Colour Personal Computer

»CPC664«

Benutzerhandbuch



Schneider
COMPUTER DIVISION

Nachtrag zum Handbuch »CPC 664«

CP/M-Systemdiskette

Bitte beachten Sie, daß sich die CP/M-Systemdiskette außen in der weißen Styroporverpackung befindet.

Handbuch »CPC 664«

Beim Druck des 480 Seiten umfassenden Handbuchs haben sich leider einige Fehler eingeschlichen. Damit Sie die Vorteile des »CPC 664» nutzen können, erhalten Sie nachstehend die Korrekturen, die Sie bei der Anwendung bitte berücksichtigen.

Grundlagen Teil 1 Seiten 4 und 5

```
 Schneider 64K Microcomputer  (v2)

 ©1984 Amstrad Consumer Electronics plc
 and Locomotive Software Ltd.

 BASIC 1.1

Ready
```

```
 Schneider 64K Microcomputer  (v2)

 ©1984 Amstrad Consumer Electronics plc
 and Locomotive Software Ltd.

 BASIC 1.1

Ready
```

Zusatz zu Cassettenrecorderanschluß am CPC 664

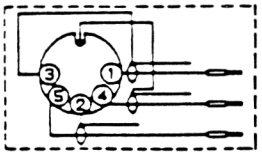
Beim Anschluß eines Cassettenrecorder an den CPC 664 ist darauf zu achten, daß nur ein Cassettenrecorder verwendet werden kann, der Anschlußmöglichkeit für einen Kopfhörer und ein Mikrofon besitzt. Sehr sinnvoll ist auch die Anschlußmöglichkeit einer Fernsteuerung. Meist wird ein solcher Anschluß zur Start-Stop Steuerung des Cassettenrecordermotors in Verbindung mit dem angeschlossenen Mikrofon benutzt. Diese Anschlüsse sind über ein Spezialkabel wie unten beschrieben mit dem CPC 664 zu verbinden.

Anschlußbelegung Cassettenanschlußbuchse CPC 664

Pin 1 u. 3 - Steuerung des Cassettenrecordermotor
Pin 2 - Signalmasse
Pin 4 - Anschluß an Kopfhörerbuchse(Recorder) Daten in Rechner einlesen
Pin 5 - Anschluß an Mikrofonbuchse(Recorder) Daten auf Cassette schreiben

Das Anschlußkabel könnte z.B. so aussehen:

Zum Cassettenrecorder



2.5 mm Klinkenstecker für Motorsteuerung
3.5 mm Klinkenstecker an Kopfhöreranschluß
3.5 mm Klinkenstecker an Mikrofonanschluß

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



```
? 3 / 3  [ENTER]
  1
```


Sie errechnen die Quadratwurzel einer Zahl, indem Sie s q r () eintippen. Die Zahl, aus der Sie die Quadratwurzel errechnen möchten, muß in Klammern stehen.

POTENZEN

(benutzen Sie die ↑ Taste ohne Shift zum Potenzieren)

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

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


Die Diskette besteht aus 40 Spuren von Spur 0 (außen) bis Spur 39 (innen) mit jeweils 9 Sektoren.

Jeder Sektor einer Spur kann bis zu 512 Bytes Daten speichern. Der gesamte verfügbare Speicherplatz auf einer Disketten-Seite beträgt demnach 180 KBytes.

(Das Symbol | ergibt sich durch Halten von [SHIFT] und Drücken der Taste @ .)


```
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
```

Grundlagen Teil 9 Seite 82

```
10 ent 1,100,2,2
20 sound 1,248,200, 15,,1
run
```

Grundlagen Teil 10 Seite 85

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

Kapitel 2 Seite 10

```
135 IF LEN (NAME$(x))> 0 THEN 180
```

Kapitel 3 Seite 2

SONDERZEICHEN

& or &H	kennzeichnet die nachfolgende Konstante als sedezimal
&X	kennzeichnet die nachfolgende Konstante als binär (dual)
#	kennzeichnet die nachfolgende Konstante als Ein-/Ausgabeeinheit (Leitfunktion stream)

Kapitel 3 Seite 4

```
10 AFTER 250 GOSUB 70:CLS
20 PRINT "Rate einen Buchstaben in 5 Sekunden"
30 a$=INKEY$:IF flag=1 THEN END
40 IF a$ <> CHR$(INT(RND*26+97)) THEN 30
50 PRINT a$; "Richtig !! Du hast Gewonnen !!"
60 SOUND 1,478 : SOUND 1,358:END
70 PRINT "Du hast zu lange gebraucht. Ich habe gewonnen !!"
80 SOUND 1,2000:flag = 1:RETURN
run
```

```
IF "Hans" < "Werner" OR "Hund" > "Maus"
THEN PRINT "richtig" ELSE PRINT "falsch"
"richtig"
IF "Werner" < "Hans" OR "Maus" > "Hund"
THEN PRINT "richtig" ELSE PRINT "falsch"
"richtig"
IF "Hans" < "Werner" OR"Maus" > "Hund"
THEN PRINT "richtig" ELSE PRINT "falsch"
"richtig"
```

Kapitel 3 Seite 12

```
10  a=PI
20  PRINT  CINT(a)
30  PRINT  CREAL(a)
run
3
3.14159265
```

Kapitel 3 Seite 44

```
10 MODE 1:ZONE 3
20 a$="ENCYCLOPAEDIA"
30 PRINT "Zeige mir wie man buchstabiert."
40 PRINT"OK ....";a$
50 FOR n=1 TO LEN(A$)
60 PRINT MID$(a$,n,1)
70 FOR t = 1 TO 700:NEXT t,n
80 PRINT:PRINT
90 INPUT "Gib mir ein anderes Wort ein.";a$
100 GOTO 50
run
```

Kapitel 3 Seite 51

```
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
```

Kapitel 3 Seite 53

```
IF "Hans" < "Werner" OR "Hund" > "Maus"
THEN PRINT "richtig" ELSE PRINT "falsch"
"richtig"
IF "Werner" < "Hans" OR "Maus" > "Hund"
THEN PRINT "richtig" ELSE PRINT "falsch"
"richtig"
IF "Hans" < "Werner" OR "Maus" > "Hund"
THEN PRINT "richtig" ELSE PRINT "falsch"
"richtig"
```

Kapitel 3 Seite 63

```
10 CLS: a$="abcdefghijklmnopqrst"
20 PRINT "Eingabezeichen = "; a$
30 PRINT: PRINT "!Spezifikation = ";a$;
40 PRINT USING "!";a$
50 PRINT: PRINT "\Leerstellen\Spezifikation = ";
60 PRINT USING "\          \";a$
70 PRINT: PRINT "& Spezifikation = ";
80 PRINT USING "&";a$
90 GOTO 90
```

Kapitel 3 Seite 69

```
10 GOSUB 50:PRINT"Nach dem GOSUB Befehl":END
50 FOR n = 1 TO 20
60 PRINT"Unterprogramm"
70 NEXT:PRINT
80 RETURN
run
```

```
PRINT STRING$ (4Ø, "*")
```



```
10 MODE 1: REM Basic Uhrzeitprogramm
20 INPUT "Geben Sie die aktuelle Uhrzeit ein     (Stunde,
Minute,Sekunden)"; stunde,minute,sekunde
30 CLS: datum=INT(TIME/300)
40 WHILE stunde < 24
50 WHILE minute < 60
60 WHILE ticken < 60
70 ticken = (INT(TIME/300)-datum)+sekunde
80 LOCATE 70,4
90 PRINT #0,USING "##";stunde,minute,ticken
100 WEND
110 ticken = 0
120 sekunde = 0
130 minute = minute + 1
140 GOTO 30
150 WEND
160 minute = 0
170 stunde = stunde + 1
180 WEND
190 stunde = 1
200 GOTO 40
```


RUN "" (Wenn Sie Cassetten-Software laden, geben Sie zuerst |TAPE ein. Danach kann der RUN-Befehl ausgelöst werden, indem Sie gleichzeitig die Taste [CTRL] und die kleine [ENTER]-Taste des rechten Tastenfeldes drücken.)

```
10 CLS
20 INPUT "Geben Sie den laufenden Monat ein";monat
30 IF monat = 12 OR monat = 1 OR monat = 2 THEN 50
40 GOTO 10
50 PRINT "Jetzt haben wir Winter"
```

```
10 CLS
20 INPUT "Geben Sie den laufenden Monat ein";monat
30 IF NOT (monat = 6 OR monat = 7 OR monat = 8) THEN 50
40 GOTO 10
50 PRINT "Es kann nicht Sommer sein"
```

```
10 CLS
20 INPUT "Geben Sie den heutigen Tag ein";tage
30 INPUT "Geben Sie den laufenden Monat ein";monat
40 IF NOT (monat = 12 OR monat = 1) AND tage = 29 THEN 60
50 CLS: GOTO 20
60 PRINT "Es ist weder Dezember noch Januar,";
70 PRINT "aber vielleicht ist es ein Schaltjahr !!"
```

Die angegebenen Korrekturen können Sie im Handbuch handschriftlich vornehmen oder ausschneiden und an entsprechender Stelle überkleben.

Viel Freude und Erfolg mit Ihrem»**CPC 664**«
wünscht Ihnen Ihre

Schneider
COMPUTER DIVISION

CPC664 COLOUR PERSONAL COMPUTER 64K

Schneider CPC664
Computer mit integriertem Disketten-System

Über uns ...

Der Schneider CPC664 folgt der Tradition des erfolgreichen Computer-Systems CPC464. Die Zustimmung und Anerkennung, die uns für diesen Computer entgegengebracht wurde, hat uns dazu ermutigt, den nächsten logischen Schritt zu tun: wir haben die besten Elemente des CPC464 und des Diskettenlaufwerks DDI-1 verbunden und noch einige neue Eigenschaften und Weiterentwicklungen hinzugefügt. Das Ergebnis - der CPC664 - bietete eine Leistungsfähigkeit und eine solche Vielfalt an Möglichkeiten, wie es heute in der Welt der Personal Computer seinesgleichen sucht.

Fertige Software und Kompatibilität

Auf dem CPC664 läuft normalerweise alles an Software, was für den CPC464 geschrieben wurde. Der Benutzer hat die Möglichkeit, sowohl die für ihn geeigneten Programme aus der Schneider Software-Bibliothek, als auch Produkte unabhängiger Softwareanbieter einzusetzen.

Weshalb Diskette?

Für alle diejenigen, die ihren Computer nicht nur gelegentlich benutzen, ist die Diskette zweifellos die richtige Alternative zur Cassette als Medium zum Speichern von Daten und Programmen. Und weil schon die Stärke und Bedienerfreundlichkeit des CPC464 die Benutzer sehr schnell zu begeisterten Schneider-Anhängern gemacht hat, ist der CPC664 sicher für viele genau der richtige Computer.

Unser Dank an Digital Research ...

Die Stärke des CP/M Betriebssystems von Digital Research war niemals vorher zu einem so niedrigen Preis verfügbar. Damit werden völlig neue Anwenderkreise angesprochen, und wir freuen uns auf die neue Software, die jetzt auf diesem System geschrieben werden kann. Dr. LOGO von Digital Research ist das universellste Lehr-Programm und vereint die Anwender-Freundlichkeit von 'Graphik mit der Schildkröte' (Turtle Graphics) mit hoher Verarbeitungsleistung. Dr. LOGO ist die derzeit umfassendste Implementierung von LOGO. CP/M und Dr. LOGO sind im Lieferumfang des CPC664 enthalten.

Was ist neu?

Der CPC664 bietet außerdem erweiterte Möglichkeiten und BASIC-Befehle, die auf dem CPC464 nicht standardmäßig zur Verfügung standen (Ausfüllen von Flächen, Zeichnen geplanter Linien, Kopieren von Zeichen auf dem Bildschirm, ein- und ausschaltbarer Cursor). Programme, bei denen diese Möglichkeiten verwendet werden, sind nicht nach unten zum CPC464 kompatibel. Jedoch die Übernahme der Software von CPC464 aufwärts zum CPC664 wird dadurch in keinem Fall berührt.

Basiswissen

Wir wollen Ihnen mit diesem Benutzerhandbuch mehr als nur eine Auspack- und Aufbauanleitung geben. Wenn Sie den Inhalt durcharbeiten, lernen Sie bereits vieles über unseren neuen Computer, seine Arbeitsweise, Programmiersprachen, Einsatzmöglichkeiten u.v.a.m.

Wiederholungen mancher Grundsätze dienen dazu, die notwendigen Handgriffe im Umgang mit BASIC besser zu erlernen und den unmittelbaren Einstieg in die Arbeit mit Ton und Graphik zu erleichtern.

Auch in der Welt der Datenverarbeitung gilt das Sprichwort "Probieren geht über studieren". Sie lernen die Datenverarbeitung am einfachsten im Umgang mit einem Computer. Und sie erkennen die Chancen, die er Ihnen persönlich und beruflich eröffnet am besten, indem Sie ihn fordern, oder sich von ihm fordern lassen.

Bald werden Sie feststellen, daß Ihnen - gemessen an seinem Preis - kein anderer Computer mehr Möglichkeiten bietet als der CPC664 von Schneider. Zudem ist er sehr bequem zu bedienen. Übrigens: Wenn Sie Ideen, Vorschläge und Verbesserungen zum Produkt (Hardware oder Software) haben, schreiben Sie uns. Über jede Anregung würden wir uns freuen.

Wir wünschen Ihnen nun viel Freude und Erfolg.

Schneider
COMPUTER DIVISION

SCHNEIDER COMPUTER DIVISION

Vervielfältigung und Weitergabe - auch auszugsweise - dieses Handbuches oder des hierin beschriebenen Produktes bedürfen der vorherigen schriftlichen Zustimmung der Schneider Computer Division.

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.

Eine Zusammenstellung etwaiger Veränderungen oder Korrekturen dieses Handbuches ist gegen Einsendung eines frankierten und rückadressierten Briefumschlages von Schneider erhältlich.

Schneider
Computer Division
Silvastraße 1
8939 Türkheim 1

Alle Wartungs- und Servicearbeiten müssen von Schneider-autorisierten Händlern durchgeführt werden. Schneider 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. Schneider 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 und CP/M sind Warenzeichen von Digital Research Inc.
Z80 ist ein Warenzeichen von Zilog Inc.
IBM und IBM PC sind Warenzeichen von International Business Machines Inc.
AMSDOS, CPC664 und CPC 464 sind Warenzeichen von AMSTRAD.

Erste Ausgabe 1985

Herausgegeben von der Schneider Computer Division

Originalausgabe in Englisch

Original Copyright © 1985 AMSOFT, AMSTRAD Consumer Electronics plc.

Druck: Holzmann Druck Service, 8939 Bad Wörishofen

WICHTIG!

Vor Inbetriebnahme sind folgende Anweisungen unbedingt zu beachten.

Hinweise zur Inbetriebnahme

1. Netzstecker immer an einer geerdeten Steckdose anschließen (wie unter'Inbetriebnahme' beschrieben).

2. Achten Sie stets darauf, das System nicht an andere Stromversorgungen als 220-240 V~ 50 Hz anzuschließen.

3. Innerhalb der Einheit befinden sich keine vom Benutzer wartbare Teile. Versuchen Sie daher niemals, das Gerät zu öffnen. Alle Servicearbeiten sind von einem hierzu autorisierten Fachpersonal auszuführen.

4. Um eine übermäßige Ermüdung der Augen zu vermeiden, ist der Monitor soweit wie möglich vom Tastenfeld aufzustellen, wobei der Betrieb innerhalb eines ausreichend ausgeleuchteten Raumes zu erfolgen hat.
   Die HELLIGKEITS-Regelung des Monitors ist dazu auf einer geringstmöglichen Schaltstellung zu belassen.

5. Im Interesse Ihrer Datensicherheit sollte der Computer so stehen, daß das Diskettenlaufwerk nicht direkt vor dem Monitor steht, sondern rechts von ihm plaziert ist. Stellen Sie Ihren Computer nicht in der Nähe von elektrischen Störfeldern auf.

6. Diskettenlaufwerk und Disketten sind stets entfernt von magnetischen Feldern zu halten.

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

8. Verstellen oder verschließen Sie niemals die Ventilator-Schlitze des Diskettenlaufwerks.

9. Achten Sie darauf, daß die Geräte niemals innerhalb einer sehr heißen, kalten, feuchten oder staubigen Umgebung eingesetzt bzw. gelagert werden.

Hinweise für den Betrieb

(Lassen Sie sich nicht durch die Vielzahl der verwendeten Fachausdrücke innerhalb dieses Abschnittes verwirren; die Bedeutung dieser Hinweise wird mit dem Durcharbeiten dieses Handbuches klarer.)

1. Schalten Sie das System niemals ein bzw. aus, solange sich eine Diskette im Disketten-Laufwerk befindet, da andernfalls auf Ihrer Diskette Programme oder Daten zerstört werden könnten.

2. Duplizieren Sie Master-CP/M-System-Disketten, die wichtige Programme enthalten. Ein etwaiger Verlust bzw. eine Zerstörung der Diskette könnte mit hohen Kosten verbunden sein.

3. Achten Sie darauf, nicht versehentlich Ihre CP/M-Systemdiskette zu überschreiben, indem Sie sicherstellen, daß die Schreibschutz-Löcher auf der Diskette stets geöffnet sind.

4. Arbeiten Sie mit einem Doppellaufwerk, d.h. nach Erwerb einer zusätzlichen Schneider FD1-Einheit, so muß zuerst das zweite Diskettenlaufwerk eingeschaltet werden, bevor Sie den Computer in Betrieb nehmen.

5. Berühren Sie niemals die Oberfläche der Diskette, die sich innerhalb des Schutzgehäuses befindet.

6. Nehmen Sie niemals eine Diskette aus dem Laufwerk, während von dieser gelesen bzw. auf diese geschrieben wird.

7. Erinnern Sie sich stets daran, daß bei der Formatierung einer Diskette alle vorherigen Inhalte gelöscht werden.

8. Das interne Disketten-Interface beansprucht einen geringen Teil des RAM-Speichers, der aber möglicherweise auch von einigen Cassetten-Programmen, die für den CPC464 erstellt wurden, benötigt wird. Diese Cassetten arbeiten nicht einwandfrei mit der Einheit CPC664 + Cassette. In der Regel läuft jedoch Schneider-Software für den CP464 auch auf dem CPC664.

9. Das Lizenzabkommen für Ihre CP/M-Systemdiskette (die durch eine elektronisch codierte Seriennummer gekennzeichnet ist) erlaubt den Einsatz nur auf einem bestimmten Computer-System.

   Hieraus ist im wesentlichen zu entnehmen, daß Sie keiner anderen Person eine Kopie Ihrer serienmäßig codierten CP/M-Diskette überlassen dürfen.
   Lesen Sie deshalb sorgfältig die Bestimmungen des Abkommens bezüglich der Benutzerlizenz am Ende dieses Handbuches.

INHALTSVERZEICHNIS

Zu diesem Benutzerhandbuch

Kapitel 1 Grundlagen-Kurs

Inbetriebnahme
Anschluß der Peripherie-Einheiten
Handhabung des Diskettenlaufwerkes
Erläuterungen zur Tastatur
Laden der Software und des Programms "Welcome"
Einführung in die BASIC-Schlüsselwörter
Einführung zur Behandlung der Diskette
Betriebsarten, Farben und Graphiken
Sound (Klangeffekte)
Einführung in AMSDOS und CP/M

Kapitel 2 Weiterführungskurs

Schreiben eines einfachen Programmes
Weiterführung und nachträgliche Erklärungen
Arbeiten mit Tabellen
Einführung eines Menüs
Laden und Sichern von Variablen auf der Diskette

Kapitel 3 Komplette Aufstellung der BASIC-Befehle Schneider CPC664

Beschreibung der verwendeten Begriffe
Alphabetische Aufstellung der Befehle unter Einschluß von:
Schlüsselwort
Formale Syntax
Beispiel
Beschreibung
Besondere Hinweise
Zugehörige Befehle

Kapitel 4 Das Arbeiten mit Disketten und Kassetten

Sicherungs-Diskette
Arbeits-Diskette für System und Dienstprogramme
BASIC-Diskette
Fertige Schneider BASIC-Disketten
Disketten konfigurieren
Start und Autostart eines CP/M-Programms
Erläuterungen bezüglich der Kassetten

Kapitel 5 AMSDOS und CP/M

AMSDOS

Disketten-Inhaltsverzeichnis
Das Wechseln von Disketten
AMSDOS Dateinamen und Dateitypen
Aufbau des Dateinamens, Dateivorspann und Wild Cards
Beispiele zu AMSDOS-Befehlen in einem Programm
Sichern von Variablen und drucken des Bildschirminhaltes
Übersicht der externen AMSDOS-Befehle
Kopieren von Dateien
Übersicht der AMSDOS-Fehlermeldungen

CP/M Einführung

CP/M System-Spuren
Konfigurations-Sektor
Control-Codes von der Konsole
Einloggen einer Diskette
Direkte Befehle von der Konsole
Transiente Befehle
Kopieren von Dateien und Disketten
System-Verwaltung

Kapitel 6 Einführung in LOGO

Was ist LOGO ?
Dr. LOGO Prozeduren
Bearbeiten von Programmen und Prozeduren
Bedienungshinweise
Verzeichnis der Dr. LOGO Grundprozeduren

Verarbeitung von Wörtern und Listen
Arithmetische Operationen
Logische Operationen
Variable
Prozeduren
Editieren
Text-Bildschirm
Graphik-Bildschirm
Schildkröten-Graphik (Turtle)
Workspace-Verwaltung
Zuordnungslisten
Diskettendateien
Tastatur und Joystick
Sound (Klangeffekte)

Fortsetzung Kapitel 6 Dr. LOGO Grundprozessuren
Steuerfluß
Bearbeitung von Ausnahmen
Einfache System-Prozeduren
System-Variablen
System-Eigenwerte

Kapitel 7 Zu Ihrer Information

Cursor-Stellungen und Steuercode-Erweiterungen
Unterbrechungen
ASCII und Graphik Zeichensatz
Befehlsangaben
Sound (Klangeffekte)
Fehlermeldungen
BASIC-Schlüsselwörter
Programmhilfen
Drucker
Verbindungen (Connections)
Joysticks
Disketten-Organisation
Speicher

Kapitel 8 Wenn Sie gerade Zeit haben ...

Allgemeines
Die Vielzahl der Microcomputer
Hardware und Software
Vergleichende Rechner
Einige gängige Fehldeutungen
Wie verarbeitet ein Computer Ihre Befehle
Die Vielzahl der digitalen Begriffe
Bits und Bytes
Das BINÄRE Zahlensystem
Das HEXADEZIMALE Zahlensystem
Spezifische Funktionen des CPC664
Zeichensatz
Variable
Logik
Vom Benutzer definierte Zeichen
Druckformatierung
Fenster
Interrupts
Daten
Sound
Graphische Zeichen
Hardware

ANHANG

Anhang 1 Lizenz-Vereinbarung für den Benutzer

Anhang 2 Begriffserklärungen

Anhang 3 Einige Programme speziell für Sie zusammengestellt...

Tennis

Elektro-Zaun

Pontoon

Bomber

Amthello

Raffles

Anhang 4 Index

SCHNEIDER CPC664 GRUNDLAGENKURS

Teil 1: Inbetriebnahme

Das CPC664-System kann wahlweise betrieben werden mit:

1.	**Schneider GT65**	**Grünmonitor**
2.	**Schneider CTM644**	**Farbmonitor**
3.	**Schneider MP2**	**Modulator/Stromversorgung und einem normalen Fernsehgerät (UHF)**

Entfernen Sie keine Schrauben bzw. öffnen Sie nicht das Gehäuse des Computers, des Monitors oder Modulators.
Beachten Sie die Hinweisschilder an der Unter- bzw. Rückseite der Geräte.

