit geladen werden. GSX muß deshalb zuerst wissen, welches das größte ist, um ihm genug Platz reservieren zu können.

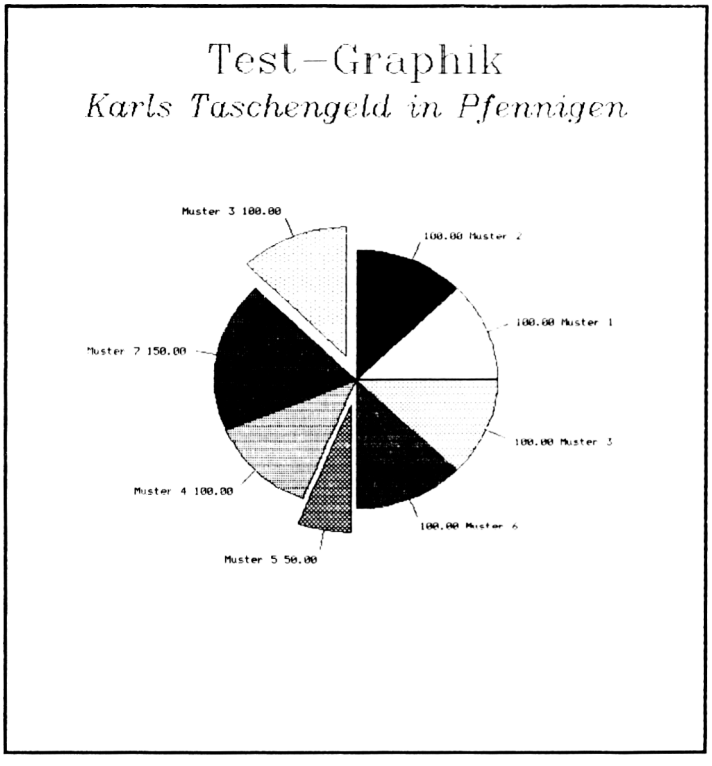
Für die verschiedenen Bildschirmmodi und für Standard-Drucker steht eine Auswahl von Steuerprogrammen zur Verfügung. Die Datei **DRIVERS.GSX** enthält eine Zusammenstellung der Steuerprogramme, die mit dem 6128 geliefert werden. Wenn Sie Seite 3 einlegen und nach Erscheinen des **A>** tippen:

```
TYPE DRIVERS.GSX
```

...können Sie die Datei lesen und das passende Steuerprogramm wählen. Die meisten Anwenderprogramme enthalten einen GSX-Lader, so daß Sie nach dem A› nur noch den Programmnamen einzutippen brauchen. Falls Ihr Anwender-Programm keinen eingebauten GSX-Lader hat, kopieren Sie die Datei **GENGRAF.COM** (von Seite 3) auf Ihre GSX-Arbeitsdiskette und schreiben:

```
GENGRAF YOURFILE
```

YOURFILE.COM ist die nicht-installierte Anwendung. Die Datei **GENGRAF.COM** kann jetzt mit **ERA** gelöscht werden, da **YOURFILE.COM** nun den GSX-Lader enthält.



Ein Beispiel für die Druckausgabe unter GSX

Der Betrieb mit CP/M 2.2

Im Gegensatz zu CP/M Plus hat CP/M 2.2 (der Vorläufer von CP/M Plus) sehr strenge Regeln für das Auswechseln der Disketten. Es erfordert ein häufiges Laden von Teilen des Betriebs-Systems von der Systemdiskette im Laufwerk A:, vor allem dann, wenn man nach dem Ablauf eines Programms zum Bereitschaftszeichen A› zurückkehrt. Deshalb arbeitet man normalerweise mit System-Disketten. Wenn Sie kein zweites Laufwerk haben, nehmen Sie anstelle von `PIP` das Einzellaufwerks-Kopierprogramm `FILECOPY`.

Wir empfehlen, CP/M 2.2 nur dann zu verwenden, wenn Sie bereits Software für die AMSTRAD-Modelle CPC664 oder CPC464 + DDI1 haben, die mit CP/M Plus nicht kompatibel ist.

> ACHTUNG! Ein Teil der CP/M 2.2-Software für den CPC664 und den CPC464 + DDI1 enthält spezifische Installations-Instruktionen und kann mit CP/M Plus nicht betrieben werden. In diesen Fällen muß mit CP/M 2.2 auf Seite 4 des System-Diskettensatzes gearbeitet werden.

Teil 2: Kassetten

Wenn Sie, wie im 1.Teil des Grundlagenkurses beschrieben, ein Kassettenlaufwerk an das System angeschlossen haben, schalten Sie den Computer mit dem Befehl `|TAPE` auf Kassettenbetrieb. Einige BASIC-Befehle werden dann an den Kassettenbetrieb angepaßt. Auf dem Bildschirm erscheinen verschiedene Software-Meldungen und Bereitschaftszeichen, die Sie beim normalen Diskettenbetrieb nicht sehen.

HINWEIS: Wenn die Tasten zum Vor- bzw. Rückspulen durch Fernsteuerung bedient werden, müssen Sie beim Spulen entweder den Stecker aus der `REM`-Buchse ziehen oder `|TAPE:CAT` eintippen, um den Motor des Kassettengerätes in Gang zu bringen. Mit **[ESC]** wird diese Funktion rückgängig gemacht.

Anders als bei der Diskettenspeicherung unterliegt die Form der Dateinamen für die Kassettenspeicherung nicht solch strengen Regeln. Sie können bis zu 16 Zeichen lang sein und dürfen Zwischenräume und Satzzeichen aufweisen. In manchen Fällen können sie auch völlig weggelassen werden.

Im folgenden ist aufgeführt, welche Befehle sich auf den Kassettenbetrieb einstellen, und welche Funktion sie dabei erfüllen. Die Beschreibung der Befehle finden Sie im Kapitel 3 'Liste aller Befehle des AMSTRAD CPC6128 BASIC'.

CAT

Sie werden angewiesen:

```
Press PLAY and then any key:
```

...woraufhin Sie die Taste **PLAY** auf Ihrem Kassettenlaufwerk und danach eine beliebige Taste auf dem Computer drücken. Das Band in der Kassette läuft an und der Computer führt die Namen aller auf der Kassette befindlichen Dateien auf, in der Reihenfolge, wie er sie vorfindet.

Jeder Block einer Datei wird angezeigt, gefolgt von einem Zeichen, das den Datei-Typ angibt:

`$`	für: ungeschützte BASIC-Datei
`%`	für: geschützte BASIC-Datei
`*`	für: ASCII-Datei
`&`	für: Binär-Datei

Am Ende der Zeile erscheint:

```
Ok
```

...um anzudeuten, daß der Lesevorgang erfolgreich war, und daß die Datei in den Computer geladen werden könnte.

Das gegenwärtig im Speicher befindliche Programm wird durch die CAT-Funktion nicht beeinträchtigt.

Eine namenlose Datei erscheint bei CAT unter der Bezeichnung:

```
Unnamed file
```

Durch Drücken der **[ESC]**-Taste wird die CAT-Funktion verlassen.

Lesefehler

Falls die oben erwähnten Meldungen beim Lesen der Datei nicht ausgegeben werden, oder die Mitteilung:

```
Read error a
```

...oder:

```
Read error b
```

auf dem Bildschirm erscheint, gibt es folgende Gründe dafür:

1. Das Kassettenlaufwerk wurde nicht richtig an die TAPE-Buchse angeschlossen (siehe Teil 2 des Grundlagenkurses).

2. Der **VOLUME**- oder **LEVEL**-Regler am Kassettengerät ist nicht richtig eingestellt.

3. Schlechte Bandqualität, bzw. beschädigtes Band

4. Das Band wurde einem Magnetfeld ausgesetzt, d.h. zu dicht neben einem Lautsprecher, Fernseher o.ä. gelagert.

5. Sie haben eine Kassette verwendet, die sich nicht für den Betrieb auf AMSTRAD Computer-Systemen eignet.

CHAIN
CHAIN MERGE
LOAD
MERGE
RUN

Wenn Sie die erste geeignete Datei auf der Kassette laden möchten, brauchen Sie ihren Namen nicht anzugeben. Beispiele:

```
CHAIN ""
CHAIN "",100

CHAIN MERGE ""
CHAIN MERGE "",100
CHAIN MERGE "",100,DELETE 30-70

LOAD ""
LOAD "",&1F40

MERGE ""

RUN ""
```

(Hinweis: Statt diesen Befehl einzutippen, können Sie auch die **[CONTROL]**-und **[ENTER]**-Tasten drücken, wenn Sie Kassetten-Software starten wollen und zuvor `|TAPE` angegeben haben.)

Folgende Aufforderung erscheint:

```
Press PLAY then any key:
```

...woraufhin Sie die **PLAY**-Taste an Ihrem Kassettengerät und danach eine beliebige Taste auf dem Computer drücken. Das Band in der Kassette fängt an zu laufen und der Computer lädt die Datei in seinen Speicher.

Auf dem Bildschirm erscheint die Meldung:

```
Loading DATEINAME block 1
```

Danach folgen Anzeigen mit den übrigen Blocknummern, bis die Datei geladen ist.

Beginnt der Dateiname mit einem Ausrufungszeichen !, so werden die oben genannten Meldungen unterdrückt, und Sie brauchen nicht eine beliebige Taste zu drücken, um den Ladevorgang einzuleiten. (Die PLAY-Taste muß jedoch gedrückt sein.) Möchten Sie Programme, die mit einem Ausrufungszeichen beginnen, auf Diskette laufen lassen, so wird das Ausrufungszeichen beim Laden von Diskette (d.h. beim Lesen des Dateinamens) ignoriert. Beachten Sie, daß das Ausrufungszeichen keine Stelle im Dateinamen einnimmt, weder bei der Kassetten- noch bei der Diskettenspeicherung.

Brechen Sie den Ladevorgang durch Drücken der **[ESC]**-Taste ab, erscheint folgende Fehlermeldung:

```
Broken in
```

Wenn das Laden nicht klappt, lesen Sie den vorigen Abschnitt 'Lesefehler'.

ACHTUNG! Das eingebaute Diskettenlaufwerk nimmt einen kleinen Teil des Speichers ein, der in manchen Fällen für Software benötigt wird, die für den CPC464 geschrieben wurde. Diese Kassetten funktionieren auf dem CPC6128 mit angeschlossenem Kassettenlaufwerk nicht einwandfrei.

EOF
POS(#9)

Die Funktion dieser Befehle bleibt beim Kassettenbetrieb erhalten.

INPUT #9
LINE INPUT #9
OPENIN und CLOSEIN

Wenn Sie die erste geeignete Datei auf der Kassette laden möchten, brauchen Sie den Namen nicht anzugeben. Beispiel:

```
OPENIN ""
```

Sie werden aufgefordert:

```
Press PLAY then any key:
```

...woraufhin Sie die PLAY-Taste auf dem Kassettengerät und danach eine beliebige Taste drücken. Das Band in der Kassette fängt an zu laufen, und der Computer lädt die ersten 2K-Byte der Datei in den Teil seines Speichers, der als 'Dateipuffer' bezeichnet wird.

Die Eingabe erfolgt vom Dateipuffer, bis er leer ist. Dann erscheint auf dem Bildschirm wieder:

```
Press PLAY then any key:
```

Nun werden die nächsten 2K-Byte in den Dateipuffer geladen.

Dabei wird wieder die Lademeldung:

```
Loading DATEINAME block 1
```

...ausgegeben, gefolgt von Anzeigen mit den übrigen Blocknummern, bis die Datei geladen ist.

Steht am Anfang des Dateinamens beim `OPENIN`-Befehl ein Ausrufungszeichen „!werden die oben genannten Meldungen unterdrückt und Sie brauchen nicht eine beliebige Taste zu drücken, um den Ladevorgang einzuleiten. (Die **PLAY**-Taste auf dem Kassettengerät muß allerdings gedrückt sein.) Möchten Sie Programme, die mit einem Ausrufungszeichen beginnen, auf Diskette laufen lassen, so wird das Ausrufungszeichen beim Laden der Diskette (d.h. beim Lesen des Dateinamens) ignoriert. Beachten Sie, daß das Ausrufungszeichen keine Stelle im Dateinamen einnimmt, weder bei der Kassetten- noch bei der Diskettenspeicherung.

Brechen Sie den Ladevorgang durch Drücken der **[ESC]**-Taste ab, so erscheint folgende Fehlermeldung:

```
Broken in
```

Hat das Laden nicht geklappt, sagt Ihnen der Abschnitt 'Lesefehler' vielleicht, warum.

LIST #9
OPENOUT und CLOSEOUT
PRINT #9
WRITE #9

Möchten Sie die Datei als `Unnamed file` (namenlose Datei) speichern, brauchen Sie keinen Namen anzugeben. Beispiel:

```
OPENOUT ""
```

Die ersten 2K-Byte der Datei, die auf Kassette gespeichert werden soll, werden zunächst in einem Teil des Speichers mit dem Namen 'Dateipuffer' gespeichert. Ist der Puffer voll, werden Sie angewiesen:

```
Press REC and PLAY then any key:
```

...woraufhin Sie die **RECORD**- und **PLAY**-Tasten auf Ihrem Kassettenlaufwerk drücken und danach eine beliebige Taste auf dem Computer. Das Band in der Kassette beginnt zu laufen und der Computer speichert den Inhalt des Puffers auf Kassette. Danach füllt er den Puffer mit den nächsten 2K-Byte auf und verlangt wieder:

```
Press REC and PLAY then any key:
```

Dann speichert er die 2K-Byte aus dem Puffer wieder auf Kassette.

Ist der Puffer nur teilweise voll, wenn der Befehl `CLOSEOUT` gegeben wird, wird dieser Rest auf Kassette gespeichert. Danach kommt die Aufforderung:

```
Press REC and PLAY then any key:
```

Nachdem Sie dies getan haben, erscheint die Meldung:

```
Saving DATEINAME block ‹x›
```

Beginnt der Dateiname im `OPENOUT`-Befehl mit einem Ausrufungszeichen ! , so wird die oben genannte Meldung unterdrückt und Sie brauchen keine beliebige Taste zu drücken, um den Speichervorgang einzuleiten. (Die **RECORD**- und **PLAY**-Tasten des Kassettenlaufwerks müssen jedoch gedrückt sein.) Möchten Sie ein Programm, das mit einem Ausrufungszeichen beginnt, auf Diskette laufen lassen, so wird das Ausrufungszeichen beim Speichern (d.h. beim Lesen des Dateinamens) ignoriert. Beachten Sie, daß das Ausrufungszeichen keine Stelle im Dateinamen einnimmt, weder bei der Kassetten- noch bei der Diskettenspeicherung.

Wird der Speichervorgang durch Drücken der **[ESC]**-Taste unterbrochen, so erscheint folgende Fehlermeldung:

```
Broken in
```

Damit das Speichern gelingt...

Beachten Sie folgende Hinweise, damit die Daten sicher gespeichert werden:

1. Prüfen Sie, ob Ihr Kassettenlaufwerk über die **TAPE**-Buchse richtig mit dem Computer verbunden ist (siehe Teil 2 des Grundlagenkurses).

2. Die **RECORD**- bzw. **LEVEL**-Regler am Kassettengerät müssen richtig eingestellt sein.

3. Verwenden Sie keine Kassetten minderwertiger Qualität oder C120-Kassetten.

4. Achten Sie darauf, die Kassetten keinen magnetischen Feldern auszusetzen, d.h. legen Sie sie nicht in die Nähe von Lautsprechern, Fernsehgeräten u.ä.

5. Bevor Sie ein Programm aus dem Speicher löschen, überprüfen Sie mit `CAT`, ob es wirklich gespeichert wurde.

6. Sorgen Sie dafür, daß das Kassettenlaufwerk regelmäßig gewartet wird und daß die Tonköpfe regelmäßig gereinigt werden.

SAVE

Wenn die Datei als namenlose Datei (`Unnamed file`) gespeichert werden soll, brauchen Sie ihr keinen Namen zu geben. Beispiel:

```
SAVE ""
```

Auf dem Bildschirm sehen Sie die Anweisung:

```
Press REC and PLAY then any key:
```

...woraufhin Sie die **RECORD**- und **PLAY**-Tasten des Kassettenlaufwerks und danach eine beliebige Taste auf dem Computer drücken. Das Band in der Kassette beginnt zu laufen, und der Computer speichert das Programm ab.

Dabei erscheint auf dem Bildschirm die Meldung:

```
Saving DATEINAME block 1
```

Danach folgen Anzeigen mit den übrigen Block-Nummern, bis die Datei gespeichert ist.

Beginnt der Dateiname mit einem Ausrufungszeichen ! , so werden die oben genannten Meldungen unterdrückt, und Sie brauchen keine beliebige Taste auf dem Computer zu drücken, um den Speichervorgang einzuleiten. (Die Tasten **RECORD** und **PLAY** auf dem Kassettengerät müssen jedoch gedrückt sein.) Möchten Sie ein Programm, das mit einem Ausrufungszeichen beginnt, auf Diskette laufen lassen, so wird das Ausrufungszeichen beim Speichern (d.h. beim Lesen des Dateinamens) ignoriert. Beachten Sie, daß das Ausrufungszeichen keine Stelle im Dateinamen einnimmt, weder bei der Kassetten- noch bei der Diskettenspeicherung.

Wird der Speichervorgang durch Drücken der **[ESC]**-Taste abgebrochen, so erscheint die Fehlermeldung:

```
Broken in
```

Lesen Sie den vorigen Abschnitt: 'Damit das Speichern gelingt...'

SPEED WRITE

Dieser Befehl wird nur beim Kassettenbetrieb ausgeführt. Er kann jedoch auch eingegeben werden, wenn der Computer auf Diskettenbetrieb geschaltet ist.

Fehlermeldungen

Während des Kassettenbetriebs können die Fehlermeldungen 7, 21, 24, 25, 27 und 32 auftreten (siehe Teil 6 des 7.Kapitels: 'Übersicht').

Externe AMSDOS-Befehle

Ob die Ein- bzw. Ausgabebefehle für Kassette oder Diskette gelten sollen, wird durch folgende Steuerbefehle bestimmt:

|TAPE (kann in |TAPE.IN und |TAPE.OUT unterteilt werden)
|DISC (kann in |DISC.IN und |DISC.OUT unterteilt werden)

Die nachfolgend aufgeführten Steuerbefehle sprechen nur die Diskette an, unabhängig davon, ob Kassettenbetrieb gewählt wurde oder nicht:

```
|A
|B
|CPM
|DIR
|DRIVE
|ERA
|REN
|USER
```

Kapitel 5
AMSDOS und CP/M

Teil 1: AMSDOS

Folgende Themen werden behandelt:

- ★ Einführung in AMSDOS
- ★ Disketten-Inhaltsverzeichnis
- ★ Disketten-Wechsel
- ★ Dateinamen und -typen
- ★ AMSDOS Datei-Vorspann
- ★ Dateinamen beim Betrieb mit zwei Diskettenlaufwerken
- ★ Wild Cards (Universalzeichen)
- ★ Beispiel-Programm mit AMSDOS-Befehlen
- ★ Liste der AMSDOS-Befehle
- ★ Kopieren von Dateien
- ★ Liste der Fehlermeldungen

Einleitung

AMSDOS erweitert das mit Ihrem Computer gelieferte BASIC durch eine Anzahl externer Befehle, die jeweils durch das vorangestellte Balkensymbol | gekennzeichnet werden.

AMSDOS ermöglicht es dem Benutzer, die Disketten nach Belieben zu wechseln, solange keine Dateien in Bearbeitung sind - sonst wird eine Fehlermeldung ausgegeben, und es kann zu Datenverlusten kommen, falls in die offene Datei geschrieben wurde.

Disketten-Inhaltsverzeichnis

Jede Diskette ist in zwei Bereiche unterteilt, einen für das Inhaltsverzeichnis und einen für die Daten. Das Inhaltsverzeichnis enthält eine Liste aller Dateinamen und einen 'Belegungsplan', der angibt, an welcher Stelle auf der Diskette sich die einzelnen Dateien befinden. AMSDOS und CP/M können die Größe einer Datei nach Einsicht der Einträge im Inhaltsverzeichnis berechnen. Die Berechnung des freien Speicherplatzes auf der Diskette geschieht durch Addition aller Dateien des Inhaltsverzeichnisses und Ermittlung des noch verbleibenden Speicherbereiches.

Beim Lesen einer Datei wird aus dem Eintrag im Inhaltsverzeichnis der Speicherort entnommen. Beim Anlegen einer neuen Datei wird der freie Speicherplatz reduziert, beim Löschen einer Datei entsteht freier Speicherplatz. Das Inhaltsverzeichnis arbeitet mit 1KByte-Einheiten und kann bis zu 64 verschiedene Einträge verwalten. Große Dateien werden im Inhaltsverzeichnis in 16KByte-Blöcke unterteilt, was dem Anwender jedoch verborgen bleibt.

Diskettenwechsel

Beim Betreib mit AMSDOS (und CP/M PLUS) können Disketten beliebig gewechselt oder aus dem Laufwerk genommen werden, solange auf das Laufwerk nicht zugegriffen wird, und solange weder Eingabe- noch Ausgabe-Dateien auf diesem Laufwerk geöffnet sind. Anders als mit CP/M 2.2 ist ein 'Log-In' der Disketten (Warmstart) nicht erforderlich.

Das Herausnehmen einer Diskette, während auf sie geschrieben wird, kann ihre Daten beschädigen. Wird eine Diskette herausgenommen, auf der sich noch offene Dateien befinden, verläßt AMSDOS alle offenen Dateien, sobald es diesen Zustand erkannt hat, und gibt eine Fehlermeldung aus. Alle Daten, die noch auf Diskette geschrieben werden sollten, werden nicht gespeichert, und der letzte Eintrag in das Inhaltsverzeichnis der Diskette wird nicht vorgenommen. AMSDOS kann das Herausnehmen der Diskette aber nur dann erkennen, wenn es das Inhaltsverzeichnis liest, also nach jedem 16KByte-Block, oder wenn eine Datei eröffnet oder geschlossen wird. Es können also maximal 16KByte Daten verloren gehen, wenn eine Diskette mit nicht geschlossenen Dateien herausgenommen wird.

AMSDOS- Dateinamen und -Dateitypen

Es ist üblich, Dateinamen so zu wählen, daß der Dateityp daraus hervorgeht. Diese Namenskonvention zwingt den Computer nicht, die Datei auf eine bestimmte Art und Weise zu behandeln. Einige Programme akzeptieren eine Datei jeoch nur, wenn der richtige Namenstyp angegeben ist. AMSDOS akzeptiert jeden Namenstyp, sucht aber bevorzugt bestimmte Dateitypen (siehe 'AMSDOS-Vorspann').

Erstellen von Dateinamen

Der Dateiname besteht aus zwei Teilen, die durch einen Punkt (.) voneinander getrennt sind. Der erste Teil kann bis zu acht, der zweite bis zu drei Zeichen lang sein. So sind beispielsweise `"ROINTIME.DEM"`, `"DISCKIT3.COM"` und `"DISC.BAS"` zulässige Dateinamen.

Der zweite Teil des Dateinamens ist der sogenannte Dateityp. Dateiname und Dateityp können aus Buchstaben und Ziffern zusammengesetzt sein. Sie dürfen jedoch keine Leerzeichen oder Satzzeichen enthalten. Einige gebräuchliche Dateitypen sind:

.‹Leerstellen›	Nicht spezifizierter Typ. Kann eine mit dem Befehl `OPENOUT "NAME"` angelegte Datei sein, oder ein durch AMSDOS (mittels `SAVE "PROGRAMM",A`) gespeichertes BASIC-Programm.
`.BAS`	BASIC-Programm, durch AMSDOS mittels `SAVE "NAME"` oder `SAVE "NAME",P` oder `SAVE "NAME.BAS",A` gespeichert.
`.BIN`	Programm oder Speicherbereich, durch AMSDOS mittels `SAVE "NAME",B,`‹binäre Parameter› gespeichert.
`.BAK`	Alte Version einer Datei. AMSDOS oder ein Dienstprogramm haben eine neue Version unter demselben Namen gespeichert. Bei Bedarf kann auf die alte Version zurückgegriffen werden.
`.COM`	Befehlsdatei. Alle CP/M-Dienstprogramme sind von diesem Dateityp.
`.SUB`	Instruktionsdatei für das CP/M `SUBMIT`-Programm.

AMSDOS-Vorspann

AMSDOS speichert Dateien automatisch mit einem passenden Typ-Identifikator. Normalerweise brauchen Sie also den Dateityp nicht anzugeben, es sei denn, Sie möchten andere als die oben genannten Standard-Dateitypen angeben. BASIC-Programmdateien, geschützte BASIC-Programmdateien, sowie Binärdateien werden mit einem Vorspann auf Diskette gespeichert, so daß der AMSDOS-Befehl

```
LOAD "Name"
```

sie wiedererkennen und entsprechend bearbeiten kann. Findet AMSDOS nach dem Befehl `LOAD` keinen Vorspann, geht es davon aus, daß es sich um eine ASCII-Datei, also eine reine Textdatei, handelt.

Ungeachtet dessen, was im Vorspann steht, sucht AMSDOS, wenn es eine Datei ohne Typspezifikation laden soll, zunächst unter den Dateien vom Typ:

```
.‹Leerstellen›
```

Ist die Datei dort nicht zu finden, sucht es unter:

```
.BAS
```

...und schließlich unter:

```
.BIN
```

Auf diese Weise kann sich der Benutzer die Angabe des Dateityps im Namen in den meisten Fällen ersparen.

Eine durch den Befehl `OPENOUT` eröffnete und danach beschriebene Disketten-Datei hat keinen Vorspann. Sie ist eine ASCII-Datei (Textdatei) und enthält nur die Daten aus den `WRITE`-, `PRINT`- und `LIST`-Anweisungen. Der Disketten-Befehl `OPENIN` sucht nach Dateien ohne Typenspezifikation in derselben Reihenfolge wie `LOAD`.

Dateinamen beim Betrieb mit zwei Diskettenlaufwerken

Beim Betrieb mit einem doppelten Diskettenlaufwerk, d.h. wenn Sie Ihr Computer-System mit einem zusätzlichen Diskettenlaufwerk ausgerüstet haben, können sich in beiden Laufwerken Dateien befinden. Da der Computer nicht automatisch in beiden Laufwerken nach einer Datei sucht, muß der Benutzer das Laufwerk angeben. Sie können zur Bestimmung des Laufwerks entweder die Befehle `|A`, `|B` oder `|DRIVE` benutzen (genaue Beschreibung folgt) und dann den Dateinamen nennen, oder dem Dateinamen `A:` oder `B:` voranstellen, und somit die vorherige Laufwerkszuweisung vorübergehend außer Kraft setzen. Ein Programm wird also sowohl durch:

```
|B
SAVE "PROG.BAS"
|A
```

als auch durch:

```
|A
SAVE "B:PROG.BAS"
```

auf die Diskette in Laufwerk B gespeichert.

Auf gleiche Weise wird die vorher zugewiesene USER- (=Benutzer) Nummer - USER- Nummern dienen zur Unterteilung des Inhaltsverzeichnisses -ignoriert, wenn dem Dateinamen eine USER-Nummer (zwischen 0 und 15) vorangestellt wird. Durch die Befehle:

```
LOAD "15:PROG.BAS"
```

und

```
SAVE "15:PROG.BAS"
```

würde das Programm im Diskettenteil der USER-Nummer 15 laden und speichern, ganz gleich, welche USER-Nummer vorher angegeben war (siehe |USER-Befehl unter 'Liste der AMSDOS-Steuerbefehle').

Schließlich ist es auch möglich, beide Einstellungen für USER und DRIVE ungültig zu machen, indem man sie in dieser Reihenfolge dem Dateinamen voranstellt, z.B.:

```
RUN "15B:PROG.BAS"
```

Universalzeichen (Wild Cards)

Häufig ist es erforderlich, einige Disketten-Operationen (kopieren, löschen usw.) mit mehreren Dateien durchzuführen. Wenn ein Dateiname für eine bestimmte Operation vorgesehen ist, sucht AMSDOS im Inhaltsverzeichnis nach genau diesem Namen. Dort wo der Befehl es zuläßt, kann die Operation auch auf mehrere Dateien bezogen werden, indem man einen Teil der Zeichen im Dateinamen durch Universalzeichen ersetzt. Jedes Fragezeichen ersetzt als Universalzeichen ein Stelle im Dateinamen. Wenn mehrere aufeinanderfolgende Zeichen im Namen durch ein Universalzeichen ersetzt werden sollen, kann man die Reihe von Fragezeichen durch das Sternchen * abkürzen. Der Dateiname FRED.* ist also eine Abkürzung für FRED.???, und F*.BAS kann für F???????.BAS stehen.

Der Ausdruck *.* bedeutet demzufolge 'alle Dateien'.

Beispiele:

Inhaltsverzeichnis	Kurzform *.BAS entspricht	Kurzform FRED?.BAS entspricht	Kurzform F*.BA? entspricht
BERT.BAS	BERT.BAS		
FRED1.BAS	FRED1.BAS	FRED1.BAS	FRED1.BAS
FRED2.BAS	FRED2.BAS	FRED2.BAS	FRED2.BAS
FRED3.BAK			FRED3.BAK
FRED3.BAS	FRED3.BAS	FRED3.BAS	FRED3.BAS
FERTIG.BAS	FERTIG.BAS		FERTIG.BAS

Beispiel zu AMSDOS-Befehlen in Programmen

Wir empfehlen Ihnen zum besseren Verständnis der AMSDOS-Befehle, die folgenden Beispiele durchzuarbeiten und sich dazu die entsprechenden Abschnitte dieses Kapitels anzusehen. Benutzen Sie dazu auf keinen Fall eine originale CP/M-Systemdiskette, verwenden Sie stattdessen eine Arbeitskopie.

Speichern von Variablen und Herstellung eines Bildschirmabdrucks

Im ersten Beispiel wird auf die Diskette geschrieben. Deshalb müssen Sie eine formatierte Leerdiskette oder Arbeitsdiskette ins Laufwerk einlegen, um das Programm ablaufen zu lassen. Mit dem Programm wird die britische Nationalflagge (Union Jack) gezeichnet und als Bild auf Diskette gespeichert.

```
10 dumpdatei$="flagdump.srn"
20 MODE 1:BORDER 0
30 DIM colour(2)
40 FOR i=0 TO 2
50  READ Farbe(i): REM Farben kommen von DATA- Anweisung
60  INK i,Farbe(i)
70 NEXT
80 ON ERROR GOTO 430
90 OPENIN "param.dat" ' testen ob Datei existiert
100 CLOSEIN:ON ERROR GOTO 0
110 IF errnum=32 AND DERR=146 THEN CLS:
    GOTO 160 ' Datei existiert nicht
```

```
120 CURSOR 1:PRINT "Soll alte Datei ueberschrieben
    werden? J/N ";
130 a$=INKEY$:ON INSTR(" JN",UPPER$(a$))
    GOTO 130,150,140:GOTO 130
140 PRINT a$:PRINT "Programm verlassen":END
150 PRINT a$:CURSOR 0
160 OPENOUT "param.dat"
170 WRITE #9,dumpdatei$,1: REM speichert Dateinamen und
    Modus
180 FOR i=0 TO 2
190 WRITE #9,Farbe(i): REM speichert Farben
200 NEXT i
210 CLOSEOUT
220 CLS
230 gp=1:GRAPHICS PEN gp:w=125
240 x=-65:a=240:y=400:b=-150:GOSUB 400
250 y=0:b=150:GOSUB 400
260 x=575:a=-240:y=400:b=-150:GOSUB 400
270 y=0:b=150:GOSUB 400
280 gp=2:GRAPHICS PEN gp:w=40
290 a=240:x=-40:y=400:b=-150:GOSUB 400
300 x=0:y=0:b=150:GOSUB 400
310 a=-240:x=640:y=0:b=150:GOSUB 400
320 x=600:y=400:b=-150:GOSUB 400
330 ORIGIN 0,0,256,380,0,400:CLG 1
340 ORIGIN 0,0,0,640,150,250:CLG 1
350 ORIGIN 0,0,280,352,0,400:CLG 2
360 ORIGIN 0,0,0,640,168,230:CLG 2
370 SAVE dumpdatei$,b,&C000,&4000
380 DATA 2,26,6
390 END
400 MOVE x,y:DRAWR a,b:DRAWR w,0:DRAWR -a,-b
410 MOVE x+a/2+w/2,y+b/2:FILL gp
420 RETURN
430 errnum=ERR:RESUME NEXT
run
```

Die Dateitypen .DAT und .SRN deuten auf den Inhalt der Dateien hin. PARAM.DAT ist eine ASCII-Datei ohne Vorspann, FLAGDUMP.SRN ist eine AMSDOS-Binär-DATEI mit Vorspann.

Sie sehen, wie das Programm absichtlich versucht, von der Datei PARAM.DAT zu lesen, bevor es auf sie schreibt, um festzustellen, ob die Datei schon existiert. Gibt es diese Datei nicht, wird von BASIC eine Fehlermeldung ausgegeben. Der Fehler wird vom Programm abgefangen und das Programm wird ohne Unterbrechung ausgeführt. Existiert die Datei schon, erscheint keine Fehlermeldung, sondern das Programm fragt automatisch, ob diese Datei überschrieben werden soll.

Die Informationen zum Bildschirmabdruck, wie Bildschirm-Modus, Paletten-Farbe und Name der Datei, die die eigentliche Information enthält, werden in einer Parameter-Datei gespeichert. Dieses Beispiel illustriert, wie Programm-Variablen (`dumpdatei$`) und Konstanten (`1`) auf eine Daten-Datei gespeichert werden (mittels `WRITE`), damit sie von anderen Programmen benutzt werden können.

Laden des Bildschirmabdrucks

Das zweite Beispiel ist ein allgemeines Programm zu Darstellung eines Bildschirmabzugs unter Steuerung einer Parameter-Datei. Beachten Sie, wie Variablen mittels `INPUT` von der Daten-Datei geholt und mit der `EOF`-Funktion die Größe der Datei automatisch variiert wird. Der durch dieses Programm dargestellte Bildschirmabdruck muß auf einer bekannten Position im Speicher liegen, sonst wird die Darstellung verschoben. Dies wird sichergestellt, indem das Speicher-Programm einen `MODE`-Befehl ausführt und der Bildschirm danach nicht 'gescrollt' wird.

```
10 DIM Farbe(15): REM bietet 16 Farben an
20 OPENIN "param.dat"
30 INPUT #9, Dateiname$,Bildschirmmodus
40 i=0
50 WHILE NOT EOF
60  INPUT #9,Farbe(i)
70 INK i,Farbe(i)
80 i=i+1
90 WEND
100 CLOSEIN
110 MODE Bildschirmmodus:BORDER 0
120 LOAD Dateiname$
run
```

Liste der externen AMSDOS-Befehle

|A

```
|A
```

KOMMANDO: Bestimmt Laufwerk A als Standard-Laufwerk. Entspricht dem Befehl `|DRIVE` mit Parameter `A`. (Das eingebaute Laufwerk im Computer ist Laufwerk A.

|B

|B

KOMMANDO: Bestimmt Laufwerk B als Standard-Laufwerk. Entspricht dem Befehl |DRIVE mit Parameter B. (Das eingebaute Laufwerk im Computer ist Laufwerk A.)

|CPM

|CPM

KOMMANDO: Schaltet um auf eine andere Disketten-Umgebung, indem es das Betriebssystem von einer System-Diskette lädt. Die mit dem Computer gelieferten Betriebssysteme sind CP/M Plus und CP/M 2.2.

Das Kommando bleibt wirkungslos, wenn sich keine System-Diskette mit CP/M im Laufwerk befindet.

|DIR

|DIR [,‹Textausdruck›]

```
|DIR,"*.BAS"
```

KOMMANDO: Zeigt das Inhaltsverzeichnis der Diskette (im CP/M-Stil) sowie den freien Speicherplatz an. Wenn der ‹Textausdruck› fehlt, wird *.* angenommen.

|DISC

|DISC

KOMMANDO: Entspricht den beiden Kommandes |DISC.IN und |DISC.OUT.

|DISC.IN

|DISC.IN

KOMMANDO: Verwendet die Diskette als Datei-Eingabe-Medium.

|DISC.OUT

|DISC.OUT

KOMMANDO: Verwendet die Diskette als Datei-Ausgabe-Medium.

|DRIVE

|DRIVE,‹Textausdruck›

```
|DRIVE,"A"
```

KOMMANDO: Legt das Standard-Laufwerk fest. Dieser Befehl funktioniert nur, wenn AMSDOS die Diskette im angegebenen Laufwerk lesen kann.

|ERA

|ERA,‹Textausdruck›

```
|ERA,"*.BAK"
```

KOMMANDO: Alle Dateien des angegebenen Namens, die nicht nur gelesen werden, werden gelöscht. Es können auch Universalzeichen eingesetzt werden.

|REN

|REN,‹Textausdruck›,‹Textausdruck›

```
|REN,"NEUNAME.BAS","ALTNAME.BAS"
```

KOMMANDO: Gibt einer Datei einen neuen Namen. Eine Datei mit dem neuen Namen darf noch nicht existieren. Unversalzeichen sind nicht erlaubt.

Für einen ‹Textausdruck› kann auch der USER-Parameter (siehe Befehl |USER) angegeben werden, um die Standard-USER-Nummer auszuschalten. Wenn der Befehl z.B. lautet:

```
|REN,"Ø:NEU.BAS","15:ALT.BAS"
```

wird die Datei "ALT.BAS" unter USER 15 nach ihrer Umbenennung in "NEU.BAS" unter USER Ø eingeordnet, gleichgültig, welche USER-Nummer vorher bestimmt worden war.

|TAPE

|TAPE

KOMMANDO: Entspricht den beiden Befehlen |TAPE.IN und |TAPE.OUT. Wird nur benötigt, wenn ein Kassettenlaufwerk angeschlossen ist.

|TAPE.IN

|TAPE.IN

KOMMANDO: Verwendet die Kassette als Datei-Eingabe-Medium, vorausgesetzt, daß ein Kassettengerät angeschlossen ist.

|TAPE.OUT

|TAPE.OUT

KOMMANDO: Verwendet die Kassette als Datei-Ausgabe-Medium, vorausgesetzt, daß ein Kassettengerät angeschlossen ist.

|USER

|USER,‹ganzzahliger Ausdruck›

```
|USER,3
```

KOMMANDO: Legt fest, für welchen der bis zu 16 verschiedenen Abschnitte des Inhaltsverzeichnisses (von 0 bis 15) die Disketten-Befehle (z.B. CAT, LOAD, |DIR, usw.) gelten sollen.

Eine Datei der einen USER-Nummer kann mit dem |REN-Kommando auf eine andere übertragen werden. So überträgt z.B.:

```
|REN,"15:BEISPIEL.BAS","Ø:BEISPIEL.BAS"
```

die Datei BEISPIEL.BAS von USER Ø auf USER 15, ohne daß ihr Name geändert wird.

Kopieren von Dateien

AMSDOS-Dateien mit Vorspann

Dateien dieses Typs können in der CP/M-Umgebung mittles PIP (siehe Teil 2 in diesem Kapitel) kopiert werden. Jede unter AMSDOS erstellte Datei mit Vorspann (s.o. unter 'AMSDOS-Vorspann') kann vollständig von Diskette auf Diskette kopiert werden. Im allgemeinen kann jedoch der Inhalt der Datei durch CP/M-Programme nicht bearbeitet werden.

ASCII-Dateien

Unter AMSDOS erstellte Dateien ohne Vorspann sind in der Regel ASCII-Dateien. Sie können sowohl kopiert, als auch von CP/M-Programmen verstanden werden. Inbesondere sollte es möglich sein, ASCII-Programm-Dateien, ASCII-Daten-Dateien und ASCII-Text-Dateien frei zwischen AMSDOS-und CP/M-Programmen auszutauschen.

Nur lesbare Dateien

Mit CP/M kann eine Datei als 'nur lesbar' deklariert und/oder im Inhaltsverzeichnis in einen besonderen Systemzustand versetzt werden. Derartige Attribute können nur in einer CP/M-Umgebung bestimmt werden; sie werden jedoch auch von AMSDOS respektiert. Weitere Einzelheiten dazu finden Sie im 2.Teil dieses Kapitels unter SET.

Kopieren zwischen Diskette und Kassette

CP/M Plus unterstützt keine Kassettendateien. Es ist deshalb nicht möglich, Kassettendateien auf Diskette zu kopieren oder umgekehrt Diskettendateien auf Band zu übertragen. Für diese Zwecke müssen Sie die Dienstprogramme CLOAD und CSAVE von CP/M 2.2 (auf Seite 4 der Systemdisketten) benutzen.

Verfahren zum Kopieren von Dateien

Die folgenden Tabellen zeigen die Kopierverfahren von Kassette (falls angeschlossen) auf Diskette und umgekehrt für Dateien jeden Typs. Es wird davon ausgegangen, daß kein weiteres Diskettenlaufwerk angeschlossen ist. Geschützte BASIC-Programme können überhaupt nicht von oder auf Kassette kopiert werden, Binär-Dateien (z.B. in Maschinensprache geschriebene Videospiele) sind nur kopierfähig, wenn die Speicheradresse bekannt ist. Weitere Einzelheiten über die PIP-, CLOAD- und CSAVE-Programme finden Sie im zweiten Teil dieses Kapitels.

Kopiertabellen

Kopierquelle

Kopierziel:	*AMSTRAD BASIC auf Kassette **	*ASCII -Daten auf Kassette **	*Binär auf Kassette **

*AMSTRAD BASIC auf Kassette **	`|TAPE`
	`LOAD "FILE"`
	‹Kassettenwechsel›
	`SAVE "FILE"`
	`|DISC`		

*Binär auf Kassette **			`H=HIMEM`
			`|TAPE`
			`MEMORY ‹s›-1`
			`LOAD "FILE"`
			‹Kassettenwechsel›
			`SAVE "FILE" ,B,`
			‹a›,‹l›[,‹s›]
			`|DISC`
			`MEMORY H`
			‹Anmerkung 2›

*ASCII auf Kassette **	`|TAPE`
	`LOAD "FILE"`
	‹Kassettenwechsel›
	`SAVE "FILE",A`
	`|DISC`
		‹Diskette mit CP/M 2.2 einlegen›
		`|CPM`
		`CLOAD "FILE",TEMP`
		‹Kassettenwechsel›
		`CSAVE TEMP ,"FILE"`
		`ERA TEMP`
		`AMSDOS`
		‹Anmerkung 1›	

*AMSTRAD BASIC auf Diskette **	`|TAPE`
	`LOAD "FILE"`
	`|DISC`
	`SAVE "FILE"`		

ASCII auf Diskette	`|TAPE`
	`LOAD "FILE"`
	`|DISC`
	`SAVE "FILE",A`
		‹Diskette mit CP/M 2.2 einlegen›
		`|CPM`
		`CLOAD "FILE"`
		`AMSDOS`	

*Binär auf Diskette **			`H=HIMEM`
			`|TAPE`
			`MEMORY ‹s›-1`
			`LOAD "FILE"`
			`|DISC`
			`SAVE"FILE" ,B,`
			‹a›,‹l›[,‹s›]
			`MEMORY H`
			‹Anmerkung 2›

* Datei mit Vorspann

‹Anmerkung 1› Erfordert freien Disketten- Speicherplatz für die temporäre Datei "TEMP".
‹Anmerkung 2› ‹a› ist die Anfangsadresse der Datei, ‹l› die Länge und ‹s› optional die Startadresse.

Kopierziel	Kopierquelle: *AMSTRAD BASIC auf Diskette **	*ASCII -Daten auf Diskette*	*AMSDOS Binär -Daten auf Diskette*	*Alle anderen auf Diskette*

AMSTRAD BASIC auf Kassette *	`LOAD "FILE"`
	`|TAPE`
	`SAVE "FILE"`
	`|DISC`			

Binär auf Kassette *			`H=HIMEM`
			`MEMORY s-1`
			`LOAD "FILE"`
			`|TAPE`
			`SAVE "FILE", B,`
			`a,l[,s]`
			`|DISC`
			`MEMORY H`
			(Anmerkung 2)	

ASCII auf Kassette *	`LOAD "FILE"`
	`|TAPE`
	`SAVE "FILE",A`
	`|DISC`
	-oder-
	(Diskette mit CP/M 2.2 einlegen)
	`|CPM`
	`CSAVE FILE`
	`AMSDOS`
		(Diskette mit CP/M 2.2 einlegen)
		`|CPM`
		`CSAVE FILE`
		`AMSDOS`
				(Diskette mit CP/M 2.2 einlegen)
				`|CPM`
				`CSAVE FILE`
				`AMSDOS`
				(Anmerkung 3)

AMSTRAD BASIC auf Diskette *	`LOAD "FILE"`
	(Diskettenwechsel)
	`SAVE "FILE"`
	- oder -
	(Diskette mit CP/M Plus einlegen)
	`|CPM`
	`PIP B:=FILE`
	`AMSDOS`			

ASCII auf Diskette	`LOAD "FILE"`
	(Diskettenwechsel)
	`SAVE "FILE",A`
		(Diskette mit CP/M Plus einlegen)
		`|CPM`
		`PIP B:=FILE`
		`AMSDOS`		

AMSDOS Binär auf Diskette *			(Diskette mit CP/M Plus einlegen)
			`|CPM`
			`PIP B:=FILE`
			`AMSDOS`	

Alle anderen auf Diskette				(Diskette mit CP/M Plus einlegen)
				`|CPM`
				`PIP B:=FILE`
				`AMSDOS`

* Datei mit Vorspann

(Anmerkung 2) (a) ist die Anfangsadresse der Datei, (l) die Länge und (s) optional die Startadresse
(Anmerkung 3) Zieldatei kann nicht direkt von BASIC benutzt werden. Diese Option ist als kostengünstiges Transport - und Backup - Medium nützlich. Die Datei kann mit `CLOAD "FILE"` zurück auf Diskette kopiert werden.

Liste der AMSDOS-Fehlermeldungen

Wenn AMSDOS irgendeinen Befehl nicht ausführen kann, wird eine Fehlermeldung ausgegeben. Gibt es Probleme mit der Hardware, folgt auf die Fehlermeldung die Frage:

```
Retry, Ignore or Cancel?
```

R (Retry) verursacht eine Wiederholung der Operation, möglicherweise nachdem der Anwender entsprechende Maßnahmen getroffen hat.

I (Ignore) bewirkt eine Fortsetzung der Bearbeitung, als wäre kein Problem aufgetreten. Dies führt häufig zu unerwarteten oder gar unerwünschten Ergebnissen.

C (Cancel) macht die Operation rückgängig, was häufig zu einer neuen Fehlermeldung führt.

Die Fehlermeldungen und ihre Bedeutung

```
Unknown command
```

Der Befehl ist nicht richtig geschrieben

```
Bad command
```

Der Befehl kann aus bestimmten Gründen nicht ausgeführt werden (Syntax- oder Hardware-Fehler).

‹Datei› `already exists`

Der Benutzer versucht, eine Datei mit einem schon existierenden Namen zu versehen.

‹Datei› `not found`

Datei existiert nicht.

`Drive` ‹Laufwerk› `: directory full`

Kein Platz mehr für neue Einträge in das Disketten-Inhaltsverzeichnis.

`Drive` ‹Laufwerk› `: disc full`

Keine Platz mehr für neue Daten auf der Diskette

`Drive ‹Laufwerk›: disc changed, closing ‹Datei›`

Diskette wurde gewechselt, ohne alle darauf gespeicherten Dateien zu schließen.

`‹Datei› is read only`

Datei kann nicht bearbeitet werden, da sie nur lesbar ist. Dateien können nur in einer CP/M-Umgebung in den Zustand 'nur lesbar' oder 'schreib- und lesbar' versetzt werden.

`Drive ‹Laufwerk›: disc missing`

Diskette wurde nicht oder nicht korrekt eingelegt, bzw. dreht sich nicht. Drücken Sie den Auswurfknopf und legen Sie die Diskette noch einmal ein. Anschließend **R** eingeben.

`Drive ‹Laufwerk›: disc is write protected`

Es wurde versucht, auf eine Diskette mit offenem Schreibschutzloch zu schreiben. Sie nehmen die Diskette heraus, schließen das Schreibschutzloch, schieben die Diskette wieder hinein und drücken **R**.

`Drive ‹Laufwerk›: read fail`

Hardware-Fehler beim Lesen der Diskette. Nehmen Sie die Diskette heraus und legen Sie sie wieder ein. Danach **R** drücken.

`Drive ‹Laufwerk›: write fail`

Hardware-Fehler beim Beschreiben der Diskette. Nehmen Sie die Diskette heraus und legen Sie sie wieder ein. Danach **R** drücken.

`Failed to load CP/M`

Lesefehler beim Laden von CP/M mit dem **\|CPM**-Befehl oder Verwendung einer untauglichen System-Diskette ohne CP/M. Beachten Sie, daß der Versuch, CP/M von einer Daten-Diskette zu laden, mit der Fehlermeldung '`read fail`' quittiert wird.

Teil 2: CP/M

CP/M Plus

In diesem Teil werden folgende Themen behandelt:

- ★ Einführung in CP/M
- ★ 'Booten' von CP/M Plus
- ★ Direkter Konsolmodus
- ★ Transiente Programme
- ★ Peripherie-Verwaltung
- ★ Einsatz von CP/M 2.2

CP/M Plus ist ein Disketten-Betriebssystem. Es ist ein spezielles Programm, welches Ihnen die volle Nutzung des CPC6128 erschließt. Die 128K RAM können voll ausgenutzt werden, wobei für Anwender-Programme 61K zur Verfügung stehen. CP/M bietet direkten Zugriff auf Daten-Dateien, und der CPC6128 verfügt über einen komplizierten Bildschirm-Anpassungsmechanismus.

Da CP/M für sehr viele unterschiedliche Computer angeboten wird, können Sie unter tausenden von Anwender-Programmpaketen wählen, um Ihr Wissen und Ihren Erfahrungshorizont in vielfältiger Weise zu erweitern.

Nähere Einzelheiten zu CP/M Plus, einschließlich Anleitungen zum Entwickeln eigener Programme, sind in SW 971 'CP/M Plus Handbuch' enthalten.

Einleitung

Das CP/M-Betriebssystem ermöglicht es dem Benutzer, mit dem Computer zu kommunizieren, sowie Dateien und Peripheriegeräte zu manipulieren. Sonderbefehle (und sogenannte Dienstprogramme auf der Diskette) helfen Ihnen bei Ihrer Hauptaufgabe, die Anwender-Programme mit Ihren Daten laufen zu lassen.

Natürlich können Sie lernen, mit CP/M und all den verschiedenen Dienstprogrammen wie ein Experte umzugehen, und dieses Wissen kann Ihnen sehr zugute kommen, wenn Sie einmal in Schwierigkeiten geraten sollten. Für die meisten von Ihnen aber genügt es zu wissen, wie man den Computer in Gang bringt. In diesem Teil des Kapitels sollen alle Merkmale und Möglichkeiten vorgestellt werden, ohne den Blick auf das Wesentliche mit zu vielen Detailinformationen zu verstellen.

Während BASIC einen Direktmodus hat, der durch die Bereitschaftsmeldung 'ready' gekennzeichnet ist, gibt es bei CP/M den 'Direkten Konsol-Modus', der sich mit `A>` oder `B>` meldet. Es stehen bestimmte eingebaute Befehle zur Verfügung, aber der Großteil der 'Hausarbeit' wird durch das Laden und Starten sogenannter transienter Programme verrichtet. Sie werden als transient (=vorübergehend) bezeichnet, weil sie sich nur im Computer befinden (von Diskette geladen), während Sie sie benutzen, im Gegensatz zu eingebauten Programmen.

Neben CP/M-Standard-Fehlermeldungen gibt das System auch eine Reihe spezieller Hardware-Fehlermeldungen aus. Man erkennt sie normalerweise daran, daß sie auf der letzten Bildschirmzeile erscheinen.

CP/M Plus auf Diskette

Der größte Teil von CP/M Plus befindet sich in einer Datei mit dem Dateityp `.EMS` auf Seite 1 des System-Diskettensatzes. Der Computer lädt CP/M in zwei Phasen von dieser Datei in seinen Speicher.

Zuerst lädt der AMSDOS-Befehl `|CPM` den ersten Sektor von Spur 0. Auf einer System-Diskette ist dieser Sektor als Programm ausgelegt, das dann die `.EMS`-Datei in den Speicher lädt. Die restlichen System-Spuren bleiben unbenutzt.

Profile-Programm für den Kaltstart

Falls sich die Datei `PROFILE.SUB` auf der Diskette befindet, während CP/M Plus geladen wird, werden die Anweisungen dieser Datei dem Computer übergeben. Auf diese Weise kann auf eine andere Tastatur umgeschaltet, der Bildschirm angepaßt, der Drucker in Gang gesetzt oder eine Anwender-Programm sofort gestartet werden. In Kapitel 4 haben wir gezeigt, wie eine `PROFILE`-Datei von Seite 1 umbenannt und somit aktiviert werden kann.

Während die `PROFILE`-Datei bearbeitet wird, wird auf der Diskette eine kleine Übergangs-Datei eröffnet, die deshalb beschreibbar gemacht werden muß. Dies ist der Grund, weshalb die System-Diskette selber keine erkennbare `PROFILE`-Datei enthalten kann.

PROFILE-Dateien können mit einer Textverarbeitung, einem Text-Editor (z.B. ED.COM) oder auch mit BASIC erstellt werden. Man könnte beispielsweise mit folgendem kleinen BASIC-Programm die PROFILE.SUB-Datei einrichten:

```
10 OPENOUT "PROFILE.SUB"
20 PRINT #9, "SETKEYS KEYS.CCP"
30 PRINT #9,"LANGUAGE 2"
40 CLOSEOUT
```

Steuer-Codes von der Konsole

In der CP/M-Umgebung werden besondere Tasten-Funktionen benutzt. Diese Tastenfunktionen ersetzen die Funktion der **[ESC]**- und Cursor-Tasten des AMSTRAD-BASIC. Die nachfolgend aufgeführten Steuer-Codes treten nach dem Befehl:

```
SETKEYS KEYS.CCP
```

in Kraft, wobei sowohl das transiente Programm SETKEYS.COM als auch die Befehls-Datei KEYS.CCP auf Seite 1 der Systemdisketten steht.

Steuer-Code	Taste	Funktion
[CONTROL]A	◊	Bewegt den Cursor um eine Stelle nach links.
[CONTROL]B	**[CONTROL]**◊ oder **[CONTROL]**◊	Bewegt den Cursor zum Anfang der Zeile. Steht er schon am Anfang, wird er ans Zeilenende gesetzt.
[CONTROL]C	**[CONTROL][ESC]**	Unterbricht.
[CONTROL]E	**[CONTROL][RETURN]**	Zeilenvorschub
[CONTROL]F	◊	Bewegt den Cursor um eine Stelle nach rechts.
[CONTROL]G	**[CLR]**	Löscht das Zeichen, auf dem sich der Cursor befindet.
[CONTROL]H	**[DEL]**	Löscht das Zeichen links vom Cursor.

[CONTROL]I	**[TAB]**	Rückt den Cursor zum nächsten Tabulator vor.
[CONTROL]J	**[RETURN]** oder **[ENTER]**	Gibt die Befehlszeile in den Computer ein.
[CONTROL]K	**[CONTROL][CLR]**	Löscht alles bis zum Ende der Zeile.
[CONTROL]M	**[RETURN]** oder **[ENTER]**	Gibt die Befehlszeile in den Computer ein.
[CONTROL]P		Schaltet auf Druckerausgabe um, d.h. schaltet die Bildschirmausgabe auf den Drucker.
[CONTROL]Q		Nimmt die Bildschirmausgabe wieder auf.
[CONTROL]R	**[CONTROL][ENTER]**	Kopiert die Befehlszeile.
[CONTROL]S	**[ESC]**	Stoppt die Bildschirmausgabe von CP/M. Wiederaufnahme mit **[CONTROL]Q.**
[CONTROL]U		Setzt die gesamte Zeile zurück.
[CONTROL]W	**[COPY]**	Ruft die zuletzt eingegebene Befehlszeile wieder auf.
[CONTROL]X	**[CONTROL][DEL]**	Löscht alles vom Anfang der Zeile bis zum Cursor.
[CONTROL]Z		Bezeichnet das Ende des Textes.

Dateinamen

Viele Befehle übernehmen Dateinamen als Parameter. Dabei können auch Universalzeichen darin enthalten sein (vergl. Abschnitt 'Universalzeichen' im ersten Teil dieses Kapitels). Alle Dateinamen werden in Großbuchstaben umgewandelt.

Bei direkten Konsol-Befehlen und in den meisten Dienstprogrammen braucht der Dateiname nicht in Anführungszeichen zu stehen. Denken Sie daran, daß dem Dateinamen `A:` oder `B:` vorangestellt werden kann, damit CP/M das gewünschte Laufwerk verwendet.

Ein typischer CP/M-Befehl sieht also so aus:

```
TYPE KEYS.CCP
```

`TYPE` ist dabei die gewünschte Funktion und heißt soviel wie:

'Stelle auf dem Bildschirm dar', während mit `KEYS.CCP` spezifiziert wird, welche Datei gezeigt werden soll.

Umschaltung des Standardlaufwerks

Wenn Sie ein zweites Diskettenlaufwerk angeschlossen haben, können Sie entweder Laufwerk A oder B zum Standardlaufwerk erklären, indem Sie nach Erscheinen des Bereitschaftszeichens `A>` oder `B>` die Zeichen `B:` bzw. `A:` eingeben. Das Bereitschaftszeichen `A>` oder `B>` gibt das momentane Standardlaufwerk an. Die Angabe `A:` oder `B:` vor einem Dateinamen bestimmt lediglich, für welches Laufwerk die nächste Operation gilt; sie ändert nicht die Standardeinstellung.

Direkte Konsol-Befehle

Es gibt eine Reihe von direkten Konsol-Befehlen (residente Befehle), die nach dem Erscheinen des Bereitschaftszeichens `A>` bzw. `B>` eingegeben werden können. Alle diese Befehle können abgekürzt werden. Während die unten beschriebenen einfachen Funktionen eingebaut sind, gibt es kompliziertere transiente Befehle mit demselben Namen.

DIR

`DIR` listet das Inhaltsverzeichnis einer Diskette auf. Die Dateinamen sind in keiner bestimmten Reihenfolge angeordnet. Die Position des Dateinamens bei der `DIR`-Darstellung ist jedoch ein Hinweis darauf, an welcher Stelle des Inhaltsverzeichnisses die Datei eingetragen ist. Universalzeichen sind erlaubt. Mit `SYS` versehene Dateinamen werden nicht aufgelistet.

`DIR`	Listet alle Dateien des Standardlaufwerks auf.
`DIR B:`	Listet alle Dateien von Laufwerk B auf.
`DIR *.BAS`	Listet alle Dateien vom Typ `.BAS` auf.
`DIR B:*.BAS`	Listet alle Dateien vom Typ `.BAS` von Laufwerk B auf.
`DIR PIP.COM`	Listet nur die Datei `PIP.COM` auf, sofern vorhanden.

DIRSYS oder DIRS

Mit `DIRSYS` oder `DIRS` werden nur mit 'SYS' versehene Einträge des Inhaltsverzeichnisses aufgelistet. Im übrigen funktioniert der Befehl wie `DIR`. Das Attribut `SYS` wird später beschrieben.

ERASE oder ERA

Mit `ERA` werden Dateien im Inhaltsverzeichnis gestrichen. Sie werden jedoch nur aus dem Inhaltsverzeichnis gelöscht, die Daten selbst verbleiben im Datenbereich der Diskette, bis der Platz von einer anderen Datei eingenommen wird. Die Information ist aber trotzdem nicht mehr abrufbar. Dateinamen mit Universalzeichen sind zugelassen. Sollen solche Dateinamen gelöscht werden, wird eine Bestätigung dafür verlangt. `ERA` listet die gelöschten Dateinamen nicht auf. Soll eine Datei gelöscht werden, die als 'nur lesbar' erkannt wird, bricht die Befehlsausführung ab. Das Attribut für 'nur lesbar' wird später beschrieben.

`ERA PIP.COM`	löscht die Datei `PIP.COM`.
`ERA B:PIP.COM`	löscht die Datei `PIP.COM` von Laufwerk B.
`ERA *.BAS`	löscht alle `.BAS`-Dateien.

RENAME oder REN

`REN` dient zur Umbenennung einer bestehenden Datei. Zuerst wird der neue Dateiname angegeben, es folgt ein Gleichheitszeichen `=` und dahinter der alte Name. Existiert bereits eine Datei mit dem neuen Dateinamen, wird eine Fehlermeldung ausgegeben.

Dateinamen mit Universalzeichen können mit dem eingebauten `REN`-Befehl nicht verwendet werden. Hier muß stattdessen mit dem transienten Programm `RENAME.COM` gearbeitet werden.

`REN NEUNAME.BAS=ALTNAME.BAS`	ändert den Namen der Datei `ALTNAME.BAS` in `NEUNAME.BAS` um.
`REN B:NEUNAME.BAS=ALTNAME.BAS`	ändert den Namen der Datei `ALTNAME.BAS` auf Laufwerk B in `NEUNAME.BAS` um.

TYPE oder TYP

TYPE stellt die angegebene Datei auf dem Bildschirm dar. Handelt es sich dabei nicht um eine ASCII-Datei, können unvorhergesehene oder unerwünschte Nebenwirkungen auftreten.

```
TYPE KEYS.CCP
```

zeigt die Datei KEYS.CCP auf dem Bildschirm.

USER oder USE

USER ändert die gegenwärtige Benutzer- (User-)Nummer. Beim Einschalten von CP/M Plus ist die Benutzer-Nummer automatisch 0. Normalerweise kann nur auf Dateien unter der gegenwärtigen Benutzer-Nummer zugegriffen werden. Auf diese Weise wird das Inhaltsverzeichnis in unterschiedliche Bereiche eingeteilt.

Auf eine Datei mit dem Attribut SYS, die sich unter der Benutzer-Nummer 0 befindet, kann von allen anderen Benutzer-Nummern zugegriffen werden. Diese Möglichkeit macht Dienst- und Anwender-Programme allen Benutzern zugänglich, ohne daß jeder Benutzer-Bereich eine Kopie dieser Programme enthalten muß.

```
USER 3
```

schaltet auf den Benutzerbereich 3.

Transiente Befehle

Um komplexere Datei-Bearbeitungen durchzuführen, als sie mit den direkten Konsol-Befehlen möglich sind, können Sie sich der verschiedenen Dienstprogramme bedienen. Diese werden einfach aufgerufen, indem Sie den Programm-Namen und eventuell den Dateinamen und/oder einige Parameter angeben. Eines der Dienstprogramme ist DISCKIT3, mit dem Sie wahrscheinlich schon gearbeitet haben.

Die Befehle fallen unter verschiedene Kategorien, wie unten angegeben. Die komplette Dokumentation dieser Programme ist sehr umfangreich. Weitere Einzelheiten erfahren Sie in den HELP-Dateien (auf Seite 3 der Systemdisketten), sowie im CP/M Plus Handbuch SW 971.

Die Befehle DISCKIT3, SETKEYS, SETLST , SETSIO, PALETTE, LANGUAGE und AMSDOS, sowie die GSX Bildschirm-Steuerbefehle und die Installation von LOGO3 sind ausschließlich für AMSTRAD CPC-Computersysteme konzipiert. Sie funktionieren nicht mit anderen CP/M-Systemen.

Es können mehrere Befehle in einer Zeile untergebracht werden. Sie sind jeweils durch ein Ausrufungszeichen zu trennen. Beispiel:

```
LANGUAGE 2!SETKEYS KEYS.WP
```

Peripherie-Verwaltung

DISCKIT3 ist ein kombiniertes Formatier-, Kopier- und Prüf-Programm. Es geht schneller, wenn die Diskette beim Kopieren formatiert wird, als wenn sie erst formatiert wird, und dann die Daten auf sie kopiert werden. Umfassende Menüs bieten mehrere Möglichkeiten an, die durch Drücken einer Taste (meistens einer Funktionstaste) gewählt werden. Vendor-Format-Disketten sind spezielle System-Disketten für den Software-Vertrieb, obwohl sich Data-Format-Disketten vermutlich besser für den Betrieb in der CP/M-Plus-Umgebung eignen würden.

VORSICHT! Die Lizenz für Ihre CP/M-Diskette (die mit einer elektromagnetischen Serien-Nummer versehen ist) erlaubt ihren Einsatz nur auf einem einzigen Computer-System. Das heißt, die Weitergabe der CP/M-Disketten Ihrer Serien-Nummer (oder einer Kopie davon) an andere Personen ist verboten. Da die Seite 1 Ihrer System-Disketten CP/M (in der .EMS-Datei) enthält, dürfen Sie Disketten, die diese Datei enthalten, weder verkaufen noch tauschen noch irgendwie anders in Umlauf bringen.

Zeichen anderer Sprachen

Der CPC6128 enthält einen vollständigen Satz internationaler Zeichen. Mit dem LANGUAGE-Befehl können bestimmte Zeichen auf der Tastatur ausgetauscht werden, so daß man mit einfacher Software Zeichen anderer Sprachen auf dem Bildschirm darstellen kann. Weitere Informationen dazu finden Sie im 16. Teil des Kapitels 'Übersicht'.

Mit dem Befehl:

```
LANGUAGE 2
```

wird der CPC6128 von amerikanischer auf deutsche Tastatur umgestellt.

Farben

Die Standardfarben für CP/M Plus auf dem 6128 (mit Farbmonitor) sind leuchtendweiße Schrift auf blauem Grund. Mit dem Befehl `PALETTE` können diese Farben geändert werden. Die zugehörigen Parameter geben die `INK`-Farbe an. Der erste Parameter bestimmt die Farbe für Rand und Schriftuntergrund, der zweite Parameter die Schriftfarbe. Jede Farbe ist mit einer Nummer zwischen 0 und 63 versehen. Beim Grün-Monitor ändert sich die Farbintensität (Helligkeit).

Im Prinzip kann jede der 16 Farben gewählt werden; allerdings sind im 80-Spalten-Modus nur die ersten zwei Farben möglich.

Durch den Befehl:

```
PALETTE 63,1
```

wird die normale Farbkombination umgekehrt: blaue (1) Schrift erscheint auf leuchtendweißem (63) Untergrund.

Aus der folgenden Tabelle können Sie die gewünschten Farben (oder Helligkeitsgrade) auswählen. Im Befehl können Sie wahlweise die hexadezimale oder dezimale Schreibweise verwenden.

Farbe	hex	dezimal	Farbe	hex	dezimal
schwarz	&00	0	pastellblau	&2B	43
blau	&02	2	orange	&2C	44
leuchtend blau	&03	3	rosa	&2E	46
rot	&08	8	pastellmagenta	&2F	47
magenta	&0A	10	leuchtend grün	&30	48
mauve	&0B	11	seegrün	&32	50
leuchtend rot	&0C	12	leuchtend blaugrün	&33	51
purpur	&0E	14	limonengrün	&38	56
leuchtend magenta	&0F	15	pastellgrün	&3A	58
grün	&20	32	pastell blaugrün	&3B	59
blaugrün	&22	34	leuchtend gelb	&3C	60
himmelblau	&23	35	pastellgelb	&3E	62
gelb	&28	40	leuchtend weiß	&3F	63
weiß	&2A	42			

Tastatur

Die von der Tastatur erzeugten Codes können mit dem Befehl SETKEYS geändert werden. So kann man Tasten und Erweiterungszeichen mit geeigneten Codes versehen. Die Codes selber müssen in die Datei geschrieben werden, dessen Name dann im SETKEYS-Befehl angegeben wird. Die Befehlsdatei kann mit einem Text-Editor, mit PIP oder mit BASIC erstellt werden. Nehmen wir als Beispiel:

```
SETKEYS KEYS.TST
```

KEYS.TST enthält:

E &8C "DIR ↑M" Erweiterungszeichen 12
8 N S C "↑H" Rückwärts-Taste=**[CONTROL]H**,ASCII 08

Der Befehl SETKEYS KEYS.TST definiert erstens das **[CONTROL] [ENTER]**-Erweiterungszeichen (dargestellt durch &8C) in den String DIR **[RETURN]** um, und ändert zweitens die Cursor-links-Taste (Tasten-Nummer 8) in eine Rückwärts-Taste um.

Mitgelieferte Standard-Dateien für den 6128 sind KEYS.CCP zur Korrektur von CP/M-Befehlen, KEYS.DRL für den Einsatz von Dr.LOGO (auf Seite 3) und KEYS.WP, die meist beim Einsatz von Textverarbeitung geeignet ist.

Drucker

Drucker können mit folgendem Befehl initialisiert werden:

```
SETLST ‹Dateiname›
```

wobei ‹Dateiname› den String oder die Strings einhält, die zum Drucker gesendet werden sollen. Wie bei der Befehlsdatei für SETKEYS können Steuer-Codes durch:

```
↑ ‹Zeichen›
```

oder:

```
↑ '‹Zeichenwert›'
```

oder:

↑ '‹Steuer-Code-Name›'

dargestellt werden, wobei Steuer-Code-Namen ESC, FF usw. sind (siehe Tabelle der ASCII-Zeichen im Kapitel 'Übersicht').

Für viele Drucker ist 15 ein nützlicher Initialisierungs-Code. Er stellt den Drucker auf ein enges Schriftbild ein.

In BASIC würde der Befehl so aussehen:

```
PRINT #8,CHR$(15)
```

In CP/M geben Sie folgenden Befehl aus:

```
SETLST CONDENSE
```

Die Datei **CONDENSE** enthält einen der folgenden Ausdrücke in einer Textzeile:

```
↑'SI'
↑'O
↑'&F'
↑'15'
```

die alle als Dezimalwert 15 interpretiert werden.

Einige Anwender-Programme setzen einen Bildschirm von 24x80 Zeichen voraus. Mit dem Befehl **SET24X80** wird die Bildschirmgröße darauf eingestellt.

Die Befehle:

```
SET24X80
```

oder:

```
SET24X80 ON
```

setzen den Bildschirm-Modus auf 24x80 Zeichen, und:

```
SET24X80 OFF
```

schaltet den 24x80-Modus ab.

Die normale Bildschirmgröße des 6128 ist 24x80, wobei die unterste Zeile für Status-Meldungen reserviert bleibt. Ist der 24x80-Modus abgeschaltet, zeigt sich dies nur daran, daß die Status-Zeile außer Kraft gesetzt ist. Im 15. Teil von Kapitel 7 finden Sie Näheres darüber, wie die Status-Zeile an- und abgeschaltet wird.

Serielles Interface

In CP/M Plus ist eine Unterstützung für ein einkanaliges serielles Eingabe/Ausgabe-Interface (RS232) eingebaut. Mit dem Befehl:

```
SETSIO
```

(ohne Parameter) können seine wichtigsten Statistiken untersucht werden. Sie können auch durch einen Befehl eingestellt werden, der aus einigen (oder allen) der folgenden Angaben bestehen kann:

```
SETSIO, RX 1200, TX 75, PARITY NONE, STOP 1, BITS 8,
HANDSHAKE ON, XOFF OFF
```

Auf diese Weise wird eine neue Konfiguration hergestellt.

Baud-Raten und XON/XOFF-Status können auch durch Zuordnungen beeinflußt werden, die mit dem Befehl DEVICE vorgenommen werden. DEVICE verknüpft logische und physikalische Elemente. Logische Elemente werden durch einen Doppelpunkt angezeigt. Um alle gegenwärtigen Element-Attribute zu untersuchen, schreiben Sie:

```
DEVICE
```

Die Attribute könne durch folgende Befehle verändert werden:

```
DEVICE SIO[1200]    setzt SIO auf 1200 Baud
DEVICE SIO[XON]     schaltet SIO XON/XOFF Protokoll an.
DEVICE SIO[NOXON]   schaltet SIO XON/XOFF Protokoll ab.
```

Die Verknüpfungen zwischen logischen und physikalischen Geräten können verändert werden. Normalerweise ist CON: mit CRT (Tastatur/Bildschirm), AUX: mit SIO (der optionalen seriellen Schnittstelle) und LST: mit LPT (der Centronics Drucker-Schnittstelle) verbunden. Mit dem Befehl:

```
DEVICE LST:=SIO
```

...wird die Drucker-Ausgabe zur seriellen Schnittstelle (sofern angeschlossen) geschickt.

Beachten Sie, daß es sich hierbei um die Umstellung auf einen anderen Kanal handelt, was nicht mit dem Datei-Kopiervorgang mittels PIP zu verwechseln ist. Die beiden Befehle GET ‹Datei› und PUT‹Datei› leiten die Konsolen-Eingabe bzw. -Ausgabe und die Drucker-Ausgabe um, wobei eine Datei und kein Geräte-Kanal benutzt werden soll.

PIP

Das Dienstprogramm PIP (Peripheral Interchange Program) ermöglicht den Transfer von Informationen zwischen dem Computer und seiner Peripherie. Die allgemeine Form des Befehls lautet:

```
PIP‹Ziel›=‹Quelle›
```

‹Ziel› und ‹Quelle› können entweder Dateinamen - in ‹Quelle› sind Universalzeichen erlaubt - oder logische Geräte sein. Folgende logische Geräte können verwendet werden:

Als Quelle:		Als Ziel:	
CON:	Konsolen-Eingabe	CON:	Konsolen-Ausgabe
AUX:	Zusatzgerät-Eingabe	AUX:	Zusatzgerät-Ausgabe
EOF:	Markierung am Ende der Datei	LST:	Drucker
		PRN:	Ein Drucker mit Tabulator-Erweiterung, Zeilen-Numerierung und Seiteneinteilung

Beispiele:

```
PIP B:=A:*.COM
```

...kopiert alle .COM-Dateien von Laufwerk A: nach Laufwerk B: .

```
PIP KEYBOARD.CPM=KEYS.CCP
```

...macht eine Kopie von KEYS.CCP und nennt sie KEYBOARD.CPM.

```
PIP CON:=KEYS.CCP
```

...gibt die Datei KEYS.CCP auf dem Bildschirm aus (ähnlich dem Befehl TYPE KEYS.CCP.

```
PIP LST:=KEYS.CCP
```

...gibt die Datei KEYS.CCP auf dem Drucker aus.

```
PIP TYPEIN.TXT=CON:
```

...übernimmt die Tastatur-Eingabe und speichert sie in der Datei TYPEIN.TXT ab.

Beachten Sie, daß die letzte Operation mit dem Steuercode **[CONTROL]Z** abgeschlossen wird. Nach jedem **[RETURN]** muß für eine neue Zeile **[CONTROL]J** eingetippt werden. **[CONTROL]J** ist der ASCII-Code für den Zeilenvorschub.

Wird PIP ohne Parameter eingetippt, erscheint das Bereitschaftszeichen * und Sie können die gewünschten Befehle eingeben. Diese Form der Operation ist besonders nützlich, wenn Dateien kopiert werden sollen, ohne daß die PIP.COM-Datei auf der Quellen- oder Ziel-Diskette enthalten ist. Wir können PIP von Seite 1 der Systemdisketten laden, die Systemdiskette herausnehmen und anschließend die Disketten einlegen, die zum Kopieren gebraucht werden.

Möchten Sie das PIP-Programm verlassen, drücken Sie **[RETURN]**, wenn das Bereitschaftszeichen * erscheint.

Dateien können auch dann mit PIP von einer Diskette auf eine andere kopiert werden, wenn nur ein Laufwerk verwendet wird. In dem Fall erscheinen automatisch die entsprechenden Bereitschaftszeichen zum Diskettenwechsel. Die Laufwerks-Bezeichnungen für Quellen- und Zieldiskette müssen sich unterscheiden.

System-Verwaltung

DIR, ERASE, RENAME und TYPE sind transiente Programme, die mehr Möglichkeiten bieten als die entsprechenden eingebauten Programme.

Wie bei vielen anderen transienten Programmen von Digital Research werden sekundäre Parameter in eckigen Klammern angegeben. In den HELP-Dateien (auf Seite 3 Ihres System-Diskettensatzes) finden Sie hierzu alle Einzelheiten. Beispiele:

`DIR [FULL]`	Anzeige von Datei-Größen und -Attributen.
`ERASE *.COM [CONFIRM]`	Für jede einzelne Datei wird eine Bestätigung zu Löschen verlangt.
`RENAME`	Alte und neue Dateinamen müssen eingegeben werden.
`RENAME *.SAV=*.BAK`	Alle Dateien vom Typ .BAK werden dem Typ .SAV zugeordnet.
`TYPE KEYS.WP [NOPAGE]`	Der Seiten-Stop auf dem Bildschirm wird unterdrückt.

Die Datei-Attribute SYS (System) und RO (Read/Only = nur lesbar) wurden schon vorher erwähnt. Solche Attribute können mit dem Befehl SET (der auch Universalzeichen akzeptiert) zugeordnet werden.

Die Befehle:

```
SET *.COM [RO]
SET KEYS.CCP [RO]
SET A: [RO]
```

...versetzen Dateien oder ein Laufwerk in den Read/Only-Zustand, um sie vor versehentlichem Löschen zu schützen.

Mit den Befehlen:

```
SET *.COM [RW]
SET KEYS.CCP [RW]
SET A: [RW]
```

...werden Dateien oder Laufwerk in den Read/Write- (Schreib-Lese-) Zustand zurückversetzt.

Die Befehle:

```
SET *.COM [SYS]
SET KEYS.CCP [SYS]
```

...versehen Dateien mit dem System-Attribut. Dateien mit einem solchen Attribut werden nach dem **DIR**-Befehl nicht angezeigt. (Dazu ist **DIRS** oder **DIRSYS** erforderlich.) Trotzdem stehen diese Dateien noch zur Bearbeitung zur Verfügung. Wenn sie sich im Benutzer-Bereich 0 befinden, sind sie darüberhinaus von allen anderen Benutzern aufrufbar.

Die Befehle:

```
SET *.COM [DIR]
SET KEYS.CCP [DIR]
```

...entfernen das System-Attribut.

Jede Diskette kann mit einem Etikett (**NAME**) versehen werden, oder mit einem Kennwort (**PASSWORD**). Das Kennwort schützt das Inhaltsverzeichnis selber. Den einzelnen Dateien kann ebenfalls ein Kennwort zugeordnet werden.

```
SET [NAME=ROLAND]
SET [PASSWORD=SALLY]
SET [PROTECT=ON]
```

...bezieht sich auf die Diskette im Standard-Laufwerk.

```
SET *.*[PASSWORD=SALLY]
SET *.*[PROTECT=READ]
```

...bezieht sich auf die Dateien der Diskette im Standard-Laufwerk. (Die Universalzeichen *.* stehen für 'alle Dateien'.)

Der Befehl `INITDIR` (auf Seite 2 der Systemdisketten) versieht die Dateien im Standard-Laufwerk mit Datum und Uhrzeit. Jedesmal, wenn CP/M gestartet wird, muß

```
DATE SET
```

eingetippt werden, um die Uhr zu stellen. Ist sie einmal gestellt, wird sie einigermaßen richtig laufen. Der 6128 stellt automatisch das richtige Datum ein. Mit den Befehlen `DATE` und `DATE CONTINUOUS` können Sie das Datum abfragen.

ACHTUNG! Disketten, die mit Kennworten oder Etiketten versehen oder mit automatischer Datierung ausgerüstet wurden, sollten danach NIEMALS von AMSDOS oder CP/M 2.2 beschrieben werden, da diese Möglichkeiten von diesen beiden Betriebssystemen nicht unterstützt werden.

Im Normalfall wird immer auf die Diskette im Standard-Laufwerk zugegriffen, wenn kein Laufwerk spezifiziert wurde. Der Befehl:

```
SETDEF *,A:
```

...(in dem * für das Standard-Laufwerk steht) weist CP/M an, eine Datei erst auf dem Standard-Laufwerk und dann auf Laufwerk `A:` zu suchen. Ist `B:` Das Standard-Laufwerk, werden die Dateien automatisch auch dann gefunden, wenn sie sich auf Laufwerk `A:` befinden.

Mit den Befehlen:

```
SETDEF [PAGE]
```

und

```
SETDEF [NOPAGE]
```

...wird der automatische Seiten-Stop auf der Konsolen-Anzeige ein- bzw. abgeschaltet.

Denken Sie daran, daß die meisten `DEVICE`-, `SET`- und `SETDEF`-Befehle, besonders, wenn sie sich auf Laufwerke und nicht auf Dateien oder Disketten beziehen, jedesmal zusammen mit dem Datum eingegeben werden müssen, wenn CP/M Plus gestartet wird. Diese Information wird am besten in einer geeigneten `PROFILE.SUB`-Datei zusammengestellt.

SUBMIT wird verwendet, um Dateien mit einzelnen Befehlen automatisch ablaufen zu lassen. Der Inhalt der Befehls-Dateien besteht aus Text und wenn eine Befehlszeile einer .SUB-Datei mit < beginnt, kann das Programm auch Eingabe-Zeilen enthalten.

Der Befehl SHOW in seinen verschiedenen Ausprägungen informiert über die Laufwerks-Kapazität, den Speicherplatz, die Anzahl der freien Plätze im Inhaltsverzeichnis, die Benutzer-Bereiche der vorhandenen Dateien und das Etikett der Diskette (sofern vorhanden). Die folgenden Beispiele beziehen sich alle auf Laufwerk B: .

```
SHOW B:
SHOW B:[LABEL]
SHOW B:[USERS]
SHOW B:[DIR]
SHOW B:[DRIVE]
```

Rückkehr zu BASIC

Der Befehl AMSDOS beendet die Steuerung durch CP/M und übergibt sie an das eingebaute AMSTRAD-BASIC, unter dem die AMSDOS-Disketten-Befehle anwendbar sind.

Programmieren für Fortgeschrittene

Seite 2 des System-Diskettensatzes enthält einige Programme für fortgeschrittene und System-Programmierer. Wir empfehlen, hierfür das CP/M Plus-Handbuch SW 971 oder ähnliche Unterlagen zu Rate zu ziehen.

Wenn Sie mit CP/M 2.2 arbeiten

Dieser Abschnitt stellt die Besonderheiten beim Betrieb mit CP/M 2.2 heraus.

CP/M 2.2 wird von den beiden äußersten Spuren der System-Diskette geladen. Der Urlader-Bereich von CP/M 2.2 liegt an anderer Stelle als der von CP/M Plus, und die beiden sollten tunlichst nicht verwechselt werden. Es ist zwar möglich, Vendor- IBM- und Datenformat-Disketten in beiden Diskettenlaufwerken zu betreiben, aber aus Betriebsgründen wird sich ihr Gebrauch auf das zweite Laufwerk beschränken.

Wenn keine besonderen Vorkehrungen durch das CP/M-Programm getroffen wurden (wie z.B. bei FILECOPY), erlaubt CP/M 2.2 solange keine Schreibzugriffe auf Diskette, bis sie dem System durch ein 'Log-in' bekannt ist. Der Format-Typ der Disketten (System-, Daten- oder IBM-Diskette) ist ebenfalls nur nach dem 'Log-in' bestimmbar. Diese Aufbereitung findet für das eingebaute Laufwerk (Drive A) immer dann statt, wenn CP/M in den direkten Konsol-Modus zurückkehrt oder wenn **[CONTROL]C** nach dem Erscheinen von A> bzw. B> eingegeben wird. Für das Zusatz-Laufwerk (Drive B) erfolgt das Log-In nach dem ersten Zugriff, sofern für Laufwerk A bereits ein Log-In durchgeführt wurde.

Wenn die Diskette ohne vorheriges Log-In beschrieben wird, erscheint die berüchtigte Fehlermeldung:

```
Bdos Err on ‹Laufwerk›: R/O
```

Zur Fortsetzung drücken Sie eine beliebige Taste. Wenn die ausgewechselte Diskette ebenfalls ein anderes Format besitzt, tritt ein Lese- oder Schreibfehler auf. Dann geben Sie C ein.

Wenn Sie Software auf einer Diskette im Vendor-Format haben, können Sie sie zu ihrer besseren Verwendung entweder mit FILECOPY oder PIP auf eine CP/M 2.2 System-Diskette kopieren, oder die Diskette in eine System-Diskette verwandeln, indem Sie Ihr CP/M daraufkopieren. Zum diesem Zweck stehen Ihnen die Befehle BOOTGEN und SYSGEN zur Verfügung. Lesen Sie dazu bitte den Abschnitt über das Endverbraucher-Lizenzabkommen im Anhang 1 aufmerksam durch.

SYSGEN (ohne Parameter) ist ein spezielles Kopierprogramm, das nach der Quellen- und Ziel-Diskette fragt, und die CP/M 2.2-System-Spuren von einer Diskette auf die andere kopiert. BOOTGEN kopiert in ähnlicher Weise Sektor 1, Spur 0 (den Lader) und das Konfigurations-Programm von einer Diskette auf die andere.

Der DIR-Befehl hat (außer einem Dateinamen) keine Parameter. Die Dateinamen sind nicht geordnet, sondern erscheinen in der Reihenfolge, wie sie in das Inhaltsverzeichnis eingetragen wurden.

STAT besitzt ähnliche Funktionen wie SET und SHOW. Die Befehle:

```
STAT
STAT A:
STAT B:
```

...zeigen Disketten-Status und freien Speicherplatz an.

Die Befehle:

```
STAT *.COM
STAT DISC.BAS
```

...geben ausführlich Auskunft über den Eintrag einer bestimmten Datei im Inhaltsverzeichnis.

Die Befehle:

```
STAT *.COM $R/O
STAT DISC.BAS $R/O
```

...versetzen eine Datei in den Read/Only (nur lesbaren) Status, so daß sie nicht versehentlich überschrieben werden kann.

Die Befehle:

```
STAT *.COM $R/W
STAT DISC.BAS $R/W
```

...stellen den Read/Write (schreib- und lesbaren) Status einer Datei wieder her.

Die Befehle:

```
STAT *.COM $SYS
STAT SECRET.BAS $SYS
```

...setzen eine Datei in den 'System'-Zustand, so daß sie beim Auflisten des Inhaltsverzeichnisses und bei Datei-Kopierprogrammen unsichtbar bleibt. Für alle anderen Zwecke bleibt die Datei jedoch zugänglich.

Die Befehle:

```
STAT *.COM $DIR
STAT SECRET.BAS $DIR
```

...versetzen eine Datei wieder vom 'System'-Zustand in den 'Inhaltsverzeichnis'-Zustand.

Mit dem Dienstprogramm `FILECOPY` können Dateien unter Benutzung nur eines Laufwerks von einer Diskette auf eine andere kopiert werden. Hierzu müssen die Disketten entsprechend den Anweisungen auf dem Bildschirm gewechselt werden. Wird ein Dateiname mit Universalzeichen angegeben, fragt `FILECOPY`, ob tatsächlich jede Datei, die unter diesen Namen fällt, kopiert werden soll. Das Programm zeigt jeweils den Namen der Datei an, die gerade kopiert wird.

```
FILECOPY *.COM          ...kopiert alle Dateien vom Typ .COM.
FILECOPY PIP.COM        ...kopiert die Datei PIP.COM.
```

DISCKIT2 erfüllt dieselbe Funktion wie DISCKIT3, mit dem Unterschied, daß es etwas langsamer kopiert, da es weniger Speicherplatz zur Verfügung hat.

Zwei Dienstprogramme ermöglichen den Datei-Transfer zwischen Diskette und Kassette. Mit Ausnahme spezieller Anwendungen ist es recht unwahrscheinlich, daß jemals andere als ASCII-Dateien übertragen werden.

CLOAD (Cassette LOAD) kann zwei Parameter annehmen. Der erste ist der Quellen- (Kassetten-) Dateiname, der in Anführungsstrichen stehen muß, der zweite ist der Ziel- (Disketten-) Dateiname. Wenn der Ziel-Dateiname ausgelassen wird, erhält die Disketten-Datei den Namen der Kassetten-Datei. Wird der Quellen-Dateiname ausgelassen, so liest CLOAD die erste Datei, die es auf der Kassette findet. Die normalen Bildschirm-Hinweise zum Kassettenbetrieb entfallen, wenn der Kassetten-Dateiname mit einem Ausrufungszeichen ! beginnt.

Beispiel:

```
CLOAD "MY LETTER" MYLETTER.TXT
```

CSAVE (Cassette SAVE) kann drei Parameter annehmen. Der erste ist der Quellen- (Disketten-) Dateiname, der zweite ist der Ziel- (Kassetten-) Dateiname, der in Anführungsstrichen stehen muß. Wenn der Ziel-Dateiname ausgelassen wird, erhält die Kassetten-Datei den Namen der Disketten-Datei. Die normalen Bildschirm-Hinweise zum Kassettenbetrieb entfallen, wenn der Kassetten-Dateiname mit einem Ausrufungszeichen ! beginnt. Sind beide Dateinamen angegeben, kann ein dritter Parameter zur Bestimmung der Band-Schreibgeschwindigkeit (0 für 1000 Baud, 1 für 2000 Baud) verwendet werden.

Beispiele:

```
CSAVE OUTPUT.TXT "OUTPUT TEXT" 1
CSAVE DATEN
```

SETUP

Mit diesem Dienstprogramm können die Eigenschaften der CPC6128-Tastatur, des Disketten-Laufwerks und der seriellen Schnittstelle umdefiniert werden. Außerdem können damit beim Laden von CP/M 2.2 verschiedene Funktionen aufgerufen werden. Im Gegensatz zu den Dienstprogrammen von CP/M Plus, die ihre Funktion bei der Ausführung des Programms erfüllen, verändert SETUP den Konfigurationssektor einer Diskette (der nur beim Kalt-Start einer Diskette aufgerufen wird). In dieser Hinsicht ähnelt es dem PROFILE.SUB-Dienstprogramm.

Das SETUP-Programm ist Menü-gesteuert. Wenn eine bestimmte Aussage auf dem Bildschirm richtig ist, oder nicht geändert werden soll, wird zur nächsten Frage umgeschaltet, indem auf die Frage:

```
Is this correct (Y/N):_
```

...ein Y eingetippt wird.

Das Programm kann mit **[CONTROL]C** abgebrochen werden. Wenn alle Änderungen durchgeführt sind, erscheint am Ende:

```
Do you want to update your system disc (Y/N):_
```

An dieser Stelle haben Sie die Möglichkeit, die bisherige Konfiguration beizubehalten, indem Sie N eintippen. Haben Sie mit Y geantwortet, werden Sie gefragt:

```
Do you want to restart CP/M (Y/N):_
```

Damit haben Sie die Möglichkeit, durch Eingabe von Y die neue Konfiguration zu laden und zu testen.

Um einen Konfigurations-Sektor auf eine andere Diskette zu kopieren, verwenden sie BOOTGEN oder laden SETUP von der Quellen-Diskette, beantworten alle Fragen mit Y und legen vor Beantwortung der Frage 'Do you want to update your system disc (Y/N):_' die Ziel-Diskette ein.

Zeichen mit einem ASCII-Wert kleiner dezimal 32 können in einen String integriert werden, indem ↑ gefolgt von einem geeigneten Zeichen aus dem Satz @,A-Z,[,\,],>,_ eingegeben wird.

Die folgenden Optionen verlangen besondere Aufmerksamkeit:

```
** Initial Command Buffer
```
(Befehls-Puffer)

Alle hier eingegebenen Zeichen erscheinen, als wären sie beim ersten Laden von CP/M unter dem direkten Konsol-Modus eingegeben worden. Dies führt zum automatischen Start eines bestimmten Programms. Denken Sie daran, statt der **[RETURN]**-Taste die beiden Zeichen ↑ M zu verwenden.

Zum Automatik-Start von DIR sollte der Befehls-Puffer also:

```
DIR↑M
```

...enthalten. Und zum Automatik-Start des Dr.LOGO-Programms müßte der Befehlspuffer:

```
SUBMIT LOGO2↑M
```

...enthalten.

`Sign-on string`

Dies ist die beim ersten Laden von CP/M am oberen Bildschirmrand erscheinende Meldung. Für Wagenrücklauf und Zeilenvorschub ist ↑J ↑M zu verwenden. Der erste Teil der Standard-Meldung setzt geeignete Bildschirm- und Bildschirmrand-Farben für den 80-Spalten-Modus fest und sollte genau kopiert werden, wenn er beibehalten werden soll.

`Keyboard translations`

Hierdurch können den Tasten neue ASCII-Werte zugeordnet werden, ähnlich dem BASIC-Befehl **KEYDEF**. Die erforderlichen Parameter sind die Tasten-Nummer und der zugehörige ASCII-Wert. Eine Aufstellung der Tasten-Nummern finden Sie rechts vom Tastenfeld auf Ihrem Computer oder im 4.Teil des Kapitels 'Übersicht'.

`Keyboard expansions`

Gleicher Effekt wie beim BASIC-Befehl **KEY**.

Nachsatz

Die Befehle **DISCKIT2**, **SYSGEN**, **BOOTGEN**, **FILECOPY**, **SETUP**, **CSAVE** und **CLOAD** sind ausschließlich für den Betrieb mit CP/M 2.2 auf AMSTRAD CPC-Computer-Systemen konzipiert. Sie funktionieren auf keinem anderen CP/M-System, obgleich andere Hersteller ähnliche Dienstprogramme (oft mit gleichen Namen) für ihre Hardware anbieten.

Die folgenden CP/M 2.2-Programme auf Seite 4 dienen speziellen Anwendungen. Wir empfehlen, hierfür das CP/M Plus-Handbuch SW 971 oder ähnliche Veröffentlichungen zu Rate zu ziehen.

ASM	8080 Assembler
DDT	8080 Assembler-Code-Testhilfe
DUMP	Dienstprogramm für hexadezimale Datei-Ausgabe
ED	Einfacher Text-Editor
LOAD	Verwandelt eine durch **ASM** erstellte **.HEX**-Datei in eine **.COM**-Datei
MOVCPM	Erstellt eine neue Version von CP/M 2.2 in anderem (kleinerem) TPA-Format.
SUBMIT	Stapel-Verarbeitung (Batch) aus dem Konsol-Modus
XSUB	Stapel-Verarbeitung (Batch) durch transiente Programme.

Diese Programm werden mit CP/M 2.2 (Seite 4) betrieben, und sollten nicht mit CP/M Plus-Programmen auf den Seiten 1, 2 und 3 verwechselt werden.

Kapitel 6
Einführung in LOGO

Dieses Kapitel führt anhand von Beispielen in die Programmiersprache LOGO ein und bietet eine Übersicht über die verfügbaren Befehle. Es sollte jedoch nicht als erschöpfendes Lehrbuch oder Nachschlagewerk betrachtet werden. Hierfür verwenden Sie besser den LOGO-Führer SW160.

Folgende Themen werden behandelt:

- ★ Der Begriff LOGO
- ★ Laden und Programmablauf von Dr.LOGO
- ★ Schildkröten-Graphik
- ★ Schreiben eigener Prozeduren
- ★ Editieren eigener Prozeduren

Was ist LOGO?

LOGO kann aus Ihnen einen guten Programmierer machen, gleichgültig, ob Sie schon Programmiererfahrung haben oder nicht. LOGO ist eine leistungsfähige Programmiersprache, die aufgrund ihrer leichten Erlernbarkeit schnell populär wurde.

Sie benutzen Prozeduren als Bausteine für die Erstellung von LOGO-Programmen. Dr.LOGO selbst besteht aus einer Sammlung von Grundprozeduren (Primitives), die zum Aufbau eigener Programme verwendet werden können.

Während der 70er Jahre entwickelte ein Team von Computer-Spezialisten unter der Leitung von Seymour Papert die Programmiersprache LOGO mit Schildkröten-Graphik, um Kindern den Umgang mit Computern zu erleichtern.

Sie entwickelten die Schildkröte, damit Kinder ein 'Denk-Objekt' (Papert) haben, sozusagen ein Werkzeug zum Lernen auf Computern. In Form einer Pfeilspitze kann die Schlidkröte mit einfachen Befehlen über den Bildschirm geführt werden.

Dr.LOGO

Dr. LOGO ist eine besondere LOGO-Version, die an den AMSTRAD Personal Computer angepaßt wurde, um das Programmieren zu erleichtern. Es wurden umfangreiche Erweiterungen eingebaut, um die großartigen Sound-Möglichkeiten des CPC6128 auszunutzen. Die Programm-Bearbeitung wurde durch das Einbeziehen der Cursortasten erleichtert.

Start von Dr. LOGO

Um mit Dr.LOGO zu arbeiten, legen Sie zunächst eine Kopie der Seite 1 Ihrer Systemdisketten in das Laufwerk und schreiben:

```
|CPM
```

Sobald das Zeichen A> erscheint, nehmen Sie die Diskette wieder heraus und legen Seite 3 (mit den Programmen DR.LOGO und HELP) ein. Um das Programm Dr.LOGO zu starten, schreiben Sie nun:

```
SUBMIT LOGO3
```

Nach wenigen Sekunden erscheint der Begrüßungstext mit dem Bereitschaftszeichen ?.

Start von Dr. LOGO mit CP/M 2.2

HINWEIS: Auf Seite 4 der Systemdisketten finden Sie eine andere Dr.LOGO-Version für den Fall, daß Sie mit dem CP/M 2.2 Betriebssystem arbeiten, das für die Modelle AMSTRAD CPC664 und CPC464+DDI1 verwendet wird. Im allgemeinen ist nicht zu empfehlen, diese Version auf dem CPC6128 laufen zu lassen. Die Version von Seite 3 entspricht vollständig dem Dr.LOGO von Digital Research.

Falls Sie jedoch aus irgenwelchen Gründen mit dem Dr.LOGO für CP/M 2.2 arbeiten möchten, schieben Sie Seite 4 ins Laufwerk und schreiben:

```
|CPM
```

Sobald das Zeichen A> erscheint, schreiben Sie:

```
SUBMIT LOGO2
```

Nach wenigen Sekunden erscheint der Begrüßungstext mit dem Bereitschaftszeichen ?.

Erste Versuche

Das Prompt ? bedeutet, daß Dr.LOGO darauf wartet, daß Sie etwas eintippen.

Tippen Sie (in Kleinbuchstaben):

```
fd 60
```

ein, und es erscheint eine Schlidkröte (ein großer Pfeil), die 60 Einheiten vorwärts marschiert und dabei von ihrem Startpunkt bis zum Endpunkt eine Linie zeichnet. Der Bildschirm wird daraufhin gelöscht und in einen großen Graphik- und einen kleinen Text-Bereich unterteilt. Letzterer beginnt mit dem ? am unteren Bildschirmrand.

Dr.LOGO wird des öfteren die Bildschirmbereiche neu einteilen, um je nach Ihren Bedürfnissen den Text- oder Graphik-Bereich zu vergrößern.

Schreiben Sie:

```
rt 90
```

Die Schildkröte dreht sich jetzt um 90 Grad nach rechts. Nun schreiben Sie:

```
fd 60
```

...woraufhin eine genauso lange Linie im rechten Winkel zur ersten gezeichnet wird.

Experimentieren Sie mit den einfachen Anweisungen `fd` (Abkürzung für 'forward'), `bk` ('back'), `rt` ('right') und `lt` ('left') und beobachten Sie, was auf dem Bildschirm geschieht.

Dr. LOGO Prozeduren

Eine Prozedur ist eine Folge von Befehlen, die Dr.LOGO anweisen, eine bestimmte Aufgabe auszuführen.

Sie werden Ihre ersten Prozeduren schreiben, indem Sie eingebaute Dr.LOGO-Grundprozeduren zusammenfügen.

`fd`, `bk`, `rt` und `lt` sind fertige Grundprozeduren (Primitives), die Sie jederzeit als Bausteine für Ihre eigenen Prozeduren verwenden können.

Eine weitere sehr nützliche Grundprozedur ist `cs` ('clear screen'), die den Bildschirm löscht und die Schildkröte an ihren Ausgangspunkt zurücksetzt.

Schreiben einer einfachen Prozedur

Es ist leicht zu erkennen, daß die Anweisungen:

```
fd 60 rt 90
```

...ein Quadrat mit der Seitenlänge 60 ergeben, wenn sie viermal wiederholt werden.

Derselbe Effekt kann durch die einfache Formel erzielt werden:

```
repeat 4 (fd 60 rt 90)
```

Löschen Sie den Bildschirm, geben Sie diese Formel ein und beobachten Sie, was geschieht.

Um aus dieser Formel eine neue Prozedur mit dem Namen 'quadrat' zu machen, tippen Sie:

```
to quadrat
repeat 4 (fd 60 rt 90)
end
```

Dr.LOGO wird von nun ab verstehen, was 'quadrat' bedeutet und jedesmal ein Quadrat zeichnen, wenn er auf das Wort 'quadrat' trifft. Wir hätten dieser Prozedur jeden beliebigen Namen geben können. Es wurde jedoch 'quadrat' gewählt, um daran zu erinnern, was diese Prozedur bewirkt.

Dr.LOGO ermöglicht die Eingabe einer ganzen Serie von Befehlen. So werden bei:

```
quadrat rt 45 quadrat
```

...zwei um 45 Grad gegeneinander versetzte Quadrate gezeichnet.

Prozeduren mit Parametern

Genauso wie wir bei eingebauten Dr.LOGO-Prozeduren angeben können, wie oft sie wiederholt werden sollen, können wir auch selbst Prozeduren erstellen, die beliebig oft wiederholbar sind. Um eine Prozedur zur Darstellung von Quadraten mit unterschiedlicher Seitenlänge zu erstellen, kann die Definition von 'quadrat' folgendermaßen abgeändert werden:

```
to verschiedenequadrate :seite
repeat 4 (fd :seite rt 90)
end
```

Diese neue Prozedur führt eine Variable mit dem Namen `:seite` ein. Der Doppelpunkt vor dem Variablennamen ist ein Unterscheidungsmerkmal. Er gibt an, daß es sich um eine Variable und keinen Befehl handelt.

Wenn wir die Prozedur `verschiedenequadrate :seite` benutzen, brauchen wir für die Variable `:seite` einen Wert. Der Befehl `verschiedenequadrate 150` wird also ein Quadrat mit einer Seitenlänge von 150 Einheiten zeichnen.

Versuchen Sie, zwei Prozduren zusammenzufügen, und sehen Sie, was passiert. Geben Sie dazu beispielsweise ein:

```
cs verschiedenequadrate 100 rt 45 verschiedenequadrate 150
```

Die Schildkröte zeichnet zwei verschieden große Quadrate, die im Winkel von 45 Grad gegeneinander versetzt sind.

Beachten Sie, wie Dr.LOGO Sie mit einem Ausrufungszeichen ! darauf aufmerksam macht, daß die Folge von Befehlen über die Bildschirmzeile hinausgeht.

Anwendung von Variablen

Dr.LOGO ermöglicht durch die Verwendung von Variablen sowohl die Übergabe von Werten an Prozeduren, als auch die Abfrage dieser Werte.

Definieren Sie zunächst eine Prozedur mit dem Namen 'dreieck':

```
to dreieck
repeat 3 (fd :kante rt 120)
end
```

Wir können diese Prozedur testen, indem wir

```
make "kante 100
dreieck
```

...eingeben. Wenn wir den Wert von `:kante` wissen wollen, brauchen wir nach der Prompt-Meldung ? nur `:kante` einzugeben, und Dr.LOGO gibt den Wert auf dem Bildschirm aus.

Schließlich soll die Variable `:kante` in einer Prozedur verwendet werden, um ein Muster zu zeichnen. Beachten Sie, wie der Wert der Variablen, und somit das Muster, in jedem Durchgang größer wird.

```
to muster
dreieck lt 60 dreieck rt 60
make "kante :kante + 4
muster
end
make "kante 10
cs muster
```

Wenn Sie genug gesehen haben, drücken Sie **[ESC]**, um das Programm zu stoppen.

Editieren von Programmen und Prozeduren

Dr. LOGO ermöglicht die Korrektur von Eingabefehlern und die Änderung von selbstdefinierten Prozeduren. Folgende Editier-Tasten stehen hierfür zur Verfügung:

- die Cursor-Tasten →←↑↓, die den Cursor um jeweils eine Stelle weiterbewegen.

- Werden die Cursor-Tasten gleichzeitig mit der **[CONTROL]**-Taste gedrückt, wird der Cursor zum Anfang oder Ende der Seite, bzw. zum linken oder rechten Rand einer Zeile bewegt.

- **[CLR]** löscht das Zeichen, auf dem der Cursor steht, **[DEL]** löscht das Zeichen links vom Cursor.

- Die **[RETURN]**-Taste drücken Sie, wenn Sie mit der Editierung einer Zeile fertig sind, oder bevor Sie eine neue Zeile in die Prozedur einschieben.

- **[ESC]** bedeutet Abbruch, und **[COPY]** teilt Dr. LOGO mit, daß die Bearbeitung einer Prozedur abgeschlossen ist.

Wenn Sie Befehle oder neue Prozeduren eingeben, editieren Sie einfach den Text auf dem Bildschirm. Einfügungen können dort vorgenommen werden, wo sich der Cursor befindet.

Um eine bereits bestehende Prozedur zu ändern, benutzen Sie den Befehl `ed`. Dr.LOGO bringt Ihnen dann die alte Version der Prozedur auf den Bildschirm, und Sie können mit dem Cursor wie oben beschrieben auf dem Bildschirm herumfahren und den Text ändern. Ändern Sie die Prozedur '`muster`', indem Sie schreiben:

```
ed "muster
```

Probieren Sie nun die Editier-Tasten aus. Wenn Sie fertig sind und **[ESC]** drücken, vergißt Dr.LOGO alle Änderungen und gibt Ihnen die Originalversion in unveränderter Form zurück. Tippen Sie noch einmal:

```
ed "muster
```

...ein, ersetzen Sie die Zahl 4 durch 8, steigen Sie mit **[COPY]** aus und lassen Sie die Prozedur nochmal laufen. Sie sehen jetzt, wie sich der Bildschirm-Ausdruck geändert hat. Vergessen Sie nicht, den Anfangswert von `:kante` zu setzen.

Bedienungshinweise

Der Arbeitsbereich von Dr.LOGO ist in Knoten (nodes) unterteilt. Wenn Sie:

```
nodes
```

...eintippen, können Sie sich die Zahl der freien Knoten anzeigen lassen. Gelegentlich, wenn fast alle Knoten belegt sind, reorganisiert Dr.LOGO den Arbeitsbereich. Die Schildkröte pausiert dann. Sie können Dr.LOGO auch direkt auffordern, den Arbeitsbereich neu zu ordnen, indem Sie:

```
recycle
```

...eintippen. Auf diese Weise können Sie die Bearbeitung fortsetzen, wenn Dr.LOGO gemeldet hat, daß keine Knoten mehr frei sind.

Wenn Sie mit der CP/M 2.2 Version von Dr.LOGO (auf Seite 4 der Systemdisketten) arbeiten, vergewissern Sie sich, daß auf der Diskette hinreichend Speicherplatz frei ist, im Ihre Prozeduren zu sichern. Sie können den freien Speicherplatz unter AMSDOS mit dem `CAT`-Befehl abfragen (siehe Teil 7 in Kapitel 1).

Sehen Sie sich die folgenden Abschnitte dieses Kapitels an, und probieren Sie ein paar Beispiele aus - auch wenn Sie nicht gleich alles auf Anhieb verstehen! Je mehr Sie sich mit Dr. LOGO beschäftigen, desto mehr Befehle werden Sie in Ihre Prozeduren einbauen können.

Wenn Sie Dr.LOGO verlassen wollen, schreiben Sie:

```
bye
```

Verzeichnis der Dr.LOGO-Grundprozeduren

Dieser Abschnitt führt die Dr.LOGO-Grundprozeduren (teilweise mit Beispiel) in alphabetischer Reihenfolge auf.

> HINWEIS: Befehle, die mit einem Sternchen * versehen sind, können nicht in der CP/M 2.2 Version von Dr.LOGO (auf Seite 4 der Systemdisketten) verwendet werden. Programme, die solche Befehle enthalten, können deshalb nicht auf dem CPC664 oder CPC464 + DDI1 laufen, die beide nur mit dem Betriebssystem CP/M 2.2 ausgerüstet sind.

Wörter- und Listen-Verarbeitung

(In den folgenden Beispielen werden die Prompt-Zeichen ? und > mit angezeigt.)

ascii

Gibt den ASCII-Wert des ersten Zeichens im eingegebenen Ausdruck an.

```
?ascii "G
71
?ascii "g
1Ø3
```

bf

(but first) Gibt alle Zeichen des eingegebenen Ausdrucks, bis auf das erste, aus.

```
?bf "klagen
lagen
?bf [1 2 3]
[2 3]
```

bl

(but last) Gibt alle Zeichen des eingegebenen Ausdrucks, bis auf das letzte, aus.

```
bl "klagen
klage
?bl [1 2 3 4]
[1 2 3]
```

char

(character) Gibt das Zeichen aus, das dem ASCII-Wert der eingegebenen Zahl entspricht.

```
?char 83
S
```

count

Gibt die Anzahl der Elemente des eingegebenen Ausdrucks an.

```
?count "sechs
5
?count [Ø 1 2 3]
4
```

emptyp

Gibt TRUE (richtig) aus, wenn der eingegebene Ausdruck ein leeres Wort oder eine leere Liste ist; andernfalls erscheint FALSE (falsch).

```
?emptyp "
TRUE
?emptyp []
TRUE
?emptyp [x]
FALSE
?make "x []
?emptyp :x
TRUE
```

first

Gibt das erste Element eines angegebenen Ausdrucks wieder.Die Klammern werden dabei weggelassen.

```
?first "zebra
z
?first [1 2 3]
1
```

fput

(firstput) Gibt einen neuen Ausdruck aus, in dem der erste eingegebene Ausdruck zum ersten Element des zweiten Ausdrucks wird.

```
?fput "k "lage
klage
?fput 1 [2 3]
[1 2 3]
```

item

Gibt das genannte Element des eingegebenen Ausdrucks an.

```
?item 4 "klagen
g
```

*
last

Gibt das letzte Element eines Ausdrucks aus. (Gegensatz zu first)

```
?last "zuletzt
t
```

*
lc

(lower case) Gibt alle Buchstaben des Ausdrucks in Kleinschreibweise aus. (Gegensatz zu uc)

```
?lc "KLEIN
klein
```

list

Gibt die eingegebenen Ausdrücke als Liste aus, wobei die äußeren Klammern erhalten bleiben. (Vergl. `se`)

```
?(list 1 2 3 4)
[1 2 3 4]
?list "grosse [klappe]
[grosse [klappe]]
?(list)
[]
```

*
listp

Gibt `TRUE` (richtig) aus, wenn es sich bei dem eingegebenen Ausdruck um eine Liste handelt, andernfalls erscheint `FALSE` (falsch).

```
?listp "mutter
FALSE
?listp [vater bruder schwester]
TRUE
```

*

lput

(lastput) Gibt einen neuen Ausdruck aus, bei dem der erste eingegebene Ausdruck zum letzten Element im zweiten Ausdruck wird.

```
?lput "e "viel
viele
?lput "e [viel]
[viel e]
```

*

memberp

Gibt TRUE aus, wenn der erste eingegebene Ausdruck ein Element des zweiten ist.

```
?memberp "z "prozedur
TRUE
?memberp "schokolade [[vanille][schokolade][erdbeer]]
FALSE
?memberp [schokolade] [vanille][schokolade][erdbeer]]
TRUE
```

*

numberp

Gibt TRUE aus, wenn der eingegebene Ausdruck eine Zahl darstellt.

```
?numberp 374.926
TRUE
?numberp "sechs
FALSE
?numberp first [2 4 5 8]
TRUE
```

*

piece

Gibt einen Ausdruck aus, der die angegebenen Elemente des eingegebenen Ausdrucks enthält.

```
?piece 3 6 "Pfaffenhofen
affe
?piece 2 4 [Nana John Michael Wendy Tinkerbell]
[John Michael Wendy]
```

se

(sentence) Gibt die eingegebenen Ausdrücke als Liste aus, wobei die äußeren Klammern weggelassen werden. (Vergl. `list`)

```
?make "befehls_liste rl
repeat 4 [fd 50 rt 90]
?run (se "cs :befehls_liste "ht)
```

Hinweis: Das Unterstreichungszeichen zwischen Befehls und liste finden Sie auf der 0-Taste.

*
shuffle

Gibt eine Liste aus, in der die Elemente der eingegebenen Liste in zufälliger Reihenfolge aufgeführt sind.

```
?shuffle [a b c d]
[c b d a]
```

*
uc

(upper case) Gibt die Buchstaben des eingegebenen Ausdrucks in Großschreibung wieder. (Gegensatz zu `lc`)

```
?uc "gross
GROSS
```

*
where

Bezieht sich auf die letzte `memberp`-Grundprozedur. Gibt an, an welcher Stelle der erste Ausdruck im zweiten Ausdruck erscheint.

```
?memberp "z "prozedur
TRUE
?show where
4
```

word

Setzt ein Wort aus mehreren eingegebenen Wörtern zusammen.

```
?word "sonnen "schein
sonnenschein
```

wordp

Gibt TRUE aus, wenn der eingegebene Ausdruck ein Wort oder eine Zahl ist.

```
?wordp "hallo
TRUE
?wordp []
FALSE
```

Arithmetische Operationen

*

arctan

Gibt den Arcus-Tangens der eingegebenen Zahl in Winkelgraden aus.

```
?arctan 0
0
?arctan 1
45
```

cos

Gibt den Cosinus des in Grad eingegebenen Winkels aus.

```
?cos 60
0.5
```

int

Gibt den ganzzahligen Anteil einer eingegebenen Zahl aus.

```
?int 4/3
1
```

quotient

Gibt den ganzzahligen Quotienten der beiden eingegebenen ganzen Zahlen aus.

```
?quotient 14 4
3
?14/4
3.5
```

random

Gibt eine positive Zufallszahl aus, die kleiner als die eingegebene Zahl ist.

```
?random 20
7
```

*

remainder

Gibt den ganzzahligen Rest aus, der nach der Division der ersten durch die zweite Zahl bleibt.

```
?remainder 7 3
1
?remainder 8 4
0
```

*

rerandom

Bewirkt, daß die nächste Zufallszahl dieselbe Folge von Zufallszahlen erzeugt wie beim letzten Mal.

```
?repeat 10 [(type random 10 char 9)]
1 3 7 5 3 2 0 4 2 6
?repeat 10 [(type random 10 char 9)]
4 9 9 1 0 6 1 3 5 1
?rerandom
?repeat 10 [(type random 10 char 9)]
5 2 9 0 3 1 6 2 3 7
?rerandom
?repeat 10 [(type random 10 char 9)]
5 2 9 0 3 1 6 2 3 7
```

*

round

Rundet die eingegebene Zahl zu einer ganzen Zahl.

```
?round 3.333333
3
?round 3.5
4
```

sin

Gibt den Sinus des in Grad eingegebenen Winkels aus.

```
?sin 30
0.5
```

+

Gibt die Summe der eingegebenen Zahlen aus.

```
?+ 2 2
4
?2+2
4
```

-

Gibt die Differenz der beiden eingegebenen Zahlen aus.

```
?- 10 5
5
?10-5
5
```

*

Gibt das Produkt der eingegebenen Zahlen aus.

```
?* 4 6
24
?4*6
24
```

/

Gibt den Quotienten der beiden eingegebenen Zahlen aus.

```
?/ 25 5
5
?25/5
5
```

Logische Operationen

and

Gibt TRUE aus, wenn das Ergebnis aller eingegebenen Ausdrücke wahr ist.

```
?and (3<4) (7>4)
TRUE
```

not

Gibt TRUE aus, wenn der eingegebene Ausdruck falsch ist.
Gibt FALSE aus, wenn der eingegebene Ausdruck wahr ist.

```
?not (3=4)
TRUE
?not (3=3)
FALSE
```

or

Gibt FALSE aus, wenn alle eingegebenen Ausdrücke falsch sind.

```
?or "TRUE "FALSE
TRUE
?or (3=4)(1=2)
FALSE
```

=

Gibt TRUE aus, wenn beide eingegebenen Ausdrücke gleich sind, andernfalls wird FALSE ausgegeben.

```
?= "LOGO "LOGO
TRUE
?1=2
FALSE
```

>

Gibt TRUE aus, wenn der erste Ausdruck größer als der zweite ist, andernfalls erscheint FALSE.

```
?> 19 20
FALSE
?20>19
TRUE
```

<

Gibt TRUE aus, wenn der erste Ausdruck kleiner als der zweite ist, andernfalls erscheint FALSE.

```
?< 27 13
FALSE
?13<27
TRUE
```

Variablen

local

Macht die angegebene(n) Variable(n) nur für die momentane Prozedur und Prozeduren, die von ihr aufgerufen werden, zugänglich.

```
>(local "x "y "z)
```

make

Weist der angegebenen Variablen den nachfolgenden Wert zu.

```
?make "seite 5Ø
? :seite
5Ø
```

*

namep

Gibt TRUE aus, wenn das eingegebene Wort eine definierte Variable darstellt.

```
?make "geschmack "schokolade
?:geschmack
schokolade
?namep "geschmack
TRUE
?namep "schokolade
FALSE
```

*

thing

Gibt den Wert der angegebenen Variablen aus.

```
?make "computer "amstrad
?thing "computer
amstrad
```

Prozeduren

*

define

Macht die eingegebene Liste von Definitionen zur Definition des angegebenen Prozedurnamens.

```
?define "sag.hallo [[] [pr "hallo]]
?po "sag.hallo
to sag.hallo
pr "hallo
?text "sag.hallo
[[] [pr "hallo]]
end
```

end

Zeigt das Ende einer Prozedur-Definition an; muß alleine am Beginn der letzten Zeile stehen.

```
?to quadrat
>repeat 4 [fd 50 rt 90]
>end
quadrat defined
?quadrat
```

po

(print out) Zeigt die Definition(en) der eingegebenen Prozeduren oder Variablen an.

```
?po "quadrat
to quadrat
repeat 4 [fd 50 rt 90]
end
?make "x 3
?po "x
x is 3
```

pots

(print out titles) Zeigt die Namen und Bezeichnungen aller Prozeduren im Arbeitsspeicher auf dem Bildschirm an.

```
?pots
```

*

text

Gibt die Definitionsliste der angegebenen Prozedur aus.

```
?to stern ;fuenfzackiger stern
>repeat 5 [fd 30 rt 144 fd 30 lt 72]
>end
stern defined
?text "stern
[[] [repeat 5 [fd 30 rt 144 fd 30 lt 72]]]
```

to

Kennzeichnet den Beginn einer Prozedur-Definition.

```
?to quadrat
>repeat 4 [fd 50 rt 90]
>end
quadrat defined
```

Editier-Befehle

ed

(edit) Lädt die angegebenen Prozedur(en) und/oder Variable(n) in den Bildschirm-Editierbereich.

```
?ed "quadrat
```

*

edall

Lädt alle Variablen und Prozeduren, die sich im Arbeitsbereich befinden, zum Editieren in den Bildschirm-Editierbereich.

```
?edall
```

*
edf

Lädt die angegebene Disketten-Datei direkt in den Bildschirm-Editierbereich, oder erstellt eine neue Datei mit einem leeren Bildschirm-Editierbereich.

```
?edf "stern
```

Drucker-Funktionen

*
copyon

Beginnt mit der Textausgabe auf dem Drucker.

```
?copyon
```

*
copyoff

Beendet die Textausgabe auf dem Drucker.

```
?copyoff
```

Text-Bildschirm

ct

(clear text) Löscht den gesamten Text in dem Fenster, in dem sich der Cursor momentan befindet, und setzt den Cursor dann in die linke obere Ecke des Fensters.

```
?ct
```

*
cursor

Gibt an, auf welchem Koordinatenpunkt (Nummer der Spalte und Zeile) des Fensters sich der Cursor gerade befindet.

```
?ct
?cursor
[Ø 1]
?(type [Der Cursor befindet sich auf dem Koordinatenpun
kt\] show cursor
Der Cursor befindet sich auf dem Koordinatenpunkt [32 2
3]
```

pr

(print) Stellt die Eingabe(n) auf dem Text-Bildschirm dar, enfernt die äußeren Klammern der Liste und fängt nach der letzten Eingabe eine neue Zeile an. (Vergl. show und type)

```
?pr [a b c]
a b c
```

*

setcursor

Setzt den Cursor an den Koordinatenpunkt, der in der Koordinaten-Liste auf dem Text-Bildschirm eingegeben wurde.

```
?ct
?to bild
>make "x random 20
>make "y random 12
>setcursor list :x :y pr "*
>end
?bild
```

setsplit

Legt die Anzahl der Zeilen im geteilten Bildschirmbereich fest.

```
?setsplit 10
```

show

Stellt die Eingabe(n) auf dem Text-Bildschirm dar, wobei die äußeren Klammern der Liste erhalten bleiben, und fängt nach der letzten Eingabe eine neue Zeile an. (Vergl. `pr` und `type`).

```
?show [a b c]
[a b c]
```

ts

(text screen) Macht den gesamten Bildschirm zum Text-Bildschirm.

```
?ts
```

type

Stellt die Eingaben auf dem Text-Bildschirm dar, entfernt die äußeren Klammern der Liste und fängt nach der letzten Eingabe keine neue Zeile an. (Vergl. `pr` und `show`)

```
?type [a b c]
a b c
```

Graphik-Bildschirm

Beachten Sie, daß sich der Bildschirm im Modus 1 befindet, vier verschiedene Farben darstellen kann und dasselbe Koordinaten-System wie AMSTRAD-BASIC verwendet. Mit anderen Worten, alle Bildschirmpositionen werden zum nächsten geradzahligen Bildschirmpunkt hin gerundet. Rot, Grün und Blau können die Werte 0, 1 oder 2 annehmen.

clean

Löscht den Graphik-Bildschirm mit Ausnahme der Schildkröte.

```
?fd 50
?clean
```

cs

(clear screen) Löscht den Graphik-Bildschirm und setzt die Schildkröte auf die Position (0,0), mit Ausrichtung nach 0 (Nord). Der Zeichenstift ist aktiviert.

```
?rt 90 fd 50
?cs
```

dot

Zeichnet einen Punkt, der durch die eingegebene Koordinaten-Liste festgelegt ist, in der momentanen Zeichenstift-Farbe.

```
?dot [50 10]
```

*

dotc

Gibt die Nummer für die Farbe an, in der der Punkt der angegebenen Koordinaten-Liste dargestellt ist.

```
?cs
?setpc 1
?dot [-50 50]
?setpc 2
?dot [50 50]
?setpc 3
?dot [50 -50]
?dotc [50 50]
2
?dotc [-50 -50]
0
?dotc [1000 3000]
-1
```

fence

Errichtet einen 'Zaun',der die Bewegungen der Schildkröte auf den sichtbaren Graphik-Bildschirm begrenzt. `window` löscht die Grenze wieder.

```
?fence
?fd 300
Turtle out of bounds
```

fs

(full screen) Macht den gesamten Bildschirm zum Graphik-Bildschirm.

```
?fs
```

pal

(palette) Gibt die einem Zeichenstift zugeordneten Farbwerte für Rot, Grün und Blau aus.

```
?pal 2
[Ø 2 2]
```

*

setbg

(set background) Färbt den Hintergrund des Graphik-Bildschirms mit der Farbe ein, die der eingegebenen Nummer zugeordnet ist.

```
?sf
[Ø SS 5 FENCE 1]
```

(Hiermit wird angegeben, daß der Hintergrund auf 0 gesetzt ist.)

```
?pal Ø
[Ø Ø 1]
?setbg 2
?sf
[2 SS 5 FENCE 1]
```

setpal

(set palette) Setzt die Farb-Palette für den Zeichenstift fest, d.h. ordnet ihm Werte für die Farben Rot, Grün und Blau zu.

```
?setpal 3 [1 1 2]
?pal 3
[1 1 2]
```

*

setscrunch

Setzt das Seitenverhältnis des Graphik-Bildschirms auf die eingegebene Zahl fest.

```
?sf
[0 SS 5 FENCE 1]
?to kreis
>repeat 360 [fd 1 rt 1]
>end
kreis defined
?setscrunch 2
?sf
[0 SS 5 FENCE 2]
?kreis
?setscrunch 2.5
?kreis
```

sf

(screen facts) Liefert Informationen über den Graphik-Bildschirm. Sie erscheinen in der Reihenfolge: [‹Hintergrundfarbe› ‹Bildschirm-Status› ‹Fenster-Status› ‹Seitenverhältnis›] Für ‹Hintergrundfarbe› erscheint die Nummer des für den Hintergrund verwendeten Zeichenstifts (bei CP/M 2.2 immer 0). Für ‹Bildschirm-Status› steht entweder `SS` (geteilter Bildschirm), `FS` (Graphik-Bildschirm) oder `TS` (Text-Bildschirm). ‹Fenstergröße› gibt die Anzahl der Text-Zeilen im Textfenster des Bildschirms an. Für ‹Fenster-Status› wird der `WINDOW-`, `WRAP-` oder `FENCE`-Modus angegeben. Das ‹Seitenverhältnis› ist immer 1, solange es nicht mit `setscrunch` verändert wird. (Unter CP/M 2.2 ist diese Möglichkeit nicht gegeben.)

```
?sf
[0 SS 5 FENCE 2.5]
```

ss

(split screen) Richtet auf dem Graphik-Bildschirm ein Textfenster ein.

```
?ss
```

window

Erlaubt der Schildkröte (nach einem `wrap`- oder `fence`-Ausdruck) außerhalb des sichbaren Graphik-Bildschirms zu zeichnen.

```
?fence fd 300
Turtle out of bounds
?window
fd 300
```

wrap

Läßt die Schildkröte auf der gegenüberliegenden Seite des Bildschirms auftauchen, wenn sie die Grenzen des Bildschirms überschreitet.

```
?cs wrap
?rt 5 fd 1000
?cs window
?rt 5 fd 1000
```

Schildkröten-Graphik

bk

(back) Bewegt die Schildkröte um die angegebene Anzahl Schritte rückwärts.

```
?cs fd 150
?bk 50
```

fd

(forward) Bewegt die Schildkröte um die angegebene Anzahl Schritte vorwärts.

```
?fd 80
```

*

home

Setzt die Schildkröte auf die Position (0/0) (Bildschirm-Mitte) mit Ausrichtung nach 0 (Nord).

```
?fd 100
?rt 45
?fd 100
?home
```

ht

(hide turtle) Macht die Schildkröte unsichtbar. Beschleunigt und verdeutlicht den Zeichenvorgang.

```
?ht
?cs fd 50
?st
```

lt

(left) Dreht die Schildkröte um die angegebene Anzahl von Winkelgraden nach links.

```
?lt 90
```

pd

(pen down) Aktiviert den Zeichenstift (pen) der Schildkröte; die Schildkröte beginnt wieder zu zeichnen.

```
?fd 20 pu fd 20
?pd
?fd 20
```

pe

(pen erase) Verwandelt die Farbe des Zeichenstifts für die Schildkröte in die Farbe des Hintergrunds; die Schildkröte löscht gezeichnete Linien.

```
?fd 50
?pe
?bk 25
?fd 50
?pd fd 25
```

pu

(pen up) Deaktiviert den Zeichenstift (pen) der Schildkröte; die Schildkröte hört auf zu zeichnen.

```
?fd 30
?pu
?fd 30
?pd fd 30
```

px

(pen exchange) Veranlaßt die Schildkröte, auf ihrem Weg die vorher gezeichneten Punkte in die Komplementärfarbe zu verwandeln.

```
?fd 20 pu fd 20
?pd setpc 3 fd 20
?px
?bk 80
?fd 80
?pd bk 100
```

rt

(right) Dreht die Schilkröte um die angegebene Anzahl von Winkelgraden nach rechts.

```
?rt 90
```

seth

(set heading) Dreht die Schildkröte in die absolute Richtung, die durch den angegebenen Winkelgrad festgelegt ist. Positive Werte drehen die Schildkröte in Uhrzeigerrichtung, negative Werte drehen sie entgegen der Uhrzeigerrichtung.

```
?seth 90
```

setpc

(set pen colour) Stattet die Schildkröte mit dem Zeichenstift aus, der der angegebenen Nummer zugeordnet ist.

```
?setpc 1
```

setpos

(set position) Setzt die Schildkröte auf die Position, die durch die eingegebenen Koordinaten festgelegt wird.

```
?setpos [30 20]
```

*

setx

Setzt die Schildkröte auf den angegebenen Punkt der x-Koordinate.

```
?setx 80
?fd 100
?setx -50
?fd 50
```

*

sety

Setzt die Schildkröte auf den angegebenen Punkt der y-Koordinate.

```
?sety 90
?fd 20
?sety -50
?fd 50
```

st

(show turtle) Macht die Schildkröte wieder sichtbar.

```
?ht
?fd 50
?st
```

tf

(turtle facts) Liefert Informationen über die Schildkröte. Sie erscheinen in der Reihenfolge: [‹xkor› ‹ykor› ‹Richtung› ‹Pen-Status› ‹Pen-Farbe› ‹sichtbar›] Mit ‹xkor› und ‹ykor› wird die x- bzw. y-Koordinate der Schildkröte angegeben. ‹Richtung› gibt an, in welche Kompaß-Richtung die Schildkröte weist. Für ‹Zeichenstift-Status› kann PD (pen down), PE (pen erase), PX (pen exchange) oder PU (pen up) erscheinen. ‹Pen-Farbe› bezeichnet die Farbnummer des Zeichenstifts. Für ‹sichtbar› erscheint TRUE, wenn die Schildkröte sichtbar ist, ansonsten FALSE.

```
?setpos [15 30]
?rt 60
?setpc 3
?pe
?ht
?tf
[15 30 60 PE 3 FALSE]
```

*

towards

Gibt die Größe des Winkels an, um den die Schildkröte sich drehen müßte, um auf den angegebenen Koordinatenpunkt zu weisen.

```
?seth towards list :x :y
```

Speicherplatz-Verwaltung

er

(erase) Löscht die angegebene(n) Prozedur(en) aus dem Arbeitsspeicher.

```
?er "quadrat
```

*

erall

Löscht alle Prozeduren und Variablen aus dem Arbeitsspeicher.

```
?erall
```

ern

(erase name) Löscht die angegebene(n) Variable(n) aus dem Arbeitsspeicher.

```
?make "seite [100]
?make "winkel [45]
?:seite :winkel
[100]
[45]
?ern [seite winkel]
?:seite
seite has no value
```

nodes

Gibt die Anzahl der freien Knoten im Speicher an.

```
?nodes
```

*

noformat

Löscht die Formatierung der Prozedur, einschließlich Kommentaren, aus dem Arbeitsspeicher, um mehr Knoten frei zu machen.

```
?noformat
```

*

poall

Zeigt die Definitionen aller Prozeduren und Variablen im Arbeitsspeicher an.

```
?poall
```

*

pons

Zeigt die Namen und Werte aller Global-Variablen im Arbeitsspeicher an.

```
?pons
medium is 40
small is 20
large is 80
```

*

pops

Zeigt die Namen und Definitionen aller Prozeduren im Arbeitsspeicher an.

```
?pops
```

recycle

Reorganisiert den Arbeitsspeicher und stellt soviele Knoten wie möglich zur Verfügung.

```
?recycle
?nodes
```

Eigenschaftslisten

glist

(get list) Gibt eine Liste aller im Arbeitsbereich gespeicherten Ausdrücke aus, welche die eingegebene Eigenschaft in ihrer Eigenschaftliste enthalten.

```
?glist ".DEF
```

gprop

(get property) Gibt den Eigenschaftswert des zum eingegebenen Objektnamen gehörenden Eigenschaftsnamens an.

```
?make "hoehe "72
?gprop "hoehe ".APV
72
```

plist

(property list) Gibt die Eigenschaftsliste des eingegebenen Objekts aus.

```
?plist "hoehe
(.APV 72)
```

pprop

(put property) Übergibt der Eigenschaftsliste des genannten Objekts das eingegebene Eigenschaftspaar.

```
?pprop "auto ".APV "farbe
? :auto
farbe
```

*

pps

Zeigt die vom Benutzer definierten Eigenschaftspaare aller Objekte im Arbeitsspeicher an.

```
?pprop "Sally "Rufnummer 213
?pps
Sallys Rufnummer ist 213
?plist "Sally
[Rufnummer 213]
```

remprop

(remove property) Entfernt die genannte Eigenschaft aus der Eigenschaftsliste des eingegebenen Objekts.

```
?remprop "auto ".APV
```

Disketten-Dateien

*

changef

(change file) Ändert den Namen einer Datei im Disketten-Inhaltsverzeichnis.

```
?dir
[QUADRAT KREIS STERNE]
?changef "viereck "quadrat
dir
[VIERECK KREIS STERNE]
```

*

defaultd

(default drive) Gibt an, welches Diskettenlaufwerk das Standardlaufwerk ist.

```
?defaultd
A:
```

dir

(directory) Gibt eine Liste mit den Dr.LOGO-Dateinamen der Standard-Diskette oder der angegebenen Diskette aus. Universalzeichen werden akzeptiert.

```
?dir "a:????????
```

(Über den Gebrauch von Universalzeichen (? und *) finden Sie Näheres im 1. Teil des Kapitels 'AMSDOS und CP/M'. Beachten Sie, daß das Universalzeichen * bei Dr.LOGO nicht verwendet werden kann.)

*

dirpic

Gibt eine Liste von Dateinamen für Bilder aus, die sich auf der Standard- oder auf der angegebenen Diskette befinden. Dateinamen mit Universalzeichen werden akzeptiert.

```
?dirpic "b:
[BILD_ELF QUADRATE STERNE NOTIZEN]
```

load

Liest die angegebene Datei von der Diskette in den Arbeitsspeicher.

```
?load "sterne
?load "b:formen
```

*

loadpic

Ruft das Muster, das in der eingegebenen Bild-Datei gespeichert wurde, auf den Graphik-Bildschirm.

```
?loadpic "bild_elf
?loadpic "b:bild_elf
```

save

Schreibt den Inhalt des Arbeitsspeichers unter dem angegebenen Dateinamen auf die Diskette.

```
?save "formen
```

HINWEIS: Bevor Sie etwas abspeichern, schieben Sie eine formatierte Diskette mit genügend freiem Speicherplatz ins Laufwerk. Speichern Sie keine Dateien auf Ihre Systemdisketten. Um jedes Risiko zu vermeiden, daß auf die Systemdisketten geschrieben wird, schließen Sie NIEMALS die Schreibschutzlöcher Ihrer Systemdisketten!

Wenn Sie mit der CP/M 2.2-Version von Dr.LOGO (auf Seite 4 der Systemdisketten) arbeiten, können Sie die Disketten während des Speichervorgangs nicht wechseln. Es ist deshalb sehr wichtig, daß auf Ihrer CP/M 2.2-Arbeitsdiskette mit Dr.LOGO genügend Platz für Programme ist.

*

savepic

Schreibt den Inhalt des Graphik-Bildschirms unter dem angegebenen Namen in die Bild-Datei.

```
?savepic "bild_elf
?savepic "b:bild_elf
```

HINWEIS: Bevor Sie etwas abspeichern, schieben Sie eine formatierte Diskette mit genügend freiem Speicherplatz ins Laufwerk. Speichern Sie keine Dateien auf Ihre Systemdisketten. Um jedes Risiko zu vermeiden, daß auf die Systemdisketten geschrieben wird, schließen Sie NIEMALS die Schreibschutzlöcher Ihrer Systemdisketten!

Wenn Sie mit der CP/M 2.2-Version von Dr.LOGO (auf Seite 4 der Systemdisketten) arbeiten, können Sie die Disketten während des Speichervorgangs nicht wechseln. Es ist deshalb sehr wichtig, daß auf Ihrer CP/M 2.2-Arbeitsdiskette mit Dr.LOGO genügend Platz für Programme ist.

*

setd

(set drive) Macht das angegebene Laufwerk zum Standardlaufwerk.

```
?defaultd
A:
?dir
[VIERECKE KREISE STERNE]
?setd "b:
defaultd
B:
?dir
[DREIECK HAUS]
```

Tastatur und Joystick

buttonp

(button pressed) Gibt TRUE aus, wenn der Knopf des angegebenen Joysticks gedrückt ist. Den beiden möglichen Joysticks sind die Nummern 0 und 1 zugeordnet.

```
?to feuer
>label "schleife
>if (button 0) [pr [feuer 0!]]
>if (button 1) [pr [feuer 1!]]
>go "schleife
>end
```

Die Position des Joysticks wird durch paddle ermittelt.

keyp

(key pressed) Gibt TRUE aus, wenn eine Taste der Tastatur gedrückt wurde und darauf wartet, eingegeben zu werden.

```
?to eintaste
>if keyp [op rc] [op "]
>end
```

paddle

Gibt den Zustand von Joystick 0 oder 1 an. Die Positionen des Joysticks werden wie folgt angezeigt:

Angezeigter Wert	Bedeutung
255	nichts gedrückt
0	aufwärts
1	aufwärts und rechts
2	rechts
3	abwärts und rechts
4	abwärts
5	abwärts und links
6	links
7	aufwärts und links

```
?paddle 0
255
```

Der Zustand der Feuer-Knöpfe wird mit buttonp ermittelt.

rc

(read character) Gibt das erste Zeichen an, das auf der Tastatur gedrückt wurde.

```
?make "taste rc
```

Drücken Sie nun die X-Taste.

```
?:taste
X
```

rl

(read list) Gibt eine Liste aus, welche die über die Tastatur eingegebene Zeile enthält. Der Eingabe muß ein Zeilenrücksprung folgen.

```
?make "bef_liste rl
repeat 4 [fd 50 rt 90]
?:bef_liste
[repeat 4 [fd 50 rt 90]]
```

rq

(read quote) Gibt ein Wort aus, das die über die Tastatur eingegebene Zeile enthält. Der Eingabe muß ein Zeilenrücksprung folgen.

```
?make "befehl rq
repeat 3 [fd 60 rt 120]
?:befehl
repeat 3 [fd 60 rt 120]
```

Sound-Befehle

Die Sound-Befehle gibt es nur in der AMSTRAD-Version von Dr.LOGO. Sie ähneln den Sound-Befehlen in AMSTRAD-BASIC.

sound

Übergibt einen Ton an die Ton-Warteschlange. Das Format ist: [‹Kanalstatus› ‹Tonperiode› ‹Dauer› ‹Lautstärke› ‹Lautstärken-Hüllkurve› ‹Ton-Hüllkurve› ‹Geräusche›]. Die vier Parameter nach ‹Dauer› sind Wahl-Parameter.

```
?sound [1 20 50]
```

env

Definiert eine Lautstärken-Hüllkurve. Das Format ist: [‹Hüllkurven-Nummer› ‹Hüllkurven-Abschnitt(e)›]

```
?env [1 100 2 20]
?sound [1 200 300 5 1]
```

ent

Definiert eine Ton-Hüllkurve. Das Format ist: [‹Hüllkurven-Nummer› ‹Hüllkurven-Abschnitt(e)›]

```
?ent [1 100 2 20]
?sound [ 1 200 300 5 1 1]
```

release

Gibt einen Ton-Kanal frei, der durch den `sound`-Befehl gesperrt wurde. Die freizugebenden Kanäle werden folgendermaßen gekennzeichnet:

Eingegebener Wert	Freigegebener Kanal
0	keiner
1	A
2	B
3	A und B
4	C
5	A und C
6	B und C
7	A, B und C

```
?release 1
```

Ablauf-Steuerung

bye

Beendet die Bearbeitung unter Dr.LOGO.

```
?bye
```

co

Beendet eine durch **[CONTROL]Z**, `pause` oder `ERRACT` verursachte Unterbrechungspause.

```
?co
```

go

Führt die Zeile innerhalb der momentanen Prozedur aus, deren Marke (label) dem eingegebenen Wort entspricht.

```
>go "schleife
```

if

Führt je nach dem Wert des eingegebenen Ausdrucks eine von zwei möglichen Befehls-Listen aus. Die eingegebenen Anweisungen müssen Direkt-Befehlslisten sein und in Klammern stehen.

```
>if (a>b) [pr [a ist groesser]]
>[pr [b ist groesser]]
```

label

Kennzeichnet die Zeile, die nach dem go-Befehl ausgeführt werden soll.

```
>label "schleife
```

op

(output) Beendet die Ausführung der Prozedur und gibt die Eingabe als Wert zurück.

```
?op [ergebnis]
```

repeat

Führt die angegebene Befehls-Liste sooft aus, wie die Zahl angibt.

```
?repeat 4 [fd 50 rt 90]
```

run

Führt die eingegebene Befehls-Liste aus.

```
?make "befehls_liste [fd 40 rt 90]
?run :befehls_liste
```

stop

Bricht die Durchführung der Prozedur ab und kehrt zu TOPLEVEL (mit dem Prompt ?) oder der aufrufenden Prozedur zurück.

```
?stop
```

wait

Unterbricht die Durchführung der Prozedur für eine bestimmte Zeit. Unterbrechungszeit = Eingabewert * 1/60 Sekunden.

```
?wait 200
```

Ausnahmefälle

catch

Erkennt Fehler und besondere Umstände, die während der Ausführung der eingegebenen Befehls-Liste auftreten.

```
>catch "error [+ [] []]
>pr [Hier bin ich]
Hier bin ich
```

error

Gibt eine Liste aus, deren Elemente den zuletzt aufgetretenen Fehler beschreiben.

```
>catch "error [do.until.error]
>show error
```

*

notrace

Schaltet die Möglichkeit, die Prozedur während ihrer Durchführung zu verfolgen, ab. (vergl. `trace`)

```
?notrace
```

*

nowatch

Schaltet die Möglichkeit, alle oder die angegebenen Prozedur(en) während ihrer Ausführung zu überprüfen, ab. (vergl. `watch`)

```
?nowatch
```

pause

Unterbricht die Ausführung der Prozedur, um den Aufruf des Interpreters oder Editors zu ermöglichen.

```
>if :groesse>5 [pause]
```

throw

Führt die Zeile aus, die durch den eingegebenen Namen im vorangegangenen catch-Ausdruck gekennzeichnet ist.

```
?throw "TOPLEVEL
```

*

trace

Ermöglicht es, eine Prozedur während ihrer Durchführung zu verfolgen.

```
?trace
```

*

watch

Ermöglicht die Überprüfung aller oder der angegebenen Prozedur(en) während ihrer Durchführung.

```
?watch
```

System-Primitive

.contents

Zeigt den Inhalt des Dr.LOGO Symbol-Speichers.

.deposit

Legt den zweiten Eingabewert unter der absoluten Speicheradresse, die durch den ersten Eingabewert bestimmt wurde, ab.

.examine

Gibt den Inhalt der angegebenen absoluten Speicheradresse aus.

*
.in

Ruft den momentanen Wert vom Port mit der eingegebenen Nummer ab.

*
.out

Schickt den eingegebenen Wert zum Port mit der angegebenen Nummer.

System-Variablen

ERRACT

Ist `ERRACT TRUE`, bewirkt sie beim Auftreten eines Fehlers eine Pause und kehrt daraufhin nach `TOPLEVEL` zurück.

FALSE

System-Wert

REDEFP

Ist `REDEFP TRUE`, können Primitive neu definiert werden.

TOPLEVEL

`throw "TOPLEVEL` beendet alle momentan bearbeiteten Prozeduren.

TRUE

System-Wert

System-Eigenschaften

.APV

(Associated Property Value) Wert einer Global-Variablen

.DEF

Definition einer Prozedur

*

.ENL

Ende einer Prozedur-Zeile, die durch einen Zeilenrücksprung oder Leerstellen unterbrochen ist.

*

.EMT

Beginn einer Prozedur-Zeile, die durch einen Zeilenrücksprung oder Leerstellen unterbrochen ist.

.PRM

Kennzeichnet eine Primitive

*

.REM (or;)

(remarks) Bemerkungen, Kommentare

Kapitel 7
Übersicht

Dieses Kapitel ist das 'Nachschlagewerk'. Sie finden hier eine Vielzahl von Informationen, die Sie brauchen, während Sie sich mit diesem Computer vertraut machen.

Behandelte Themen:

- ★ Cursorpositionen und Steuer-Codes
- ★ Unterbrechungsfunktionen
- ★ ASCII-Zeichensatz und graphische Zeichen
- ★ Tastenbezeichnungen
- ★ Noten und Tonperioden
- ★ Fehlermeldungen
- ★ BASIC-Schlüsselwörter
- ★ Entwurfsgitter
- ★ Anschlüsse
- ★ Drucker
- ★ Joysticks
- ★ Disketten-Organisation
- ★ Eingebaute System-Erweiterungen (RSX)
- ★ Speicher
- ★ CP/M Plus Bildschirmanpassung
- ★ CP/M Plus Zeichensatz

Umfassende Informationen über BASIC und Firmware für den CPC6128 finden Sie in den AMSTRAD Handbüchern SW 967 bzw. SW 968.

Teil 1:
BASIC Cursor-Positionen und Steuer-Codes

In zahlreichen Anwender-Programmen liegt der Cursor außerhalb des Bildschirmbereiches (`window`). Um ihn vor Ausführung des Programms in eine erlaubte Position zu rücken, stehen folgende Operationen zur Verfügung:

1. Ein Zeichen schreiben

2. Den Cursor auf den Bildschirm bringen

3. Den Steuer-Codes folgen, die in der BASIC-Steuerzeichen-Liste mit einem Sternchen versehen sind.

Je nach seiner Position außerhalb des Fensters erscheint der Cursor dann an unterschiedlichen Stellen innerhalb des Fensters:

1. Befindet sich der Cursor rechts vom rechten Fenster-Rand, rückt er auf die nächste Zeile in die Spalte ganz links.

2. Befindet sich der Cursor links vom linken Fenster-Rand, rückt er auf die vorige Zeile in die Spalte ganz rechts.

3. Befindet sich der Cursor über dem oberen Rand des Fensters, wird das Fenster um eine Zeile nach unten gerollt und der Cursor in die oberste Zeile des Fensters gesetzt.

4. Befindet sich der Cursor unterhalb des unteren Fenster-Randes, wird das Fenster um eine Zeile nach oben gerollt und der Cursor in die unterste Zeile des Fensters gesetzt.

Die Tests und Operationen werden in der genannten Reihenfolge durchgeführt. Die unerlaubten Cursor-Positionen können Null oder negativ sein, d.h. links oder oberhalb des Fensters.

Werden Zeichenwerte zwischen 0 und 31 auf den Text-Bildschirm ausgegeben, so werden sie nicht angezeigt, sondern als Steuer-Codes interpretiert. Sie sollten daher nicht leichtfertig eingegeben werden. Einige Codes ändern die Bedeutung eines oder mehererer der folgenden Zeichen, die die Parameter des Codes darstellen.