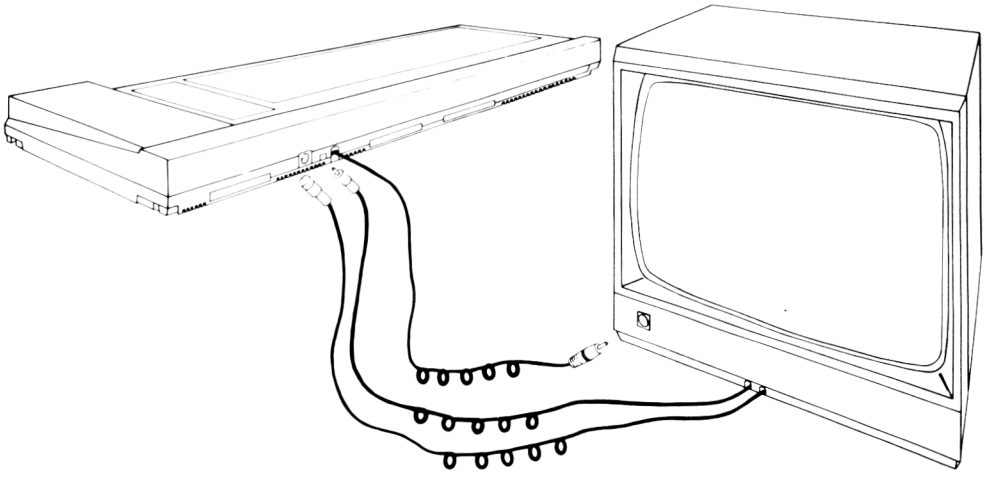
Achtung:

Stromführende Teile im Innern - keine Schrauben lösen.

Anschluß des Computers an einen Monitor

(Bei Einsatz des Systems CPC664 mit der MP2-Modulator/Stromversorgungseinheit ist der nächste Abschnitt maßgebend.)

1. Sicherstellen, daß die Netzzuleitung des Monitors nicht eingesteckt ist.
2. Die Leitung von der Vorderseite des Monitors, mit dem größeren Stecker (6-Anschlüsse, DIN) in die mit MONITOR bezeichnete Buchse des Computers einstecken.
3. Stecken Sie das Kabel mit dem kleineren Stecker (5V Gleichstrom), das vom Monitor kommt, in die mit 5V DC bezeichnete Buchse auf der Computer-Rückseite.
4. Verbinden Sie die Leitung von der Rückseite des Computers, die mit einem kleinen Stecker (12V DC) ausgerüstet ist, in die Buchse an der Vorderseite des Monitors.

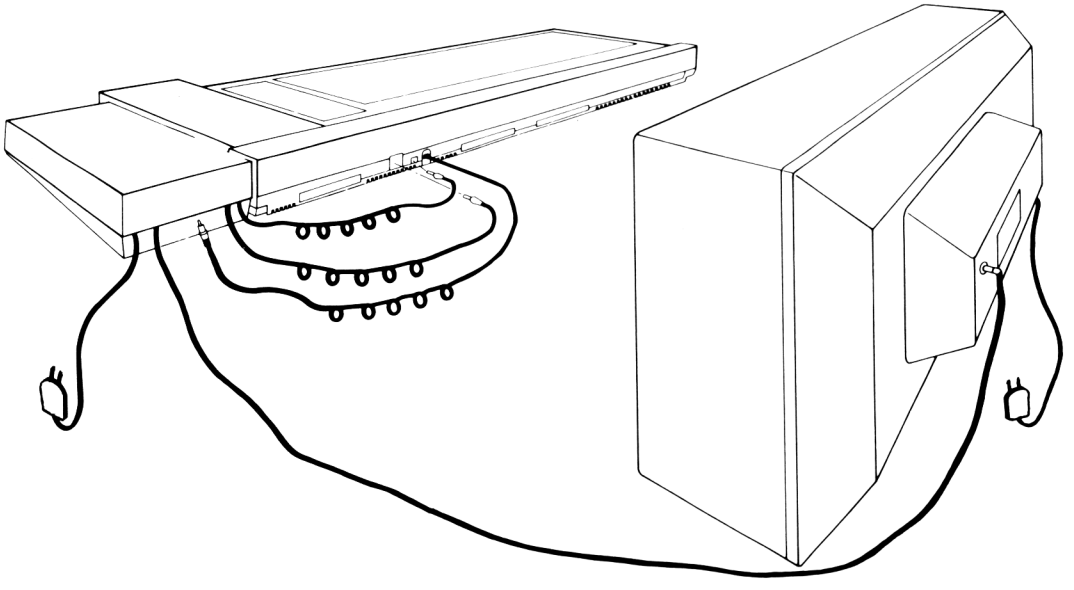


Anschluß des Computers an eine MP2-Modulator-Stromversorgungseinheit

Das Netzteil MP2 ist ein Zusatzgerät, mit dem Sie Ihren Computer CPC664 an einen Farbfernseher anschließen können. Damit brauchen Sie nicht auf die hervorragenden Farbeigenschaften zu verzichten, auch wenn Sie einen grünen GT65-Monitor benutzen.

Die MP2-Einheit sollte rechts neben der CPC664-Einheit aufgestellt werden. Bei der Installation des Gerätes sollten folgende Punkte beachtet werden:

1. Sicherstellen, daß das MP2 nicht mit der Netzsteckdose verbunden ist.
2. Verbinden Sie den größeren (6-Pin-DIN) Stecker des MP2 mit dem als MONITOR bezeichneten Anschluß am Computer.
3. Stecken Sie den kleineren Stecker (5V DC) des MP2 in die mit 5V DC bezeichnete Buchse auf der Rückseite des Computers.
4. Verbinden Sie das Kabel mit dem Antennenstecker des MP2 mit dem Antenneneingang Ihres Fernsehgerätes.
5. Verbinden Sie das 12V DC-Kabel von der Rückseite des Computers mit dem entsprechenden Anschluß am MP2.



Einschalten des Systems CPC664 und GT65/CTM644

Nach Anschluß des Systems gemäß den obigen Angaben ist der Netzstecker einzustecken. Drücken Sie den Knopf POWER an der rechten Seite des Monitors, so daß sich dieser in Stellung ON befindet.

Schalten Sie den Computer über den Schiebeschalter (POWER) auf der rechten Seite der Tastatur auf 'ON'.

Die rote Lampe auf der Computer-Tastatur, oben Mitte, sollte jetzt aufleuchten, wobei der Monitor das folgende Bild zeigt:

```
Schneider 64K Microcomputer  (v1)

©1984 Amstrad Consumer Electronics plc
and Locomotive Software Ltd.

BASIC 1.1

Ready
```

Um eine unnötige Belastung der Augen zu vermeiden, ist die mit BRIGHTNESS bezeichnete Steuerung soweit einzustellen, bis sich eine ausreichende Helligkeit ohne Blendung bzw. Verzerrung der Buchstaben auf dem Bildschirm ergibt.
Der Regler für die Helligkeit befindet sich beim GT65-Monitor auf der Vorderseite, bzw. beim CTM644 auf der rechten Seite.

Wenn Sie mit dem Grünmonitor GT65 arbeiten, könnte es erforderlich sein, CONTRAST und Bildfang am unteren Feld nachzustellen.

Die CONTRAST-Steuerung ist so einzustellen, daß sich für Sie eine angenehme Bildwiedergabe ergibt.

Die Bildfangregelung am GT65 ist mit V-HOLD bezeichnet und ist so zu steuern, daß sich die Anzeige genau in die Mitte des Bildschirms ohne zu zittern bzw. durchzulaufen einstellt.

Einschalten des CPC664 und MP2 Modulator/Stromversorgungssystem

Nach Anschluß des Systems in oben beschriebener Form ist der Netzstecker **anzuschließen.**

Einschalten des Computers unter Verwendung des Schiebeschalters POWER auf der rechten Seite der Tastatur (auf 'ON').

Die rote Lampe ON auf der Computer-Tastatur sollte jetzt aufleuchten, wobei Sie nun den Fernseher auf das vom Computer übertragene Signal einzustellen haben.

Verfügen Sie über einen Fernsehr mit Druckknopf-gesteuerter Kanalwahl, so ist ein Kanalknopf zur **Bestimmung** eines nicht genutzten Kanals zu drücken. Einstellung der entsprechenden Abstimmung in Übereinstimmung mit den Angaben des Fernsehgeräte-Herstellers (das Signal kann in etwa auf Kanal 36 liegen, sofern Ihr Fernsehgerät über eine solche Skala verfügt), wobei die Einstellung solange zu erfolgen hat, bis sich folgende Bild ergibt:

```
Schneider 64K Microcomputer  (v1)

©1984 Amstrad Consumer Electronics plc
and Locomotive Software Ltd.

BASIC 1.1

Ready
```

Stellen Sie Ihr Fernsehgerät genau ein, bis sich ein klares Bild ergibt.

Die Schrift erscheint gelb auf dunkelblauem Hintergrund.

Verfügt Ihr Fernsehgerät über einen drehbaren Programmwahlknopf, so ist dieser zu drehen, bis sich oben gezeigtes Bild ergibt **und stabil bleibt.** (Hierbei wiederum etwa auf Kanal 36).

Weitere Verbindungen

Falls Sie noch weitere Peripheriegeräte an das Standardsystem anschließen wollen, insbesondere:

Joystick(s)

Kassetteneinheit

Drucker

2. Diskettenlaufwerk

Externe Verstärker/Lautsprecher

Erweiterungsvorrichtung(en)

sind weitere Anleitungen in Teil 2 dieses Kurses angegeben.

Prüfen Sie bitte abschließend, ob alle am Anfang dieses Handbuches genannten **WICHTIGEN HINWEISE** eingehalten wurden.

Teil 2: Anschluß Ihrer Peripheriegeräte

Dieser Abschnitt erläutert den Anschluß der einzelnen Peripheriegeräte bzw. Zubehörelemente an den Computer CPC664.

Einzelheiten bezüglich des Einsatzes dieser Vorrichtungen sind in den jeweiligen Abschnitten dieses Handbuches enthalten.

Joystick

Den Schneider Joystick JY2 können Sie als Zubehör dazukaufen, wenn Sie auf Ihrem CPC664 Computerspiele einsetzen, die mit einem Joystick gespielt werden und bei denen man auch "schießen" kann.

Der JY2-Stecker wird an der mit JOYSTICK bezeichneten Buchse am Computer angeschlossen. Der Schneider CPC664 kann mit zwei Joysticks betrieben werden; der zweite Joystick kann in eine Buchse im ersten Joystick eingesteckt werden.

Der Schneider-Joystick, Typ JY1, kann ebenfalls an diesen Computer angeschlossen werden.

Weitere Angaben bezüglich der Joysticks werden im weiteren Verlauf dieses Handbuches beschrieben.

Cassetteneinheit

Anstelle von Diskette können von einer externen Cassetteneinheit Programme bzw. Daten eingelesen oder darauf abgespeichert werden. Die Befehle, über die der Computer Daten auf bzw. von der Diskette oder dem Band überträgt, werden im weiteren Verlauf dieses Handbuches beschrieben.

Zur Verbindung Ihrer Cassetteneinheit mit dem CPC664-System ist jede entsprechende Standard-Cassettenverbindungsleitung verwendbar.

Hierbei ist unbedingt darauf hinzuweisen, daß eine einwandfreie Datenübertragung zwischen CPC664 und der Cassette im wesentlichen von der korrekten Einstellung der Steuerungen LEVEL bzw. VOLUME auf Ihrer Cassetteneinheit abhängt. Falls es Ihnen nicht gelingen sollte, Programme einwandfrei abzuspeichern bzw. zu übernehmen, empfehlen wir, mit verschiedenen LEVEL-Kontrolleinstellungen zu arbeiten, bis die optimale Einstellung erreicht ist.

Hierbei ist unbedingt darauf hinzuweisen, daß eine einwandfreie Datenübertragung zwischen CPC664 und der Cassette im wesentlichen von der korrekten Einstellung der Steuerungen LEVEL bzw. VOLUME auf Ihrer Cassetteneinheit abhängt. Falls es Ihnen nicht gelingen sollte, Programme einwandfrei abzuspeichern bzw. zu übernehmen, empfehlen wird, mit verschiedenen LEVEL-Kontrolleinstellungen zu arbeiten, bis die optimale Einstellung erreicht ist.

Drucker

Der CPC664 kann im Prinzip an jeden Drucker angeschlossen werden, der mit einer standardisierten Centronics-Schnittstelle ausgerüstet ist.
Wenn Sie den Schneider Matrixdrucker NLQ401 mit dem CPC664 verbinden wollen, reicht hierzu eine einfache Verbindung mit der dem Drucker beigegebenen Anschlußleitung.

Der Schneider Matrixdrucker NLQ401 ist exakt auf den CPC664 abgestimmt: Er ist in der Lage, den deutschen Zeichensatz zu drucken; außerdem ist es möglich, sämtliche Graphikzeichen des CPC664 dank der Einzelnadelansteuerung des NLQ401 komplett auch "zu Papier" zu bringen.

Falls Sie irgendeinen anderen kompatiblen Centronics-Drucker einsetzen sollen, benötigen Sie ein entsprechendes Druckerverbindungskabel (z.B. das Schneider Druckerkabel).

Stecken Sie das mit einem flachen Stecker ausgerüstete Verbindungskabel in die mit PRINTER gekennzeichnete Buchse auf der Rückseite des Computers ein.

Verbinden Sie das andere Ende der mit einem Centronics-Stecker ausgerüsteten Leitung mit der Buchse auf der Rückseite des Druckers. Ist der Drucker mit Sicherheitsklammern auf jeder Seite der Buchse ausgerüstet, so sind diese in die Ausschnitte an der Seite des Druckersteckers einzusetzen.

Einzelheiten in Bezug auf den Betrieb des Druckers sind im weiteren Verlauf dieses Handbuches beschrieben.

Zweites Diskettenlaufwerk (Schneider FD1)

Eine Schneider FD1-Einheit kann innerhalb des Systems als zweites Diskettenlaufwerk betrieben werden. Die Vorteile eines zweiten Diskettenlaufwerkes zeigen sich insbesondere für die regelmäßige Verwendung als CP/M-System, da viele Programme mit Hilfe der Bibliotheksprogramm-Diskette innerhalb eines Laufwerkes arbeiten, während die Arbeitsdisketten auf einer Diskette im zweiten Laufwerk betrieben werden.

Operationen unter CP/M setzen immer voraus, daß ein Programm von Diskette geladen wird (auf das ROM - BASIC wird nicht zugegriffen). Da CP/M die Benutzung mehrerer Dateien erlaubt und durch eine spezielle Überlappungtechnik zuläßt, daß Programme größer sind als der RAM-Speicher, kann die Bibliotheksdiskette so viele Programmdateien enthalten, daß für die Daten nur noch wenig Raum verbleibt.

Durch die Vielseitigkeit der utilities, die Ihnen durch Ihre Systemdiskette zur Verfügung stehen, können Sie alle erforderlichen Dateivorgänge ausführen, wie z.B. Kopieren, Löschen, usw. und zwar auch über ein einziges Diskettenlaufwerk. Demgegenüber lassen sich jedoch durch Einsatz eines zweiten Diskettenlaufwerkes diese Prozesse beschleunigen und die Gefahr von Zwischenfällen verringern.

Für die Verbindung zwischen FD1 und CPC664 benötigen Sie die Verbindungsleitung Schneider DI2.

Verbinden Sie das Ende der Leitung (mit dem kleineren Stecker) mit der Buchse auf der Rückseite des FD1-Diskettenlaufwerkes.

Wichtige Hinweise:

Schalten Sie immer zuerst das Diskettenlaufwerk, dann den Computer ein.

Disketten vor dem Ein- oder Ausschalten des Systems immer aus dem Laufwerk entfernen.

Während eines Disketten-Zugriffs niemals den Auswurf-Knopf betätigen. Der Auswurf führt in diesem Fall zu Schreib- bzw. Lesefehlern.

Das System ausschalten, bevor Sie das Verbindungskabel anschließen oder abtrennen!

Einzelheiten des Betriebes mit zwei Diskettenlaufwerken werden im weiteren Verlauf dieses Handbuches beschrieben.

Externe Verstärker/Lautsprecher

Das CPC664-System kann an einen Stereo-Verstärker sowie an Lautpsrecher angeschlossen werden, um die 3-Kanal-Möglichkeiten des Computers voll auszunutzen.

Die Eingangsleitung Ihres Stereoverstärkers ist mit einem Stereostecker 3,5 mm auszurüsten, der in die mit 'STEREO' Computer bezeichnete Buchse einzustecken ist.

Hierzu ergeben sich folgende Steckeranschlüsse:

Steckerspitze - Linker Kanal

Innenring des Steckers - Rechter Kanal

Hintere Steckerhülse - Erde (gemeinsam)

Das CPC664-System liefert ein festes Spannungssignal an der STEREO-Buchse, wobei Sie die Steuerungen am Verstärker verwenden sollten, um Lautstärke, Balance und Tonhöhe einzustellen.

Kopfhörergarnituren mit hoher Impedanz arbeiten ebenfalls mit dem CPC664, jedoch läßt sich hierbei die Lautstärke nicht durch die Lautstärken-Steuerung am Computer abgleichen. Mit geringer Impedanz arbeitende Kopfhörergarnituren, wie sie normalerweise bei Stereosystemen verwendet werden, arbeiten nicht direkt bei Verbindung mit dem Computer.

Einzelheiten in Bezug auf den Abgleich der Tonskala auf den erforderlichen Ausgangskanal sind im weiteren Verlauf dieses Handbuches beschrieben.

Erweiterungen

Erweiterungen wie z.B. serielle Schnittstellen, Modems, Lichtgriffel, ROMs können an das CPC664-System unter Verwendung der mit EXPANSION bezeichneten Buchse auf der Rückseite des Computers verbunden werden.

Einzelheiten bezüglich der Anschlüsse an die Buchse EXPANSION sind im weiteren Verlauf des Kapitels mit dem Titel "Zu Ihrer Information" angegeben.

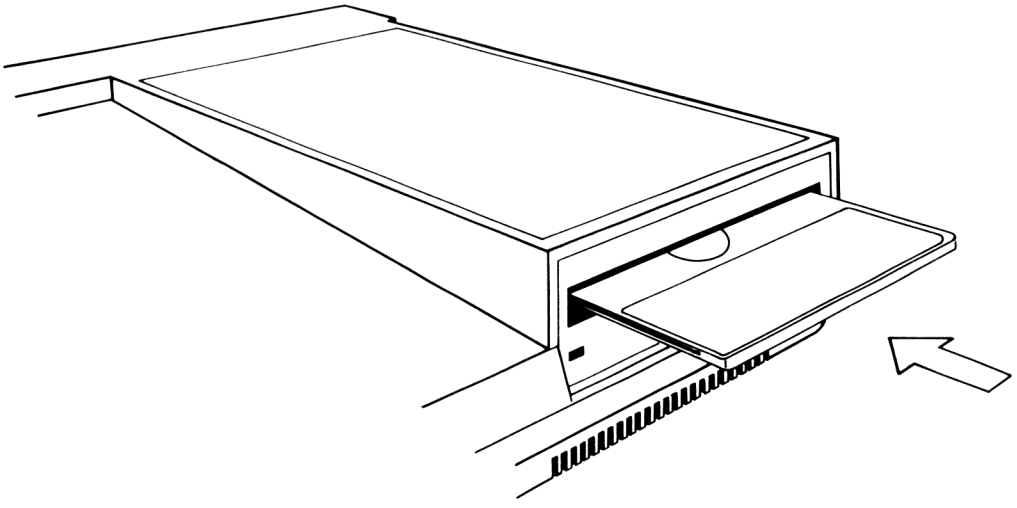
Abschließend ist zu prüfen, ob die Hinweise zu Beginn dieses Handbuches mit dem Abschnitt **WICHTIGE HINWEISE** eingehalten wurden.

Teil 3: Handhabung der Disketten

Das Schneider Disketten-Laufwerk CPC664 verwendet 3 Zoll-Kompakt-Floppy Disks. Für einen zuverlässigen Daten-Transfer zwischen Computer und Diskette empfehlen wir, Schneider CF2-Kompakt-Floppy Disks zu verwenden. Disketten anderer führender Hersteller sind jedoch auch verwendbar.

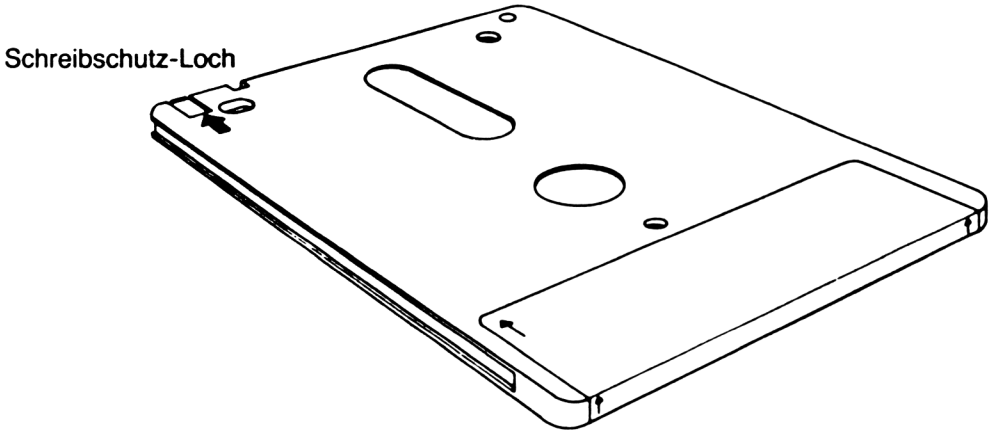
Einschieben

Jede Disketten-Seite kann separat verwendet werden. Eine Diskette muß so in das Laufwerk eingeschoben werden, daß ihr Etikett aus dem Laufwerk zeigt und die gewünschte Seite nach oben weist.



Schreibschutz

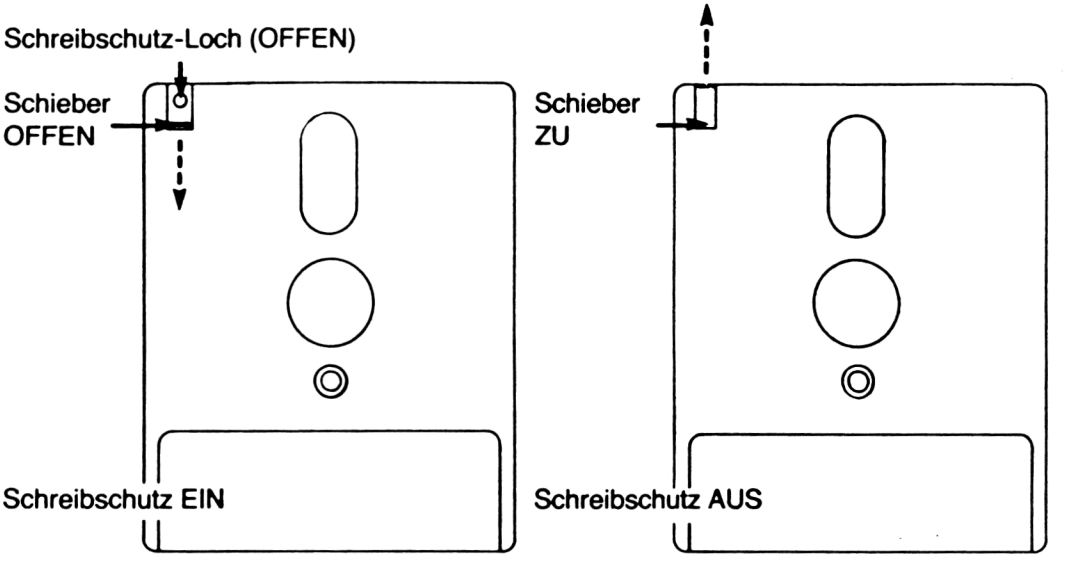
Am linken Rand jeder Diskette befindet sich ein mit einem Pfeil versehenes kleines Loch. Dies ist das sogenannte Schreibschutz-Loch, das vor Überschreiben oder Löschen gespeicherter Daten schützt:



Wenn das Loch geschlossen ist, lassen sich vom Computer Daten auf die Diskette schreiben. Ist das Loch jedoch offen, so können keine Daten auf die Diskette geschrieben werden. Hierdurch sind wertvolle Programme vor versehentlichem Löschen geschützt.