Ein Steuer-Code, der über die Tastatur eingegeben und zum Graphik-Bildschirm geschickt wird, erscheint lediglich als übliches, seiner Funktion entsprechendes Symbol (z.B. &07 'BEL'-**[CONTROL] G**). Er wird seine Funktion ausführen, wenn er mit einem Befehl angesprochen wird:

`PRINT CHR$(&Ø7)` oder `PRINT " `  `"` (wobei für in der `PRINT`-Anweisung **[CONTROL]G** gedrückt wird).

Die mit einem Sternchen * versehenen Codes bringen den Cursor in eine erlaubte Position innerhalb des Fensters, bevor sie ausgeführt werden. Dabei kann der Cursor in einer unerlaubten Position belassen werden. Für die Codes und ihre Bedeutungen werden erst ihre HEX-Werte (&XX) und dann die entsprechenden Dezimalwerte angegeben.

BASIC Steuer-Codes

Wert	Name	Parameter	Bedeutung
&00 0	NUL		Kein Einfluß. Wird ignoriert.
&01 1	SOH	0 bis 255	Gibt das zum Parameter-Wert gehörende Symbol aus. Symbole im Bereich 0 bis 31 werden angezeigt.
&02 2	STX		Schaltet den Text-Cursor ab. Entspricht dem CURSOR-Befehl mit dem Parameter-Wert 0 für ‹Benutzerschalter›.
&03 3	ETX		Schaltet den Text-Cursor an. Entspricht dem CURSOR-Befehl mit dem Parameter-Wert 1 für ‹Benutzerschalter›. Hinweis: Um den Cursor innerhalb eines BASIC-Programms auf den Bildschirm zu holen (im Gegensatz zum Direkt-Modus, wo der Cursor automatisch erscheint), muß der CURSOR-Befehl mit dem ‹Benutzerschalter›-Parameter 1 verwendet werden.
&04 4	EOT	0 bis 2	Setzt den Bildschirm-Modus. Parameter ist MOD 4. Entspricht dem MODE-Befehl.
&05 5	ENQ	0 bis 255	Sendet das Parameter-Zeichen zum Graphik-Cursor.

Wert		Name	Parameter	Bedeutung
&06 6		ACK		Schaltet auf Text-Bildschirm um. (Vergl. &15 NAK.)
&07 7		BEL		Pieps-Ton. Löscht die Ton-Warteschlangen.
&08 8	*	BS		Setzt Cursor um eine Stelle zurück.
&09 9	*	TAB		Bewegt Cursor eine Stelle vorwärts.
&0A 10	*	LF		Bewegt Cursor eine Zeile abwärts.
&0B 11	*	VT		Bewegt Cursor eine Zeile aufwärts.
&0C 12		FF		Löscht das Text-Fenster und setzt Cursor in die obere linke Ecke. Entspricht dem Befehl `CLS`.
&0D 13	*	CR		Setzt Cursor an den linken Rand des Text-Fensters auf derselben Zeile.
&0E 14		SO	0 bis 15	Bestimmt den Farbstift für den Schrift-Untergrund. Parameter ist MOD 16. Entspricht dem Befehl `PAPER`.
&0F 15		SI	0 bis 15	Bestimmt den Farbstift für die Schrift. Parameter ist MOD 16. Entspricht dem Befehl `PEN`.
&10 16	*	DLE		Löscht den Buchstaben unter dem Cursor. Füllt die Buchstabenzelle mit dem `PAPER`-Farbstift.
&11 17	*	DC1		Löscht alles in der Zeile vom linken Rand des Fensters bis zur Cursor-Position einschließlich. Füllt die betreffenden Buchstabenzellen mit dem `PAPER`-Farbstift.

Wert		Name	Parameter	Bedeutung
&12 18	*	DC2		Löscht alles in der Zeile von der Cursor-Position einschließlich bis zum rechten Rand des Fensters. Füllt die betreffenden Buchstabenzellen mit dem PAPER-Farbstift.
&13 19	*	DC3		Löscht alles vom Beginn des Fensters bis zur Cursor-Position einschließlich. Füllt die betreffenden Buchstabenzellen mit dem PAPER-Farbstift.
&14 20	*	DC4		Löscht alles von der Cursor-Position einschließlich bis zum Ende des Fensters. Füllt die betreffenden Buchstabenzellen mit dem PAPER-Farbstift.
&15 21		NAK		Schaltet den Text-Bildschirm ab. Der Bildschirm reagiert auf keine Eingaben, bis er mit ACK (&06 6) wieder in Betrieb genommen wird.
&16 22		SYN	0 bis 1	Parameter MOD 2. Transparent-Modus, mit 0 eingeschaltet, mit 1 ausgeschaltet.
&17 23		ETB	0 bis 3	Parameter MOD 4. 0 setzt normalen Graphik-Ink-Modus 1 setzt Graphik-Ink-Modus nach XOR 2 setzt Graphik-Ink-Modus nach AND 3 setzt Graphik-Ink-Modus nach OR.
&18 24		CAN		Tauscht Farbstifte von Pen und Paper aus.

Wert	Name	Parameter	Bedeutung
&19 25	EM	0 bis 255 0 bis 255 0 bis 255 0 bis 255 0 bis 255 0 bis 255 0 bis 255 0 bis 255 0 bis 255	Setzt Matrix für Benutzer-definiertes Zeichen. Entspricht dem Befehl `SYMBOL`. Hat 9 Parameter. Der erste Parameter gibt an, welches Zeichen umdefiniert werden soll, die anderen acht definieren die Matrix. Das Bit mit der höchsten Signifikanz aus dem ersten Byte entspricht dem Bildpunkt links oben in der Buchstabenzelle, das Bit mit der niedrigsten Signifikanz aus dem letzten Byte entspricht dem Bildpunkt unten rechts in der Buchstabenzelle.
&1A 26	SUB	1 bis 80 1 bis 80 1 bis 25 1 bis 25	Definiert ein Fenster (window). Entspricht dem `WINDOW`-Befehl. Die beiden ersten Parameter bestimmen die rechte (größerer Wert) und linke (kleinerer Wert) Begrenzung des Fensters, die nächsten beiden Parameter bestimmen die obere (kleinerer Wert) und die untere (größerer Wert) Begrenzung.
&1B 27	ESC		Kein Einfluß. Wird ignoriert.
&1C 28	FS	0 bis 15 0 bis 31 0 bis 31	Legt Farbenpaar für Farbstift fest. Entspricht dem Befehl `INK`. Der erste Parameter (MOD 16) nennt den Farbstift, die nächsten beiden Parameter (MOD 32) bestimmen die Farben. (Die Parameter-Werte 27 bis 31 stehen für undefinierte Farben.)

Wert	Name	Parameter	Bedeutung
&1D 29	GS	0 bis 31 0 bis 31	Bestimmt Farbenpaar für den Rand. Entspricht dem Befehl BORDER. Die beiden Parameter (MOD 32) bestimmen die beiden Farben. (Die Parameter-Werte 27 bis 31 stehen für undefinierte Farben.
&1E 30	RS		Setzt Cursor in die linke obere Ecke des Fensters.
&1F 31	US	1 bis 80 1 bis 25	Setzt Cursor auf die angegebene Position innerhalb des Fensters. Entspricht dem Befehl LOCATE. Der erste Parameter definiert die Spalte, der zweite die Zeile.

Die Steuerung des CPC6128 wird von einem komplizierten Echtzeit-Betriebssystem übernommen. Das Betriebssystem steuert den 'Verkehrsfluß' durch den Computer, von der Eingabe bis zu Ausgabe.

Es stellt in erster Linie das Bindeglied zwischen der Hardware und dem BASIC-Interpreter dar. Bei der Funktion der wechselnden Ink-Farben z.B. übergibt BASIC die Parameter, und das Betriebssystem übernimmt die Weiterbehandlung: Ein Teil analysiert, was zu tun ist, der andere überwacht den zeitlichen Ablauf.

Das Betriebssystem eines Computers wird allgemein als 'Firmware' bezeichnet. Es besteht aus Maschinencode-Routinen, die BASIC mit seinen Kommandos aufruft.

Falls Sie versuchen möchten, mit POKE-Befehlen die internen Speicherinhalte zu verändern oder mit CALL die Systemroutinen aufzurufen, sichern Sie vorher ihr Programm, da es zerstört werden könnte. Eine ausführliche Beschreibung der Betriebssystem-Firmware des CPC6128 steht in einem zusätzlichen Handbuch, das weit über den Umfang dieser Einführung hinausgeht.

Teil 2: Unterbrechungsfunktionen

Der CPC6128 nutzt die Unterbrechungsmöglichkeiten des Z80 in hohem Maße aus, so daß das Betriebssystem Multi-Tasking-Funktionen (mehrere Anweisungen werden gleichzeitig bearbeitet) ausüben kann. Ein Beispiel dafür sind die AFTER- und EVERY-Kommandos, die in Kapitel 8 beschrieben sind. Die Unterbrechungen haben folgende Prioritätenfolge:

Break (**[ESC][ESC]**)
Zeitgeber 3
Zeitgeber 2 (und die drei Ton-Warteschlangen)
Zeitgeber 1
Zeitgeber 0

Unterbrechungen sollten nicht eingebaut werden, ohne die möglichen Zustände der verschiedenen Variablen zum Zeitpunkt der Unterbrechung zu durchdenken. Das Unterbrechungs-Unterprogramm sollte ungewollte Veränderungen der Variablen im Hauptprogramm vermeiden.

Die Ton-Warteschlangen haben unabhängige Unterbrechungen mit gleicher Priorität. Sobald eine solche Unterbrechung eingetreten ist, wird sie von keiner anderen nochmals unterbrochen. Für Sound-Unterbrechungs-Unterprogramme können deshalb anders als bei Zeitunterbrechungen die gleichen Variablen benutzt werden.

Wenn eine Ton-Warteschlangen-Unterbrechung mit dem Befehl ON SQ GOSUB in Kraft gesetzt ist, so tritt sofort eine Unterbrechung ein, wenn die Ton-Warteschlange für diesen Kanal nicht voll ist. Ansonsten tritt eine Unterbrechung nur ein, sobald der nächste Ton gestartet wurde und wieder Platz in der Warteschlange vorhanden ist. Durch den Unterbrechungsvorgang werden weitere Unterbrechungen außer Kraft gesetzt, weshalb das Unterbrechungs-Unterprogramm eine Unterbrechungsmöglichkeit für weitere Fälle bereitstellen muß.

Wird ein Ton ausgelöst oder der Zustand der Warteschlange getestet, ist die Unterbrechungsfunktion ebenfalls außer Kraft gesetzt.

Teil 3: ASCII-Zeichensatz und graphische Zeichen

ASCII

In dieser Tabelle ist der Standard-ASCII-Zeichensatz mit dezimaler, oktaler und hexadezimaler Schreibweise zusammen mit den zugehörigen ASCII-Codes aufgeführt. Die folgenden Seiten zeigen alle CPC6128-Zeichen im Detail.

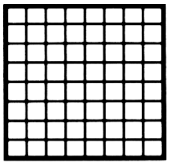
DEZ	OKTAL	HEX	ASCII Zeichen
0	000	00	NUL ([CTRL]@)
1	001	01	SOH ([CTRL]A)
2	002	02	STX ([CTRL]B)
3	003	03	ETX ([CTRL]C)
4	004	04	EOT ([CTRL]D)
5	005	05	ENQ ([CTRL]E)
6	006	06	ACK ([CTRL]F)
7	007	07	BEL ([CTRL]G)
8	010	08	BS ([CTRL]H)
9	011	09	HT ([CTRL]I)
10	012	OA	LF ([CTRL]J)
11	013	OB	VT ([CTRL]K)
12	014	OC	FF ([CTRL]L)
13	015	OD	CR ([CTRL]M)
14	016	OE	SO ([CTRL]N)
15	017	OF	SI ([CTRL]O)
16	020	10	DLE ([CTRL]P)
17	021	11	DC1 ([CTRL]Q)
18	022	12	DC2 ([CTRL]R)
19	023	13	DC3 ([CTRL]S)
20	024	14	DC4 ([CTRL]T)
21	025	15	NAK ([CTRL]U)
22	026	16	SYN ([CTRL]V)
23	027	17	ETB ([CTRL]W)
24	030	18	CAN ([CTRL]X)
25	031	19	EM ([CTRL]Y)
26	032	1A	SUB ([CTRL]Z)
27	033	1B	ESC
28	034	1C	FS
29	035	1D	GS
30	036	1E	RS
31	037	1F	US
32	040	20	SP
33	041	21	!
34	042	22	"
35	043	23	#
36	044	24	$
37	045	25	%
38	046	26	&
39	047	27	'
40	050	28	(
41	051	29	)
42	052	2A	*
43	053	2B	+
44	054	2C	,
45	055	2D	-
46	056	2E	.
47	057	2F	/
48	060	30	0
49	061	31	1

DEZ	OKTAL	HEX	ASCII
50	062	32	2
51	063	33	3
52	064	34	4
53	065	35	5
54	066	36	6
55	067	37	7
56	070	38	8
57	071	39	9
58	072	3A	:
59	073	3B	;
60	074	3C	<
61	075	3D	=
62	076	3E	>
63	077	3F	?
64	100	40	@
65	101	41	A
66	102	42	B
67	103	43	C
68	104	44	D
69	105	45	E
70	106	46	F
71	107	47	G
72	110	48	H
73	111	49	I
74	112	4A	J
75	113	4B	K
76	114	4C	L
77	115	4D	M
78	116	4E	N
79	117	4F	O
80	120	50	P
81	121	51	Q
82	122	52	R
83	123	53	S
84	124	54	T
85	125	55	U
86	126	56	V
87	127	57	W
88	130	58	X
89	131	59	Y
90	132	5A	Z
91	133	5B	[
92	134	5C	\
93	135	5D	]
94	136	5E	^
95	137	5F	_
96	140	60	`
97	141	61	a
98	142	62	b
99	143	63	c

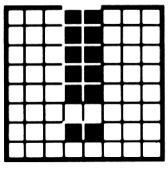
DEZ	OKTAL	HEX	ASCII
100	144	64	d
101	145	65	e
102	146	66	f
103	147	67	g
104	150	68	h
105	151	69	i
106	152	6A	j
107	153	6B	k
108	154	6C	l
109	155	6D	m
110	156	6E	n
111	157	6F	o
112	160	70	p
113	161	71	q
114	162	72	r
115	163	73	s
116	164	74	t
117	165	75	u
118	166	76	v
119	167	77	w
120	170	78	x
121	171	79	y
122	172	7A	z
123	173	7B	{
124	174	7C	|
125	175	7D	}
126	176	7E	~

Maschinenspezifischer BASIC-Graphik-Zeichensatz

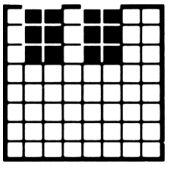
Die hier abgebildeten Zeichen sind in einer 8x8 Matrix dargestellt, wie sie auf dem CPC6128-Bildschirm erscheinen. Benutzer-definierte Zeichen können für spezielle Effekte zusammengestellt werden und bündig aneinander stoßen - siehe 'Benutzer-definierte Zeichen' im 9. Kapitel.



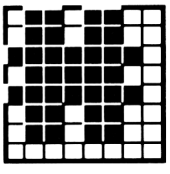
32 &H20
&X00100000



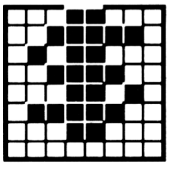
33
&H21
&X00100001



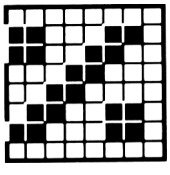
34
&H22
&X00100010



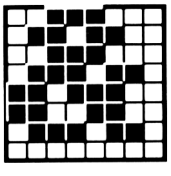
35
&H23
&X00100011



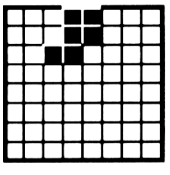
36
&H24
&X00100100



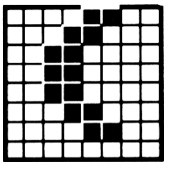
37
&H25
&X00100101



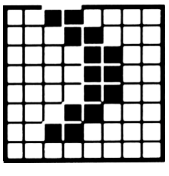
38
&H26
&X00100110



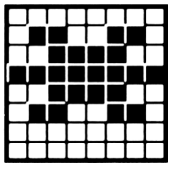
39
&H27
&X00100111



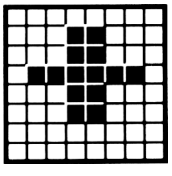
40
&H28
&X00101000



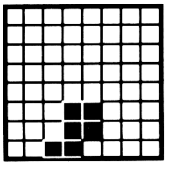
41
&H29
&X00101001



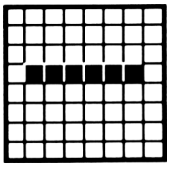
42
&H2A
&X00101010



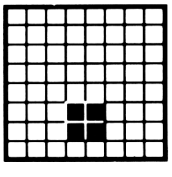
43
&H2B
&X00101011



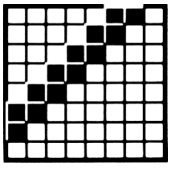
44
&H2C
&X00101100



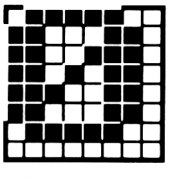
45
&H2D
&X00101101



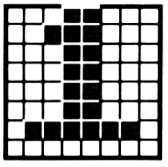
46
&H2E
&X00101110



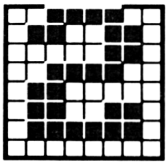
47
&H2F
&X00101111



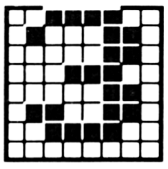
48
&H30
&X00110000



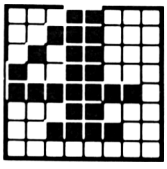
49
&H31
&X00110001



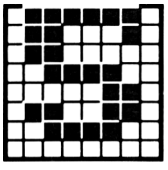
50
&H32
&X00110010



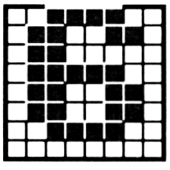
51
&H33
&X00110011



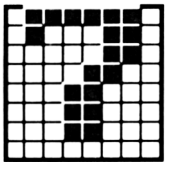
52
&H34
&X00110100



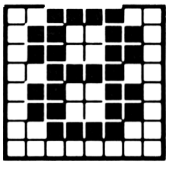
53
&H35
&X00110101



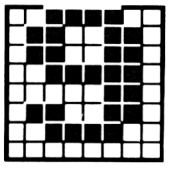
54
&H36
&X00110110



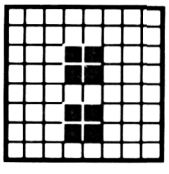
55
&H37
&X00110111



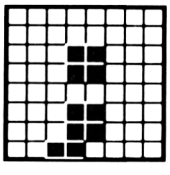
56
&H38
&X00111000



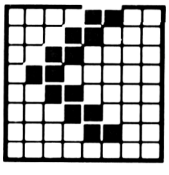
57
&H39
&X00111001



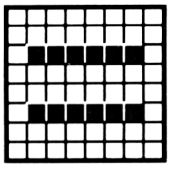
58
&H3A
&X00111010



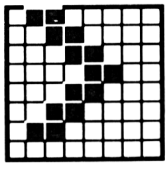
59
&H3B
&X00111011



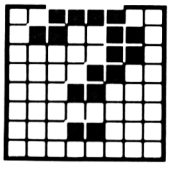
60
&H3C
&X00111100



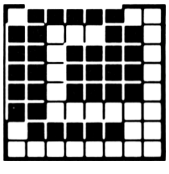
61
&H3D
&X00111101



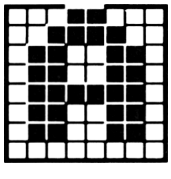
62
&H3E
&X00111110



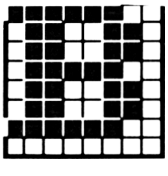
63
&H3F
&X00111111



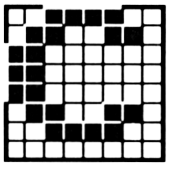
64
&H40
&X01000000



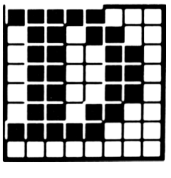
65
&H41
&X01000001



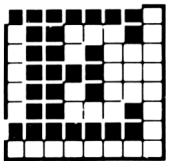
66
&H42
&X01000010



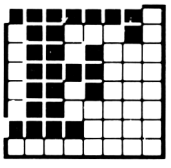
67
&H43
&X01000011



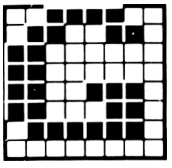
68
&H44
&X01000100



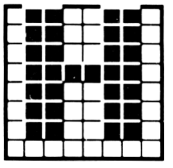
69
&H45
&X01000101



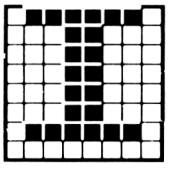
70
&H46
&X01000110



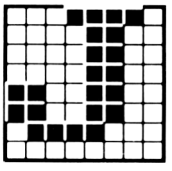
71
&H47
&X01000111



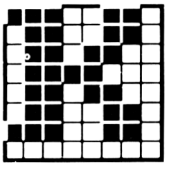
72
&H48
&X01001000



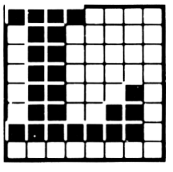
73
&H49
&X01001001



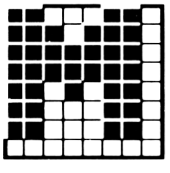
74
&H4A
&X01001010



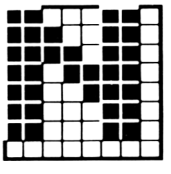
75
&H4B
&X01001011



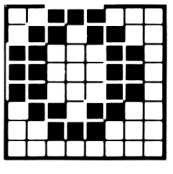
76
&H4C
&X01001100



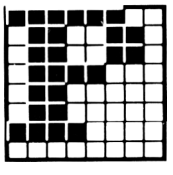
77
&H4D
&X01001101



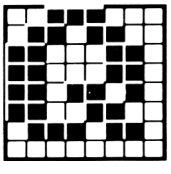
78
&H4E
&X01001110



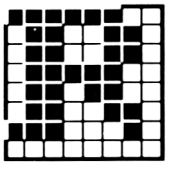
79
&H4F
&X01001111



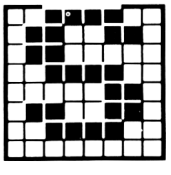
80
&H50
&X01010000



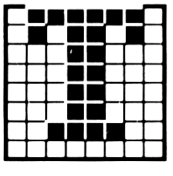
81
&H51
&X01010001



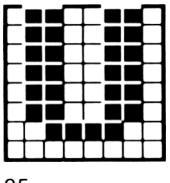
82
&H52
&X01010010



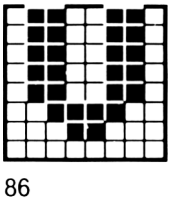
83
&H53
&X01010011



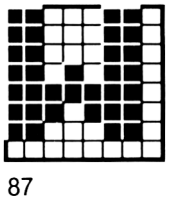
84
&H54
&X01010100



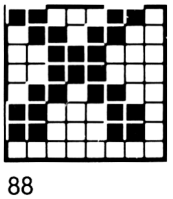
85
&H55
&X01010101



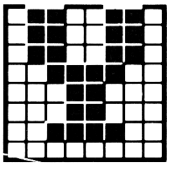
86
&H56
&X01010110



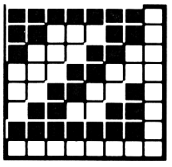
87
&H57
&X01010111



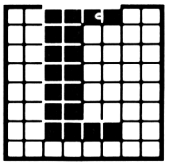
88
&H58
&X01011000



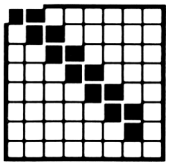
89
&H59
&X01011001



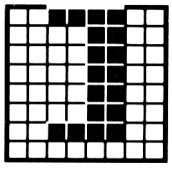
90
&H5A
&X01011010



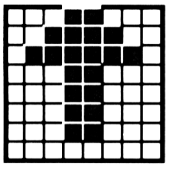
91
&H5B
&X01011011



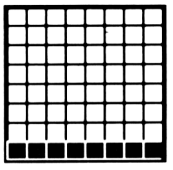
92
&H5C
&X01011100



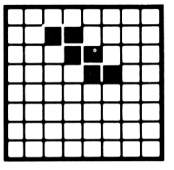
93
&H5D
&X01011101



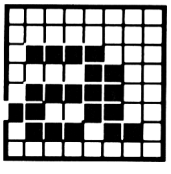
94
&H5E
&X01011110



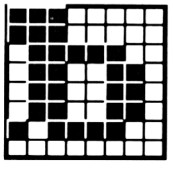
95
&H5F
&X01011111



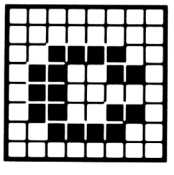
96
&H60
&X01100000



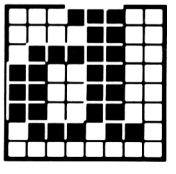
97
&H61
&X01100001



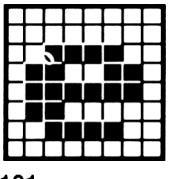
98
&H62
&X01100010



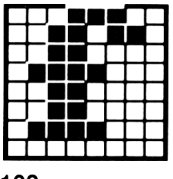
99
&H63
&X01100011



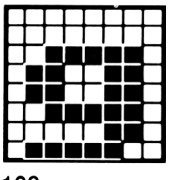
100
&H64
&X01100100



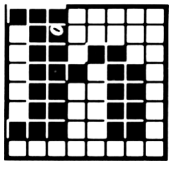
101
&H65
&X01100101



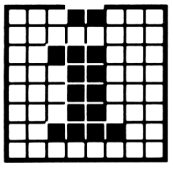
102
&H66
&X01100110



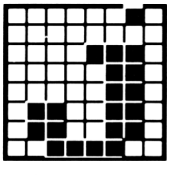
103
&H67
&X01100111



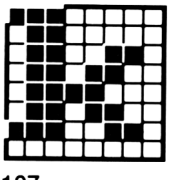
104
&H68
&X01101000



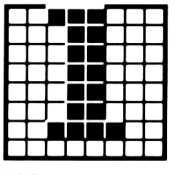
105
&H69
&X01101001



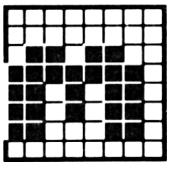
106
&H6A
&X01101010



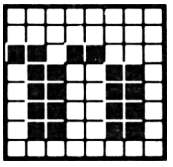
107
&H6B
&X01101011



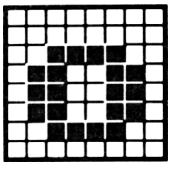
108
&H6C
&X01101100



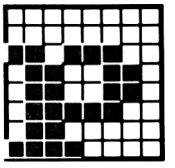
109
&H6D
&X01101101



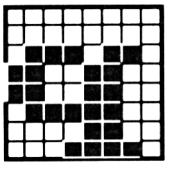
110
&H6E
&X01101110



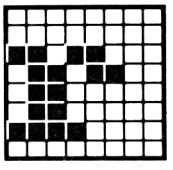
111
&H6F
&X01101111



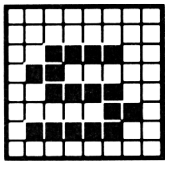
112
&H70
&X01110000



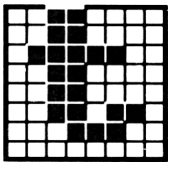
113
&H71
&X01110001



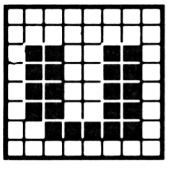
114
&H72
&X01110010



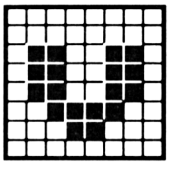
115
&H73
&X01110011



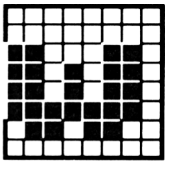
116
&H74
&X01110100



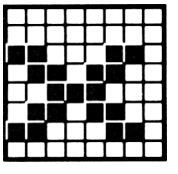
117
&H75
&X01110101



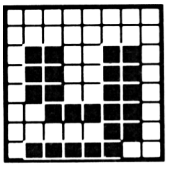
118
&H76
&X01110110



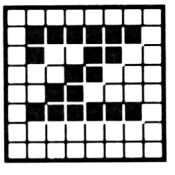
119
&H77
&X01110111



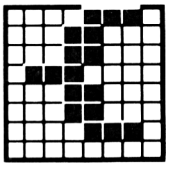
120
&H78
&X01111000



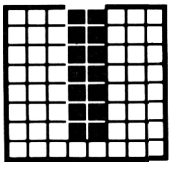
121
&H79
&X01111001



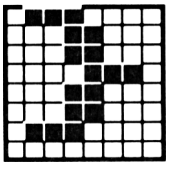
122
&H7A
&X01111010



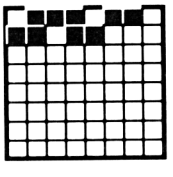
123
&H7B
&X01111011



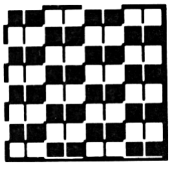
124
&H7C
&X01111100



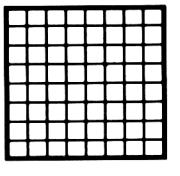
125
&H7D
&X01111101



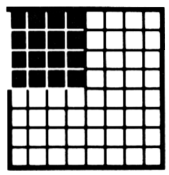
126
&H7E
&X01111110



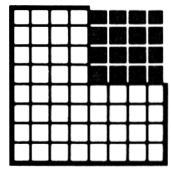
127
&H7F
&X01111111



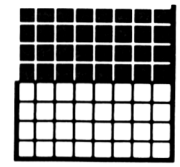
128
&H80
&X10000000



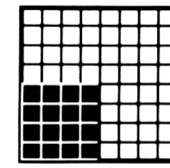
129
&H81
&X10000001



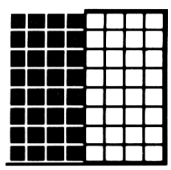
130
&H82
&X10000010



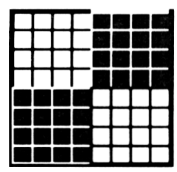
131
&H83
&X10000011



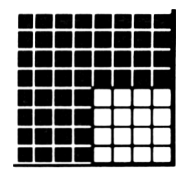
132
&H84
&X10000100



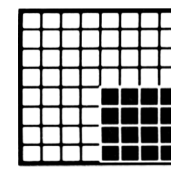
133
&H85
&X10000101



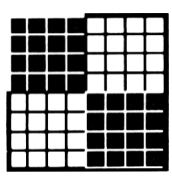
134
&H86
&X10000110



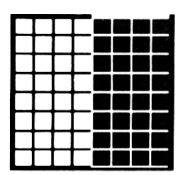
135
&H87
&X10000111



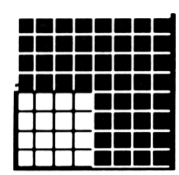
136
&H88
&X10001000



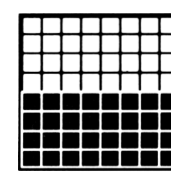
137
&H89
&X10001001



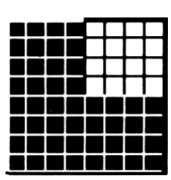
138
&H8A
&X10001010



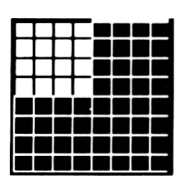
139
&H8B
&X10001011



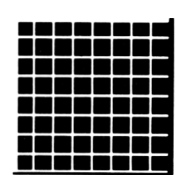
140
&H8C
&X10001100



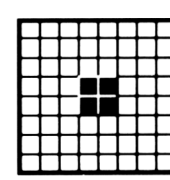
141
&H8D
&X10001101



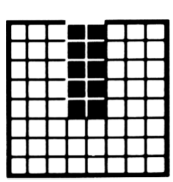
142
&H8E
&X10001110



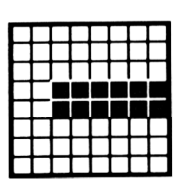
143
&H8F
&X10001111



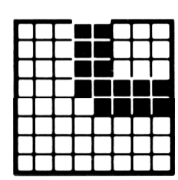
144
&H90
&X10010000



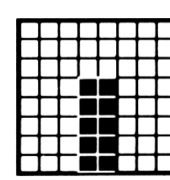
145
&H91
&X10010001



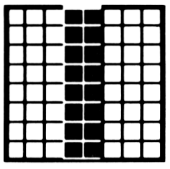
146
&H92
&X10010010



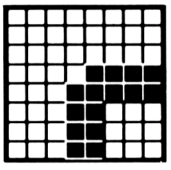
147
&H93
&X10010011



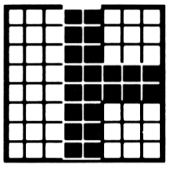
148
&H94
&X10010100



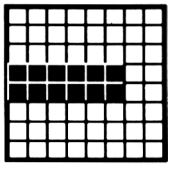
149
&H95
&X10010101



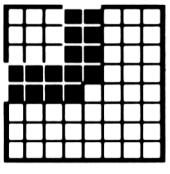
150
&H96
&X10010110



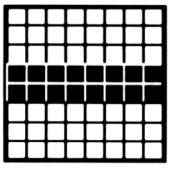
151
&H97
&X10010111



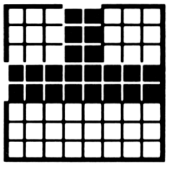
152
&H98
&X10011000



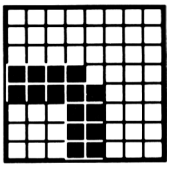
153
&H99
&X10011001



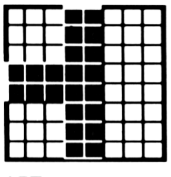
154
&H9A
&X10011010



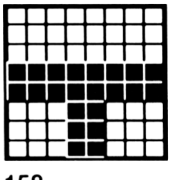
155
&H9B
&X10011011



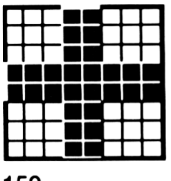
156
&H9C
&X10011100



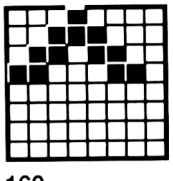
157
&H9D
&X10011101



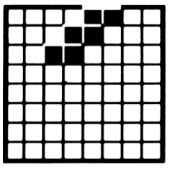
158
&H9E
&X10011110



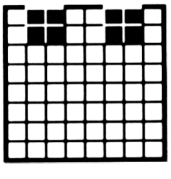
159
&H9F
&X10011111



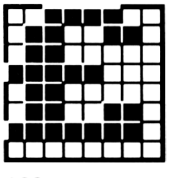
160
&HA0
&X10100000



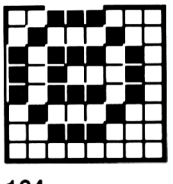
161
&HA1
&X10100001



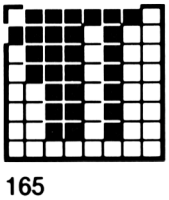
162
&HA2
&X10100010



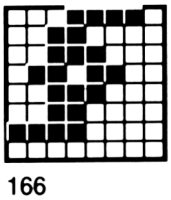
163
&HA3
&X10100011



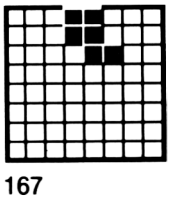
164
&HA4
&X10100100



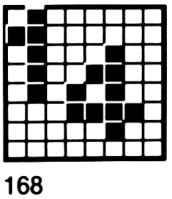
165
&HA5
&X10100101



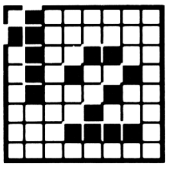
166
&HA6
&X10100110



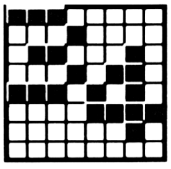
167
&HA7
&X10100111



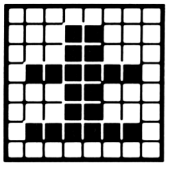
168
&HA8
&X10101000



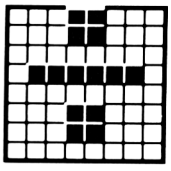
169
&HA9
&X10101001



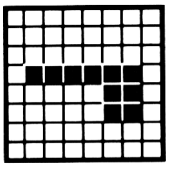
170
&HAA
&X10101010



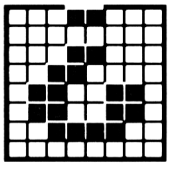
171
&HAB
&X10101011



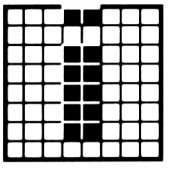
172
&HAC
&X10101100



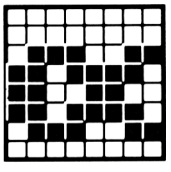
173
&HAD
&X10101101



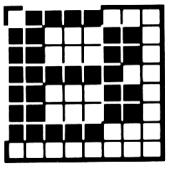
174
&HAE
&X10101110



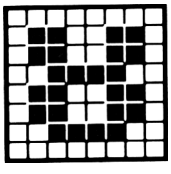
175
&HAF
&X10101111



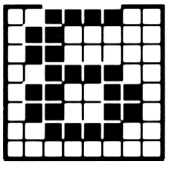
176
&HB0
&X10110000



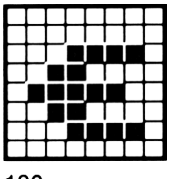
177
&HB1
&X10110001



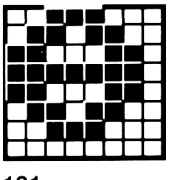
178
&HB2
&X10110010



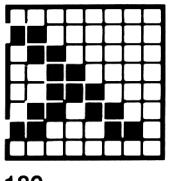
179
&HB3
&X10110011



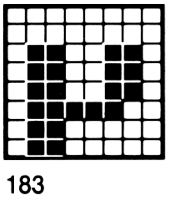
180
&HB4
&X10110100



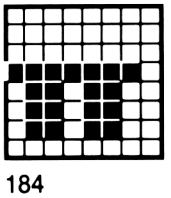
181
&HB5
&X10110101



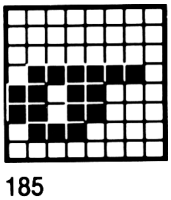
182
&HB6
&X10110110



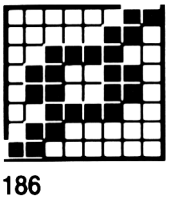
183
&HB7
&X10110111



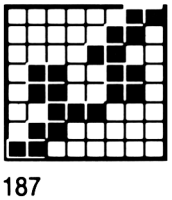
184
&HB8
&X10111000



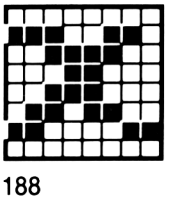
185
&HB9
&X10111001



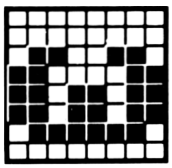
186
&HBA
&X10111010



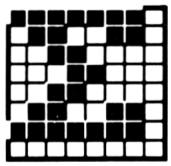
187
&HBB
&X10111011



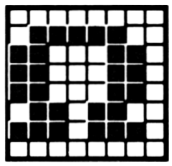
188
&HBC
&X10111100



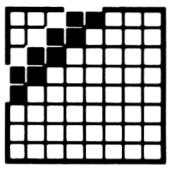
189
&HBD
&X10111101



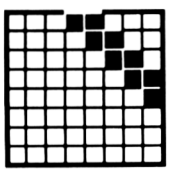
190
&HBE
&X10111110



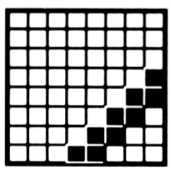
191
&HBF
&X10111111



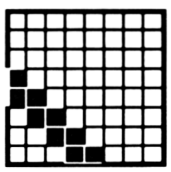
192
&HC0
&X11000000



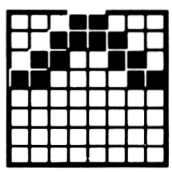
193
&HC1
&X11000001



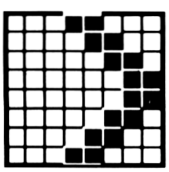
194
&HC2
&X11000010



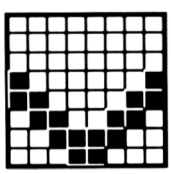
195
&HC3
&X11000011



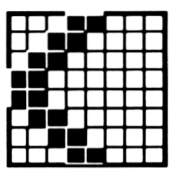
196
&HC4
&X11000100



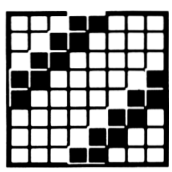
197
&HC5
&X11000101



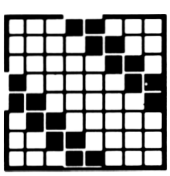
198
&HC6
&X11000110



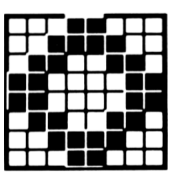
199
&HC7
&X11000111



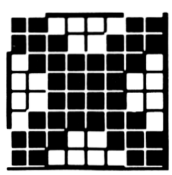
200
&HC8
&X11001000



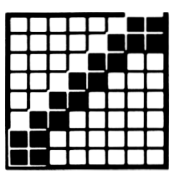
201
&HC9
&X11001001



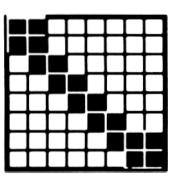
202
&HCA
&X11001010



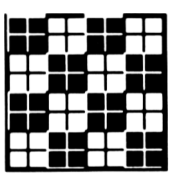
203
&HCB
&X11001011



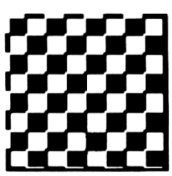
204
&HCC
&X11001100



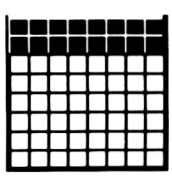
205
&HCD
&X11001101



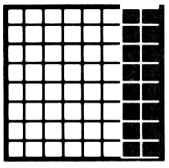
206
&HCE
&X11001110



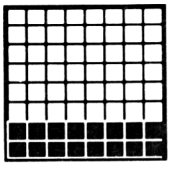
207
&HCF
&X11001111



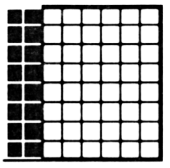
208
&HD0
&X11010000



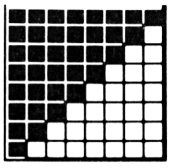
209
&HD1
&X11010001



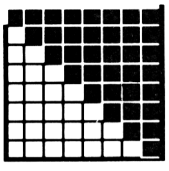
210
&HD2
&X11010010



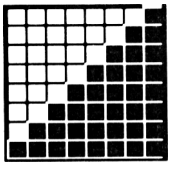
211
&HD3
&X11010011



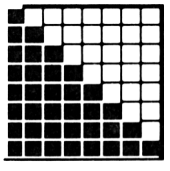
212
&HD4
&X11010100



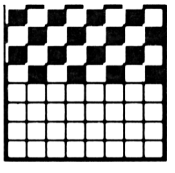
213
&HD5
&X11010101



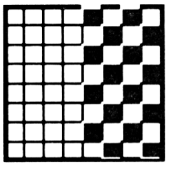
214
&HD6
&X11010110



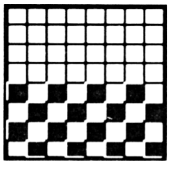
215
&HD7
&X11010111



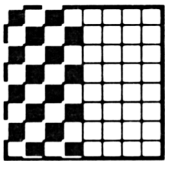
216
&HD8
&X11011000



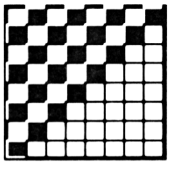
217
&HD9
&X11011001



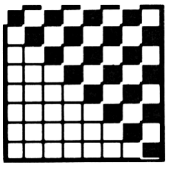
218
&HDA
&X11011010



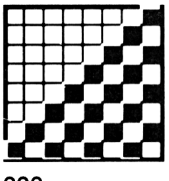
219
&HDB
&X11011011



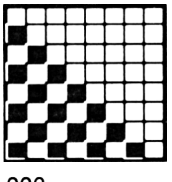
220
&HDC
&X11011100



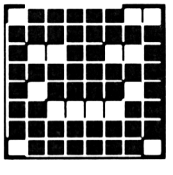
221
&HDD
&X11011101



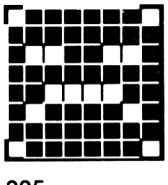
222
&HDE
&X11011110



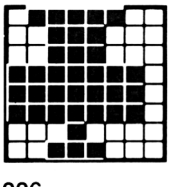
223
&HDF
&X11011111



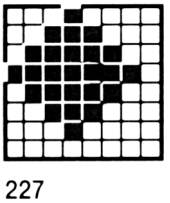
224
&HE0
&X11100000



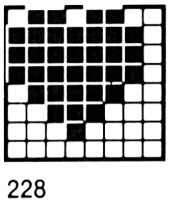
225
&HE1
&X11100001



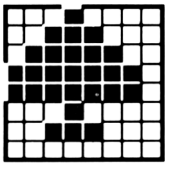
226
&HE2
&X11100010



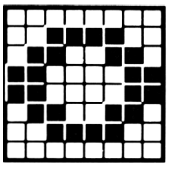
227
&HE3
&X11100011



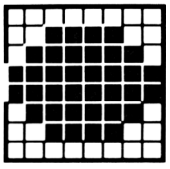
228
&HE4
&X11100100



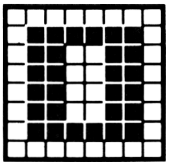
229
&HE5
&X11100101



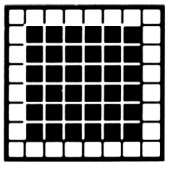
230
&HE6
&X11100110



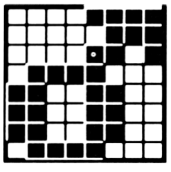
231
&HE7
&X11100111



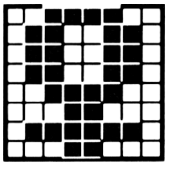
232
&HE8
&X11101000



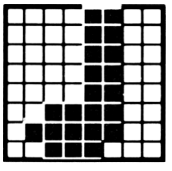
233
&HE9
&X11101001



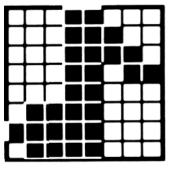
234
&HEA
&X11101010



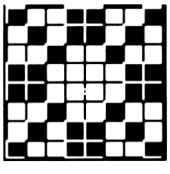
235
&HEB
&X11101011



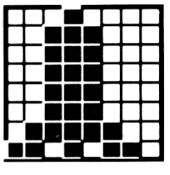
236
&HEC
&X11101100



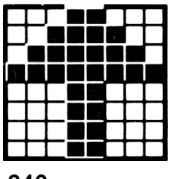
237
&HED
&X11101101



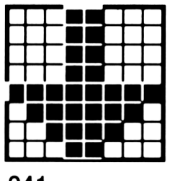
238
&HEE
&X11101110



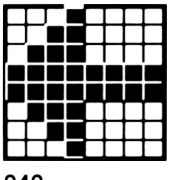
239
&HEF
&X11101111



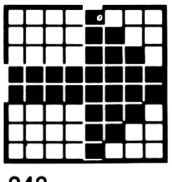
240
&HF0
&X11110000



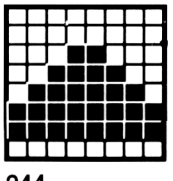
241
&HF1
&X11110001



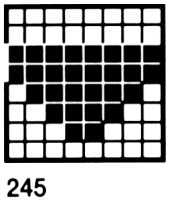
242
&HF2
&X11110010



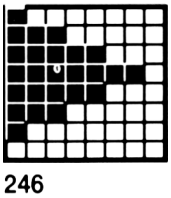
243
&HF3
&X11110011



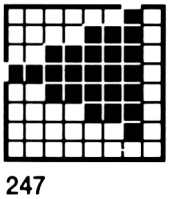
244
&HF4
&X11110100



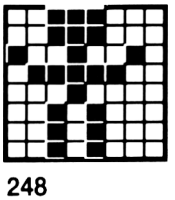
245
&HF5
&X11110101



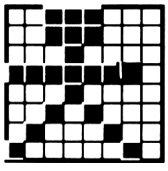
246
&HF6
&X11110110



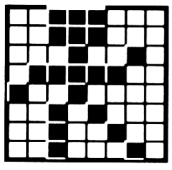
247
&HF7
&X11110111



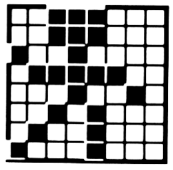
248
&HF8
&X11111000



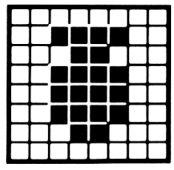
249
&HF9
&X11111001



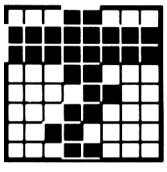
250
&HFA
&X11111010



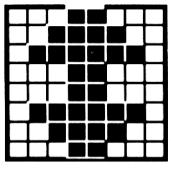
251
&HFB
&X11111011



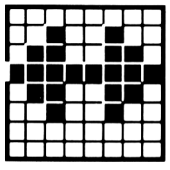
252
&HFC
&X11111100



253
&HFD
&X11111101



254
&HFE
&X11111110



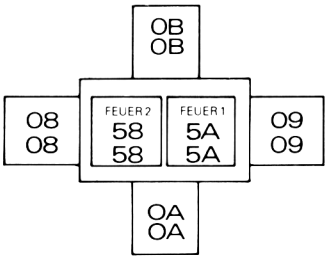
255
&HFF
&X11111111

Teil 4: Tastenbezeichnungen

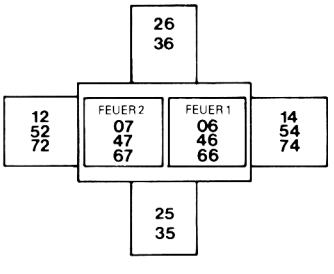
Standard-ASCII-Werte (HEX)

N/A	21 31	7E 22 32	23 33	24 34	25 35	26 36	27 37	28 38	29 39	1F 5F 30	3D 2D	1E A3 5E	10 10 10	7F 7F 7F	N/A	N/A	N/A
E1 09 09	11 51 71	17 57 77	05 45 65	12 52 72	14 54 74	19 59 79	15 55 75	09 49 69	0F 4F 6F	10 50 70	00 7C 40	1B 7B 5B	0D 0D 0D		N/A	N/A	N/A
N/A	01 41 61	13 53 73	04 44 64	06 46 66	07 47 67	08 48 68	0A 4A 6A	0B 4B 6B	0C 4C 6C	2A 3A	2B 3B	1D 7D 5D			N/A	N/A	N/A
N/A	1A 5A 7A	18 58 78	03 43 63	16 56 76	02 42 62	0E 4E 6E	0D 4D 6D	3C 2C	3E 2E	3F 2F	1C 60 5C	N/A			N/A	F8 F4 F0	N/A
N/A	E0 E0 E0				20 20							N/A			FA F6 F2	F9 F5 F1	FB F7 F3

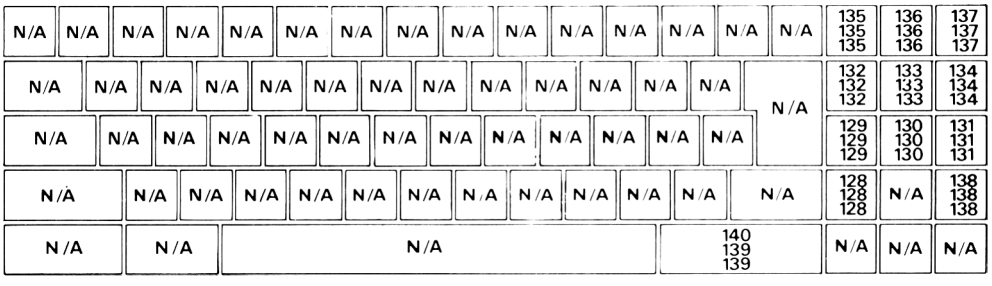
JOYSTICK 0



JOYSTICK 1



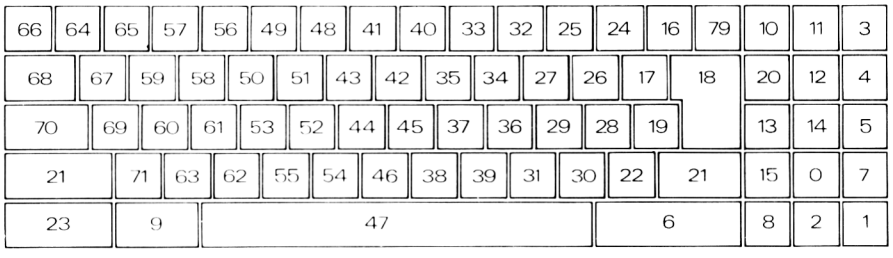
Erweiterungszeichen, Standardpositionen und Werte

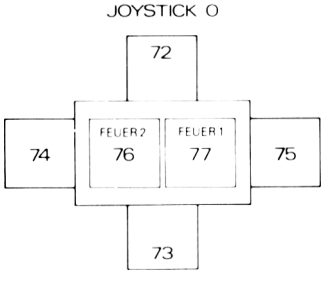


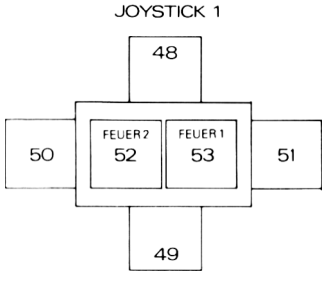
Erweiterungs-zeichen	Standardeinstellung	
	Zeichen	ASCII-Wert
0 (128)	0	&30
1 (129)	1	&31
2 (130)	2	&32
3 (131)	3	&33
4 (132)	4	&34
5 (133)	5	&35
6 (134)	6	&36
7 (135)	7	&37
8 (136)	8	&38
9 (137)	9	&39
10 (138)	.	&2E
11 (139)	[RETURN]	&0D
12 (140)	RUN"[RETURN]	&52 &55 &4E &22 &0D

Anmerkung: Erweiterungszeichen 13 bis 31 (141 bis 159) haben standardmäßig einen Nullwert. Sie erhalten durch das BASIC-Kommando KEY einen Wert und werden durch das Kommando KEY DEF einer Taste zugeordnet.

Nummernzuordnung zu Tasten und Joystick-Signalen







Teil 5: Noten und Tonperioden

Die folgende Tabelle gibt die empfohlenen Tonperioden für die Noten aller acht Oktaven an.

Die erzeugte Frequenz ist nicht genau die gewünschte Frequenz, da die Periodenangabe ganzzahlig sein muß. Der RELATIVE FEHLER gibt die prozentuale Differenz zwischen der gewünschten und der tatsächlich erzeugten Frequenz an.

NOTE	FREQUENZ	PERIODE	RELATIVER FEHLER	
C	16.352	3822	-0.007%	
C#	17.324	3608	+0.007%	
D	18.354	3405	-0.007%	
D#	19.445	3214	-0.004%	
E	20.602	3034	+0.009%	
F	21.827	2863	-0.016%	Oktave -4
F#	23.125	2703	+0.009%	
G	24.500	2551	-0.002%	
G#	25.957	2408	+0.005%	
A	27,500	2273	+0.012%	
A#	29.135	2145	-0.008%	
H	30.868	2025	+0.011%	

NOTE	FREQUENZ	PERIODE	RELATIVER FEHLER	
C	32.703	1911	-0.007%	
C#	34.648	1804	+0.007%	
D	36.708	1703	+0.022%	
D#	38.891	1607	-0.004%	
E	41.203	1517	+0.009%	
F	43.654	1432	+0.019%	Oktave -3
F#	46.249	1351	-0.028%	
G	48.999	1276	+0.037%	
G#	51.913	1204	+0.005%	
A	55.000	1136	-0.032%	
A#	58.270	1073	+0.039%	
H	61.735	1012	-0.038%	

NOTE	FREQUENZ	PERIODE	RELATIVER FEHLER	
C	65.406	956	+0.046%	
C#	69.296	902	+0.007%	
D	73.416	851	-0.037%	
D#	77.782	804	+0.058%	
E	82.407	758	-0.057%	
F	87.307	716	+0.019%	Oktave -2
F#	92.499	676	+0.046%	
G	97.999	638	+0.037%	
G#	103.826	602	+0.005%	
A	110.000	568	-0.032%	
A#	116.541	536	-0.055%	
H	123.471	506	-0.038%	

NOTE	FREQUENZ	PERIODE	RELATIVER FEHLER	
C	130.813	478	+0.046%	
C#	138.591	451	+0.007%	
D	146.832	426	+0.081%	
D#	155.564	402	+0.058%	
E	164.814	379	-0.057%	
F	174.614	358	+0.019%	Oktave -1
F#	184.997	338	+0.046%	
G	195.998	319	+0.037%	
G#	207.652	301	+0.005%	
A	220.000	284	-0.032%	
A#	233.082	268	-0.055%	
H	246.942	253	-0.038%	

NOTE	FREQUENZ	PERIODE	RELATIVER FEHLER	
C	261.626	239	+0.046%	Mittleres C
C#	277.183	225	-0.215%	
D	293.665	213	+0.081%	
D#	311.127	201	+0.058%	
E	329.628	190	+0.206%	
F	349.228	179	+0.019%	Oktave 0
F#	369.994	169	+0.046%	
G	391.995	159	-0.277%	
G#	415.305	150	-0.328%	
A	440.000	142	-0.032%	Kammerton A
A#	466.164	134	-0.055%	
H	493.883	127	+0.356%	

NOTE	FREQUENZ	PERIODE	RELATIVER FEHLER	
C	523.251	119	-0.374%	
C#	554.365	113	+0.229%	
D	587.330	106	-0.390%	
D#	622.254	100	-0.441%	
E	659.255	95	+0.206%	
F	698.457	89	-0.543%	Oktave 1
F#	739.989	84	-0.548%	
G	783.991	80	+0.350%	
G#	830.609	75	-0.328%	
A	880.000	71	-0.032%	
A#	932.328	67	-0.055%	
H	987.767	63	-0.435%	

NOTE	FREQUENZ	PERIODE	RELATIVER FEHLER	
C	1046.502	60	+0.462%	
C#	1108.731	56	-0.662%	
D	1174.659	53	-0.390%	
D#	1244.508	50	-0.441%	
E	1318.510	47	-0.855%	
F	1396.913	45	+0.574%	
F#	1479.978	42	-0.548%	Oktave 2
G	1567.982	40	+0.350%	
G#	1661.219	38	+0.992%	
A	1760.000	36	+1.357%	
A#	1864.655	34	+1.417%	
H	1975.533	32	+1.134%	

NOTE	FREQUENZ	PERIODE	RELATIVER FEHLER	
C	2093.004	30	+0.462%	
C#	2217.461	28	-0.662%	
D	2349.318	27	+1.469%	
D#	2489.016	25	-0.441%	
E	2637.021	24	+1.246%	
F	2793.826	22	-1.685%	
F#	2959.955	21	-0.548%	Oktave 3
G	3135.963	20	+0.350%	
G#	3322.438	19	+0.992%	
A	3520.000	18	+1.357%	
A#	3729.310	17	+1.417%	
H	3951.066	16	+1.134%	

Diese Werte werden alle vom Kammerton A aus wie folgt errechnet:

```
FREQUENZ =440*(2↑(OKTAVE+((N-10)/12)))
PERIODE = ROUND(62500/FREQUENZ)
```

...mit `N=1` für `C`, `2` für `C#`, `3` für `D`, usw.

Teil 6:
BASIC-Fehlermeldungen

1 unexpected NEXT - unerwartetes NEXT

Ein NEXT-Kommando wurde außerhalb einer FOR-Schleife gefunden, oder die Steuervariable im NEXT-Kommando paßt nicht zu der im FOR-Kommando.

2 Syntax Error - Syntaxfehler

BASIC versteht die eingegebene Zeile nicht, da irgendetwas in ihr unzulässig ist.

3 Unexpected RETURN - unerwartetes RETURN

Ein RETURN-Kommando wurde außerhalb eines Unterprogramms gefunden.

4 DATA exhausted - Daten zu Ende

Ein READ-Kommando wurde gegeben, obwohl in DATA-Anweisungen keine Daten mehr vorhanden sind.

5 Improper argument - ungültiges Argument

Dies ist eine Mehrzweck-Meldung. Der Wert eines Arguments in einer Funktion oder ein Befehlsparameter ist falsch.

6 Overflow - Überlauf

Das Ergebnis einer arithmetischen Operation ist 'übergelaufen'. Es kann ein Gleitkommawert-Überlauf sein, weil das Ergebnis größer als 1.7E ↑ 38 ist. Oder es wurde erfolglos versucht, eine Gleitkommazahl in eine 16 Bit große Ganzzahl mit Vorzeichen umzuwandeln.

7 Memory full - Speicher voll

Entweder sind das Programm oder seine Variablen einfach zu umfangreich, oder das Programm ist zu sehr verschachtelt (mit GOSUBs, WHILEs und FORs).

Ein MEMORY-Kommando kann diese Meldung auslösen, falls der BASIC zur Verfügung gestellte Speicherplatz zu klein oder viel zu groß gewählt wurde. Beachten Sie, daß eine geöffnete Datei Pufferplatz belegt, so daß der verfügbare Speicherplatz kleiner wird.

8 Line does not exist - Zeile nicht vorhanden

Die angesprochene Zeile kann nicht gefunden werden.

9 Subscript out of range - Subskript außerhalb des erlaubten Bereichs

Ein Subskript (Indexzahl) in einer Matrix ist zu groß oder zu klein.

10 Array already dimensioned - Matrix schon definiert

Eine der Matrizen eines DIM-Kommandos wurde schon definiert.

11 Division by zero - Division durch Null

Kann bei ganzzahliger Division, Modulo- (Divisionsrest-) Bestimmung oder Potenzierung auftreten.

12 Invalid direct command - Falsches Direktkommando

Das zuletzt versuchte Kommando ist im Direkt-Modus nicht zulässig.

13 Type mismatch - Typenunstimmigkeit

Statt des geforderten String-Wertes wurde ein numerischer Wert eingegeben oder umgekehrt, oder in einem READ- oder INPUT-Befehl wurde ein falscher Zahlentyp eingegeben.

14 String space full - Zeichenfolgen-Bereich voll

Es wurden soviele Zeichenfolgen erstellt, daß selbst kleinste Speicherplätze belegt sind.

15 String too long - Zeichenfolge zu lang

Die Zeichenkette hat mehr als 255 Zeichen. Kann auftreten, wenn mehrere Zeichenketten aneinandergehängt werden.

16 String expression too complex - Textausdruck zu komplex

Textausdrücke können zu viele Zeichenfolgen enthalten. Wenn deren Anzahl eine vernünftige Grenze überschreitet, gibt BASIC diese Meldung aus.

17 Cannot CONTinue - CONT funktioniert nicht

Das Programm kann aus irgendeinem Grund mit CONT nicht wieder aufgenommen werden. Beachten Sie, daß CONT nur für das Wiederstarten eines Programms nach einem STOP-Kommando, **[ESC][ESC]** oder nach einem Fehler ist, und eine zwischenzeitige Programmänderung ein Wiederstarten unmöglich macht.

18 Unknown user function - unbekannte Anwender-Funktion

Für die aufgerufene Funktion wurde kein DEF FN-Kommando gefunden.

19 RESUME missing - RESUME fehlt

Das Programmende wurde während einer Fehlerbehandlungsroutine erreicht, z.B. in einer ON ERROR GOTO -Routine.

20 Unexpected RESUME - Unerwartetes RESUME

RESUME ist nur während einer Fehlerbehandlung erlaubt, z.B. in einer ON ERROR GOTO -Routine.

21 Direct command found - Direktkommando gefunden.

Beim Laden einer Datei wurde eine Zeile ohne Zeilennummer gefunden.

22 Operand missing - Operand fehlt

BASIC ist auf eine unvollständige Anweisung gestoßen.

23 Line too long - Zeile zu lang

Eine Zeile wird im internen BASIC-Format zu lang.

24 EOF met - EOF erreicht

Es wurde versucht, eine Eingabedatei weiter zu lesen, obwohl das Dateiende (end of file) schon erreicht ist.

25 File type error - Falscher Dateityp

Die gelesene Datei ist vom falschen Typ. Mit OPENIN können nur ASCII-Textdateien eröffnet werden. LOAD, RUN usw. sind nur auf Dateitypen anwendbar, die mit SAVE gespeichert wurden.

26 NEXT missing - NEXT fehlt

Ein passendes NEXT-Kommando für ein gegebenes FOR-Kommando kann nicht gefunden werden.

27 File already open - Datei schon geöffnet

Ein OPENIN- oder OPENOUT-Befehl wurde gegeben, obwohl die Datei schon geöffnet ist.

28 Unknown command - Kommando unbekannt

BASIC kennt keine externen Befehle, z.B. solche, die mit einem Balken beginnen.

29 WEND missing - WEND fehlt

Kann für eine gegebene WHILE-Anweisung kein passendes WEND finden.

3Ø Unexpected WEND - unerwartetes WEND

Ein WEND wurde außerhalb einer WHILE-Schleife angetroffen, oder ein WEND paßt nicht zur aktuellen WHILE-Schleife.

31 File not open - Datei nicht geöffnet

(Siehe folgenden Abschnitt mit dem Titel 'AMSDOS Diskettenfehler'.)

32 Broken in

(Siehe folgenden Abschnitt mit dem Titel 'AMSDOS Diskettenfehler'.)

AMSDOS Diskettenfehler

Bei der Bearbeitung von Dateien können diverse Fehler auftreten. BASIC reagiert auf all diese Fehler mit der Fehlermeldung 32. Nähere Information kann nach Auftreten der Fehlernummer über die Funktion DERR eingeholt werden. Die DERR-Werte und ihre Bedeutung sind nachstehend aufgeführt:

AMSDOS-Fehler	DERR-Wert	Fehlerursache
0	Ø oder 22	**[ESC]**wurde gedrückt
14	142(128+14)	Stream ist nicht im richtigen Zustand
15	143(128+15)	Ende der Datei (hard end) erreicht
16	144(128+16)	Fehlerhaftes Kommando, normalerweise unzutreffender Dateiname
17	145(128+17)	Datei dieses Namens besteht bereits
18	146(128+18)	Datei besteht nicht
19	147(128+19)	Inhaltsverzeichnis ist voll
20	148(128+20)	Diskette ist voll
21	149(128+21)	Diskette wurde entnommen, während Dateien geöffnet waren.
22	150(128+22)	Datei nur lesbar
26	154(128+26)	Dateiende (soft end) erreicht

Hat AMSDOS bereits einen Fehler angegeben, wird Bit 7 gesetzt; demzufolge wird der Wert von DERR um 128 verschoben.

Andere von DERR gemeldete Werte stammen vom Disk-Controller und sind bit-signifikant, wobei stets Bit 6 gesetzt ist. Bit 7 gibt an, ob der Fehler von AMSDOS (siehe oben) gemeldet wurde. Die einzelnen Bits haben folgende Bedeutung:

Bit	**Bedeutung**
0	Adressenbezeichnung fehlt
1	Nicht beschreibbar - Diskette mit Schreibschutz
2	Keine Daten - Sektor unbekannt
3	Laufwerk nicht bereit - keine Diskette im Laufwerk
4	Überlauffehler
5	Datenfehler - CRC-Fehler
6	Immer auf 1 gesetzt, um Fehler von Disk-Controller anzuzeigen
7	Auf 1 gesetzt, wenn Fehler bereits von AMSDOS gemeldet wurde

ERR kann auch 31 ausgeben, wenn ein Zugriff versucht wurde, jedoch keine Datei geöffnet war. Die normale Form des Einsatzes von ERR und DERR wäre eine Einbeziehung eines ON ERROR GOTO-Befehls, der über eine kurze Routine prüfen läßt, ob ERR den Wert 31 oder 32 hat. Handelt es sich um 32, kann DERR nach weiteren Informationen über die Art des Fehlers abgefragt werden.