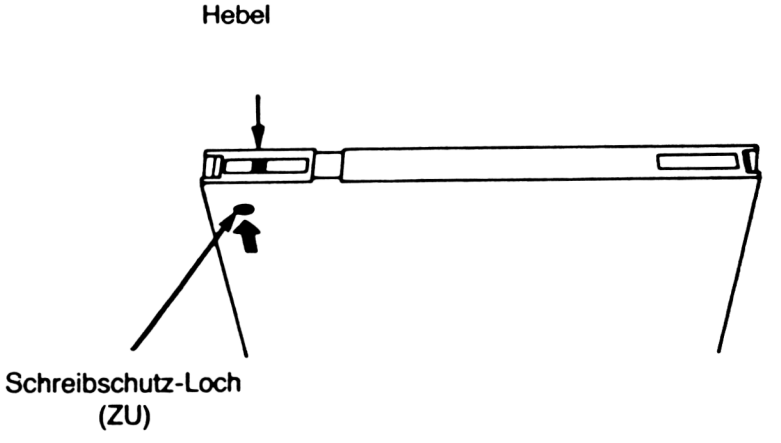
Einige Kompakt Floppy Disk-Hersteller liefern Disketten mit anderen Verschlußtechniken für die Schreibschutz-Löcher. Für die Schneider CP/M-Masterdiskette sind folgende Handgriffe erforderlich:

Zum Öffnen des Schreibschutzloches schieben Sie den kleinen Verschlußriegel am linken Rand der Diskette nach unten und das Loch ist geöffnet:

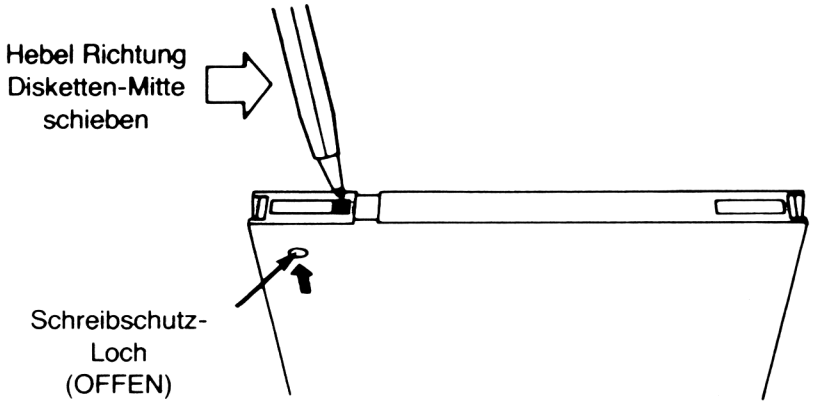


Zum Schließen des Schreibschutz-Loches schieben Sie den Verschlußriegel nach oben.

Die Schneider Leerdisketten CF2 haben am linken Rand der Diskette einen kleinen Kunststoff-Hebel in einem Schlitz:



Zum Öffnen des Schreibschutz-Loches muß bei diesem Diskettentyp der Hebel mit einer Bleistiftspitze o.ä. in Richtung Disketten-Mitte geschoben werden.



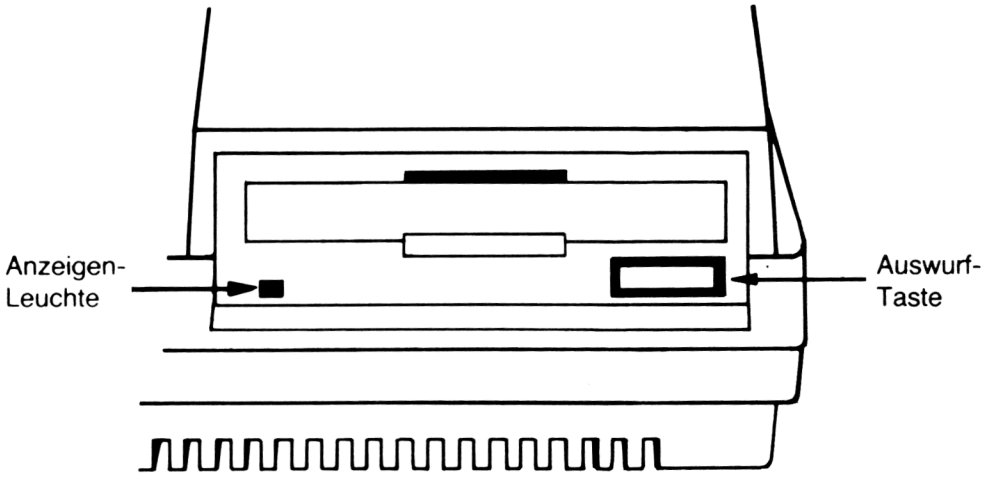
Unabhängig von der verwendeten Methode schützt ein geöffnetes Loch gegen jegliches Überschreiben.

WICHTIGER HINWEIS

Das Schreibschutz-Loch der Master CP/M-diskette muß immer geöffnet sein.

Bei eingelegter Diskette

Auf der Frontseite des Laufwerkes des Computers befindet sich eine rote Anzeigeleuchte und ein Druckknopf für den Disketten-Auswurf:



Anzeigenleuchte

Die rote Anzeigenleuchte hat zwei Funktionen:

Bei nur einem angeschlossenen Diskettenlaufwerk zeigt sie einen Lese- oder Schreibzugriff auf die Diskette an.

Bei zwei angeschlossenen Diskettenlaufwerken leuchtet die rote Anzeigeleuchte des zweiten Diskettenlaufwerkes ständig und gibt Ihnen an, welche Einheit Laufwerk B ist.
Wird jedoch auf eine Diskette zugegriffen, so übernimmt die Anzeigenleuchte ihre ursprüngliche Funktion und zeigte Lese- oder Schreibzugriffe an.

Disketten-Auswurfknopf

Durch Betätigen des Auswurfknopfes kann die Diskette aus dem Laufwerk entnommen werden.

Abschließend ist zu prüfen, inwieweit die am Beginn des Handbuches gegebenen Sicherheitsvorschriften mit dem Titel **WICHTIGE HINWEISE** eingehalten wurden.

Teil 4: Einleitung des Startvorganges

Vor dem Laden der Software und der Abspeicherung von Programmen auf die Diskette möchten wir Sie mit einigen Tasten am Computer vertraut machen. Diejenigen unter Ihnen, die bereits im Umgang mit Computern vertraut sind, können diesen Abschnitt überspringen.

Nach Einschalten des Computers und mit Erscheinen der ersten Meldung auf dem Bildschirm befassen wir uns mit der Bedeutung der einzelnen Tasten.

Pfeiltasten (Cursor-Tasten)

Die vier großen Pfeiltasten werden als "Cursor-Tasten" bezeichnet. Über diese Tasten wird die Stellung des Cursors (der kleine feste Block) auf dem Bildschirm verändert.

Drücken Sie nacheinander jeder der Cursor-Tasten und Sie werden feststellen, wie sich der Cursor auf dem Bildschirm bewegt.

[E N T E R]

Es gibt zwei [E N T E R] Tasten. Beide Tasten geben die Information, die Sie eingetippt haben, weiter an den Computer. Nachdem die [E N T E R] Taste gedrückt ist, wird eine neue Zeile am Bildschirm begonnen. Jede Anweisung, die Sie eintippen, muß mit [E N T E R] abgeschlossen werden.

Ab jetzt schreiben wir noch [E N T E R] , wenn eine Anweisung oder Programm-Zeile durch drücken der [E N T E R] -Taste abgeschlossen werden soll.

[D E L]

Diese Taste dient zum Löschen eines Schriftzeichens links vom Cursor auf dem Bildschirm (beispielsweise ein Buchstabe oder eine Zahl).

Geben Sie nacheinander a b c d ein und Sie sehen, daß der Buchstabe d sich links vom Cursor befindet. Sollten Sie den Buchsta-

ben d löschen wollen, so drücken Sie einmal [DEL] und Sie werden feststellen, daß der Buchstabe d verschwunden ist. Drücken Sie [DEL] und halten diese Taste, so werden die Buchstaben a b c d ebenfalls gelöscht.

[SHIFT]

Zwei [SHIFT] Tasten sind vorhanden. Wenn Sie eine dieser Tasten drücken und unten halten, während Sie ein Zeichen eingeben, wird ein Großbuchstabe oder das obere Symbol auf der Taste am Bildschirm angezeigt.

Tippen Sie den Buchstaben a ein, dann halten Sie die [SHIFT] Taste gedrückt und tippen Sie das a noch einmal. Am Bildschirm sehen Sie:

aA

Jetzt tippen Sie einige Leerstellen ein, indem Sie die Leertaste gedrückt halten. Benutzen Sie jetzt die oberste Tastenreihe mit Nummern. Tippen Sie die Ziffer 2 ein, dann halten Sie die [SHIFT] Taste gedrückt und tippen Sie die Ziffer 2 noch einmal. Am Bildschirm sehen Sie:

2"

Sie sehen nun, was passiert, wenn die [SHIFT] Taste gedrückt wird, und gleichzeitig eine zweite Taste gedrückt wird. Probieren Sie dies auch mit anderen Tasten aus.

[CAPS LOCK]

Diese Taste hat eine ähnliche Funktion wie die [SHIFT] Taste, jedoch braucht diese Taste nur einmal gedrückt zu werden. Ab dann werden alle eingetippten Buchstaben als Großbuchstaben ausgegeben, die Nummerntasten werden allerdings nicht als Symbole abgebildet. Drücken Sie [CAPS LOCK] und tippen Sie ein:

a b c d e f 1 2 3 4 5 6

Sie werden am Bildschirm merken, wie die Buchstaben in Großbuchstaben umgewandelt wurden, die Nummern bleiben unverändert. Falls einzelne Symbole nötig sind während die [CAPS LOCK] Funktion aktiv ist, brauchen Sie nur auf die [SHIFT] Taste zu drücken, wenn die gewünschte Symboltaste gedrückt ist. Tippen

Sie die folgenden Tasten ein, während Sie die [SHIFT] Taste gedrückt halten:

A B C D E F 1 2 3 4 5 6

Am Bildschirm sehen Sie:

A B C D E F ! " # $ % &

Wenn Sie jetzt zu Kleinbuchstaben zurückkehren möchten, brauchen Sie nur noch einmal auf die [CAPS LOCK] Taste zu drücken.

Falls Sie Großbuchstaben und Symbole eintippen wollen, ohne ständig auf die [SHIFT] Taste zu drücken, dann halten Sie die [CTRL] Taste gedrückt und tippen Sie auf die [CAPS LOCK] Taste. Geben Sie ein:

a b c d e f 1 2 3 4 5 6

Am Bildschirm sehen Sie:

A B C D E F ! " # $ % &

Während die [CTRL] und [CAPS LOCK] Tasten aktiv sind, können Sie weiterhin Ziffern eintippen, indem Sie das Zifferntastenfeld rechts vom Haupttastenfeld benutzen. Wenn Sie jetzt die [CTRL] Taste festhalten und die [CAPS LOCK] Taste drücken, kehren Sie zu der Verarbeitungsart zurück, in der Sie vorher waren - entweder [CAPS LOCK] oder Kleinbuchstaben. Sie kommen wieder zu Kleinbuchstaben von CAPS LOCK, indem Sie noch einmal auf die [CAPS LOCK] Taste drücken.

[CLR]

Die Taste löscht ein Zeichen innerhalb des Cursors.

Tippen Sie A B C D E F G H ein. Der Cursor steht rechts vom letzten eingetippen Zeichen (H). Drücken Sie jetzt auf die Cursorlinkstaste [←] vier Mal. Der Cursor steht auf dem E.

Beobachten Sie, wie der Buchstabe E innerhalb des Cursors noch sichtbar ist. Drücken Sie die [CLR] Taste einmal, und sehen Sie wie das E verschwindet, und die Buchstaben F G H jeweils eine Stelle nach links verschoben werden. Der Buchstabe F ist jetzt innerhalb des Cursors. Drücken Sie noch einmal auf die [CRL] Taste und halten Sie sie fest. Sehen Sie, wie das F ver-

schwindet, gefolgt vom G und H.

[E S C]

Die [E S C] Taste wird benutzt, um aus einer Funktion auszubrechen (escape). Drücken Sie einmal auf die [E S C] Taste und der Computer unterbricht seine laufende Funktion, er arbeitet jedoch weiter, sobald eine andere Taste gedrückt wird.

Zweimaliges auf die [E S C] Taste drücken verursacht, daß der Computer seine Funktion verläßt. Der Computer erwartet jetzt neue Anweisungen von Ihnen.

Jetzt drücken Sie zweimal auf die [E S C] Taste.

WICHTIG

Falls Sie den rechten Rand erreichen, indem Sie mehr als 40 Zeichen eintippen, erscheint das nächste Zeichen auf der nächsten Zeile am linken Rand. Sie sollen **NICHT** auf die [E N T E R] Taste drücken, wie Sie auf einer Schreibmaschine den Zeilenvorschub am rechten Zeilenende betätigen würden.

Der Computer fügt die Funktion automatisch für Sie ein, und wird auf ein unerwartetes [E N T E R] mit einer Fehlermeldung - normalerweise S Y N T A X E R R O R - entweder sofort oder beim Programmablauf reagieren.

SYNTAX ERROR (Syntaxfehler)

Die Meldung SYNTAX ERROR erscheint am Bildschirm, wenn der Computer nicht versteht, was Sie eingetippt haben. Tippen Sie z.B. ein:

p r i n t t [E N T E R]

Am Bildschirm erscheint:

S Y N T A X E R R O R

Die Meldung erscheint, weil der Computer p r i n t t nicht versteht.
Wenn Sie eine fehlerhafte Programmzeile eintippen, wie z.B.:

10 printt "abc" [ENTER]

erscheint die Meldung SYNTAX ERROR erst, wenn der Computer diese Anweisung verarbeitet, also dann, wenn das Programm läuft.

Tippen Sie ein:

run [ENTER]

Am Bildschirm erscheint:

```
SYNTAX ERROR in 10
10 printt "abc"
```

Diese Meldung sagt Ihnen, in welcher Zeile der Fehler aufgetreten ist und zeigt Ihnen die Programmzeile mit Korrekturcursor, so daß Sie den Fehler korrigieren können.

Drücken Sie auf die Cursorrechtstaste (→ bis der Cursor auf dem t in printt steht. Jetzt drücken Sie auf die [CLR] Taste, um das ungewollte t zu entfernen, und auf die [ENTER] Taste um die korrigierte Zeile wieder in den Computer einzugeben.

Tippen Sie run [ENTER] ein, und Sie sehen, daß der Computer die Anweisung jetzt akzeptiert und abc am Bildschirm ausgibt.

Teil 5: Laden der Software und der Spiele

An dieser Stelle begrüßen wir wieder diejenigen Leser, die den vorausgegangenen Abschnitt überspringen konnten!

Um Ihnen zu demonstieren, wie schnell Disketten-Software geladen wird, schalten Sie das System ein und legen anschließend die mitgelieferte CP/M-Master-Diskette mit der Seite 1 nach oben in das Diskettenlaufwerk ein.

Geben Sie ein:

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

Nach einigen Sekunden ist das Programm in den Speicher geladen. Beantworten Sie die Frage auf dem Bildschirm, ob Sie einen Grünmonitor verwenden mit Y für Ja, bzw. N für Nein. Danach sehen Sie eine Demonstration des Spieles "Roland in Time" auf dem Bildschirm. Vielleicht sind Sie davon so begeistert, daß Sie das Spiel gleich kaufen.

Wenn Sie genug gesehen haben, können Sie das Programm dadurch verlassen, daß Sie gleichzeitig die Tasten [CTRL], [SHIFT] und ESC drücken. Hierdurch erfolgt eine völlige Zurücksetzung des Computers, eine Einstellung, die Sie dann benutzen können, wenn Sie neu beginnen wollen. (Wenn Sie den Computer auf diese Weise zurücksetzen, ist es nicht erforderlich, irgendeine Diskette aus dem Laufwerk herauszunehmen.)

Wurde das Programm nicht geladen, interpretieren Sie bitte die Fehlermeldungen auf dem Bildschirm, um festzustellen, an welcher Stelle ein Fehler vorliegt.

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

Die Fehlermeldung bedeutet, daß Sie entweder die Diskette nicht korrekt eingelegt haben, oder daß Sie über ein Doppellaufwerk verfügen, und die Diskette in Laufwerk B eingelegt wurde.

```
ROINTIME.DEM not found
```

bedeutet, daß entweder die falsche Diskette (bzw. die falsche Seite der Diskette) eingelegt wurde, oder daß der Programm-Name falsch eingegeben wurde.

R O I N T I M E . D E M b a d c o m m a n d

bedeutet, daß Sie bei der Eingabe des Namens ROINTIME.DEM irgendwo ein überflüssiges Leer- oder Satzzeichen eingegeben haben.

T y p e m i s m a t c h

deutet auf ausgelassenes Anführungszeichen (") hin.

S y n t a x e r r o r

zeigt die falsche Eingabe des Wortes "run" an.

D r i v e A : r e a d f a i l
R e t r y , I g n o r e o r C a n c e l ?

bedeutet, daß der Computer die Disketten-Daten fehlerhaft gelesen hat. Prüfen Sie, ob die korrekte Diskette eingelegt ist und drücken Sie dann R für Retry. Diese Meldung erscheint dann erneut, wenn Ihre Diskette dadurch gestört wurde, daß das System ein- bzw. ausgeschaltet wurde, während sich die Diskette im Laufwerk befand.

Nachdem Ihnen der Vorgang des Duplizierens von Disketten bekannt ist, copieren Sie auf diese Art und Weise wichtige Programme, auf jeden Fall aber Ihre CP/M-Systemdiskette.

Laden der Schneider-Software und des WELCOME-Programms

Nachdem Ihr Interesse am Umgang mit dem Computer schon etwas geweckt ist, lassen Sie uns ein Spiel laden:

Legen Sie Ihre Software-Diskette in das Laufwerk ein und geben Sie ein:

r u n " d i s c " [E N T E R]

Nach einigen Sekunden wird Ihr Spiel geladen sein und zur Verfügung stehen.

Versuchen Sie die Eingabe von: r u n " d i s c " mit Seite 1 Ihrer Master-CP/M-Systemdiskette im Laufwerk und Sie werden die fortlaufende CPC664-Begrüßungsdemonstration sehen und hören.

Setzen Sie nach dieser Vorführung den Computer mit den Tasten [CTRL], [SHIFT] und [ESC] zurück.

In allen Fällen befinden sich jedoch die Ladeangaben am Bezeichnungsschild der Software-Diskette, das Sie stets sorgfältig beachten sollten.

Über den oben genannten Befehl (run "disc") läuft der größte Teil der Schneider Disketten-Software ab. Es könnte jedoch vorkommen, daß Sie ein Programm auf andere Weise laden müssen.

Teil 6: Befassen wir uns nunmehr mit dem Rechner ...

Wir sind inzwischen darüber informiert, was wir mit dem Rechner tun können und was wir nicht tun dürfen; wir wissen, wie dieser aufgestellt wird und wie Peripheriegeräte anzuschließen sind. Wir kennen bereits die Bedeutung einiger Tasten auf dem Computer und beherrschen das Laden der Software. Nunmehr werden wir uns mit einigen der Befehle befassen, die in den Computer eingegeben werden können, um weitere Ereignisse hervorzurufen...

Wie Sie oder auch ich kann der Computer nur Befehle in einer Sprache verstehen, die er kennt und diese Sprache wird als **BASIC** bezeichnet (eine Kurzbezeichnung für Beginners All Purpose Symbolic Instruction Code).

Die Bezeichnungen innerhalb des BASIC-Vokabulars werden als Schlüsselwörter bezeichnet, und jedes von ihnen hat für den Computer eine bestimmte Funktion. Alle Sprachen müssen den grammatikalischen Regeln entsprechen, wobei BASIC keine Ausnahme bildet. Hierbei bezieht sich die Grammatik auf einen "Syntax", wobei Ihnen der Computer stets mitteilt, wenn Sie einen Fehler begehen.

Eine Einführung in SCHNEIDER BASIC

Sie finden in Kapitel 3 "Komplette Aufstellung der BASIC-Befehle Schneider CPC664" eine ausführliche Beschreibung aller BASIC-Befehle, die im SCHNEIDER BASIC vorkommen. In diesem Kapitel wollen wir einige der gebräuchlichsten BASIC-Befehle vorstellen.

CLS

Tippen Sie c l s ein (clear screen: Bildschirm löschen). Sie können entweder Groß- oder Kleinbuchstaben verwenden. Dann drücken Sie auf [E N T E R]. Sie merken, daß der Bildschirm gelöscht wird, und das Wort R e a d y und der Cursor oben links erscheint neu.

PRINT

Diese Anweisung wird benutzt, um Zeichen, Wörter oder Figuren im Programm auf dem Bildschirm anzuzeigen. Tippen Sie die folgende Anweisung ein:

p r i n t "h a l l o" [ENTER]

Am Bildschirm erscheint dann:

h a l l o

Die Anführungszeichen " " werden benutzt, um dem Computer mitzuteilen, was ausgegeben werden soll. h a l l o erschien am Bildschirm, sobald [ENTER] gedrückt wurde. Tippen Sie c l s [ENTER] , um den Bildschirm wieder zu löschen.

RUN

Das eben gezeigte Programmbeispiel enthielt nur eine Zeile. Sobald jedoch die Taste [ENTER] gedrückt wurde, wurde der Befehl ausgeführt und anschließend vergessen. Es ist jedoch möglich, eine Reihe von auszuführenden Befehlen in einer bestimmten Reihenfolge im Computer zu speichern. Dies erfolgt durch Schreiben eines "Programms". Bei den in einem Programm vorkommenden Befehlen handelt es sich um die gleichen BASIC-Befehle, wie wir dargestellt haben, wobei jedoch vor jeder Befehlszeile eine Zeilennummer eingegeben wird. Umfaßt das Programm mehr als eine Zeile, so sagen diese Zeilennummern dem Computer, in welcher Reihenfolge das Programm ablaufen soll. Wenn [ENTER] gedrückt wird, so bleibt die Zeile im Speicher, bis das Programm ausgeführt ist. Geben Sie nunmehr ein:

1Ø p r i n t "h a l l o" [ENTER]

Sie werden feststellen, daß nach Drücken von [ENTER] hallo **nicht** auf dem Bildschirm erschien, jedoch in den Speicher des Computers als Einzeilenprogramm einging. Um dieses Programm auszuführen, muß das Wort r u n eingetippt werden.
Tippen Sie ein:

r u n [ENTER]

Sie sehen jetzt h a l l o auf dem Bildschirm.
Anstelle des ständigen Eingebens von p r i n t kann das Fragezeichen-Symbol verwendet werden, beispielsweise
1Ø ? "h a l l o" [ENTER]

LIST

Nachdem ein Programm eingespeichert wurde, ist es möglich, dieses Programm anzuschauen, um zu prüfen, was hierbei zwischengespeichert wurde. Tippen Sie ein:

L i s t [ENTER]

und am Bildschirm erscheint:

1Ø P R I N T "h a l l o"

wobei es sich um das Programm handelt, das im Speicher steht.

Beachten Sie, daß das Wort P R I N T jetzt in Großbuchstaben erscheint. Das bedeutet, daß der Computer jetzt P R I N T als bekanntes BASIC-Schlüsselwort erkannt hat.

Tippen Sie c l s [ENTER] ein, um den Bildschirm zu löschen. Dabei wird Ihr Programm nicht aus dem Speicher gelöscht, auch wenn es nicht mehr am Bildschirm zu sehen ist.

GOTO

Der G O T O Befehl sagt dem Computer, daß er von einer Zeile zu einer anderen springen soll, entweder um einige Zeilen zu überspringen, oder um eine Schleife zu bilden. Tippen Sie ein:

1Ø p r i n t "h a l l o" [ENTER]

2Ø g o t o 1Ø [ENTER]

Tippen Sie dann ein:

r u n [ENTER]

und Sie sehen, wie h a l l o wiederholt auf dem Bildschirm, Zeile für Zeile auf der linken Seite, erscheint. Der Grund hierfür liegt darin, daß die Zeile 2Ø des Programms den Computer anweist, auf Zeile 1Ø zu gehen, um von dort aus das Programm weiterzuführen.