Beispiel:

```
10 ON ERROR GOTO 1000
20 OPENOUT "datei.asc"
30 WRITE #9,"test-daten"
40 CLOSEOUT
50 END
1000 amsdoserr=(DERR AND &7F):REM maskieren mit Bit 7
1010 IF ERR<31 THEN END
1020 IF ERR=31 THEN PRINT "Sind Sie sicher, dass Zeile
     20 richtig geschrieben ist?":END
1030 IF amsdoserr=20 THEN PRINT "Diskette ist voll, ne
     hmen Sie doch eine neue Daten-Diskette":END
1040 IF amsdoserr=&X01001000 THEN PRINT "Legen Sie eine
     Diskette ins Laufwerk, und druecken Sie eine Tast
     e":WHILE INKEY$<>"":
     WEND:RESUME
1050 END
```

Teil 7: BASIC Schlüsselwörter

Im folgenden sind alle AMSTRAD CPC6128 BASIC-Schlüsselwörter aufgeführt. Diese Wörter sind reserviert und dürfen nicht als Variablennamen verwendet werden.

```
ABS, AFTER, AND, ASC, ATN, AUTO

BIN$, BORDER

CALL, CAT, CHAIN, CHR$, CINT, CLEAR, CLG, CLOSEIN, CLOSEOUT, CLS,
CONT, COPYCHR$, COS, CREAL, CURSOR

DATA, DEC$, DEF, DEFINT, DEFREAL, DEFSTR, DEG, DELETE, DERR, DI,
DIM, DRAW, DRAWR
```

EDIT, EI, ELSE, END, ENT, ENV, EOF, ERASE, ERL, ERR, ERROR, EVERY, EXP

FILL, FIX, FN, FOR, FRAME, FRE

GOSUB, GOTO, GRAPHICS

HEX$, HIMEM

IF, INK, INKEY, INKEY$, INP, INPUT, INSTR, INT

JOY

KEY

LEFT$, LEN, LET, LINE, LIST, LOAD, LOCATE, LOG, LOG10, LOWER$

MASK, MAX, MEMORY, MERGE, MID$, MIN, MOD, MODE, MOVE, MOVER

NEXT, NEW, NOT

ON, ON BREAK, ON ERROR GOTO Ø, ON SQ, OPENIN, OPENOUT, OR, ORIGIN, OUT

PAPER, PEEK, PEN, PI, PLOT, PLOTR, POKE, POS, PRINT

RAD, RANDOMIZE, READ, RELEASE, REM, REMAIN, RENUM, RESTORE, RESUME, RETURN, RIGHT$, RND, ROUND, RUN

SAVE, SGN, SIN, SOUND, SPACE$, SPC, SPEED, SQ, SQR, STEP, STOP, STR$, STRING$, SWAP, SYMBOL

TAB, TAG, TAGOFF, TAN, TEST, TESTR, THEN, TIME, TO, TROFF, TRON

UNT, UPPER$, USING

VAL, VPOS

WAIT, WEND, WHILE, WIDTH, WINDOW, WRITE

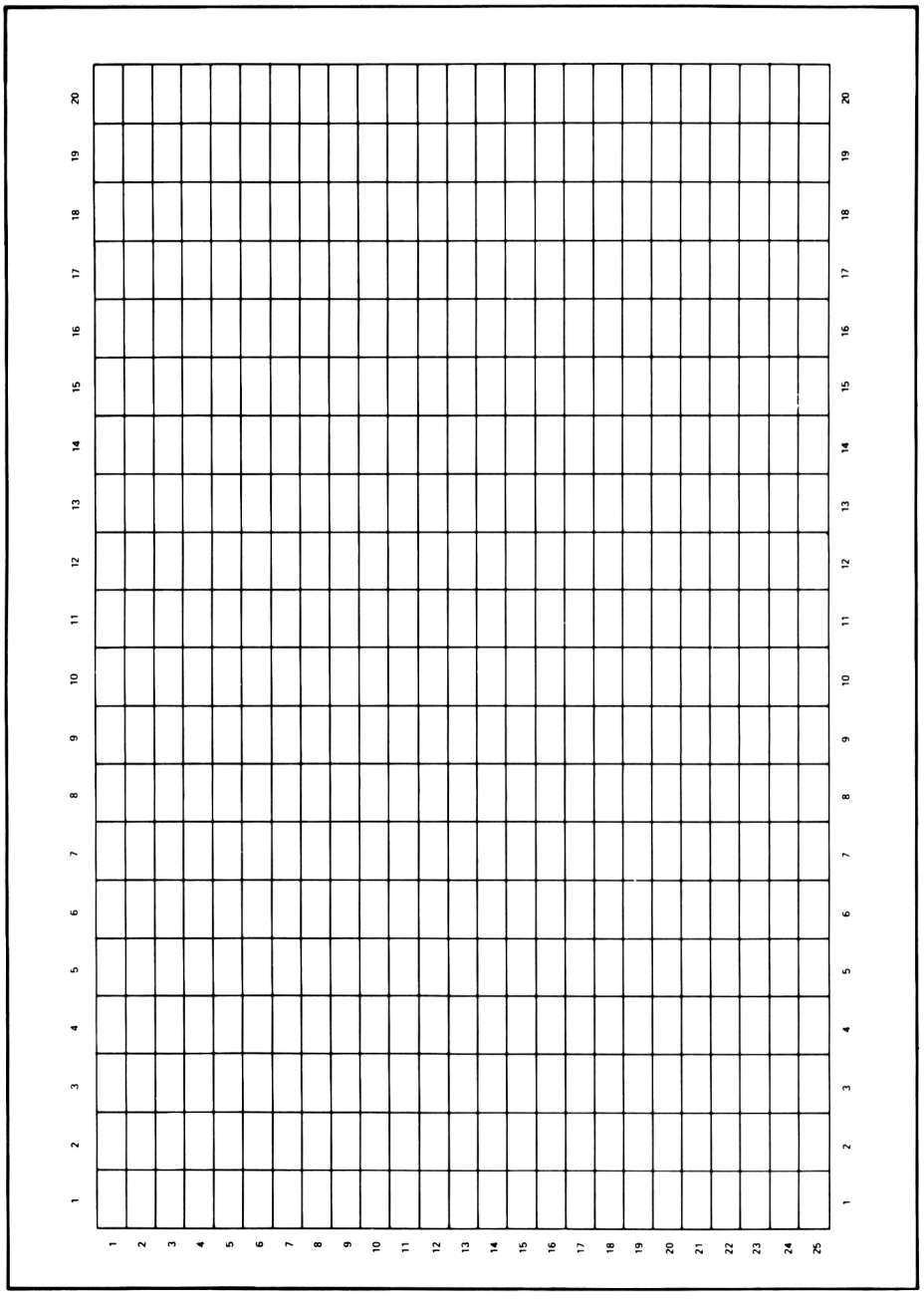
XOR, XPOS

YPOS

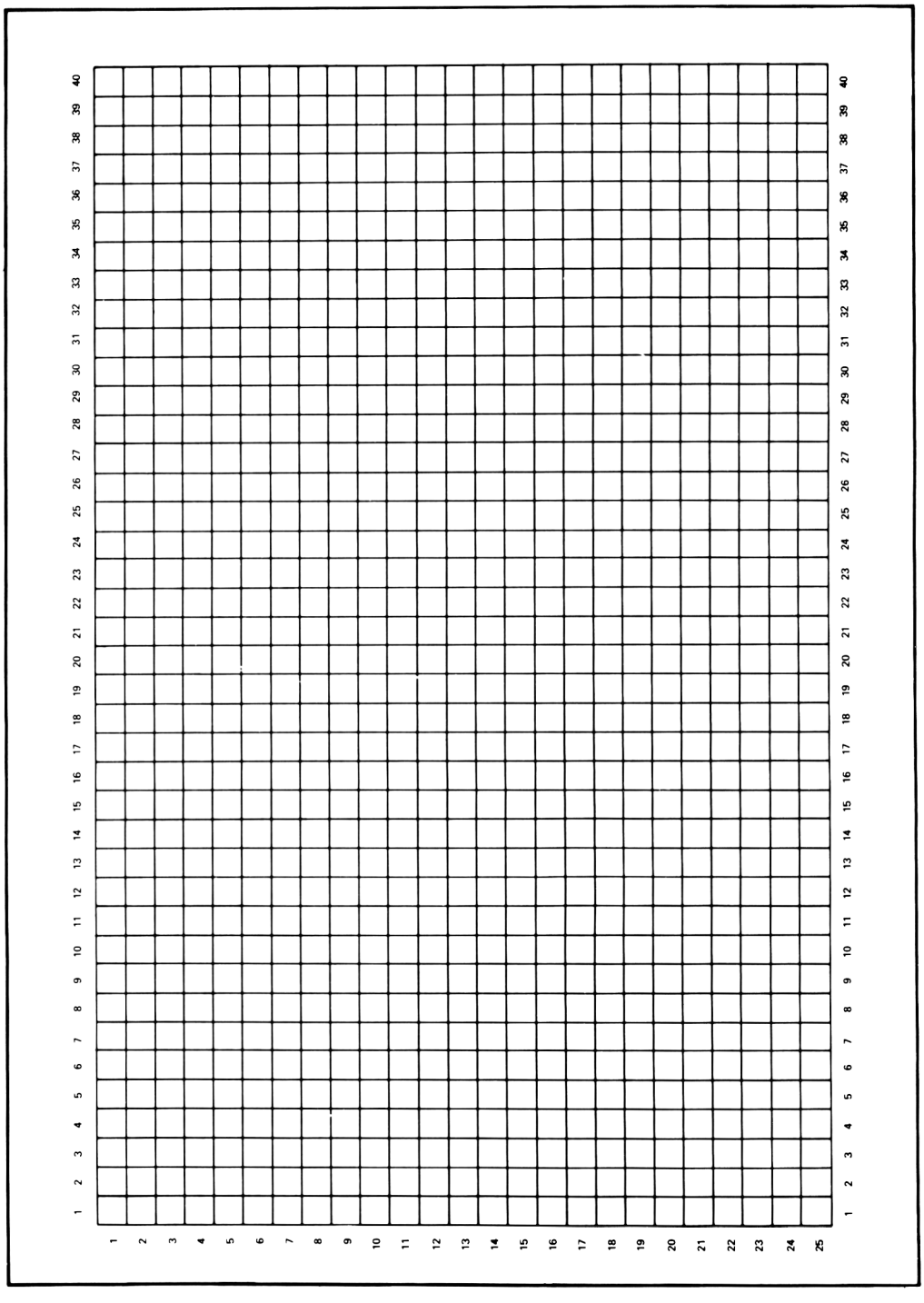
ZONE

Teil 8: Entwurfsgitter

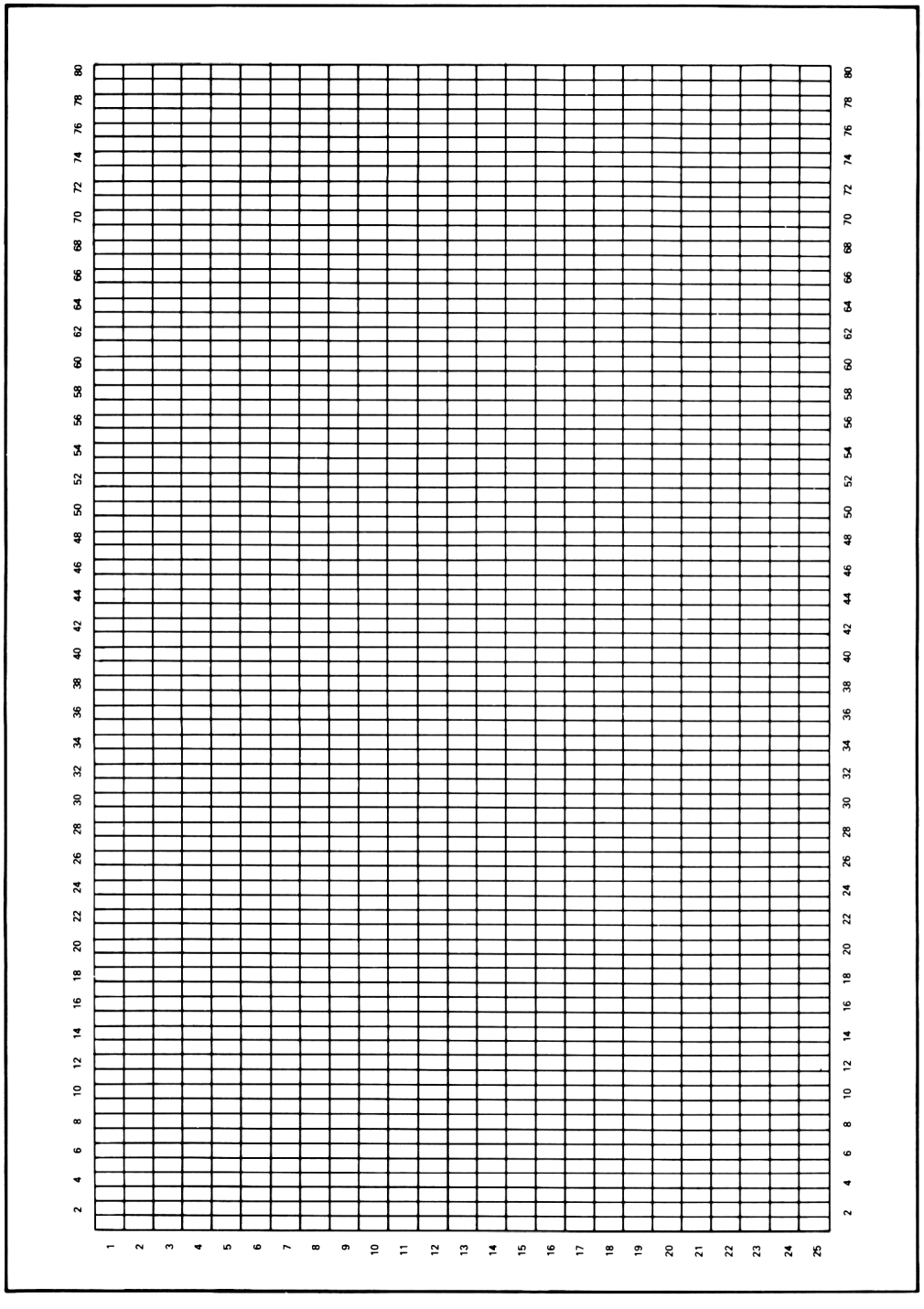
Bildschirmplan für Text- und Window-Entwurf - Modus 0 (20 Spalten)



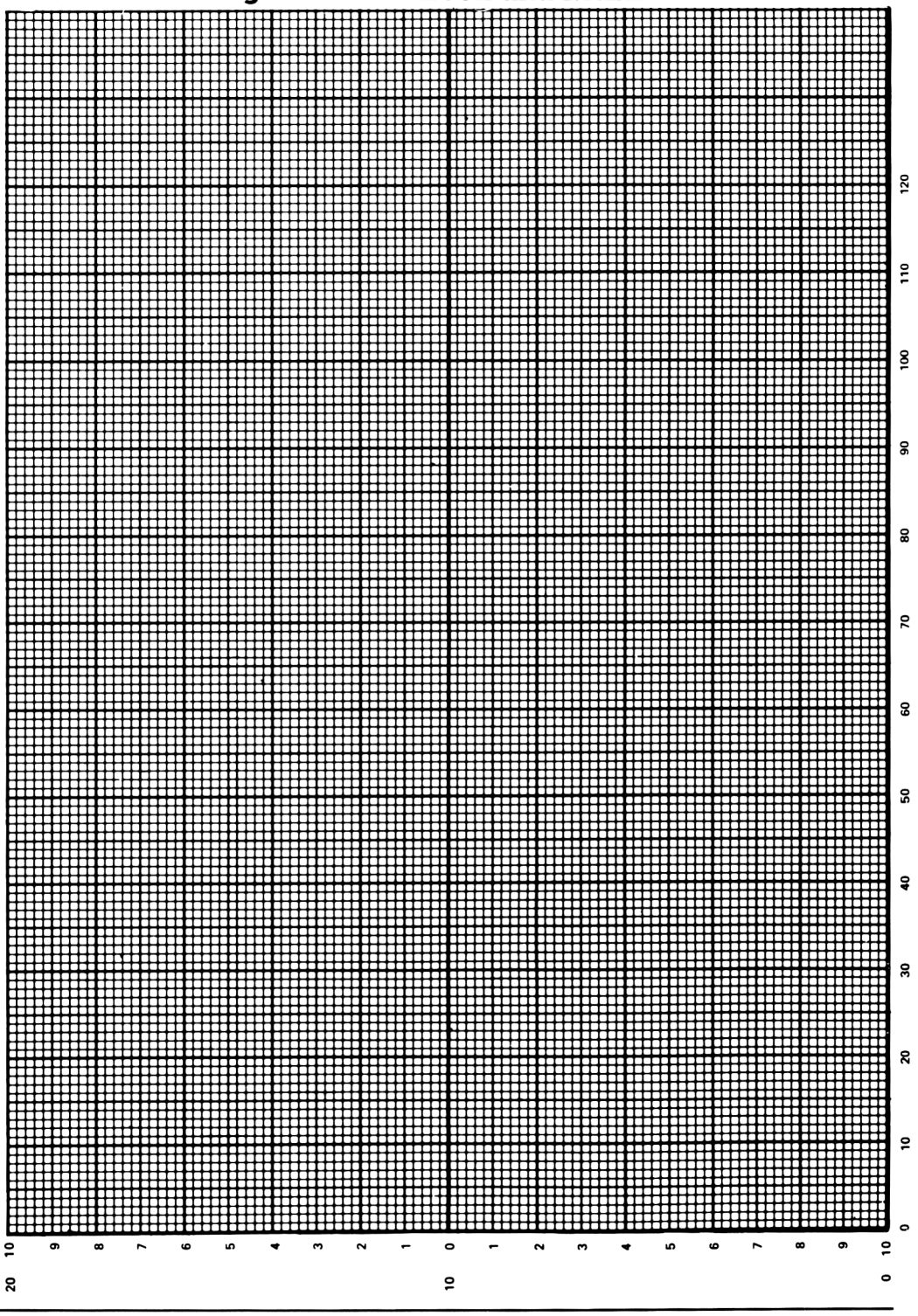
Bildschirmplan für Text- und Window-Entwurf -Modus 1 (40 Spalten)



Bildschirmplan für Text- und Window-Entwurf -Modus 2 (80 Spalten)

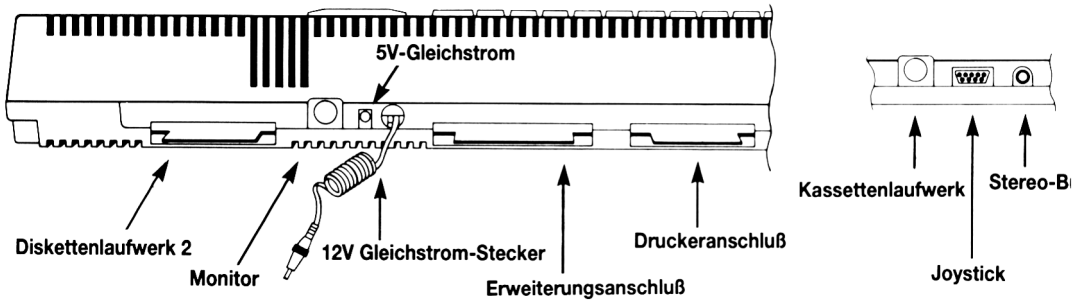


Gitter zur Planung von Hüllkurven und Musik



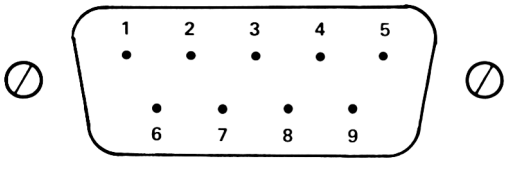
Teil 9: Anschlüsse

CPC6128 Eingangs-/Ausgangs-Buchsen



Joystick-Buchse

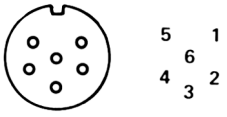
Rückansicht



Stift 1	aufwärts	Stift 6	Feuer 2
Stift 2	abwärts	Stift 7	Feuer 1
Stift 3	links	Stift 8	gemeinsam
Stift 4	rechts	Stift 9	gemeinsam 2
Stift 5	frei		

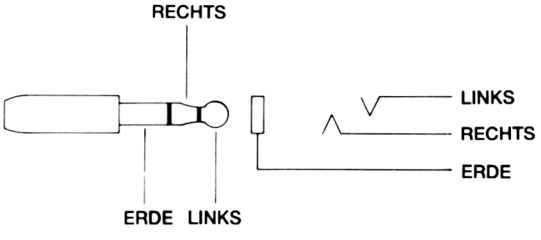
Monitor-Buchse

Rückansicht



Stift 1	rot	Stift 4	Synchronisation
Stift 2	grün	Sttift 5	Erde
Stift 3	blau	Stift 6	Luminanz

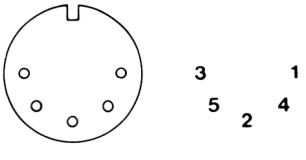
Stereo-Anschluß



Stift 1	linker Kanal
Stift 2	rechter Kanal
Stift 3	Erde

Kassetten-Buchse

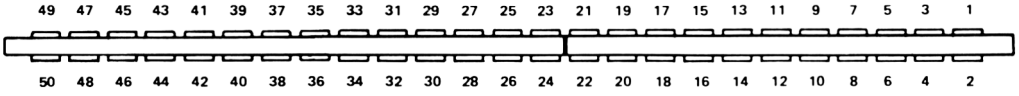
Rückansicht



Stift 1	Fernsteuerung	Stift 4	Dateneingang
Stift 2	Erde	Stift 5	Datenausgang
Stift 3	Fernsteuerung		

Erweiterungs-Anschluß

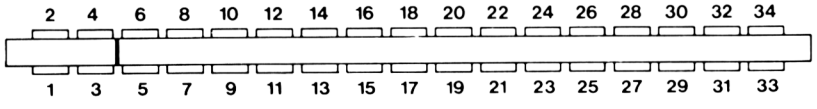
Rückansicht



PIN	1	SOUND	PIN	18	A0	PIN	35	$\overline{\text{INT}}$
PIN	2	GND	PIN	19	D7	PIN	36	$\overline{\text{NMI}}$
PIN	3	A15	PIN	20	D6	PIN	37	$\overline{\text{BUSR2}}$
PIN	4	A14	PIN	21	D5	PIN	38	$\overline{\text{BUSAK}}$
PIN	5	A13	PIN	22	D4	PIN	39	READY
PIN	6	A12	PIN	23	D3	PIN	40	$\overline{\text{BUS RESET}}$
PIN	7	A11	PIN	24	D2	PIN	41	$\overline{\text{RESET}}$
PIN	8	A10	PIN	25	D1	PIN	42	$\overline{\text{ROMEN}}$
PIN	9	A9	PIN	26	D0	PIN	43	$\overline{\text{ROMDIS}}$
PIN	10	A8	PIN	27	+ 5v	PIN	44	$\overline{\text{RAMRD}}$
PIN	11	A7	PIN	28	$\overline{\text{MREQ}}$	PIN	45	RAMDIS
PIN	12	A6	PIN	29	$\overline{\text{M1}}$	PIN	46	CURSOR
PIN	13	A5	PIN	30	$\overline{\text{RFSH}}$	PIN	47	L. PEN
PIN	14	A4	PIN	31	$\overline{\text{IORQ}}$	PIN	48	$\overline{\text{EXP}}$
PIN	15	A3	PIN	32	$\overline{\text{RD}}$	PIN	49	GND
PIN	16	A2	PIN	33	$\overline{\text{WR}}$	PIN	50	ϕ
PIN	17	A1	PIN	34	$\overline{\text{HALT}}$			

Anschluß für zweites Diskettenlaufwerk

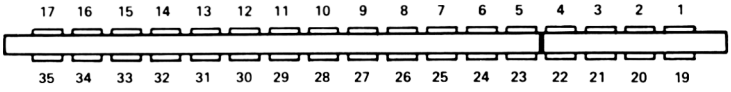
Rückansicht



PIN 1	$\overline{\text{READY}}$	PIN 18	$\overline{\text{GND}}$
PIN 2	$\overline{\text{GND}}$	PIN 19	MOTOR ON
PIN 3	SIDE 1 SELECT	PIN 20	GND
PIN 4	$\overline{\text{GND}}$	PIN 21	N/C
PIN 5	READ DATA	PIN 22	$\overline{\text{GND}}$
PIN 6	$\overline{\text{GND}}$	PIN 23	DRIVE SELECT 1
PIN 7	WRITE PROTECT	PIN 24	GND
PIN 8	$\overline{\text{GND}}$	PIN 25	N/C
PIN 9	TRACK 0	PIN 26	$\overline{\text{GND}}$
PIN 10	$\overline{\text{GND}}$	PIN 27	INDEX
PIN 11	WRITE GATE	PIN 28	GND
PIN 12	$\overline{\text{GND}}$	PIN 29	N/C
PIN 13	WRITE DATA	PIN 30	GND
PIN 14	$\overline{\text{GND}}$	PIN 31	N/C
PIN 15	STEP	PIN 32	GND
PIN 16	$\overline{\text{GND}}$	PIN 33	N/C
PIN 17	DIRECTION SELECT	PIN 34	GND

Drucker-Anschluß

Rückansicht



PIN	1	$\overline{\text{STROBE}}$	PIN	19	GND
PIN	2	D0	PIN	20	GND
PIN	3	D1	PIN	21	GND
PIN	4	D2	PIN	22	GND
PIN	5	D3	PIN	23	GND
PIN	6	D4	PIN	24	GND
PIN	7	D5	PIN	25	GND
PIN	8	D6	PIN	26	GND
PIN	9	GND	PIN	27	GND
PIN	11	BUSY	PIN	28	GND
PIN	14	GND	PIN	33	GND
PIN	16	GND	Alle weiteren Stifte NC		

Teil 10: Drucker

Drucker-Schnittstelle

Das CPC6128-System ermöglicht den Anschluß und Betrieb eines Indurstrienorm-Druckers mit 'Centronics-Schnittstelle'.

Das Druckerkabel ist eine einfache eins-zu-eins-Verbindung zwischen der PRINTER-Buchse an der Rückseite des Computers und der Anschlußstelle des Parallel-Druckers. Beachten Sie, daß die Platine des Computers zwei 'Finger' weniger hat als die Anschlußstelle des Druckers. Dies ermöglicht die Verwendung von Standard-Platinensteckern.

Die Belegung der Schnittstelle im einzelnen ist im 9. Teil dieses Kapitels behandelt.

Das Kabel sollte Stift 1 des Computers mit Stift 1 des Druckers, Stift 19 des Computers mit Stift 19 des Druckers usw. verbinden. Die Stifte 18 und 36 sollten jedoch nicht mit dem Computer verbunden werden.

Beachten Sie, daß sich 17 'Finger' in der oberen Reihe der Drucker-Anschlußleiste befinden, daß jedoch die untere Reihe mit Nummer 19 und nicht mit 18 beginnt. Damit ist gewährleistet, daß die Leitungen jeweils den Finger der Computer-Platine mit dem gleich numerierten Stift der Druckerbuchse verbinden.

Der Computer setzt das BUSY-Signal (Stift 11) ein, um zu prüfen, ob der Drucker bereit ist; er wartet, wenn sich der Drucker im OFF-LINE-Zustand befindet.
Sie brauchen hierbei keine besonderen Benutzer-Befehle einzugeben; durch Angabe von #8 wird die Ausgabe zum Drucker geschickt.

Die Drucker-Schnittstelle im CPC6128 ist so ausgelegt, daß nicht nur einfache Matrixdrucker, sondern mit entsprechender Schnittstelle auch Typenraddrucker, Graphikplotter und Farb-Tintenstrahldrucker angeschlossen werden können. In jedem Falle wird die erforderliche Kompatibilität durch die genormte parallele Schnittstelle gewährleistet.

Die kundenspezifisch angepaßte Software im AMSTRAD NLQ401 ermöglicht den Punktgraphik-Betrieb und das Ausdrucken ganzer Bildschirmabzüge.

Der NLQ401 ist außerdem in der Lage, den kompletten deutschen Zeichensatz zu drucken.

Teil 11: Joysticks

Die im AMSTRAD CPC6128 eingebaute Software unterstützt einen oder zwei Joysticks, die wie ein Teil der Tastatur behandelt werden und deshalb mit `INKEY` und `INKEY$` abgefragt werden können.

Hierzu ist anzumerken, daß der 6128 in den meisten Fällen den Haupt-Feuerknopf des Joysticks als 'Feuer' 2 interpretiert.

Die Funktionen `JOY(Ø)` und `JOY(1)` ermöglichen eine direkte Zustandsprüfung des ersten bzw. zweiten Joysticks. Die Funktion liefert ein bit-signifikantes Ergebnis, das die Bewegungsrichtung des Joysticks zum Zeitpunkt der letzten Tastatur-Abfrage anzeigt.

Die nachstehende Tabelle zeigt die von beiden Joysticks gelieferten Werte. Hinter den Werten, die nach einem JOY-Befehl ausgegeben werden, sind die entsprechenden Tastenwerte aufgeführt, die z.B. in `INKEY`- oder `KEY DEF`-Befehlen als Parameter eingesetzt werden.

Bewegungs-richtung	JOY-Befehl		Tastenwerte		
	Bit	angezeigter Wert	Erster Joystick	Zweiter Joystick	Entsprechende Taste
aufwärts	0	1	72	48	'6'
abwärts	1	2	73	49	'5'
links	2	4	74	50	'R'
rechts	3	8	75	51	'T'
Feuer 2	4	16	76	52	'G'
Feuer 1	5	32	77	53	'F'

Wenn der zweite Joystick geprüft wird, kann der Computer nicht unterscheiden, ob der Joystick oder die entsprechende Taste der Tastatur (siehe letzte Spalte in obiger Tabelle) bedient wurde. Dies bedeutet, daß anstelle des zweiten Joysticks auch die Tastatur verwendet werden kann.

Teil 12: Disketten-Organisation

Das BIOS unterstützt drei Diskettenformate: SYSTEM-, DATA ONLY-und IBM-Format. Unter AMSDOS wird das Diskettenformat automatisch beim Zugriff auf eine Diskette mit noch nicht eröffneten Dateien erkannt. Um diese automatische Unterscheidung zu ermöglichen, besitzt jedes Format eigene Sektor-Nummern.

3 Zoll-Disketten sind beidseitig verwendbar. Der Computer kann jedoch immer nur auf den Datenbestand einer Diskettenseite zugreifen. (Zum Bearbeiten der anderen Seite wird die Diskette umgedreht.) Die beiden Seiten der Diskette können unterschiedliche Formate besitzen.

Gemeinsame Eigenschaften aller Formate

- Die beiden Seiten einer 3 Zoll-Diskette werden nie gleichzeitig bearbeitet (single sided)

- Physikalische Sektorgröße: 512 Byte
- 40 Spuren, von 0 bis 39 numeriert
- CP/M-Blockgröße: 1024 Byte
- 64 Plätze im Inhaltsverzeichnis

System-Format

9 Sektoren pro Spur, mit den Nummern &41 bis &49
2 reservierte Spuren

Das System-Format ist das wichtigste Format, da CP/M nur von einer Diskette mit System-Format geladen werden kann. Beim Warmstart mit CP/M 2.2 muß ebenfalls eine System-Format-Diskette eingelegt werden. Die reservierten Spuren werden wie folgt verwendet:

	CP/M 2.2	CP/M Plus
Spur 0 Sektor &41	Urlader für CP/M 2.2	Urlader für CP/M Plus
Spur 0 Sektor &42	Konfigurations Sektor	unbenutzt
Spur 0 Sektoren &43 bis &47	unbenutzt	
Spur 0 Sektoren &48 und &49 Spur 1 Sektoren &41 bis &49	CCP und BDOS	

Hinweis: Das `VENDOR`-Format ist eine spezielle Variante des System-Formats, das keine Software auf den beiden reservierten Spuren enthält. Dieses Format wird für handelsübliche Software verwendet.

DATA-ONLY-Format

9 Sektoren pro Spur, mit den Nummern &C1 bis &C9
Keine reservierten Spuren

Dieses Format ist nicht für die Anwendung mit CP/M 2.2 geeignet, da mit ihm kein 'Warmstart' möglich ist. Wird jedoch nur CP/M Plus oder AMSDOS verwendet, steht mit einer DATA-ONLY-Diskette etwas mehr Speicherplatz zur Verfügung.

IBM-FORMAT *nur für CP/M 2.2

8 Sektoren pro Spur, mit den Nummern &1 bis &8
Eine reservierte Spur

Dieses Format ist logisch das gleiche Format wie das mit CP/M auf dem IBM PC verwendete Single-Side-Format. Der CPC6128 kann auf IBM-Format-Disketten schreiben, bzw. von ihnen lesen, er kann sie jedoch nicht kopieren oder IBM-Format erzeugen.

Teil 13: Eingebaute Systemerweiterungen (RSX)

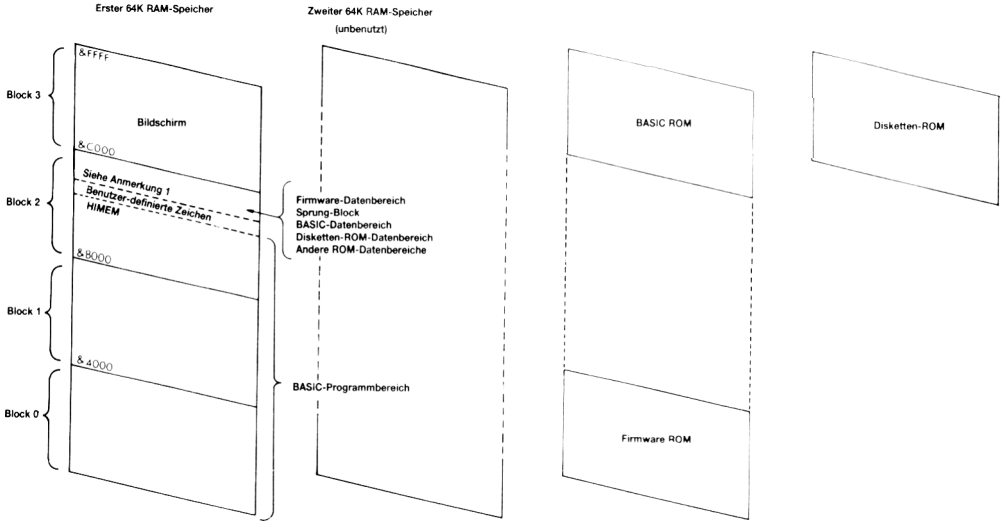
Im 5. Kapitel wurden unter dem Titel 'AMSDOS' externe Befehle vorgestellt. Externe Befehle dienen im Grunde dazu, den BASIC-Wortschatz zu erweitern. Diesen zusätzlichen Befehlen wird ein Balken | vorangestellt. Die Maschinenanweisungen für die neuen AMSDOS-Befehle sind im ROM enthalten; die Einbeziehung der neuen Befehle erfolgt automatisch mit dem Einschalten des 6128 BASIC.

Es ist auch möglich, weitere externe Befehle nach dem BASIC-Start hinzuzufügen, indem die Maschinenanweisungen ins RAM geladen werden. Derartige neue Befehle werden RSXe (Resident System eXtensions) genannt und arbeiten genauso wie ROM-Erweiterungen. RSXe müssen jedesmal, wenn BASIC gestartet (oder wieder gestartet) wird, von Diskette oder Kassette geladen werden. RSXe werden gewöhnlich verwendet, um irgendein intelligentes Peripherie-Gerät, wie einen Lichtgriffel oder Sprachsynthesizer, zu steuern.

Im 8. Kapitel wird beschrieben, wie RSXe benutzt werden, um auf die zweiten 64K im Speicherbereich des 6128 zuzugreifen.

Teil 14: Speicher

Der CPC 6128 besitzt 128K RAM und 48K ROM. Dieser Speicherplatz steht BASIC 1.1 zur Verfügung, wie die Skizze unten darstellt. Die ersten 64K RAM sind nominell in vier Blöcke zu je 16K eingeteilt, die Block 0 bis Block 3 genannt werden. Der Bildschirm benutzt Block 3, und der obere Teil von Block 2 ist wie angegeben mit System-Variablen belegt.



Anmerkung 1: Abhängig von zusätzlich angeschlossenen ROMs; &A6FC, wenn keine ROMs angeschlossen sind.

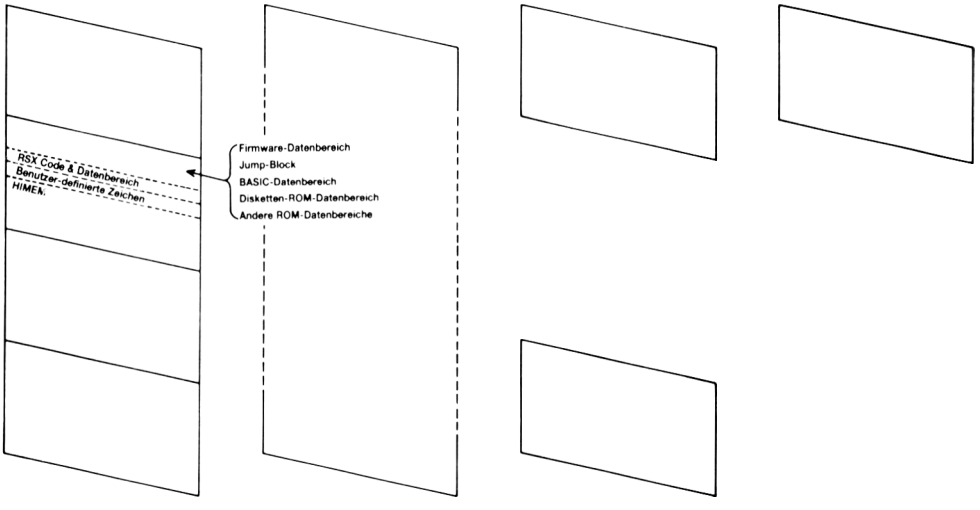
Speicherbelegungsplan für BASIC 1.1

Die Benutzer-definierten Zeichen werden anfänglich direkt über HIMEM (die höchste dem Benutzer zur Verfügung stehende Speicheradresse) plaziert. HIMEM kann durch den `MEMORY`-Befehl verändert werden. Wenn AMSDOS-Dateien geöffnet sind, wird HIMEM automatisch um 4K heruntergesetzt, um einen Pufferbereich zu schaffen. Die Anzahl der Benutzer-definierten Zeichen kann nur verändert werden, wenn HIMEM seit der letzten Definition von Zeichen nicht geändert wurde (es sei denn, der Befehl `SYMBOL AFTER 256` ließ beim letzten Mal keine Zeichen als Benutzer-definierbar zu). Wenn BASIC gestartet wird, sind soviele Zeichen Benutzer-definierbar, als wäre die Anweisung `SYMBOL AFTER 240` gegeben worden.

Es ist deshalb zu empfehlen, den Speicherbereich für Benutzer-definierte Zeichen vor einer dauerhaften Neu-Festlegung von HIMEM zu löschen und die Benutzer-definierten Zeichen in einem neuen Bereich anzusiedeln. Dadurch kann in nachfolgenden Programmen der Wert von **SYMBOL AFTER** geändert werden.

Das folgende Beispiel zeigt diesen Vorgang, wenn **HIMEM** im Zusammenhang mit dem Laden eines RSX tiefer gesetzt wird:

```
100 SYMBOL AFTER 256 ' benutzer-def. zeichen loeschen
110 rsxadresse=HIMEM-rsxlaenge
120 MEMORY rsxadresse-1
130 LOAD "rsxcode",rsxadresse
140 CALL rsxadresse ' rsx log in
150 SYMBOL AFTER 140 ' benutzer-def. zeichen wiederhers
    tellen
```



Speicherbelegungsplan mit RSX an empfohlener Position

Zusätzliche Ein-/Ausgabe

Die meisten Eingabe-/Ausgabeadressen werden vom Computer belegt, insbesondere sollten Adressen unterhalb &7FFF nicht benutzt werden.

Der Adressenteil A0 - A7 sollte die Art des externen Ein-/Ausgabegerätes angeben, während die Adressenleitungen A8 und A9 decodiert werden können, um Register innerhalb des Ein-/Ausgabegerätes zu wählen. Von den übrigen Adressenleitungen muß nur A10 decodiert werden (als niedrig), während die Leitungen A11 - A15 als hoch gelten. Somit kann jedes Gerät mit Registern ausgerüstet werden, die &F8??, &F9??, &FA?? und &FB?? als Adresse haben, wobei ?? den Bereich DC bis DF für Kommunikations-Schnittstellen und den Bereich EO bis FE für Benutzer-Peripheriegeräte angibt.

Hierbei ist zu beachten, daß Z80-Anweisungen benutzt werden müssen, die das B-Register auf die obere Hälfte des Adressenbus setzen.

Zusätzliche ROMs

Anstelle der vorhandenen ROMs können zusätzliche ROMs eingesetzt werden. Die Adressenzuordnung und Bankauswahl-Logik sind in einem Modul enthalten, das an den Erweiterungsbus angeschlossen ist. Alle erforderlichen Signale werden zum Erweiterungsbus übertragen.

Teil 15: CP/M Plus Bildschirm-Anpassung

Im ersten Teil dieses Kapitels wurden die Steuer-Zeichen und ihre Funktionen aufgeführt. Diese Funktionen werden ausgeübt, wenn von BASIC oder CP/M 2.2 Text zum Bildschirm geschickt wird. Sie sind einfach zu benutzen und spiegeln die Möglichkeiten des Firmware Text-Terminals wider. Diese Möglichkeiten bieten nur AMSTRAD-Computer, weshalb die Software entsprechend angepaßt werden muß.

In der Umgebung kommerziell genutzter CP/M Plus-Software wird normalerweise eine Reihe von Standard-Text-Terminal-Möglichkeiten erwartet, damit die Software auf unterschiedlichen Computern eingesetzt werden kann. Das CP/M Plus-Betriebssystem des CPC6128 besitzt eine Bildschirmanpassung, die ähnliche Voraussetzungen schafft wie ein Zenith Z19/Z29 Terminal. Das Installations-Programm für CP/M Plus-Software wird normalerweise standardmäßig eine Option für diese Art von Terminal enthalten.

Die Möglichkeiten der CP/M Plus Bildschirm-Anpassung sind zum Teil dieselben, die vorher durch das Firmware Text-Terminal geboten wurden. Allerdings werden hierfür andere Steuer-Codes verlangt.

Darüberhinaus bietet die CP/M Plus Bildschirm-Anpassung eine Reihe neuer und komplexerer Operationen.

Zeichen zwischen &20 und &FF werden an der gegenwärtigen Cursor-Position dargestellt. Befindet sich der Cursor nicht in der rechten Spalte, wird er um eine Spalte nach rechts gerückt. Wenn der Cursor in der rechten Spalte liegt und der Bildumlauf (wrapping) in Kraft ist, taucht er in der nächsten Zeile auf der linken Spalte auf, wobei das Bild nötigenfalls nach oben rollt.

Zeichen zwischen &00 und &1F haben folgende Steuer-Funktionen:

&07 BEL Erzeugt einen Piepston.

&08 BS (backspace) Bewegt den Cursor um eine Spalte nach links. Befindet er sich auf der linken Spalte und nicht in der obersten Zeile, springt er zur rechten Spalte in der darüberliegenden Zeile, vorausgesetzt, der Bildumlauf ist in Kraft.

&0A LF (linefeed) Bewegt den Cursor um eine Zeile nach unten, wobei das Bild nötigenfalls nach oben gerollt wird.

&0D CR (carriage return) Bewegt den Cursor zum linken Rand in derselben Zeile.

&1B ESC Führt eine Escape-Folge ein.

Folgende Escape-Folgen sind möglich. Werden andere als die aufgeführten Zeichen in einer Escape-Folge eingegeben, so werden sie dargestellt und der Cursor rückt eine Stelle vor. Auf diese Weise können die Zeichen dargestellt werden, die den Steuer-Codes &00 bis &1F entsprechen. (Beachten Sie, daß viele Sprachen-Anwendungen den Steuer-Code &09 **[TAB]** um Leerstellen erweitern, und deshalb die Folge **[ESC][TAB]** häufig nicht das Zeichen für &09 ausgibt.)

[ESC]0 Setzt die Status-Zeile außer Betrieb. Diskettensystem-Meldungen werden über den CRT-Bildschirm ausgegeben. Die unterste Zeile des Bildschirms kann von CRT genutzt werden.

[ESC]1 Setzt die Status-Zeile in Betrieb. Die unterste Bildschirmzeile ist für Diskettensystem-Meldungen reserviert.

[ESC]2‹n› Ändert den Zeichensatz (siehe Teil 16 in diesem Kapitel). ‹n› ist der Sprachen-Parameter, der mit &07 maskiert wird. Bestimmte Zeichen-Matrizen zwischen &20 und &7F werden mit anderen Zeichen zwischen &80 und &FF ausgetauscht. Dieser Befehl ist vergleichbar mit Steuer-Befehlen bei Druckern, die mit Software zur Auswahl internationaler Zeichensätze ausgerüstet sind.

‹n›=0 amerikanisch
‹n›=1 französisch
‹n›=2 deutsch
‹n›=3 englisch
‹n›=4 dänisch
‹n›=5 schwedisch
‹n›=6 italienisch
‹n›=7 spanisch

[ESC]3‹m› Ändert den Bildschirm-Modus. ‹m›=Bildschirm-Modus+&20. Der Wert wird mit &3 maskiert, um die Modi 0 bis 2 zu ergeben. Der Bildschirm wird gelöscht, die Cursor-Position bleibt erhalten.

[ESC]A Bewegt Cursor aufwärts. Liegt er in der obersten Zeile, passiert nichts.

[ESC]B Bewegt Cursor abwärts. Liegt er in der untersten Zeile, passiert nichts.

[ESC]C Bewegt Cursor vorwärts. Liegt er auf der Spalte ganz rechts, passiert nichts.

[ESC]D Bewegt Cursor rückwärts. Liegt er auf der Spalte ganz links, passiert nichts.

[ESC]E Löscht die Seite. Cursor-Position bleibt erhalten. Dieser Befehl löscht den gesamten Bildschirm, auch im 24x80-Modus. (Andere Escape-Codes gelten nur für den 24x80-Bereich, wenn der 24x80-Modus gesetzt ist.)

[ESC]H Schickt Cursor in die Ausgangsposition (oberste Zeile, Spalte ganz links).

[ESC]I Bewegt Cursor eine Zeile nach oben. Rollt den Bildschirm nötigenfalls nach unten.

[ESC]J Löscht alle Zeichen von der Cursor-Position einschließlich bis zum Ende des Bildschirms. Cursor bleibt an seiner Position.

[ESC]K Löscht alle Zeichen von der Cursor-Position einschließlich bis zum Ende der Zeile. Cursor bleibt an seiner Position.

[ESC]L Schiebt eine Leerzeile ein. Alle Zeilen ab der Zeile, in der sich der Cursor befindet, werden um eine Zeile nach unten verschoben. Der Cursor bleibt in seiner Zeile.

[ESC]M Löscht die Zeile, in der sich der Cursor befindet. Alle darunter liegenden Zeilen werden nach oben geschoben. Die unterste Zeile bleibt frei. Der Cursor verändert seine Position nicht.

[ESC]N Löscht das Zeichen, auf dem sich der Cursor befindet. Alle Zeichen rechts vom Cursor werden um eine Position nach links gerückt. Die Stelle am Ende der Zeile wird frei. Der Cursor verändert seine Position nicht.

[ESC]Y ‹Z›‹S› Rückt den Cursor an die angegebene Position. Liegt diese Position außerhalb des Bildschirmbereichs, rückt der Cursor zum Bildschirmrand. ‹Z›= Zeile+&20, ‹S›=Spalte+&20. Die linke obere Ecke des Bildschirms ist als Zeile 0, Spalte 0 definiert.

[ESC]b ‹FP› Definiert die Schriftfarbe für alle Zeichen auf dem Bildschirm. ‹FP› ist der Farb-Parameter. Er wird mit &3F maskiert und dann als eine von drei 2-Bit-Zahlen behandelt, die den Intensitätsgrad der drei Grundfarben darstellen. Bits 0 und 1 stehen für blau, Bits 2 und 3 für rot, Bits 4 und 5 für grün. Der CPC6128 bietet drei Farbstärkegrade, die auf den vier definierbaren Ebenen folgendermaßen dargestellt sind:

CPC6128	Keine Farbstärke	Halbe Farbstärke	Volle Farbstärke
Farb-Bits	00 binär	01 oder 10 binär	11 binär

[ESC]c ‹FP› Legt die Farbe des Hintergrunds fest. Gilt für den gesamten Hintergrund und den Rand. Die Farben sind wie oben angegeben definiert.

[ESC]d Löscht alle Zeichen von der Cursor-Position einschließlich bis zum Bildschirm-Anfang. Die Position des Cursors bleibt unverändert.

[ESC]e Macht den Cursor sichtbar. Um ein unschönes Blinken zu vermeiden, wird der Cursor nicht während der normalen Textausgabe eingeschaltet, sondern erst 1/10 Sekunde nachdem das letzte Zeichen geschrieben wurde.

[ESC]f Macht den Cursor unsichtbar.

[ESC]j Speichert die Cursor-Position.

[ESC]k Setzt den Cursor wieder an die Position, die mit [ESC]j gespeichert wurde.

[ESC]l Löscht die Zeile, in der sich der Cursor befindet. Die Position des Cursors bleibt unverändert.

[ESC]o Löscht alle Zeichen von der Cursor-Position einschließlich bis zum Zeilenbeginn. Position des Cursors bleibt unverändert.

[ESC]p Schaltet auf invertierte Darstellung um, d.h. Schrift- und Hintergrundfarbe werden vertauscht.

[ESC]q Macht die invertierte Darstellung rückgängig.

[ESC]r Schaltet Unterstreichungsmodus ein. (Wird vom CPC6128 nicht unterstützt.)

[ESC]u Schaltet Unterstreichungsmodus ab. (Wird vom CPC6128 nicht unterstützt.)

[ESC]v Schaltet Bildumlauf ein. Am Ende einer Zeile angelangt, wird automatisch in der nächsten Zeile weitergeschrieben.

[ESC]w Schaltet Bildumlauf ab. Alles, was über eine Zeile hinaus geschrieben wird, wird ignoriert.

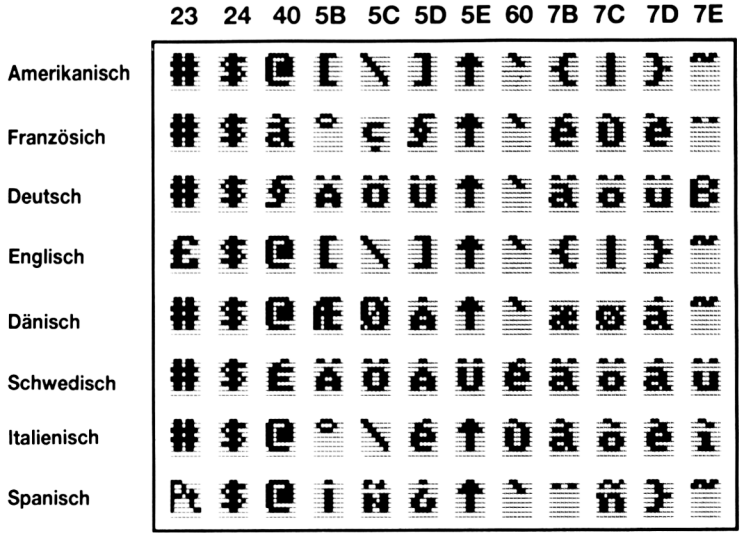
[ESC]x Schaltet auf 24x80-Modus um. Manche Anwender-Programme erfordern einen 'Standard'-Bildschirm von 24x80. Dieser Befehl stellt einen solchen Bildschirm her, gleichgültig, welche Ausmaße der Bildschirm tatsächlich hat. (Dies hängt ab vom Gerät, vom Land, und davon, ob die Status-Zeile an- oder abgeschaltet ist.)Der Bildschirm wird gelöscht.

[ESC]y Schaltet 24x80 Modus ab. Bildschirm wird gelöscht.

Teil 16: CP/M Plus Zeichensatz

Im 10. Teil dieses Kapitels wurde eine Übersetzungstabelle für den Drucker vorgestellt. Mit Hilfe dieser Tabelle können bestimmte Zeichen des BASIC- und CP/M 2.2- Zeichensatzes umgewandelt werden, damit der Drucker die Zeichen der Sprache ausgibt, die gewählt wurde. Diese Möglichkeiten sind etwas beschränkt, da nur sehr wenige Drucker fremdsprachige Zeichen haben, die auch im BASIC-Zeichensatz vertreten sind.

Obwohl solch eine Übersetzungstabelle auch unter CP/M Plus anwendbar ist, wurde der Zeichensatz erweitert, um eine fast völlige Übereinstimmung zwischen den Zeichen auf dem Bildschirm und den gedruckten Zeichen zu erreichen. (Das einzige fehlende Bildschirm-Zeichen ist das Symbol für die schwedische Währung anstelle des Dollarzeichens). Die Abbildung unten zeigt die Umwandlungstabelle für den internationalen Zeichensatz:



CP/M Plus Internationaler Zeichensatz

Um den Computer in einer Fremdsprache zu betreiben, sind zwei Voraussetzungen zu erfüllen:

1. Der Drucker selbst sollte auf die Fremdsprache eingestellt sein. (Dies geschieht häufig mit kleinen DIP-Schaltern, obwohl einige Drucker auch mit Steuer-Codes umgestellt werden können.)

2. Der gewünschte Bildschirm-Zeichensatz muß aufgerufen werden, entweder durch den transienten Befehl:

   ```
   LANGUAGE <n>
   ```

   ...oder indem der Bildschirm mit:

   [ESC]2<n>

   ...angepaßt wird.

In der Praxis wird die Umschaltung auf einen fremdsprachigen Zeichensatz als Teil des `PROFILE.SUB`-Programms mit den Dienstprogrammen `LANGUAGE` und `SETLST` vorgenommen. CP/M Plus ist ursprünglich auf den amerikanischen Zeichensatz eingestellt.

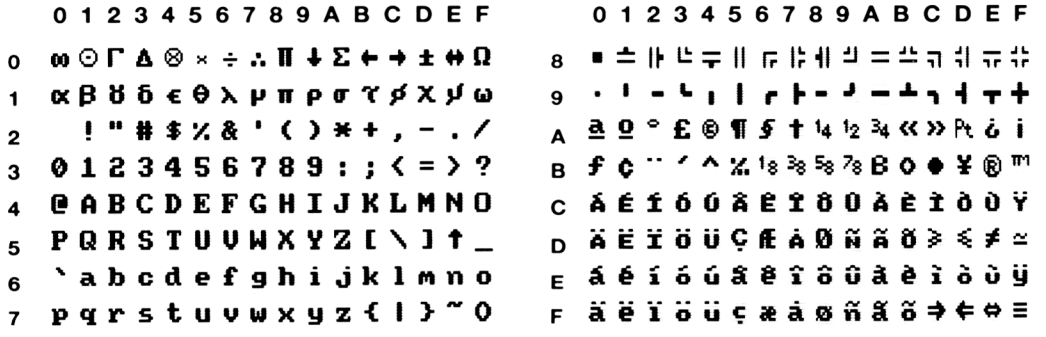
7 Bit Software

Die Umstellung auf Zeichensätze anderer Sprachen hat zwar große Vorteile, es muß aber darauf hingewiesen werden, daß nach solch einer Umstellung einige Zeichen der amerikanischen Tastatur nicht mehr dargestellt werden können. Dies ist ein Problem, das sich aus der 7Bit-Software ergibt: Fast alle Software, einschließlich der meisten CP/M Plus Dienstprogramme, Textverarbeitungsprogramme und Computer-Sprachprogramme, basiert auf einem 7Bit-Zeichensatz.

Der englische Zeichensatz unterscheidet sich nur durch ein Zeichen vom amerikanischen. In anderen Zeichensätzen werden hingegen mehr Zeichen, z.B. das Balken-Symbol oder die eckigen Klammern, durch Akzente, Umlaute usw. ersetzt. Dadurch wird zwar die Lesbarkeit von Texten verbessert. In Situationen jedoch, wo ein Anwenderprogramm den Balken oder eckige Klammern verlangt (z.B. `DIR [ALL]`), wird die Lesbarkeit und - falls die Tastenknöpfe ausgetauscht wurden - die Schreibbarkeit erheblich beeinträchtigt. Denken Sie daran, daß das Anwenderprogramm mit ASCII-Werten arbeitet, ganz gleich, in welche Sprache ein Zeichen auf dem Bildschirm übersetzt wurde. Das Problem ist, daß mit 7Bit-Software einfach nicht genügend ASCII-Werte zur Verfügung stehen.

8 Bit-Zeichensätze

Der BASIC-Zeichensatz besteht aus 256 Zeichen mit den Werten 128 bis 255 (&80 bis &FF). Zum Teil handelt es sich um graphische Symbole, die für Computerspiele benutzt werden (tanzende Männchen, Herz-, Pik-, Kreuz-, Karo-Symbole usw.) CP/M Plus hat ebenfalls 256 Zeichen, wobei sich die letzten 128 vom BASIC-Zeichensatz unterscheiden: sie machen CP/M Plus für die internationale Geschäftswelt gebrauchsfähig.



Zeichen 0 bis 127 (&00 bis 7F) **Zeichen 128 bis 255 (&80 bis &FF)**

Der Standard- CP/M Plus- Zeichensatz (amerikanisch)

Software, die mit 8 Bit-Zeichen betrieben werden kann, ist mit diesem Zeichensatz in der Lage, auf alle fremsprachigen Zeichen direkt zuzugreifen, ohne die Sprache wechseln zu müssen. Auf diese Weise bleiben Zeichen wie Balken und eckige Klammern erhalten.

Bitte beachten Sie, daß das 8 Bit-Softwareangebot z.Z. noch sehr begrenzt ist, und daß Zeichen zwischen 0 und 31 sowie 128 und 255 auf dem Bildschirm ausschließlich in der oben dargestellten Form erscheinen. Ein Drucker wird dagegen andere Vorstellungen vom Aussehen dieser Zeichen haben.

Kapitel 8:
Näheres über den Bank Manager

BASIC-Erweiterungen für den Zugriff auf die zweiten 64K im RAM-Speicher

Behandelte Themen:

- ★ Speichern von Bildschirminhalten
- ★ Pseudo-Datei-Operation

Aus dem Speicherbelegungsplan für BASIC 1.1 (Kapitel 7, Teil 14) ist zu entnehmen, daß 64K des 128K umfassenden RAMs ungenutzt sind. BASIC und die Firmware sind im ROM-Speicher untergebracht, der zusammen mit dem Disketten-ROM die normale Speicherkapazität von 64K auf 112K erweitert (64K RAM + 48K ROM).

Jeweils 16K im Speicher bilden einen Block, und eine Einheit aus vier Blöcken (=64K) nennt man Bank. Den Vorgang zur Blockauswahl nennt man 'bank switching'.

Der Z80 Mikroprozessor kann sich immer nur mit 64K auf einmal beschäftigen. Deshalb enthält das Betriebssystem Anweisungen, mit denen das Firmware-ROM anstelle von Block 0 des RAM hinzugeschaltet werden kann. Ebenso können das BASIC ROM oder das Disketten-ROM anstelle von Block 3 eingeschaltet werden. Dieses Umschalten findet automatisch statt, wenn BASIC oder Firmware gefordert wird. Mit der zweiten RAM-Bank wird dieses Prinzip ausgeweitet, indem statt zwischen RAM und ROM zwischen RAM und RAM hin- und hergeschaltet wird. Das Umschalten wird von einem Assembler-Programm gesteuert.

Auf der Seite 1 der System-Disketten finden Sie das Programm `BANKMAN.BAS`. Wird das Programm unter BASIC gestartet, setzt es den RSX-Code der Standard-Bank-Verwaltung ein. Deshalb wird das Programm 'BANK MANAGER' genannt.

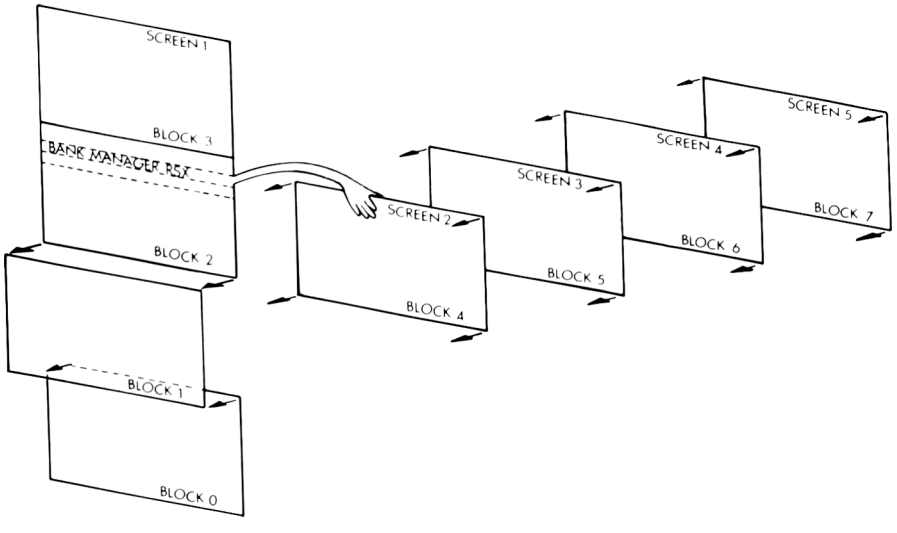
Die zweiten 64K können benutzt werden, um Bildschirminhalte zu speichern. Unter solche Anwendungen fällt z.B. ein 'Bildschirmentwurfs-Programm', das verschiedene Bildschirmdarstellungen speichert, oder ein Video-Spiel mit verschiedenen vorbereiteten Bildern.

Die zusätzlichen 64K können außerdem als Erweiterungen des variablen Speichers benutzt werden, um beispielsweise den Text-Array-Platz zu erweitern, oder einfach als 'RAM-Diskette'.

Teil 1:
Speichern von Bildschirminhalten

Wählen Sie Ihren Bildschirm!

Der BANK MANAGER kann Block 1 ausschalten und dafür einen der vier Blöcke aus den zusätzlichen 64K einschalten. Die Abbildung unten illustriert diesen Vorgang. Beachten Sie, daß jeder Block aus den zweiten 64K an derselben Adresse abgelegt wird (&4000 bis &7FFF). Der Inhalt von Block 1 (wahrscheinlich die Mitte Ihres BASIC-Programms) bleibt erhalten und kommt wieder an seinen alten Platz, wenn das BANK MANAGER-Programm beendet ist. Es sind noch drei andere Bank-Auswahlmöglichkeiten vorhanden (außer den unten gezeigten fünf), aber die sind nur für den Einsatz von CP/M Plus von Nutzen.



Umschalten zwischen verschiedenen Blöcken

Der BANK MANAGER unterstützt zwei Befehle, mit denen zwischen verschiedenen Bildschirm-Inhalten hin- und hergeschaltet werden kann. Blöcke 4 bis 7 nehmen dabei nach Bedarf den Platz von Block 1 auf dem Speicherbelegungsplan ein.

Der Befehl:

```
!SCREENSWAP,[‹Bildschirm-Abschnitt›,]‹Bildschirmnummer›,
                                    ‹Bildschirmnummer›
```

...tauscht den Inhalt zweier Blöcke aus, während:

```
!SCREENCOPY,[‹Bildschirm-Abschnitt›,]‹Ziel-Bildschirmnummer›,
                                    ‹Quellen-Bildschirmnummer›
```

...den Inhalt eines Blockes in einen anderen kopiert.

Der Wahl-Parameter ‹Bildschirm-Abschnitt› gibt an, wieviele Teile eines Blockes, in 64stel (1/64=256 Byte von 64K) gemessen, kopiert werden sollen. Für ‹Bildschirm-Abschnitt› können daher Werte von 0 bis 63 eingesetzt werden. Dieser Operations-Modus ist nützlich, wenn neben dem Bildschirmtausch ein anderer Verarbeitungsprozeß ablaufen soll. Ein Bildschirmwechsel dauert ungefähr 150/300 Sekunden (150 TIME-Einheiten).

Die Angabe 1 für ‹Bildschirm-Nummer› bezeichnet den normalen, sichtbaren Bildschirm, die Nummern 2, 3, 4 oder 5 stehen für die anderen Bildschirmdarstellungen. `SCREENCOPY` und `SCREENSWAP`-Operationen laufen viel schneller ab, wenn der Quellen- oder Ziel-Bildschirm 1 ist. Achten Sie darauf, daß der Bildschirm nicht nach oben oder unten rollt, was beim Operieren mit Disketten-gespeicherten Bildschirm-Kopien vorkommt. Alle Bildschirmabzüge sollten so beschaffen sein, daß Bildschirm 1 immer in derselben Hardware-Position zu sehen ist. Am einfachsten wird die Standard-Position durch einen `MODE`-Befehl festgelegt.

Experimentieren mit den Bildschirmtausch-Befehlen

Lassen Sie zunächst das BANK MANAGER-Programm von Seite 1 ihrer System-Disketten laufen, indem Sie eintippen:

```
RUN "BANKMAN"
```

Schreiben Sie nun:

```
MODE 1
```

Der Bildschirm wird gelöscht. Tippen Sie:

```
' Dies ist das Original-Bild
!SCREENCOPY,3,1 'Schicke das Originalbild zu Speicherbl
ock 3

CLS
```

Wieder wird der Bildschirm gelöscht. Tippen Sie:

```
' Dies ist das Ersatz-Bild
!SCREENCOPY,2,1 ' Schicke das Ersatz-Bild nach Speicher
block 2
!SCREENSWAP,2,3 ' Tausche Speicherblock 2 und Speicherb
lock 3 aus
!SCREENCOPY,1,3 ' Hole das Ersatz-Bild von Speicherbloc
k 3 auf den Bildschirm
!SCREENCOPY,1,2 ' Hole das Original-Bild von Speicherbl
ock 2 auf den Bildschirm
```

Als weiteren Beitrag zu diesem Thema werden im letzten Teil von Kapitel 9 die Bildschirm-Austausch-Möglichkeiten des BANK MANAGERs anhand eines umfassenden 'Bildschirmentwurfs-Programms' vorgeführt.

Teil 2: Pseudo-Datei-Operation

Auf die lange Bank geschoben...

Wenn man die gesamten zweiten 64K RAM als RAM-Diskette betrachtet, so bilden sie eine RAM-Datei mit einer Anzahl von gleich langen Datensätzen. Die Datensatzlänge kann zwischen 0 und 255 Byte liegen, wobei eine Minimal-Länge von 2 Byte empfohlen wird. Ist die ‹RAM-Satzlänge› einmal festgelegt, kann jeder Satz durch seine ‹RAM-Satznummer› aufgerufen werden. Beim Schreiben in eine RAM-Datei kann ohne weiteres eine andere Satzlänge angegeben werden, als beim Lesen.

ACHTUNG! Die RAM-Datei darf nur Daten enthalten; sie ist nicht zum Speichern von Programm-Anweisungen geeignet.

Wie bei RAM-Disketten-Speicher-Systemen üblich, gilt auch hier das Prinzip der 'fortlaufenden Satznummern'. Damit wird automatisch immer auf die nächste Satznummer zugegriffen, was besonders nützlich ist, wenn nacheinander mit allen Sätzen in der Datei eine Operation durchgeführt werden soll.

Der Befehl:

```
!BANKOPEN,‹RAM-Satzlänge›
```

...setzt alle Datensätze auf eine Länge fest, macht Satz Nummer 0 zum aktuellen Satz, löscht jedoch NICHTS aus dem Speicher.

Der Befehl:

```
!BANKWRITE,@‹Rückkehr-Code›,‹Textausdruck›[,‹RAM-Satznummer›]
```

...schreibt den ‹Textausdruck› in die RAM-Datei.

Die ‹RAM-Satznummer› gibt an, welcher Satz geschrieben wird. Wird dieser Parameter ausgelassen, wird automatisch die aktuelle Satznummer angenommen. Danach wird der Satz mit der nächsten Nummer zum aktuellen Satz.

Füllt der ‹Textausdruck› den Satz nicht ganz aus, so bleiben die Zeichen aus dem alten Satz am Ende bestehen, da sie nicht überschrieben werden. Ist der ‹Textausdruck› länger als der Satz, so werden die überschüssigen Zeichen ignoriert, damit sie keinen Platz im nächsten Satz besetzen.

‹Rückkehr-Code› ist eine ganzzahlige Variable, die die Nummer des Satzes wiedergibt, in den geschrieben wurde (falls die Operation erfolgreich verlief). Hat die Schreib-Operation aus irgendeinem Grund nicht geklappt, wird ein negativer Fehler-Code ausgegeben.

-1 'End of file' -Fehler. Die Adresse der geforderten Satznummer liegt außerhalb der 64K.

-2 Umschaltfehler bei der Bankauswahl. (Dürfte eigentlich nicht vorkommen.)

Beispiele:

```
!BANKOPEN,10
!BANKWRITE,@r%,"123 probe",0
!BANKWRITE,@r%,w$
```

Der Befehl:

```
!BANDREAD,@‹Rückkehr-Code›,@‹Stringvariable›[,‹RAM-Satznummer›]
```

...liest einen Satz von der RAM-Datei in die ‹Stringvariable›.

Die ‹RAM-Satznummer› gibt an, welcher Satz gelesen werden soll. Wird der Parameter ausgelassen, so wird automatisch die aktuelle Satznummer angenommen. Danach ist der Satz mit der nächsten Nummer aktuell.

Wenn der Satz-Inhalt die ‹Stringvariable› nicht ganz ausfüllt, bleiben alte, nicht überschriebene Zeichen am Ende der ‹Stringvariable› bestehen. Enthält der Satz mehr Zeichen, als die ‹Stringvariable› aufnehmen kann, werden die überschüssigen Zeichen nicht berücksichtigt, da es nicht möglich ist, die Länge einer Stringvariablen während eines externen Befehls auszudehnen.

‹Rückkehr-Code› ist eine ganzzahlige Variable, die die Nummer des Satzes, der gelesen wurde, wiedergibt (falls die Operation erfolgreich verlief). Hat die Lese-Operation aus irgendeinem Grund nicht geklappt, wird ein negativer Fehler-Code ausgegeben:

-1 'End of file'-Fehler. Die Adresse der geforderten Satznummer liegt außerhalb der 64K.

-2 Umschaltfehler bei der Bankauswahl. (Dürfte eigentlich nicht vorkommen.)

Beispiel:

```
!BANKREAD,@r%,i$,Ø
```

Suchaktion

Mit folgenden Befehl werden alle gespeicherten Datensätze nach einem bestimmten Eintrag durchsucht:

!BANKFIND,@‹Rückkehr-Code›,‹gesuchter String›
[,‹erste Satznummer›[,‹letzte Satznummer›]]

Die ‹erste Satznummer› gibt an, bei welchem Satz die Suche beginnen soll. Wird dieser Parameter ausgelassen, wird ab dem aktuellen Satz gesucht.

Der gesamte 64K-Speicher wird in ‹RAM-Satzlänge›-Schritten durchgekämmt, bis der ‹gesuchte String› gefunden ist.

Ist eine ‹letzte Satznummer› angegeben, wird die Suche abgebrochen, nachdem dieser Satz geprüft wurde (falls der gesuchte String nicht schon vorher gefunden wurde).

War die Suche erfolgreich, wird der Satz, in dem der String gefunden wurde, zum aktuellen Satz.

‹Rückkehr-Code› ist eine ganzzahlige Variable, die die Satznummer angibt, in der der gesuchte String gefunden wurde (falls die Operation erfolgreich verlief). Wurde der String aus irgendeinem Grunde nicht gefunden, wird ein negativer Code ausgegeben:

-1 'End of file'-Fehler. Die ‹erste Satznummer› liegt außerhalb der 64K oder ist größer als die ‹letzte Satznummer›.

-2 Umschaltfehler bei der Bankauswahl. (Dürfte eigentlich nicht vorkommen.)

-3 Keinen entsprechenden String gefunden

Der ‹gesuchte String› darf Universalzeichen in Form von Nullen enthalten (`CHR$(Ø)`). Ist der ‹gesuchte String› länger als die ‹RAM-Satzlänge›, werden zur Vergleichsoperation nur soviele Zeichen des Strings benutzt, wie die ‹RAM-Satzlänge› angibt.

Beispiele:

```
|BANKFIND,@r%,"123 test",Ø
|BANKFIND,@r%,f$,1ØØ,2ØØ
```

Offensichtliche Fehler, wie z.B. die falsche Anzahl von Parametern, werden mit der Meldung '`Bad Command`' quittiert. Jedoch werden bei externen Befehlen keine Fehler vom Typ '`Type mismatch`' entdeckt, und der Benutzer muß darauf achten, die richtige Form von Parametern zu verwenden.

Im unten angegebenen Beispiel-Programm werden RAMdisc-Befehle verwendet, um eine Datenbank einzurichten und abzufragen, die Anagramme von 7-Buchstaben-Wörtern enthält. Es sucht nach gleichen Wortteilen, wobei auch Universalzeichen eingesetzt werden können.

Bildet man z.B. Anagramme aus dem Wort `FIGUREN`, erhält man sechs Möglichkeiten, die in das Schema `?RUNG??` (die letzten beiden `??` können auch weggelassen werden) passen: `FRUNGIE`, `FRUNGEI`, `ERUNGIF`, `ERUNGFI`, `IRUNGEF` und `IRUNGFE`.

Es dauert einige Zeit, bis diese Datenbank errichtet ist, aber schließlich gilt es, 64K Speicherplatz zu füllen!

```
1Ø 'ANAGRAMS by ROLAND PERRY
2Ø ' copyright (c) AMSOFT 1985
3Ø '
4Ø 'Vor diesem Programm das "BANKMAN" Programm laufen lassen
5Ø '******************************************************
6Ø '
7Ø MODE 2
8Ø DEFINT a-z
9Ø r%=Ø:|BANKOPEN,7
1ØØ INPUT"Ein Wort mit 7 Buchstaben zum Durcheinanderschuet
    teln
11Ø IF LEN(s$)<>7 THEN 1ØØ
```

```
120 PRINT"Bitte warten..."
130 LOCATE 1,5:PRINT"Anagramm:"
140 FOR c1=1 TO 7
150 FOR c2=1 TO 7
160 IF c2=c1 THEN 370
170 FOR c3=1 TO 7
180 IF c3=c2 OR c3=c1 THEN 360
190 FOR c4=1 TO 7
200 IF c4=c3 OR c4=c2 OR c4=c1 THEN 350
210 FOR c5=1 TO 7
220 IF c5=c4 OR c5=c3 OR c5=c2 OR c5=c1 THEN 340
230 FOR c6=1 TO 7
240 IF c6=c5 OR c6=c4 OR c6=c3 OR c6=c2 OR c6=c1 THEN 3
    30
250 FOR c7=1 TO 7
260 IF c7=c6 OR c7=c5 OR c7=c4 OR c7=c3 OR c7=c2 OR c7=
    c1 THEN 320
270 o$=MID$(s$,c1,1)+MID$(s$,c2,1)+MID$(s$,c3,1)+MID$(s
    $,c4,1)+MID$(s$,c5,1)+MID$(s$,c6,1)+MID$(s$,c7,1)
280 LOCATE 12,5:PRINT x;o$
290 |BANKWRITE,@r%,o$
300 IF r%<0 THEN STOP
310 x=x+1
320 NEXT c7
330 NEXT c6
340 NEXT c5
350 NEXT c4
360 NEXT c3
370 NEXT c2
380 NEXT c1
390 letztersatz=r%
400 REM Jetzt werden alle durchgegangen
410 r%=0:g$=SPACE$(7)
420 PRINT:INPUT"Welches Buchstabenschema soll das Anag
    ramm enthalten? Benutze Universalzeichen?:
430 m$=LEFT$(m$,7)
440 FOR x=1 TO LEN(m$)
450 IF MID$(m$,x,1)="?"THEN MID$(m$,x,1)=CHR$(0)
460 NEXT
470 |BANKFIND,@r%,m$,0,letztersatz
480 IF r%<0 THEN GOTO 420
490 |BANKREAD,@r%,g$
500 PRINT g$,
510 |BANKFIND,@r%,m$,r%+1,letztersatz
520 GOTO 480
```

Kapitel 9:
Wenn Sie mal Zeit haben

Dieses Kapitel befaßt sich ganz zwanglos mit einigen Hintergrund-Informationen über Computer im allgemeinen und den CPC6128 im besonderen. Es ist nicht unbedingt erforderlich, daß Sie dieses Kapitel lesen, bevor Sie mit dem Computer experimentieren, es hilft Ihnen jedoch, sein Innenleben zu entdecken.

Teil 1: Computer im allgemeinen...

Selbst wenn Sie Ihren CPC6128 nur erworben haben, um sich mit anspruchsvollen Computerspielen zu beschäftigen, möchten Sie vielleicht einiges über den Computer erfahren, was mit dem Namen 'Hardware' bezeichnet wird.

Unter 'Hardware' versteht man alles am Computer, was man anfassen und wegtragen kann, z.B. die Haupttastatur des Computers, den Monitor, die Verbindungskabel usw. Darunter fällt im Grunde fast alles, was nicht als 'Software' bezeichnet wird. 'Software' sind Programme, Handbücher und der Inhalt von Disketten und Kassetten.

Bestimmte Fähigkeiten von Computern werden durch die Hardware ermöglicht, z.B. Farbdarstellungen auf dem Monitor. Es ist Aufgabe der Software, diese Möglichkeiten der Hardware auszunutzen, um besondere Zeichen oder Formen auf dem Bildschirm entstehen zu lassen.

Die Hardware leitet den Elektronenstrahl auf die Leuchtfläche innerhalb des Bildschirms, um ein Bild aufleuchten zu lassen. Die Software sagt der Hardware mit Befehlen und Informationen, wann sie was darstellen soll. Sie sorgt für richtigen Zeitablauf, Steuerung und Reihenfolge, um ein startendes Raumschiff, oder etwas Naheliegenderes wie einen eingetippten Buchstaben auf dem Bildschirm erscheinen zu lassen.

Worin unterscheiden sich Computer?

Hardware ohne Software ist zwecklos. Software ohne Hardware ist genauso zwecklos. Der Computer hat erst dann einen Wert, wenn beide zusammenwirken, um verschiedene Aufgaben zu erfüllen. Es gibt einige einfache Maßstäbe, mit denen man die Leistung von Hardware und Software messen kann.

Für die Qualitätsbewertung von Personal-Computern gelten heute allgemein folgende Kriterien:

1. DIE AUFLÖSUNG - Die Dichte der kleinsten erkennbaren Einheiten auf dem Bildschirm

Hierbei handelt es sich um eine Kombination von Faktoren: Die Anzahl der Farben, die dem Programmierer zur Verfügung stehen, die Anzahl der Bildpunkte (Pixel) auf dem Bildschirm, und die Anzahl der Textzeichen, die in einem Bildschirmbereich dargestellt werden können.

Sie werden feststellen, daß der CPC6128 im Vergleich mit Systemen derselben Preisklasse in dieser Hinsicht sehr gut abschneidet.

2. Der BASIC-INTERPRETER

Fast alle Heimcomputer sind mit einem BASIC-Interpreter ausgestattet, der es dem Benutzer erlaubt, Programme zu schreiben, um die Hardware-Möglichkeiten zu nutzen. Die eingebaute Programm-Sprache BASIC, die mit Ihrem Computer geliefert wurde, ist selbst ein Programm - ein enorm kompliziertes und verwickeltes Programm, das sich durch die Erfahrung von über einer Million Mann-Jahren entwickelt hat, seit BASIC in den USA erfunden wurde. BASIC (Abkürzung für: Beginners All-purpose Symbolic Instruction Code) ist sicher die in der Welt am weitesten verbreitete Computersprache, und wie jede Sprache weist sie eine Vielzahl lokaler 'Dialekte' auf.

Die mit dem CPC6128 gelieferte BASIC-Version ist einer der kompatibelsten BASIC-Dialekte. Mit ihr können auch Programme gesteuert werden, die für den Betrieb unter Einsatz des CP/M Disketten-Betriebssystems geschrieben wurden. Diese BASIC-Version ist sehr schnell, d.h. sie führt ihre Rechnungen sehr schnell aus. Es ist Ihnen vielleicht egal, ob ein Computer 0,05 Sekunden oder 0,075 Sekunden benötigt, um 3 mal 5 auszurechnen und anzuzeigen, wenn aber ein Programm eine Graphik erstellen soll und dafür tausende von einfachen wiederkehrenden Rechnungen durchführen muß, summiert sich der Unterschied zwischen 0,05 und 0,075 Sekunden zu einer beträchtlichen Leistungsdifferenz.

Häufig werden Sie dem Begriff 'Maschinencode' begegnen. Maschinencode ist die Form des Befehlscodes, die an den Prozessor weitergegeben wird. Der Prozessor braucht weniger Zeit, um herauszufinden, was er tun soll, und kann deshalb eine Anweisung in Maschinencode 5 bis 15 Mal schneller ausführen, als eine Operation, die über den BASIC-Interpreter läuft. Andererseits kann es 5 bis 50 Mal mehr Zeit in Anspruch nehmen, ein Programm in Maschinencode zu schreiben, als das Schreiben desselben Programms in BASIC samt Ausführung dauern würde.

Die BASIC-Version in Ihrem AMSTRAD-Computer gehört zu den schnellsten und leistungsfähigsten, die in Heimcomputern zu finden sind. Sie besitzt eine Vielzahl von Eigenschaften, die es dem erfahrenen BASIC-Programmierer erleichtern, einige der systembedingten Nachteile eines hochentwickelten Sprach-Interpreters zu überwinden, um erstaunlich dynamische visuelle und musikalische Effekte zu erzeugen.

3. ERWEITERUNGSFÄHIGKEIT

Bei den meisten Computern wird auf erforderliche zusätzliche Hardware-Geräte verwiesen, wie Drucker, Joysticks, zusätzliche Diskettenlaufwerke. Paradoxerweise braucht man bei einigen der erfolgreichsten Heimcomputer zusätzlich noch sogenannte 'Erweiterungsschnittstellen', bevor auch nur ein einfacher Drucker oder Joystick angeschlossen werden kann.

Nicht jeder Käufer denkt an seine zukünftigen Bedürfnisse; im Endeffekt kann es daher günstiger sein, ein Gerät zu kaufen, das Anschlüsse für einen geeigneten (Centronics-kompatiblen) Drucker und einen Joystick besitzt.

Der CPC6128 besitzt einen eingebauten Anschluß für einen Centronics-Drucker, einen Anschluß für ein zusätzliches Diskettenlaufwerk, eine Anschlußbuchse für Kassettenbetrieb, Anschlußmöglichkeiten für bis zu zwei Joysticks, einen Stereo-Tonausgang sowie einen umfassenden Erweiterungs-Bus zum Anschluß einer seriellen Schnittstelle (RS232), eines MODEMs, eines Sprachsynthesizers, eines Lichtgriffels usw.

4. TONQUALITÄT

Die akustischen Qualitäten eines Computers entscheiden, ob sich die erzeugten Töne wie ein Brummer in einer Blechdose anhören, oder ob sie wie aus einem elektronischen Musikinstrument klingen.

Der CPC6128 arbeitet mit einem Tongenerator mit drei Kanälen, der Tonumfang beträgt acht Oktaven. Über die Steuerung von Amplitude und Hüllkurven wird so eine äußerst ansprechende Musikqualität erreicht. Darüberhinaus besitzt der 6128 auch Stereo-Eigenschaften, indem ein Kanal der linken und ein Kanal der rechten Seite zugeordnet wird, während der dritte Kanal in der Mitte bleibt.

Mit diesen Möglichkeiten lassen sich Töne und Geräusche als Kulisseneffekte in Programme einbauen, die simultan zur Bewegung auf dem Bildschirm verlaufen.

Letztendlich müssen Sie als Benutzer selbst entscheiden, welche Eigenschaften eines Computers Ihnen am wichtigsten sind. Wir hoffen, daß Sie alles ausprobieren, um die Möglichkeiten Ihres Computers optimal zu nutzen.

Warum geht das nicht?

Von Computer-Neulingen wird in Anbetracht der Leistungsfähigkeit der modernen Technologie oft die Frage gestellt, warum selbst ein so hochentwickeltes Gerät wie der 6128 offenbar nicht in der Lage ist, die Art von Bildern auf den Bildschirm zu bringen, die jedes Fernsehgerät produziert. Es wird z.B. gefragt, weshalb der Computer keine Figur mit natürlichen Bewegungen über den Bildschirm laufen lassen kann, und warum alle Figuren wie Strichmännchen mit 'Charly Chaplin-Bewegungen' aussehen.

Die Anwort ist einfach und schwierig zugleich. Die einfache Antwort ist, daß Sie sich nicht dazu verleiten lassen dürfen zu glauben, daß der Bildschirm des Computers ein ebenso kompliziertes Gebilde ist wie ein Fernsehschirm. Ein Fernseher arbeitet mit 'linearen' Informationen, die nahezu unendlich viele Abstufungen zwischen hell und dunkel durch das gesamte Farbspektrum ermöglichen. Der Bildschirmspeicher für ein Fernsehbild müßte, so gesehen, 20 mal größer sein als der Speicher für ein Computer-Bild.

Dies ist nur eine Seite des Problems. Um dieses Bild in Bewegung zu setzen, müßte dieser enorme Speicher-Inhalt in hohem Tempo verarbeitet werden (ca. 50 mal pro Sekunde). Dies ist möglich - aber nur mit Maschinen, die einige tausendmal teurer sind als ein Heimcomputer - wenigstens vorläufig.

Bis die Kosten für Hochgeschwindigkeitsspeicher drastisch fallen (und das werden sie irgendwann), müssen Kleincomputer mit verhältnismäßig wenig Speicher auskommen, um den Bildschirm zu steuern. Dies bedeutet aber niedrigere Auflösung und zackige Bewegungen. Überlegte Hardware-Planung und gute Programmierung können viel dazu beitragen, das beste aus dieser Situation zu machen, aber wir sind noch weit davon entfernt, auf billigen Computern fließende Bewegungen und realistische Bilder darzustellen, die wenigstens Zeichentrickfilm-Qualität erreichen.

Die Tastatur kommt mir bekannt vor...

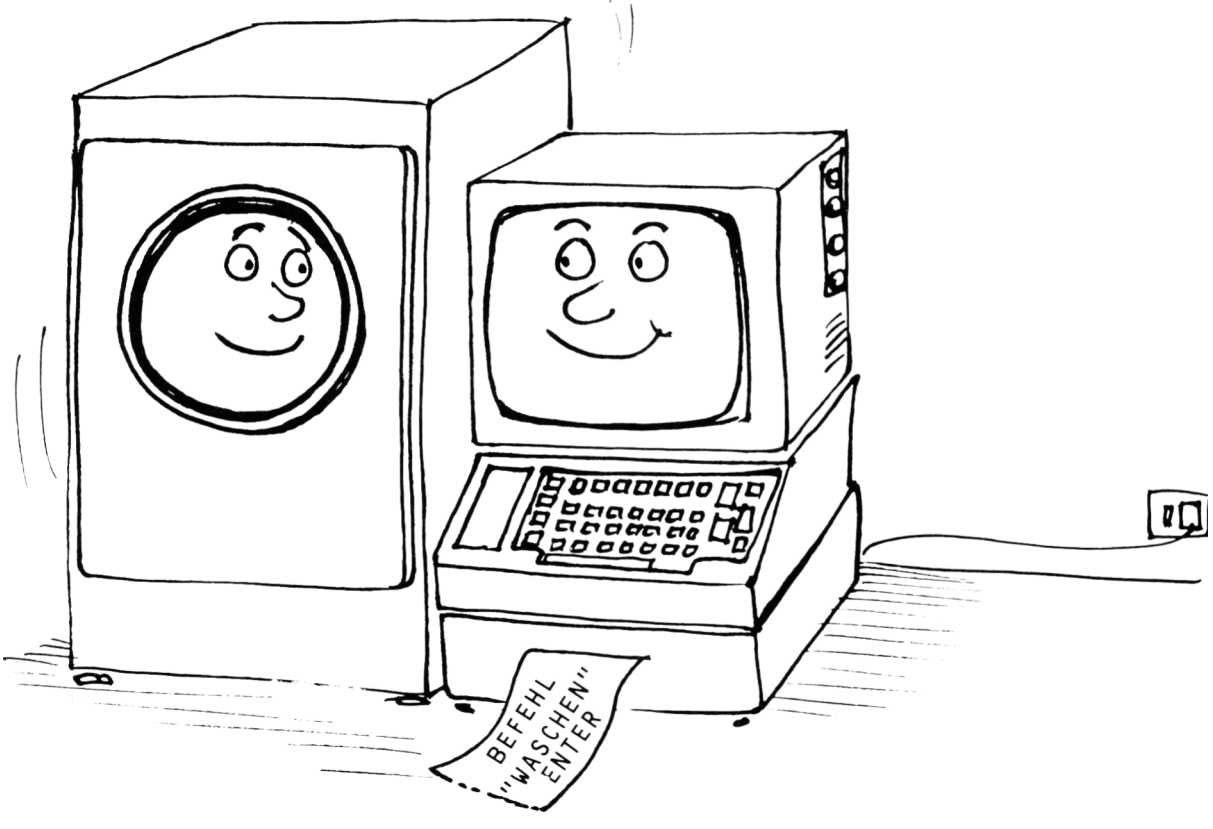
Warum kann ich nicht einfach zum Computer gehen und eine Seite Text in die Maschine tippen?

Lassen Sie sich nicht durch die Tatsache irreführen, daß der Computer wie eine Schreibmaschine mit elektronischer Anzeige aussieht. Der Bildschirm ist kein 'elektronisches Blatt Papier', sondern eine 'Befehlskonsole', d.h. er gibt Ihnen nur die Möglichkeit, mit der Programmiersprache und den Programmen im Computerspeicher Kontakt aufzunehmen.

Solange Sie ihm nichts anderes sagen, versucht der Computer, sämtliche Zeichen, die Sie eintippen, als Programm-Anweisungen zu interpretieren. Wenn Sie auf **[RETURN]** drücken, sieht sich der Computer an, was Sie eingetippt haben, und wenn es für sein eingebautes BASIC keinen Sinn ergibt, weist er die 'Eingabe' mit dem Kommentar `Syntax error` zurück.

Wenn Sie aber gerade ein Textverarbeitungs-Programm im Speicher haben, dürfen Sie beliebig Wörter eintippen, auf **[RETURN]** drücken und weitermachen, als ob Sie tatsächlich mit einer elektronischen Schreibmaschine auf ein elektronisches Blatt Papier schreiben würden. Doch dazu muß erst ein Textverarbeitungs-Programm in den Computerspeicher geladen werden.

Der Computer scheint mehrere Gerätekomponenten zu vereinen, die zu Hause und im Büro bekannt sind, z.B. den Fernseher-ähnlichen Bildschirm und die Tastatur. Sie müssen sich jedoch bewußt machen, daß diese Ähnlichkeiten nur oberflächlicher Natur sind, und daß der Computer zwar eine Kombination vertraut aussehender Hardware darstellt, jedoch eine ganz eigene Persönlichkeit hat!



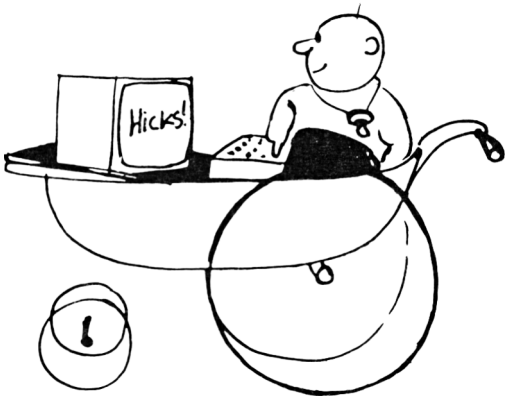
Wer hat Angst vor der Fachsprache?

Wie in allen Spezialbereichen hat sich auch in der Datenverarbeitung eine eigene Fachsprache entwickelt, eine Art Kurzschrift für die Übermittlung komplizierter Begriffe, die sonst viele 'Klartext'-Worte in Anspruch nehmen würden. Es sind nicht nur die hochtechnisierten Arbeitsgebiete, die sich hinter einer Mauer von Schlagwörtern und fachsprachlichem Kauderwelsch verbergen. Die meisten Berufe und Arbeitsgebiete haben ihre eigene Sprache, mit der wohl jeder schon zu kämpfen hatte.

Der größte Unterschied zwischen der Sprache der Datenverarbeitung und beispielsweise der Juristensprache besteht darin, daß in der Juristensprache die Satzkonstruktionen kompliziert sind, während in der Datenverarbeitung die Wörter an sich das Haupthindernis zum Verständnis darstellen. Die Leute, die sich in der Terminologie der Datenverarbeitung auskennen, bemühen sich, diese Begriffe in einen klaren Zusammenhang zu stellen, um Mißverständnisse in der Kommunikation so weit wie möglich auszuschalten. Lassen Sie sich durch die 'einfache Sprache' der Datenverarbeitung nicht täuschen: Sie hat nichts mit Literatur zu tun, sondern mit präziser Wissenschaft. Die Begriffe sind schwer zu verstehen, die Kommunikationsstruktur jedoch ist sehr einfach und nicht im mindesten verwirrend oder widersprüchlich.

Im Informatik-Unterricht werden Programme noch nicht nach künstlerischen Maßstäben bewertet.Trotzdem sticht sofort ins Auge, ob ein Programm elegant oder unübersichtlich aufgebaut ist, auch wenn der Zweck des Programms nicht auf Anhieb klar zu erkennen ist. Heute wird wesentlich mehr Wert auf ordentlichen Programmaufbau gelegt, als es im allgemeinen Wirbel am Anfang der Micro-Revolution üblich war.

Viele junge Leute lernen schnell, mit dem Computer umzugehen, da sie für die Genauigkeit und Einfachheit der Ideen und für die Art der Kommunikation ein Gespür haben. Man wird kaum einen zehnjährigen Juristen finden - aber es gibt sehr viele zehnjährige Programmierer!



Die Basis des BASIC

Fast alle Heimcomputer arbeiten mit BASIC, einer Programmiersprache, die der normalen (englischen) Sprache am nächsten kommt. Der Name BASIC sagt heute nichts mehr über den Schwierigkeitsgrad aus: Viele sehr komplexe und leistungsfähige Programme werden in BASIC geschrieben.

Es kann aber nicht daran gezweifelt werden, daß der Name BASIC viele Anfänger für sich gewonnen hat, da er in dem Wirrwarr der vielen vorhandenen Programmiersprachen vielversprechender klingt. Dies hat wahrscheinlich zu der weiten Verbreitung von BASIC geführt.

BASIC ist eine Computersprache, die eine Anzahl zulässiger Befehle interpretiert, und dann während des Programmablaufs Operationen an Daten durchführt. Im Gegensatz zum menschlichen Wortschatz, der 5000 bis 8000 Wörter umfaßt, kommt BASIC mit ungefähr 200 Wörtern aus. Der Einsatz dieser Wörter in einem Programm unterliegt strengen Regeln. Die Syntax muß genau eingehalten werden, und jeder Versuch, eine andere Schreibweise oder Sprache zu verwenden, endet mit der kalten und nüchternen Meldung:

```
Syntax error
```

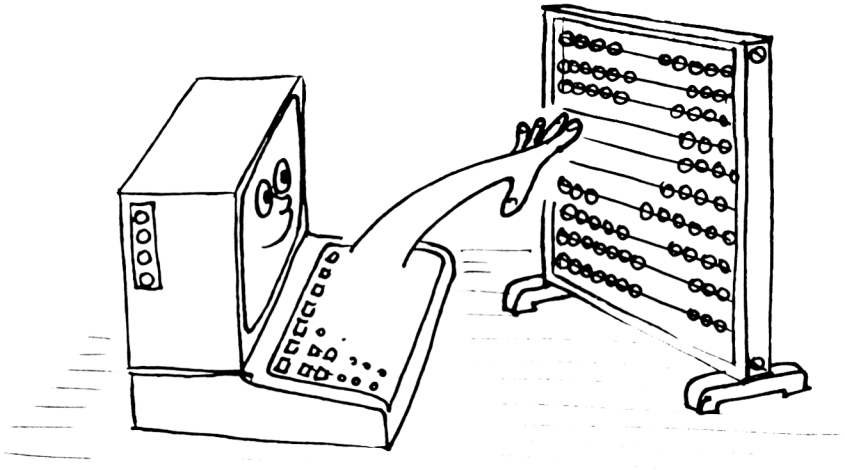
Dies ist nicht so einschränkend, wie es auf den ersten Blick erscheint, da die BASIC-Sprache (die Syntax) in erster Linie für die Bearbeitung numerischer Daten gedacht ist. Die Worte stellen im wesentlichen nur Erweiterungen der bekannten mathematischen Operatoren wie + und - dar. Die wichtigste Tatsache, die ein Anfänger begreifen muß, ist die, daß der Computer nur mit numerischen Daten arbeiten kann, da jede Information nur in der Form von numerischen Daten an die Zentral-Einheit (CPU = Central Processor Unit) übergeben werden kann.

Zahlen bitte!

Würde der Computer dazu benutzt, Goethes gesammelte Werke zu speichern, so wäre im ganzen System kein einziger Buchstabe und kein einziges Wort zu finden. Jedes Stück der Information wird erst in eine Zahl umgewandelt, bevor der Computer mit ihm arbeiten kann.

BASIC übersetzt die Worte in Zahlen, die der Computer dann mit Mitteln der Addition, Subtraktion und der Boole'schen Logik bearbeitet. Bei Boole'schen Operationen werden Daten verglichen und Attributen zugeordnet. Es wird z.B. geprüft, ob eine Zahl größer als eine andere ist, und wenn eine Zahl eine gegebene Voraussetzung erfüllt, wird ein bestimmter Ablauf eingeleitet.

Mit Hilfe des Programms zerlegt der Computer jede Aufgabe in eine Serie von einfachen Ja/Nein-Operationen.



Wenn Ihnen dieser Prozeß umständlich vorkommt, haben Sie recht, und die erste und wichtigste Wahrheit der Datenverarbeitung erkannt. Ein Computer ist ein Werkzeug zur Ausführung allereinfachster, immer wiederkehrender Aufgaben -sehr schnell und absolut präzise. BASIC interpretiert also die in einem Programm enthaltenen Anweisungen und wandelt sie in eine Sprache (Maschinencode) um, die von der Zentraleinheit verstanden wird. Der Computer versteht nur zwei Zustände: 'Ja' und 'Nein', die in binärer Schreibweise mit 1 und 0 dargestellt werden. In der Boole'schen Logik gibt es nur 'wahr' und 'falsch', kein 'jein' oder 'vielleicht'!

Das Umschalten zwischen diesen beiden bestimmten Zuständen bildet den Kern des Begriffs 'digital'. In der Natur verlaufen die meisten Prozesse allmählich und geradlinig von einem stabilen Zustand zu einem anderen; der Übergang verläuft langsam über eine Verbindungslinie zwischen den beiden Zuständen. In einer idealen digitalen Umgebung dagegen wird buchstäblich in Null-Komma-Nichts von einem Zustand auf einen anderen umgeschaltet. In Wirklichkeit verursachen die Grenzen der Halbleiter-Physik jedoch eine minimale Verzögerung, die sogenannte Laufzeitverzögerung. Die Summe vieler Laufzeitverzögerungen sind der Grund, weshalb es etwas Zeit braucht, bis die Information verarbeitet ist und die Antwort ausgegeben wird.

In jedem Fall muß der Computer eine bestimmte Zeit warten, bis eine Aufgabe fertig ist und er deren Ergebnis für die nächste Operation verwenden kann. In einer idealen Umgebung müßte also eine künstliche Verzögerung eingebaut werden. Der digitale Prozeß kennt nur schwarz und weiß; die dazwischenliegenden Grautöne sind bedeutungslos. Der lineare oder 'analoge' Übergang hingegen führt über alle Grautöne.

Da die letztendliche Antwort nur 0 oder 1 lauten kann, gibt es kein 'fast richtig'. Daß der Computer manchmal scheinbar Fehler bei Operationen mit numerischen Daten macht, hängt damit zusammen, daß er nur Zahlen bis zu einer bestimmten Größe verarbeiten kann. Numerische Daten, die diese Grenze überschreiten, werden abgeschnitten oder gerundet, damit sie in den verfügbaren Platz passen. Beispielsweise wird 999,999,999 zu 1,000,000,000.

Wie kann man nun in einer Welt, die nur 0 und 1 kennt, weiter als bis 1 zählen?

Bits und Bytes

Wir haben uns zufällig daran gewöhnt, mit Zahlen des Dezimalsystems umzugehen, in dem der Bezugspunkt die Zahl 10 ist, d.h. zur Darstellung von Mengen zwischen 0 und 9 (0 bis 9 ist der Schreibweise 1 bis 10 vorzuziehen) stehen uns zehn Ziffern zur Verfügung. Das Zahlensystem mit zwei Ziffern (0 und 1) heißt Dualsystem oder Binärsystem, und die Einheiten, mit denen das System arbeitet, heißen Bits (Abk. für Binary digIT).

Der Zusammenhang zwischen Bits und der dezimalen Schreibweise ist leicht zu verstehen:

Normalerweise wird die maximale Anzahl der verwendeten Binärziffern in einer Zahl angegeben, indem soviele Nullen vorangestellt werden, daß die Bit-Anzahl vollständig ist.

Aus der Dezimalzahl 7 wird:

00111

...in 5-Bit-Schreibweise.

Die Ziffern im Binärsystem können einfach als Indikatoren angesehen werden, die für ihre Spalte angeben, ob eine bestimmte Potenz von 2 gegeben ist. 1 = ja; 0 = nein

$2^0=1$
$2^1=2=2=2(2^0)$
$2^2=4=2x2=2(2^1)$
$2^3=8=2x2x2=2(2^2)$
$2^4=16=2x2x2x2=2(2^3)$

Die Spalten können wie folgt aussehen:

2^4 2^3 2^2 2^1 2^0

1 0 0 1 1
(16 + 0 + 0 + 2 + 1) = 19

Acht Bits, also acht Einheiten einer Binärzahl, werden als Byte bezeichnet. Die höchste Zahl, die in einem Byte gespeichert werden kann, ist (binär) 11111111 -oder dezimal 255. Das ergibt 256 echte Variationen, einschließlich 00000000 (auch ein zulässiger Datenwert für den Computer).

Computer behandeln Daten als Vielfache von 8 Bits. 256 ist keine sehr große Zahl, so daß zur Adressierung des Speichers zwei Bytes verwendet werden. Der Speicher kann als Bereich mit einer horizontalen und einer vertikalen Achse betrachtet werden, in dem jedes Element lokalisiert werden kann.

	0	1	2	3	4	5	6	7	8	9
0										
1										
2										
3										
4										
5				1		1				
6										
7										
8										
9										

Der abgebildete Bereich kann bis zu 10x10 Informationseinheiten aufnehmen, deren Adressen die Zahlen 0 bis 9 enthalten. Auf der Position 3,5 ist die Information 1 gespeichert, ebenso auf der Position 5,5.

Ein Bereich von 256 x 256 Einheiten kann 65.536 Einzelpositionen enthalten, die mit 8-Bit-Adressen für die waagrechten und senkrechten Achsen angesprochen werden. Mit '0' und '1' können wir also immerhin 65.536 Elemente addressieren.

Die nächstgrößere binäre Einheit heißt Kilobyte (kByte oder einfach K). Ein Kilobyte hat 1024 Bytes. 1024 ist das binäre Vielfache, das der Dezimaleinheit 'Kilo' (1000) am nächsten kommt. Dies erklärt, warum ein Computer mit einem '64K'-Speicher im Wirklichkeit mit 65.536 Bytes (64x1024) ausgestattet ist.

Glücklicherweise führt BASIC alle notwendigen Umformungen für Sie durch, und Sie können ein fähiger Programmierer werden, ohne das Binärsystem vollständig zu durchschauen. Das Verständnis des Binärsystems wird Ihnen jedoch helfen, die Bedeutung der 'magischen' Zahlen zu erfassen, die unvermeidlich auftauchen werden, wenn Sie sich intensiver mit der Datenverarbeitung beschäftigen.

Es lohnt sich, die Mühe aufzuwenden, um das Binärsystem und verschiedene wichtige Zahlen wie 255, 1024 usw. zu verstehen, da es sehr unwahrscheinlich ist, daß sich diese grundlegende Arbeitsweise von Computern in absehbarer Zeit ändern wird. Die Zuverlässigkeit und Einfachheit, die sich aus der Arbeit mit nur zwei Zuständen ergibt, wird der weitaus größeren Komplexität, die jedes andere Zahlensystem mit sich bringt, auf jeden Fall vorzuziehen sein.

Aber...

Die binäre Schreibweise ist denkbar einfach und elegant, aber sie ist aufwendig zu schreiben und führt leicht zu Ungenauigkeiten, da sie nicht leicht zu lesen ist. Deshalb werden neben dem Binärsystem einige verwandte Zahlensysteme verwendet, die dem Programmierer sozusagen als Kurzschrift dienen. Eines dieser Zahlensysteme, das vor allem bei Mikro-Computern eingesetzt wird, heißt HEX (Kurzform von Hexadezimalsystem).

Die Basiszahl in Hex ist 16, d.h. die Zahlen 0 bis 15 werden mit einem einzigen Zeichen dargestellt:

Dezimal

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Hex

0 1 2 3 4 5 6 7 8 9 A B C D E F

Das Hexadezimalsystem kann die acht Bits eines Bytes in zwei Blöcke zu je vier Bits aufteilen, da 15 eine 4-Bit-Zahl ist: 1111 binär. Der erste Block enthält die Anzahl kompletter '15er'-Einheiten, der zweite Block enthält den Divisionsrest -hier kommt die ganze Eleganz der Binär- und Hex-Systeme zum Tragen.

Hier noch einmal die Tabelle zur Einführung in die Binär-Schreibweise:

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

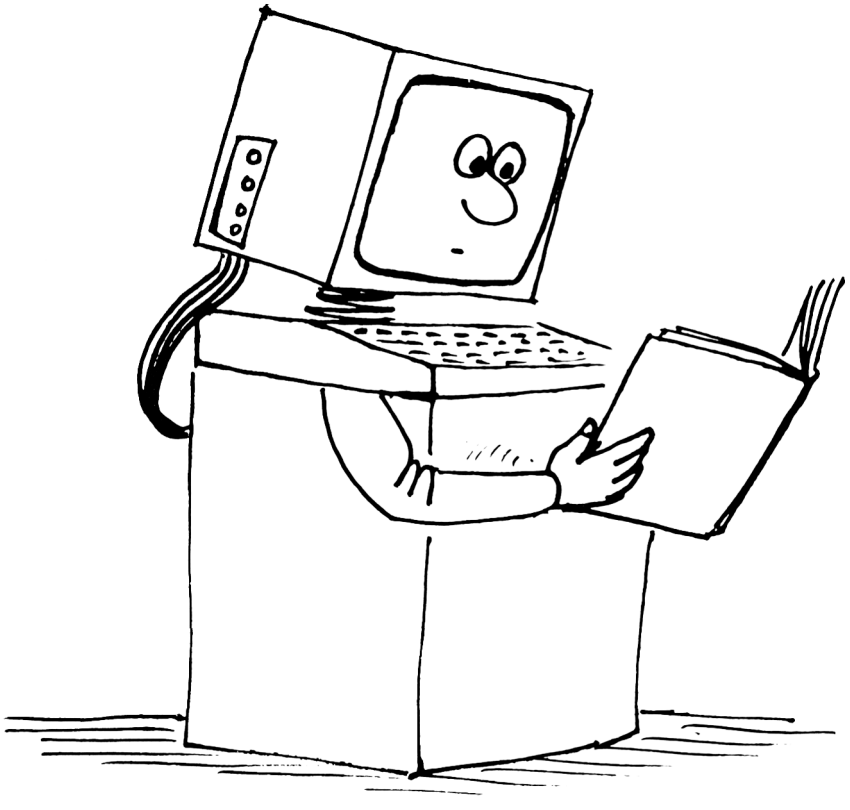
Eine 8-Bit-Zahl, z.B. 11010110 (&D6 hex), kann in zwei 4-Bit-Zahlen, sogenannte Halbbytes, unterteilt werden. In diesem Handbuch werden Hex-Zahlen immer durch ein '&'-Zeichen kenntlich gemacht. Das Hex-System wird auch von den meisten Programmierern für Assembler-Programme benutzt. Ein Programm in Assembler-Sprache kommt einem Maschinencode-Programm am nächsten, da hierbei mnemonische Symbole zur Bezeichnung von Maschinencode-'Nummern' verwendet werden.

Wenn Sie eine Hex-Zahl lesen, stellen Sie anhand der ersten Ziffer fest, wieviel mal die 16 darin enthalten ist. Dann addieren Sie den Rest, der in der zweiten Hälfte der Hex-Zahl angegeben ist, und erhalten so den Dezimalwert der Zahl. Man ist nur allzu leicht geneigt, z.B. die Zahl &D6 als 13+6 = 136 zu interpretieren. Richtig umgewandelt lautet die Zahl aber (13x16)+6 = 214.

Genauso verhält es sich beim Dezimalsystem: Die Zahl 89 z.B. bedeutet (8x10)+9. Die Multiplikation mit 10 kommt Ihnen einfacher vor, weil Sie viel mehr Übung damit haben als mit 16.

Wenn Sie soweit gekommen sind, ohne allzu verwirrt zu sein, haben Sie schon eine gute Vorstellung davon, wie ein Computer funktioniert. Und wenn Sie sich fragen, wozu das alles - haben Sie recht. Ein Computer ist eine Maschine, die mit ganz simplen Konzepten und Prinzipien arbeitet. Das einzig bemerkenswerte daran ist, daß er seine Aufgaben mit unglaublich hoher Geschwindigkeit bewältigt (in Millionsteln von Sekunden), und eine enorme Fähigkeit hat, sich sowohl die eingegebenen Daten als auch die Zwischenergebnisse von Tausenden von Einzelrechnungen auf dem Weg zum Endergebnis zu merken.

Wenn Sie sich weiter in die Computer-Theorie vertiefen möchten, können Ihnen Tausende von Büchern weiterhelfen. Einige werden Sie nur verwirren, andere werden Sie sehr viel weiterbringen und Ihnen enthüllen, wie schlicht und einfach die grundlegenden Zusammenhänge zwischen Zahlensystemen und der Arbeitsweise Ihres Computers sind.



Teil 2:
...der CPC6128 im besonderen

Dieser Abschnitt behandelt einige maschinenspezifische Aspekte des CPC6128. Hintergrund-Informationen zu diesen Sachgebieten finden Sie sowohl im einführenden 1. Kapitel, als auch im 3. Kapitel über die AMSTRAD CPC6128-BASIC-Schlüsselwörter.

Behandelte Themen dieses Abschnitts:

- ★ Zeichensatz
- ★ ASCII
- ★ Variablen
- ★ Logik
- ★ Benutzer-definierte Zeichen
- ★ Druckformatierung
- ★ Windows
- ★ Unterbrechungen
- ★ Daten
- ★ Tonerzeugung
- ★ Graphik
- ★ Graphik unter Verwendung des zweiten 64k-Speichers

Zeichensatz

Wenn Sie die Tasten auf Ihrem 6128 drücken, sollten Sie es nicht als Selbstverständlichkeit betrachten, daß auf dem Bildschirm erkennbare Buchstaben und Zeichen erscheinen. Wie wir bereits ausgeführt haben, handelt es sich beim Computer schließlich nicht um eine Schreibmaschine. Tatsächlich werden beim Druck auf eine Taste eine Reihe kombinierter Schalter betätigt.

on den Schaltkreisen im Dadurch werden elektrische Impulse ausgelöst, die v
setzt werden. Computer in ein Punktmuster auf dem Bildschirm umge

Wir identifizieren dieses Muster als einen Buchstaben, eine Zahl oder irgendein anderes Zeichen aus dem Zeichensatz des 6128.

Ein Teil der Zeichen, die der 6128 produzieren kann, kommt nicht durch einen einfachen Tastendruck zustande, sondern wird nur durch den Befehl `PRINT CHR$(‹Zahl›)` angezeigt. Das kommt daher, daß jede Information in einer Dateneinheit, Byte genannt, gespeichert wird. Wie bereits im 1. Teil dieses Kapitels beschrieben, kann ein Byte einen von 256 möglichen Werten darstellen. Da der Computer mindestens ein ganzes Byte zum Speichern eines Zeichens benötigt (dies ist der kleinste Wert, den der Computer aufnimmt, ob es uns gefällt oder nicht), können wir ebensogut 256 Zeichen bilden, statt uns mit den ca. 96 Standardzeichen zu begnügen, über die die meisten Schreibmaschinen verfügen, und 160 Möglichkeiten ungenutzt zu lassen.

Die Standardzeichen aus dem Zeichenvorrat werden als 'Untermenge' bezeichnet. Sie ist überall in der Computerwelt als ASCII-Zeichensatz bekannt (ASCII steht für 'American Standard Code for Information Interchange'). Es ist in erster Linie ein Code, der dafür sorgt, daß Daten bei der Übertragung von einem Computer zum anderen ihre Form behalten. Im Kapitel 7 sind die ASCII-Zeichen, sowie die zusätzlichen 6128-spezifischen Zeichen, mit Angabe ihrer Code-Nummern aufgelistet.

Wie gehen wir am besten vor?

Mittlerweile ist Ihnen dieses Programm wahrscheinlich nicht mehr vollkommen unverständlich:

```
10 FOR n=32 TO 255
20 PRINT CHR$(n);
30 NEXT
```

Es führt den gesamten Zeichensatz auf. Was bedeutet dieses Programm nun im einzelnen?

Zunächst ist festzustellen, daß der Computer nicht angewiesen wurde: `PRINT "abcdefghijklmn..."` sondern: `PRINT CHR$(n)`. n ist dabei eine Variable. Eine Variable ist eine Informationseinheit, deren Wert den Anweisungen des Programms entsprechend variiert.

Die Wahl des Buchstaben n in diesem Fall ist rein zufällig. Es kann stattdessen jeder andere Buchstabe oder eine Buchstabenfolge eingesetzt werden, solange es sich nicht um ein Schlüsselwort handelt.

Woran erkennt man eine Variable?

Eine Zahl wie 5 hat einen festen Wert. Sie hat einen festen Platz (zwischen 4 und 6) und ist deshalb unveränderlich. Der Buchstabe n hat auch einen festen Platz -im Alphabet.

Wie erkennt der Computer den Unterschied? Wäre der Buchstabe n als alphabetisches Zeichen erklärt worden, so hätten wir n in Anführungszeichen eingeschlossen ('n') und der Computer hätte mit der Fehlermeldung Syntax error reagiert, da er die Anweisung `FOR "n"=32 TO 255` nicht versteht.

Einfach durch die Tatsache, daß n ohne Anführungszeichen erschien, wurde dem Computer mitgeteilt, daß es sich dabei um eine Variable handelt. In einer Anweisung mit `FOR` muß nach dem Schlüsselwort eine Variable stehen, deshalb betrachtet der Computer das, was auf `FOR` folgt, als Variable.

Wir haben dem Computer fernerhin mitgeteilt, daß `n=32` bis (`TO`) `255` ist. Somit haben wir den Bereich der Variablen festgelegt. Es handelt sich tatsächlich um eine Zahlenfolge, die mit `32` beginnt und mit `255` endet.

Nachdem die Variable definiert ist, muß dem Computer gesagt werden, was mit ihr geschehen soll. Dies steht in Zeile 20:

```
20 PRINT CHR$(n);
```

Die Anweisung bedeutet, daß der Computer unabhängig vom laufenden numerischen Wert von n in seinem Speicher nachzusehen hat, welche Zeichennummer diesem Wert entspricht und das zugehörige Zeichen auf dem Bildschirm wiedergeben soll.

Das Semikolon am Ende von Zeile 20 verhindert, daß nach jedem Zeichen, das angezeigt wird, eine neue Zeile angefangen wird. Ansonsten wird jedes Zeichen in der ersten Spalte einer neuen Zeile ausgedruckt.

Nachdem die Anweisung von Zeile 20 mit dem ersten Wert der Zahlenfolge von n (nämlich 32) durchgeführt wurde, wird der Computer in Zeile 30 angewiesen, zu der Zeile zurückzukehren, in der sich das `FOR` befindet, um dieselbe Aufgabe mit dem nächsten Wert von n durchzuführen. Diesen wiederkehrenden Vorgang nennt man Schleife (engl. loop). Schleifen gehören zu den wichtigsten und fundamentalsten Bestandteilen von Programmen. Sie ersparen das Schreiben vieler wiederkehrender Programmteile, und Sie werden diese Technik bald schätzen lernen, wenn Sie selber Programme schreiben.

Wenn die **FOR NEXT**-Schleife ihren Grenzwert (255) erreicht hat, ist die Operation ausgeführt, und der Computer sucht nach der Zeile, die auf 30 folgt. Es liegt jedoch keine weitere Anweisung mehr vor. Also beendet er das Programm und kehrt zum Direktmodus zurück, d.h. auf dem Bildschirm erscheint **Ready**. Dies ist ein Bereitschaftszeichen (auch 'Prompt' genannt), mit dem der Computer signalisiert, daß er auf weitere Anweisungen wartet. Sie können nun z.B. **RUN** eintippen, und das Programm noch einmal von vorne ablaufen lassen. Das Programm ist sicher im Speicher verwahrt, bis Sie es durch einen Befehl löschen oder den Computer abschalten.

Dieses Programm illustriert einen wesentlichen Aspekt über das Arbeiten mit Computern, daß nämlich alles, was der Computer tut, über Zahlen läuft. Der Computer hat das Alphabet - und eine ganze Reihe anderer Zeichen -ausgedruckt, indem er jeweils eine Zahl für das gewünschte Zeichen benutzt hat. Wenn Sie die **A**-Taste drücken, sagen Sie dem Computer nicht, daß er **A** auf dem Bildschirm darstellen soll, sondern Sie beauftragen ihn, in seinem Speicher die numerische Information zu suchen, die das **A** auf den Bildschirm bringt. Die Stelle im Speicher, wo sich diese Information befindet, ist durch den Zahlencode definiert, der durch das Drücken der Taste aktiviert wird. Jedem Zeichen ist eine Nummer zugeordnet; eine Liste aller Zeichen und der zugehörigen Nummern finden Sie im 3. Teil von Kapitel 7.

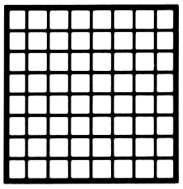
In gleicher Weise wird auch der dargestellte Buchstabe nicht auf den Bildschirm 'geschrieben'; es läuft alles über Nummern.

Der ASCII-Code für den Buchstaben A beispielsweise ist 97. Der Computer versteht aber auch 97 nicht (dieses blöde Ding!). Die uns verständliche Dezimalzahl muß erst in einen Maschinen-Code umgesetzt werden. Das Grundprinzip des Maschinen-Codes wurde weiter vorn in diesem Kapitel erläutert.

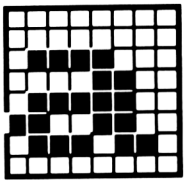
Anfangs fällt es uns sehr schwer, Zahlen von der gewohnten Dezimalschreibweise in Hexadezimalzahlen umzuwandeln. Das Dezimalsystem ist für uns so natürlich, daß uns die Arbeit mit einem anderen System ähnlich ungewohnt vorkommt, als müßten wir beim Essen plötzlich Messer und Gabel in der anderen Hand halten.

Es erfordert ein wenig Übung, sich mit der Hex-Schreibweise zurechtzufinden, aber wenn Sie es erst einmal geschafft haben, wird Ihnen manches leichter fallen und verständlicher werden. Wenn Sie das Binär- und Hexadezimalsystem noch nicht richtig verstanden haben, empfehlen wir Ihnen, Teil 1 dieses Kapitels noch einmal durchzuarbeiten.

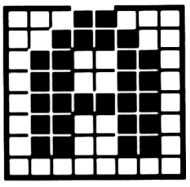
Wenn der Computer den Anschlag der **A** -Taste in eine für ihn verständliche Zahl umgesetzt hat, aktiviert er den Teil seines Speichers, der mit dieser Zahl angesprochen wird. Dort befindet sich eine Serie von Zahlen, die das A -Zeichen definieren. Das Zeichen, das dann auf dem Bildschirm erscheint, ist aus einem Datenblock zusammengesetzt, der im Speicher als numerische Matrix abgelegt ist.



Matrix (Raster) für ein Leerzeichen



Kleines a



Großes A

Die Matrix besteht aus Spalten und Zeilen für Punkte. Ein Zeichen wird dargestellt, indem das erforderliche Punktmuster aufleuchtet. Jeder Punkt wird durch Daten im Computer-Speicher gesteuert. Beim CPC6128 hat die Matrix oder 'Zelle' jedes Zeichens acht Zeilen und acht Spalten. Möchten Sie ein Zeichen darstellen, das im Zeichenvorrat des Computers nicht enthalten ist, haben Sie die Möglichkeit, mit Hilfe des Befehls `SYMBOL` selbst ein Zeichen zu definieren. Wie Sie das anstellen, wird einige Seiten weiter beschrieben.

Um ein Zeichen selbst zu definieren, können Sie zwischen 0 und 64 Zeichen beliebig kombinieren und anordnen. Der 'komplette' Zeichensatz mit Zeichen in allen möglichen Punkte-Kombinationen würde also weit mehr als 255 Zeichen umfassen. Wenn Sie dann noch bedenken, daß sich mehrere Zeichen zu großen Zeichenblöcken zusammenfügen lassen, erkennen Sie, daß Sie nahezu unbegrenzte Möglichkeiten haben, eigene Zeichen zu erfinden.

Logik

Ein grundlegender Unterschied zwischen einer Rechenmaschine und einem Computer ist die Fähigkeit des Computers, logische Operationen, wie sie in der `IF THEN` -Anweisung gefordert werden, durchführen zu können. Die logischen Operatoren zerlegen die Werte, die ihnen angegeben werden, in Bit-Muster, und vergleichen sie Bit für Bit. Das ist alles ganz logisch - es ist nur furchtbar schwierig, Logik in einfachen Worten ohne präzise Definitionen zu beschreiben.

Die beiden Hälften eines logischen Ausdrucks werden Argumente genannt. Ein logischer Ausdruck ist so aufgebaut:

‹Argument›[‹Vergleichsoperator›‹Argument›]

‹Argument› kann dabei folgendes sein:
- Nicht ‹Argument›
- ‹numerischer Ausdruck›
- ‹Vergleichsausdruck›
- ‹logischer Ausdruck›

Beide Argumente für eine logische Operation müssen ganze Zahlen sein, sonst erscheint Fehlermeldung 6.

Die Vergleichsoperatoren in der Reihenfolge ihrer Priorität mit dem Ergebnis für jeden Bit-Vergleich sind:

AND Ergebnis ist 0, es sei denn, beide Argument-Bits sind 1.
OR Ergebnis ist 1, es sei denn, beide Argument-Bits sind 0.
XOR Ergebnis ist 1, es sei denn, beide Argument-Bits sind gleich.

AND ist der meistbenutzte Operator und bedeutet NICHT dasselbe wie 'addieren'!

```
PRINT 10 AND 10
```

...führt zum Ergebnis 10.

```
PRINT 10 AND 12
```

...führt zum Ergebnis 8.

```
PRINT 10 AND 1000
```

... führt auch zum Ergebnis 8.

Das gleiche Ergebnis kommt zustande, weil die Zahlen 10 und 1000 in Binärdarstellung umgewandelt wurden:

```
      1010
1111101000
```

Bei der Operation mit AND wird Bit für Bit verglichen. Wenn das Bit in der oberen UND in der unteren Zeile 1 ist, lautet das Ergebnis 1:

```
0000001000
```

Dies ist die binäre Darstellung der Dezimalzahl 8. Der Vergleichsoperator AND prüft also, wo zwei Bedingungen gleichzeitig erfüllt sind. Das folgende Beispiel soll dies verdeutlichen:

```
10 INPUT "Der wievielte Tag des Monats ist heute";datum
20 INPUT "Welchen Monat haben wir";monat
30 IF datum=24 AND monat=12 GOTO 50
40 CLS:GOTO 10
50 PRINT "Frohe Weihnachten!"
```

OR vergleicht ebenfalls Bits. Das Ergebnis ist immer 1, außer wenn beide Bits 0 sind. Dann lautet das Ergebnis 0. Nehmen wir dieselben Zahlen wie im AND-Beispiel:

```
PRINT 1000 OR 10
1002
```

Bitmuster:

```
      1010
1111101000
```

Das Ergebnis ist:

```
1111101010
```

Anwendung von OR im Programmbeispiel:

```
10 CLS
20 INPUT "Welchen Monat haben wir";monat
30 IF monat=12 OR monat=1 OR monat=2 THEN 50
40 GOTO 10
50 PRINT "Jetzt ist Winter"
```

Der NOT-Operator kehrt das Bitmuster des Arguments um, d.h. aus 0 wird 1 und aus 1 wird 0:

```
10 CLS
20 INPUT "Welchen Monat haben wir";monat
30 IF NOT (monat=6 OR monat=7 OR monat=8) THEN 50
40 GOTO 10
50 PRINT "Jetzt ist nicht Sommer!"
```

Bedenken Sie, daß Sie mehrere logische Bedingungen (bis zur maximalen Zeilenlänge) miteinander verknüpfen können, um einen Zustand herauszukristallisieren:

```
10 INPUT "Der wievielte Tag des Monats ist heute";datum
20 INPUT "Welchen Monat haben wir";monat
30 IF NOT (monat=12 OR monat=1) AND datum=29 THEN 50
40 CLS:GOTO 10
50 PRINT "Wir haben weder Dezember noch Januar, aber
 dieses Jahr koennte ein Schaltjahr sein"
```

Das Ergebnis eines Vergleichsausdrucks ist entweder -1 oder Ø. Beim Bitmuster für -1 sind alle Bits der ganzen Zahl 1; beim Bitmuster für Ø sind alle Bits der ganzen Zahl 0. Das Ergebnis einer Vergleichsoperation mit zwei solchen Argumenten lautet entweder -1 für 'wahr' (True) oder Ø für 'falsch' (False).

Prüfen Sie dies, indem Sie dem Programm zwei Zeilen hinzufügen:

```
60 PRINT NOT (monat=12 OR monat=1)
70 PRINT (monat=12 OR monat=1)
```

Wenn Sie das Programm laufen lassen, und für den Tag 29 und den Monat 2 angeben, erscheint auf dem Bildschirm die Aussage aus Zeile 50, gefolgt von den Werten, die sich aus den logischen Ausdrücken in Zeile 60 und 70 ergeben.

Bei einer Operation mit XOR (ausschließendes oder eXklusives OdeR) schließlich wird 'wahr' ausgegeben, wenn beide Argumente unterschiedlich sind.

Die folgende Tabelle, eine sogenannte 'Wahrheitstafel' zeigt alle Möglichkeiten, die es bei logischen Vergleichsoperationen gibt. Sie können daran bequem ablesen, was bei Bit-weisen Vergleichsoperationen passiert.

Argument A	1010
Argument B	0110
AND Ergebnis	0010
OR Ergebnis	1110
XOR Ergebnis	1100

Benutzer-definierte Zeichen

Das erste Mal werden Sie vielleicht mit Binärzahlen zu tun haben, wenn Sie mit dem SYMBOL-Befehl eigene Zeichen entwerfen wollen. Wird das Zeichen auf einer 8x8 Matrix vorbereitet, dann kann jede der 8 Reihen in einen Binärwert umgesetzt werden, indem man 1 für jedes Pixel einsetzt, das sichtbar sein soll, und 0 für die Pixel, die unsichtbar, d.h. in der Grundfarbe des Bildschirms dargestellt werden sollen. Diese 8 Ziffern werden im SYMBOL-Kommando eingesetzt. Möchten Sie z.B. ein Zeichen mit einem Haus definieren:

```
   *     = ØØØØ1ØØØ = &Ø8 =                8        = 8
  ****   = ØØ1111ØØ = &3C =          32+16+8+4      = 6Ø
 *    *  = Ø1ØØØØ1Ø = &42 =      64              +2 = 66
* *  * * = 1Ø1ØØ1Ø1 = &A5 = 128   +32     +4   +1 = 165
*      * = 1ØØØØØØ1 = &81 = 128                 +1 = 129
* ** * * = 1Ø11Ø1Ø1 = &B5 = 128   +32+16  +4   +1 = 181
* **   * = 1Ø11ØØØ1 = &B1 = 128   +32+16       +1 = 177
******** = 11111111 = &FF = 128+64+32+16+8+4+2+1 = 255
```

...lautet der Befehl:

```
SYMBOL 240,8,60,66,165,129,181,177,255
```

oder:

```
SYMBOL 240,&08,&3C,&42,&A5,&81,&B5,&B1,&FF
```

oder:

```
SYMBOL 240, &X00001000, &X00111100, &X01000010, &X10100101,
            &X10000001, &X10110101, &X10110001, &X11111111
```

Um dieses Zeichen mit dem Haus abzubilden, müßten Sie eingeben:

```
PRINT CHR$(240)
```

Um schließlich mehrere Zeichen zu einem Block zusammenzuschließen, schreiben Sie:

```
doppel$=CHR$(240)+CHR$(240)
PRINT doppel$
```

...oder:

```
reihen$=STRING$(15,240)
PRINT reihen$
```

Druck-Formatierung

PRINT ist einer der Befehle, die Sie bei Ihren allerersten Programmierversuchen kennenlernen. Sie können mit ihm aber nicht nur angeben, was auf den Bildschirm gedruckt wird, sondern auch wo und wie.

In der einfachsten Form des PRINT-Befehls wird hinter PRINT angegeben, was gedruckt werden soll. Dies kann eine Zahl, eine Zeichenfolge (String) oder der Name einer Variablen sein:

```
PRINT 3
 3

PRINT "hallo"
hallo

a=5
PRINT a
 5

a$="test"
PRINT a$
test
```

In einer **PRINT**-Anweisung können mehrere Elemente aufgezählt werden, die jeweils durch ein Trennzeichen, durch **TAB** oder durch **SPC** gegeneinander abgegrenzt werden können. Trennzeichen sind Semikolon und Komma. Wird ein Semikolon verwendet, so werden die Elemente direkt hintereinander aufgereiht, bei einem Komma wird das folgende Element in die nächste Druckzone gesetzt. Die Breite einer Zone beträgt normalerweise 13 Spalten, doch kann die Breite mit dem Befehl **ZONE** verändert werden.

```
PRINT 3;-4;5
 3 -4 5

PRINT "halli ";"hallo"
halli hallo

PRINT "halli","hallo"
halli         hallo

PRINT 3,-4,5
 3            -4              5

ZONE 4
PRINT 3,-4,5
 3  -4   5
```

Anmerkung: Positiven Zahlen geht eine Leerstelle voraus, während negative Zahlen mit einem vorangestellten Minus-Zeichen gekennzeichnet werden. Außerdem folgt allen Zahlen eine Leerstelle. Zeichenfolgen werden genauso wiedergegeben, wie sie zwischen den Anführungszeichen dargestellt sind.

In der Funktion SPC wird ein numerischer Ausdruck als Parameter angegeben, der die Anzahl der auszudruckenden Leerstellen angibt. Ist dieser Wert negativ, wird Null angenommen, ist er größer als der gegenwärtige Bildschirmbereich (Window), wird die Breite des Windows angenommen.

```
PRINT SPC(5)"hi"
    hi
x=3
PRINT SPC(x*3)"hi"
        hi
```

Die Funktion TAB ist ähnlich, mit dem Unterschied, daß so viele Leerstellen gedruckt werden, wie nötig sind, um den Ausdruck in die gewünschte Spalte zu setzen.

Der Ausgabebereich (Stream) ist normalerweise 0 (der volle Bildschirm), wenn nicht vor der Liste der auszudruckenden Elemente eine Stream-Nummer (#) angegeben wird. Andere Stream-Nummern bezeichnen Teilbereiche des Bildschirms. Stream 8 und 9 sind Sonderfälle: Alles, was über Stream 8 geschickt wird, gibt der Drucker aus (sofern einer angeschlossen ist). Wird Stream 9 angegeben, so werden die auszudruckenden Elemente in eine Disketten- (oder Kassetten-) Datei geschrieben. Für diesen Zweck sollte jedoch anstelle von PRINT lieber der Befehl WRITE benutzt werden.

```
PRINT "hallo"          -Fenster 0
hallo

PRINT #0,"hallo"       -ebenfalls Fenster 0
hallo

PRINT #4,"hallo"       -Fenster 4
hallo                  (Im oberen Teil des Bildschirms)

PRINT #8,"hallo"       -auf dem Drucker (falls angeschlossen)
hallo
```

TAB und SPC eignen sich für einfache Druckformate; um jedoch detailliertere Angaben zum Format zu machen, verwendet man den Befehl PRINT USING und eine passende Formatschablone. Eine Formatschablone ist ein aus Sonderzeichen bestehender Textausdruck. Jedes Sonderzeichen definiert einen Teil des Formats. Diese 'Formatzeichen' werden unter dem Schlüsselwort PRINT USING im 3. Kapitel genau beschrieben. Vielleicht läßt sich ihre Verwendung jedoch durch die folgenden Beispiele noch weiter verdeutlichen.

```
PRINT USING "\      \";"Test String"
Test S
```

"!" kann benutzt werden, um das erste Zeichen eines Strings zu drucken.

```
PRINT USING "!";"Test String"
T
```

Das gebräuchlichste String-Formatzeichen ist '&'. Normalerweise setzt BASIC einen String in eine neue Zeile, wenn er nicht mehr in die alte paßt. Mit `PRINT USING "&"` wird dieser Automatismus unterbunden.

(Wählen Sie `BORDER 0`, damit Sie den Rand der Schreibfläche erkennen können.)

```
MODE 1:LOCATE 37,1:PRINT "zu lang"

                                                    -Zeile 1
zu lang                                             -Zeile 2

MODE 1:LOCATE 37,1:PRINT USING "&";"zu lang"

                                                zu l    -Zeile 1
ang                                                     -Zeile 2
```

Zur Darstellung von Zahlen lassen sich viele verschiedene Schablonen erstellen. Die einfachste ist wahrscheinlich `PRINT USING "#####"`, wobei jedes # für eine Ziffer innerhalb der Zahl steht.

```
PRINT USING "######";123
   123
```

Durch '.' wird die Stelle des Dezimalpunkts festgelegt:

```
PRINT USING "####.#####";12.45
  12.45000
```

Wird ',' vor dem Dezimalpunkt in die Schablone eingesetzt, so werden die Ziffern vor dem Komma zu Dreiergruppen zusammengefaßt und durch Komma voneinander abgegrenzt:

```
PRINT USING "########,.####";123456.78
  123,456.7800
```

Darüberhinaus kann das Format die Währungszeichen für Dollar und Pfund enthalten, die dann direkt vor die jeweilige Zahl gesetzt werden, auch wenn diese Zahl das Format nicht ausfüllt. Die Währungszeichen werden als '$$' und '££' in die Formatschablone eingefügt:

```
PRINT USING "$$##";7
  $7

PRINT USING "$$##";351
$351

PRINT USING "ff#####,.##";1234.567
 f1,234.57
```

Wie Sie sehen, wird in diesem Fall der Teil hinter dem Dezimalpunkt gerundet.

Die freien Stellen vor einer Zahl können mit Sternchen gekennzeichnet werden, indem in der Formatschablone "**" angegeben wird:

```
PRINT USING "**#####.#";12.22
****12.2
```

Die Sternchen können mit den Währungszeichen kombiniert werden, wobei letztere in der Formatschablone nur einfach angegeben werden: "**$..." oder "**f...".

Ein "+" am Beginn der Formatschablone bedeutet, daß das Vorzeichen (+ oder -) immer vor die Zahl gesetzt wird. Steht ein "+" am Ende der Schablone, so wird das Vorzeichen nachgestellt.

Ein "-" kann nur am Ende der Schablone erscheinen. Es bedeutet, daß ein Minus-Zeichen nachgestellt wird, falls der Zahlenwert negativ ist.

```
PRINT USING "+##";12
+12

PRINT USING "+##";-12
-12

PRINT USING "##+";12
12+

PRINT USING "##-";-12
12-

PRINT USING "##-";12
12
```

Durch Einbeziehung von "↑↑↑↑" kann eine Zahl in Exponential-Schreibweise wiedergegeben werden:

```
PRINT USING "###.##↑↑↑↑";123.45
 12.35E+Ø1
```

Wenn eine Zahl zu lang für eine definierte Formatschablone ist, erscheint vor dem Ergebnis ein %, um diesen Fall anzuzeigen. Die Zahl wird NICHT gekürzt, um in die Schablone zu passen.

```
PRINT USING "####";123456
%123456
```

Hätten sie gerne Fenster?

Das BASIC des CPC6128 bietet die Möglichkeit, den Bildschirm in bis zu acht Bereiche, sogenannte Textfenster oder 'Windows', einzuteilen. Jeder Befehl für den Text-Bildschirm kann dann an ein beliebiges Fenster gerichtet werden.

Der Befehl zur Einrichtung eines Fensters heißt `WINDOW` (engl. für Fenster). Dazu werden fünf Werte angegeben. Der erste gibt dem Fenster eine Nummer. Wird der Wert ausgelassen, erhält das Fenster automatisch die Nummer 0. Alle BASIC-Meldungen (z.B. `Ready`) erscheinen grundsätzlich in Fenster 0. Der Fenster-Nummer wird das Kreuz # vorangestellt, um es als Bildschirmbereich bzw. Ausgabegerät (Stream) auszuweisen. Die nächsten vier Werte hinter dem Befehl `WINDOW` geben die linke, rechte, obere und untere Begrenzung des Fensters an. Diese Werte sind Spalten- und Zeilen-Nummern; sie müssen also für links/rechts zwischen 1 und 80 und für oben/unten zwischen 1 und 25 liegen.

Das folgende Beispiel definiert ein Fenster mit der Nummer 4, das zwischen der 7. und 31. Spalte sowie zwischen der 6. und 18. Zeile liegen soll. Setzen Sie den Computer zurück und schreiben Sie:

```
WINDOW #4,7,31,6,18
```

Es passiert scheinbar nichts nach diesem Befehl, aber schreiben Sie jetzt folgendes:

```
INK 3,9
PAPER #4,3
CLS #4
```

Damit erscheint ein großes grünes Rechteck auf dem Bildschirm: das Fenster Nummer 4. Aus obigem Programm geht auch hervor, daß die Befehle `PAPER` und `CLS` für jedes einzelne der acht Fenster verwendet werden können, sofern eine Stream-Nummer angegeben wird. Wird keine Stream-Nummer genannt, so wird der Befehl automatisch an Fenster 0 gerichtet.

Folgende Befehle können mit einer Stream-Nummer versehen werden, um anzuzeigen, für welches Fenster der Befehl gelten soll:

```
CLS
COPYCHR$
INPUT
LINE INPUT
LIST
LOCATE
PAPER
PEN
POS
PRINT
TAG
TAGOFF
VPOS
WINDOW
WRITE
```

Das grüne Fenster, das Sie auf den Bildschirm gezaubert haben, wird einen Teil des Textes verdecken, der vorher in Fenster 0 geschrieben wurde.

Ein Text kann in ein bestehendes Fenster geschrieben werden, wenn in der PRINT-Anweisung die entsprechende Stream-Nummer angegeben wird:

```
PRINT #4,"halli hallo"
```

Diese Worte erscheinen jetzt oben im grünen Rechteck, und nicht unter der Anweisung, wie es im Falle von:

```
PRINT "halli hallo"
```

...geschehen wäre. Bei der ersten Version des Befehls haben Sie gemerkt, daß ein Teil der grünen Fläche von der Schrift überdeckt wurde.

Wenn Sie alle BASIC-Meldungen in Fenster 4 erscheinen lassen wollen, können Sie es mit dem Befehl **WINDOW SWAP** anstelle von Fenster **Ø** zum Standard-Fenster machen:

```
WINDOW SWAP Ø,4
```

Das `Ready` nach diesem Befehl erscheint dann im grünen Fenster, und der Cursor direkt darunter. Schreiben Sie jetzt:

```
PRINT #4,"halli hallo"
```

...und Sie sehen, daß die Worte halli hallo nun unter der WINDOW SWAP-Anweisung im früheren Fenster Ø, das jetzt aber die Nummer 4 trägt, erscheint. Daran wird übrigens auch deutlich, daß sich der Computer für jedes Fenster die Stelle merkt, wo der Text aufhört, so daß auch nach einem Fenster-Tausch der neue Text weiter unten in Fenster 4, und nicht in der ersten Zeile erscheint. Schreiben Sie nun:

```
LOCATE #4,2Ø,1
PRINT "Dies ist Fenster Ø"
PRINT #4,"Dies ist Fenster 4"
```

Die 'Fenster Ø'-Meldung steht nun in der Zeile unter PRINT, während die 'Fenster 4'-Meldung in der Mitte der ersten Zeile des ganzen Bildschirms erscheint.

Bevor ein WINDOW-Befehl ausgegeben wird, bedecken alle acht Fenster den gesamten Bildschirm. Dies gilt auch nach einem MODE-Kommando. Wenn Sie also Fenster benutzt haben, und der Cursor in einem sehr kleinen Fenster bleibt, tippen Sie einfach MODE 1 ein:

```
MODE 1
WINDOW 2Ø,21,7,18
MO
DE
 1
```

Es macht nichts, daß MODE über zwei Zeilen veteilt ist; vergessen Sie aber nicht die Leerstelle zwischen MODE und 1.

Da Sie nun über Fenster ein wenig Bescheid wissen, probieren Sie das folgende kleine Programm aus:

```
1Ø MODE Ø
2Ø FOR n=Ø TO 7
3Ø WINDOW #n,n+1,n+6,n+1,n+6
4Ø PAPER #n,n+4
5Ø CLS #n
6Ø FOR c=1 TO 2ØØ:NEXT c
77Ø NEXT n
```

Sie erhalten acht sich überlappende Fenster in verschiedenen Farben. Wenn die Durchführung des Programms beendet ist, und Ready erscheint, drücken Sie einige Male auf die **[RETURN]**-Taste. Sie sehen, wie das Durchrollen von Fenster Ø die bunten Blöcke auf dem Bildschirm beeinflußt. Obwohl diese Blöcke sich jedoch verschieben, verändert sich die Stellung der anderen Fenster nicht. Probieren Sie folgendes:

```
CLS #4
```

...und Sie sehen, daß sich das 4. Fenster noch an seiner alten Stelle befindet und nun die darunterliegenden Fenster verdeckt. Sehen Sie sich die unterschiedlichen Effekte an, wenn Sie folgende Befehle eingeben:

```
LIST
LIST #4
LIST #3
```

Eine weitere Eigenschaft des Befehls **WINDOW** wird im letzten Beispiel dieses Teils demonstriert: Es ist gleichgültig, ob Sie erst die linke oder die rechte Begrenzung des Fensters angeben, da BASIC automatisch erkennt, daß es sich bei dem größeren Wert um die rechte Begrenzung handelt. Dasselbe gilt für die Parameter des oberen und unteren Fenster-Randes.

```
10 MODE 0
20 a=1+RND*19:b=1+RND*19
30 c=1+RND*24:d=1+RND*24
40 e=RND*15
50 WINDOW a,b,c,d
60 PAPER e:CLS
70 GOTO 20
```

Darf ich mal unterbrechen?

Falls Sie es noch nicht bemerkt haben sollten: Eine wichtige Neuheit in der Software der AMSTRAD-Computer ist ihre einzigartige Fähigkeit, mit Unterbrechungen in BASIC-Programmen zu arbeiten. Dies bedeutet, daß das AMSTRAD-BASIC in der Lage ist, unterschiedliche Operationen eines Programms gleichzeitig durchzuführen. Diese Fähigkeit wird manchmal als 'Multitasking' bezeichnet und wird durch die Befehle **AFTER** und **EVERY** hervorgerufen.

Diese Eigenschaft tritt auch bei der Tonerzeugung in Form von Warteschlangen und Rendezvous-Vorgängen zutage.

Alle zeitabhängigen Vorgänge werden von einer zentralen Uhr gesteuert, einem quarzbetriebenen Zeitgeber-System im Computer, das den zeitlichen Ablauf und die Synchronisation der Vorgänge im Computer überwacht, wie z.B. die Abtastfrequenz und die Vorgänge im Prozessor. Jede zeitabhängige Hardware-Funktion wird von der zentralen Quarzuhr gesteuert.

Auf der Software-Seite haben wir die Kommandos **AFTER** und **EVERY**. Wie beim AMSTRAD-BASIC üblich, sind die Kommandos benutzerfreundlich und sagen auch, was sie tun. Die **AFTER**-Anweisung bewirkt, daß nach der angegebenen Zeit in das gewünschte Unterprogramm gesprungen wird, um die darin enthaltenen Anweisungen auszuführen.

Der CPC6128 besitzt eine Echtzeituhr. Das AFTER-Kommando ermöglicht, daß Unterprogramme erst nach einer bestimmten Zeit ausgeführt werden. Vier unabhängige Zeitgeber stehen zur Verfügung, die je mit einem eigenen Unterprogramm verbunden sein können.

Sobald die vorgegebene Zeit verstrichen ist, wird das Unterprogramm automatisch aufgerufen, als hätte an dieser Stelle im Programm ein GOSUB-Kommando gestanden. Am Ende des Unterprogramms, das mit einem normalen RETURN-Kommando abschließt, kehrt das Programm zu der Stelle zurück, wo es vorher unterbrochen worden war.

Mit dem Befehl EVERY kann innerhalb eines BASIC-Programms in regelmäßigen Abständen ein Unterprogramm wiederholt aufgerufen werden. Auch hierfür stehen vier Zeitgeber zur Verfügung, die je mit einem eigenen Unterprogramm verbunden sein können.

Die Zeitgeber haben verschiedene Unterbrechungsprioritäten. Zeitgeber 3 hat die höchste, Zeitgeber 0 die niedrigste Priorität. (Siehe auch Kapitel 'Übersicht')

```
10 MODE 1:n=14:x=RND*400
20 AFTER x,3 GOSUB 80
30 EVERY 25,2 GOSUB 160
40 EVERY 10,1 GOSUB 170
50 PRINT "Reaktionstest"
60 PRINT "Druecke die Leertaste.";
70 IF Flagge=1 THEN END ELSE 70
80 z=REMAIN(2)
90 IF INKEY(47)=-1 THEN 110
100 SOUND 1,900:PRINT "geschummelt!":GOTO 150
110 SOUND 129,20:PRINT "JETZT":t=TIME
120 IF INKEY(47)=-1 THEN 120
130 PRINT "Du hast";
140 PRINT (TIME-t)/300;"Sekunden gebraucht."
150 CLEAR INPUT:Flagge=1:RETURN
160 SOUND 1,0,50:PRINT ".";:RETURN
170 n=n+1:IF n>26 THEN n=14
180 INK 1,n:RETURN
```

AFTER- und EVERY-Befehle sind überall einsetzbar, wobei für den jeweiligen Zeitgeber immer andere Unterprogramme und Zeiten angegeben werden können. AFTER und EVERY müssen sich die Zeitgeber teilen, so daß ein AFTER-Kommando ein vorheriges EVERY-Kommando für einen bestimmten Zeitgeber überschreibt und umgekehrt.

Mit den Befehlen DI und EI werden Zeitgeber-Unterbrechungen außer Kraft bzw. wieder in Kraft gesetzt, während die dazwischenliegenden Befehle ausgeführt werden. Dadurch wird verhindert, daß ein Unterbrechungsprogramm höherer Priorität einsetzt, während noch eine Unterbrechung niedrigerer Priorität bearbeitet wird.

Die Funktion REMAIN setzt einen der vier Zeitgeber außer Kraft und gibt die verbleibende Restzeit an.

READ und DATA

Für ein Programm, das in seiner Startphase immer mit demselben Satz an Informationen gefüttert werden muß, wäre es sinnvoller, wenn man diese Informationen eingeben könnte, ohne den Benutzer jedesmal zu bitten, sie einzutippen. Diese Möglichkeit ist mit den Befehlen READ und DATA gegeben. Ähnlich wie INPUT dient der Befehl READ dazu, Variablen einen Wert zuzuordnen. Er unterscheidet sich jedoch dadurch, daß Werte von DATA-Anweisungen gelesen werden, und nicht über die Tastatur eingegeben werden müssen. Das folgende Beispiel mag dies verdeutlichen:

```
10 INPUT "Gib 3 durch Komma getrennte Zahlen ein";a,b,c
20 PRINT "Die Zahlen heissen";a;"und";b;"und";c
run
10 READ a,b,c
20 PRINT "Die Zahlen heissen";a;"und";b;"und";c
30 DATA 12,14,21
run
```

Genauso wie beim INPUT-Befehl werden die Angaben in der DATA-Anweisung durch Kommata getrennt.

Außer numerischen Werten kann eine DATA-Anweisung auch konstante Textfolgen (Strings) enthalten:

```
10 DIM a$(9)
20 FOR i=1 TO 9
30 READ a$(i)
40 NEXT
50 FOR i=0 TO 9
60 PRINT a$(i);" ";
70 NEXT
80 DATA Der,flinke,rote,Fuchs,springt,ueber,den,faulen
   Hund
```

Wie Sie sehen, sind die Textfolgen in der DATA-Anweisung nicht in Anführungszeichen " " eingeschlossen. Anführungszeichen zur Trennung der einzelnen Strings können in DATA-Anweisungen angegeben oder weggelassen werden, genauso wie bei Strings, die Sie als Anwort auf eine INPUT-Anweisung eintippen. In einem Fall sind Anführungszeichen in DATA-Anweisungen jedoch hilfreich, nämlich dann, wenn der String selbst Kommas enhält. Ist der String unter solchen Umständen nicht durch Anführungszeichen gekennzeichnet, wird READ das Komma in diesem String als Trennungszeichen zwischen verschiedenen Strings interpretieren.

```
10 READ a$
20 WHILE a$<>"*"
30 PRINT a$
40 READ a$
50 WEND
60 DATA Das alte,verfallene,verrottete Haus knarrte im
   Wind
70 DATA "Der grosse,schlanke,dunkle Mann hustete verne
   hmlich."
80 DATA *
```

Da der String in Zeile 60 Kommas enthält, wird jeder Teil einzeln durch den Befehl READ gelesen und gedruckt. Der String in Zeile 70 wird, da er in Anführungszeichen eingeschlossen ist, wie beabsichtigt als Ganzes gedruckt.

An diesem Beispiel wird klar, daß DATA-Anweisungen in mehreren Zeilen vorkommen können. READ geht die Zeilen der Reihe nach durch (60, 70, 80 usw.). Eine weitere, vielleicht nicht so offensichtliche Tatsache ist, daß DATA-Anweisungen an beliebiger Stelle im Programm auftreten können: vor oder nach der READ-Anweisung, die die Information aufnimmt.

Hat ein Programm mehr als eine READ-Anweisung, macht das zweite READ an der Stelle weiter, wo das erste aufgehört hat:

```
10 DATA 123,456,789,321,654,2343
20 FOR i=1 TO 5
30 READ num
40 total=total+num
50 NEXT
60 READ total2
70 IF total=total2 THEN PRINT "Die Daten stimmen" ELSE
   PRINT "Die Daten sind fehlerhaft"
```

Verändern Sie Zeile 10, indem Sie eine der ersten 5 Zahlen verfälschen, und lassen Sie das Programm erneut laufen. Diese Methode, am Ende der DATA-Anweisung einen Wert anzugeben, der die Summe der vorangehenden Daten bildet, ist eine wirksame Möglichkeit, Fehler beim Eintippen von Daten aufzuspüren, besonders, wenn es sich um sehr lange DATA-Zeilen handelt. Der letzte Wert wird auch 'Kontrollsumme' genannt.