Sie können Das Programm anhalten, indem Sie einmal [ESC] drücken.
Wenn Sie jetzt irgendeine andere Taste drücken, läuft das Programm weiter. Das Programm kann abgebrochen werden, indem Sie zweimal auf [ESC] drücken. Tippen Sie ein:
c l s [ENTER]
um den Bildschirm zu löschen.

Tippen Sie ein:

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

Sie können sehen, wie h a l l o hintereinander auf jeder Zeile den ganzen Bildschirm füllt.
Der Strichpunkt nach dem Anführungszeichen (;) sagt dem Computer, daß er das nächste Zeichen bzw. eine Gruppe von Zeichen unmittelbar hinter das vorangehende schreiben soll (ausgenommen, die nächste Zeichengruppe ist zu groß, um auf der gleichen Zeile untergebracht werden zu können).

Sie können das Programm abbrechen, wenn Sie zweimal [ESC] drücken. Geben Sie nochmals Zeile 10 ein; dieses Mal jedoch mit einem Komma (,) anstelle des Strichpunktes (;):

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

Sie werden feststellen, daß das Komma , dem Computer sagt, das nächste Zeichen (oder eine Zeichengruppe) im Abstand von 13 Spalten vom Beginn der ersten Zeichengruppe zu schreiben. Diese Einrichtung ist sehr hilfreich, wenn man Informationen zu verschiedenen Spalten ausgeben will.

Beachten Sie jedoch dabei, daß das nächste Zeichen um weitere 13 Spalten verschoben wird, wenn die Anzahl der Zeichen einer Gruppe 12 übersteigt.

Drücken Sie wieder zweimal auf [ESC] um das Programm abzubrechen. Um den Computerspeicher komplett zu löschen, halten Sie nacheinander die [SHIFT], [CTRL] und [ESC] Taste gedrückt, der Computer wird zurückgesetzt.

INPUT

Mit diesem Kommando sagen Sie dem Computer, daß eine Eingabe auf ihn wartet, z.B. die Antwort auf eine Frage. Tippen Sie ein:

INPUT

Mit diesem Kommando sagen Sie dem Computer, daß eine Eingabe auf ihn wartet, z.B. die Antwort auf eine Frage. Tippen Sie ein:

```
10 input "wie alt sind Sie"; Alter [ENTER]
20 print "Sie sehen jünger als": Alter;
         "Jahre aus." [ENTER]
run [ENTER]
```

Am Bildschirm sehen Sie:

```
Wie alt sind sie ?
```

Geben Sie ihr Alter ein, dann [ENTER]. Wenn Sie 18 sind, erscheint:

```
Sie sehen jünger als 18 Jahre aus.
```

Dieses Beispiel zeigt, wie man das input Kommando und eine numerische Variable anwendet. Das Wort alter wurde am Ende von Zeile 10 in den Speicher gebracht, so daß der Computer das Wort alter mit der eingegebenen Zahl in Verbindung bringen konnte. Obwohl wir den Ausdruck alter für das Alter benutzt haben, hätten wir genauso gut den Buchstaben b in unserem Beispiel benutzen können.

Setzen Sie den Computer zurück, um den Speicher zu löschen (mit den [CTRL], [SHIFT] und [ESC] Tasten). Falls Sie als Eingabe (input) Buchstaben, bzw. Buchstaben und Ziffern gewünscht hätten, so hätten Sie am Ende der Variablen das Dollarzeichen $ anfügen müssen.

Tippen Sie das folgende Programm ein: (Achten Sie darauf, in Zeile 20 nach dem o in hallo und vor dem i in ich eine Leerstelle einzugeben).

```
10 input "Wie heissen Sie"; name$ [ENTER]
20 print "Hallo"; name$; ",ich heisse Peter" [ENTER]
run [ENTER]
```

Am Bildschirm sehen Sie:

```
Wie heissen Sie?
```

Geben Sie Ihren Namen ein: (z.B. Fred)

Am Bildschirm erscheint:

```
Hallo Fred ich heisse Peter
```

Oben haben wir name$ für die Zeichenvariable Name benutzt, obwohl es genau so gut mit einem Buchstaben, z.B. a$, gegangen wäre. Wir wenden jetzt die letzten beiden Beispiele in einem Programm an.

Setzen Sie den Computer zurück mit den [CTRL], [SHIFT] und [ESC] Tasten, und tippen Sie ein:

```
5 cls [ENTER]
10 input "Wie heissen Sie"; a$ [ENTER]
20 "Wie alt sind Sie"; b [ENTER]
30 print "Ich muss sagen,";a$;" Sie
   sehen nicht wie"; b;"Jahre aus" [ENTER]
run [ENTER]
```

Wir haben zwei Variable in diesem Programm benutzt, a$ für den Namen und b für das Alter. Am Bildschirm sehen Sie:

```
Wie heissen Sie?
```

Geben Sie Ihren Namen (z.B. Fred) und [ENTER] ein, dann werden Sie gefragt:

```
Wie alt sind Sie?
```

Geben Sie Ihr Alter (z.B. 18) und [ENTER] ein. Wenn Sie Franz heißen und 18 sind, sehen Sie jetzt am Bildschirm:

```
Ich muss sagen, Fred, Sie sehen nicht
wie 18 Jahre aus
```

EDITIEREN EINES PROGRAMMS

Falsch eingegebene Programmzeilen, die Syntaxfehler oder sonstige Fehler verursacht haben, können korrigiert (editiert) werden, anstatt sie neu einzutippen. Um zu zeigen, wie so etwas geht, tippen Sie das vorige Programm folgendermaßen fehlerhaft ein:

```
5 clss [ENTER]
10 input "Wie heissen Sie"; a$ [ENTER]
20 input "Wie alt sin Sie"; b [ENTER]
30 print "Ich muss sagen,";a$;",Sie
   sehen nicht wie";b "Jahre aus" [ENTER]
```

Wir haben 3 Fehler im Programm:

In Zeile 5 haben wir c l s s anstatt c l s eingetippt.
In Zeile 20 haben wir s i n anstatt s i n d eingetippt.
In Zeile 30 haben wir die Leerstelle zwischen s a g e n und " vergessen.

Wir haben 3 Möglichkeiten, ein Programm zu editieren. Die erste ist, die Zeile komplett neu einzutippen. Wenn eine Zeile eingetippt wird, ersetzt sie eine ggf. vorhandene Zeile im Speicher mit der gleichen Nummer. Die zweite Möglichkeit ist die EDIT Methode und drittens haben wir die sog. COPY Cursor Methode.

EDIT-Methode

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

```
edit 5 [ENTER]
```

Zeile 5 wird ausgedruckt mit dem Cursor auf der 5. Drücken Sie die Cursorrechtstaste [→], bis der Cursor auf dem zusätzlichen s in c l s s steht, und drücken Sie jetzt auf die CLR Taste. Das s verschwindet. Drücken Sie jetzt [ENTER] und Zeile 5 wird im Speicher korrigiert.

Tippen Sie ein:

```
list [ENTER]
```

um die Korrektur der Zeile 5 zu überprüfen.

Der im weiteren Verlauf dieses Handbuches beschriebene Auto-Befehl kann zur Korrektur einer Anzahl aufeinanderfolgender

Zeilen in gleicher Weise verwendet werden wie die eben beschriebene EDIT-Methode.

COPY Cursor-Methode

Bei dem Copy Cursor handelt es sich um einen zweiten Cursor(zusätzlich zu dem sich bereits auf dem Bildschirm befindlichen), der dann sichtbar wird, wenn Sie die Taste [SHIFT] gedrückt halten und eine der Cursor-Tasten betätigen. Er trennt sich vom Haupt-Cursor und kann unabhängig auf dem Bildschirm bewegt werden.

Um die Fehler in den Zeilen 2Ø und 3Ø zu korrigieren, drücken Sie gleichzeitig auf die [SHIFT]-Taste und auf die Cursor oben-Taste[↑]bis der Copy Cursor auf den Anfang Zeile 20 zeigt. Sie werden feststellen, daß sich der Haupt-Cursor nicht bewegt hat, und daß sich jetzt zwei Cursor auf dem Bildschirm befinden. Drücken Sie jetzt die [COPY]-Taste, bis der Copy-Cursor auf der Leerstelle zwischen s i n und s i e steht. Sie werden feststellen, daß Zeile 20 nochmals auf der untersten Zeile geschrieben wird und der Hauptcursor an der gleichen Stelle hält wie der COPY-Cursor. Geben Sie nunmehr den Buchstaben d ein. Dies wird nur auf der untersten Zeile eingefügt.

Der Hauptcursor hat sich bewegt, aber der COPY-Cursor ist stehengeblieben. Drücken Sie jetzt auf die [C O P Y] Taste bis die Zeile 20 vollständig geschrieben ist. Drücken Sie die [E N T E R] Taste und diese neue Zeile 20 wird in den Speicher übernommen. Der COPY-Cursor verschwindet und der Hauptcursor steht jetzt unter der neuen Zeile 20.

Der verbleibende Fehler kann in ähnlicher Weise korrgiert werden. Drücken Sie wieder gleichzeitig auf die [S H I F T] Taste und auf die Cursorobertaste[↑]bis der COPY-Cursor auf den Anfang von Zeile 30 zeigt.

Drücken Sie die [C O P Y] Taste bis der COPY-Cursor über dem Anführungszeichen" neben s a g e n steht. Drücken Sie einmal auf die Leerstelle. Eine Leertaste wird in der untersten Zeile eingefügt. Drücken Sie erneut die [C O P Y] Taste bis die Zeile 30 komplett übernommen ist. Dann drücken Sie auf [E N T E R] . Sie können das ganze Programm überprüfen, indem Sie l i s t [E N T E R] eintippen.

ANMERKUNG: Um während des Korrigierens den Cursor schnell an den linken bzw. rechten Rand einer Zeile zu bewegen, drücken Sie die [C T R L] -Taste und anschließend einmal die linke bzw. rechte Cursor-Taste.

Setzen Sie anschließend den Computer mit [C T R L] ,[S H I F T] und [E S C] zurück.

IF THEN

Mit Hilfe des Kommandos IF und THEN wird der Computer aufgefordert, eine bestimmte Bedingung zu überprüfen, um anschließend die entsprechende Aktion als Ergebnis dieser Prüfung auszuführen. Innerhalb des folgenden Befehles ergibt sich z.B.:

if 1+1=2, then print "richtig" [E N T E R]

Der Computer prüft nunmehr diese angegebene Bedingung und führt die entsprechende Aktion aus.

Das Schlüsselwort ELSE kann verwendet werden, um das IF THEN-Kommando über die jeweilige Aktion zu informieren, die der Computer ausführen soll, wenn die geprüfte Bedingung falsch ist. Beispielsweise:

if 1+1=Ø, then print "richtig" else print "falsch" [E N T E R]

Wir erweitern nunmehr das vorhergehende Programm, indem wir die neuen Kommandos IF und THEN einbauen.

Tippen Sie folgendes ein: Beachten Sie, daß wir zwei neue Symbpole eingefügt haben, < bedeutet kleiner als und ist neben der M-Taste (unter gleichzeitiger Benutzung des SHIFT -Taste), > bedeutet größer als und steht neben der < Taste (kleiner als).

```
5 CLS
10 INPUT "Wie heissen Sie";a$
20 INPUT "Wie alt sind Sie";alter
30 IF alter < 13 THEN 60
40 IF alter < 20 THEN 70
50 IF alter > 19 THEN 80
60 PRINT "Also ";a$;", du bist mit";alter;
"Jahren noch kein Teenager":END
70 PRINT "Also ";a$;", du bist ein Teenager
 mit";alter;"Jahren":END
80 PRINT "Also ";a$;", mit"alter"sind Sie
kein Teenager mehr"
```

Überprüfen Sie Ihr Programm mit:

list [ENTER]

Jetzt tippen Sie ein:

run [ENTER]

Beantworten Sie jetzt alle Fragen des Computers und sehen Sie, was passiert.

Sie sehen jetzt, welche Funktionen IF und THEN im Programm haben. Wir haben auch das Wort END am Ende der Zeile 60 und 70 hinzugefügt. Das Schlüsselwort END bedeutet wortwörtlich "beende das Programm". Ohne END in Zeile 60 würde das Programm weiterlaufen und die Zeilen 70 und 80 bearbeiten.

Ebenso würde ohne END am Ende von Zeile 70 das Programm weiterlaufen und Zeile 80 ausführen. Der Doppelpunkt: erlaubt es Ihnen, zwei oder mehr Anweisungen in eine Programmzeile zu schreiben. Der Doppelpunkt : vor dem Schlüsselwort END trennt zwei Anweisungen voneinander. Wir haben auch die Zeile 5 eingebaut, um den Bildschirm am Programmanfang zu löschen. Wir werden dies von jetzt an in den folgenden Programmen immer tun, damit der Bildschirminhalt übersichtlicher aussieht.

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

FOR NEXT

Der Befehl FOR und NEXT werden dann eingesetzt, wenn Sie einen bestimmten Vorgang in einer bestimmten An zahl von Fällen ausführen wollen. Die Befehle für die jweilige Operation müssen in der Schleife FOR NEXT enthalten sein.

Geben Sie ein:

```
 5 cls [ENTER]
10 for a=1 to 10 [ENTER]
20 print "Operation Nummer"; a [ENTER]
30 next a [ENTER]
run [ENTER]
```

Sie sehen, daß das Kommando in Zeile 20 zehnmal ausgeführt wurde, wie durch den FOR-Befehl in Zeile 10 angegeben. Außerdem werden Sie feststellen, daß sich der Wert der Variablen a je-

weils um 1 erhöht hat.

Das Schlüsselwort STEP kann dafür eingesetzt werden, den FOR NEXT-Befehl darüber zu informieren, um wieviel die Variable 'pro Schritt' erhöht werden soll. Ändern Sie beispielsweise Zeile 10 auf:

```
10 for a=10 to 50 Step 5 [ENTER]
run [ENTER]
```

Es können auch negative Schritte eingesetzt werden, z.B.:

```
10 for a=100 to 0 Step - 10 [ENTER]
run [ENTER]
```

REM

REM ist die Kurzform für REMark. Dieser Befehl veranlaßt den Computer die diesem Kommando folgenden Zeichen nicht zu beachten. Somit können Sie den Befehl REMark dazu verwenden, den Titel eines Programms oder die Bedeutung einer Variablen wie folgt zu beschreiben:

```
10 REM Invasion aus dem All [ENTER]
20 L = 5 : REM Anzahl der Leben [ENTER]
```

Das einfache Anführungszeichen ' (das durch Drücken von SHIFT und der Taste 7 ausgegeben wird) kann als Ersatz für : REM verwendet werden, z.B.:

```
10 'Invasion aus dem All [ENTER]
20 L=5 'Anzahl der Leben [ENTER]
```

GOSUB RETURN

Handelt es sich innerhalb eines Programms um eine Reihe von Befehlen, die mehrere Male auszuführen sind, so müssen diese Befehle nicht jedesmal erneut eingegeben werden, wenn sie innerhalb des Programmes benötigt werden; anstelle dessen können diese in ein "Unterprogramm" aufgenommen werden, das über den GOSUB-Befehl mit der entsprechenden Zeilennummer aufgerufen wird. Das Ende der GOSUB-Routine wird durch den Befehl RETURN markiert. An dieser Stelle kehrt der Computer zu dem Befehl zurück, der unmittelbar dem GOSUB-Kommando folgte, das gerade ausgeführt wurde.

(Die folgenden beiden Programme, die Sie nicht extra eingeben müssen, bewirken nicht mehr als ein Ausdrucken von Wörtern auf dem Bildschirm. Sie sollen nur darstellen, in welcher Form über Unterprogramme wiederholt benötigte Abläufe aufgerufen werden können.)

In dem folgenden Programm werden Sie eine Anzahl von Zeilen erkennen, die an verschiedenen Programmpunkten zu wiederholen sind, beispielsweise die Zeile 18Ø. Wir werden diese Zeile als Unterprogramm aufnehmen und den Befehl RETURN am Ende hinzufügen. Anschließend brauchen wir dieses Unterprogramm über den Befehl GOSUB 18Ø nur dann aufzurufen, wenn wir es einsetzen wollen. Dieses Programm sieht dann folgendermaßen aus.

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

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

Sicher werden Sie jetzt feststellen, wieviel Schreibarbeit wir uns hiermit erspart haben. Gut organisierte Unterprogramme sind für den Umgang mit dem Computer sehr wesentlich. Sie führen zu strukrurierten und übersichtlichen Programmen.

Denken Sie bei der Erstellung von Unterprogrammen immer daran, daß Sie nicht immer am gleichen Punkt in das Unterprogramm einspringen müssen. Ein Unterprogramm, zwischen den Zeilen 5ØØ bis 8ØØ kann z.B. mit GOSUB 5ØØ bzw. GOSUB 64Ø bzw. GOSUB 79Ø aufgerufen werden, wenn Sie nur bestimmte teile des Unterprogramms benötigen.

Im obengenannten Programm wurde der END-Befehl in Zeile 17Ø eingesetzt. Ansonsten würde das Programm nach 16Ø Zeile 18Ø ausführen. Die Zeilen ab 18Ø sollen aber nur ausgeführt werden, wenn das Unterprogramm mit GOSUB aufgerufen wird.

EINFACHE ARITHMETIK

Ihr CPC664 kann sehr einfach als Tischrechner eingesetzt werden.

Die folgenden Beispiele erläutern das näher. Wir verwenden in diesem Abschnitt das ? anstatt PRINT zu schreiben. Die Antwort wird ausgegeben, sobald die [ENTER] Taste gedrückt wird.

ADDITION

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

Tippen Sie:

```
? 3 + 3    [ENTER]
  6
```

(das Gleichheitszeichen = wird nicht eingegeben)

Tippen Sie:

```
? 8 + 4    [ENTER]
  1 2
```

SUBTRAKTION

(benutzen Sie die =Taste ohne Shift für Minus)

Tippen Sie:

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

Tippen Sie:

```
? 8 - 4    [ENTER]
  4
```

MULTIPLIKATION

(benutzen Sie die [SHIFT] und die :Taste zum Multiplizieren, * bedeutet x)

Tippen Sie:

```
? 3 * 3    [ENTER]
  9
```

Tippen Sie:

```
? 8 * 4    [E N T E R]
  3 2
```

DIVISION

(benutzen Sie die ?Taste ohne Shift zum Dividieren, /bedeutet ÷)

Tippen Sie:

```
? 3 / 3    [E N T E R]
```

Tippen Sie:

```
? 8 / 4    [E N T E R]
  2
```

INTEGER DIVISION

Verwenden Sie \ zur Division mit ganzzahligem Ergebnis (integer)

Geben Sie ein:

```
? 1 0 \ 6    [E N T E R]
  1
```

Geben Sie ein:

```
? 2 0 \ 3    [E N T E R]
  6
```

MODULE

Ergibt den ganzzahligen Rest einer Division.

Tippen Sie ein:

```
? 1 Ø  M O D  4
  2
```

Tippen Sie ein:

```
? 9  M O D  3
  Ø
```

QUADRATWURZEL

Sie errechnen die Quadratwurzel einer Zahl, indem Sie s q r ( ) eintippen. Die Zahl, deren Quadratwurzel Sie möchten, muß in Klammern stehen.

Tippen Sie:

```
? s q r ( 1 6 )   [E N T E R]   (das bedeutet √16)
  4
```

Tippen Sie:

```
? s q r  ( 1 Ø Ø )   [E N T E R]
  1 0
```

POTENZEN

(benutzen Sie die ↑ Taste ohne Shift zum Potenzieren)

Das Produkt aus n gleichen Faktoren b wird als Potenz von b bezeichnet z.B. 3 Quadrat (3^2), 3 Kubik (3^3), usw., tippen Sie:

```
? 3 ↑ 3   [E N T E R]  (das bedeutet 3³)
  2 7
```

Tippen Sie:

```
? 8 ↑ 4   [E N T E R]  (das bedeutet 8⁴)
  4 Ø 9 6
```

KUBIKWURZEL

Die Kubikwurzel wird ähnlich berechnet wie die Quadratwurzel.

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

```
? 2 7 ↑ ( 1 / 3 )   [E N T E R]
  3
```

Für die Kubikwurzel von 125 tippen Sie:

```
? 1 2 5 ↑ ( 1 / 3 )   [E N T E R]
  5
```

GEMISCHTE KALKULATION

(+,-,*,/)

Gemischte Kalkulationen werden vom Computer verstanden, werden jedoch unter Berücksichtigung einiger Prioritäten berechnet.

Höchste Priorität hat die Multiplikation, danach die Division, dann die Addition und letztlich die Subtraktion. Diese Prioritäten gelten jedoch nur, wenn nur diese vier Rechnungsarten vorhanden sind. Die Prioritäten werden später einschließlich Potenzen usw. weiter erläutert.

Nehmen wir folgende Berechnung

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

Vielleicht denken Sie, daß das so gerechnet wird:

```
3 + 7 - 2   * 7   / 4
=       8   * 7   4
=           5 6   / 4
=           1 4
```

in Wirklichkeit wird folgendermaßen gerechnet:

```
3 + 7 - 2 * 7 / 4
= 3 + 7 - 1 4 / 4
= 3 + 7 - 3 , 5
= 1 Ø - 3 , 5
= 6 . 5
```

Sie können dies nachvollziehen, indem Sie folgendes eintippen:

```
? 3 + 7 - 2 * 7 / 4    [E N T E R]
  1 4
```

Die Priorität sämtlicher mathematischer Operatoren lautet wie folgt:

↑	Potenzierung
MOD	Modulus
-	Minus (erklärt eine Zahl als negativ)
*und /	Multiplikation und Division
\	Integer Division
+ und -	Addition und Subtraktion

WEITERE POTENZEN

Falls Sie sehr große oder sehr kleine Zahlen brauchen, ist eine andere Schreibweise vorteilhafter. Der Buchstabe E wird als Potenz zur Basis 10 verstanden. Sie dürfen sowohl das große E als auch das kleine e verwenden.

Die Zahl 300 kann z.B. auch als $3x10^2$ geschrieben werden. In der wissenschaftlichen Schreibweise schreibt man diese 3E2. Ähnlich ist es mit 0,03 (auch $3x10^{-2}$). In der neuen Schreibweise haben wir 3E-2. Probieren Sie folgende Beispiele aus:

1. 30x10

Sie können eintippen:

```
? 3 Ø * 1 Ø  [E N T E R]
  3 Ø Ø
```

oder Sie können eintippen:

```
? 3 e 1 * 1 e 1  [E N T E R]
  3 Ø Ø
```

2. 3000x1000

Tippen Sie:

```
? 3 e 3 * 1 e 3  [E N T E R]
  3 Ø Ø Ø Ø Ø Ø
```

3. 3000x0,001

Tippen Sie:

```
? 3 e 3 * 1 e - 3     -[E N T E R]
```

Teil 7: SPEICHERN SIE AB ...

Sie haben nunmehr über die Eingabe einiger Befehle Ihre Finger soweit geübt und sind sicher interessiert, zu erfahren, wie ein Computerprogramm auf Diskette abgespeichert wird und wie dieses von der Diskette wiederum in den Computer geladen werden kann.

Selbst wenn Sie bereits mit den Vorgängen des Abspeicherns und Ladens der Kassette vertraut sind, sollten wir Ihre Kenntnisse noch durch einige Informationen ergänzen, die dann von Bedeutung sind, wenn es sich um Disketten-Dateien handelt.

Zuerst die beiden wesentlichen Unterschiede: Eine neue Diskette kann nicht ohne weiteres aus ihrer Verpackung herausgenommen und bespielt werden, wie dies bei normalen Kassetten der Fall ist. Eine neue Diskette ist zuerst zu "formatieren"; dieser Vorgang wird jetzt gleich anschließend erläutert.

Ein anderer, an dieser Stelle wichtiger Punkt ist die Bedeutung einer korrekten Namensgebung der Disketten-Dateien. Dateibezeichnungen von Kassetten sind an keine besonderen Namen gebunden, in ihrer Länge frei wählbar oder können sogar ganz weggelassen werden.
Anders verhält es sich mit Disketten. Die Dateibezeichnungen von Disketten müssen genau den CP/M-Normen entsprechen und werden im weiteren Verlauf dieses Abschnittes erläutert.