Wenn ein Programm gemischte Daten enthalten soll, d.h. sowohl Strings wie Zahlen, ist es zulässig, in **READ**- und **DATA**-Anweisungen Elemente beider Kategorien anzugeben, solange sie richtig gelesen werden. Wenn z.B. in einer **DATA**-Anweisung jeweils zwei Zahlen, gefolgt von einem String, auftreten, ist das Ergebnis nur sinnvoll, wenn die **READ**-Anweisung zwei Zahlenvariablen und eine Stringvariable enthält:

```
10 DIM a(5),b(5),s$(5)
20 FOR i=1 TO 5
30 READ a(i),b(i),s$(i)
40 NEXT
50 DATA 1,7,Fred,3,9,Jim,2,2,Erich,4,6,Peter,9,1,Alfons
60 FOR i=1 TO 5
70 PRINT s$(i),":";a(i)*b(i)
80 NEXT
```

Andererseits haben Sie die Möglichkeit, die verschiedenen Datentypen getrennt zu behandeln:

```
10 DIM a(5),b(5),s$(5)
20 FOR i=1 TO 5
30 READ a(i),b(i)
40 NEXT
50 FOR i=1 TO 5
60 READ s$(i)
70 NEXT
80 DATA 1,7,3,9,2,2,4,6,9,1
90 DATA Fred,Jim,Erich,Peter,Alfons
100 FOR i=1 TO 5
110 PRINT s$(i),":";a(i)*b(i)
120 NEXT
```

Wenn Sie die FOR-Schleife in Zeile 20 nun folgendermaßen ändern:

```
20 FOR i=1 TO 4
```

...kommt bei den ersten beiden Leseversuchen '9' und '1' heraus. Dies sind natürlich gültige Strings, aber so war das Ergebnis nicht geplant! Wir haben die Möglichkeit, das Programm richtig laufen zu lassen, indem wir zwei Zeilen einfügen:

```
15 RESTORE 80
45 RESTORE 90
```

Der RESTORE-Befehl setzt den DATA-Lese'zeiger' in die angegebene Zeile. RESTORE ist daher für IF...THEN-Anweisungen geeignet, damit ein bestimmter Datenblock in Abhängigkeit von einem Kriterium ausgewählt und gelesen wird. Wenn Sie z.B. ein mehrstufiges Programm mit verschiedenen Bildschirmdarstellungen haben, kann jeder Bildschirm mit einer bestimmten Variablen, z.B. 'stufe' ausgewählt werden. Zur Illustration zeigen wir hier einen entsprechenden Abschnitt aus einem Programm:

```
1000 REM Teil, der den Bildschirm zeichnet
1010 IF stufe=1 THEN RESTORE 2010
1020 IF stufe=2 THEN RESTORE 2510
1030 IF stufe=3 THEN RESTORE 3010
1040 FOR y=1 TO 25
1050 FOR x=1 TO 40
1060 READ zeichen
1070 locate x,y: PRINT CHR$(zeichen);
1080 NEXT x,y
:
2000 REM Daten fuer Bildschirm 1
2010 DATA 200,190,244,244,210,...usw.
#:
2500 REM Daten fuer Bildschirm 2
2510 DATA 100,103,245,243,251,...usw.
:
3000 REM Daten fuer Bildschirm 3
3010 DATA 190,191,192,193,194,...usw.
```

Ein weiteres Anwendungsbeispiel für DATA, READ und RESTORE ist ein Programm für eine Melodie. Dabei liest READ die Werte der Tonperioden aus der DATA-Anweisung, und RESTORE dient dazu, einen Teil der Melodie zu wiederholen, indem es den Zeiger immer an den Anfang eines bestimmten Datenelementes setzt:

```
10 FOR i=1 TO 3
20 RESTORE 100
30 READ note
40 WHILE note <>-1
50 SOUND 1,note,35
60 READ note
70 WEND
80 NEXT
90 SOUND 1,142,100
100 DATA 95,95,142,127,119,106
110 DATA 95,95,119,95,95,119,95
120 DATA 95,142,119,142,179,119
130 DATA 142,142,106,119,127,-1
```

Der Ton macht die Musik

Von allen Möglichkeiten des CPC6128 wirken die SOUND- und Hüllkurven-Befehle auf den ersten Blick am undurchdringlichsten - aber haben Sie keine Befürchtungen. Mit etwas Übung sollten Sie bald in der Lage sein, eine Vielzahl verschiedener Töne zu erzeugen, oder gar eine ganze Melodie mit Harmonien zu spielen.

Sehen wir uns zu Beginn noch einmal die vier Teile des SOUND-Befehls an. Es sind: Kanalnummer, Tonperiode, Dauer der Note und Lautstärke. Sie werden sich jetzt fragen, in welcher Form diese Angaben erscheinen.

Wir lassen den ersten Teil (Kanalnummer) zunächst beiseite, da seine Beschreibung ziemlich kompliziert ist. Für den zweiten Teil (Tonperiode) kann im Prinzip jede ganze Zahl zwischen 0 und 4095 angegeben werden, aber nur wenige davon erzeugen tatsächlich Töne, die als Noten erkennbar sind. Diese Tonperioden sind im 5. Teil des Kapitels 'Übersicht' aufgeführt. Nummer 239 z.B. erzeugt das eingestrichene C, und 253 die Note H, eine halbe Stufe unter dem C. Die Werte 240 bis 252 erzeugen zwar ebenfalls Töne, doch sind diese nicht im Tonumfang eines Klaviers enthalten. Tonperiode 0 ergibt überhaupt keinen Ton - dies erweist sich für den Einsatz von Geräuschen als vorteilhaft, wie wir später noch sehen werden.

Für den dritten Teil des SOUND-Befehls wird ein Wert angegeben, der die Länge des Tones in Hundertstelsekunden mißt. Der Wert liegt im allgemeinen zwischen 1 und 32767. Ist der Wert jedoch 0, wird die Länge des Tones durch die Hüllkurve bestimmt. (Dazu später mehr.) Ein negativer Wert gibt an, wie oft die Hüllkurve wiederholt werden soll. -3 bedeutet also eine dreimalige Wiederholung der Hüllkurve.

Die Werte für den vierten Teil, der die Lautstärke bestimmt, können zwischen 0 und 15 liegen. Werden keine Lautstärke-Werte angegeben, nimmt der Computer automatisch 12 an. Bei den einfachen Tönen, die bisher erzeugt wurden, blieb die Lautstärke während ihrer ganzen Länge konstant. Wenn sie mit einer 'Lautstärken-Hüllkurve' variiert werden, so bildet der Wert für 'Lautstärke' im SOUND-Befehl die Ausgangslautstärke.

Kommen wir nun zum ersten Teil des Befehls: Kanalnummer. Der hierfür angegebene Wert ist Bit-signifikant; um das zu verstehen, müßten Sie sich in den ersten Teil dieses Kapitels über das Binärsystem vertiefen, sofern Sie es noch nicht getan haben.

Ein Ton kann über einen der drei möglichen Kanäle ausgegeben werden. Ist der Computer an einen Stereo-Verstärker angeschlossen, läuft ein Kanal über den linken, der andere über den rechten Lautsprecher, und der dritte über beide (der Ton scheint dann aus der Mitte zu kommen). Um zu bestimmen, über welchen Kanal der Ton bzw. die Töne kommen soll(en), werden folgende Nummern eingesetzt:

1 für Kanal A
2 für Kanal B
4 für Kanal C

Um mehr als einen Kanal zu benutzen, können die Nummern für die gewünschten Kanäle addiert werden. Um einen Ton über A und C zu erzeugen, wird also 1+4=5 eingesetzt:

```
SOUND 5,284
```

Sie werden sich wundern, daß Kanal C die Nummer 4 und nicht 3 hat. Das liegt daran, daß alle Nummern für die Kanäle Potenzen von 2 sind (1=2 ↑ 0, 2=2 ↑ 1, 4=2 ↑ 2), so daß in der Kombination eine Binärzahl entsteht. Wenn Sie sich eine dreistellige Binärzahl denken, ist jede Ziffer entweder 0 oder 1, und dieses Schema wird bei der Kanalnummer benutzt, um anzuzeigen, ob der zugehörige Kanal an- oder abgeschaltet sein soll.

Dezimal 5 entspricht `1*4+0*2+1*1 = 101` in der Binärschreibweise. Durch die Benennung jeder Spalte mit C, B und A ergibt sich folgendes Bild:

```
C  B  A
1  0  1
```

Das heißt, Kanal C ist angeschaltet, B ist aus- und A ist angeschaltet. Sollte der Ton über die Kanäle A und B ausgegeben werden müßte es so aussehen:

```
C  B  A
0  1  1
```

Die Binärzahl 011 entspricht aber `0*4+1*2+1*1 = 3`, so daß der **SOUND**-Befehl lauten müßte:

```
SOUND 3,142
```

Dies ist natürlich derselbe Wert, der sich ergibt, wenn die Nummern der Kanäle, die benutzt werden sollen, addiert werden (A=1, B=2, C=4). Um einen Ton über A und B zu schicken, muß als Kanalnummer 1+2=3 angegeben werden.

Falls Sie nicht ganz mitkommen, seien Sie unbesorgt. Eigentlich genügt es, zu wissen, daß sich eine Kombination von Kanälen durch die Addition der betreffenden Kanalnummern erreichen läßt.

Leider gibt es jedoch noch andere Werte, die für 'Kanalnummer' eingesetzt werden können. Die Werte 8, 16 und 32 werden benutzt, um anzugeben, daß ein Ton sich mit einem anderen Kanal zum 'Rendezvous' treffen soll. Wahrscheinlich wundern Sie sich über den Begriff 'Rendezvous'. Bis jetzt haben wir die erzeugten Töne direkt zu den gewünschten Kanälen geschickt. Probieren Sie folgendes aus:

```
SOUND 1,142,2000
SOUND 1,90,200
```

Sofern Sie nicht sehr langsam getippt haben, werden Sie bemerkt haben, daß Sie den zweiten Befehl eingeben konnten, bevor der erste fertig ausgeführt war. Dies war möglich, weil das System in der Lage ist, für jeden Kanal bis zu fünf SOUND-Befehle in einer sogenannten 'Warteschlange' zu speichern. Wenn wir einen Ton auf Kanal A und danach einen Ton gleichzeitig auf A und B spielen wollten, müßten wir den Computer irgendwie wissen lassen, daß der Ton auf B nicht beginnen soll, bevor der erste Ton auf A fertig ist, d.h. die Kanäle müssen aufeinander warten. Dies wird 'Rendezvous' genannt, und kann auf zweierlei Weise hervorgerufen werden:

```
SOUND 1,200,1000
SOUND 3,90,200
```

Der zweite Ton wird hierbei nach A und B geschickt; er kann also nicht beginnen, bevor der Ton auf A geendet hat. Die Grenze dieses Verfahrens, bei dem ein Ton in den Wartezustand gesetzt wird, bis alle erforderlichen Kanäle für ihn frei sind, besteht darin, daß zu jedem dieser Kanäle derselbe Ton geschickt wird. In unserem Beispiel ging `,90,200` sowohl zu A als auch zu B. Die zweite Methode sieht folgendermaßen aus:

```
SOUND 1,200,2000
SOUND 1+16,90,200
SOUND 2+8,140,400
```

Hier trifft sich die zweite Note auf A mit dem Ton auf B (und der Ton auf B trifft sich mit der Note von A). Der Vorteil bei diesem Verfahren liegt auf der Hand: Obwohl sich die zweite Note auf A von der Note auf B unterscheidet, sind die beiden miteinander verbunden, so daß keiner von beiden beginnen kann, bevor beide Kanäle frei sind. Dies nennt man Rendezvous. Die hierfür erforderlichen Werte sind ebenfalls Bit-signifikant:

8=2 ↑ 3, 16=2 ↑ 4, 32=2 ↑ 5

Die Kanalnummer ist also eine Binärzahl mit jeweils folgenden Kanalbezeichnungen:

Rendez-vous C plus 32	Rendez- vous B plus 16	Rendez- vous A plus 8	Ausgabe über C plus 4	Ausgabe über B plus 2	Ausgabe über A plus 1

Soll z.B. eine Note auf Kanal C gespielt werden und sich mit A treffen, muß folgendes angegeben werden:

0	0	1	1	0	0

Dies ist die Binärzahl 1100, die in dezimaler Form 8+4=12 lautet.

Wird also für 'Kanalnummer' der Wert 12 eingesetzt, weiß der Computer, daß er eine Note auf Kanal C spielen und auf eine Note warten soll, die zum Rendezvous mit ihr auf Kanal A vorgesehen ist.

Wenn zur Kanalnummer 64 (2 ↑ 6) hinzuaddiert wird, wird die Note gehalten. Das heißt im wesentlichen, daß der Ton erst nach dem Befehl RELEASE gespielt wird.

Und wird 128 (2 ↑ 7) zur Kanalnummer hinzuaddiert, so wird die Warteschlange des betreffenden Kanals gelöscht. Falls Sie einen Ton starten, der zu lang für einen bestimmten Kanal ist, können Sie ihn schnell stoppen, indem Sie den Kanal löschen:

```
SOUND 1,248,30000          (Der Ton würde 5 Minuten laufen)
SOUND 1+128,0              (Dadurch wird der Ton gestoppt.)
```

Im direkten Befehlsmodus haben Sie eine schnellere Möglichkeit, lange Töne zu stoppen, indem Sie die **[DEL]**-Taste drücken. Nach einem kurzen Warn-Piepston werden alle Tonkanäle gelöscht.

Da wir nun hoffentlich in der Lage sind, einen Ton nach Belieben zu einem der drei Kanäle zu schicken (wenn nötig, mit Rendezvous), wäre es doch schön, wenn wir etwas mehr produzieren könnten als ein ziemlich langweiliges 'Pieps', das der einfache SOUND-Befehl hervorruft. Diese Möglichkeit haben wir, wenn wir den Ton mit Hilfe einer Hüllkurve während seiner Dauer lauter oder leiser werden lassen. Ein auf einem Musikinstrument erzeugter Ton schwillt zunächst stark und plötzlich an, bleibt dann eine gewisse Zeit auf einem etwas niedrigeren Niveau und klingt dann bis auf 0 ab. Auch die durch SOUND erzeugten Töne können mit einer derartigen Lautstärke-Kurve versehen werden. Dazu benutzt man den Befehl ENV. Sehen wir uns dazu erst ein einfaches Beispiel an:

```
ENV 1,5,3,4,5,-3,8
SOUND 1,142,0,0,1
```

Der Befehl ENV wird im SOUND-Befehl verwendet, und muß daher vor ihm stehen. Die Nummer der in ENV definierten Hüllkurve wird dann an fünfter Stelle im SOUND-Befehl eingesetzt. In unserem Beispiel trägt die Hüllkurve die Nummer 1. Im ENV-Befehl erscheint die Nummer der Hüllkurve an erster Stelle. Weiterhin enthält er Informationen über die Dauer des Tones und seinen Lautstärkeverlauf, so daß im zugehörigen SOUND-Befehl für 'Dauer' und 'Lautstärke' einfach 0 eingesetzt werden kann. Die im obigen Beispiel definierte Hüllkurve hat einen fünfstufigen Lautstärke-Anstieg, wobei jede Stufe um den Lautstärkewert 3 zunimmt und vier Hundertstelsekunden dauert. Danach fällt die Lautstärke in fünf Stufen um jeweils -3 ab, wobei jede Stufe eine Dauer von acht Hundertstelsekunden hat. Beim ENV-Befehl steht mit anderen Worten an

erster Stelle die Nummer der Hüllkurve, die darin definiert wird. Es folgen zwei Gruppen zu jeweils drei Zahlenangaben, sogenannte Hüllkurven-Abschnitte. Die erste Zahl in jedem Abschnitt gibt an, in wieviel Stufen der Ton an- bzw. abschwellen soll, die zweite Zahl bestimmt, um wieviel Lautstärkegrade jede Stufe ansteigen bzw. abfallen soll, und die dritte Zahl zeigt an, wie lange der Ton auf jeder Lautstärkestufe verharren soll.

Die Gesamtdauer jedes Hüllkurvenabschnitts ergibt sich also aus der Multiplikation des ersten Wertes (Anzahl der Stufen) mit dem dritten Wert (Dauer pro Stufe). Die gesamte Lautstärke-Differenz innerhalb eines Hüllkurvenabschnitts errechnet sich aus der Anzahl der Stufen, multipliziert mit dem Anstieg bzw. Abfall der einzelnen Stufen. Die Gesamtlänge einer Hüllkurve mit mehr als einem Abschnitt ist die Summe der Länge der einzelnen Abschnitte.

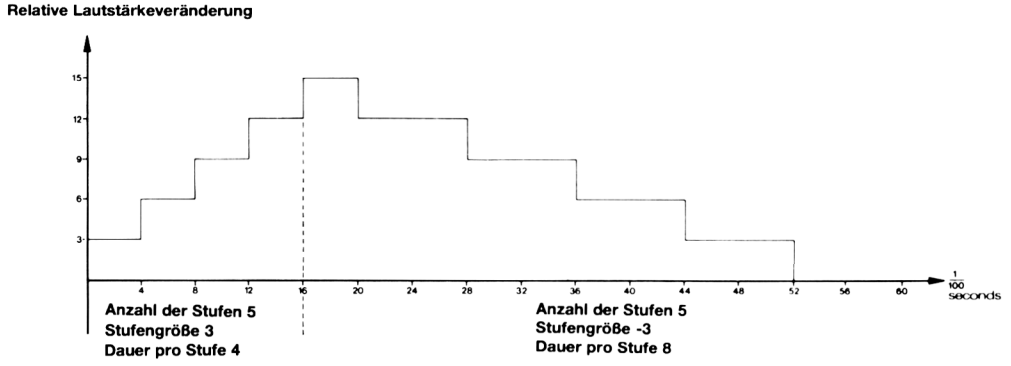
Natürlich braucht die Ausgangslautstärke nicht unbedingt Ø (im SOUND-Befehl) zu sein. Im obigen Beispiel schwoll der Ton erst an und dann ab. Im nächsten Beispiel verhält es sich umgekehrt:

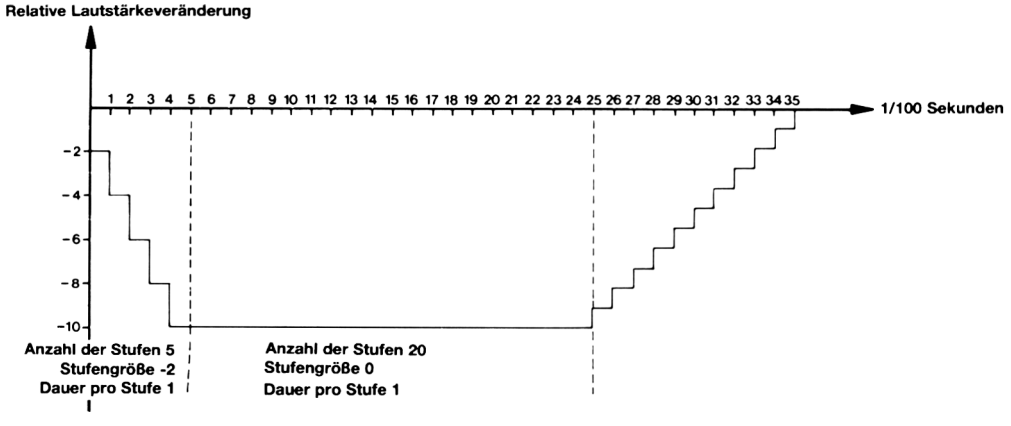
```
ENV 2,5,-2,1,2Ø,Ø,1,1Ø,1,1
SOUND 1,248,Ø,15,2
```

Diese Hüllkurve trägt die Nummer 2 und besteht aus drei Abschnitten. Im ersten wird die Lautstärke in 5 Stufen um jeweils -2 reduziert, d.h. der Abschnitt hat fünf Lautstärkestufen, und mit jeder Stufe nimmt der Ton um zwei Lautstärkegrade ab. Jede einzelne Stufe hat eine Länge von 1 Hunderstelsekunde. Der nächste Abschnitt der Hüllkurve hat 2Ø Stufen, und auf jeder Stufe beträgt der Lautstärkeanstieg bzw. -abfall Ø, d.h. die Lautstärke bleibt konstant. Auch hier dauert jede Stufe 1 Hundertstelsekunde. Der dritte Abschnitt schließlich besteht aus 1Ø Stufen, die um 1 Lautstärkegrad zunehmen und 1 Hunderstelsekunde lang sind.

Im SOUND-Befehl ist eine Anfangslautstärke von 15 angegeben, d.h. nach dem ersten Hüllkurvenabschnitt liegt die Lautstärke bei 5. Auf diesem Niveau bleibt er für 20 Hundertstelsekunden, um im letzten Abschnitt wieder auf 15 anzusteigen.

Es ist vielleicht etwas schwierig, sich die Form solcher Hüllkurven vorzustellen. Es hat sich als hilfreich erwiesen, sie auf Millimeterpapier darzustellen, und daraus die Werte für den ENV-Befehl abzuleiten. Die folgenden beiden Skizzen stellen die eben beschriebenen Hüllkurven bildlich dar.

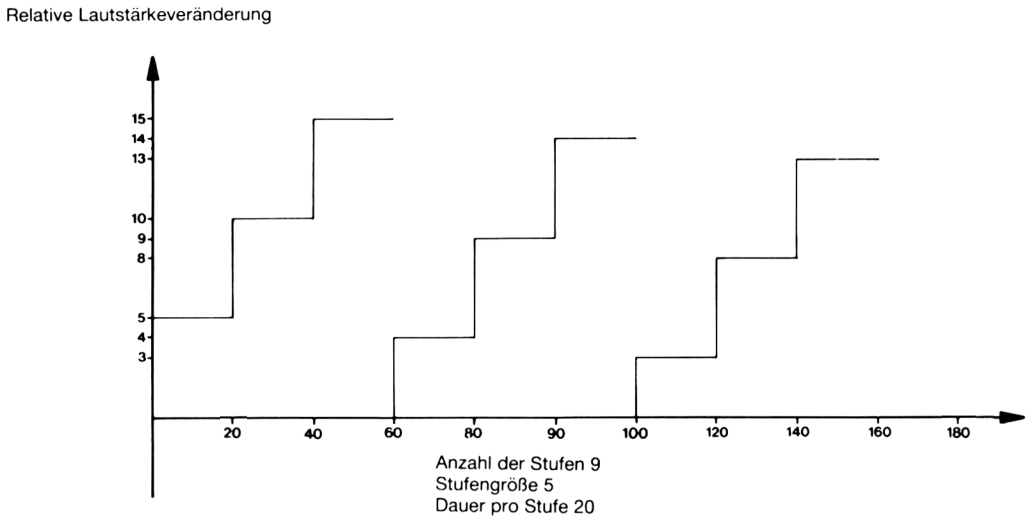




Eine Hüllkurve kann höchstens fünf Abschnitte mit je drei Werten umfassen, d.h. der ENV-Befehl kann aus maximal 16 Teilen bestehen. (Darin ist der erste Wert eingeschlossen, der angibt, um welche der 15 Hüllkurven (1 bis 15) es sich handelt.) Geht die Lautstärke beim Anstieg über 15 hinaus oder beim Abfall unter 0, so springt sie zum entgegengesetzten Lautstärke-Extrem, d.h. der Wert nach 15 ist 0, und der Wert nach 0 ist 15:

```
ENV 3,9,5,20
SOUND 1,142,0,0,3
```

Diese einfache Hüllkurve erzeugt 9 Stufen mit einem Lautstärkeanstieg von je 5; jede Stufe hat eine Dauer von 20 Hundertstelsekunden. Nach den ersten drei Stufen ist die maximale Lautstärke von 15 erreicht. In der nächsten Stufe liegt sie deshalb auf 4, dann auf 9 usw. Die nächste Skizze verdeutlicht diesen Vorgang:



Der Wertebereich für 'Anzahl der Stufen' liegt zwischen 0 und 127; die 'Stufengröße' kann zwischen -128 und +127 variieren, wobei negative Werte einen Lautstärkeabfall bewirken; für 'Dauer pro Stufe' kann ein Wert zwischen 0 und 255 angegeben werden.

Nachdem Sie nun wissen, wie man die Lautstärke eines Tones variiert, möchten wir Ihnen erläutern, wie man den Ton in Schwingungen versetzt, um einen Vibrato-ähnlichen Effekt zu erzielen.

Dazu geht man ähnlich vor wie bei der Definition der Lautstärken-Hüllkurven: Man definiert eine Ton-Hüllkurve mit dem Befehl **ENT**.

Beispiel:

```
ENT 1,5,1,1,5,-1,1
SOUND 1,142,10,15,,1
```

An sechster Stelle im **SOUND**-Befehl steht die Nummer der Ton-Hüllkurve, die zur Erzeugung dieses Tones verwendet werden soll. Daher muß auch die **ENT**-Anweisung vor dem **SOUND**-Befehl erscheinen.

Dieses erste Beispiel für ENT definiert die Ton-Hüllkurve Nr.1. Ihr erster Abschnitt besteht aus 5 Stufen, in denen die Tonperiode um jeweils 1 erhöht wird; jede Stufe dauert 1 Hundertstelsekunde. Für den nächsten Abschnitt sind 5 Stufen angegeben, in denen die Tonperiode um jeweils -1 abnimmt; wieder hat jede Stufe eine Dauer von 1 Hundertstelsekunde. Das ergibt eine Gesamtlänge von 5+5=10. Beachten Sie, daß dieser Wert im SOUND-Befehl angegeben wird, da die Ton-Hüllkurve NICHT die Länge eines Tones bestimmt, in der Weise, wie es die Lautstärken-Hüllkurve tut. Ist die im SOUND-Befehl angegebene Dauer kürzer als die Länge der Ton-Hüllkurve, geht der letzte Teil der Ton-Hüllkurve verloren. Ist die Dauer im SOUND-Befehl umgekehrt länger als die Dauer der Ton-Hüllkurve, so bleibt der Ton für den Rest seiner Dauer auf dem zuletzt erreichten Niveau. Dasselbe trifft zu, wenn eine Lautstärken-Hüllkurve benutzt wird, um die Länge einer Note zu bestimmen.

(Beachten Sie, daß für 'Lautstärken-Hüllkurve' im SOUND-Befehl nichts angegeben wurde, weil wir für diesen Ton keine Lautstärken-Hüllkurve definiert haben.)

In den meisten Fällen wird die Ton-Hüllkurve wahrscheinlich viel kürzer sein als die Länge des Tones. Deshalb kann man veranlassen, daß sich eine Ton-Hüllkurve während der gesamten Dauer eines Tones ständig wiederholt, indem man der Ton-Hüllkurve eine negative Nummer gibt. Im SOUND-Befehl muß diese Nummer aber als positive Zahl angegeben werden:

```
ENT -5,4,1,1,4,-1,1
SOUND 1,142,100,12,,5
```

Dies erzeugt einen schwingenden Ton. Bei der Definition einer Ton-Hüllkurve sollte man im allgemeinen darauf achten, daß der Ton symmetrisch um die anfängliche Tonperiode schwingt, so daß sich durch die Wiederholung der Ton-Hüllkurve die Tonhöhe nicht stetig weiter von der ursprünglichen Frequenz entfernt. Probieren Sie folgendes aus:

```
ENT -6,3,1,1
SOUND 1,142,90,12,,6
```

Sie werden merken, daß die Frequenz dieses Tones rapide abnimmt, weil die Tonperiode mit jeder Wiederholung der Hüllkurve um 3 zunimmt, und das 30 Mal (90/3). Dieser Effekt läßt sich gut zur Erzeugung trillernder und heulender Töne nutzen:

```
ENT -7,20,1,1,20,-1,1
SOUND 1,100,400,12,,7

ENT -8,60,-1,1,60,1,1
SOUND 1,100,480,12,,8
```

Es können 15 verschiedene Ton-Hüllkurven definiert werden (Bereich 1 bis 15), wobei ein negativer Wert angibt, daß die Hüllkurve sich wiederholen soll. Die 'Anzahl der Stufen', ein Wert, der an erster Stelle in jedem Abschnitt genannt wird, kann zwischen 0 und 239 betragen. Wie bei der Lautstärken-Hüllkurve liegt die 'Stufengröße' zwischen -128 und 127, und der Wert für 'Zeit pro Stufe' zwischen 0 und 255. Und wie der ENV-Befehl kann der ENT-Befehl aus bis zu fünf Abschnitten bestehen, die mit jeweils drei Werten definiert werden.

Als siebter und letzter Teil des SOUND-Befehls kann ein Wert angegeben werden, der die Art des Rauschens bestimmt, das den Ton begleiten soll. Wenn Sie ein Rauschen einbauen wollen, müssen Sie daran denken, daß dafür nur ein Kanal zur Verfügung steht, d.h. jeder neue Wert für das Rauschen beendet das vorherige Rauschen.

Das Rauschen kann dazu dienen, den Klang eines Tones abzuwandeln, oder es kann für sich eingesetzt werden. In letzterem Falle muß der zweite Teil des SOUND-Befehls ('Tonperiode') Ø lauten, damit lediglich das Rauschen zu hören ist. Mit dieser Methode lassen sich Schlagzeug-artige Geräusche erzeugen. Hören Sie sich dazu folgendes Beispiel an:

```
ENT -3,2,1,1,2,-1,1
ENV 9,15,1,1,15,-1,1
FOR a=1 TO 1Ø: SOUND 1,4ØØØ,Ø,Ø,9,3,15: NEXT
```

Es klingt ungefähr wie ein fahrender Zug. Sie haben sicherlich bemerkt, daß in diesem Beispiel beide Hüllkurven definiert wurden, und im SOUND-Befehl ein Wert für das Rauschen enthalten ist.

Sowohl für 'Dauer' als auch für 'Lautstärke' wurde im SOUND-Befehl Ø angegeben, da beide Werte schon in der Lautstärke-Hüllkurve enthalten sind.

Da Sie nun hoffentlich die Befehle ENV, ENT und SOUND richtig beherrschen, wollen wir noch verschiedene andere verwandte Kommandos und Funktionen betrachten.

Sie wissen, daß durch die Addition von 64 zum Wert für 'Kanalnummer' der Ton in einen Wartezustand versetzt wird, bis er durch den Befehl RELEASE 'losgelassen' wird. Im Befehl RELEASE wird eine Bit-signifikante Zahl angegeben. Jedes Bit dieser Zahl ist ein Indikator für einen der drei Kanäle. Sie brauchen die Zusammenhänge auch hier nicht bis in alle Einzelheiten zu verstehen; es genügt, wenn Sie folgendes beachten:

4 bedeutet Kanal C
2 bedeutet Kanal B
1 bedeutet Kanal A

Mehrere Kanäle können gleichzeitig freigegeben werden, wenn man die Werte für jeden dieser Kanäle addiert. Um also Töne auf allen drei Kanälen freizugeben, müßte der Befehl:

```
RELEASE 7
```

...lauten, da 1+2+4=7 ergibt. Ist keiner der Kanäle gesperrt, wird RELEASE ignoriert. Probieren Sie folgendes aus:

```
SOUND 1+64,90
SOUND 2+64,140
SOUND 4+64,215
RELEASE 3: FOR t=1 TO 1000: NEXT: RELEASE 4
```

Die Töne, die man nach diesen SOUND-Kommandos erwartet, sind erst zu hören, wenn das erste RELEASE die Kanäle A und B freigibt. Nach einer Verzögerung wird auch Kanal C freigegeben.

Daraus ergibt sich noch eine weitere Rendezvous-Technik für mehrere Töne. Wenn einer Warteschlange, die sich (durch die hinzuaddierten 64) im Haltezustand befindet, ein Ton hinzugefügt wird, befindet nicht nur er sich im Wartezustand, sondern alle weiteren Töne, die in die Schlange eingereiht werden. Sind schließlich mehr als vier Töne in einer blockierten Schlange, pausiert der Computer, bis der Kanal freigegeben wird - etwa durch ein Unterprogramm, das nach einer bestimmten Zeit (mittels AFTER oder EVERY) aufgerufen wird. Dies ist jedoch keine besonders gute Methode, das SOUND-System zu nutzen, da das Programm mit den SOUND-Anweisungen von Zeit zu Zeit pausiert, um die Warteschlangen aufzufüllen. Dasselbe passiert, wenn viele lange Töne direkt aufeinander folgen. Probieren Sie folgendes Programm aus:

```
10 FOR a=1 TO 8
20 SOUND 1,100*a,200
30 NEXT
40 PRINT "hallo"
run
```

Sie merken, daß hallo nicht sofort auf dem Bildschirm erscheint, sondern erst nach etwa drei Sekunden. Dies liegt daran, daß die Programmausführung solange blockiert ist, bis in der Schlange ein Platz frei ist.

BASIC hat einen Unterbrechungsmechanismus, der ähnlich funktioniert wie nach AFTER und EVERY, oder nach ON BREAK GOSUB. Er versetzt Sie in die Lage, ein Ton-erzeugendes Unterprogramm zu benennen, das nur dann aufgerufen wird, wenn in der erforderlichen Schlange ein Platz frei wird. Beispiel:

```
10 a=0
20 ON SQ(1) GOSUB 1000
30 PRINT a;
40 GOTO 30
1000 a=a+10
1010 SOUND 1,a,200
1020 IF a<200 THEN ON SQ(1) GOSUB 1000
1030 RETURN
run
```

Sie werden merken, daß dieses Programm keine Pause macht. Der **SOUND**-Befehl kommt nur zum Zuge, wenn die Warteschlange von Kanal A (Nummer 1) einen freien Platz hat. Der Befehl **ON SQ(1) GOSUB**, deckt diesen Tatbestand auf. Er löst einen Unterbrechungsmechanismus aus, der das in Zeile **1000** beginnende Ton-Unterprogramm aufruft, wenn im angegebenen Kanal ein Platz frei wird. Wurde das Unterprogramm einmal ausgeführt, muß **ON SQ GOSUB** wieder aktiviert werden. Das geschieht mit Zeile **1020** im Unterprogramm. In diesem Beispiel wird das Unterprogramm nur dann wieder aufrufbereit, wenn der Wert für **a** unter 200 liegt.

In einem richtigen Programm, das Objekte auf dem Bildschirm hin- und herbewegt, oder Zahlen addiert, kann im Hintergrund ständig eine Melodie spielen, wenn man ein Unterprogramm aufrufen läßt, das jede Note nur dann spielt, wenn in der Warteschlange ein Platz frei ist. Dadurch wird verhindert, daß das Programm pausiert, um auf einen freien Platz zu warten. Wenn dabei die Notenwerte für die Melodie von einer **DATA**-Anweisung gelesen werden, kann das Ton-Unterprogramm so verfaßt werden, daß kurz vor Ende der Daten der Re-Aktivierungsprozeß gestoppt wird.

Beim Befehl **ON SQ() GOSUB** kann entweder 1, 2 oder 4 in die Klammern gesetzt werden, je nachdem, welche Schlange auf freie Plätze hin untersucht werden soll.

BASIC hat außerdem die Funktion **SQ()**. Sie informiert über einen der drei Tonkanäle. Die Zahl in Klammern, wiederum 1, 2 oder 4, gibt den Kanal an, über dessen Zustand die Information gewünscht wird. Die Funktion gibt eine Bit-signifikante Zahl aus, die Sie nur mit Kenntnissen über das binäre Zahlensystem verstehen können. Die Bits im ausgegebenen Wert haben folgende Bedeutung:

BIT	DEZIMAL	BEDEUTUNG
0,1,2	1,2,4	Anzahl der freien Plätze in der Warteschlange
3	8	Note am Kopf der Schlange ist für Rendezvous mit A vorgesehen
4	16	Note am Kopf der Schlange ist für Rendezvous mit B vorgesehen
5	32	Note am Kopf der Schlange ist für Rendezvous mit C vorgesehen
6	64	Für oberste Note der Schlange ist Halte-Bit gesetzt (Kanal ist blockiert)
7	128	Es wird gerade eine Note gespielt

Lassen Sie dieses einfache Beispielprogramm laufen:

```
10 SOUND 2,200
20 x=SQ(2)
30 PRINT BIN$(x)
run
```

Sie erhalten als Ergebnis die Binärzahl **10000100**, in dem Bit 7 gesetzt ist. Das bedeutet, daß der Ton in dem Augenblick gespielt wurde, als die **SQ**-Funktion bearbeitet wurde. Die letzten drei Stellen **100** ergeben als Dezimalwert 4. Damit wird ausgesagt, daß in der Schlange drei Plätze frei waren. Die Funktion **SQ** wird eingesetzt, um an einer bestimmten Stelle im Programm den Zustand der Warteschlange eines Kanals zu erfahren. Im Gegensatz dazu testet **ON SQ() GOSUB** nicht nur den Zustand einer Warteschlange, sondern springt auch gegebenenfalls in ein Unterprogramm.

In den bisherigen Beispielen wurden nur einzelne Töne erzeugt. Möchte man eine ganze Gruppe von Tönen für eine Melodie erzeugen, so kann man die gewünschten Noten in einer **DATA**-Anweisung auflisten, von wo aus sie mit **READ** in einen **SOUND**-Befehl eingelesen werden:

```
10 FOR Oktave=-1 TO 2
20 FOR x=1 TO 7: REM Noten pro Oktave
30 READ Note
40 SOUND 1,Note/2↑8'Oktave
50 NEXT
60 RESTORE
70 NEXT
80 DATA 426,379,358,319,284,253,239
```

Das letzte Beispielprogramm dieses Abschnitts ist zwar viel komplizierter, basiert aber auf demselben Prinzip. Melodie und Rhythmus spielen auf Kanal A und B, wobei der Takt durch das Rendezvous eingehalten wird. Das Beispiel zeigt eine der Möglichkeiten auf, um DATA so zu strukturieren, daß Informationen über Note, Oktave, Länge und Rendezvous darin enthalten sind:

```
10 REM Zeile 180 erzeugt Melodie im Violin-Schlussel
20 REM Zeile 190 erzeugt Melodie im Bass-Schlussel
30 DIM scale%(12):FOR x%=1 TO 12:READ scale%(x%):NEXT
40 ch1%=1:READ ch1$:ch2%=1:READ ch2$
50 CLS
60 Spd%=12
70 scale$=" a-b b c+c d-e e f+f g+g"
80 ENV 1,2,5,2,8,-1,10,10,0,15
90 ENV 2,2,7,2,12,-1,10,10,0,15
100 ENT -1,1,1,1,2,-1,1,1,1,1
110 DEF FNM$(s$,s)=MID$(s$,s,1)
120 ch1%=1:GOSUB 200
130 ch2%=1:GOSUB 380
140 IF ch1%+ch2%>0 THEN 140
150 END
160 DATA &777,&70c,&6a7,&647,&5ed,&598
170 DATA &547,&4fc,&4b4,&470,&431,&3f4
180 DATA 4cr4f4f1f1g1A1-B2C2f4g2g1A1-B6A2Cr1f1g1f1g1a1-
    b1A1-b2C2g2A2g2f1g1a2g2f6e2c2e2c2g2e2c1-B1A2g2f4e4d
    8c4f3f1c2d4-b2fr2-B2A2g2f6e2gr4C4-B1a1f1-b1g2c2-b4a
    4g4fr6A2A2-B4-B2Ar2-B2A2g2f6e2g4C4-B1A1f1-B1g2C2-B4
    A4g8f.
190 DATA r4f4f8f4e4c4fr8f4e2f2e4d2e2d8c8c6e2f4g4g8e4f3f
    1c4dr8g4cr4e4c6f2d4c4c8fr8-e4dr8g8c4e4c6f2d4c4c8f.
200 REM sendet Ton zum Kanal A
210 p1$=FNM$(ch1$,ch1%)
220 IF p1$<>"r" THEN r1%=0:GOTO 240
230 r1%=16:ch1%=ch1%+1:p1$=FNM$(ch1$,ch1%)
240 IF p1$="." THEN ch1%=0:RETURN ELSE l1%=VAL(p1$)
250 ch1%=ch1%+1
260 n1$=FNM$(ch1$,ch1%)
270 ch1%=ch1%+1
280 IF n1$="+" OR n1$="-" THEN 350
290 n1$=" "+n1$
300 nd1%=(1+INSTR(scale$,LOWER$(n1$)))/2
310 IF ASC(RIGHT$(n1$,1))>96 THEN o1%=8 ELSE o1%=16
320 SOUND 1+r1%,scale%(nd1%)/o1%,Spd%*l1%,0,1,1
330 ON SQ(1) GOSUB 200
340 RETURN
350 n1$=n1$+FNM$(ch1$,ch1%)
360 ch1%=ch1%+1
370 GOTO 300
380 REM sendet Ton zum Kanal B
390 p2$=FNM$(ch2$,ch2%)
```

```
400 IF p2$<>"r" THEN r2%=0:GOTO 420
410 r2%=8:ch2%=ch2%+1:p2$=FNm$(ch2$,ch2%)
420 IF p2$="." THEN ch2%=0:RETURN ELSE l2%=VAL(p2$)
430 ch2%=ch2%+1
440 n2$=FNM$(ch2$,ch2%)
450 ch2%=ch2%+1
460 IF n2$="+" OR n2$="-" THEN 530
470 n2$=" "+n2$
480 nd2%=(1+INSTR(scale$,LOWER$(n2$)))/2
490 IF ASC(RIGHT$(n2$,1))>96 THEN o2%=4 ELSE o2%=8
500 SOUND 2+r2%,scale%(nd2%)/o2%,Spd%*l2%,0,2
510 ON SQ(2) GOSUB 380
520 RETURN
530 n2$=n2$+FNM$(ch2$,ch2%)
540 ch2%=ch2%+1
550 GOTO 480
run
```

Bildlich gesprochen

In diesem Abschnitt werden die graphischen Möglichkeiten des 6128 erläutert. Die wichtigsten Merkmale werden anhand eines Programmbeispiels beschrieben, das nach und nach erweitert wird.

Zu Beginn teilen wir den Bildschirm in ein Text-Fenster (Zeile `40`) und ein Graphik-Fenster (Zeile `30`) ein. Nebenbei bestimmen wir den Modus und zwei blinkende Farben (Zeile `20`).

```
10 REM mask und tag im Fenster
20 MODE 1:INK 2,10,4:INK 3,4,10
30 ORIGIN 440,100,440,640,100,300
40 WINDOW 1,26,1,25
50 CLG 2:GRAPHICS PAPER 0
```

Wenn Sie dieses Programm laufen lassen, sehen Sie auf dem Bildschirm rechts auf halber Höhe ein Quadrat blinken. Es wird durch Zeile `50` mit Farbstift Nr. `2` (magenta/blaugrün) eingefärbt; der Koordinaten-Ursprung wurde mit `ORIGIN` in die linke untere Ecke des Quadrats verlegt und mit dem `MODE`-Kommando wurde der Graphik-Cursor auf den Koordinaten-Ursprung (X=0, Y=0) gelegt, so daß wir mit Zeile 60 eine Diagonale durch das Quadrat ziehen können:

```
60 DRAW 200,200,3
```

Lassen Sie das erweiterte Programm laufen und fügen Sie dann hinzu:

```
80 MOVE 0,2:FILL 3
```

Zeile 80 rückt den Graphik-Cursor in eine der beiden Quadrat-Hälften und füllt sie mit Farbstift 3 aus. Die Grenzen des ausgefüllten Bereiches werden vom Rand des Graphik-Fensters (in diesem Fall gleichbedeutend mit dem Rand des Quadrats) gebildet, sowie von Linien in der Farbe des Graphik-Pens (3) oder in der Farbe, mit der ausgefüllt wird (ebenfalls 3).

Lassen Sie dieses Programm jetzt laufen.

Um die Sache mit der Flächenbegrenzung zu verdeutlichen, fügen Sie Zeile 70 hinzu (unten). Beachten Sie, daß der ausgefüllte Bereich nur deswegen auf die Hälfte des Quadrats begrenzt wird, weil die Ausfüll-Farbe dieselbe Farbe hat wie die Diagonale.

```
70 GRAPHICS PEN 1
run
```

Ändern Sie zum Beweis den FILL-Farbstift in Zeile 80 um in 1, und lassen Sie das Programm laufen. Stellen Sie dann den ursprünglichen Zustand wieder her (FILL 3).

Ergänzen Sie das Programm jetzt um die Zeilen 100 bis 140, mit denen ein Kasten gezeichnet wird:

```
100 MOVE 20,20
110 DRAW 180,20
120 DRAW 180,180
130 DRAW 20,180
140 DRAW 20,20
run
```

Der Kasten wird wegen Zeile 70 mit Farbstift 1 gezeichnet. Hätten wir Zeile 70 ausgelassen, müßten wir entweder den Befehl MOVE in Zeile 100 oder den Befehl DRAW in Zeile 110 um den dritten Parameter ',1' ergänzen, damit der Computer seine Graphik-Pens auswechselt.

Punkt für Punkt...

Linien müssen nicht durchgezogen sein, sie können auch gepunktet (geplottet) werden. Mit dem Befehl MASK kann man die Größe der Punkte bestimmen. Das Punktmuster wiederholt sich alle 8 Pixel, und jede Punktlinie setzt da an, wo die letzte aufgehört hat. Ein neuer MASK-Befehl (der möglicherweise denselben Parameter wie der gegenwärtige hat) setzt ein neues Punkteschema, ohne das alte Muster einzubeziehen.

Das Punktmuster ist einfach eine achtstellige Binärzahl, in der die Ziffer 1 jeweils angibt, daß der Farbstift einen Punkt setzt. In unserem Beispiel wollen wir eine binäre Konstante (gekennzeichnet durch '&X') verwenden, die angibt, daß in der Mitte jeder Gruppe von 8 Pixeln vier Pixel gezeichnet werden sollen. Die beiden Pixel rechts und links davon bleiben unsichtbar. Auf diese Weise erhalten wir eine gestrichelte Linie aus je vier sichtbaren und vier unsichtbaren Pixeln:

```
90 MASK &X00111100
run
```

Aber warten Sie! Die Linie zieht sich nicht gleichmäßig um die Ecken herum. Das liegt daran, daß jeder Eckpunkt zweimal geplottet wird - einmal als letztes Pixel der einen Linie, und zweitens als erstes Pixel der nächsten Linie. Eine umständliche Möglichkeit, diesen Effekt auszuschalten, wäre folgende:

```
115 MOVE 180,22
125 MOVE 178,180
135 MOVE 20,178
run
```

Wir haben jedoch eine einfachere Methode zur Verfügung, indem wir dem MASK-Befehl den Paramter ',0' anhängen. Dadurch weiß der Computer, daß er den ersten Punkt jeder Linie NICHT plotten soll. Ändern Sie also Zeile 90 um in:

```
90 MASK &X00111100,0
```

...und löschen Sie die eben ergänzten Zeilen, indem Sie einfach eintippen:

```
115
125
135
```

Wenn Sie das Programm in dieser Form laufen lassen, erhalten Sie einen symmetrisch gestrichelten Kasten. Ist der zweite Parameter nach MASK ',1', so erhalten Sie übrigens eine durchgezogene Linie, einschließlich des ersten Pixels.

Sehen Sie sich jetzt einmal die Zwischenräume der gestrichelten Linien an. In dem Dreieck unten rechts sehen sie anders aus als im Dreieck oben links. Das kommt daher, weil der Hintergrund für das Dreieck unten rechts durch den Befehl CLG 2 in Zeile 50 mit Farbstift 2 eingefärbt wurde, während im Dreieck oben links die Hintergrundfarbe mit der Bildschirmfarbe übereinstimmt. Ändern Sie Zeile 50 um:

```
50 clg 2:graphics paper 0
```

...und lassen Sie das Programm wieder laufen. Nun ist die Hintergrundfarbe für den Kasten einheitlich.

Wir haben die Möglichkeit, den Hintergrund für das Graphik-Fenster unsichtbar, oder mit dem Fachwort 'transparent', zu machen. Dies bedeutet, daß die Zwischenräume einer gepunkteten Linie, die über eine bestehende Abbildung gezogen wird, erhalten bleiben. Die bestehende Abbildung wird an diesen Stellen unsichtbar gemacht, indem wir an die `GRAPHICS PEN`-Anweisung '`,1`' anhängen. (Mit '`,0`' wird der nicht-transparente Zustand wieder hergestellt.) Ändern Sie also Zeile `70` folgendermaßen um:

```
70 GRAPHICS PEN 1,1
```

...und sehen Sie sich das Ergebnis an.

Von der Position des Graphik-Cursors können nicht nur durchgezogene und gepunktete Linien, sondern auch normale Textzeichen ausgehen. Dies hat den Vorteil, daß man die Textzeichen viel genauer plazieren, d.h. Pixel-weise statt 8-Pixel-weise verschieben kann. Zum anderen lassen sich dadurch die Graphik-Farbstift-Modi (siehe unten) auch auf Textzeichen anwenden.

Um Zeichen an die Position des Graphik-Cursors zu setzen, schieben Sie den Graphik-Cursor an die Stelle, wo der linke obere Rand des Zeichens erscheinen soll. Dann geben Sie den Befehl `TAG` (oder `TAG#1` usw. für andere Text-Fenster) ein, und danach die üblichen `PRINT`-Anweisungen. Der Graphik-Cursor rückt nun automatisch nach jedem Zeichen um 8 Pixel nach rechts. Fügen Sie hinzu:

```
160 MOVE 64,108
170 TAG
180 PRINT "Peter"
190 TAGOFF
run
```

(BASIC gibt seine Meldungen immer auf den Text-Bildschirm aus, gleichgültig, ob auf `TAG` oder `TAGOFF` geschaltet ist. Trotzdem ist es zu empfehlen, `TAG` nach Beendigung seiner Tätigkeit auszuschalten.)

Was haben aber die Pfeile nach `Peter` zu bedeuten? Nun, dies sind Symbole für Wagenrücklauf `CHR$(13)` und Zeilenvorschub `CHR$(10)`. Die Software für die Graphik übersetzt die ersten 32 ASCII-Zeichen in ihre druckbare Version (als würde ihnen immer `CHR$(1);` vorangestellt, wenn sie zum Text-Bildschirm geschickt werden). Dies hat seinen Grund darin, daß die meisten der ersten 32 Steuer-Zeichen nur für den Text-Bildschirm von Bedeutung sind. Deshalb wird auch nicht in der folgenden Zeile weitergemacht, wenn wir über den rechten Rand des Graphik-Fensters hinausgehen.

Diese Pfeile können wir verhindern, wenn wir die PRINT-Anweisung einfach mit einem Semikolon abschließen:

```
180 PRINT "Peter";
run
```

Für einen Text, der unter Verwendung von TAG auf dem Graphik-Bildschirm geschrieben wird, gelten dieselben GRAPHICS PEN-Befehle wie für das Zeichnen von Linien. Der Name, den wir gerade auf den Graphik-Bildschirm ausgeben ließen, wurde deshalb mit GRAPHICS PEN 1 geschrieben und ist transparent. Mit dem Befehl:

```
150 GRAPHICS PEN 1,0
run
```

...machen wir das 'Papier' nicht-transparent, während durch:

```
150 GRAPHICS PEN 0,1
```

...mit Farbstift 0 im Transparent-Modus geschrieben wird.

Löschen Sie Zeile 150 wieder und lassen Sie das Programm noch einmal laufen. Der in Zeile 70 angegebene Zustand (Farbstift 1 und Transparent-Modus für den GRAPHICS PEN) ist nun wieder hergestellt.

Transparente Schrift

Sie können auch auf dem Text-Bildschirm transparente Zeichen darstellen, indem Sie einen der Steuer-Codes verwenden. Fügen Sie hinzu:

```
200 PRINT #2,CHR$(22);CHR$(1)
210 LOCATE #2,32,14:PRINT #2,"******"
220 LOCATE #2,32,14:PRINT #2,"------"
230 PRINT #2,CHR$(22);CHR$(0)
run
```

Zeile 200 setzt den Bildschirmbereich #2 in den Transparent-Modus. Beachten Sie, wie die Sternchen unterstrichen werden. Dies zeigt, daß man Zeichen übereinander lagern kann, sogar in verschiedenen Farben. Zeile 230 schaltet wieder auf Nicht-Transparent-Modus um.

Farbstift-Modi

Es besteht die Möglichkeit, in einem Graphik-Farbstift-Modus zu zeichnen, der das gegenwärtig Gezeichnete mit dem Farbstift, der schon auf dem Bildschirm ist, kominiert. Die Farbe, in der das Pixel letztendlich dargestellt wird, wird errechnet, indem der alte Farbstift für das Pixel mit dem Farbstift für GRAPHICS PEN oder PAPER logisch verknüpft wird. Die logischen Verknüpfungen werden mit XOR, AND und OR hergestellt. Der Farbstift-Modus kann entweder als vierter Parameter in den Befehlen DRAW/DRAWR, PLOT/PLOTR und MOVE/MOVER angegeben werden, oder als Steuer-Code-Folge CHR$(23); CHR$(‹Modus›) in einem PRINT-Befehl. In allen Fällen bewirkt der Wert 1 ein Plotten gemäß XOR, der Wert 2 gemäß AND und der Wert 3 gemäß OR. Der Wert 0 stellt den 'Zwangs'-Modus wieder her, bei dem der gewöhnliche Graphik-Farbstift verwendet wird.

Im nächsten Beispiel wird mit einer XOR-Verknüpfung gearbeitet. XOR wird gerne in der sogenannten Schildkröten-Graphik angewendet, da es die Eigenschaft hat, durch zweimaliges Zeichnen des selben Musters das ursprüngliche Bild wiederherzustellen. Das Unterprogramm, welches das Quadrat zeichnet, wird also zweimal durchgeführt (in Zeile 110 und 130), und auch das mittels TAG dargestellte Zeichen wird zweimal produziert (in Zeile 170 und 190). Die FRAME-Kommandos erzeugen eine ausreichende Verzögerung, um die Wirkung sichtbar zu machen. Beachten Sie, daß in den Anweisungen in Zeile 90 der erste Parameter ausgelassen wurde. Das ist in diesen Befehlen ohne weiteres möglich; dadurch bleiben die vorher gesetzten Werte für diesen Parameter gültig.

Der dritte Parameter des MOVE-Befehls (,1) in Zeile 220 setzt den GRAPHICS PEN auf 1 und überschreibt somit die '3' in Zeile 60. Der XOR-Modus wird durch den vierten Parameter von DRAWR in Zeile 230 gesetzt. Beachten Sie auch hier den fehlenden Parameter.

Wenn Sie den MASK-Befehl in Zeile 90 weglassen, wird der erste Punkt in jeder Linie geplottet, d.h. die Ecken werden doppelt gezeichnet. Durch XOR heben sich die doppelten Punkte auf; deswegen sind die Ecken verschwunden.

```
10 REM XOR ink modes
20 MODE 1:INK 2,10:INK 3,4
30 ORIGIN 440,100,440,640,100,300
40 WINDOW 1,26,1,25
50 CLG 2:GRAPHICS PAPER 0
60 DRAW 200,200,3
70 MOVE 2,0:FILL 3
80 ORIGIN 440,0,440,640,0,400
```

```
90 GRAPHICS PEN ,1:MASK ,0
100 FOR y=60 TO 318 STEP 2
110 GOSUB 220
120 FRAME:FRAME
130 GOSUB 220
140 NEXT
150 TAG
160 FOR y=60 TO 318 STEP 2
170 MOVE 96,y:PRINT CHR$(224);
180 FRAME:FRAME
190 MOVE 96,y:PRINT CHR$(224);
200 NEXT
210 END
220 MOVE 90,y,1
230 DRAWR 20,0,,1
240 DRAWR 0,20
250 DRAWR -20,0
260 DRAWR 0,-20
270 RETURN
run
```

Bewegungseffekte

Durch wechselnde Farben für einen Farbstift lassen sich Bewegungseffekte erzielen. Obwohl der Inhalt des Bildschirmspeichers unverändert bleibt, scheint es auf dem Bildschirm eine Bewegung zu geben. Ein Beispiel dafür finden Sie im 'Welcome'-Programm auf Seite 4 Ihrer Systemdisketten (Tippen Sie `RUN "disc"` ein, um dieses Programm abzuspielen.)

Der einfache Farbwechsel in diesem Beispiel reicht jedoch nicht aus, wenn die Muster sich überlappen sollen. Im nächsten Beispiel werden die Farbwerte durch `OR` verknüpft, um die Zahlen 1 bis 4 auf den Bildschirm zu bringen. Die Form wird bestimmt, indem das Zeichen links unten abgetastet und als große Blockgraphik wiedergegeben wird. Die Nummern werden unter Verwendung der Farbstifte `1`, `2`, `4` und 8 mit eingeschaltetem `OR`-Modus abwechselnd geschrieben -in diesem Fall mit der Steuer-Zeichen-Folge, siehe Zeile `50`.

Ab Zeile `60` rotiert die Palette entsprechend einer mathematischen Formel, die bewirkt, daß immer eine Blockgraphik-Nummer dargestellt wird. Die Auswahl des Farbstifts erfolgt, indem der Reihe nach jeder Farbstift untersucht wird, um festzustellen, ob er die gesuchte binäre Komponente enthält. Die Nummer drei wurde z.B. mit Farbstift 4 gezeichnet; deshalb mußten wir, um die Zahl 3 darzustellen, allen Farbstiften, deren Nummer eine binäre 4 enthält, eine sichtbare Farbe zuordnen. Diese Farbstifte sind:

4(0100), 5(0101), 6(0110), 7(0111), 12(1100), 13(1101), 14(1110), 15(1111)

In einer praktischen Anwendung würden die Farbstifte berechnet, die in jedem Bewegungsstadium zu wechseln sind, und die Zeilen 180 bis 200 würden durch einen zügigeren Programmteil ersetzt.

```
10 REM latch animation
20 ON BREAK GOSUB 220
30 FOR i=1 TO 15:INK i,26:NEXT
40 m(1)=1:m(2)=2:m(3)=4:m(4)=8
50 MODE 0:PRINT CHR$(23);CHR$(3);:TAG
60 FOR P=1 TO 4
70 GRAPHICS PEN m(p),1
80 LOCATE #1,1,25:PRINT#1,CHR$(48+p);
90 FOR x=0 TO 7
100 FOR y= 0 TO 14 STEP 2
110 IF TEST(x*4,y)=0 THEN 140
120 MOVE (x+6)*32,(y+6)*16:PRINT CHR$(143);
130 MOVE (x+6)*32,(y+7)*16:PRINT CHR$(143);
140 NEXT y,x,p
150 LOCATE #1,1,25:PRINT#1," ";
160 FOR p=1 TO 4
170 FOR i= 1 TO 25:FRAME:NEXT
180 FOR i=0 TO 15
190 IF (i AND m(p))=0 THEN INK i,0 ELSE INK i,26
200 NEXT i,p
210 GOTO 160
220 INK 1,26
run
```

Bild-Ebenen mit verschiedenen Farben

Im obigen Beispiel haben wir gesehen, wie in den mit den Farbstiften 1, 2, 4 und 8 gezeichneten Graphiken ein Bewegungseffekt durch Farbwechsel entstand. Wenn wir dieselben Farbstifte benutzen, aber die Farben in anderer Weise bestimmen, entsteht ein völlig anderer Effekt mit verschiedenen Farbebenen. Folgendes Beispiel soll dies demonstrieren:

```
10 REM Berge
20 DEFINT a-z
30 INK 0,1:INK 1,26
40 INK 2,6:INK 3,6
```

```
50 FOR i=4 TO 7:INK i,9:NEXT
60 FOR i=8 TO 15:INK i,20:NEXT
70 MODE 0:DEG:ORIGIN 0,150:CLG:MOVE 0,150
80 FOR x=16 TO 640 STEP 16
90 DRAW x,COS(x)*150+RND*100,4
100 NEXT
110 MOVE 0,0:FILL 4
120 cx=175:GOSUB 320
130 cx=525:GOSUB 320
140 SYMBOL 252,0,0,&C,&1F,&30,&7F,&FF
150 SYMBOL 253,0,6,&E,&F2,2,&F2,&FE
160 SYMBOL 254,0,&60,&70,&7F,&7F,&7F,&7F
170 SYMBOL 255,0,0,0,&F8,&EC,&FE,&FF
180 pr$=CHR$(254)+CHR$(255)
190 pl$=CHR$(252)+CHR$(253)
200 TAG:t!=TIME
210 FOR x=-32 TO 640 STEP 4
220 x2=((608-x)*2)MOD 640:hl=RND*10:hr=50*SIN(x)
230 GRAPHICS PEN 8,1:MOVE x,100+hr,,3:PRINT pr$;
240 GRAPHICS PEN 2,1:MOVE x2,115+hl,,3:PRINT pl$;
250 IF (TEST(x2-2,115+hl-12) AND 8)=8 THEN 380
260 IF TIME-t!<30 THEN 260
270 FRAME:t!=TIME
280 GRAPHICS PEN 7,1:MOVE x,100+hr,,2:PRINT pr$;
290 GRAPHICS PEN 13,1:MOVE x2,115+hl,,2:PRINT pl$;
300 NEXT
310 GOTO 210
320 MOVE cx,100
330 FOR x=0 TO 360 STEP 10
340 DRAW cx+SIN(x)*50+10*RND,100+COS(x)*25+10*RND,1
350 NEXT
360 DRAW cx,100:MOVE cx,90:FILL 1
370 RETURN
380 ENT -1,1,1,1
390 SOUND 1,25,400,15,,1,15
400 FOR y=100+hr TO -132 STEP -2
410 GRAPHICS PEN 7,1:MOVE x,y,,2:PRINT pr$;
420 GRAPHICS PEN 8,1:MOVE x,y-2,,3:PRINT pr$;
430 NEXT
440 GOTO 70
run
```

Um uns klar zu machen, wie das Programm funktioniert, müssen wir uns die Farbstift-Nummer wieder als Binärzahl vorstellen. Beginnend mit der höchsten Farbstift-Nummer (15) werden alle Farbstifte, die das Bit für die Zahl 8 enthalten (15 bis 8) mit dem Befehl INK auf blaugrün gesetzt. Dann werden alle Farbstifte, die das Bit für 4 enthalten (7 bis 4), auf grün gesetzt. Die Farbstifte 2 und 3, die jeder das Bit für 2 enthalten, werden auf rot, und Farbstift 1 schließlich wird auf leuchtend weiß gesetzt. Farbstift 0 bleibt blau.

Mit OR werden die Graphiken auf dem Bildschirm an ihren Platz gerückt - siehe Zeile 230 und 240. Die Farbe für jedes Pixel wird durch das Bit bestimmt, das an dieser Position im Ergebnis die höchste Signifikanz aufweist. Deshalb wird ein Bild einer 'signifikanteren' Ebene immer das Bild einer 'weniger signifikanten' Ebene überdecken. Der Hintergrund bleibt aber erhalten und wird wieder sichtbar, wenn das 'signifikantere' Bild entfernt wird. Um eine Bildebene zu entfernen, plottet man das Bild mit dem AND-Farbstift-Modus unter Verwendung der INK-Farben 7, 11, 13 oder 14, wobei die vorherigen INK-Farben 8, 4, 2 und 1 jeweils gelöscht werden -siehe Zeile 280 und 290.

Zusätzlicher Speicherplatz für Graphik-Programme

Wir schließen dieses Kapitel mit einem umfassenden Programm ab, mit dem Sie ihre eigenen Bilder entwerfen können. Dazu wird der zusätzliche 64K RAM-Speicher des 6128 beansprucht:

```
10 'SCREEN DESIGNER von DAVID RADISIC
20 ' copyright (c) AMSOFT 1985
30 '
40 'Vor diesem Programm das "BANKMAN" -Programm laufen lassen
50 '*********************************************************
60 '
70 ON ERROR GOTO 2740
80 DEFINT a-x
90 MODE 1:ch=127:cmnd=1:pn(0)=0:pn(1)=26:pn(2)=15:pn(3)
   =6:pn(4)=0:pn=1:norx=1:menu=1:zzz=HIMEM
100 DIM command$(22)
110 norx$(0)="Normal":norx$(1)="XOR    ":norx$(2)="Trans
    p":norx$(3)="XOR    "
120 RESTORE:READ cmnds$(1),cmnds$(2):cmnd$=CHR$(16)+CHR
    $(&7F)+cmnds$(1)+cmnds$(2)
130 READ cmno:FOR i=1 TO cmno:READ command$(i):NEXT
140 READ st$:IF st$<>"**" THEN cmnd$(cmnd)=st$:cmnd=cmn
    d+1:GOTO 140
150 WINDOW #0,1,40,1,3:PAPER #0,0:PEN #0,1:CLS #0
```

```
160 WINDOW #1,1,40,4,4:PAPER #1,3:PEN #1,1:CLS #1
170 ORIGIN 0,0,0,640,0,334
180 x=320:y=200:MOVE x,y
190 BORDER pn(4):FOR i=0 TO 3:INK i,pn(i):NEXT
200 MASK 255,0:PAPER 0:PEN 1:PAPER #1,3:PEN #1,1:GRAPHI
    CS PEN pn,norx
210 IF flag<>5 THEN 280
220 IF pn<2 THEN pnt$=CHR$(240):px=(pn+1)*13 ELSE IF pn
    <4 THEN pnt$=CHR$(241):px=(pn-1)*13 ELSE pnt$=CHR$(
    243):px=37
230 LOCATE px,2:PRINT pnt$;
240 LOCATE 1,1:PRINT USING"   PEN 0 : ##   PEN 1 : ##";
    pn(0);pn(1);
250 LOCATE 29,2:PRINT USING"Border : ##";pn(4)
260 LOCATE 1,3:PRINT USING"   PEN 2 : ##   PEN 3 : ##";
    pn(2);pn(3);
270 LOCATE px,2:PRINT " ";
280 LOCATE #1,1,1:PRINT#1,USING"X :####  Y :####     ";
    x;y;:PRINT #1,"Plot mode : ";norx$(norx+(undraw*2))
    ;" ";
290 IF flag=0 THEN GOSUB 2260
300 '
310 GOSUB 970
320 '
330 IF flag>0 THEN 390
340 IF i$="" THEN 390
350 cmnd=INSTR(cmnd$,i$):IF cmnd=0 THEN 390
360 IF cmnd=1 THEN CLG:x=320:y=200:GOTO 390
370 IF cmnd=2 THEN RUN 70
380 ON cmnd-2 GOSUB 1240,1410,1520,1640,1840,1860,1950,
    2020,2090,2120,2170,2200,2660,2660,2660,2660,2390,2
    330,2200
390 IF tx=0 AND ty=0 THEN 200
400 IF flag>0 THEN 440
410 GOSUB 630
420 GOSUB 680:FRAME:GOSUB 680
430 GOTO 200
440 MOVE tempx,tempy,pn,1
450 ON flag GOSUB 470,490,550,640
460 GOTO 200
470 PLOT x,y:GOSUB 630:PLOT x,y
480 RETURN
490 DRAW tempx+x,tempy:DRAW tempx+x,tempy+y
500 DRAW tempx,tempy+y:DRAW tempx,tempy
510 GOSUB 630
520 DRAW tempx+x,tempy:DRAW tempx+x,tempy+y
```

```
530 DRAW tempx,tempy+y:DRAW tempx,tempy
540 RETURN
550 MOVE tempx,tempy:DRAWR x,y
560 IF triside=0 THEN 580
570 DRAW tempxx,tempyy:DRAW tempx,tempy
580 GOSUB 630
590 MOVE tempx,tempy:DRAW tempx+x,tempy+y
600 IF triside=0 THEN RETURN
610 DRAW tempxx,tempyy:DRAW tempx,tempy
620 RETURN
630 x=x+tx:y=y+ty:RETURN
640 MOVE tempx,tempy:DRAW x,y
650 GOSUB 630
660 MOVE tempx,tempy:DRAW x,y
670 RETURN
680 ' draw and undraw cursor
690 IF flag=5 THEN RETURN
700 MASK 255,1
710 IF flag>1 THEN xx=tempx+x:yy=tempy+y ELSE xx=x:yy=y
720 IF flag=4 THEN xx=x:yy=y
730 IF flag=1 THEN xx=x:yy=y
740 IF undraw=1 THEN 820
750 GOSUB 790
760 MASK 255,0
770 IF i$=" " THEN GOSUB 2150:i$=""
780 RETURN
790 MOVE xx-4,yy,pn,1:DRAW xx+4,yy
800 MOVE xx,yy-4:DRAW xx,yy+4
810 MOVE xx,yy,,xorn:RETURN
820 nx=1:GOSUB 1220
830 FRAME:GOSUB 1220
840 IF i$=" " THEN nx=norx:GRAPHICS PEN pn,1:GOSUB 1220
850 i$=""
860 IF flag<>6 THEN 760
870 IF moved=0 AND j$<>"" AND (j$<CHR$(240) OR j$>CHR$(
    247)) THEN ch=ASC(j$):moved=1
880 IF moved=0 THEN RETURN
890 LOCATE 5,2
900 FOR i=ch-5 TO ch+5
910 PEN ABS(i<>ch)+1
920 ch$=CHR$(1)+CHR$(ABS(i+256)MOD 256)
930 IF ch=i THEN PRINT" "ch$" "; ELSE PRINT ch$;
940 NEXT
950 PEN 1:PRINT"   = "ch"  ";
960 GOTO 760
970 ty=0:tx=0:GOSUB 680:FRAME:GOSUB 680
```

```
980 IF INKEY(0)<>-1 OR INKEY(72)<>-1 THEN ty=16
990 IF INKEY(2)<>-1 OR INKEY(73)<>-1 THEN ty=-16
1000 IF INKEY(8)<>-1 OR INKEY(74)<>-1 THEN tx=-16
1010 IF INKEY(1)<>-1 OR INKEY(75)<>-1 THEN tx=16
1020 IF INKEY(21)<>-1 OR INKEY(76)<>-1 THEN tx=tx/8:ty=
     ty/8
1030 IF tx=0 AND ty=0 THEN moved=0 ELSE moved=1
1040 j$=INKEY$:i$=UPPER$(j$)
1050 IF (i$=" " OR i$=CHR$(13)) AND flag>0 THEN 1090
1060 IF flag=5 THEN 1120
1070 IF flag=6 THEN 1170
1080 RETURN
1090 ON flag GOSUB 1240,1410,1640,1860,1950,2020
1100 i$=""
1110 RETURN
1120 IF moved=0 THEN RETURN
1130 IF tx>2 THEN pn=(pn+1) MOD 5 ELSE IF tx<-2 THEN pn
     =ABS((pn<1))*5-1+pn
1140 IF ty>2 THEN pn(pn)=(pn(pn)+1) MOD 27 ELSE IF ty<-
     2 THEN pn(pn)=ABS((pn(pn)<1))*27-1+pn(pn)
1150 GRAPHICS PEN pn:PEN #1,pn
1160 tx=0:ty=0:BORDER pn(pn):RETURN
1170 IF tx<0 THEN ch=ABS(ch+255) MOD 256
1180 IF ty<0 THEN ch=ABS(ch+246) MOD 256
1190 IF tx>0 THEN ch=(ch+1) MOD 256
1200 IF ty>0 THEN ch=(ch+10) MOD 256
1210 tx=0:ty=0:RETURN
1220 TAG:MOVE xx-8,yy+6,pn,nx:PRINT CHR$(ch);:TAGOFF
1230 RETURN
1240 ' C
1250 IF flag=1 THEN 1290
1260 ro=1:GOSUB 2240
1270 tempx=x:tempy=y:flag=1
1280 RETURN
1290 IF tempx=x AND tempy=y THEN 1390
1300 PLOT x,y,,1
1310 tix=MAX(x,tempx)-MIN(tempx,x):tiy=MAX(y,tempy)-MIN
     (tempy,y)
1320 ti=SQR((tix↑2)+(tiy↑2))
1330 ORIGIN tempx,tempy
1340 PLOT 0,0,pn,0:MOVE 0,-ti
1350 FOR z=0 TO PI*2+0.01 STEP PI/(ti/2)
1360 DRAW SIN(z+PI)*ti,COS(z+PI)*ti,pn,norx
1370 NEXT z
1380 ORIGIN 0,0
1390 x=tempx:y=tempy:tempx=0:tempy=0:flag=0
```

```
1400 RETURN
1410 ' B
1420 IF flag=2 THEN 1470
1430 ro=2:GOSUB 2240
1440 tempx=x:tempy=y:flag=2
1450 x=0:y=0
1460 RETURN
1470 IF norx=1 THEN 1500
1480 MOVE tempx,tempy:DRAW tempx+x,tempy,,norx
1490 DRAW tempx+x,tempy+y:DRAW tempx,tempy+y:DRAW tempx
     ,tempy
1500 x=tempx:y=tempy:flag=0
1510 RETURN
1520 ' F
1530 ro=3:GOSUB 2240
1540 GOSUB 1620:IF i$=" " THEN 1600
1550 edgecol=VAL(i$)
1560 ro=4:GOSUB 2240
1570 GOSUB 1620:IF i$=" " THEN 1600
1580 filler=VAL(i$)
1590 MOVE x,y,edgecol:FILL filler
1600 flag=0:i$=""
1610 RETURN
1620 i$=INKEY$:IF (i$<"0" OR i$>"3") AND i$<>" " THEN 1
     620
1630 RETURN
1640 ' T
1650 IF flag=3 THEN 1700
1660 flag=3:ro=5:GOSUB 2240
1670 tempx=x:tempy=y
1680 x=0:y=0
1690 RETURN
1700 IF triside<>0 THEN 1770
1710 ro=6:GOSUB 2240
1720 MOVE 0,0,pn,1:GOSUB 590
1730 tempxx=tempx+x:tempyy=tempy+y:x=x/2:y=20
1740 triside=1
1750 GOSUB 550:GOSUB 590
1760 RETURN
1770 IF norx=1 THEN 1800
1780 MOVE tempxx,tempyy,,norx:DRAW tempx,tempy
1790 DRAW tempx+x,tempy+y:DRAW tempxx,tempyy
1800 tempxx=0:tempyy=0
1810 x=tempx:y=tempy:triside=0
1820 tempx=0:tempy=0:flag=0
```

```
1830 RETURN
1840 ' @
1850 norx=1:undraw=undraw XOR 1:RETURN
1860 ' L
1870 IF flag=4 THEN 1910
1880 ro=7:GOSUB 2240
1890 tempx=x:tempy=y:flag=4
1900 RETURN
1910 IF norx=1 THEN 1930
1920 MOVE tempx,tempy,,norx:DRAW x,y
1930 x=tempx:y=tempy:flag=0
1940 RETURN
1950 ' I
1960 IF flag=5 THEN flag=0:CLS:INK 3,tmpcol:INK pn,col:
     GOTO 1990
1970 CLS:flag=5:BORDER pn(pn)
1980 RETURN
1990 FOR i=0 TO 3:INK i,pn(i):NEXT:BORDER pn(4)
2000 IF pn=4 THEN pn=1
2010 CLS:RETURN
2020 ' A
2030 IF flag=6 THEN 2070
2040 tempx=0:tempy=0:CLS
2050 undraw=1:flag=6:norx=1:moved=1
2060 RETURN
2070 flag=0
2080 RETURN
2090 ' N
2100 norx=0
2110 RETURN
2120 ' E
2130 GRAPHICS PEN pn,0:TAG:MOVE xx-8,yy+6,,0:PRINT " ";
     :TAGOFF
2140 RETURN
2150 '<SPACE>
2160 PLOT x,y,pn,norx:RETURN
2170 ' X
2180 norx=1
2190 RETURN
2200 ' M
2210 menu=menu MOD 2+1
2220 GOSUB 2260:RETURN
2230 i$=UPPER$(INKEY$):IF i$="" OR INSTR(ser$,i$)=0 THE
     N 2230 ELSE RETURN
```

```
2240 CLS:undraw=0:PRINT cmnd$(ro);:LOCATE 1,3:PRINT"<Le
     ertaste> ";:IF ro=3 OR ro=4 THEN PRINT"fuer Menu"
2250 RETURN
2260 CLS:flag=-1
2270 FOR i=1 TO LEN(cmnds$(menu))
2280 ps=i+ABS(menu=2)*LEN(cmnds$(1))
2290 PEN 1:PRINT"<"MID$(cmnds$(menu),i,1)">"MID$(comman
     d$(ps),2,4)" ";
2300 NEXT
2310 PRINT"<CLR>   <DEL>   <SPACE>";
2320 RETURN
2330 ' S
2340 GOSUB 2460:IF filename$="" THEN 2370
2350 GOSUB 2550
2360 SAVE filename$,b,&C000,&4000
2370 GOSUB 2260
2380 RETURN
2390 ' R
2400 GOSUB 2460:IF filename$="" THEN 2440
2410 GOSUB 2730
2420 LOAD filename$,&C000
2430 GOSUB 2570
2440 GOSUB 2260
2450 RETURN
2460 CLS:LOCATE 10,3:PRINT"<RETURN> um Funktion abzubre
     chen!";
2470 LOCATE 1,1:PRINT"Dateinamen eingeben :";
2480 INPUT "",filename$:IF filename$="" THEN RETURN
2490 n=INSTR(filename$,"."):IF n=0 THEN 2520
2500 IF n=1 THEN 2460
2510 filename$=LEFT$(filename$,n-1)
2520 filename$=LEFT$(filename$,8)+".scn"
2530 CLS
2540 RETURN
2550 FOR i=0 TO 4:POKE &C000+i,pn(i):NEXT
2560 RETURN
2570 FOR i=0 TO 4:pn(i)=PEEK(&C000+i) MOD 27:NEXT
2580 cn=0:FOR i=0 TO 2:IF pn(i)=pn(i+1) THEN cn=cn+1
2590 NEXT:IF cn=3 THEN 2630
2600 FOR i=0 TO 3:INK i,pn(i):NEXT
2610 BORDER pn(4):pn=1:GRAPHICS PEN pn
2620 RETURN
2630 pn(0)=0:pn(1)=26:pn(2)=15:pn(3)=6:pn(4)=0
2640 GOTO 2600
2650 ' 1, 2, 3, & 4
```

```
2660 CLS:PRINT"Wollen Sie das Bild <S>peichern":PRINT T
     AB(16)"<H>ervorholen":PRINT TAB(13)"oder <T>ausche
     n?"
2670 ser$="SHT"+CHR$(13):GOSUB 2230:IF i$=CHR$(13) THEN
      2260
2680 bnk2=(cmnd-13):bnk1=1
2690 IF i$="S" THEN CLS:GOSUB 2550:|SCREENCOPY,bnk2,bnk
     1
2700 IF i$="H" THEN GOSUB 2730:|SCREENCOPY,bnk1,bnk2:GO
     SUB 2570
2710 IF i$="T" THEN CLS:GOSUB 2730:GOSUB 2550:|SCREENSW
     AP,bnk2,bnk1:GOSUB 2570
2720 GOSUB 2260:RETURN
2730 FOR i=0 TO 3:INK i,0:NEXT:BORDER 0:RETURN
2740 CLS:GOSUB 2600:RESUME 2260
2750 DATA "CBFT@LIANEXM","1234RSM"
2760 DATA 19,Circle,"Box  ","Fill ",Triangle,Alternate,
     "Line ","Inks ",ASCII,Normal,Erase,"Xor  ","Menu "
     ,"1    ","2    ","3    ","4    ",Restore,"Save ","
     Menu "
2770 DATA Circle,Box,Edge colour,Filler colour,Triangle
      1,Triangle 2,Line,**
```

Die beiden RSX-Befehle |SCREENCOPY und |SCREENSWAP in diesem Programm stellt das 'Bank Manager'-Dienstprogramm auf Seite 1 Ihrer System-Disketten zur Verfügung. Diese Befehle erleichtern das Kopieren und Austauschen verschiedener Bildschirminhalte zwischen den zugehörigen Speicherblöcken und zwischen den beiden 64K-Bänken des Speichers.

Bevor Sie das obige Programm laufen lassen, müssen Sie deshalb das 'Bank Manager'-Programm von Seite 1 einlegen und:

```
run "bankman"
```

...eintippen. Danach können Sie das Bildentwurfsprogramm laufen lassen.

Was passiert danach?

Nachdem Sie das Bank Manager-Programm und das Bildentwurfsprogramm haben ablaufen lassen, erscheint auf dem Bildschirm ein Menü mit einem blinkenden Graphik-Cursor in der Mitte des Bildschirms.

Sie brauchen nur den Anfangsbuchstaben der gewünschten Option zu drücken (z.B. C für Circle), damit das zugehörige Programm ausgeführt wird. Tippen Sie zur Demonstration:

```
C
```

Drücken Sie die Cursor-oben Taste, bis der Cursor ungefähr drei Zentimeter oberhalb der Bildschirmmitte liegt.

Drücken Sie dann die Leertaste, um die Kreis-Funktion auszuführen. Anschließend kommen Sie wieder ins Menü.

Wenn Sie `M` (für Menü) drücken, kommen Sie zum alternativen Menü, mit dem Sie Bildschirmabdrücke auf Diskette speichern (S) oder abrufen (R) können; außerdem haben Sie die Möglichkeit, die Inhalte des 1., 2., 3. oder 4. Bildschirms, die sich alle in der zweiten 64K-Bank befinden, zu 'verschieben'.

Wenn Sie solche 'Verschiebungen' vornehmen möchten, tippen Sie die Speicher-Nr. (1, 2, 3 oder 4) und sehen nun ein weiteres Menü:

S	(Store) Legt einen Bildschirmabdruck im Speicher ab
R	(Retrieve) Ruft einen Bildschirmabdruck aus dem Speicher
E	(Exchange) Tauscht Bildschirmabdrücke aus
[RETURN]	Bricht die Operation ab

Möchten Sie beispielsweise das Bild, das Sie gerade auf dem Bildschirm haben, im Speicher Nr. 2 ablegen, tippen Sie `2` und dann `S` ein.

Nach Ausführung der S-, R- oder E-Funktion kehren Sie zum alternativen Menü zurück. Wenn Sie jetzt `M` drücken, sind Sie wieder im ursprünglichen Menü.

Unser Beispielprogramm bietet eine Vielzahl von Möglichkeiten: Es zeichnet Kästen, Kreise, Dreiecke, Linien, Punkte, es kann Formen farblich ausfüllen und Symbole zeichnen.

Ist ein Bild fertig, können Sie es zur späteren Verwendung auf Diskette speichern, indem Sie S aus dem alternativen Menü auswählen, und mit R aus demselben Menü können Sie es wieder auf den Bildschirm holen.

Damit ist nun das letzte Kapitel dieses Handbuches abgeschlossen. Wenn Sie unsere zahlreichen Beispielprogramme analysiert und ausprobiert haben, sollten Sie den Umgang mit dem AMSTRAD-BASIC und dem CPC6128 mittlerweile recht gut beherrschen.

Weitergehende Informationen

Aus vielen anderen Publikationen von AMSTRAD und unabhängigen Autoren erfahren Sie weitere Einzelheiten über den 6128, sowie über BASIC, die Firmware, das CP/M-Betriebssystem und die Computer-Sprache Dr.LOGO.

Am Schluß dieses Buches finden Sie noch einen vierteiligen Anhang mit dem CP/M-Benutzer-Lizenzabkommen, Begriffserklärungen, einigen Spiel-Programmen und einem Index für dieses Handbuch.

Wir hoffen, daß dieses Handbuch für Sie eine interessante und informative Lektüre war und danken Ihnen für den Kauf des CPC6128.

Anhang 1
Endabnehmer-Lizenzabkommen mit Digital Research und AMSTRAD

Hinweis für den Anwender:

Lesen Sie bitte diesen Abschnitt sorgfältig durch, bevor Sie das Disketten-Paket öffnen.

Durch das Öffnen des Disketten-Pakets zeigen Sie sich mit dem Inhalt des Endabnehmer-Lizenzabkommens und den darin genannten Bedingungen einverstanden.

1. Definitionen

1.1 DRI steht für DIGITAL RESEARCH (CALIFORNIA) INC., P.O. Box 579, Pacific Grove, California 93950, dem Inhaber des Copyright für das Programm.

1.2 Maschine steht für den einen Mikrocomputer, auf dem Sie das Programm betreiben. Weitere CPU-Systeme erfordern zusätzliche Lizenzen.

1.3 Programm steht für den Programm-Satz, die Dokumentation und zusätzliches Material in diesem Paket, sowie ergänzende Nachträge und Zusätze, die von DRI geliefert werden, ungeachtet spezieller Verwendungen und Modifikationen Ihrerseits.

1.4 AMSTRAD steht für AMSTRAD PLC., Brentwood House, 169 Kings Road, Brentwood, Essex CM14 4EF, England.

Die Verantwortung für die Programm-Auswahl, ihre Verwendung, sowie die Installation liegt beim Käufer.

2. Lizenz

Sie dürfen:

2.1 das Programm nur auf einer einzigen (dieser) Maschine betreiben.

2.2 das Programm auf dieser Maschine für Reservekopien oder Änderungen in beliebige maschinenlesbare oder gedruckte Form übertragen, wenn es Ihren persönlichen Zwecken dient. Die Anfertigung von bis zu drei Programm-Kopien ist erlaubt. Einige Programme verbieten oder beschränken die Anfertigung von Kopien. Sie sind durch den Hinweis "copy protected" gekennzeichnet. Das Kopieren und Vervielfältigen der Dokumentation und des zugehörigen gedruckten Materials ist verboten. Das Disassemblieren des Programm-Codes ist verboten.

2.3 das Programm auf dieser Maschine verändern und/oder es in andere Programme einfügen, sofern es ihrer persönlichen Verwendung dient. Jeder in ein anderes Programm integrierte Programmteil unterliegt ebenfalls diesen Lizenzbestimmungen.

2.4 das Programm und die Lizenz an eine dritte Person übertragen, sofern Sie den Namen und die Adresse an DRI melden und die dritte Person:

a) den Inhalt und die Bedingungen dieses Lizenzabkommens akzeptiert

b) eine Kopie der Registrierkarte unterschreibt und an DRI sendet

c) die Lizenzgebühr entrichtet

Wenn Sie das Programm übertragen, müssen Sie entweder gleichzeitig alle Kopien, einschließlich Original, in gedruckter oder maschinenlesbarer Form mit übergeben, oder alle nicht übergebenen Kopien vernichten. Dies gilt auch für sämtliche modifizierten Programme und Programmteile, die in andere Programme integriert wurden.

Der Text des Copyright muß auf jeder Kopie, sowie auf jedem veränderten Programm und auf jedem Programmteil stehen, der in andere Programme integriert wurde.

JEDE DISKETTE TRÄGT EINE SERIENNUMMER. DAS PROGRAMM, EINE KOPIE, EINE MODIFIKATION UND JEDER PROGRAMMTEIL, DER IN EIN ANDERES PROGRAMM INTEGRIERT WURDE, DARF, SOFERN IM LIZENZABKOMMEN NICHT AUSDRÜCKLICH GENEHMIGT, WEDER GANZ NOCH TEILWEISE EINER DRITTEN PERSON ZUGÄNGLICH GEMACHT WERDEN.

BEI ZUWIDERHANDLUNGEN ERLÖSCHEN ALLE RECHTE AUS DIESEM LIZENZABKOMMEN.

3. Auflösung

Diese Lizenz ist bis zu ihrer Auflösung gültig. Sie können die Lizenzübereinkunft vorzeitig auflösen, indem Sie das Programm zusammen mit allen Kopien, Modifikationen und Programmteilen vernichten. Die Lizenzrechte erlöschen ebenfalls unter bestimmten, in diesem Abkommen dargelegten Bedingungen, und bei Verletzung des Abkommens. Sie erklären sich hiermit einverstanden, bei Beendigung des Abkommens das Programm mit allen Kopien, Modifikationen und Programmteilen zu vernichten.

4. Eingeschränkte Garantie

AMSTRAD UND DRI ÜBERNEHMEN KEINERLEI GARANTIE FÜR DIE EIGNUNG DES PROGRAMMS FÜR BESTIMMTE ANWENDUNGEN. DAS RISIKO BEZÜGLICH QUALITÄT UND LEISTUNGSFÄHIGKEIT LIEGT BEIM KÄUFER. SOLLTE DAS PROGRAMM DEFEKT SEIN, TRÄGT DER KÄUFER DIE NOTWENDIGEN KOSTEN FÜR REPARATUR UND SERVICE.

Weder AMSTRAD noch DRI übernehmen die Garantie dafür, daß die im Programm enthaltenen Funktionen den Anforderungen des Benutzers genügen bzw. fehlerfrei sind.

AMSTRAD gibt jedoch die Gewähr, daß sich die Diskette mit dem Programm in einwandfreiem Zustand hinsichtlich Material und Verarbeitung befindet. Die Garantiezeit für diese Eigenschaften beträgt 90 Tage ab Lieferdatum.

5. Haftung

AMSTRAD verpflichtet sich lediglich, Disketten, die der eingeschränkten Garantie nicht genügen, zu ersetzen.

DRI UND AMSTRAD HAFTEN NICHT FÜR SCHÄDEN, DIE IN IRGENDEINER FORM BEI DER BENUTZUNG DIESES PRODUKTS ENTSTEHEN.

6. Registrierkarten

Die Programme werden von Zeit zu Zeit von DRI überarbeitet. Diese überarbeiteten Versionen werden nur an Kunden ausgeliefert, deren unterzeichnete Registrierkarte bei DRI oder einem authorisierten Händler vorliegt. DRI ist jedoch nicht verpflichtet, die Programme zu überarbeiten oder überarbeitete Versionen zu liefern.

7. Allgemeines

Das Programm darf nur im Rahmen dieses Abkommens lizenziert, übereignet oder übertragen werden. Jeder Versuch, diese Rechte auf andere Weise zu lizenzieren oder zu übertragen, zieht keine Verpflichtungen des Herstellers nach sich.

Fragen, die sich aus diesem Lizenzabkommen ergeben, können direkt an:

Digital Research
P.O. Box 579
Pacific GroveCalifornia 93950
USA

...gerichtet werden.

DIESES LIZENZABKOMMEN DARF NICHT IN KAUFVERTRÄGEN, WERBEANGEBOTEN ODER ANDEREN WIEDERGABEN MODIFIZIERT WERDEN, SONDERN NUR DURCH EINEN SCHRIFTLICHEN VERTRAG MIT EINEM AUTHORISIERTEN HÄNDLER ODER AMSTRAD

DER KÄUFER BESTÄTIGT DURCH DEN ERWERB DIESES PRODUKTS SEIN EINVERSTÄNDNIS MIT DEN BEDINGUNGEN UND VERPFLICHTUNGEN DIESES ABKOMMENS. ER BESTÄTIGT FERNER, DASS ES ZWISCHEN IHM UND DRI SOWIE AMSTRAD KEINE ZUSÄTZLICHEN SCHRIFTLICHEN ODER MÜNDLICHEN ABSPRACHEN GIBT.

Anhang 2
Fachwörterverzeichnis

Einige häufig vorkommende Begriffe in der Datenverarbeitung werden hier für den Benutzer des CPC6128 erklärt.

Abenteuerspiel (Adventure Game)

Für einige das Höchste, für andere schlicht langweilig. Ein Computerspiel mit viel Text, bei dem der Spieler eingeladen ist, Abenteuer zu bestehen, um aus einem Irrgarten oder Labyrinth herauszufinden.

Abschneiden (Truncate)

Eine Zahl oder ein Text werden gekürzt, indem die ersten oder letzten Zeichen entfernt werden. Dieser Vorgang kann absichtlich herbeigeführt werden, um z.B. eine Zahl zu runden. Eine Zahl oder ein Text können auch unbeabsichtigt abgeschnitten werden, wenn nicht genug Platz für sie vorhanden ist. Der abgeschnittene Teil geht verloren.

Adresse

Die Zahl in einer Anweisung, die die Lage einer 'Zelle' im Computerspeicher angibt. Über die Adresse kann eine bestimmte Speicherstelle gefunden und ihr Inhalt 'gelesen' werden. Handelt es sich um ein RAM, kann auch etwas an eine Adresse 'geschrieben' werden.

Akkumulator (Accumulator)

Eine Speicherstelle im Schaltkreis des Mikroprozessors im Herzen des Mikrocomputers, in der Daten während ihrer Verarbeitung vorübergehend zwischengespeichert werden. Beim Programmieren in Maschinen-Code ist er ständig in Benutzung - für BASIC-Anwender ist er ohne Belang.

Akustikkoppler (Acoustic coupler)

Auch bekannt als akustisches Modem. Eine elektronische Zusatzeinrichtung, die einen Telefonhörer mit einem Computer verbindet, und diesen damit an das normale Telefonnetz anschließt. Auf diese Weise kann ein Computer mit öffentlichen Informationssystemen wie z.B. Btx verkehren, oder mit anderen Benutzern von Heimcomputern, um Software, Daten oder Informationen auszutauschen.

Algorithmus (Algorithm)

Ein großartig klingender Name für eine komplizierte Formel oder Rechenaufgabe. Eine Folge logischer und arithmetischer Schritte, um eine genau definierte Aufgabe mit dem Computer zu erledigen.

Alphanumerisch

Bezeichnet Buchstaben und Zahlen im Gegensatz zu graphischen Symbolen.

ALU

Arithmetic Logic Unit (arithmetisch-logische Einheit). Der Teil des Mikroprozessors, der die arithmetischen und logischen Operationen ausführt. Ist nur beim Programmieren in Maschinen-Code von Bedeutung.

AMSDOS

Abkürzung für AMStrad Disc Operating System. Dieses Programm ermöglicht BASIC den Zugriff auf Disketten.

Analog

Charakterisiert den Übergang zwischen einem Anfangs- und Endzustand als allmählich oder stufenweise und nicht abrupt. Computer arbeiten im Gegensatz dazu digital, so daß die in der natürlichen Welt meist analog vorkommenden Vorgänge erst in digitale Einheiten umgeformt werden müsssen, bevor ein Computer etwas damit anfangen kann.

Animation

Zeichentrickfilme sind die bekannteste Art von Animation. Bei der Computer-Animation werden graphische Darstellungen variiert, so daß der Eindruck von 'echter' Bewegung entsteht.

Anweisung (Statement)

Ein Befehl oder eine Folge von Befehlen in einem Programm.

Anwender-Programm (Applications programm)

Ein Programm zur Bearbeitung eines bestimmten Anwendungsproblems (z.B. Buchhaltung) im Gegensatz zu Systemprogrammen, die die Arbeit des Computers steuern.

Architektur (Architecture)

Allgemeiner Aufbau und interne Beziehung zwischen Datenbus, peripheren Geräten und CPU eines Mikrocomputers. Kein Thema für Leser dieser Begriffserklärungen - noch nicht!

Argument

Eine unabhängige Variable. In dem Ausdruck x+y=z z.B. sind x, y und z Argumente.

Array

Eine zweidimensionale Matrix (Gitter), in der Daten gespeichert und mit waagrechten und senkrechten Koordinaten adressiert werden.

ASCII

Steht für 'American Standard Code for Information Interchange'. Genormter Code für die Darstellung von Zahlen, Buchstaben und Symbolen, die über die Tastatur eingegeben oder durch eine Vielzahl anderer Befehle hervorgerufen werden.

Assembler

Eine maschinenorientierte Programmiersprache, bei der für Maschinen-Code-Anweisungen eine symbolische (mnemotechnische) Schreibweise verwendet wird.

Auffrischen (Refresh)

Das Aktualisieren von Daten auf dem Bildschirm oder im Speicher. Die schon vorhandenen Daten werden dabei nicht unbedingt vernichtet, sondern möglicherweise nur auf den neuesten Stand gebracht.

Auflösung (Resolution)

Die Möglichkeit festzustellen, wo ein Element einer Bildschirm-Darstellung endet, und wo das nächste beginnt. Manchmal auch für die Fähigkeit des Computers, Berechnungen mit sehr großen Zahlen durchzuführen.

Ausdruck (Expression)

Eine einfache oder komplizierte Formel in einem Programm zur Berechnung von Daten. Der Ausdruck beschreibt meist die Art von Daten, die bearbeitet werden kann. In der LOGO-Sprache besteht ein Ausdruck aus einem Prozedurnamen, gefolgt von den erforderlichen Eingabe-Elementen für die Prozedur.

Ausgabe (Output)

Alles was der Computer aufgrund seiner Funktionen ausgibt.

Austesten (Debugging)

Fehlersuche im Programm durch eine Kombination aus Versuchen und wissenschaftlichen Methoden.

Automatenspiel (Arcade Game)

Computerspiel mit viel 'Action', in denen z.B. Eindringlinge aus dem Weltall angreifen oder unersättliche Monster durch Labyrinthe jagen und unachtsame Opfer auffressen. Der Spieler muß seine Figuren so steuern, daß sie ihrem unangenehmen Schicksal entgehen. Diese Spiele machen Spaß und trainieren die Reflexe, bieten aber nur wenig für den, der sich ernsthaft mit dem Computer beschäftigt.

Backup

Reservekopie, die bei Verlust oder Beschädigung des Originals die Informationen rettet. Sie wird durch Duplizieren einer Diskette oder Disketten-Datei erzeugt.

BASIC

Beginners' All-purpose Symbolic Instruction Code. Eine interpretative Programmiersprache, mit der fast alle Heimcomputer betrieben werden. BASIC ist leicht zu erlernen und einfach zu benutzen, da Programme während ihrer Entwicklung zusammengestückelt und getestet werden können und nicht wie bei Compilersprachen erst das gesamte Programm fertig sein muß, bevor man es starten kann.

Basis (Base)

Grundzahl eines Zahlensystems. Das Binärsystem hat die Basis 2, das Dezimalsystem die Basis 10 und das Hexadezimalsystem die Basis 16.

Baud

Ein Bit pro Sekunde. Maßeinheit für die Geschwindigkeit bei der Datenübertragung.

BCD

Binary Coded Decimal. Ein System zur Codierung von Dezimalzahlen, in dem jede Zahl mit vier Binärziffern dargestellt wird.

BDOS

Basic Disc Operating System. Bestandteil des CP/M-Betriebssystems, der eine Anwender-Schnittstelle zum Aufruf der CP/M-Funktionen bereitstellt.

Befehl (Instruction)

Aufforderung oder Kommando an den Computer, eine bestimmte Operation auszuführen. Eine Folge von Befehlen bildet ein Programm.

Die primären logischen und mathematischen Prozesse, die ein Mikroprozessor ausführt. Jeder Befehl einer höheren Programmiersprache (auch Assembler-Befehle) muß in einzelne Befehle aufgegliedert werden, die von einer Computer-CPU verstanden werden können. Ein einziges Kommando kann viele Befehle der CPU auslösen.

Bemerkung (Remark)

Eine nicht ausführbare Programmanweisung (REM) im Programm, um zu dokumentieren, was an dieser Stelle getan wird, oder auch zur Kennzeichnung des Programmentwicklungsstandes.

Benutzerdefinierte Taste (User defined key - UDK)

Der CPC6128 hat 32 Tasten, die umdefiniert werden können, um verschiedene Aufgaben zu erfüllen.

Betriebssystem (Operating system)

Software, welche die Verwaltung des Speichers und die Koordinierung der Aktionen des Computers steuert.

Bildschirmeditor (Screen editor)

Ein Aufbereitungsprogramm für Texte und Programme, bei dem der Cursor an jede Stelle des Bildschirms positioniert werden kann, um die dort stehenden Zeichen zu ändern.

Binärsystem (Binary)

(Siehe Basis) Auch Dualsystem genannt. Das Zahlensystem zur Basis 2, in dem alle Zahlen nur mit den beiden Ziffern 0 und 1 dargestellt werden.

Binärzahl (Binary number)

Eine in binärer Schreibweise dargestellte Zahl. Bei der Programmierung des CPC6128 werden sie mit einem vorangestellten &X gekennzeichnet (z.B. bedeutet &X0101 in dezimaler Schreibweise 5).

BIOS

Steht für 'Basic Input/Output System'. Dies ist der hardware-abhängige Bestandteil von CP/M, der für einen speziellen Computertyp entwickelt wurde. Alle Ein- und Ausgabefunktionen, die über Bildschirm, Tastatur, Diskette usw. laufen, werden durch BIOS bearbeitet.

Bit

Kurzform für Binary digit (Binärziffer). Eine Ziffer im binären Zahlensystem kann nur den Wert 0 oder 1 annehmen.

Bitsignifikant (Bit significant)

Ergibt sich aus dem Zustand (0 oder 1) jeder Binärstelle in einem Byte eine Information, so nennt man diese Binärzahl bitsignifikant.

Boolesche Algebra (Boolean algebra)

Logische Verknüpfung, bei der nur zwei Antworten möglich sind: wahr und falsch, normalerweise mit 1 bzw. 0 dargestellt.

Booten

Laden eines Betriebssystems in den Speicher. Beim CP/M-Start unter BASIC wird automatisch ein kleines 'Boot-Programm' (siehe Urlader) von der Diskette geladen, das dann den Rest des Betriebssystems in den Speicher lädt.

Built-in commands

Befehle, die Bestandteil eines Betriebssystems sind. Sie werden schneller als transiente Befehle abgearbeitet, da sie nicht von Diskette geladen werden müssen.

Bus

Eine Gruppe von Verbindungen entweder im Computer selbst oder als Verbindung zu Peripheriegeräten. Sie übertragen Informationen über den Zustand der CPU, des RAM oder andere Hardwareeigenschaften. Der CPC6128-Bus befindet sich in der Platine der **EXPANSION**-Buchse an der Rückseite des Computers.

Byte

Eine Gruppe von acht Bits, welche die kleinste adressierbare Einheit im Speicher darstellt.

CAD

Computer Aided Design (Computer-unterstützter Entwurf). Normalerweise ein Zusammenspiel von Rechenleistung und Computergraphik als elektronisches Reißbrett, obwohl jede Art von Berechnung, die der Computer für einen 'Entwurf' durchführt, als CAD gelten kann.

CAE

Computer Aided Education (Computer-unterstützte Schulbildung). Noch eine Abkürzung aus der Computer-Welt. Hiermit ist alles gemeint, was mit Computern im Schulunterricht zu tun hat. Darunter fallen CAI (Computer Aided Instruction) und CAE (Computer Aided Education).

CCP

Abkürzung für 'Console Command Processor'. Dieses CP/M-Modul interpretiert und bearbeitet die über die Tastatur erfolgten Anwender-Eingaben. In der Regel sind diese Eingaben Befehle, die durch CCP geladen und ausgeführt werden.

Chip

Ein irreführender, umgangssprachlicher Begriff für eine integrierte Schaltung. Der Chip ist in Wirklichkeit ein Plättchen aus speziell hergestelltem Silizium, auf der die Schaltung untergebracht ist.

Code

Außer in seiner normalen Bedeutung wird dieses Wort von Programmierern oft als Kurzform für 'Maschinencode' benutzt.

Compiler

Ein komplexes Programm, das ganze, in einer höheren Programmiersprache wie BASIC geschriebene Programme in den direkten Befehlscode des Mikroprozessors umwandelt. Dadurch wird eine sehr viel schnellere Durchführung der Operationen erreicht.

Computer der fünften Generation (Fifth generation computers)

Meist sehr große Computer, die durch sogenannte künstliche Intelligenz selbst programmieren können.

Computergenerationen

Klassifizierung der großen technischen Meilensteine in der Computerentwicklung. Die sich daraus ergebenden Gruppierungen werden als Generationen bezeichnet.

CP/M

Steuerprogramm für Mikrocomputer. Ein Diskettenbetriebssystem von Digital Research mit einem Standard-System-Interface für Software, die für einen Großteil der Mikrocomputersysteme geschrieben wurde.

CPU

Central Processing Unit (Zentrale Prozessoreinheit). Das Herz jedes Computersystems, in dem Anweisungen interpretiert und deren Ausführung ausgelöst werden. In einem Mikrocomputer ist die CPU der Mikroprozessor selbst.

Cursor

Ein beweglicher Zeiger, der anzeigt, wo das nächste Zeichen auf dem Bildschirm erscheint.

Cursor-Tasten (Cursor control keys)

Tasten, auf denen Pfeile abgebildet sind. Mit ihnen läßt sich der Cursor auf dem Bildschirm umherbewegen. Sie werden auch häufig zum Steuern von Bewegungen in Automatenspielen benutzt.

Datei (File)

Eine Datensammlung, die meistens auf Kassette oder Diskette gespeichert ist.

Dateiname (Filename)

Name einer Datei. Bei Dr.LOGO besteht ein Dateiname aus bis zu acht alphabetischen oder numerischen Zeichen. Unter CP/M sind drei zusätzliche Zeichen für den Dateityp zugelassen, die durch einen Punkt vom ersten Namensteil getrennt werden.

Datenbasis (Data base)

Ein Bereich mit verschiedenen Datenarten in mehreren adressierbaren Formaten.

Datenerfassung (Data capture)

Vorgang, bei dem Daten von externen Geräten aufgenommen werden, die in irgendeiner Weise mit dem zentralen Computer verbunden sind.

Datensatz (record)

Eine Gruppe von Bytes in einer Datei. CP/M verwendet Datensätze zu 128 Byte.

Datenübertragung (Download)

Die Übertragung von Information von einem Computer zum anderen, wobei der Empfänger als 'downloading' bezeichnet wird. Der sendende Computer ist 'uploading'.

Dezimalsystem (Decimal system)

Zahlensystem mit der Basis 10. Mit den Ziffern von 0 bis 9 werden Einer-, Zehner-, Hunderter-, Tausender-Stellen usw. dargestellt.

Diagnose (Diagnostic)

Meldung, die der Computer automatisch ausgibt, um einen Fehler im Programm anzuzeigen.

Dienstprogramm (Utility program)

Programm auf Diskette, mit deren Hilfe der Anwender bestimmte Aufgaben bewältigen kann. (Siehe transiente Programme)

Digital

Beschreibt den Wechsel von einem Zustand zu einem anderen als abrupt, ohne Übergang. Gegenteil von analog.

Digitalisierer (digitiser)

Gerät, das analoge Information in das digitale Äquivalent umwandelt. Wird oft im Zusammenhang mit Graphik-Tabletts verwendet.

Diskette (Floppy disc, Disk)

Eine dünne Plastikscheibe, die auf einer oder beiden Seiten mit einer Magnetschicht versehen ist. Sie dient zur Speicherung von Daten. Die Scheibe befindet sich in einem Schutzgehäuse mit einem Fenster für das Lesegerät. Bei 3 Zoll-Disketten wird das Fenster automatisch mit einer Metallplatte verschlossen, wenn die Diskette aus dem Laufwerk genommen wird.

Diskettenlaufwerk (Disc drive)

Gerät, das die Diskette antreibt, und Informationen auf die magnetisierbare Oberfläche der sich drehenden Diskette schreibt, sowie dort gespeicherte Informationen wieder liest.

Dokumentation (Documentation)

Handbücher für die Benutzung von Computern oder Software.

Doppelseitig (Double sided)

Bezeichnet eine Diskette, die auf beiden Seiten Informationen speichern kann. Zwei-Kopf-Laufwerke können auf beide Seiten der Diskette zugreifen, ohne daß sie gewendet werden muß.

DOS

Abkürzung für 'Disc Operating System' (Disketten-Betriebssystem). Software, die alle Operationen im Zusammenhang mit dem Diskettenlaufwerk steuert.

Dr.LOGO

LOGO-Version von Digital Research. LOGO ist eine Programmiersprache mit Graphik-Schildkröte.

Drucker (Printer)

Gerät, um Texte oder Graphiken auf Papier auszudrucken.

E/A (I/O)

Eingabe/Ausgabe (Input/Output)

Echtzeit (Real time)

Ereignisse, die sich vor Ihren Augen abspielen, im Gegensatz zu jenen, die erst nach Ablauf eines Prozesses, der sie bewirkt hat, zutage treten.

Editieren (Edit)

Das Korrigieren oder Ändern von Daten, Programmen oder Texten.

Editor

Ein Programm, das sich normalerweise im ROM befindet und das Editieren ermöglicht.

Eingabe (Input)

Alles, was in den Computerspeicher gelangt, über die Tastatur, das Diskettenlaufwerk, die serielle Schnittstelle oder eine andere Quelle.

Einseitig (Single sided)

Bezieht sich auf eine Diskette, die nur auf einer Seite Daten speichern kann.

EPROM

Erasable, Programmable Read only Memory. Lösch- und programmierbarer Speicher, der nur gelesen werden kann. Ähnlich dem PROM, nur sind die Daten darauf mit ultraviolettem Licht löschbar, so daß neue Programme darauf gespeichert werden können. Ein EEPROM ist ähnlich, wird aber elektronisch gelöscht.

Fehler (Bug)

Die Fehlerskala reicht vom unerwarteten Programmablauf durch falsche Anwendung eines Programms (z.B. ändert sich die Bildschirmfarbe, wenn gleichzeitig vier Tasten gedrückt werden), bis hin zur Fehlerkette, die ein Programm vollständig und unwiderruflich abstürzen läßt und sämtliche Daten im Speicher löscht.

Festkommazahl (Fixed-point number)

Eine Zahl, deren Dezimalpunkt an einer fest definierten Position steht.

Firmware

Software im ROM - ein Zwischending zwischen reiner Software und reiner Hardware.

Flußdiagramm (Flow chart)

Eine Darstellung von Programmablauf und logischen Prozessen, womit die Reihenfolge der einzelnen Schritte bei der Ausführung des Programms verfolgt werden kann.

Forth

Eine sehr schnelle Programmiersprache, deren Geschwindigkeit und Komplexität zwischen höherer Programmiersprache und Maschinen-Code liegt. Für Anfänger nicht geeignet.

Funktionstaste (Function key)

Eine Taste der Tastatur, der eine besondere Funktion zugeordnet wurde, welche sie zusätzlich oder an Stelle der eigentlich vorgesehenen, auf der Taste dargestellten Funktion ausübt.

Ganzzahl (Integer number)

Eine Zahl ohne Bruchteil, d.h. ohne Ziffern rechts vom Dezimalpunkt, im Gegensatz zur reellen Zahl.

Gatter (Gate)

Logische Gatter erlauben die Durchlaß von Daten, wenn bestimmte Bedingungen erfüllt sind. Es gibt verschiedene Arten von Gattern, z.B. `OR`, `AND` und `XOR` (siehe Boolesche Algebra).

Geräusch (Noise)

Der Tongenerator des CPC6128 ermöglicht die Erzeugung von Geräuscheffekten, z.B. Explosionen.

Gleitkommazahl (Floating point number)

Reelle Zahl, deren Dezimalpunkt beim Speichern und Bearbeiten verschoben werden kann. Dieses Vorgehen empfiehlt sich besonders beim Umgang mit sehr großen Zahlen.

Graphik (Graphics)

Teil der Bildschirmdarstellung auf dem Computer, der nichts mit der Darstellung von Text zu tun hat. Graphik bedeutet das Zeichnen und Plotten von Linien, Kreisen, Mustern usw.

Graphikcursor (Graphics cursor)

Ähnlich dem Textcursor, jedoch für den Graphikbildschirm. Beim CPC6128 ist er zwar unsichtbar, aber zum Positionieren von Graphiken unerläßlich. Nicht mit graphischen Symbolen verwechseln, die Teil des Zeichensatzes sind und an der Stelle ausgegeben werden, wo sich der Textcursor befindet.

Graphikmodus (Graphics mode)

Frühere Mikrocomputer mußten in den jeweiligen Modus zur Verarbeitung von entweder Text oder Graphik umgeschaltet werden. Moderne Personal-Computer können gleichzeitig mit Text und Graphik arbeiten.

Graphik-Tablett (Graphics tablet)

Ein Gerät, das die Koordinatenpunkte eines Bildes oder einer Zeichnung zur Verarbeitung an den Computer weitergibt. Eine Art Analog/Digitalwandler.

Graphisches Symbol (Graphics character)

Besonderes Zeichen oder Muster, das für die Erstellung von Bildern sehr nützlich ist.

Halbbyte (Nibble)

Aus vier Bits bestehender Ausdruck. Jede Ziffer im hexadezimalen Ausdruck &F6 stellt ein Halbbyte dar.

Hard copy

Papierausdruck von einem Programm oder Text, oder von einer graphischen Darstellung. Der vorübergehende Ausdruck auf dem Bildschirm heißt Softcopy.

Hardware

Die elektronischen und mechanischen Teile eines Computersystems, alles was nicht unter Software oder Firmware fällt.

Hexadezimalsystem (Hexadecimal system)

Zahlensystem zur Basis 16. In Programmen für den 6128 werden Hexadezimalzahlen durch ein vorgestelltes & oder H gekennzeichnet, z.B. &FF = (dezimal) 255 (siehe Teil 1 des 9. Kapitels).

Hex-Datei (Hex file)

ASCII-Darstellung einer Befehls- oder Maschinencode-Datei.

Höchstwertiges Bit (Most significant bit, MSB)

Das in einer Binärzahl am weitesten links stehende Bit.

Höhere Programmiersprache (Higher level language)

Sprache wie BASIC, die der menschlichen Sprache angelehnt ist, so daß sie sehr leicht interpretiert werden kann. Langsamer als Maschinencode-ähnliche Sprachen, dafür viel leichter zu verstehen.

IEEE-488

Eine der Standardschnittstellen für den Anschluß von Geräten an einen Mikrocomputer. Ähnlich der Centronics-Parallelschnittstelle, jedoch nicht völlig mit ihr kompatibel.

Informationstechnik (Information technology)

Alles was mit elektronischer Nachrichtenverarbeitung und Kommunikation zu tun hat: Textverarbeitung, Datenübertragung, Btx, usw.

Inhaltsverzeichnis (Directory)

Diskettenabschnitt, der für jede auf der Diskette gespeicherte Datei einen Eintrag enthält.

Initialisieren (Initialise)

Ein System einschalten, oder vor dem Start des Hauptprogramms Variablen mit einem bestimmten Wert belegen, d.h. die Dimension eines Arrays festlegen, alle Variablen als Ganzzahlen deklarieren o.ä.

Integrierte Schaltung (Integrated circuit)

Eine Sammlung winzig kleiner elektronischer Schaltungselemente auf einem einzigen Siliziumplättchen (siehe Chip).

Intelligentes Datensichtgerät (Intelligent terminal)

Ein Datensichtgerät, bei dem außer der Behandlung von Eingabe und Ausgabe für den Computer auch eine lokale Bearbeitung möglich ist, wenn das Gerät 'offline' ist.

Interaktiv (Interactive)

Dialog zwischen Anwender und Programm, wobei der Anwender aufgefordert wird, etwas einzugeben. Das kann die Steuerung eines Raumschiffs in einem Automatenspiel oder die Beantwortung von Fragen in einem Lehrprogramm sein. Die Handlung des Anwenders beeinflußt den Ablauf des Programms unmittelbar.

Interpreter

Element der Systemsoftware, das die höhere Programmiersprache für die CPU in Maschinensprache verwandelt. Über die Tastatur eingegebene BASIC-Befehle werden also vom Interpreter in eine für den Computer verständliche Form gesetzt.

Iteration

Ein Prinzip der Datenverarbeitung. Der Computer zerlegt alle Aufgaben in einfache kleine Schritte, die von der CPU bewältigt werden können. Dazu muß er zwischen vielen einfachen Elementen hin-und herwandern, bis eine gegebene Bedingung erfüllt ist.

Joystick

Auch Steuerknüppel genannt. Ein Eingabegerät, das im wesentlichen die Funktionen der Cursortasten übernimmt. Damit läßt sich in Spielen leichter und schneller steuern.

K

Die Abkürzung für Kilo (1000). In der Datenverarbeitung als Kurzform für Kilobyte verwendet. Eine Kilobyte sind 1024 Byte, ein Binärwert (2 hoch 10), der dem Dezimalwert 1000 am nächsten kommt.

Kaltstart (Cold start)

Booten und Initialisieren eines Betriebssystems. CP/M wird durch den Befehl `|CPM` kaltgestartet.

Knoten (Node)

Speichereinheit im LOGO-Arbeitsbereich, die normalerweise 4 Bytes umfaßt.

Kommando (Command)

Eine Anweisung in einem Programm.

Konsolmodus (Console mode)

CP/M-Direktmodus. Auf dem Bildschirm erscheint `A>`, d.h. das System erwartet die Eingabe eines CP/M- oder Dienstprogramm-Befehls.

Korrumpierung (Corruption)

Unerwünschte und möglicherweise irreversible Zerstörung oder Veränderung des Inhalts einer Datei oder des Speichers.

Künstliche Intelligenz (Artificial intelligence, AI)

Eine Konstruktion in der Programmtechnik, die es dem Programm ermöglicht, aus vorheriger Erfahrung zu lernen.

Leistungstest (Benchmark)

Eine Standardaufgabe für die Ermittlung von Geschwindigkeit, Leistungsfähigkeit und Genauigkeit verschiedener Computer, z.B. Wurzel aus 99.999 zum Quadrat.

Lichtgriffel (Light pen)

Eine andere Eingabemöglichkeit, bei der ein Stift oder 'Zauberstab' benutzt wird.

Lisp

Kurzform für LISt Processor Language. Eine andere höhere Programmiersprache.

Logik (Logic)

Die elektronischen Komponenten, die die elementaren logischen Operationen und Funktionen ausführen, aus denen jeder Vorgang im Computer letztendlich besteht.

Logisches Gerät (Logical device)

Ausformung eines Geräts, das sich von der physikalischen Erscheinungform unterscheiden kann. So kann beispielsweise das logische CP/M-Gerät LST der Centronicsschnittstelle oder dem Datensichtgerät zugeordnet werden.

LOGO

Eine einfach erlernbare, graphikorientierte Programmiersprache, die in Schulen häufig im Programmierunterricht verwendet wird. Der Name ist vom griechischen Wort 'logos' für 'Wort' abgeleitet.

LSI

Large Scale Integration (Hochgradige Integration). Die Entwicklung von integrierten Schaltungen, bei der immer mehr Funktionen auf immer kleineren Stückchen Silizium untergebracht werden.

Maschinenlesbar (Machine readable)

Charakterisiert die Darstellungsform von Daten, die das Lesen der Daten durch Maschinen gestattet.

Maschinencode (Machine code)

Programmiersprache, die der Mikroprozessor direkt versteht, da alle ihre Kommandos als Bitmuster dargestellt werden.

Matrix

Die Anordnung von Punkten, in der eine Zeichen'zelle' auf dem Bildschirm oder dem Druckkopf eines Matrixdruckers dargestellt wird. Wird auch in der Mathematik und Informatik zur Beschreibung von Gittern verwendet.

Maus (Mouse)

Ein Eingabegerät auf Rollen. Die Maus wird auf dem Tisch hin- und herbewegt, meist um den Cursor auf dem Bildschirm zu steuern. Ursprünglich wurde die Maus entwickelt, um die Angst vor Tastaturen zu nehmen und die Software anwenderfreundlicher zu machen.

Mensch-Maschinen-Schnittstelle (Man-machine interface)

Berührungspunkt zwischen Computer und Anwender: Tastatur, Bildschirm, usw.

Menü

Eine Auswahlliste der verschiedenen Funktionen, die ein Programm ausführen kann.

Mikroprozessor (Microprocessor)

Integrierte Schaltung im Herzen des Mikrocomputers, welche die vom BASIC-Interpreter übergebenen Anweisungen ausführt, um die verschiedenen Ausgabegeräte und Optionen zu steuern.

Modem

Kombinierter MOdulator/DEModulator, der den Anschluß des Computers an eine Telefonleitung oder sonstige serielle Datenübertragungseinrichtung, einschließlich Glasfaserkabel, ermöglicht. Siehe auch Akustikkoppler.

Monitor

Der Bildschirm eines Computerterminals. Auch Bezeichnung für ein Programm in Maschinensprache, das die elementaren Maschinenoperationen des Computers regelt.

Netz (Network)

Ein System aus zwei oder mehr Computern, die durch Leitungen oder Modems miteinander verbunden sind, um Daten oder Informationen auszutauschen.

Niedere Programmiersprache (Low level language)

Programmiersprache wie z.B. Assembler, bei der jede Anweisung einer Maschinencode-Anweisung entspricht.

Niederwertigstes Bit (Least significant bit)

In einer Binärzahl das am weitesten rechts stehende Bit.

Numerisches Tastenfeld (Numeric keypad)

Bereich der Tastatur, in dem Zahlentasten zusammengruppiert sind, um numerische Daten leichter eintippen zu können. Beim 6128 können diese Tasten zusätzlich als Funktionstasten definiert werden.

OCR

Optical Character Recognition (Optische Zeichenerkennung). System, das gedruckte oder handgeschriebene Zeichen durch ein optisches Lesegerät erfaßt und direkt in Daten umwandelt, die für den Computer lesbar sind.

Off line

Peripheres Gerät - meist ein Datensichtgerät oder ein Drucker - das nicht an die Hauptprozessoreinheit angeschlossen ist, auf das also nicht zugegriffen werden kann.

Oktalsystem (Octal)

Zahlensystem mit der Basis 8, in dem jede Ziffer (0 bis 7) aus drei Bits gebildet wird.

On line

Gegenteil von 'Off line'.

Operator

Der Teil eines arithmetischen Ausdrucks, der eine Zahl mit einer anderen verknüpft, z.B. + - * .

Page Zero

Bezieht sich auf einen Speicherbereich in der CP/M-Umgebung, im Adressbereich zwischen &0000 und &0100, der wichtige System-Parameter enthält.

Paperware

Unter Paperware wird das gedruckte Begleitmaterial zu einem Produkt (z.B. dieses Handbuch) zusammengefaßt.

Parallele Schnittstelle (Parallel interface)

Die CPC6128-Druckerschnittstelle unterstützt einen parallelen Drucker, d.h. jeder Datenausgang des Bus ist mit einem entsprechenden Dateneingang des Druckers verbunden. Die Datenübertragung über eine Parallelschnittstelle läuft viel schneller als über einer serielle Schnittstelle, da bei serieller Verarbeitung die Daten erst in eine Bytedarstellung mit Synchronisationsinformation umgeformt werden müssen.

Pascal

Höhere, systematisch strukturierte Programmiersprache, bei der ein Programm erst compiliert werden muß, bevor es ablaufen kann - dafür läuft es dann sehr schnell. Im allgemeinen die Sprache, die sich ein eifriger BASIC-Schüler als nächstes aneignet.

PEEK

BASIC-Funktion, die Einblick in den Computerspeicher ermöglicht und den an einer bestimmten Stelle befindlichen Wert ausgibt.

Peripheres Gerät (Peripheral)

Drucker, Modems, Joysticks, Kassettenlaufwerke - alles was an den Computer angeschlossen werden kann, um seine Fähigkeiten zu erweitern.

Physikalisches Gerät (Physical device)

Ein tatsächlich existierendes Hardware-Gerät. Physikalische Geräte können logischen Geräten zugeordnet werden.

Pixel

Der kleinste Teilbereich des Bildschirms, auf den durch die Harware bedingt zugegriffen werden kann.

Plotter

Eine Abwandlung des Druckers, die Strichzeichnungen mit Zeichenstiften erstellt, und im Gegensatz zu einem Drucker ohne Druckkopf arbeitet.

POKE

Eine BASIC-Anweisung, mit der ein Wert an eine bestimmte Stelle des Speichers geschrieben werden kann.

Port

Eine adressierbare Stelle auf einer Schnittstelle (Interface) für die Eingabe oder Ausgabe von Daten.

Portabilität (Portability)

Die Möglichkeit, Software auf Computern verschiedener Hersteller laufen zu lassen. Meist wird dies durch kompatible Betriebssysteme, wie CP/M von Digital Reasearch, erreicht.

Primitive (Primitives)

Prozeduren, Funktionen oder Befehle, aus denen die LOGO-Sprache besteht.

Programm

Eine Kombination von Befehlen, die den Computer veranlassen, eine Aufgabe zu erfüllen. Hierunter fällt alles von der einfachen Maschinencode-'Routine' bis zum vollständigen Anwenderprogramm, wie z.B. Textverarbeitung.

Programmentwicklung (Software engineering)

Ein Ausdruck für Programmierung mittels Strukturierung und Planung statt mit willkürlichen Techniken.

Programmiersprache (Programming language)

Mittel zum Schreiben eines Programms. Eine Programmiersprache zeichnet sich durch strikte Regeln für die Gebrauch der Wörter und Zahlen und für die Einhaltung der richtigen Reihenfolge aus.

PROM

Programmable Read Only Memory (Programmierbarer, nur lesbarer Speicher). Eine integrierte Speicherschaltung, die, einmal beschrieben, nicht mehr gelöscht werden kann. (Siehe EPROM)

Prompt

Bereitschaftszeichen. Kurze Meldung oder Zeichenfolge, die den Anwender zur Eingabe einer Anweisung auffordert. Das CP/M-Prompt ist >, bei Dr.LOGO heißt es ? .

Prozedur (Procedure)

Eine Reihe von Ausdrücken oder Programm-Anweisungen, die die Ausführung einer bestimmten Aufgabe bewirken.

Puffer (Buffer)

Zwischenspeicher. Speicherbereich, der Daten während einer Informationsübertragung vorübergehend aufnimmt.

Punktmatrix (Dot matrix)

Rechteckiges Gitter, auf dem ein Zeichen durch Auswahl einzelner Felder (Punkte) dargestellt werden kann.

Quittungsbetrieb (Handshaking)

Eine Folge elektronischer Signale, die den Datenaustausch zwischen Computer und Peripheriegerät, bzw. zwischen zwei Computern in Gang setzen, prüfen und synchronisieren.

RAM

Random Access Memory. Speichereinheit, die mit der internen Schaltung des Computers während eines normalen Programmablaufs sowohl gelesen als auch beschrieben werden kann.

Rasterverfahren (raster scan)

Darstellungstechnik auf Bildschirmen. Das Bild entsteht durch horizontales Abtasten des Bildschirms mit einem Elektronenstrahl, der dabei bestimmte Punkte aufleuchten läßt.

Read only, R/O

Bezeichnet Disketten, Diskettendateien oder -laufwerke, die nicht beschrieben oder verändert werden können.

Read write, R/W

Bezeichnet Disketten, Diskettendateien oder -laufwerke, die sowohl gelesen als auch beschrieben werden können.

Reelle Zahl (Real number)

Eine Zahl mit Bruchteil, die also im Gegensatz zur Ganzzahl auch Stellen hinter dem Dezimalpunkt aufweist.

Register

Speicher in der CPU, wo Daten zwischengespeichert werden.

Rekursion (Recursion)

Reihe sich wiederholender Schritte in einem Programm oder einer Routine, wobei jede Wiederholung auf dem Ergebnis des vorangegangenen Durchgangs aufbaut.

Reserviertes Wort (Reserved word)

Ein Wort, das in einem Programm eine besondere Bedeutung hat, und nur in einem bestimmten Zusammenhang gebraucht werden darf. Z.B. wird BASIC das Wort TAG nicht als Variablennamen annehmen, da es schon für einen anderen Zweck reserviert ist.

RF Modulator

Mittel, durch das Video-Signale vom Computer verschlüsselt und zum Antenneneingang eines Fernsehgerätes gesendet werden können.

ROM

Read Only Memory (nur lesbarer Speicher). Bezieht sich im allgemeinen auf Halbleiterspeicher, die einmal beschrieben, weder gelöscht noch überschrieben werden können.

Routine

Teil des Programms, der eine Routineaufgabe erledigt. Eine Routine ist entweder Teil eines Hauptprogramms oder ein Programm, das in verschiedenen Programmen mitverwendet wird. Ein Programm z.B., das die Computer-Uhr in eine 12-Stunden-Uhr umgestaltet, kann als Routine betrachtet werden.

RPN

Reverse Polish Notation (umgekehrte polnische Notation). Von manchen Herstellern bevorzugte Darstellung arithmetischer Operationen, bei der die Operatoren (+, -, *, / usw.) hinter den zugehörigen Werten erscheinen.

RS232C

Standardnorm für serielle Datenübertragungsschnittstellen. Geräte an beiden Enden der Übertragungsstrecke müssen an die spezifischen Bedingungen der benutzten RS232-Schnittstelle angepaßt sein. Die entsprechende Standardnorm für parallele Datenübertragung ist die Centronics-Parallelschnittstelle.

Schildkröte (Turtle)

Graphiksymbol in der Form einer Pfeilspitze, das bei Dr.LOGO als Graphik-Cursor dient.

Schildkröten-Graphik (Turtle graphics)

Graphische Darstellung, die durch die Spur der umherwandernden Schildkröte auf dem Bildschirm erzeugt wird.

Schildkrötenschritt (Turtle step)

Kürzeste Entfernung, die eine Schildkröte zurücklegen kann. In der Regel ist dies ein Pixel.

Schleife (Loop)

Ein Prozeß im Programm, der solange wiederholt wird, bis eine bestimmte Bedingung eingetreten ist.

Schlüsselfertig (Turn key)

Bezeichnung für ein Programm, das nach dem Laden des Betriebssystems automatisch abläuft.

Schlüsselwort (Keyword)

Ein Wort, dessen Gebrauch in einem Programm oder einer Computersprache für eine spezielle Funktion oder einen Befehl reserviert ist.

Schnittstelle (Interface)

Ein- und Ausgang des Computers. Als Schnittstellen beim CPC6128 gelten die Tastatur (Eingabe), der Bildschirm (Ausgabe) und die Vorrichtungen für den Anschluß der Peripheriegeräte an die verschiedenen Buchsen.

Schreibmaschinentastatur

Begriff für die Computertastatur, bei der die Tasten wie auf einer Schreibmaschine angeordnet sind.

Schreibschutz (Write protection)

Eine Sicherungsvorrichtung zum Schutz vor dem Überschreiben einer Diskette oder Diskettendatei. Eine schreibgeschützte Diskette oder Datei ist nur lesbar.

Scrolling

Vorgang, bei dem die Zeilen des Bildschirms nach oben wegrollen, um in der untersten Zeile Platz für neue Eingaben oder Ausgaben zu machen.

Sektor (Sector)

Datenblock auf einer Diskette. Im Schneider Diskettensystem umfaßt ein Sektor 512 Bytes.

Serielle Schnittstelle (Serial interface)

Bezieht sich fast immer auf die RS232-Schnittstelle, obwohl es noch andere serielle Schnittstellen für die serielle Datenübertragung gibt.

Simulation

Eine Technik, lebensnahe Aktionen mit Hilfe des Computers nachzuahmen, z.B. Flug-, Fahrsimulation usw.

Software

Programme auf Diskette, Kassette, ROM usw.

Speicher (Memory)

Der Bereich im Computer, wo Informationen und Daten säuberlich geordnet zwischengelagert werden, so daß auf jedes Element einzeln zugegriffen werden kann. Zum Speicher gehören der RAM (Random Access Memory), der beliebig beschrieben und gelesen werden kann, und der ROM (Read Only Memory), der nur gelesen, aber nicht beschrieben werden kann. Disketten und Magnetbänder sind Beispiele für Massenspeicher.

Speicherbelegungsplan (Memory map)

Darstellung des Speichers, aus der die verschiedenen Adressbereiche und die Speicherzuordnung für die einzelnen Funktionen, wie Bildschirm und Diskettenbetriebssystem, hervorgehen.

Spracherkennung (Speech recognition)

Umwandlung gesprochener Wörter in maschinenlesbare Anweisungen.

Sprachsynthese (Speech synthesis)

Erzeugung künstlicher Sprache mit Hilfe von Hard- und Software.

Sprite

Ein von spezieller Hardware oder Software erzeugtes Zeichen, das sich frei auf dem Bildschirm bewegen und beliebig auftauchen oder wieder verschwinden kann.

Spuren (Tracks)

Konzentrische Ringe auf einer Diskette. Jede Spur ist in mehrere Sektoren eingeteilt. Beim Formatieren wird die Diskette in Spuren und Sektoren eingeteilt.

Standardannahme (default)

Der Wert, den der Computer einsetzt, wenn der Benutzer nichts anderes angibt. Wenn beispielsweise CP/M gestartet wird, nimmt der Computer Laufwerk A: als Standard-Laufwerk an.

Stapel (Stack)

Ein Speicherbereich, in dem Informationen gestapelt werden. Es kann immer nur das 'obenliegende', d.h. zuletzt eingelesene Element gelesen werden.

Steckmodul (Cartridge)

Eine speziell verpackte Speichereinheit mit integrierten Schaltungen, die Software enthält und direkt in dafür vorgesehene Buchsen des Computers gesteckt werden können. Diese Software wird schneller geladen und läuft auch schneller als Diskettensoftware, ist jedoch wesentlich teurer.

Steuerknüppel

Siehe Joystick

Stream

Das Gerät, über das der Computer Informationen ausgibt, z.B. Bildschirm, Drucker oder Diskette.

Strichcode (Bar code)

Beispiele befinden sich auf vielen Lebensmittelpackungen. Ein Code aus Linien unterschiedlicher Stärke, der mit einem Lesegerät erfaßt und vom Computer ausgewertet werden kann.

String

Aus einer Zeichenfolge bestehende Daten, die nicht als numerisch betrachtet werden können. Selbst wenn sie nur aus numerischen Zeichen bestehen, gelten sie nicht als numerisch, solange sie nicht durch ein entsprechendes Umwandlungskommando einer numerischen Variablen zugewiesen worden sind.

Strukturierte Programmierung (Structured programming)

Art der Programmierung, die logischen Aufbau und Übersichtlichkeit anstrebt. Im Ergebnis verlaufen diese Programme in klaren Schritten von 'oben nach unten'.

Syntaxfehler (Syntax error)

Wenn die Regeln für die Benutzung der Schlüsselwörter und Variablen nicht eingehalten wurden, gibt BASIC diese Meldung für den Anwender aus.

Systemspuren (System tracks)

Für das CP/M-System reservierte Disketten-Spuren.

Tabellenkalkulation (Spreadsheet)

Ein Programm, bei dem Zahlen in eine Tabelle eingetragen und vom Computer berechnet werden. Wird eine Zahl geändert, so werden alle damit verbundenen Berechnungen erneut durchgeführt und das neue Ergebnis angezeigt.

Tastatur (Keyboard)

Mehrere Reihen von alphanumerischen Tastenschaltern, um Kommandos und andere Informationen in den Computer eingeben zu können.

Terminal

Ein Texteingabegerät, bestehend aus einer Tastatur mit Bildschirm oder Fernschreiber.

Tongenerator (Sound generator)

Teil eines Computers, der Töne oder Geräusche erzeugt. Kann sowohl Hardware als auch Software sein.

TPA

Transient Program Area. Speicherbereich ab Adresse &0100, wo CP/M-Anwenderprogramme und -daten gespeichert werden.

Transientes Programm (Transient program)

Ein CP/M-Dienstprogramm wie `PIP`, das in den TPA geladen und durch Eingabe seines Namens gestartet werden kann.

Trennzeichen (Separator)

Markiert die Grenze zwischen reservierten Wörtern und anderen Elementen des Programms oder zwischen Daten.

Typenrad-Drucker (Daisy-wheel printer)

Drucker, der Dokumente in Schreibmaschinenqualität ausdruckt. Die Zeichen werden durch Druck einer Type gegen ein Farbband erzeugt.

Überschreiben (Overwrite)

Löschen von Daten im Speicher durch Überschreiben mit neuen Daten.

Uhr (Clock)

Zeitmeßsystem im Computer, das den zeitlichen Ablauf und die Synchronisation seiner Vorgänge steuert. Eine Echtzeituhr gibt Zeit und Datum an.

Umgekehrte Polnische Notation

Siehe RPN

Unintelligentes Datensichtgerät (Dumb terminal)

Ein Datensichtgerät, das nur zur Datenein- und -ausgabe dient, ohne die Daten bearbeiten zu können. Alle Abbildungen auf dem Bildschirm sind reine Videodarstellungen.

Universalzeichen (Wild card character)

Die Zeichen * und ? sind Universalzeichen. Dr.LOGO unterstützt nur ?. Das Sternchen * steht für eine beliebige Anzahl von Fragezeichen ? . Diese Zeichen werden benutzt, um einen mehrdeutigen Dateinamen anzugeben. Dabei ersetzt jedes Fragezeichen einen Buchstaben oder eine Zahl im Dateinamen.

Unterprogramm (Subroutine)

Siehe Routine

Urlader (Bootstrap)

Kleines Programm im ROM (normalerweise), das den Anfangsteil eines Programms oder Betriebssystems in den Speicher lädt, so daß das Programm oder System gestartet werden kann und den Ladevorgang selbständig weiterführt.

Variable

Ein Element in einem Programm, das mit einem Namen angesprochen werden kann, dessen tatsächlicher Wert sich jedoch während des Programmablaufs ändern kann.

Vorzeichenlose Zahl (Unsigned number)

Eine Zahl, die nicht als positiv oder negativ gekennzeichnet ist.

Wahrheitstafel (Truth table)

Das Ergebnis einer logischen Operation ist entweder 'wahr' oder 'falsch', vom Computer durch 1 bzw. 0 dargestellt. Eine Aufstellung aller möglichen Ergebnisse einer logischen Operation (z.B. `IF A>B THEN C`) heißt Wahrheitstafel.

Warmstart (Warm start)

Ein Warmstart wird unter CP/M durch Betätigen der **[CONTROL]C**-Taste ausgelöst. Er reinitialisiert das Disketten-Subsystem und übergibt die Steuerung an CP/M, woraufhin Befehle eingegeben werden können.

Wild card

Siehe Universalzeichen

Zeichen (Character)

Buchstabe, Zahl oder graphisches Symbol, das im Computer dargestellt und von ihm ausgegeben werden kann.

Zeichenfolge (String)

Eine Folge von Zeichen, die als Einheit gespeichert oder bearbeitet werden kann, z.B. ein Wort oder eine Kette von Worten.

Zeichensatz (Character set)

Alle Buchstaben, Zahlen und Symbole, die in einem Computer oder auf einem Drucker zur Verfügung stehen. Wenn ein Zeichen im Computer vorhanden ist, bedeutet dies nicht unbedingt, daß es auch zum Zeichensatz des Druckers gehört.

Zeilennummer (Line number)

Bei BASIC und anderen Sprachen sind die Anweisungen durch Zeilennummern geordnet.

Zufallszahl (Random number)

Eine vom Computer erzeugte Zahl, die weder wiederholbar noch vorraussagbar ist.