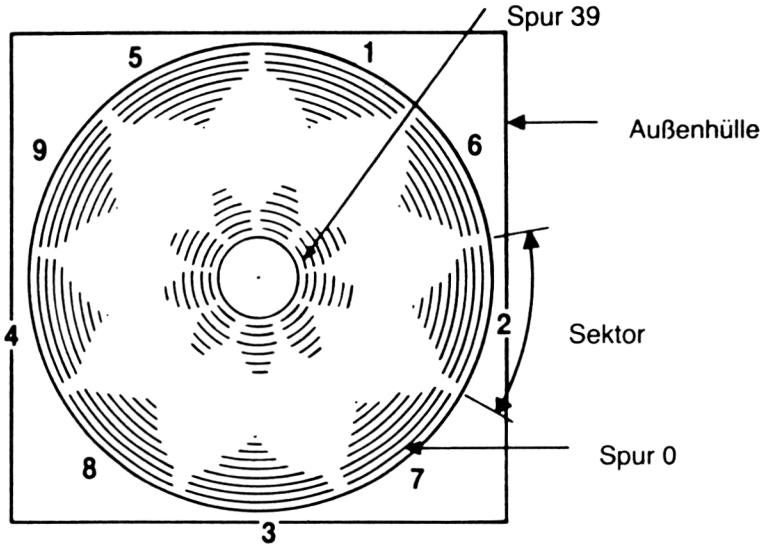
Formatierung und Datenorganisation für den Einsatz der Diskette

Bevor Daten auf eine neue Leer-Diskette geschrieben werden können, muß diese formatiert sein. Formatierung kann als Erzeugung von Fächern und Unterteilungen auf der Diskette verstanden werden, in die dann die Informationen eingespeichert werden können. Damit wird sozusagen ein untergliedertes Gerät zum Ablegen der Daten angelegt.

Die Formatierung unterteilt eine Disketten-Seite in 360 separate Bereiche.

Die Diskette besteht aus 40 Spuren von Spur 0 (außen) bis Spur 39 (innen) und 9 Sektoren.

Jede Spur eines Sektors kann bis zu 512 Bytes Daten speichern. Die gesamte verfügbare Speicherplatz auf einer Disketten-Seite beträgt demnach 180 KBytes.



Erste Schritte für den Einsatz der CP/M-Disketten

Zum Speichern Ihrer Programme auf einer neuen Leer-Diskette muß diese mit Hilfe der CP/M-Diskette formatiert werden.

Schalten Sie hierzu das System ein und legen Sie die mitgelieferte CP/M-Diskette in das Laufwerk ein.

Arbeiten Sie mit zwei Laufwerken, so führen Sie die CP/M-Diskette immer in Laufwerk A ein.

Geben Sie ein:

```
|cpm  [ENTER]
```

(Das Symbol | ergibt sich durch Halten von [SHIFT] und Drücken der Taste e .)

Nach einigen Sekunden erscheint folgende Meldung auf dem Bildschirm:

```
CP/M 2.2 - Amstrad Consumer Electronics plc
A>
```

Dies ist eine Kennzeichnung für das CP/M-Betriebssystem.

Das auf dem Bildschirm dargestellt A> ist ein Prompt-Zeichen (ähnlich dem Ready bei BASIC) und besagt, daß der Computer Ihre

Befehle erwartet.

Wenn Sie mit CP/M arbeiten, können Sie keine BASIC-Befehle in den Computer eingeben, da diese nicht verstanden werden.

Geben Sie beispielsweise den BASIC-Befehl:

c l s [ENTER]

ein, so gibt der Computer Ihre Eingabe mit einem Fragezeichen zurück:

C L S ?

Der Computer versteht diesen Befehl nicht.

Um kurz einige der CP/M-Befehle zu untersuchen, geben Sie ein:

d i r [ENTER]

Auf dem Bildschirm erscheint nun das Inhaltsverzeichnis (directory) mit den Dienstprogrammen (utility COMands) von CP/M. Eines von ihnen ist format.

Geben Sie

f o r m a t [ENTER]

ein, auf dem Bildschirm erscheint:

```
Please insert disc to be formatted into drive A
then press any key.
```

Entfernen Sie die CP/M-Diskette und legen Sie die neue Leer-Diskette ein. Drücken Sie irgendeine Taste. Die Formatierung beginnt mit Spur 0 und endet mit Spur 39. Daraufhin erscheint auf dem Bildschirm:

```
Do you want to format another disc (Y/N):
```

Wenn Sie die andere Seite der Leer-Diskette oder eine weitere Diskette formatieren wollen, geben Sie Y (für YES) ein und die Anfangs-Meldung erscheint erneut.

Die Formatierung kann beliebig oft wiederholt werden, bis auf die Wiederholungs-Frage mit N (für No) geantwortet wird. Das System meldet sich dann mit:

```
Please insert a CP/M system disc into drive A
then press any key:
```

Daraufhin kehrt der Computer in den direkten CP/M-Modus zurück und erwartet neue Befehle. Weitere CP/M-Befehle werden später in diesem Handbuch behandelt.

Nachdem wir mit CP/M gelernt haben zu formatieren, setzen wir den Computer mit [C T R L] [S H I F T] [E S C] zurück.

Sie sollten stets eine Master-Copy Ihrer CP/M-Diskette an einem sicheren Ort aufbewahren, da diese der Schlüssel zu Ihrem System ist.

Im weiteren Verlauf dieses Handbuches wird erläutert, wie eine Arbeitskopie von der CP/M-Diskette angefertigt wird, so daß Sie in Ruhe Ihre Master-Kopie wegschließen können!

ACHTUNG!
Die Formatierung einer beschriebenen Diskette löscht ihren Inhalt.

Sie können keine Diskette mit geöffnetem Schreibschutz-Loch formatieren. Der Versuch wird mit der Meldung:

```
Drive A: disc is write protected
RETRY, Ignore or Cancel?
```

beantwortet.

Drücken Sie C für cancel und folgen Sie den Anweisungen auf dem Bildschirm. Der Formatierungs-Modus wird daraufhin verlassen.

Wenn wir nunmehr über eine formatierte Diskette verfügen, können wir damit beginnen, BASIC-Programme auf und von der Diskette zu übernehmen.

Sichern von Programmen auf Diskette

Ein in ein Computer-Speichersystem eingegebenes Programm kann durch Eingabe von

s a v e " D a t e i n a m e " [E N T E R]

auf die Diskette geschrieben werden.

Es ist zu beachten, daß die Namensgebung für die Programme vorgeschrieben ist.
Ein Dateiname besteht aus zwei Teilen (Feldern). Das erste Feld ist obligatorisch und kann bis zu 8 Zeichen enthalten. Hierbei

können Buchstaben und Zahlen, **nicht** aber Leerzeichen bzw. Satzzeichen verwendet werden. Das erste Feld enthält normalerweise den Namen des Programms.
Das zweite Feld ist wahlfrei. Sie können bis zu 3 Buchstaben einsetzen, jedoch auch hier weder Leer- bzw. Satzzeichen verwenden. Ein Punkt dienst zur Trennung der beiden Felder.

Wenn Sie kein zweites Feld spezifizieren wollen, erzeugt das System automatisch eine eigene Bezeichnung wie z.B. .BAS für BASIC-Dateien bzw. .BIN für Binärdateien (Maschinencode).

Als Beispiel zum Sichern einer Datei auf Diskette schreiben Sie bitte ein kurzes Programm in den Speicher, legen eine formatierte Diskette ein und geben folgendes ein:

s a v e " B e i s p i e l " [E N T E R]

Nach ein paar Sekunden erscheint R e a d y auf dem Bildschirm und das Programm ist auf Diskette gesichert. Falls nicht, überprüfen Sie bitte die Fehlermeldungen auf dem Bildschirm: ob die Diskette im korrekten Laufwerk steckt, das Schreibschutz-Loch geschlossen, oder der Befehl falsch eingegeben ist.

Katalogisierung

Nach Abspeicherung des oben genannten Programms, geben Sie ein:

c a t [E N T E R]

Auf dem Bildschirm erscheint:

D r i v e A : u s e r Ø

B e i s p i e l . B A S 1 K

1 6 8 K f r e e

Der Dateiname wird gemeinsam mit der Dateilänge (aufgerundet auf volle KByte) und dem spezifizierten bzw. standardmäßigen zweiten Feld dargestellt. Darüber hinaus wird der verbleibende freie Speicherplatz auf der Diskette angegeben.

Laden von Diskette

Programme können von der Diskette geladen und gestartet werden, indem die Befehle

```
Load "Dateiname"  [ENTER]
run  [ENTER]
```

eingegeben werden. Laden und Starten kann auch direkt mit dem Befehl

```
run "Dateiname"  [ENTER]
```

erfolgen.

Geschützte Programme können nur direkt gestartet werden.

|A und |B

Arbeiten Sie mit 2 Disketten-Laufwerken, so müssen Sie das Laufwerk (A oder B), auf das sich ein Befehl beziehen soll, bestimmen, indem Sie a oder b vor dem save-, cat-, load oder run-Befehl angeben.

Kopieren der Programme von Diskette auf Diskette

Unter Verwendung der bereits in diesem Abschnitt gelernten Befehle, können Programme von Diskette auf Diskette kopiert werden, indem das Programm von der Quellendiskette in den Speicher kopiert, die Quellendiskette entfernt und das Programm auf die Zieldiskette gesichert wird.

Um ein Programm bei einem System mit zwei Diskettenlaufwerken von einer Diskette auf die andere zu kopieren, können Sie Ihre Quellendiskette beispielsweise in Laufwerk B und Ihre Zieldiskette in Laufwerk A einlegen.

Zum Kopieren eines Programms führen Sie nun folgende Eingabe aus:

```
|b  [ENTER]
load "Dateiname"  [ENTER]
|a  [ENTER
save "Dateiname"  [ENTER]
```

Zur Abspeicherung von Dateien beim CPC664 sind vier Möglichkeiten gegeben: Zusätzlich zur normalen BASIC-Datei-Abspeicherung durch

s a v e "D a t e i n a m e" [E N T E R]

weitere drei alternative Verfahren für spezielle Zwecke.

s a v e "D a t e i n a m e", a [E N T E R]

Durch Hinzunahme des Suffix , a speichert der Computer die Datei nicht in der sonst üblichen Maschinensprache ab, sondern als ASCII-Zeichen. Diese Art Daten zu speichern wird normalerweise in der Textverarbeitung und bei anderen Anwenderprogrammen benützt. Eine detaillierte Beschreibung erfolgt, wenn diese Programme erklärt werden.

s a v e "D a t e i n a m e", p [E N T E R]

Wenn das Suffix p beim Abspeichern hinzugefügt wird, wird das Programm in codierter Form gespeichert. Wenn Sie das Programm mit RUN ablaufen lassen, können Sie es nicht auflisten, wenn es vorher mit E S C unterbrochen wurde.

Die in dieser Form abgespeicherten Programme können nur noch direkt verarbeitet werden und zwar unter Verwendung der Befehle

r u n "D a t e i n a m e" [E N T E R]

bzw.

c h a i n "D a t e i n a m e" [E N T E R]

Wenn Sie das Programm korrigieren bzw. verändern wollen, so sollten Sie in diesem Fall eine ungeschützte Version des Programms in Reserve haben.

save "Dateiname",b, Startadresse , Länge ,
Anfangspunkt [ENTER]

Diese Option ermöglicht Ihnen eine Binärabspeicherung eines kompletten Datenblocks im RAM-Speicher. Hierbei ist es erforderlich, dem Computer mitzuteilen, wo der zu sichernde Speicherbereich beginnt, dessen Länge in Byte und erforderlichenfalls die Speicheradresse, zu der bei Ausführung die Datei als Programm zu übertragen ist.

Diese binäre Sicherung ermöglicht eine Übertragung vom Bildschirmspeicher direkt auf Diskette in Form eines Bildschirm-

Speicherauszuges. Die Inhalte des Bildschirms werden hierbei genau wie dargestellt abgespeichert und zwar unter Einsatz der folgenden Steuerung:

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

wobei 49152 die Startadresse des Bildschirmspeichers und 16384 die Länge des Bildschirmspeichers darstellt, die Sie abspeichern möchten.

Für die erneute Darstellung auf dem Bildschirm, geben Sie ein:

```
Load "scrndump"    [ENTER]
```

Im weiteren Verlauf dieses Handbuches werden Sie noch mehr Informationen über die Arbeit mit Disketten- (bzw. Kassetten-) Dateien erhalten.

Prüfen Sie abschließend die Einhaltung der Hinweise am Anfang dieses Handbuches mit dem Titel **"WICHTIGE HINWEISE".**

Teil 8: DARSTELLUNGSARTEN; FARBEN UND GRAPHIK

Auf dem Bildschirm des CPC664 Colour Personal Computer sind drei verschiedene Darstellungarten (Modi) möglich: Modus 0, Modus 1 und Modus 2.

Nach dem Einschalten des Computers ist automatisch Modus 1 eingestellt.

Um die einzelnen Darstellungarten zu verdeutlichen, drücken Sie auf die Taste mit der Ziffer 1. Halten Sie die Taste fest, bis zwei Zeilen mit 1-ern voll sind. Wenn Sie jetzt zählen, sehen Sie, daß auf jeder Zeile 40 Ziffern sind. Das heißt, daß im Modus 1 vierzig Spalten zur Verfügung stehen. Drücken Sie jetzt auf [ENTER] - Sie bekommen zwar einen Syntaxfehler, aber der stört uns nicht. Wir kommen nur schneller auf Ready zurück, so daß wir die nächste Anweisung eingeben können.

Tippen Sie: mode Ø [ENTER]

Sie sehen, daß die Zeichen auf dem Bildschirm jetzt größer sind. Drücken Sie wieder auf die 1-er Taste bis zwei Zeilen vollgeschrieben sind. Jetzt sind nur noch 20 Ziffern in einer Zeile. Das heißt, daß wir in Mode 0 nur 20 Spalten zur Verfügung haben. Drücken Sie erneut auf [ENTER]

Tippen Sie jetzt: mode 2 [ENTER]

Dies ist der kleinste Modus, und wenn Sie jetzt eine Reihe von 1-ern eingeben, dann sehen Sie 80 Ziffern. In Mode 2 haben wir 80 Spalten. Nochmal:

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

Drücken Sie jetzt noch einmal auf [ENTER]

FARBEN

Wir können unter 27 Farben auswählen. Diese erscheinen auf einem grünen Bildschirm (GT65) als unterschiedlich helle Grünstufen. Wenn Sie den Monitor GT65 besitzen, können Sie einen Schneider MP2 Modulator/Stromversorgung dazukaufen um die Farbmöglichkeiten Ihres Computers auf Ihrem eigenen Fernsehgerät

einzusetzen.

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

Sie können die Farben des Bildrandes (BORDER), des Papiers (PAPER) - die Fläche, auf der die Zeichen stehen - oder des Schreibstiftes (PEN) - das Zeichen selbst - unabhängig voneinander verändern.

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

Zur einfacheren Übersicht erscheint diese Tabelle auch auf der Diskettenabdeckung auf der rechten Seite des Computers.

FARBTABELLE

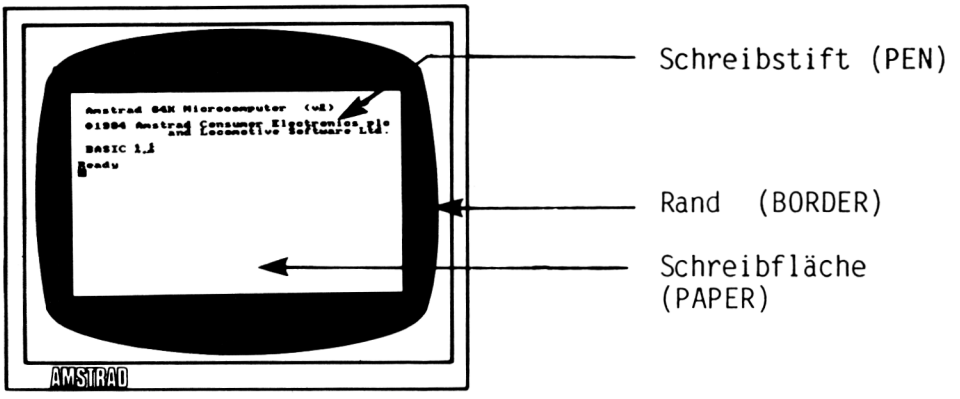
Tinte (INK)	Farbe	Tinte (INK)	Farbe
0	Schwarz	14	Pastellblau
1	Blau	15	Orange
2	Hellblau	16	Rosa
3	Rot	17	Pastellmagenta
4	Magenta	18	Hellgrün
5	Hellviolett	19	Seegrün
6	Hellrot	20	helles Blaugrün
7	Purpur	21	Limonengrün
8	helles Magenta	22	Pastellgrün
9	Grün	23	Pastellblaugrün
10	Blaugrün	24	Hellgelb
11	Himmelblau	25	Pastellgelb
12	Gelb	26	Leuchtendweiß
13	Weiß		

Tabelle 1: Die Tinten-Nummern und Farben

Wie vorher erklärt, befindert sich der Computer nach dem Einschalten in Modus 1. Um auf Modus 1, von einer anderen Betriebsart aus zurückzukommen, geben Sie ein:

m o d e 1 [E N T E R]

Die Anzeige-Schreibfläche



Der Rand (BORDER) ist die Zone um die eigentliche Schreibfläche (PAPER). (Nach dem Einschalten des Computers sind sowohl RAND als auch PAPIER blau.) PAPIER (PAPER) ist die eigentliche Schreibfläche innerhalb des Randes (BORDER), in der die Zeichen stehen können, während PEN die Farbe des Zeichens ist.

Im Anschluß möchten wir Ihnen erklären, wie die Farben, die Sie auf dem Bildschirm sehen, ausgewählt werden und wie diese nach Ihrer Wahl wechseln können.

Sie schalten zuerst den Computer ein bzw. setzen diesen zurück, wobei BORDER auf Farbe Nr. 1 gesetzt ist. Suchen Sie Nummer 1 auf der Tabelle 1 auf und Sie werden feststellen, daß die Farbnummer 1 der Farbe blau entspricht. Die Farbe des Randes läßt sich unter Zuhilfenahme folgender Steuerung ändern:
BORDER, mit anschließender Farbnummer. Für den Wechsel von BORDER auf weiß ist einzugeben:

b o r d e r 1 3 [E N T E R]

Soweit, so gut. Nun aber zu den Feinheiten ...

Beim ersten Einschalten bzw. Zurückstellen des Computers werden PAPER Ø und PEN 1 stets automatisch gewählt. Hiermit ist nicht gesagt, daß Sie die Nummern 0 und 1 nur auf der Tabelle 1 suchen brauchen und sich ansonsten um nicht kümmern müssen ...

Wichtig hierbei ist, sich daran zu erinnern, daß Ø und 1 Nummern für PAPER und PEN sind; es handelt sich hierbei nicht um Farbnummern. Zum Verständnis gehen wir davon aus, daß wir über vier Schreibstifte an Ihrem Schreibtisch mit den Nummern 0, 1, 2 und 3 verfügen und daß wir in der Lage sind, jeden dieser Schreibstifte mit einer beliebigen Farbe Ihrer Wahl aus 27 Tintenflaschen, numeriert zwischen 0 und 26, zu füllen.

Hieraus ist auch ersichtlich, daß PEN 1 nicht unbedingt immer die gleiche Farbe ausdrücken muß, da diese mit einer anderen Farbe gefüllt werden kann, so wie Sie auch jeden der 4 Schreibstifte mit der gleichen Farbe füllen könnten.

STANDARDEINSTELLUNGEN

	Tintenfarbe		
Paper/Pen	Mode 0	Mode 1	Mode 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 Verweistabelle

In gleicher Weise verhält es sich mit dem Computer. Unter Einsatz der Befehle PEN und INK haben Sie die Möglichkeit, die Schreibstift-Nummer und anschließend die Farbnummer für diesen Schreibstift zu wählen.

Wir gehen davon aus, daß wir im Modus 1 (40 Spalten) arbeiten, mit Blick auf die vorstehende Tabelle 2, wobei aus der ersten und der dritten Spalte hervorgeht, daß die Schreibstift Nummer 1 der Farbnummer 24 entspricht. Wenn Sie nunmehr die Farbnummer 24 auf der Farbtabelle (Tabelle 1) aufsuchen, werden Sie feststellen, daß es sich bei der aufgeführten Farbe um Hellgelb handelt, d.h. die Farbe der Zeichen auf dem Bildschirm, nachdem Sie den Computer eingeschaltet haben.

Demgegenüber sind jedoch die Beziehungen zwischen PAPER/PEN/INK gemäß Tabelle 2 nicht feststehend. Es handelt sich hierbei um Standardeinstellungen bei der Ersteinschaltung bzw. Rückschaltung des Computers. Sie können diese unter Einsatz des INK-Befehles ändern. Dieser Befehl umfaßt zwei Teile (bzw. "Parameter"). Der erste Teil betrifft die Nummer von PAPER bzw. PEN, d.h. den Schreibstift, den Sie gerade mit Farbe füllen, wobei der zweite Teil die Farbe der Füllung selbst betrifft. Beide Teile der Steuerungen werden durch ein Komma getrennt.

Da uns nunmehr bekannt ist, daß wir PEN 1 einsetzen, wollen wir die Farbe dieses Schreibstiftes zu Orange wechseln. Geben Sie nunmehr ein:

```
ink 1,15  [ENTER]
```

... und Sie werden feststellen, daß die Zeichen am Bildschirm ihre Farbe geändert haben.

Mit dem INK-Befehl läßt sich darüber hinaus auch die Farbe des Hintergrundes verändern. Wir wissen, daß beim Einschalten die PAPER-Nummer Ø gewählt ist; wechseln wir nunmehr die Farbe der PAPER-Nummer Ø auf grün (Farbnummer 9) durch Eingabe folgenden Befehls:

```
ink Ø,9  [ENTER]
```

Lassen Sie uns nunmehr auch PEN wechseln. Geben Sie ein:

```
pen 3  [ENTER]
```

Vergegenwärtigen Sie sich, daß sich nur die Farbe der neuen Zeichen (nach der Steuerung) ändert. Sie arbeiten nunmehr mit Schreibstift Nummer 3 und Sie können aus den Tabellen 1 und 2 entnehmen, daß die ursprünglich für Nr. 3 verwendete Farbe Nummer 6 (hellrot) ist. Wechseln Sie PEN auf rose und zwar durch Eingabe von:

```
ink 3,16  [ENTER]
```

Erinnern Sie sich daran, 3 war die Farbe des Schreibstiftes, den Sie vorher mit dem Befehl: PEN 3 wählten und 16 ist die Farbe - rosa.

Lassen Sie uns numehr PAPER wechseln. Bei Wahl eines neuen PAPER ändert sich die vorherige Hintergrundfarbe hinter den Zeichen **nicht**, da diese Farbe durch ein anderes PAPER "gedruckt" wurde.

Um diesen Vorgang zu verfolgen, geben Sie ein:

```
paper 2  [ENTER]
```

Prüfen Sie erneut in den Tabellen 1 und 2, warum die Hintergrundfarbe für PAPER-Nummer 2 ein helles Blaugrün ist. Ändern Sie auf schwarz, indem Sie folgende Eingabe vornehmen:

```
ink 2,Ø  [ENTER]
```

Auf dem Bildschirm haben wir nunmehr Zeichen, die mit den Schreibstift Nummern 1 und 3 auf einem Hintergrund Ø und 2 geschrieben wurden. Die INKs lassen sich für PEN bzw. PAPER ändern, das Sie gerade verwenden. Geben Sie beispielsweise ein:

```
ink 1,2  [ENTER]
```

...wodurch sich die Farbe aller vorhergehenden Zeichen ändert, die mit PEN-Nummer 1 geschrieben wurden.

Geben Sie ein:

```
cls  [ENTER]
```

... um den Bildschirm zu löschen.

Es müßte nunmehr für Sie möglich sein, den Computer anzuweisen, auf seine ursprünglichen Farben zurückzukehren (blauer Rand und Hintergrund mit hellgelben Zeichen) und zwar unter Verwendung der Befehle BORDER, PAPER, PEN und INK. Versuchen Sie es. Falls

es Ihnen nicht möglich sein sollte, setzen Sie den Computer mit [CTRL] [SHIFT] und [ESC] zurück.

BLINKENDE FARBEN

Wir können die Zeichen in zwei verschiedenen Farben abwechselnd blinken lassen. Diese Funktion erreichen wir, indem wir eine zweite Farb-Nummer im INK-Befehl für PEN angeben.

Sie sehen die Zeichen am Bildschirm abwechselnd leuchtend weiß und hellrot blinken, wenn Sie folgenden Befehl eingeben:

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

In diesem Falle ist 1 die PEN-Nummer, während 26 leuchtend-weiß und 6 die Alternativfarbe hellrot bedeutet.

Wir können auch die PAPER-Farbe hinter den Zeichen mit zwei Farben blinken lassen. Dies erfolgt durch Hinzunahme einer weiteren Farbnummer für den INK-Befehl für PAPER.

Um festzustellen, wie PAPER zwischen grün und hellgelb hinter den Zeichen blinkt, geben Sie ein:

```
ink Ø, 9, 24 [ENTER]
```

Hier ist die Ø die PAPER-Nummer, 9 ist die Farbe grün und 24 ist die Alternativfarbe hellgelb.

Setzen Sie jetzt den Computer zurück mit [CTRL], [SHIFT] und [ESC].

Beachten Sie, daß im Modus 0 für PEN 14 und 15 bzw. PAPER 14 und 15 schon zwei blinkende Farben vorgesehen sind. Hierzu ist es nicht erforderlich, eine extra Ziffer im INK-Kommando anzugeben.

Tippen Sie:

```
mode Ø [ENTER]
pen 15 [ENTER]
```

Am Bildschirm blinkt nun das Wort ready zwischen himmelblau und rosa. Tippen Sie jetzt:

```
paper 14 [ENTER]
cls      [ENTER]
```

Sie werden sehen, daß nicht nur das Wort r e a d y zwischen himmelblau und rosa blinkt, sondern auch die Hintergrundfarbe (PAPER) zwischen gelb und blau.

Die PEN- und PAPER-Nummern 14 und 15 können erneut programmiert werden und zwar unter Verwendung des INK-Kommandos für andere Blinkfarben bzw. für eine ständig erscheinende Farbe.

Abschließend ist es möglich, den Rand (BORDER) zwischen zwei Farben blinken zu lassen, indem eine weitere Farbnummer dem BORDER-Befehl zugeordnet wird. Geben Sie ein:

b o r d e r 6 , 9 [E N T E R]

Sie werden feststellen, daß der Rand (BORDER) nunmehr zwischen hellrot und grün blinkt. Hierbei ist zu beachten, daß der Rand auf irgendeine bzw. auf eine Kombination von 27 Farben eingestellt werden kann und zwar unabhängig davon, ob Sie in Modus 0, 1 bzw. 2 arbeiten.

Setzen Sie nunmehr den Computer zurück mit [C T R L] [S H I F T] und [E S C]

Zur weiteren Demonstration der verfügbaren Farben können Sie folgendes Programm eingeben und starten:

```
10 MODE 0
20 zeit=600: REM Ablaufgeschwindigkeit
30 FOR b=0 TO 26
40 LOCATE 3,12
50 BORDER b
60 PRINT "Farbe Border";b
70 FOR t=1 TO zeit
80 NEXT t,b
90 CLG
100 FOR p=0 TO 15
110 PAPER p
120 PRINT "Paper";p
130 FOR n=0 TO 15
140 PEN n
150 PRINT "Pen";n
160 NEXT n
170 FOR t=1 TO zeit*2
180 NEXT t,p
190 MODE 1
200 BORDER 1
210 PAPER 0
220 PEN 1
230 INK 0,1
240 INK 1,24
```

WICHTIGER HINWEIS

Innerhalb des obigen Programms sowie in weiteren Kapiteln und Aufstellen innerhalb dieses Handbuches erscheinen die BASIC-Schlüsselworte in GROSSBUCHSTABEN.
In dieser Form erscheinen BASIC-Schlüsselwörter dann, wenn ein Programm vom Computer gelistet wird. Im allgemeinen ist es vorzuziehen, Befehle bzw. Programme in Kleinbuchstaben zu schreiben, da es Ihnen bei Fehlern hilft, wenn das Programm gelistet wird - (aus dem Grunde, weil fehlerhafte BASIC-Schlüsselworte **nicht** in Großbuchstaben übernommen werden.

Für den verbleibenden Teil des Grundlagenkurses listen wir Programme sowohl in Groß- als auch in Kleinbuchstaben, so daß Sie mit dieser Betriebsart vertraut werden.

Eine Bezeichnung einer Variablen wie z.B. x bzw. a$, wird **nicht** in Großbuchstaben umgesetzt, wenn das Programm gelistet wird, da hierbei der Computer in jedem Falle den Namen erkennt, unabhängig davon, ob diese Variable innerhalb des Programms in Groß- bzw. Kleinbuchstaben erscheint.

ACHTUNG

Ab dieser Seite des Handbuches werden Sie nicht mehr darauf hingewiesen, die ENTER -Taste nach jeder Zeile zu drücken; es wird davon ausgegangen, daß Sie dies automatisch vornehmen.

GRAPHIK

Der Computer hat eine ganze Reihe von Symbolen intern gespeichert. Zur Ausgabe dieser Symbole können wir folgendes Schlüsselwort einsetzen:

```
chr$ ( )
```

In den Klammern sollte die Nummer des Symbols, eine Zahl zwischen 32 und 255 stehen.

Setzen Sie den Computer mit [CTRL] [SHIFT] und [ESC] zurück und tippen Sie dann:

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

Vergessen Sie nicht [ENTER] zu drücken. Am Bildschirm sehen Sie Symbol 250, einen Mann, der nach rechts läuft.

Das folgende Programm können Sie benutzen, um alle vorhandenen Symbole mit ihren zugehörigen Nummern auszugeben. Tippen Sie das Programm ein, vergessen Sie dabei aber nicht, nach jeder Zeile die ENTER -Taste zu drücken.

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

Im Kapitel 7 "Zu Ihrer Information..." finden Sie alle Symbole mit ihren zugehörigen Referenznummern.

LOCATE

Dieses Kommando bringt den Cursor an eine gewünschte Stelle auf dem Bildschirm. Ohne LOCATE-Kommando fängt der Cursor in der obersten, linken Ecke des Bildschirms an. Diese hat die x-, y-Koordinaten 1,1 (x ist die Waagrechte und y ist die senkrechte Position). Im Modus 1 stehen 40 Spalten und 25 Zeilen zur Verfügung. Die Cursor-Position in der Mitte der obersten Zeile hat in Modus 1 ide x-, y-Koordinaten 20,1.

Um dies zu sehen, tippen Sie (denken Sie an [ENTER]):

```
mode 1                    ...der Bildschirm wird gelöscht,
                          der Cursor oben links

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

Um sich zu überzeugen, daß dies die oberste Zeile ist, schreiben Sie:

```
border 0
```

Der Rand (BORDER) ist jetzt schwarz und der kleine Mann ist in der Mitte der obersten Bildschirmzeile zu sehen.

In Modus 0 stehen nur 20 Spalten, aber noch die gleichen 25 Zeilen zur Verfügung. Tippen Sie:

```
mode 0
run
```

...der kleine Mann steht jetzt in der rechten oberen Ecke des Bildschirms, weil die X-Koordinate 20 die letzte Spalte im Modus 0 ist.

In Modus 2 gibt es 80 Spalten und 25 Zeilen. Wir benutzen das gleiche Programm. Sie können sicherlich erraten, wo der kleine Mann jetzt steht. Geben Sie ein:

```
mode 2
run
```

Rückkehr in Modus 1 durch Eingabe von:

```
mode 1
```

Probieren Sie jetzt selbst ein wenig, indem Sie die Nummern für LOCATE und chr$ ( ) verändern und damit verschiedene Symbole überall auf dem Bildschirm plazieren können. Tippen Sie z.B.:

```
locate 2Ø, 12:print chr$(24Ø)
```

Sie sehen einen Pfeil in der Bildschirmmitte. In diesem Befehl ist

```
 2Ø die waagrechte (X)-Koordinate (Bereich 1-40)
 12 die senkrechte (Y)-Koordinate (Bereich 1-25)
24Ø die Nummer des gezeigten Symbols (Bereich 32-255)
```

Das Symbol können Sie mit dem folgenden Programm wiederholt über den ganzen Bildschirm ausgeben. Tippen Sie:

```
1Ø  CLS
2Ø  FOR x=1 TO 39
3Ø  LOCATE x,2Ø
5Ø  PRINT CHR$(25Ø)
6Ø  NEXT x
7Ø  GOTO 1Ø
run
```

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

Wenn das vorherige Zeichen am Bildschirm gelöscht werden soll, bevor das nächste ausgegeben wird, tippen Sie:

```
5Ø print " "; chr$(25Ø)
```

(Diese neue Zeile 5Ø überschreibt automatisch die bisherige Zeile 5Ø).

Tippen Sie nunmehr ein:

```
run
```

FRAME

Um die Bewegung des Zeichens quer über den Bildschirm zu verbessern, fügen Sie folgende Zeile hinzu:

```
40 frame
```

Die FRAME-Steuerung synchronisiert die Bewegung von Objekten auf dem Bildschirm mit der Abtastfrequenz des Bildschirms. Sollte Ihnen diese Erklärung etwas zu technisch vorkommen, denken Sie daran, daß diese Steuerung immer dann einzusetzen ist, wenn Sie Zeichen bzw. Graphiken langsam auf dem Bildschirm wandern lassen wollen.

Dieses Programm kann weiterhin dazu verwendet werden, die Bewegung durch Hinzunahme einiger Verzögerungsschleifen zu verbessern sowie durch Einsatz verschiedener Rückkehrsymbole. Tippen Sie ein:

```
list
```

Ergänzen Sie nunmehr das Programm mit den folgenden Zeilen:

```
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-Kommando benutzen wir das PLOT-Kommando um den Graphik-Cursor zu positionieren, in diesem Falle mit Pixel-Koordination.
(Als Pixel wird das kleinstmögliche Segment des Bildschirms bezeichnet.)

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

Wir haben 640 waagrechte Pixel und 400 senkrechte Pixel.

Die x-, y-Koordinaten beziehen sich auf die unterste, linke Ecke und fangen mit 0,0 an. Im Gegensatz zu der für Zeichen verwendeten LOCATE-Steuerung bleiben die Koordinaten im Modus 0,1 bzw. 2 immer gleich.

Um dies zu prüfen, setzen Sie den Computer mit [C T R L], [S H I F T] und [E S C] zurück und tippen Sie dann:

```
plot 320, 200
```

Ein kleiner Punkt erscheint in der Bildmitte.

Jetzt ändern Sie die Darstellungart mode mit:

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

Sie werden feststellen, daß sich der Punkt noch immer in der Mitte befindet, jetzt aber größer ist. Ändern Sie noch einmal den Modus und geben Sie ein:

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

Auch hier ist der Punkt noch immer in der Mitte, in diesem Falle jedoch wesentlich kleiner.

Versuchen Sie jetzt einige Punkte auf den Bildschirm zu bringen und zwar an verschiedenen Stellen und in verschiedenen Größen, um sich selbst mit dieser Steuerung vertraut zu machen. Anschließend löschen Sie den Bildschirm und kehren in den Modus 1 zurück und zwar unter Eingabe von:

```
mode 1
```

DRAW

Setzen Sie zuerst den Computer zurück [C T R L], [S H I F T] und E S C . Das DRAW-Kommando zeichnet eine Linie, ausgehend von der aktuellen Position des Graphikzeigers. Zeichnen Sie ein Rechteck am Bildschirm und zwar mit folgendem Programm.

Wir setzen zhächst den Graphikzeiger mit dem PLOT-Kommando an die gewünschte Stelle. Dann zeichnen wir eine Linie ab dieser Stelle nach oben in Richtung der obersten, linken Ecke, von hier aus zeichnen wir eine Linie zur rechten Ecke, usw., usw.

Anschließend geben Sie ein:

```
5  c l s
1Ø  p l o t  1Ø , 1Ø
2Ø  d r a w  1Ø , 3 9Ø
3Ø  d r a w  6 3Ø , 3 9Ø
4Ø  d r a w  6 3Ø , 1Ø
5Ø  d r a w  1Ø , 1Ø
6Ø  g o t o  6Ø
r u n
```

Drücken Sie jetzt zweimal [E S C] um das Programm abzubrechen.

(Beachten Sie Zeile 6Ø dieses Programms; der Computer erhielt die Anweisung, ständig um Linie 6Ø zu kreisen, bis Sie vom Programm aus durch doppeltes Drücken von E S C "abbrechen". Diese Anweisung ist dann hilfreich, wenn Sie es vermeiden wollen, daß der Computer automatisch am Ende des Programms die r e a d y -Anzeige auf dem Bildschirm bringt.)

Fügen Sie jetzt die folgenden Zeilen in das Programm ein, damit ein zweites Rechteck innerhalb des ersten dargestellt wird. Folgende Eingabe:

```
6Ø  p l o t  2Ø , 2Ø
7Ø  d r a w  2Ø , 3 8Ø
8Ø  d r a w  6 2Ø , 3 8Ø
9Ø  d r a w  6 2Ø , 2Ø
1ØØ  d r a w  2Ø , 2Ø
1 1Ø  g o t o  1 1Ø
r u n
```

Betätigen Sie erneut zweimal [E S C] , um das Programm abzubrechen.

MOVE

Das MOVE-Kommando arbeitet vergleichbar zu PLOT und zwar insofern, daß der Graphik-Cursor zu dem durch die y-, x-Koordinaten genannten Punkt bewegt wird; demgegenüber wird jedoch die Punktstellung (dot) am neuen Graphik-Cursor-Standort **nicht** auf dem Bildschirm sichtbar.

Tippen Sie ein:

```
c l s
m o v e  6 3 9 ,  3 9 9
```

Obwohl auf dem Bildschirm keine Wahrnehmung zu treffen ist, haben Sie den Graphik-Cursor in die obere rechte Ecke bewegt.

Prüfen Sie diese Bewegung durch Zeichnen einer Linie von dieser Stellung zum Mittelpunkt des Bildschirms und zwar durch Eingabe von:

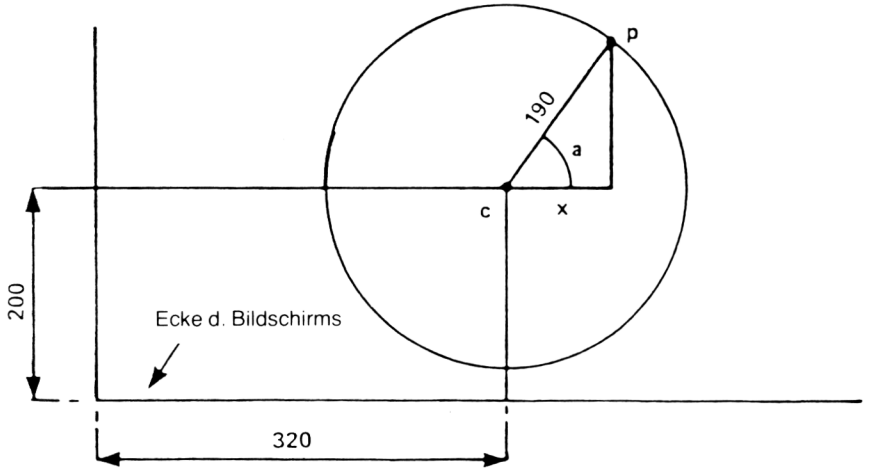
```
d r a w  3 2 Ø ,  2 Ø Ø
```

KREISE

Kreise können entweder geplottet oder gezeichnet werden. Man kann einen Kreis bilden, wenn man die x, y-Koordinaten jedes Punktes auf dem Kreisumfang plottet.

Schauen Sie sich das Bild unten an und Sie entdecken, daß der Punkt p auf dem Kreisumfang mit dem x- und y-Koordinaten geschrieben werden kann. Diese sind:

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



WIE MAN DIE PUNKTE EINES KREISES PLOTTET

In vorigen Programmen haben wir alle Punkte, bezogen auf die linke untere Ecke des Bildschirms geplottet. Wir wollen den Kreis in der Bildschirmmitte haben, also um die Koordinaten 320,200, so daß wir diese Koordinaten zu unseren errechneten Punkten addieren müssen.

Ein Programm für den Kreis könnte wie folgt aussehen:
Geben Sie demnach ein:

```
new
10 cls
20 deg
30 for a=1 to 360
40 move 320,200
50 draw 320+190*COS (a), 200 + 190*SIN(a)
60 next
run
```

Wir haben bereits das Schlüsselwort NEW benutzt, bevor wir das eigentliche Programm eintippten. NEW sagt dem Computer, daß der Speicher zu löschen ist, ungefähr so, als wir [CTRL], [SHIFT] und [ESC] drücken würden. Der Bildschirm wird jedoch nicht gelöscht, sondern nur der interne Speicher. Es ist manchmal nützlich das alte Programm vor Augen zu haben, während wir das neue Programm schreiben.

Der Radius des Kreises kann reduziert werden, wenn wir die Zahl 190 verkleinern (190 bezieht sich auf die Pixel).

Um die Auswirkung des anders geplotteten Kreises (in Radianen) festzustellen, löschen Sie Zeile 20 aus dem Programm durch Eingabe von:

```
20
```

Ein geschlossener Kreis wird gezeichnet, wenn wir das Schlüsselwort PLOT durch DRAW ersetzen. Zeile 50 lautet somit:

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

Probieren Sie es auch, mit und ohne Zeile 20.

Sie haben in Zeile 60 dieses Programms festgestellt, daß wir NEXT statt NEXT a schrieben. Es ist erlaubt, auf "NEXT" zu ver-

einfachen; der Computer ermittelt die richtige FOR-Anweisung zum angegebenen "NEXT". Für Programme, in denen viele FOR und NEXT-Schleifen vorkommen, ist es übersichtlicher, wenn Sie den Namen der Variablen in der NEXT-Anweisung ausgeben.

ORIGIN

Im vorigen Programm haben wir das Kommando MOVE benutzt, um den Kreismittelpunkt zu zeichnen, und haben dann die x-, y-Koordinaten zu dieser Position addiert. Wir können darauf verzichten, wenn wir das ORIGIN-Kommando anwenden. Hiermit bestimmen Sie die Mitte des Kreises, so daß wir nur alle Punkte an der Peripherie des Kreises bestimmen müssen.

```
new
10 cls
20 for a=1 to 360
30 origin 320,200
40 plot 190*cos(a), 190*sin(a)
50 next
run
```

Um vier kleinere Kreise zu zeichnen, geben Sie folgendes Programm ein:

```
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
```

Um auf andere Art einen Kreis auf dem Bildschirm dazustellen geben Sie folgendes Programm ein:

```
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
```

In diesem Falle wird von Koordinate zu Koordinate eine Linie um den Umfang eines Kreises gezogen; Sie werden hierbei feststellen, daß der Kreis viel schneller gezeichnet als geplottet ist.

Verfolgen Sie nochmals die Wirkung durch Verzicht auf die DEG-Steuerung, durch Entfallen der Zeile 30, mit anschließendem neuen Start des Programms (RUN).

FILL

Das FILL-Kommando wird zum Ausfüllen eines Bildschirmbereiches eingesetzt, der entweder durch gezeichnete Umrandungen oder von Bildschirm/Graphik-Fenstergrenzen abgeschlossen wird. Neueinstellung des Computers [CTRL],[SHIFT] und [ESC], anschließend Eingabe von:

```
10  cls
20  move  20,20
30  draw  620,20
40  draw  310,380
50  draw  20,20
run
```

Auf dem Bildschirm werden Sie ein Dreieck feststellen. Bewegen Sie den Graphik-Cursor zum Mittelpunkt des Bildschirms durch Eingabe von:

```
move  320,  200
```

Geben Sie den BASIC-Befehl FILL ein und anschließend eine Schreibstift-Nummer, beispielsweise 3; Sie können nunmehr den Bildschirm unter Verwendung des angegebenen Schreibstiftes füllen und zwar ausgehend von der augenblicklichen Stellung des Graphik-Cursors (Bildschirmmitte) zu den gezogenen Grenzen. Eingabe von:

```
fill 3
```

Anschließend bewegen wir den Graphik-Cursor außerhalb des Dreiecks durch Eingabe von:

```
move Ø,Ø
```

Prüfen Sie die Auswirkung nach folgender Eingabe:

```
fill 2
```

Der Computer hat Schreibstift-Nummer 2 gewählt, um die Fläche auszufüllen, die durch die gezogenen Linien und durch die Kanten des Bildschirms begrenzt wird.

Verändern Sie nunmehr das Programm durch Eingabe folgender Linien und prüfen Sie die Auswirkungen:

```
5Ø draw 5Ø,5Ø
6Ø move 32Ø,2ØØ
7Ø fill 3
run
```

Sie werden feststellen, daß an den Zwischenräumen der gezogenen Grenzen die Farbe vom Schreibstift 'durchdringt'.
Dieser Sachverhalt wird darüber hinaus durch AUFFÜLLEN eines geplotteten Kreises und anschließend eines gezogenen Kreises demonstiert. Tippen Sie ein:

```
new
1Ø CLS
2Ø FOR a=1 TO 36Ø
3Ø ORIGIN 32Ø,2ØØ
4Ø PLOT 19Ø*COS(a), 19Ø*SIN(a)
5Ø NEXT
6Ø MOVE - 188,Ø
7Ø FILL 3
run
```

Versuchen Sie nunmehr:

```
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
```

Sie können nunmehr die Umrisse des Kreises unsichtbar ausfüllen, indem Sie die Farbe des Schreibstiftes an die Farbe des Papiers anpassen. Fügen Sie hinzu:

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

Mit dem Befehl GRAPHICS PEN wird der Schreibstift ausgewählt, der zur Zeichnung der Graphik auf dem Bildschirm verwendet wird. Anschließend gibt die INK-Steuerung die Farbe für den Schreibstift an, bei dem es sich in diesem Fall um die gleiche wie die des Papiers handelt (d.h. Papier Nummer 1).

WEITERE EINZELHEITEN ...

Um die graphischen Darstellungen des CPC664 näher zu erläutern, siehe Abschnitt "Erläuterung der graphischen Symbole" des Abschnitts mit dem Titel "Wenn Sie gerade Zeit haben"...