Anhang 3
6 Programme für Computerspiele

Ausbrechen

Einfach, aber macht süchtig! Für einen Spieler gegen den Computer. Mit Tasten oder Joystick.

```
10 'BUSTOUT by ALEXANDER MARTIN
20 'copyright (c) AMSOFT 1984
30 '
40 MODE 1:BORDER 1:INK 0,1:INK 1,26:INK 2,24:INK 3,6
50 SPEED KEY 15,2
60 ENV 1,1,18,0,11,0,10
70 ENT 1,10,2,2
80 ENV 3,1,0,16,5,-3,2
90 ENV 2,5,3,3,1,-21,22,9,-3,2
100 ENT -2,10,2,2,5,-7,1,2,11,3,2,-4,8
110 '
120 '
130 MOVE 30,32:DRAWR 0,400,1:MOVE 610,32:DRAWR 0,400,1
140 PEN 3:LOCATE 3,1:PRINT STRING$(36,143)
150 PEN 2:LOCATE 3,2:PRINT STRING$(36,143)
160 PEN 1:FOR r=5 TO 6:LOCATE 3,r:PRINT STRING$(36,143)
    :NEXT r
170 bx=9
180 lives=5:score=0
190 PEN 1:GOSUB 680:CLEAR INPUT
200 IF INKEY$<>CHR$(32) AND JOY(0)<16 THEN 200
210 LOCATE 11,23:PRINT SPACE$(20):LOCATE 1,24:PRINT SPA
    CE$(40);
220 GOSUB 690:GOSUB 660:GOTO 280
230 '
240 '
250 LOCATE bx,24:PRINT"  ";STRING$(4,131);"  ":RETURN
260 '
270 '
280 xa=1:ya=1:IF INT(RND*2)=1 THEN xa=-xa
290 PEN 1:GOSUB 250
300 ORIGIN 0,400
310 x=bx+4:y=11:x1=x:y1=y
320 '
330 '
340 x1=x+xa:y1=y+ya
```

```
350 IF x1=3 OR x1=38 THEN xa=-xa
360 GOSUB 540
370 IF y1=24 AND x1>bx+1 AND x1<bx+6 THEN ya=-ya:y1=y1-
    2:SOUND 130,44,8,7,1,1:a=((x>bx+5)OR(x<bx+2)):IF a=
    -1 THEN xa=xa*a:x1=x1+xa:y1=y1+1
380 IF y1=25 THEN LOCATE x,y:PRINT" ":GOTO 500
390 GOSUB 250
400 t=TEST((16*x1)-1,-(16*y1)-1)
410 IF t<>0 THEN ya=-ya:xz=x1:yz=y1:y1=y1+ya:GOSUB 590:
    IF t=2 THEN score=score+10:GOSUB 660
420 IF t=3 THEN score=score+20:GOSUB 660
430 IF t=1 THEN score=score+5:GOSUB 660
440 IF y1=1 THEN ya=1
450 LOCATE x,y:PRINT " ":LOCATE x1,y1:PRINT CHR$(233):x
    =x1:y=y1
460 IF y=1 OR x=3 OR x=38 THEN SOUND 129,78,8,7,1,1
470 GOTO 340
480 '
490 '
500 lives=lives-1:SOUND 132,19,46,12,2,2:IF lives=0 THE
    N GOTO 620
510 GOSUB 660:GOTO 280
520 '
530 '
540 IF (INKEY(8)=0 OR INKEY(74)=0) AND bx>2 THEN bx=bx-
    2:RETURN
550 IF (INKEY(1)=0 OR INKEY(75)=0) AND bx<32 THEN bx=bx
    +2:RETURN
560 RETURN
570 '
580 '
590 LOCATE xz,yz:PRINT " ":RETURN
600 '
610 '
620 IF score>=hiscore THEN hiscore=score
630 GOSUB 660:score=0:lives=5:GOTO 130
640 '
650 '
660 SOUND 130,0,20,13,3,0,31:LOCATE 1,25:PRINT TAB(4)"H
    ISCORE";hiscore;
670 LOCATE 18,25:PRINT"SCORE";score:LOCATE 30,25:PRINT"
    LIVES";lives:RETURN
680 LOCATE 11,23:PRINT"Druecke Leertaste zum Start":RET
    URN
690 LOCATE 1,25:PRINT SPACE$(40);:RETURN
```

Bomber

Eine Variation zum klassischen Thema. Für einen Spieler gegen den Computer. Nur mit Tasten.

```
10 'BOMBER von DAVE TOWN
20 'copyright (c) AMSOFT 1984
30 '
40 MODE 1:CLS:INK 0,0:BORDER 0:INK 1,18:INK 2,6:INK 3,4
   :INK 5,15:INK 6,2:INK 7,24:INK 8,8:INK 9,26:INK 10,1
   0:INK 11,20:INK 12,12:INK 13,16:INK 14,14:INK 15,21
50 SYMBOL AFTER 240:SYMBOL 241,&40,&60,&70,&7F,&7F,&3F,
   &7,&0:SYMBOL 242,&0,&32,&7A,&FE,&FA,&F2,&E0,&0
60 score=0:hiscore=0:plane$=CHR$(241)+CHR$(242):x=2:y=2
   :drop=0:a=2:b=2
70 GOSUB 480
80 CLS
90 PEN 2:LOCATE 1,15:INPUT"Schwierigkeitsgrad angeben:
   0 (Profil) bis 5 (Anfaenger) : ",skill
100 IF skill<0 OR skill>5 GOTO 90
110 skill=skill+10
120 LOCATE 1,15:PRINT CHR$(18);:LOCATE 1,15:INPUT"Gesch
    windigkeit angeben: (schnell) bis 100 (langsam) : "
    ,rate
130 IF rate>100 OR rate<0 GOTO 120
140 '
150 'Gebaeude
160 '
170 MODE 0:FOR base=5 TO 15:FOR height=21 TO INT(RND(1)
    *8+skill) STEP -1:LOCATE base,height:PEN base-2:PRI
    NT CHR$(143)+CHR$(8)+CHR$(11)+CHR$(244);:NEXT :NEXT
180 PLOT 0,20,4:DRAW 640,20,4
190 LOCATE 1,25:PEN 2:PRINT"STAND";score;:LOCATE 13,25:
    PRINT"REK";hiscore;
200 '
210 'Spiel
220 '
230 LOCATE x-1,y:PRINT"   ";
240 PEN 1:LOCATE x,y:PRINT plane$;:PEN 2
250 IF y=21 AND x=15 THEN GOTO 290:ELSE GOTO 340
260 '
270 'Gelandet
280 '
290 FOR c=0 TO 1000:NEXT
```

```
300 score=score+100-(skill*2):skill=skill-1:x=2:y=2:a=2
    :b=2:drop=0
310 IF skill<10 THEN skill=10:rate=rate-20
320 IF rate<0 THEN rate=0
330 GOTO 150
340 FOR c=0 TO rate:NEXT
350 x=x+1
360 IF x=18 THEN LOCATE x-1,y:PRINT CHR$(18);:x=2:y=y+1
    :LOCATE x,y:PEN 1:PRINT plane$;:PEN 2
370 a$=INKEY$:IF a$=" " AND drop=0 THEN drop=1:b=y+2:a=
    x
380 IF y=21 THEN drop=0
390 IF drop=1 THEN LOCATE a,b:PRINT CHR$(252);:LOCATE a
    ,b-1:PRINT" ";:b=b+1:IF b>21 THEN LOCATE a,b:PRINT"
     ";:LOCATE a,b-1:PRINT" ";:a=0:b=0:drop=0:SOUND 3,4
    000,10,12,0,0,10
400 ga=(a-0.5)*32:gb=400-(b*16):bomb=TEST(ga,gb)
410 IF bomb>0 THEN GOTO 670
420 gx=((x+1.5)*32):gy=408-(y*16):crash=TEST(gx,gy)
430 IF crash>0 GOTO 570
440 GOTO 230
450 '
460 'Anweisungen
470 '
480 LOCATE 1,2:PEN 1:PRINT"Sie steuern ein Flugzeug ueb
    er einer verlassenen Stadt und muessen alle Gebaeud
    e zerstoeren, um landen und tanken zu koennen. Ihr
    Flugzeug bewegt sich von links nach rechts ueber de
    n Bildschirm.";:PRINT
490 PRINT:PRINT"Wenn Sie die rechte Seite erreichen, ta
    ucht Ihr Flugzeug EINE ZEILE TIEFER am linken Rand
    des Bildschirms wieder auf. Durch Druck auf die LEE
    RTASTE koennen Sie eine Bombe aus Ihrem unbegrenzte
    n Vorrat abwerfen";:PRINT
500 PRINT:PRINT"Jedesmal, wenn Sie landen, werden entwe
    der die Gebaeude hoeher oder Ihre Geschwindigkeit s
    teigt.";:PRINT:PRINT:PRINT"WENN SIE EINE BOMBE ABGE
    WORFEN HABEN, MUESSEN SIE WARTEN, BIS SIE EXPLODIER
    T IST, BEVOR SIE EINE WEITERE ABWERFEN KOENNEN!!!";
510 PEN 2:LOCATE 1,24:PRINT:PRINT"START mit beliebiger
    Taste.";
520 a$=INKEY$:IF a$="" GOTO 520
530 RETURN
540 '
550 'Zusammenstoss
```

```
560 '
570 LOCATE x-1,y:PRINT CHR$(32)+CHR$(32)+CHR$(32)+CHR$(
    253)+CHR$(8)+CHR$(238)+CHR$(8);
580 FOR t=1 TO 10:SOUND 7,4000,5,15,0,0,5:PEN t:PRINT C
    HR$(253)+CHR$(8)+CHR$(238)+CHR$(8)+CHR$(32)+CHR$(8)
    ;:FOR tm=0 TO 50:NEXT:NEXT:PEN 2
590 CLS:LOCATE 1,5:PRINT"Punkte";score;
600 IF score>hiscore THEN hiscore=score:LOCATE 1,8:PRIN
    T"NEUER REKORD!!";
610 score=0:LOCATE 1,12:PRINT"Neuer Start mit R";
620 a$=INKEY$:IF a$="r" OR a$="R" GOTO 630 ELSE GOTO 62
    0
630 PEN 1:MODE 1:x=2:y=2:a=2:b=2:GOTO 90
640 '
650 'Bombardierte Gebaeude
660 '
670 LOCATE a,b-1:PRINT" "+CHR$(8);:PEN 4:FOR tr=1 TO IN
    T(RND(1)*3)+1:score=score+5:SOUND 3,4000,10,12,0,0,
    10:LOCATE a,b:FOR t=0 TO 4:PRINT CHR$(253)+CHR$(8)+
    CHR$(32)+CHR$(8);:NEXT:b=b+1
680 IF b=24 THEN b=b-1
690 NEXT
700 LOCATE 6,25:PRINT score;:drop=0:a=x:b=y:GOTO 230
```

Telly tennis

Dieser "Urvater" aller Computerspiele macht auch heute noch Spaß. Sie können alleine gegen den Computer oder zu zweit spielen. Sie steuern mit Tasten oder Joystick(s).

```
10 'TELLY TENNIS von DAVID RADISIC
20 'copyright (c) AMSOFT 1985
30 '
40 DEFINT a-z
50 comp=1
60 ENV 1,=11,20,=9,5000
70 MODE 1:INK 0,10:BORDER 10:INK 1,26:INK 2,18:INK 3,0
80 GOSUB 710
90 GOSUB 150
100 GOSUB 330
110 GOSUB 420
120 LOCATE 13,1:PRINT USING"#### ";score1;
130 LOCATE 35,1:PRINT USING"#### ";score2;
140 GOTO 100
150 PEN 2
160 x(1)=3:y(1)=5
170 x(2)=37:y(2)=22
180 edge$=CHR$(233):edge2$=STRING$(2,207)
190 LOCATE 1,3:PRINT STRING$(39,edge$):PRINT STRING$(39
    ,edge$)
200 FOR i=1 TO 19
210 PRINT edge2$;TAB(38);edge2$
220 NEXT
230 PRINT STRING$(39,edge$):PRINT STRING$(39,edge$);
240 WINDOW #1,3,37,5,23
250 CLS#1
260 SYMBOL 240,0,60,126,126,126,126,60,0
270 bat$="|"+CHR$(8)+CHR$(10)+"|"
280 clr$=" "+CHR$(8)+CHR$(10)+" "
290 ball$=CHR$(240)
300 PEN 3
310 LOCATE 2,1:PRINT"Spieler 1 :    0";:LOCATE 24,1:PRI
    NT"Spieler 2 :    0";
320 RETURN
330 n=INT(RND*2):CLS #1:scored=0
340 PEN 3
350 FOR i=1 TO 2:LOCATE x(i),y(i):PRINT bat$;:NEXT
360 ON n GOTO 390
```

```
370 xb=21:dx=1
380 GOTO 400
390 xb=19:dx=-1
400 yb=12:dy=INT(RND*3)-1
410 RETURN
420 GOSUB 600
430 oxb=xb:oyb=yb
440 GOSUB 500
450 IF note>0 THEN SOUND 129,note,50,15,1
460 LOCATE oxb,oyb:PRINT" ";
470 LOCATE xb,yb:PRINT ball$
480 IF scored=0 THEN 420
490 RETURN
500 LOCATE xb+dx,yb+dy:ch$=COPYCHR$(#0)
510 note=0
520 IF ch$=" " THEN xb=xb+dx:yb=yb+dy:RETURN
530 IF ch$="I" THEN dx=2-dx-2:dy=INT(RND*3)-1:note=200:
    RETURN
540 IF ch$=LEFT$(edge2$,1) THEN 570
550 IF ch$=edge$ THEN dy=2-dy-2:note=250
560 RETURN
570 IF dx>0 THEN score1=score1+1 ELSE score2=score2+1
580 scored=1:note=2000
590 RETURN
600 p(1)=(INKEY(69)>=0)+(INKEY(72)>=0)+ABS((INKEY(71)>=
    0)+(INKEY(73)>=0))*2
610 IF comp=1 THEN p(2)=ABS(y(2)<yb)*2+(y(2)>yb):GOTO 6
    30
620 p(2)=(INKEY(4)>=0)+(INKEY(48)>=0)+ABS((INKEY(5)>=0)
    +(INKEY(49)>=0))*2
630 PEN 3
640 FOR i=1 TO 2
650 LOCATE x(i),y(i)+p(i):ch$=COPYCHR$(#0)
660 IF ch$=" " THEN LOCATE x(i),y(i):PRINT clr$;:y(i)=y
    (i)+ROUND(p(i)/2)
670 LOCATE x(i),y(i):PRINT bat$;
680 NEXT
690 PEN 1
700 RETURN
710 PEN 2:PRINT:PRINT TAB(15)"Ping-Pong":PRINT TAB(15)"
    ---------"
720 PEN 3:PRINT:PRINT TAB(14)"Steuerung :
730 PRINT:PRINT:PEN 1
740 PRINT" Spieler 1       Spieler 2       Richtung":PR
    INT
```

```
750 PRINT"    A              6              AUF"
760 PRINT"    Z              3              AB":PRINT
770 PEN 3:PRINT:PRINT TAB(14)"Oder Joystick"
780 PRINT:PRINT:PRINT:PRINT
790 PEN 2
800 PRINT TAB(6)"Waehlen Sie <1> oder <2> Spieler"
810 i$=INKEY$:IF i$<>"1" AND i$<>"2" THEN 810
820 IF i$="1" THEN comp=1 ELSE comp=0
830 MODE 1:RETURN
```

Elektrozaun

Versuchen Sie, Ihren Gegner auszustechen! Nur für zwei Spieler. Mit Tasten oder Joystick(s).

```
10 'ELECTRIC FENCING von ALEXANDER MARTIN
20 'copyright (c) AMSOFT 1985
30 '
40 DEFINT a-z
50 MODE 0
60 GOSUB 980
70 GOSUB 1370
80 GOSUB 270
90 GOSUB 1520
100 GOSUB 1370
110 GOSUB 1270
120 '
130 '
140 REM start
150 IF finished THEN GOTO 100
160 GOSUB 240
170 FRAME:IF p1dir THEN GOSUB 570 ELSE FRAME:FRAME
180 FRAME:IF p2dir THEN GOSUB 620 ELSE FRAME:FRAME
190 IF p1sa=-1 THEN GOSUB 670
200 IF p2sa=-1 THEN GOSUB 720
210 GOTO 140
220 '
230 '
240 IF j THEN  380 ELSE  480
250 '
260 '
270 CLS:PEN 6
280 PRINT:PRINT"   WELCHE STEUERUNG?"
290 PRINT:PRINT:PRINT:PRINT"Druecken Sie J oder T, dann
    [RETURN]"
300 LOCATE 4,10:PRINT"JOYSTICK";TAB(5);"TASTATUR"
310 LOCATE 12,10:IF j THEN PRINT"*":ELSE PRINT" "
320 LOCATE 12,11:IF j THEN PRINT" ":ELSE PRINT"*"
330 IF NOT(INKEY(45)) THEN j=-1
340 IF NOT(INKEY(37)) THEN j=0
350 IF NOT(INKEY(18)) THEN RETURN ELSE 310
360 '
370 '
380 p1=JOY(0):p2=JOY(1)
390 p1dir=(p1 AND 1)*-1+(p1 AND 2)*0.5
```

```
400 p2dir=(p2 AND 1)*-1+(p2 AND 2)*0.5
410 IF P1 AND 16 THEN p1sa=p1sa-1:IF p1sa=-1 THEN AFTER
     15 GOSUB 770
420 IF P2 AND 16 THEN p2sa=p2sa-1:IF p2sa=-1 THEN AFTER
     15 GOSUB 770
430 IF p1sa THEN p1dir=0
440 IF p2sa THEN p2dir=0
450 RETURN
460 '
470 '
480 p2dir=((INKEY(4)=0)*1)+((INKEY(5)=0)*-1)
490 p1dir=((INKEY(69)=0)*1)+((INKEY(71)=0)*-1)
500 IF INKEY(63)=0 THEN p1sa=p1sa-1:IF p1sa=-1 THEN AFT
    ER 15 GOSUB 770
510 IF INKEY(10)=0 THEN p2sa=p2sa-1:IF p2sa=-1 THEN AFT
    ER 15 GOSUB 770
520 IF p1sa THEN p1dir=0
530 IF p2sa THEN p2dir=0
540 RETURN
550 '
560 '
570 pt=p1wp+p1dir:IF pt>25 OR pt<6 THEN RETURN ELSE p1w
    p=pt
580 p1dir=0
590 PEN 1:LOCATE 3,p1wp:CLS #3:PRINT CHR$(209);:RETURN
600 '
610 '
620 pt=p2wp+p2dir:IF pt>25 OR pt<6 THEN RETURN ELSE p2w
    p=pt
630 p2dir=0
640 PEN 2:LOCATE 18,p2wp:CLS #5:PRINT CHR$(211);:RETURN
650 '
660 '
670 PAPER #4,4:WINDOW #4,4,17,p1wp,p1wp:CLS#4:FRAME:FRA
    ME
680 PAPER #4,0:CLS#4
690 GOTO 570
700 '
710 '
720 PAPER #6,5:WINDOW #6,4,17,p2wp,p2wp:CLS#6:FRAME:FRA
    ME
730 PAPER #6,0:CLS#6
740 GOTO 620
750 '
```

```
760 '
770 pwpe=(p1wp=p2wp):IF p1sa AND NOT(p2sa) AND pwpe THE
    N p1sc=p1sc+1:SOUND 132,120,10,0,1,0:PRINT#1,a$(p1s
    c);:IF p1sc=9 THEN 860
780 IF p2sa AND NOT(p1sa) AND pwpe THEN p2sc=p2sc+1:SOU
    ND 132,100,10,0,1,0:PRINT#2,a$(p2sc);:IF p2sc=9 THE
    N 860
790 IF p1sa THEN SOUND 132,40,70,0,1,1
800 IF p2sa THEN SOUND 132,56,70,0,1,1
810 p1sa=0
820 p2sa=0
830 RETURN
840 '
850 '
860 PEN 6
870 LOCATE 6,10:PRINT"Spiel aus"
880 IF p1sc=9 THEN INK 1,2,20:INK 2,0 ELSE INK 2,6,17:I
    NK 1,0
890 SOUND 129,1000,0,12,3:SOUND 130,900,0,12,3
900 WHILE INKEY$<>"":WEND
910 t!=TIME:WHILE t!+2000>TIME:WEND
920 WHILE INKEY$="":WEND
930 CLS
940 finished=-1
950 RETURN
960 '
970 '
980 a$(0)="111101101101111"
990 a$(1)="001001001001001"
1000 a$(2)="111001111100111"
1010 a$(3)="111001111001111"
1020 a$(4)="100100101111001"
1030 a$(5)="111100111001111"
1040 a$(6)="111100111101111"
1050 a$(7)="111001001010010"
1060 a$(8)="111101111101111"
1070 a$(9)="111101111001001"
1080 FOR n=0 TO 9
1090 wielang=LEN(a$(n))
1100 FOR n2=1 TO wielang
1110 IF MID$(a$(n),n2,1)="1"THEN MID$(a$(n),n2,1)=CHR$(
     143)ELSE MID$(a$(n),n2,1)=CHR$(32)
1120 NEXT n2,n
1130 '
```

```
1140 '
1150 b$="Elektrozaun"
1160 c$=CHR$(32)+CHR$(164)+" Alexander Martin"
1170 ENV 1,=9,2000:ENT -1,6,3,1
1180 ENV 2,127,0,0,127,0,0,127,0,0,127,0,0,127,0,0
1190 ENV 3,=9,9000
1200 '
1210 '
1220 BORDER 0
1230 INK 0,12:PEN #4,1:PEN #6,2:PEN #1,1:PEN #2,2:PAPER
      #1,3:PAPER #2,3:PEN #0,6
1240 RETURN 'Damit wurden die Konstanten gesetzt
1250 '
1260 '
1270 INK 0,12:INK 1,2:INK 2,6:INK 3,13:INK 4,20:INK 5,1
     7:INK 6,20
1280 WINDOW #3,3,3,6,25:WINDOW #5,18,18,6,25
1290 WINDOW #1,3,5,1,5:WINDOW #2,16,18,1,5:WINDOW #7,1,
     20,1,5:PAPER #7,3
1300 CLS:CLS#7:PRINT#1,a$(0);:PRINT#2,a$(0);:p1sc=0:p2s
     c=0:p1wp=5:p2wp=24:p1dir=1:p2dir=1
1310 GOSUB 570:GOSUB 620
1320 SOUND 1,1000,0,12,2:SOUND 2,900,0,12,2
1330 p1sa=0:p2sa=0:finished=0
1340 RETURN 'Damit wurde das Bild erstellt
1350 '
1360 '
1370 CLS
1380 PEN 7
1390 FOR n=1 TO LEN(b$)
1400 LOCATE 2+n,10
1410 FOR n2=LEN(b$) TO n STEP-1
1420 PRINT MID$(b$,n2,1)
1430 LOCATE 2+n,10
1440 SOUND 135,20*n2,5,12,2,1
1450 NEXT n2,n
1460 SOUND 135,100,0,13,3,1,20
1470 PEN 6:PRINT:PRINT:PRINT:PRINT c$
1480 FOR n=1 TO 5000:NEXT
1490 RETURN
1500 '
1510 '
1520 IF j THEN RETURN
1530 CLS
```

```
1540 LOCATE 7,5
1550 PRINT"STEUERUNG"
1560 PRINT
1570 PRINT"  SPIELER 1   SPIELER 2"
1580 PRINT
1590 PRINT"  A        auf       6"
1600 PRINT"  Z        ab        3"
1610 PRINT"  X       Feuer      7"
1620 t!=TIME:WHILE t!+1000>TIME:WEND
1630 RETURN
```

Amthello

Ein Spiel für Denker. Versuchen Sie, die Felder Ihres Gegners einzukreisen und zu erobern, ohne Ihre eigenen Felder zu gefährden! Für einen Spieler gegen den Computer. Nur mit Tasten.

```
10 'AMTHELLO von M.J.GRIBBINS
20 'copyright (c) AMSOFT 1984
30 '
40 BORDER 14
50 CLEAR
60 MODE 1:PEN 0:PAPER 1:CLS
70 INK 0,0:INK 1,14:INK 2,18:INK 3,26
80 LOCATE 2,3:PEN 3:PRINT"A":LOCATE 3,4:PRINT"M":LOCATE
    4,5:PRINT"T":LOCATE 5,6:PRINT"H"
90 LOCATE 6,7:PRINT"E":LOCATE 7,8:PRINT"L":LOCATE 8,9:P
   RINT"L":LOCATE 9,10:PRINT"O"
100 WINDOW #1,2,39,22,25:PAPER #1,1:PEN #1,0:CLS #1
110 PEN 0
120 LOCATE #1,8,1:PRINT #1,"SCHWARZ SPIELT IMMER ZUERST"
130 LOCATE #1,1,3:PRINT #1,"DRUECKEN SIE S ODER W (SCHW
    ARZ/WEISS)"
140 B$=INKEY$:IF B$="" THEN 140
150 IF B$="W" OR  B$="w" THEN Q%=3:N%=0:GOTO 210
160 IF B$="S" OR  B$="s" THEN Q%=0:N%=3:GOTO 210
170 CLS #1:LOCATE #1,4,3
180 PRINT #1,"       NUR SCHWARZ ODER WEISS"
190 FOR T=0 TO 1000:NEXT T
200 GOTO 140
210 DIM C%(10,10),P%(9,9),C1%(8),C2%(8),CX%(9),CY%(9)
220 I1%=2:J1%=2:I2%=7:J2%=7
230 FOR I%=0 TO 9
240 C%(I%,0%)=6:C%(0,I%)=6
250 C%(9,I%)=6:C%(I%,9)=6
260 NEXT I%
270 FOR I%=1 TO 8
280 READ C1%(I%),C2%(I%)
290 FOR J%= 1 TO 8
300 READ P%(I%,J%)
310 C%(I%,J%)=6
320 NEXT J%:NEXT I%
330 C%(4,4)=3:C%(4,5)=0:C%(5,4)=0:C%(5,5)=3
```

```
340 FOR K%=1 TO 58
350 READ AR%,BR%,CR%,DR%
360 PLOT AR%,BR%:DRAW CR%,DR%,0
370 NEXT K%
380 GOSUB 1460
390 IF Q%=3 GOTO 770
400 CLS #1:INPUT #1,"  WELCHE ZEILE WAEHLEN SIE ";E%
410 IF E% <1 OR E% >8 GOTO 400
420 LOCATE #1,1,3:INPUT #1,"WELCHE SPALTE WAEHLEN SIE "
    ;D%
430 IF D% <1 OR D% >8 GOTO 420
440 IF C%(D%,E%)=6 GOTO 480
450 CLS #1:LOCATE #1,5,2:PRINT #1,"DIESES FELD IST SCHO
    N BESETZT !"
460 FOR T=1 TO 1000:NEXT T
470 GOTO 400
480 PLOT 270+(30*D%),70+(30*E%):DRAW 290+(30*D%),89+(30
    *E%),Q%
490 PLOT 290+(30*D%),70+(30*E%):DRAW 270+(30*D%),89+(30
    *E%),Q%
500 GOTO 540
510 FOR M%= 0 TO 19 STEP 2:PLOT 270+(30*D%),70+M%+(30*E
    %)
520 DRAW 290+(30*D%),70+M%+(30*E%),6:NEXT M%
530 GOTO 400
540 VRX%=0
550 FOR K%=1 TO 8
560 VR%=0:C3%=D%:C4%=E%
570 C3%=C3%+C1%(K%):C4%=C4%+C2%(K%)
580 IF C%(C3%,C4%)=N% GOTO 590 ELSE 600
590 VR%=VR%+1:GOTO 570
600 IF C%(C3%,C4%)=6 GOTO 610 ELSE 620
610 NEXT K%:GOTO 670
620 IF VR%=0 GOTO 610 ELSE 630
630 VRX%=VRX%+VR%
640 C3%=C3%-C1%(K%):C4%=C4%-C2%(K%)
650 IF C%(C3%,C4%)=6 GOTO 610 ELSE 660
660 C%(C3%,C4%)=Q%:GOTO 640
670 IF VRX%=0 GOTO 680 ELSE 710
680 CLS #1:PRINT #1," UNGUELTIGE AUSWAHL"
690 FOR T=1 TO 1000:NEXT T
700 GOTO 510
710 E%=E%:D%=D%:VRX%=VRX%
720 CLS #1:PRINT #1,"SIE SPIELTEN ZEILE NUMMER ";E%
```

```
730 PRINT #1,"          UND SPALTE NUMMER ";D%
740 LOCATE #1,2,4:PRINT #1,"DAMIT GEWINNEN SIE ";VRX%;"
    FELDER(S)"
750 C%(D%,E%)=Q%:GOSUB 1710
760 GOSUB 1460
770 CLS #1:LOCATE #1,10,2:PRINT #1,"JETZT BIN ICH DRAN!
    "
780 P%=0:VRX%=0:VRY%=0
790 IF I1%*J1%=1 AND I2%*J2%=64 GOTO 860
800 FOR K%=2 TO 7
810 IF C%(2,K%) <> 6 THEN I1%=1
820 IF C%(7,K%) <> 6 THEN I2%=8
830 IF C%(K%,2) <> 6 THEN J1%=1
840 IF C%(K%,7) <> 6 THEN J2%=8
850 NEXT K%
860 FOR I%=I1% TO I2%
870 FOR J%=J1% TO J2%
880 IF C%(I%,J%)=6 GOTO 1030
890 NEXT J%:NEXT I%
900 IF P% > 0 GOTO 1000
910 IF PAS%=1 GOTO 920 ELSE 940
920 CLS #1:PRINT #1," SACKGASSE! ICH MUSS AUCH PASSEN.
    SPIEL AUS"
930 FOR T=1 TO 1000:NEXT T:GOTO 1550
940 CLS #1:LOCATE #1,18,2:PRINT #1,"ICH PASSE"
950 GOSUB 2720
960 IF PAS%=1 GOTO 970 ELSE 990
970 CLS #1:PRINT #1,"SACKGASSE! SIE MUESSEN AUCH PASSEN
    . SPIEL AUS"
980 FOR T=1 TO 1000:NEXT T:GOTO 1550
990 GOTO 400
1000 IF LC%=0 THEN LC%=1:RANDOMIZE LC%:RL%=RND(LC%)
1010 CX1%=CX%(RL%):CX2%=CY%(RL%)
1020 GOTO 1220
1030 VRX%=0
1040 FOR K%=1 TO 8
1050 VR%=0:C3%=I%:C4%=J%
1060 C3%=C3%+C1%(K%):C4%=C4%+C2%(K%)
1070 IF C%(C3%,C4%)=Q% GOTO 1080 ELSE 1090
1080 VR%=VR%+1:GOTO 1060
1090 IF C%(C3%,C4%)=6 GOTO 1100 ELSE 1110
1100 NEXT K%:GOTO 1130
1110 IF VR%=0% GOTO 1100 ELSE 1120
1120 VRX%=VRX%+VR%:GOTO 1100
```

```
1130 IF VRX%=0 GOTO 890
1140 IF P%(I%,J%) < P% GOTO 890
1150 IF P%(I%,J%) > P% GOTO 1160 ELSE 1170
1160 P%=P%(I%,J%):VRY%=VRX%:LC%=0:CX%(0)=I%:CY%(0)=J%:G
     OTO 890
1170 IF VRY% > VRX% GOTO 890
1180 IF VRY% < VRX% GOTO 1190 ELSE 1200
1190 LC%=0:VRY%=VRX%:CX%(0)=I%:CY%(0)=J%:GOTO 890
1200 LC%=LC%+1:CX%(LC%)=I%:CY%(LC%)=J%
1210 GOTO 890
1220 CX2%=CX2%:CX1%=CX1%:VRY%=VRY%
1230 CLS #1:PRINT #1," ICH WAEHLE ZEILE NUMMER ";CX2%
1240 PRINT #1,"    UND SPALTE NUMMER ";CX1%
1250 LOCATE #1,1,4:PRINT #1,"DAMIT GEWINNE ICH ";VRY%;"
     FELDER"
1260 PLOT 270+(30*CX1%),70+(30*CX2%):DRAW 290+(30*CX1%)
     ,89+(30*CX2%),N%
1270 PLOT 290+(30*CX1%),70+(30*CX2%):DRAW 270+(30*CX1%)
     ,89+(30*CX2%),N%
1280 FOR T=1 TO 1000:NEXT T
1290 FOR K%=1 TO 8
1300 VR%=0:C3%=CX1%:C4%=CX2%
1310 C3%=C3%+C1%(K%):C4%=C4%+C2%(K%)
1320 IF C%(C3%,C4%)=Q% GOTO 1330 ELSE 1340
1330 VR%=VR%+1:GOTO 1310
1340 IF C%(C3%,C4%)=6 GOTO 1350 ELSE 1360
1350 NEXT K%:GOTO 1400
1360 IF VR%=0 GOTO 1350
1370 C3%=C3%-C1%(K%):C4%=C4%-C2%(K%)
1380 IF C%(C3%,C4%)=6 GOTO 1350
1390 C%(C3%,C4%)=N%:GOTO 1370
1400 C%(CX1%,CX2%)=N%
1410 GOSUB 2720
1420 GOSUB 1460
1430 IF PAS%=1 GOTO 1440 ELSE 1450
1440 CLS #1:PRINT #1,"     SIE MUESSEN PASSEN":FOR T=1
     TO 1000:NEXT T:GOTO 770
1450 GOTO 400
1460 FOR I%=1 TO 8
1470 FOR J%=1 TO 8
1480 FOR M%=0 TO 19 STEP 2
1490 Z%=270+(30*I%):H%=70+(30*J%):W%=H%+M%
1500 PLOT Z%,W%:DRAW Z%+20,W%,C%(I%,J%)
1510 NEXT M%:NEXT J%:NEXT I%
```

```
1520 X%=X%+1
1530 IF X%=61 GOTO 1550
1540 RETURN
1550 CQ%=0:CN%=0
1560 FOR I%=1 TO 8
1570 FOR J%=1 TO 8
1580 IF C%(I%,J%)=Q% THEN CQ%=CQ%+1
1590 IF C%(I%,J%)=N% THEN CN%=CN%+1
1600 NEXT J%:NEXT I%
1610 IF CQ% > CN% GOTO 1680
1620 IF CQ%=CN% GOTO 1630 ELSE 1650
1630 CLS #1:LOCATE #1,25,2:PRINT #1,"SACKGASSE"
1640 END
1650 CLS #1:LOCATE #1,5,1:PRINT #1,"SIE HABEN ";CQ%;" F
     ELDER;ICH HABE ";CN%
1660 LOCATE #1,11,3:PRINT #1,"ICH HABE GEWONNEN!!!!"
1670 END
1680 CLS #1:LOCATE #1,5,1:PRINT #1,"SIE HABEN ";CQ%;" F
     ELDER;ICH HABE ";CN%
1690 LOCATE #1,5,3:PRINT #1,"GUT GEMACHT. SIE HABEN GEW
     ONNEN!!"
1700 END
1710 IF C%(2,2)=Q% AND (C%(3,1)=N% OR C%(1,3)=N%) GOTO
     1720 ELSE 1730
1720 P%(3,1)=1:P%(1,3)=1
1730 IF C%(7,7)=Q% AND (C%(8,6)=N% OR C%(6,8)=N%) GOTO
     1740 ELSE 1750
1740 P%(8,6)=1:P%(6,8)=1
1750 IF C%(2,7)=Q% AND (C%(1,6)=N% OR C%(3,8)=N%) GOTO
     1760 ELSE 1770
1760 P%(1,6)=1:P%(3,8)=1
1770 IF C%(7,2)=Q% AND (C%(6,1)=N% OR C%(8,3)=N%) GOTO
     1780 ELSE 1790
1780 P%(6,1)=1:P%(8,3)=1
1790 IF D%=1 OR D%=8 OR E%=1 OR E%=8 GOTO 1820
1800 IF CX1%=1 OR CX1%=8 OR CX2%=1 OR CX2%=8 GOTO 1820
1810 RETURN
1820 FOR J%=1 TO 8 STEP 7
1830 FOR I%=2 TO 7
1840 IF C%(I%,J%)=N% GOTO 1850 ELSE 1860
1850 P%(I%+1,J%)=21:P%(I%-1,J%)=21
1860 IF C%(J%,I%)=N% GOTO 1870 ELSE 1880
1870 P%(J%,I%+1)=21:P%(J%,I%-1)=21
1880 NEXT I%
```

```
1890 FOR I%=2 TO 7
1900 IF C%(I%,J%)=Q% GOTO 1910 ELSE 1920
1910 P%(I%+1,J%)=2:P%(I%-1,J%)=2
1920 IF C%(J%,I%)=Q% GOTO 1930 ELSE 1940
1930 P%(J%,I%+1)=2:P%(J%,I%-1)=2
1940 NEXT I%:NEXT J%
1950 P%(1,2)=1:P%(1,7)=1:P%(2,1)=1:P%(7,1)=1
1960 P%(2,8)=1:P%(7,8)=1:P%(8,2)=1:P%(8,7)=1
1970  FOR I%=2 TO 7
1980 IF C%(1,I%-1)=Q% AND C%(1,I%+1)=Q% THEN P%(1,I%)=2
     5
1990 IF C%(8,I%-1)=Q% AND C%(8,I%+1)=Q% THEN P%(8,I%)=2
     5
2000 IF C%(I%-1,1)=Q% AND C%(I%+1,1)=Q% THEN P%(I%,1)=2
     5
2010 IF C%(I%-1,8)=Q% AND C%(I%+1,8)=Q% THEN P%(I%,8)=2
     5
2020 NEXT I%
2030 FOR J%=1 TO 8 STEP 7
2040 FOR I%=4 TO 8
2050 IF C%(J%,I%) <> N% GOTO 2140
2060 IC%=I%-1:IF C%(J%,IC%)=6 GOTO 2140
2070 IF C%(J%,IC%)=Q% GOTO 2080 ELSE 2090
2080 IC%=IC%-1:GOTO 2070
2090 IF C%(J%,IC%)=6 GOTO 2110
2100 GOTO 2140
2110 IF IC%=0 GOTO 2140
2120 IF C%(J%,I%+1)=Q% AND C%(J%,IC%-1)=6 GOTO 2140
2130 P%(J%,IC%)=26
2140 IF C%(I%,J%) <> N% GOTO 2230
2150 IC%=I%-1:IF C%(IC%,J%)=6 GOTO 2230
2160 IF C%(IC%,J%)=Q% GOTO 2170 ELSE 2180
2170 IC%=IC%-1:GOTO 2160
2180 IF C%(IC%,J%)=6 GOTO 2200
2190 GOTO 2230
2200 IF IC%=0 GOTO 2230
2210 IF C%(I%+1,J%)=Q% AND C%(IC%-1,J%)=6 GOTO 2230
2220 P%(IC%,J%)=26
2230 NEXT I%
2240 FOR I%=1 TO 5
2250 IF C%(J%,I%) <> N% GOTO 2340
2260 IC%=I%+1:IF C%(J%,IC%)=6 GOTO 2340
2270 IF C%(J%,IC%)=Q% GOTO 2280 ELSE 2290
2280 IC%=IC%+1:GOTO 2270
```

```
2290 IF C%(J%,IC%)=6 GOTO 2310
2300 GOTO 2340
2310 IF IC%=9 GOTO 2340
2320 IF C%(J%,I%-1)=Q% AND C%(J%,IC%+1)=6 GOTO 2340
2330 P%(J%,IC%)=26
2340 IF C%(I%,J%) <> N% GOTO 2430
2350 IC%=I%+1:IF C%(IC%,J%)=6 GOTO 2430
2360 IF C%(IC%,J%)=Q% GOTO 2370 ELSE 2380
2370 IC%=IC%+1:GOTO 2360
2380 IF C%(IC%,J%)=6 GOTO 2400
2390 GOTO 2430
2400 IF IC%=9 GOTO 2430
2410 IF C%(I%-1,J%)=Q% AND C%(IC%+1,J%)=6 GOTO 2430
2420 P%(IC%,J%)=26
2430 NEXT I%:NEXT J%
2440 IF C%(1,1)=N% GOTO 2450 ELSE 2460
2450 FOR I%=2 TO 6:P%(1,I%)=20:P%(I%,1)=20:NEXT I%
2460 IF C%(1,8)=N% GOTO 2470 ELSE 2480
2470 FOR I%=2 TO 6:P%(I%,8)=20:P%(1,9-I%)=20:NEXT I%
2480 IF C%(8,1)=N% GOTO 2490 ELSE 2500
2490 FOR I%=2 TO 6:P%(9-I%,1)=20:P%(8,I%)=20:NEXT I%
2500 IF C%(8,8)=N% GOTO 2510 ELSE 2520
2510 FOR I%=3 TO 7:P%(I%,8)=20:P%(8,I%)=20:NEXT I%
2520 IF C%(1,1) <> 6 THEN P%(2,2)=5
2530 IF C%(1,8) <> 6 THEN P%(2,7)=5
2540 IF C%(8,1) <> 6 THEN P%(7,2)=5
2550 IF C%(8,8) <> 6 THEN P%(7,7)=5
2560 P%(1,1)=30:P%(1,8)=30:P%(8,1)=30:P%(8,8)=30
2570 FOR I%=3 TO 6
2580 IF C%(1,I%)=N% THEN P%(2,I%)=4
2590 IF C%(8,I%)=N% THEN P%(7,I%)=4
2600 IF C%(I%,1)=N% THEN P%(I%,2)=4
2610 IF C%(I%,8)=N% THEN P%(I%,7)=4
2620 NEXT I%
2630 IF C%(7,1)=Q% AND C%(4,1)=N% AND C%(6,1)=6 AND C%(
     5,1)=6 THEN P%(6,1)=26
2640 IF C%(1,7)=Q% AND C%(1,4)=N% AND C%(1,6)=6 AND C%(
     1,5)=6 THEN P%(1,6)=26
2650 IF C%(2,1)=Q% AND C%(5,1)=N% AND C%(3,1)=6 AND C%(
     4,1)=6 THEN P%(3,1)=26
2660 IF C%(1,2)=Q% AND C%(1,5)=N% AND C%(1,3)=6 AND C%(
     1,4)=6 THEN P%(1,3)=26
2670 IF C%(8,2)=Q% AND C%(8,5)=N% AND C%(8,3)=6 AND C%(
     8,4)=6 THEN P%(8,3)=26
```

```
2680 IF C%(2,8)=Q% AND C%(5,8)=N% AND C%(3,8)=6 AND C%(
     4,8)=6 THEN P%(3,8)=26
2690 IF C%(8,7)=Q% AND C%(8,4)=N% AND C%(8,5)=6 AND C%(
     8,6)=6 THEN P%(8,6)=26
2700 IF C%(7,8)=Q% AND C%(4,8)=N% AND C%(5,8)=6 AND C%(
     6,8)=6 THEN P%(6,8)=26
2710 RETURN
2720 PAS%=0
2730 FOR I%=1 TO 8
2740 FOR J%=1 TO 8
2750 IF C%(I%,J%)=Q% GOTO 2780
2760 NEXT J%:NEXT I%
2770 PAS%=1:RETURN
2780 FOR K%=1 TO 8
2790 VR%=0:C3%=I%:C4%=J%
2800 C3%=C3%+C1%(K%):C4%=C4%+C2%(K%)
2810 IF C3% < 1 OR C3% > 8 GOTO 2820 ELSE 2830
2820 NEXT K%:GOTO 2760
2830 IF C4% < 1 OR C4% > 8 GOTO 2820 ELSE 2840
2840 IF C%(C3%,C4%)=N% GOTO 2850 ELSE 2860
2850 VR%=VR%+1:GOTO 2800
2860 IF C%(C3%,C4%)=Q% GOTO 2820 ELSE 2870
2870 IF VR% > 0 THEN RETURN
2880 GOTO 2820
2890 DATA 1,0,30,1,20,10,10,20,1,30,1,1,1,1,3
2900 DATA 3,3,3,1,1,0,1,20,3,5,5,5,5,3,20,-1,1,10,3,5
2910 DATA 0,0,5,3,10,-1,0,10,3,5,0,0,5,3,10,-1
2920 DATA -1,20,3,5,5,5,5,3,20,0,-1,1,1,3,3,3,3,1,1,1,-
     1,30,1,20,10,10,20,1,30
2930 DATA 263,100,263,120,270,130,255,130,255,130,255,1
     40,255,140,270,140
2940 DATA 270,140,270,150,270,150,255,150,255,160,270,1
     60,270,160,270,180
2950 DATA 270,180,255,180,270,170,255,170,270,190,270,2
     10,270,200,255,200
2960 DATA 255,200,255,210,255,220,270,220,270,220,270,2
     30,270,230,255,230
2970 DATA 255,230,255,240,255,240,270,240,255,250,270,2
     50,270,250,270,260
2980 DATA 270,260,255,260,255,250,255,270,270,280,270,3
     00,270,300,255,300
2990 DATA 255,310,255,330,255,330,270,330,270,330,270,3
     10,270,310,255,310
3000 DATA 255,320,270,320
```

```
3010 DATA 310,355,310,375,350,355,335,355,335,355,335,3
     65,335,365,350,365
3020 DATA 350,365,350,375,350,375,335,375,365,355,380,3
     55,380,355,380,375
3030 DATA 380,375,365,375,380,365,365,365,410,355,410,3
     75,410,365,395,365
3040 DATA 395,365,395,375,425,355,440,355,440,355,440,3
     65,440,365,425,365
3050 DATA 425,365,425,375,425,375,440,375,455,375,455,3
     55,455,355,470,355
3060 DATA 470,355,470,365,470,365,455,365,485,375,500,3
     75,500,375,500,355
3070 DATA 515,375,515,355,515,355,530,355,530,355,530,3
     75,530,375,515,375
3080 DATA 515,365,530,365
```

Juwelenraub

Brechen Sie in das Schloß seiner Lordschaft ein und machen Sie reiche Beute! Sie müssen jedoch viele Hindernisse überwinden - vor allem hüten Sie sich vor dem Hund! Für einen Spieler gegen den Computer. Mit Tasten oder Joystick.

```
10 'RAFFLES von DAVID RADISIC
20 'copyright (c) AMSOFT 1985
30 '
40 MODE 0:INK 0,0:BORDER 0:INK 1,26:INK 2,15:INK 3,25
50 INK 4,14:INK 5,24,12:INK 6,0:INK 7,0:INK 8,0:PAPER #
   1,7
60 delay=200
70 DIM objx(5,20),objy(5,20),gemx(5,20),gemy(5,20)
80 GOSUB 380
90 GOSUB 720
100 pause=200:GOSUB 340
110 IF gems=0 THEN GOSUB 970
120 PEN 4
130 FOR i=10 TO 12
140 LOCATE 15,i:PRINT"BEUTE";
150 NEXT
160 PAPER 0:CLS#2:PAPER 8
170 GOSUB 1170
180 GOSUB 1230
190 GOSUB 1370
200 GOSUB 1510
210 IF rm=0 THEN GOSUB 1900
220 IF dead=0 THEN 160
230 pause=100:GOSUB 340
240 PAPER 0:CLS:PEN 1
250 LOCATE 4,3:PRINT"Wollen Sie";
260 LOCATE 4,5:PRINT"noch ein Spiel";
270 PEN 5:LOCATE 9,7:PRINT"J/N";
280 i$=UPPER$(INKEY$):IF i$<>"J" AND i$<>"N" THEN 280
290 IF i$="N" THEN MODE 2:PEN 1:STOP
300 RUN
310 IF dog=1 THEN RETURN
320 dog=1:dogx=minx(rm):dogy=miny(rm)
330 RETURN
340 FOR loop=1 TO pause
350 FRAME
360 NEXT
370 RETURN
```

```
380 rm=1:xp=6:yp=4:man$=CHR$(224):dog=0:stolen=0
390 SYMBOL 240,8,8,8,8,8,8,8,8
400 SYMBOL 241,0,0,0,0,255,0,0,0
410 SYMBOL 242,0,0,0,0,15,8,8,8
420 SYMBOL 243,0,0,0,0,248,8,8,8
430 SYMBOL 244,8,8,8,8,248,0,0,0
440 SYMBOL 245,8,8,8,8,15,0,0,0
450 SYMBOL 246,8,12,13,14,12,12,8,8
460 SYMBOL 247,8,12,12,14,13,12,8,8
470 SYMBOL 248,8,24,88,56,24,24,8,8
480 SYMBOL 249,8,24,24,56,88,24,8,8
490 SYMBOL 250,0,0,255,129,129,129,255,0
500 SYMBOL 251,28,20,20,20,20,20,20,28
510 SYMBOL 252,0,0,255,255,255,255,255,0
520 SYMBOL 253,28,28,28,28,28,28,28,28
530 SYMBOL 255,195,165,60,126,90,60,36,24
540 ENT 1,12,-4,1
550 ENT -2,=1000,60,=3000,40
560 ENV 1,10,1,5,2,-4,1,2,-1,20
570 windw$(1)=STRING$(2,250):windw$(2)=CHR$(251)+CHR$(8
    )+CHR$(10)+CHR$(251)+CHR$(8)+CHR$(10)+CHR$(251)
580 door$(1)=STRING$(2,252):door$(2)=CHR$(253)+CHR$(8)+
    CHR$(10)+CHR$(253)+CHR$(8)+CHR$(10)+CHR$(253)
590 switch$(1,0)=CHR$(246):switch$(1,1)=CHR$(247)
600 switch$(2,0)=CHR$(248):switch$(2,1)=CHR$(249)
610 gem$=CHR$(144):obj$=CHR$(233):dog$=CHR$(255)
620 hit$=CHR$(246)+CHR$(248)+CHR$(247)+CHR$(249)+CHR$(2
    52)+CHR$(253)+CHR$(250)+CHR$(251)+gem$+obj$+dog$
630 RESTORE 3010
640 FOR i=1 TO 5
650 READ minx(i),miny(i),maxx(i),maxy(i)
660 READ dir(i,1),dir(i,2),dir(i,3),dir(i,4)
670 NEXT
680 WINDOW #1,minx(rm)-1,maxx(rm)+1,miny(rm)-1,maxy(rm)
    +1
690 WINDOW #2,1,14,1,25
700 CLS #1:PAPER #0,8
710 RETURN
720 ORIGIN 50,50
730 INK 6,24,12
740 RESTORE 3060
750 GOSUB 1280
760 LOCATE 3,20
770 PEN 5:PRINT">";
```

```
780 PEN 1:PRINT"Fluchtwege";
790 PEN 5:PRINT"<";:PEN 1
800 LOCATE 10,2:PRINT"Eingang";
810 pause=300:GOSUB 340
820 CLS:LOCATE 1,3:INK 6,0
830 PEN 1:PRINT man$;"  Sie, der Dieb":PRINT
840 PEN 2:PRINT LEFT$(door$(1),1);LEFT$(door$(2),1);" D
    oors":PRINT
850 PEN 3:PRINT switch$(1,0);switch$(2,0);" Lichtschalt
    er (AUS)"
860 PEN 3:PRINT switch$(1,1);switch$(2,1);" Lichtschalt
    er (AN)":PRINT
870 PEN 4:PRINT LEFT$(windw$(1),1);LEFT$(windw$(2),1);"
     Fenster":PRINT
880 PEN 5:PRINT gem$;"  Wertvolle Juwelen":PRINT
890 PAPER 1:PEN 0:PRINT obj$;"  Hindernisse ":PEN 1:PAP
    ER 0:PRINT
900 PEN 1:PRINT dog$;"  Der Hund"
910 PEN 5:PRINT:PRINT:PRINT
920 PRINT"  Joystick":PRINT"   oder Cursor-Tasten"
930 dummy=REMAIN(1)
940 AFTER delay*4,1 GOSUB 340
950 RETURN
960 '
970 'Erzeugt Juwelen/Hindernisse
980 '
990 FOR room=1 TO 5
1000 gemr=INT(RND*8)+2:objr=INT(RND*10)+5
1010 minx=minx(room):miny=miny(room):maxx=maxx(room):ma
     xy=maxy(room)
1020 FOR i=1 TO gemr
1030 x=INT(RND*(maxx-minx+1))+minx
1040 y=INT(RND*(maxy-miny+1))+miny
1050 gemx(room,i)=x:gemy(room,i)=y
1060 gems=gems+1
1070 NEXT i
1080 FOR i=1 TO objr
1090 x=INT(RND*(maxx-minx+1))+minx
1100 y=INT(RND*(maxy-miny+1))+miny
1110 objx(room,i)=x:objy(room,i)=y
1120 NEXT i
1130 gems(room)=gemr:obj(room)=objr
1140 NEXT room
1150 CLS
```

```
1160 RETURN
1170 ON rm GOTO 1180,1190,1200,1210,1220
1180 RESTORE 2670:RETURN
1190 RESTORE 2740:RETURN
1200 RESTORE 2810:RETURN
1210 RESTORE 2880:RETURN
1220 RESTORE 2960:RETURN
1230 PAPER 0:READ rm$:PAPER 8
1240 WINDOW #1,minx(rm)-1,maxx(rm)+1,miny(rm)-1,maxy(rm
     )+1:CLS #1
1250 PEN 1:LOCATE 1,1:PRINT SPACE$(19);
1260 LOCATE 1,1:PRINT "Zimmer :";rm$;
1270 IF lights(rm) THEN INK 7,10:INK 8,10 ELSE INK 7,0:
     INK 8,0
1280 READ a$:IF a$="END" THEN RETURN
1290 IF a$="D" THEN 2180
1300 IF a$="W" THEN 2260
1310 IF a$="L" THEN GRAPHICS PEN 1:GOTO 2340
1320 IF a$="S" THEN 2420
1330 IF a$="F" THEN GRAPHICS PEN 6:GOTO 2340
1340 PRINT"***FEHLER ***";
1350 STOP
1360 '
1370 'Zeigt Juwelen/Gegenstaende an
1380 '
1390 PEN 6
1400 FOR i=1 TO obj(rm)
1410 LOCATE objx(rm,i),objy(rm,i)
1420 PRINT obj$;
1430 NEXT
1440 PEN 5
1450 FOR i=1 TO gems(rm)
1460 LOCATE gemx(rm,i),gemy(rm,i)
1470 PRINT gem$;
1480 NEXT
1490 PEN 1:LOCATE xp,yp:PRINT man$;
1500 RETURN
1510 xf=0:yf=0:PEN 1
1520 IF INKEY(0)<>-1 OR INKEY(72)<>-1 THEN yf=-1
1530 IF INKEY(2)<>-1 OR INKEY(73)<>-1 THEN yf=1
1540 IF INKEY(8)<>-1 OR INKEY(74)<>-1 THEN xf=-1
1550 IF INKEY(1)<>-1 OR INKEY(75)<>-1 THEN xf=1
1560 IF xf=0 AND yf=0 THEN 1630
1570 LOCATE xp+xf,yp+yf:ht$=COPYCHR$(#0)
```

```
1580 IF ASC(ht$)>239 AND ASC(ht$)<246 THEN 1510
1590 IF ht$<>" " THEN 1660
1600 LOCATE xp,yp:PRINT " ";
1610 PAPER 0:LOCATE 15,5:PRINT"     ";:PAPER 8
1620 xp=xp+xf:yp=yp+yf
1630 LOCATE xp,yp:PRINT man$;
1640 IF dog>0 THEN dog=dog MOD 2+1:IF dog=2 THEN 2550
1650 GOTO 1510
1660 hit=INSTR(hit$,ht$):char=ASC(MID$(hit$,hit,1))
1670 ON hit GOTO 1690,1690,1690,1690,1750,1750,1850,190
     0,1970,2090,2650
1680 GOTO 1600
1690 IF hit>2 AND hit<5 THEN char=char-1
1700 IF hit<3 THEN char=char+1
1710 PEN 3:LOCATE xp+xf,yp+yf:PRINT CHR$(char);
1720 lights(rm)=lights(rm) XOR 1
1730 IF lights(rm) THEN INK 7,10:INK 8,10 ELSE INK 7,0:
     INK 8,0
1740 GOTO 1510
1750 IF xf<>0 AND yf<>0 THEN 1630
1760 IF xf<0 THEN dir=4 ELSE IF xf>0 THEN dir=3
1770 IF yf<0 THEN dir=1 ELSE IF yf>0 THEN dir=2
1780 IF dir(rm,dir)=-1 THEN 1630 ELSE rm=dir(rm,dir)
1790 IF dog>0 THEN GOSUB 310
1800 IF dir=1 THEN xp=6:yp=maxy(rm)
1810 IF dir=2 THEN xp=6:yp=miny(rm)
1820 IF dir=3 THEN xp=minx(rm):yp=13
1830 IF dir=4 THEN xp=maxx(rm):yp=13
1840 RETURN
1850 IF xp>5 AND xp<8 THEN 1880
1860 IF xp<6 THEN dir=4 ELSE dir=3
1870 GOTO 1780
1880 IF yp>13 THEN dir=2 ELSE dir=1
1890 GOTO 1780
1900 PAPER 0:CLS:PEN 1
1910 LOCATE 3,3:PRINT"Sie sind entwischt";
1920 LOCATE 9,5:PRINT"mit";
1930 IF gems=stolen THEN LOCATE 8,7:PRINT"allen"; ELSE
     LOCATE 9,7
1940 PRINT USING " ##";stolen;
1950 PEN 5:LOCATE 9,9:PRINT"Juwelen";
1960 dead=1:RETURN
1970 LOCATE xp,yp:PRINT" ";:xp=xp+xf:yp=yp+yf
```

```
1980 i=0
1990 i=i+1
2000 IF i>gems(rm) THEN 1510
2010 IF gemx(rm,i)<>xp OR gemy(rm,i)<>yp THEN 1990
2020 IF i=gems(rm) THEN 2050
2030 gemx(rm,i)=gemx(rm,gems(rm))
2040 gemy(rm,i)=gemy(rm,gems(rm))
2050 gems(rm)=gems(rm)-1:stolen=stolen+1
2060 MOVE 400,150+(stolen*2),1,1:DRAW 520,150+(stolen*2
     ),1,1
2070 SOUND 129,248,10,12,0,1
2080 GOTO 1980
2090 noise=INT(RND*15)
2100 SOUND 1,3000,10,noise,0,0,10
2110 PAPER 0:LOCATE 15,5:PRINT"Krach!";:PAPER 8
2120 IF noise<10 OR delay=50 THEN 1630
2130 delay=delay-50
2140 dummy=REMAIN(1)
2150 AFTER delay*4,1 GOSUB 310
2160 GOTO 1630
2170 '
2180 'Zeichnet Tueren
2190 '
2200 READ no,dr$
2210 IF dr$="V" THEN dr=2 ELSE dr=1
2220 PEN 2
2230 pic$=door$(dr):GOSUB 2500
2240 GOTO 1280
2250 '
2260 'Zeichnet Fenster
2270 '
2280 READ no,wi$
2290 IF wi$="V" THEN wi=2 ELSE wi=1
2300 PEN 4
2310 pic$=windw$(wi):GOSUB 2500
2320 GOTO 1280
2330 '
2340 'Zeichnet Linien
2350 '
2360 READ x1,y1,x2,y2
2370 MOVE x1,y1,,0
2380 DRAW x1,y2,,0:DRAW x2,y2,,0
2390 DRAW x2,y1,,0:DRAW x1,y1,,0
2400 GOTO 1280
```

```
2410 '
2420 'Zeichnet Schalter
2430 '
2440 READ no,sw$
2450 IF sw$="L" THEN sw=1 ELSE sw=2
2460 PEN 3
2470 pic$=switch$(sw,0):GOSUB 2500
2480 GOTO 1280
2490 '
2500 'Gibt ein Zeichen aus
2510 '
2520 READ x,y:LOCATE x,y:PRINT pic$;
2530 no=no-1:IF no>0 THEN 2520
2540 RETURN
2550 PEN 1:LOCATE dogx,dogy:PRINT" ";
2560 man$=CHR$(225)
2570 IF (dogx=xp AND dogy=yp) OR (dogx=xp+xf AND dogy=y
     p+yf) THEN 2650
2580 IF dogx<xp THEN dogx=dogx+1
2590 IF dogx>xp THEN dogx=dogx-1
2600 IF dogy<yp THEN dogy=dogy+1
2610 IF dogy>yp THEN dogy=dogy-1
2620 LOCATE dogx,dogy:PRINT dog$;
2630 SOUND 1,0,RND*40,10,1,2,31
2640 GOTO 1510
2650 PRINT"Schnapp";
2660 dead=1:RETURN
2670 DATA Flur
2680 DATA L,64,308,226,4
2690 DATA D,2,H,6,3,6,22
2700 DATA D,2,V,4,12,9,11
2710 DATA S,1,L,4,11
2720 DATA S,1,R,9,14
2730 DATA END
2740 DATA Wohnzimmer
2750 DATA L,2,308,258,4
2760 DATA D,1,V,10,12
2770 DATA W,1,H,6,3
2780 DATA W,1,V,2,12
2790 DATA S,2,R,10,11,10,15
2800 DATA END
2810 DATA Esszimmer
2820 DATA L,2,308,258,4
2830 DATA W,1,V,10,12
```

```
2840 DATA W,1,H,6,3
2850 DATA D,1,V,2,12
2860 DATA S,2,L,2,11,2,15
2870 DATA END
2880 DATA Kueche
2890 DATA L,2,276,384,4
2900 DATA D,2,H,6,5,6,22
2910 DATA W,1,H,10,22
2920 DATA W,1,V,14,13
2930 DATA D,1,V,2,13
2940 DATA S,1,L,2,16
2950 DATA END
2960 DATA Speisekammer
2970 DATA L,2,276,256,4
2980 DATA D,1,V,10,12
2990 DATA S,1,R,10,11
3000 DATA END
3010 DATA 5,4,8,21,0,4,3,2
3020 DATA 3,4,9,21,-1,-1,1,-1
3030 DATA 3,4,9,21,-1,-1,-1,1
3040 DATA 3,6,13,21,1,0,-1,5
3050 DATA 3,6,9,21,-1,-1,4,-1
3060 DATA L,64,308,480,100
3070 DATA F,250,98,294,102
3080 DATA F,250,306,294,310
3090 DATA F,390,94,430,106
3100 DATA F,390,302,430,314
3110 DATA F,474,240,488,270
3120 DATA F,474,124,488,154
3130 DATA F,58,240,72,270
3140 DATA L,226,308,322,180
3150 DATA L,160,180,480,100
3160 DATA L,64,180,160,100
3170 DATA END
```

Anhang 4
Index

Alle Angaben dieses Anhangs beziehen sich auf das entsprechende Kapitel mit Seitenzahl. 1.44 beispielweise bedeutet Kapitel 1 Seite 44.

A

| A 1.48 1.81 5.8
ABS 3.4
AFTER 3.4 9.30
AMSDOS 1.81 5.1 5.33
AMSDOS-Befehle 5.8
AMSDOS Fehlermeldungen 5.15 7.31
AND 3.5 9.19
AND (LOGO) 6.16
Animation 9.55
Anschließen der Peripheriegeräte 1.7 7.39 7.40 7.41 7.42
Anschließen des Computers 1.2
Antennenbuchse 1.3
.APV 6.43
ARCTAN 6.13
Arithmetische Operationen 1.36 7.28 7.29
Arithmetische Operationen (LOGO) 6.13
Arrays 2.3 3.19 3.26 7.29
ASC 3.5
ASCII 7.9 7.22
ASCII-Zeichen 7.9 7.10
ASCII-Datei 1.49 3.73 5.4 5.12 5.13 5.14 7.30
ASCII (LOGO) 6.8
ASM 5.38
ASSIGN.SYS 4.10
ATN 3.6
Auswurfknopf 1.14
AUTO 3.6 2.11
AUX 5.29

B

| B 1.48 1.81 5.9
| BANKFIND 1.94 8.6
BANK MANAGER 1.90 8.1 9.65

| BANKOPEN 1.93 8.5
| BANKREAD 1.93 8.5
| BANKWRITE 1.93 8.5
BASIC 1.24 3.1 7.33 9.7
BASIC-Dateien 1.49 3.73
Befehle 1.24 3.1
Benutzerdefinierte Zeichen 3.81 7.47 9.21
BF 6.8
Bildfangregelung 1.4
Bildschirmabdruck 1.50 3.73 5.6
Bildschirmanpassung 4.6 7.49
Bildschirmpläne 7.35 7.36 7.37
BIN$ 3.7
Binärdateien 1.50 3.73 5.13 5.14
Binärsystem 9.9
Bit 9.9
BK 6.26
BL 6.8
Blinkende Farben 1.56 3.77
BOOTGEN 5.34
BORDER 1.53 3.7 7.7
BREAK 3.7
BUTTONP 6.36
BYE 6.7 6.38
Byte 9.10

C

CALL 3.8
CAPS LOCK-Taste 1.17
CAT 1.47 3.8
CAT (Kassette) 4.12
CATCH 6.40
CHAIN 3.8
CHAIN MERGE 3.9
CHAIN MERGE (Kassette) 4.14
CHANGEF 6.33
CHAR 6.8
CHR$ 1.59 3.9 9.16
CINT 3.10
CLEAN 6.22
CLEAR 3.10
CLEAR INPUT 3.10

CLG 3.11
CLOAD 5.13 5.14 5.36
CLOSEIN 2.11 3.11
CLOSEIN (Kassette) 4.15
CLOSEOUT 2.10 3.11
CLOSEOUT (Kassette) 4.16
CLEAR-Taste 1.18
CLS 1.24 3.11 7.4
CO 6.39
CON 5.29
CONT 3.12 7.30
CONTENTS 6.42
COPY-Cursor-Editiermethode 1.30
COPYCHR$ 3.12
COPYON 6.20
COPYOFF 6.20
COPY-Taste 1.31
COS 3.12
COS (LOGO) 6.13
COUNT 6.9
CP/M 5.17
CP/M 2.2 ..... 1.88 4.11 5.32 5.33 6.2
|CPM 1.42 5.9
CREAL 3.13
CRT 5.28
CS 6.22
CSAVE 5.13 5.14 5.36
CT 6.20
CURSOR 3.13 7.3
CURSOR (LOGO) 6.20
CURSOR-Tasten 1.15 6.6

D

DATA 3.14 7.28 9.32
DATA-Format 1.44 5.33 7.45
DATE 5.32
Datierung 5.32
DC-Buchsen 1.2 1.3
DDT 5.38
DEC$ 3.14
.DEF 6.43

DEFAULTD 6.33
DEF FN 3.15 7.30
DEFINE 3.15
DEFINT 3.15
DEFREAL 3.16
DEFSTR 3.16
DEG 3.17
DELETE 3.17
DEL-Taste 1.16
.DEPOSIT 6.42
DERR 3.18 7.31 7.32 7.33
DEVICE 5.28
DI 3.18 9.32
DIM 2.3 3.19 7.28
I DIR 5.9
DIR (CP/M) 1.43 5.21 5.30
DIR (LOGO) 6.33
DIRPIC 6.34
DIRS 5.22
DIRSYS 5.22
I DISC 1.82 5.9
DISC DRIVE 2-Buchse 1.9 7.41
I DISC.IN 1.81 5.9
DISCKIT2 ..... 1.88 5.36
DISCKIT3 ..... 1.43 1.83 5.24
I DISC.OUT 1.81 5.10
Disketten 1.11 4.1
Disketten einlegen 1.11
Diskettenlaufwerk (zusätzliches) 1.8 1.14 1.43 1.46 1.86 5.4 7.41
Diskettenorganisation 7.44
DOT 6.22
DOTC 6.23
DRAW 1.63 3.20
DRAWR 3.20
I DRIVE 5.10
DRIVERS.GSX 4.10
Dr. LOGO 6.2
Drucker 1.8 5.26 7.42
Druckformatierung 3.63 9.22
DUMP 5.38

E

ED 5.38
ED (LOGO) 6.19
EDALL 6.19
EDF 6.20
EDIT 1.30 3.21
Editieren 1.29
Editieren (LOGO) 6.6 6.19
EI 3.21 9.32
Einschalten des Computers 1.4 1.5
ELSE 1.31 3.21
EMPTYP 6.9
.EMT 6.43
END 3.22
END (LOGO) 6.18
.ENL 6.43
ENT 1.77 3.22 9.42
ENT (LOGO) 6.38
ENTER-Taste 1.16
ENV 1.75 3.24 9.39
ENV (LOGO) 6.38
EOF 3.26 4.15 5.8 7.30
EOF (CP/M) 5.29
ER 6.30
ERA 5.22
| ERA 5.10
ERALL 6.30
ERASE 3.26
ERASE (CP/M) 5.22 5.30
ERL 3.27
ERN 6.30
ERR 3.27 7.33
ERRACT 6.42
ERROR 3.27
ERROR (LOGO) 6.40
Erweiterungszeichen 3.39 7.23
ESC-Taste 1.18 3.51 3.52
EVERY 3.28 9.30
.EXAMINE 6.42
EXP 3.29
Externe Befehle 1.81 1.91 5.1 7.31 8.7

F

FALSE 6.42
Farben 1.52 5.25
Farbstift-Modi 7.5 9.54
FD 6.26
Fehlermeldungen 7.28
Fehlermeldungen (AMSDOS) 5.15 7.31
Fehlernummern 7.28
FENCE 6.23
Fernsehgerät 1.3 1.5 1.73
FILECOPY 4.11 5.35
FILL 1.67 3.29 9.50
Firmware 7.47 8.1
FIRST 6.9
FIX 3.29
FN 3.30
FOR 1.33 3.30 7.28 7.31 9.16
Formatieren von Disketten 1.41 1.85 5.33 7.44
FPUT 6.9
FRAME 1.61 3.31
FRE 3.31
FS 6.24

G

GENGRAF 4.10
Geschützte Dateien 1.49 3.73
GLIST 6.32
GO 6.39
GOSUB 1.34 3.32 7.28
GOTO 1.26 3.32
GRAPHICS PAPER 3.32 9.51
GRAPHICS PEN 1.69 3.33 9.50
Graphik 1.51 1.59 9.49
GPROP 6.32
GSX 4.9

H

Hardware 9.1
Helligkeitsregler 1.4
HELP-Programm 4.3
HEX$ 3.33
Hexadezimalsystem 9.11
HIMEM 3.34 7.47
HOME 6.26
HT 6.27
Hüllkurvenplanung 7.38

I

IBM-Format 5.33 7.45
IF 1.31 3.34
IF (LOGO) ..... 6.39 .IN 6.42
Inbetriebnahme des Computers 1.1
INK 1.54 3.35 7.6
INKEY 3.35 7.43
INKEY$ 2.13 3.36 7.43
INP 3.37
INPUT 1.27 2.2 3.37 7.29
INPUT (Kassette) 4.15
INSTR 2.5 3.38
INT 3.38
INT (LOGO) 6.13
I/O (E/A) 7.39 7.48
ITEM 6.10

J

JOY 3.39 7.43
Joystick 1.7 7.22 7.24 7.43
Joystick-Befehle (LOGO) 6.36
Joystick-Buchse 1.7 7.39

K

Kassettenbetrieb 1.7 1.82 4.12 5.12 5.13 5.14 5.36
Kennwort 5.31
KEY 3.39 7.23
KEYDEF 3.40 7.23 7.43
KEYP 6.36
KEYS.CCP 4.3 5.26
KEYS.DRL 5.26
KEYS.WP 5.26
Knoten 6.7
Konfigurationssektor 5.36 5.37
Konfigurieren von Programmen/Software-Paketen 4.7 4.8
Kontrast-Einstellung 1.4
Kontrollampe (Diskettenlaufwerk) 1.14
Kontrollsumme 9.33
Kopfhörer 1.9 1.73
Kopieren von Dateien 1.49 1.83 5.12 5.13 5.14
Kopieren von Disketten 1.83
Kreise 1.6K
Kubikwurzel 1.38

L

LABEL 6.39
Laden (Programme/Software) 1.21 1.48
LANGUAGE 4.3 5.24 7.55
LAST 6.10
Lautsprecher 1.9 1.73
Lautstärken-Hüllkurve 1.75 3.24 9.39
Lautstärke-Regulierung 1.9 1.73
LC 6.10
LEFT$ 3.41
LEN 2.9 3.41
Lesefehler 4.13
LET 3.41
LINE INPUT 3.42
LINE INPUT (Kassette) 4.15
LIST 1.26 1.58 3.42
LIST (LOGO) 6.10
LISTP 6.10
LOAD 3.43

LOAD (Kassette) 4.14
LOAD (CP/M) 5.38
LOAD (LOGO) 6.34
LOADPIC 6.34
LOCAL 6.17
LOCATE 1.60 3.43 7.7
LOG 3.44
LOG10 ..... 3.44
Logik 3.5 3.50 3.56 3.94 9.18
Logische Operationen (LOGO) 6.16
LOGO 6.1
LPT 5.28
LPUT 6.11
LST 5.28
LT 6.27
LOWER$ 3.44

M

Maschinencode 7.7
MAKE 6.17
MASK 3.45 9.50
MAX 3.45
MEMBERP 6.11
MEMORY 3.46 7.28 7.47
Menü 2.6
MERGE 3.46
MERGE (Kassette) 4.14
MID$ 3.46 3.47
MIN 3.47
MOD 1.37 3.48
MODE 1.51 3.48 7.3
Modulator/Netzteil (MP2) 1.3-1.5 1.52 1.73
Monitor 1.2
MONITOR-Buchse 1.2 1.3 7.40
MOVCPM 5.38
MOVE 1.64 3.48
MOVER 3.49

N

Namenlose Dateien 4.13 4.16 4.18
NAMEP 6.17
Netzstecker 1.1
NEW 3.50
NEXT 1.33 1.66 3.50 7.28 7.31 9.17
NODES 6.30
NOFORMAT 6.3
NOT 3.50 9.20
NOT (LOGO) 6.16
Noten 7.25
NOTRACE 6.41
NOWATCH 6.41
NUMBERP 6.41

O

ON BREAK CONT 3.51
ON BREAK GOSUB 3.51
ON BREAK STOP 3.52
ON ERROR GOTO 3.52 7.30 7.33
ON GOSUB 2.7 3.53
ON GOTO 3.54
ON SQ GOSUB 3.54 7.8 9.46
OP 6.39
OPENIN 2.11 3.55 7.30 7.31
OPENIN (Kassette) 4.15
OPENOUT 2.10 3.55 7.31
OPENOUT (Kassette) 4.16
Operatoren 1.40 9.18
OR 3.56 9.19
OR (LOGO) 6.16
ORIGIN 1.66 3.56
OUT 3.57
.OUT (LOGO) 6.42

P

PADDLE 6.36
PAL 6.24
PALETTE 5.25
PAPER 1.53 3.57 7.4
PAUSE 6.41
PD 6.27
PE 6.27
PEEK 3.58
PEN 1.53 3.58 7.4
Peripheriegeräte 1.7 9.3
PI 3.59
PIECE 6.11
PIP 4.4 5.29
PLIST 6.32
PLOT 1.62 3.59
PLOTR 3.60
Plotten 9.50
PO 6.18
POALL 6.31
POKE 3.60
PONS 6.31
POPS 6.31
POS 3.61 4.15
Potenzen 1.38
POTS 6.19
POWER-Schalter 1.4 1.5
PPROP 6.32
PPS 6.32
PR 6.21
PRINT 1.25 3.61 9.22
PRINT SPC 3.62 9.24
PRINT TAB 3.62 9.24
PRINT USING 3.63 9.24
PRINTER-Buchse 1.8
.PRM 6.43
PRN 5.29
PROFILE-Programm 4.3 5.18
Prüfung von Disketten 1.88
PU 6.27
PX 6.28

Q

Quadratwurzel 1.38 3.79
QUOTIENT 6.13

R

RAD 3.66
RAM 5.17 8.4
RAM-Diskette 1.92 8.4
RANDOM 6.14
RANDOMIZE 3.66
RC 6.37
READ 3.66 7.28 7.29 9.31
Read/Only-Dateien 5.12 5.31 5.35 7.32
RECYCLE 6.7 6.31
REDEFP 6.42
RELEASE 3.67 9.44
RELEASE (LOGO) 6.38
REM 1.33 2.2 3.67
.REM 6.43
REMAIN 3.68 9.32
REMAINDER 6.14
REMPROP 6.33
REN 5.22
| REN 5.10
RENAME 5.22 5.30
Rendezvous 3.75 9.37
RENUM 2.8 3.68
REPEAT 6.39
RERANDOM 6.14
Reservekopien 4.2
RESTORE 3.69 9.35
RESUME 3.69 7.30
RESUME NEXT 3.70
RETURN 1.34 3.70 7.28
RETURN-Taste 1.15 1.19
RIGHT$ 3.71
RL 6.37
RND 3.71
RQ 6.37
ROINTIME.DEM-Programm 1.21

ROUND 3.72
ROUND (LOGO) 6.14
RS232 ..... 5.28 9.3
RSX 7.46
RT 6.28
RUN 1.25 1.48 3.72
RUN (LOGO) 6.40
RUN (Kassette) 4.14

S

SAVE 1.47 1.49 3.73
SAVE (Kassette) 4.18
SAVE (LOGO) 6.34
SAVEPIC 6.35
Schlüsselfertige Disketten 4.5 4.6
Schlüsselwörter 1.24 7.33
Schreibschutz 1.12 1.86
| SCREENCOPY 1.90 8.3 9.65
| SCREENSWAP 1.90 8.3 9.65
SE 6.12
Serielle Schnittstelle 5.28 9.3
SET 5.30
SET24X80 ..... 5.27
SETBG 6.24
SETCURSOR 6.21
SETD 6.35
SETDEF 5.32
SETH 6.28
SETLST 5.26
SETKEYS 4.3 5.26
SETPAL 6.24
SETPC 6.28
SETPOS 6.28
SETSCRUNCH 6.25
SETSIO 5.28
SETSPLIT 6.21
SETUP 5.36
SETX 6.29
SETY 6.29
SF 6.25
SGN 3.74

SHIFT-Tasten 1.16
SHOW (CP/M) 5.33
SHOW (LOGO) 6.21
SHUFFLE 6.12
SIN 3.74
SIN (LOGO) 6.15
SIO 5.28
Software 1.21 1.22 7.55 7.56
SOUND 1.73 3.73 9.35
SOUND (LOGO) 6.37
Sound-Befehle (LOGO) 6.37
SPACE$ 3.77
SPC 3.77 9.24
SPEED INK 3.77
SPEED KEY 3.78
SPEED WRITE 3.78 4.19
Speicher 1.90 7.47 8.1 8.2 9.58
Speichern auf Diskette 1.41
Speichern auf Kassette 4.16
Speichern von Variablen 2.10 3.93 5.6
Speicherplatz (Diskette) 1.42 1.48
Sprachsynthesizer 9.3
SQ 3.79 9.46
SQR 1.38 3.79
SS 6.25
ST 6.29
Starten von Programmen 1.25 1.48 3.72
STAT 5.34
STEP 1.33 3.78
Stereo 1.9 1.73
STEREO-Buchse 1.9 1.73 7.40
Steuer-Codes 5.19 7.1 7.3 7.50 9.52
STOP 3.80
STOP (LOGO) 6.40
STR$ 3.80
STRING$ 3.81
Stringvariablen 1.28
SUBMIT 4.9 5.33
SWAP 3.81
SYMBOL 3.81 7.6 9.21
SYMBOL AFTER 3.83 7.47
Syntax error 1.19 7.28
SYSGEN 5.34
System-Format 1.44 5.34 7.45

T

TAB 3.84 9.24
TAG 3.84 9.52
TAGOFF 3.85 9.52
TAN 3.85
| TAPE 1.82 5.11
| TAPE.IN 1.82 5.11
| TAPE.OUT 183 5.11
TAPE-Buchse 1.7 7.40
Tastatur 1.15 5.26 7.22 7.23 7.24 7.43
TEST 3.86
TESTR 3.86
TEXT 6.19
TF 6.29
THEN 1.31 3.86
THING 6.18
THROW 6.41
TIME 3.87
TO 3.87
TO (LOGO) 6.19
Tonerzeugung 1.73 3.75 9.36
Ton-Hüllkurve 1.77 3.22 9.41
Tonperioden 7.25
TOPLEVEL 6.43
TOWARDS 6.30
TRACE 6.41
Transparent-Modus 7.5 9.53
TROFF 3.88
TRON 3.88
TRUE 6.43
TS 6.21
TYP 5.23
TYPE (CP/M) 5.23 5.30
TYPE (LOGO) 6.22

U

UC 6.12
UNT 3.88
Unterbrechungen 7.8 9.30
Universalzeichen 5.5

UPPER$ 3.89
USE 5.23
USER 5.23
| USER 5.11
USING 3.89 9.24

V

VAL 3.89
Variablen 1.28 7.33 9.16
Vendor-Format 1.44 5.33 7.45
Vibrato 9.42
Vorführprogramm 1.21
VPOS 3.90

W

WAIT 3.90
WAIT (LOGO) 6.40
WATCH 6.41
WEND 3.91 7.31
WHERE 6.12
WHILE 3.91 7.28 7.31
WIDTH 3.91
WINDOW 2.12 3.92 7.6 9.27
WINDOW (LOGO) 6.26
WINDOW SWAP 3.92 9.28
WORD 6.12
WORDP 6.13
WRAP 6.26
WRITE 2.10 3.93 9.24
WRITE (Kassette) 4.16

X

XOR 3.94 9.21
XPOS 3.94
XSUB 5.38

Y

YPOS 3.95

Z

Zeichen 1.59 7.9 7.54 7.56 9.14
Zufallszahlen 3.71
Zurücksetzen des Computers 1.21
Zusätzliche ROMs 7.49
ZONE 3.95 9.23



Document numérisé avec amour par

AMSTRAD CPC

MÉMOIRE ÉCRITE

https://acpc.me/