Um diesen Abschnitt abzuschließen, werden an dieser Stelle einige Graphikprogramme angeboten, die eine Vielzahl von Programmsteuerungen und Befehle enthalten, mit deren Umgang Sie sich vertraut machen sollten. Jedes dieser Programme erzeugt fortlaufende Schemen auf der Bildschirmfläche.

```
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
```

```
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
```

```
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
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: SOUND

Klangeffekte werden über einen internen Lautsprecher ausgegeben. Wenn Sie mit einem MP2 Modulator/Stromversorgung und Ihrem Fernseher arbeiten, sollten Sie die Lautstärke des Fernsehgerätes ganz zurücknehmen.

Die Lautstärke am Computer kann mit dem VOLUME-Regler auf der rechten Seite Ihres Computers eingestellt werden. Sie können den Ton auch auf den auxiliary-Eingang Ihrer Stereoanlage geben, indem Sie die mit Stereo bezeichnete Buchse verwenden. Sie können dann den vom Computer erzeugten Geräuschepegel in STEREO über Ihre Hi-Fi-Anlage oder über den Kopfhörer empfangen. Hinweise bezüglich des Anschlusses an die STEREO-Buchse des Computers sind in Teil 2 dieses Grundlagenkurses enthalten.

Die Sound-Steuerung

Das SOUND-Kommando hat sieben Parameter; die ersten zwei müssen Sie angeben, die restlichen sind wahlweise. Das Kommando sieht folgendermaßen aus:

SOUND Kanalstatus, Tonperiode, Dauer, Lautstärke, Lautstärken-Hüllkurve, Ton-Hüllkurve, Geräuschperiode.

Das Ganze macht einen recht komplizierten Eindruck, analysieren wir doch jedes Parameter, so werden Sie schnell hiermit vertraut. Lassen Sie uns nacheinander mit den einzelnen Parametern beginnen und Sie werden sehr schnell verstehen, wie man mit diesem Befehl umgeht.

Kanalstatus

Um im Augenblick noch die Angelegenheit einfach zu halten, betrachten wir den "Kanalstatus" als Referenznummer für den Tonkanal. Wir unterscheiden zwischen 3 Tonkanälen und beginnen mit dem Tonstatus Nr. 1.

Tonperiode

Die Tonperiode bezeichnet die technische Art der Definition der Tonhöhe bzw. in anderen Worten, die Festlegung, um welche Note es sich handelt (c d e f g , usw.). Jeder Note ist eine bestimmte Zahl zugeordnet, und diese Zahl bezeichnet die Tonperiode.

Siehe hierzu in Kapitel 7 "Zu Ihrer Information" und Sie werden feststellen, daß die Note mittleres c die Tonperiode 478 hat.

Anschließend setzen wir den Computer mit [CTRL] , [SHIFT] und [ESC] zurück und geben ein:

```
10 sound 1,478
run
```

...und Sie hören anschließend, wie die Note c 0,2 Sekunden lang ertönt.
Sollten Sie keinerlei Tonwahrnehmung machen, stellen Sie sicher, daß sich der Lautstärkeknopf am Computer nicht auf Null befindet, anschließend geben Sie erneut RUN ein.

Dauer

Dieses Parameter bestimmt die Länge des Tons, in anderen Worten, "wie lange der Ton anhält". Die Einheit für die Dauer beträgt 0,01 Sekunden. Sollten Sie die "Dauer" nicht angeben, so nimmt der Computer eine Zahl von 20 an, da die zuletzt gehörte Note 0,2 Sekunden dauerte, d.h. 0,01 multipliziert mit 20.

Um einen Ton 1 Sekunde lang erklingen zu lassen, müssen Sie 100 angeben, für 2 Sekunden 200, usw. Geben Sie ein:

```
10 sound 1,478, 200
run
```

Das mittlere c wird 2 Sekunden lang erklingen.

Lautstärke

Hiermit geben Sie die Anfangslautstärke einer Note an. Sie können Werte von 0 bis 15 angeben. Eine Lautstärke von 0 ist ein Minimalwert, während 15 ein Maximalwert ist. Wird keine Zahl angegeben, so wird 12 angenommen. Geben Sie ein:

```
10 sound, 1,478, 200, 5
run
```

Beachten Sie die Lautstärke dieses Tones. Verwenden Sie nunmehr eine höhere Lautstärkenzahl:

```
10 sound, 1,478, 200, 15
run
```

Sie werden feststellen, daß diese Note jetzt wesentlich lauter ist.

Lautstärken-Hüllkurve (env)

Um die Lautstärke innerhalb der Dauer eines Tones zu verändern, kann eine Lautstärken-Hüllkurve unter Verwendung des getrennten ENV-Befehles angegeben werden. Sie können hierbei tatsächlich eine Anzahl verschiedener Lautstärken-Hüllkurven erstellen und, wie die SOUND-Steuerung, besitzt jede ihre eigene Referenznummer. Wenn Sie eine Lautstärken-Hüllkurve mit der Nummer 1 festgelegt haben, uns Sie wollen diese Hüllkurve in einem Sound-Befehl benutzen, geben Sie an der Stelle, an der der Parameter <Lautstärken-Hüllkurve> erwartet wird, eine 1 ein.
Die Erzeugung einer Lautstärken-Hüllkurve wird später kurz erläutert.

Ton-Hüllkurve (ent)

Zur Veränderung des Tones bzw. der Höhe innerhalb der Dauer des Tones können Sie eine Ton-Hüllkurve unter Verwendung des getrennten ENT-Befehles angeben. Somit können Sie tatsächlich eine Anzahl verschiedener Ton-Hüllkurven erzeugen und, wie die SOUND-Steuerung, besitzt jede ihre eigene Referenznummer. Haben Sie eine Ton-Hüllkurve mit einer Referenznummer 1 erzeugt, und möchten Sie diese innerhalb einer SOUND-Steuerung einsetzen, dann ist an Stellen, wo das Parameter <Ton-Hüllkurve> erforderlich ist, eine 1 einzugeben. Die Erzeugung einer Ton-Hüllkurve wird später kurz erklärt.

Geräusche (noise)

Sie können am Ende des SOUND-Befehls auch Geräusche anfügen. Es gibt eine Geräuschbreite im Bereich von 1 bis 31. Fügen Sie ein "Geräusch"-Parameter von 2 am Ende der SOUND-Steuerung hinzu und beachten Sie die Wirkung. Ändern Sie anschließend den Geräuschparameter auf 27 und beobachten die Abweichung. Geben Sie ein:

```
10 sound 1,478,200, 15,,,2
```

Beachten Sie darüber hinaus die "Auslass"-Parameter(,,,) vor dem Geräusch Parameter von 2. Die Erklärung liegt darin, daß wir weder eine "Lautstärken"-Hüllkurve" noch eine "Ton-Hüllkurve" angegeben haben.

Erzeugung einer Lautstärken-Hüllkurve

Das Kommando für die Lautstärken-Hüllkurve heißt env (envelope volume). Es hat normalerweise 4 Parameter, von denen die letzten drei in bis zu 5 Hüllkurven-Abschnitte vorkommen können. Wir benutzen hier nur einen Abschnitt.

ENV "Hüllkurvennummer", "Anzahl der Schritte", "Schrittamplitude" (Größe), "Schrittzeit"

Wie auch vorher werden wir die einzelnen Parameter einzeln untersuchen.

Hüllkurvennummer

Diese Nummer wird im SOUND-Kommando angegeben, um die Beziehung zu dieser Hüllkurve herzustellen. Der Bereich für die Hüllkurvennummer geht von 0 bis 15.

Anzahl Schritte

Dieser Parameter wird in Verbindung mit der Schrittzeit benutzt. Sie wollen z.B. 10 Schritte von je 1 Sekunde haben. In diesem Fall ist die Anzahl Schritte 10. Erlaubt sind 0 bis 127 Schritte.

Schrittamplitude

Jeder Schritt kann eine Amplitude von 0 bis 15, bezogen auf den letzten Schritt, haben. Diese 16 Lautstärken sind die gleichen wie im SOUND-Kommando. Der Schritt kann jedoch von -128 bis +127 variieren, so daß die Amplitude nicht nur ganz normal rauf und runter zu verändern ist, sondern auch höher als 15, wodurch wir einige außergewöhnliche Effekte erzielen. Die Schrittamplitude kann Werte von -128 bis+127 annehmen.

Schrittzeit

Dieses Parameter legt die Zeit zwischen den Schritten fest und wird in Einheiten von 0,01 Sekunden (einhundertstel Sekunden) gemessen. Der Wertbereich liegt zwischen Ø und 255, so daß die längste Zeit zwischen den Schritten 2,56 Sekunden beträgt (Ø wird als 256 behandelt).

Diesem Umstand ist es zu entnehmen, daß das Parameter "Schrittzahl" multipliziert mit der "Schrittzeit", nicht größer als das Parameter "Dauer" innerhalb der SOUND-Steuerung sein darf, da ansonsten der Ton abklingt, bevor alle Lautstärkenschritte durchlaufen wurden.
(In einem solchen Fall werden die verbleibenden Inhalte der Lautstärken-Hüllkurve ausgeblendet.)

Liegt andererseits das Parameter "Dauer" innerhalb der SOUND-Steuerung über der "Anzahl der Schritte", multipliziert mit der "Schrittzeit", so dauert der Ton an, nachdem alle Lautstärkenschritte durchlaufen sind und bleibt konstant auf dem letzten Pegel.

Um mit der Lautstärken-Hüllkurve zu üben, geben Sie folgendes Programm ein:

```
1Ø  env 1,1Ø,1,1ØØ
2Ø  sound 1,284,1ØØØ,1,1
run
```

Zeile 2Ø definiert einen Ton mit einer Tonperiode von 284 (Kammerton a), der 10 Sekunden andauert, eine Anfangslautstärke von 1 hat und die Lautstärken-Hüllkurve 1 benutzt, hierbei 1Ø Schritte umfaßt, das Volumen jedes Schrittes jeweils nach einer Sekunde (1ØØ x 0,01 Sekunden) um 1 erhöht.

Ändern Sie Zeile 1Ø jeweils wie folgt, und hören Sie die unterschiedlichen Effekte der verschiedenen Hüllkurven:

```
1Ø  env 1,1ØØ,1,1Ø
1Ø  env 1,1ØØ,2,1Ø
1Ø  env 1,1ØØ,4,1Ø
1Ø  env 1,5Ø,2Ø,2Ø
1Ø  env 1,5Ø,2,2Ø
1Ø  env 1,5Ø,15,3Ø
```

Probieren Sie schließlich folgende Möglichkeit aus:

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

Sie werden hierbei feststellen, daß die Lautstärke nach der Hälfte der Zeit konstant bleibt. Der Grund hierfür liegt darin, daß wir 50 Schritte hatten, die jeweils 0,1 Sekunden lang waren. Die gesamte Zeit der Lautstärkenveränderung betrug demnach lediglich 5 Sekunden, wobei jedoch das Parameter für die Dauer innerhalb der SOUND-Steuerung in Zeile 20 1000 betrug, d.h. 10 Sekunden.

Probieren Sie jetzt für sich, welcherlei Klangarten Sie erzeugen können.

Falls Sie die Lautstärken-Hüllkurve noch in komplizierterer Form verändern wollen, so können die drei folgenden Parameter: "Schrittzahl", "Schrittgröße", "Schrittzeit", am Ende der ENV-Steuerung noch weitere 4 mal wiederholt werden, um einen abweichenden "Abschnitt" der gleichen Hüllkurve anzugeben.

Erzeugung einer Ton-Hüllkurve

Das Kommando für die Ton-Hüllkurve heißt ENT (ENvelope Tone). In seiner einfachsten Form umfaßt das Kommando 4 Parameter: Die Steuerung wird wie folgt eingegeben:

ENT "Hüllkurvennummer", "Anzahl Schritte", "Tonperiode des Schrittes", "Schrittzeit"

Lassen Sie uns erneut die Parameter einzeln untersuchen.

Hüllkurvennummer

Hierbei handelt es sich um die Referenznummer (zwischen 0 und 15), die einer bestimmten Ton-Hüllkurve mit dem Ziel zugeordnet wird, daß ein Abruf innerhalb des SOUND-Befehls möglich ist.

Anzahl der Schritte

Dieses Parameter gibt an, wieviel verschiedene Ton(Höhen)-Schritte der Ton durchlaufen soll, bevor er abklingt. Innerhalb eines Tones, der 10 Sekunden andauert, können Sie beispiels-

weise 10 Tonschritte von jeweils 1 Sekunde vorsehen. In einem solchen Falle ist das Parameter der Tonschritte auf 1Ø zu setzen.

Erlaubt sind hierbei Ø bis 239 Schritte.

Tonperioden des Schrittes

Die Tonperiode darf je Schritt von -128 bis +127, gegenüber dem vorhergehenden Schritt, variieren. Negative Schritte erhöhen die Frequenz der Töne, negative Schritte senken die Tonhöhe ab. Die kürzeste Tonperiode ist Ø. Daran müssen Sie denken, wenn Sie die Ton-Hüllkurve berechnen. Im Kapitel 7 "Zu Ihrer Information..." ist eine Übersicht aller möglichen Tonperioden enthalten.

Schrittzeit

Dieses Parameter gibt die Zeit zwischen Schritten in 0,01 Sekunden (hundertstel Sekunden) an. Der Bereich der "Schrittzeit" liegt zwischen Ø und 255, so daß die längste Zeit zwischen den Schritten 2,56 Sekunden betragen kann (Ø wird hierbei als 256 behandelt).

Hierbei ist zu beachten, daß das Parameter "Anzahl der Schritte", multipliziert mit dem Parameter "Schrittzeit" nicht über dem Parameter "Dauer" innerhalb der SOUND-Steuerung liegen darf, da ansonsten der Ton abklingt, bevor alle Tonschritte durchlaufen sind. (In diesem Falle werden die verbleibenden Inhalte der Ton-Hüllkurve gelöscht.)

Ist andererseits das Parameter "Dauer" innerhalb der SOUND-Steuerung länger als die "Schrittzahl", multipliziert mit der "Schrittzeit", so dauert der Ton an, nachdem alle Tonschritte durchgelaufen sind und bleibt konstant mit der letzten Tonhöhe.

Zur Übung der Ton-Hüllkurve ist folgendes Programm einzugeben:

```
1Ø  ent  e,1ØØ,2,2
2Ø  sound  1,248,2ØØ,  15,,1
run
```

Zeile 2Ø definiert die Ton-Hüllkurve 1 mit 1ØØ Schritten, wobei die Tonperiode um 2 alle 0,02 Sekunden (2 hundertstel) er-

höht wird (2x 0,01 Sekunden).

Ändern Sie nunmehr die Zeile 10 jeweils wie folgt, dann lassen Sie das Programm jedesmal ablaufen (RUN) und hören Sie sich den jeweiligen Effekt bei der Änderung der Ton-Hüllkurve an:

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

Ersetzen Sie jetzt das sound-Kommando und die Ton-Hüllkurve durch Eingabe von:

```
10 ent 1,2,17,70
20 sound 1,142,140,15,,1
30,goto 10
run
```

Drücken Sie zweimal auf [ESC], um das Programm zu beenden.

Sie können nun die Lautstärken-Hüllkurve, die Ton-Hüllkurve und das SOUND-Kommando kombinieren, um verschiedene Klänge zu erzeugen. Beginnen Sie hierbei mit folgendem Beispiel:

```
10 env 1,100,1,3
20 ent 1,100,5,3
30 sound 1,284,300,1,1,1
run
```

Ersetzen Sie jetzt alle Zeichen wie folgt:

```
10 env 1,100,2,2
20 ent 1,100,-2,2
30 sound 1,284,200,1,1,1
run
```

Falls Sie noch ausgefallenere Ton-Hüllkurven erzeugen wollen, so können die 3 Parameter, "Schrittanzahl", "Tonperiode der Schritte", "Schrittzeit", zu Ende der ENT-Steuerung bis zu viermal wiederholt werden, um einen anderen "Abschnitt" der gleichen Hüllkurve anzugeben.

Versuchen Sie selbst noch weitere Variationen.

Der Abschnitt "Komplette Aufstellung der Schneider CPC664 BASIC-Schlüsselworte" enthält sämtliche Einzelheiten der verschiedenen Tonsteuerungen. Falls Sie an weiteren Tonaspekten interessiert sind, schlagen Sie den Abschnitt "Sound" auf, der sich in dem Kapitel 8 "Wenn Sie einmal Zeit haben ..." befindet.

Teil 10: EINFÜHRUNG IN AMSDOS UND CP/M

Worum handelt es sich hierbei?

Beim Einschalten bzw. der Rückstellung des Computers arbeitet das System standardgemäß mit AMSDOS. AMSDOS ist eine Abkürzung von AMStrad Disc Operating System und bietet die folgenden Datei-Steuerungen:

```
load "filename"
run "filename"
save "filename"
chain "filename"
merge "filename"
chain merge "filename"
openin "filename"
openout "filename"
closein
closeout
cat
input #9
line input#9
list #9
print #9
write #9
```

Darüber hinaus bietet AMSDOS eine Anzahl von weiteren Steuerungen für die Disketten-Organisation.

Diese Steuerungen werden als externe Steuerungen bezeichnet und jeweils mit einem davorstehenden Balkensymbol | gekennzeichnet. (Das Symbol ergibt sich durch Niederdrücken von [SHIFT] und Drücken der Taste [@]).

Eine der gebräuchlicheren externen Steuerungen, die Sie einsetzen werden, sind:

|a

|b

|tape (kann unterteilt werden in tape.in und tape.put)

|disc (kann unterteilt werden in disc.in und disc.out).

Die Steuerungen a und b werden benötigt, wenn Sie ein zweites Diskettenlaufwerk anschließen, um dem Computer mitzuteilen, welches Laufwerk anzusteuern ist, um eine darauffolgende Disketten-

Operation auszuführen. Beispielsweise kann eingegeben werden:

```
|a
load   "filename"
```

Hiermit wird dem Computer mitgeteilt, das angegebene Programm von der Diskette zu laden, die sich im Laufwerk A befindet.

Wird weder |a noch |b eingegeben, bzw. ist der Computer zurückgestellt, so arbeitet das System standardmäßig mit Laufwerk A.

Arbeiten Sie ausschließlich mit dem Diskettenlaufwerk innerhalb des Computers, so kann dieses als Laufwerk A angesehen werden, wobei |a bzw. |b nicht eingegeben werden muß. Die Eingabe von |b, wenn kein zusätzliches Disketten-Laufwerk angeschlossen ist, führt zu folgender Nachricht auf dem Bildschirm:

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

eine Angabe, die mit C (Cancel) beantwortet werden sollte.

Was passiert, wenn Sie die Cassette benutzen möchten? ...

Der Befehl |tape weist den Computer an, alle Lade- und Sicherungsbefehle usw. auf die externe Cassetteneinheit und nicht auf die Diskette zu beziehen. Ohne Eingabe von |tape richtet der Computer nach dem Einschalten oder Reset alle Befehle standardmäßig an das Diskettenlaufwerk.

Alternativ besteht die Möglichkeit, von der Cassette zu laden und auf die Diskette zu sichern. Hierzu kann der Befehl

```
|tape.in
```

verwendet werden, der Daten von der Cassette liest und sie standardmäßig auf die Diskette schreibt.

Diese Steuerung weist den Computer an, die Daten von der Cassette zu laden, jedoch weiterhin Daten auf die Diskette zu übertragen (standardmäßig)
Um umgekehrt Daten von der Diskette zu lesen und sie auf die Cassette zu schreiben, muß

```
|disc.in
```

eingegeben werden, um das vorher erteilte |tape.in (siehe oben) rückgängig zu machen.

Anschließend ist

|tape.out

einzugeben, um die Daten auf Cassette schreiben zu können.

|tape.in und |tape.out sind die Gegenbefehle für |disc.in und |disc.out und umgekehrt.

Weitere Informationen bezüglich der Übertragung der Daten zwischen Disketten und Cassette sind im weiteren Verlauf dieses Handbuches im Kapitel 4 "Das Arbeiten mit Disketten und Cassetten" enthalten.

In Teil 7 des Grundlagenkurses erfuhren Sie, wie eine Leer-Cassette unter Verwendung Ihrer Master-CP/M-System-Diskette zu formatieren ist und in welcher Form Programme zwischen den einzelnen Disketten dupliziert werden.

Ausgehend von der vorherigen Beschreibung der Steuerungen:

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

sollten Sie nunmehr in der Lage sein, Laden und Sichern auf Diskette oder Cassette (sofern angeschlossen) auszuführen, entweder in Laufwerk A bzw. Laufwerk B (sofern angeschlossen).

Weitere externe Steuerungen:

|dir |drive |era |ren |user

werden im weiteren Verlauf dieses Handbuches im Kapitel 5 "AMSDOS und CP/M" behandelt.

Kopieren mit Hilfe der CP/M-System-Diskette

Der gesamte Inhalt einer Diskette läßt sich unter Verwendung der CP/M-System-Diskette auf eine andere kopieren. Wenn Sie nur ein Disketten-Laufwerk an den Computer angeschlossen haben, sollte der Befehl DISCCOPY verwendet werden.

Mit zwei angeschlossenen Disketten-Laufwerken bietet sich der schnellere und einfachere Befehl COPYDISC an.

Kopieren mit DISCCOPY

Legen Sie die CP/M-System-Diskette ein und tippen Sie:

```
I c p m
```

Nach dem Erscheinen von A auf dem Bildschirm geben Sie dann d i s c c o p y ein.

Der Computer bitter Sie:

```
Please insert source disc into drive A then press
any key
```

Nehmen Sie also die CP/M-System-Diskette aus dem Laufwerk und legen Sie die zu kopierende Diskette ein. Wenn Sie die CP/M-Diskette selbst kopieren wollen, lassen Sie diese bitte im Laufwerk.

Nach dem Einlegen Ihrer Quellen-Diskette und dem Drücken einer beliebigen Taste, gibt der Computer folgende Meldung aus:

```
C o p y i n g  s t a r t e d
R e a d i n g  t r a c k  Ø  t o  7
```

Daraufhin fordert der Computer auf:

```
Please insert destination disc into drive A
then press any key
```

Demzufolge sollten Sie die Quellen-Diskette entfernen und die Zieldiskette einlegen, auf die Sie kopieren wollen.

Achtung!
Beachten Sie aber, daß alle vorher auf der Ziel-Diskette befindlichen Daten während des Kopiervorganges überschrieben werden.

Eine nicht korrekt formatierte oder neue, unformatierte Diskette wird automatisch neu formatiert. Nach dem Einlegen der Ziel-Diskette und der Betätigung einer beliebigen Taste erscheint vom Computer her folgende Meldung:

```
W r i t i n g  t r a c k  Ø  t o  7
```

Daraufhin werden Sie aufgefordert, die Quellendiskette erneut einzulegen, um die Spuren 8 bis 15 lesen zu können. Dieser Schreib-/Lese-Prozeß setzt sich fort (jeweils 8 Spuren) bis die letzte Spur (39) erreicht ist.

Dann erscheint die Frage:

Do you want to copy another disc (Y/N):

Wenn Sie keine weitere Diskette duplizieren wollen, geben Sie N ein und folgen den Instruktionen auf dem Bildschirm, um den DISCCOPY-Modus zu verlassen.

Kopieren mit COPYDISC

COPYDISC kann nur mit zwei Diskettenlaufwerken eingesetzt werden und ermöglicht Ihnen das Kopieren des gesamten Inhaltes einer Diskette auf eine andere, ähnlich dem vorher beschriebenen DISCCOPY. Der Vorteil von COPYDISC besteht darin, daß Sie nicht ständig die Quellen- und Zieldiskette einlegen und entfernen müssen.

Wenn Sie die Vorgehensweise bei DISCCOPY gelesen und verstanden haben, kann COPYDISC wie folgt bearbeitet werden:

Legen Sie die CP/M-System-Diskette ein und tippen Sie:

| c p m

Nach Erscheinen von A geben Sie:

c o p y d i s c

ein.

Folgen Sie nun den auf dem Bildschirm erscheinenden Angaben und der Inhalt Ihrer Quellen-Diskette wird auf die Ziel-Diskette (jeweils immer 8 Spuren) kopiert, bis die letzte Spur (39) erreicht ist.

Ebenso wie bei DISCCOPY wird auch bei COPYDISC gegebenenfalls eine automatische Formatierung durchgeführt.
COPYDISC kann zur Erzeugung einer Arbeitskopie der CP/M-System-Diskette verwendet werden.

Prüfung der Disketten

Die CP/M-System-Diskette ermöglicht den Vergleich zweier Diskketteninhalte zur Überprüfung auf fehlerhafte Datenkopie.

Mit nur einem angeschlossenen Laufwerk ist hierzu CP/M anzuwählen und

d i s c c h k

einzugeben.

Folgen Sie dann den Angaben auf dem Bildschirm und die Ziel-Diskette wird mit der Quellen-Diskette verglichen. Wenn der Computer Unterschiede zwischen den beiden Disketten feststellt, erscheint folgende Meldung:

Failed to verify destination disc correctly:
(track x sector y)

Ansonsten setzt der Computer die Prüfung fort (jeweils immer 8 Spuren). Wenn ein Fehler erkannt wurde, erscheint eine WARNUNG auf dem Bildschirm, bevor der Computer nach der Prüfung einer anderen Diskette fragt.

Haben Sie ein zusätzliches Diskettenlaufwerk angeschlossen, ist die Diskettenprüfung deutlich schneller und einfacher durchzuführen. Hierzu dient der CHKDISC-Befehl, der ähnlich dem vorher beschriebenen DISCCHK-Befehl arbeitet. Der vorteil von CHKDISC besteht jedoch darin, daß die Quellen- und Ziel-Disketten nicht ständig eingelegt bzw. entfernt werden müssen. Für den Einsatz von CHKDISC ist CP/M zu wählen und

c h k d i s c

einzugeben.

Anschließend folgen Sie bitte den Angaben auf dem Bildschirm.

Abbruch einer Operation

Funktionen der CP/M-System-Diskette können durch Drücken von [C T R L] und der C-Taste abgebrochen werden.

Daraufhin kehrt das System in den CP/M-Console-Modus zurück. ("Direct Console Mode")

Wählen Sie beispielsweise CP/M und geben Sie ein:

d i s c c o p y

Wenn Sie der Computer auffordert, eine Diskette einzulegen, drücken Sie [C T R L] und die C-Taste. Der Befehl wird daraufhin abgebrochen.

Weitere Informationen bezüglich DISCCOPY, COPYDISC, DISCCHK und CHKDISC, zusammen mit FORMAT und anderen CP/M- Befehlen sind im weiteren Verlauf dieses Handbuches im Kapitel 5 "AMSDOS und CP/M" beschrieben.

Prüfen Sie abschließend, daß Sie die am Anfang des Handbuches genannten Vorsichtsmaßnahmen mit dem Titel **"WICHTIGE HINWEISE"** beachtet haben.

Hiermit schließen wir nunmehr den 10 Teile umfassenden Grundkurs bezüglich des CPC664-Systems ab. Inzwischen sollten Sie die Funktion der Mehrzahl der auf der Tastatur befindlichen Tasten kennen, außerdem einige einfachere BASIC-Kommandos, die Form, wie eine neue Diskette so formatiert wird, daß sie einsatzbereit ist sowie die Ausführung der elementarsten Disketten-Funktionen, wie z.B. LOADing, SAVEing, CATalogisieren usw., zusammen mit einigen einfachen AMSDOS und CP/M-Kommandos. Der nächste Teil dieses Handbuches befaßt sich mit spezialisierten Aspekten des Computers und des Schneider-BASIC. Ferner werden nähere Einzelheiten des Diskettenbetriebes behandelt, in Verbindung mit Abschnitten über AMSDOS und CP/M. Ferner werden Sie in eine neue Programmier-Sprache Dr. LOGO von Digital Research eingeführt.

Für Ihre Notizen:

Kapitel 2
Weiterführungskurs

Sie haben nun den Grundlagenkurs durchgearbeitet und schon erste Erfahrungen im Umgang mit Ihrem CPC664 gewonnen, die Ihnen den nächsten Schritt erleichtern werden. Denn jetzt geht es ans Programmieren und damit an kompliziertere Zusammenhänge! Doch keine Sorge: Auch wenn Sie noch nie mit einem Computer gearbeitet haben, wird Ihnen das sicherlich sehr viel leichter fallen, als Sie erwarten.
Betrachten Sie BASIC einfach als eine Kurzform des Englischen, das sich auf wenige Worte beschränkt und auch ohne Fremdsprachenkenntnisse sehr leicht zu erlernen ist. Die dem Englischen entnommenen Befehle sind oftmals noch abgekürzt, um Vorgänge wie das manchmal zeitraubende "Eintippen" zu beschleunigen. So handelt es sich bei CLS beispielsweise nicht um einen magischen Code, sondern um die Verkürzung von CLear Screen (Löschen des Bildschirms).

Alle BASIC-Befehle, ihre Anwendung und Bedeutung, finden Sie detailliert im nächsten Kapitel erklärt, alphabetisch geordnet und übersichtlich gegliedert, so daß Sie nach und nach die für Sie interessanten Befehle erlernen können. Ihre Bemühungen werden in jedem Falle belohnt werden, wenn Sie die ersten Erfolge sehen.

Und denken Sie daran: Solange Sie nicht versehentlich Ihre Master-CP/M-Systemdiskette beschreiben, können Sie Ihrem Computer kaum schaden. Versuchen Sie also ruhig mal etwas Neues! Sie lernen Ihren CPC664 nur besser kennen.

Die Frage: Wo beginne ich?

Der Beginn ist oft der schwierigste Teil des Programms für den Anfänger. Was Sie jedoch vermeiden sollten, ist einfach an der Tastatur darauf loszuarbeiten, ohne sich etwas dabei zu denken.

Bevor Sie anfangen zu programmieren, überlegen Sie sich am besten, was Sie als Programmausführung wünschen und in welcher Form die Ergebnisse geliefert werden sollen.

Denken Sie daran: Je exakter Sie die einzelnen Programmschritte in ihrer logischen Reihenfolge vollziehen, umso weniger müssen Sie später mit GOTO-Schleifen, Rücksprüngen etc. arbeiten. Wenn ein Programm klar strukturiert ist, können Sie zu einem späteren Zeitpunkt ziemlich leicht Änderungen oder Fehlerkorrekturen einbauen.

Programmierfehler sind nicht auszuschließen. Das Schneider-BASIC hilft Ihnen vielfach durch Anzeige einer Fehlernachricht am Bildschirm und blendet außerdem die fehlerhafte Programmzeile ein. Zusätzliche Programmzeilen können Sie ohne weiteres zwischen bestehende Zeilen einfügen.

Das Schreiben eines einfachen Programms

Jetzt geht's aber wirklich los: Wir werden ein Programm mit dem Ziel aufstellen, eine Liste der Namen Ihrer Freunde und Ihrer Telefonnummern anzulegen. Wir bezeichnen das Programm als "Telefonbuch". Lassen Sie uns hierbei die obengenannten Regeln anwenden: "Worin besteht die Aufgabe des Programms?" und "Wie sind die Ergebnisse darzustellen?"

Wir gehen einmal davon aus, daß das Programm Eingeben und Abspeichern von 100 Namen und Telefonnummern zur Speicherung ermöglichen soll. Wenn Sie eine Nummer suchen, so wollen Sie nur den Namen eingeben müssen. Darüber hinaus soll es möglich sein, eine komplette Liste aller auf dem Bildschirm befindlichen Informationen zu erhalten. Nehmen Sie nebenbei zur Kenntnis, daß wir automatisch die Frage untersuchen, wie die Ergebnisse des Programms darzustellen sind.

Über die Tastatur geben wir als erstes den Titel ein (So ein Programmname ist nicht zwingend erforderlich. Wenn Sie jedoch später eine ganze Programmsammlung haben, behalten Sie dadurch besser den Überblick):

```
10 REM Telephonverzeichnis
```

Wir wollen beliebig Namen eingeben (INPUT) und zwar als Variable; wir bezeichnen diese Variable mit NAME$.

Mit der Telefonnummer geht's genauso: TEL$.

Erinnert Sie dieses Beispiel an Programme des Grundlagenkures? Dort wurden die INPUT-Kommandos zur Eingabe des Wertes der

Variablen eingesetzt; somit geben wir ein:

```
2Ø INPUT "Name"; NAME$
3Ø INPUT "Telephonnummer"; TEL$
run
```

Jetzt können Sie bereits Namen eingeben, z.B.: Meier.
Anschließend werden Sie nach der "Telefonnummer" gefragt.

Jetzt folgt der nächste Schritt:
Wir gingen davon aus, daß wir bis zu 100 Namen und 100 Telefonnummern im Programm speichern wollten. Mit Sicherheit brauchen wir kein Programm mit 100 unterschiedlichen INPUT-Kommandos sowie 200 verschiedenen PRINT-Kommandos! Machen Sie sich keine Gedanken, der Computer trifft hierfür seine Vorkehrungen:

Sie können mit einer Matrix bzw. Tabelle arbeiten. Somit hat die Variable (z.B. NAME$) eine beliebige Anzahl von "DIMensionen" (in unserem Programm benötigen wir 100). Anschließend möchten wir als Inhalt der Variablen ihre Bezeichnung mit nachstehender Referenznummer (in Klammern) haben. Diese Referenznummer wird als "subscript" bezeichnet und der Ausdruck NAME$ (27) wird beispielsweise als eine "subscripted variable" bezeichnet. Verwenden wir nunmehr eine Zahlenvariable als subscript, beispielsweise NAME$ (x), so können wir die gesamte Liste der Variablen von 1 BIS 1ØØ verarbeiten, indem wir den Wert von x in einer FOR NEXT-Schleife ändern, d.h. FOR x = 1 TO 1ØØ.
Da sich der Wert von x um 1 erhöht, ändert sich der Wert für subscript und bezieht sich auf ein verschiedenes "Element" bzw. Namen innerhalb der NAME$-Matrix bzw. Tabelle.

Wir möchten zwei Tabellen mit jeweils 100 Elementen anlegen, eine für NAME$ und eine für TEL$. Bevor wir beginnen, eine Tabelle einzusetzen, müssen wir Ihre DIMensionen angeben. Wir überschreiben die Zeilen 1Ø und 2Ø mit diesen Angaben:

```
2Ø  DIM  NAME$  (1ØØ)
3Ø  DIM  TEL$   (1ØØ)
```

Nun schreiben wir die Programmteile, mit denen wir später Namen und Nummern eingeben können:

```
40 FOR x=1 TO 100
50 INPUT;"Name"; NAME$(x)
60 INPUT "Telephonnummer"; TEL$ (x)
70 NEXT
run
```

Soweit so gut, jedoch möchten wir aber nicht alle 100 Bezeichnungen auf einmal eingeben. Was wir hier brauchen, ist ein Unterbrechungs-Bit. Bevor wir mit der Eingabe neuer Bezeichnungen und Nummern beginnen, löschen wir vorerst überflüssige Textstellen vom Bildschirm und zwar durch die Funktion CLEAR SCREEN [C L S] . Geben Sie daher ein:

```
45 [C L S]
```

Wie teilen wir nunmehr dem Rechner mit, daß wir die Eingabe der Information im Augenblick abgeschlossen haben? Mit Drücken von [E S C] würde zwar das Programm unterbrochen, sobald Sie jedoch erneut RUN eingeben, verlieren Sie alle sorgfältig eingegebenen Werte!

Lösung: Während das Programm die neue Bezeichnung aufnimmt, lassen wir es prüfen, ob ein Element für den Wert von NAME$ (x) eingegeben wurde, mit anderen Worten, ob der Wert von NAME$ (x) eine "Leerfolge" ist oder nicht. Trifft dies zu, veranlassen wir das Programm abzubrechen. Ist Ihnen klar geworden, in welcher Form wir dies tun können? Geben Sie ein:

```
55 IF NAME$ (x) = "" THEN 80
80 PRINT "Eingabe abgeschlossen"
```

Um zu erreichen, daß Ihnen das Programm mitteilt, wie die Eingabe unterbrochen wird, geben Sie hinzu:

```
47 PRINT "Eingabe beenden mit ENTER"
```

Lassen Sie uns nunmehr über das Programm die gespeicherte Information drucken (PRINT). Geben Sie ein:

```
90 FOR x=1 TO 100
100 PRINT NAME$(x);" "; TEL$(x)
110 NEXT
```

Auch hier weiß das Programm nicht, an welcher Stelle unterbrochen werden soll, bevor das hunderste Element der Tabelle er-

reicht ist; somit geben wir hinzu:

```
95 IF NAME$(x)=" "THEN 12Ø
12Ø PRINT "Verzeichnis zu Ende"
```

Die Zeile 95 erfaßt, ob es sich bei NAME$(x) um eine Leerfolge handelt; ist dies der Fall, so wird der Druckvorgang unterbrochen, indem die Zeilen 1ØØ und 11Ø ausgelassen werden.

Und nunmehr weiter in Bezug auf Ihre forderungen. Wir wollen jetzt programmieren, daß ein bestimmter Name gesucht wird, den Sie eingeben. Fügen Sie hinzu:

```
13Ø INPUT "Suchbegriff"; SEARCH$
14Ø FOR x=1 TO 1ØØ
15Ø IF INSTR (NAME$(x), SEARCH$)= Ø THEN 18Ø
16Ø PRINT NAME$(x);" "; TEL$(x)
17Ø END
18Ø NEXT
19Ø PRINT "Name nicht gefunden"
run
```

In Zeile 15Ø INSTR befindet sich eine neue Abfrage:
Diese Zeile weist den Computer an, den ersten Begriff abzufragen und einen zugehörigen Wert des zweiten Begriffs zu suchen. Sie ermittelt also NAME$(x) für SEARCH$ (die Variable aus Zeile 13Ø, die den gesuchten Namen enthält). Findet der Befehl INSTR diesen nicht, so fragt Zeile 18Ø den nächsten Wert von x ab. Sobald ein Name gefunden wird (INSTR liefert dann einen Wert ungleich Null), führt das Programm die Zeilen 15Ø und 16Ø aus. Ist die gesamte Datei leer, endet das Programm in Zeile 19Ø.

Wie Sie nunmehr feststellen werden, entwickelt sich unser Programm sehr schnell, wobei jedoch noch sehr viel Arbeit auf uns wartet. Lehnen wir uns einen Augenblick zurück und betrachten uns einige Eigenschaften dieses Programms.
Beginnen wir mit dem Programmverlauf: Zuerst geben Sie die Information ein, anschließend erhalten Sie eine Liste, dann suchen Sie einen bestimmten Namen aus.

Was ist, wenn?

Was passiert, wenn Sie diese Reihenfolge unterbrechen? Was für eine Situation ergibt sich, wenn Sie mit dem Aufsuchen eines Namens beginnen, den Sie gestern in das Programm eingegeben haben? Wie können Sie einige weitere Namen und Nummern zu den bereits bestehenden hinzufügen? Mit allen diesen Programmaspekten haben Sie sich auseinanderzusetzen und hier Lösungen zu finden; hierin besteht das eigentliche Programmieren. Wie bereits vorher erwähnt, läßt BASIC zu, auch nachträglich noch Gedanken in das Programm einzubauen, wobei jedoch ein guter Programmierer diese Probleme möglichst bereits vorher gelöst haben sollte.

So werden Sie z.B. sicher nicht daran interessiert sein, die Informationen jedesmal neu bei Benutzung des Programmes einzugeben. Demzufolge ist erforderlich, die Werte aller NAME$ und TEL$-Veränderlichen abzuspeichern, bevor Sie ausschalten. Jedesmal, wenn Sie das Programm laufen lassen, sollten Sie außerdem die Werte der Variablen abspeichern können.

Aber eines nach dem anderen: Um beim Einstieg in das Programm die gewünschten Funktionen wählen zu können, erstellen wir jetzt ein "Menü". Wie auf einer ausländischen Speisenkarte können Sie hierdurch über eine Nummer bestimmte (Programm-)"Gänge" auswählen.

```
32 PRINT "1. Daten eingeben"
33 PRINT "2. Daten anzeigen"
34 PRINT "3. Suchen"
35 PRINT "4. Daten sichern"
36 PRINT "5. Datei einlesen"
37 INPUT "Menueauswahl mit [ENTER] eingeben";ms
38 ON ms GOSUB 40,90,130
85 RETURN
125 RETURN
170 RETURN
200 RETURN
```

Wir haben das Programm veranlaßt, das Menü der Möglichkeiten auszudrucken. Anschließend wählen Sie den INPUT Ihrer Wahl und setzen die Veränderliche m s ein. Die Steuerung ON ms GOSUB in der Zeile 38 teilt dem Programm mit, daß, falls ms=1, dann GO zur ersten SUB-Routine (Zeilenzahl 4Ø); falls ms=2, dann GO zur zweiten SUB-Routine (Zeilenzahl 9Ø) usw.

Bei jeder dieser Funktionen handelt es sich um Unterprogramme (sub-routines), die über die ON ms GOSUB-Steuerung abgerufen werden. Am Ende ist jedes Unterprogramm mit RETURN abzuschliessen. Erinnern Sie sich noch an die Funktion der RETURN-Steuerung? Hierdurch kehrt BASIC vom Unterprogramm an den Programmpunkt zurück, der unmittelbar der entsprechenden GOSUB-Steuerung folgt. In diesem Falle bedeutet das Rückkehr zu dem Befehl nach Zeile 38 (das heißt, daß das Programm auf Zeile 4Ø fortsetzt - dem Eingabepunkt der Information!). Da wir diesen Vorgang nicht wünschen, haben wir hinzuzufügen:

```
39 GOTO 32
```

...um das Programm zurückzuführen und das Menü erneut anzuzeigen. Nunmehr lassen wir das Programm weiterlaufen (RUN), um festzustellen, wieweit wir gekommen sind.

Wir brechen das Programm ab (zweimaliges Drücken von [ESC]) und sehen es uns insgesamt an. Bitte geben Sie ein:

```
LIST
```

Demzufolge sollten sie folgendes Ergebnis vorfinden:

```
10 REM Telephonverzeichnis
20 DIM name$(100)
30 DIM tel$(100)
32 PRINT "1. Daten eingeben"
33 PRINT "2. Daten anzeigen"
34 PRINT "3. Suchen"
35 PRINT "4. Daten sichern"
36 PRINT "5. Datei einlesen"
37 INPUT "Menueauswahl mit [ENTER] eingeben";ms
38 ON ms GOSUB 40,90,130
```

(Fortsetzung)

```
39 GOTO 32
40 FOR x=1 TO 100
45 CLS
47 PRINT "Eingabe beenden mit [ENTER]"
50 INPUT;" Name";name$(x)
55 IF NAME$(x)="" THEN 80
60 INPUT " Telephonnummer";tel$(x)
70 NEXT
80 PRINT "Eingabe abgeschlossen"
85 RETURN
90 FOR x=1 TO 100
95 IF name$(x)="" THEN 120
100 PRINT name$(x);" ";tel$(x)
110 NEXT
120 PRINT "Verzeichnis zu Ende"
125 RETURN
130 INPUT "Suchbegriff";such$
140 FOR x=1 TO 100
150 IF INSTR(name$(x),search$)=0 THEN 180
160 PRINT name$(x);" ";tel$(x)
170 RETURN
180 NEXT
190 PRINT "Name nicht gefunden"
200 RETURN
```

Wie Sie sehen, ist durch Einfügen mehrerer Befehle z.B. der Raum zwischen Anweisung 30 und 40 nahezu ausgeschöpft. Lassen Sie uns deshalb jetzt unser Programm neu durchnumerieren. Geben Sie ein:

```
R E N U M
L I S T
```

Nunmehr haben wir folgendes Bild:

```
10 REM Telephonverzeichnis
20 DIM name$(100)
30 DIM tel$(100)
40 PRINT "1. Daten eingeben"
50 PRINT "2. Daten anzeigen"
60 PRINT "3. Suchen"
70 PRINT "4. Daten sichern"
80 PRINT "5. Date einlesen"
90 INPUT "Menueauswahl mit [ENTER] eingeben";ms
100 ON ms GOSUB 120,210,270
110 GOTO 40
120 FOR x=1 TO 100
130 CLS
140 PRINT "Eingabe beenden mit [ENTER]"
150 INPUT;" Name";name$(x)
160 IF NAME$(x)="" THEN 190
170 INPUT " Telephonnummer";tel$(x)
180 NEXT
190 PRINT "Eingabe abgeschlossen"
200 RETURN
210 FOR x=1 TO 100
220 IF name$(x)="" THEN 250
230 PRINT name$(x);" ";tel$(x)
240 NEXT
250 PRINT "Verzeichnis zu Ende"
260 RETURN
270 INPUT "Suchbegriff";such$
280 FOR x=1 TO 100
290 IF INSTR(name$(x),search$)=0 THEN 320
300 PRINT name$(x);" ";tel$(x)
310 RETURN
320 NEXT
330 PRINT "Name nicht gefunden"
340 RETURN
```

Jetzt können wir das Programm weiter ausbauen. Ein neuer Befehl sorgt dafür, daß bei Eingabe neuer Namen und Telefonnummern der Computer diese zu den bereits bestehenden hinzufügt und zwar an die erste freie Stelle der Tabelle. In diesem Falle verwenden wir die neue Steuerung LEN. Wir geben folgendes an:

Ist LEN (Länge) des Namens (NAME$) (x)) größer als 0, d.h., befindet sich bereits ein Eintrag innerhalb dieses Tabellenelements, so springen wir auf Zeile 18Ø und prüfen somit ob das nächste Element frei ist.

```
135 IF LEN /NAME$(x))  Ø THEN 18Ø
```

Ist das nicht eine einfache Lösung? Probleme dieser Art lassen sich stets mit Hilfe Ihrer Aufstellung der BASIC-Schlüsselworte und etwas Nachdenken lösen. Es wird nahezu immer zumindest einen möglichen Ablauf geben, der Ihre Programmforderungen erfüllt. Mit fortschreitender Erfahrung bei der Programmierung werden Sie in der Lage sein, Lösungen aus dem Stegreif zu finden.

Jetzt wollen wir das Abspeichern der eingegebenen Daten programmieren.

Teil 7 des Grundlagenkurses erläuterte die Abspeicherung des Programmes selbst und zwar unter Verwendung der SAVE-Steuerung.

Wir haben einen Programmabschnitt zu schreiben, der die Werte der Variablen auf die Diskette abspeichert.

Zuerst eröffnen wir eine Ausgangs-Datei (OPENOUT) und geben einen Namen wie z.B."Datei" an. Anschließend schreiben (WRITE) wir die Werte der Variablen (NAME$ (x) und TEL$ (x) von 1 bis 100 in die Datei, anschließend schließen (CLOSEOUT) wir die Ausgangsdatei und kehren (RETURN) in das Menü zurück. Die Anfügung an das Programm übernehmen wir ab Zeile 35Ø. Um es uns zu sparen, jedes neue Zeilennummer einzugeben, verwenden wir folgendes Kommando:

```
AUTO  35Ø
```

...hierdurch beginnt eine automatische Zeilennumerierung ab der geforderten Zeile:

```
350 OPENOUT "datei"
360 FOR x=1 TO 100
370 WRITE #9,name$(x),tel$(x)
380 NEXT
390 CLOSEOUT
400 PRINT "Daten gesichert"
410 RETURN
```

Drücken Sie [E S C] , um die automatische Zeilennumerierung zu beenden.

Anschließend geben wir in Zeile 100 unseres Programms an, zu welchem Punkt bei der Menü-Auswahl "4" zu verzweigen ist.

```
100 ON ms GOSUB 120, 210, 270, 350
```

Wenn Sie nunmehr die Optionsnummer 4 des Menüs wählen, speichert das Programm alle eingegebenen Informationen auf Diskette ab.
Hierbei ist zu beachten, daß in Zeile 370, in der das Programm die Werte von NAME$(x) und TEL$(x) auf Diskette schreibt, der Ausdruck # 9 nach dem Wort WRITE verwendet wird. Das Zeichen # ist eine Leitfunktion für Datenreihen (stream director), die den Computer anweist, welcher Datenreihe die Daten zuzuordnen sind.

Der Computer besitzt 10 Leitfunktionen:

- Die Übertragung der Daten durch die Leitfunktionen 0 bis 7 (#0 bis #7) führt zu einer Darstellung auf dem Bildschirm uasi als "Fenster" (WINDOW).
- Leitfunktion #8 überträgt die Daten an den Drucker (sofern letzterer angeschlossen ist).
- Leitfunktion #9 bewirkt eine Übertragung an das Diskettenlaufwerk

Wir haben nunmehr die Befehle hinzugefügt, die Information auf Diskette abzuspeichern. Nun werden wir ermöglichen, daß unser Programm jederzeit die auf Diskette gespeicherten Daten wieder in den Arbeitsbereich des Computers laden kann. Hierzu müssen wir eine andere Menü-Option in Zeile 100 anfügen:

```
100 ON ms GOSUB 120, 210, 270, 350, 420
```

Wir eröffnen eine Eingangs-Datei (OPENIN) und laden von Diskette die unter #9 dort abgespeicherten Werte der Veriablen NAME$ (x) und TEL$ (x) von 1 bis 100. Nach dem Schließen (CLOSEIN) der Eingangs-Datei kehren wir in das Menü zurück. Geben Sie ein:

```
420 OPENIN "datei"
430 FOR x=1 TO 100
440 INPUT #9,name$(x),tel$(x)
450 NEXT
460 CLOSEIN
470 PRINT "Datei eingelesen"
480 RETURN
```

Das Ende des Anfangs ...

Wir haben nunmehr ein Programm geschrieben, das die anfangs formulierten Forderungen erfüllt. Jetzt ist noch die Darstellung auf Bildschirm und/oder Drucker zu verbessern.

Der Anfang des Endes ...

Wir wollen einige Befehle eingeben, um die Programmdarstellung zu straffen:

```
34 MODE 1
```

Hierdurch wird die Betriebsart des Bildschirms festgelegt und zu Beginn des Programms der Bildschirm gelöscht. Geben Sie nunmehr ein:

```
36 WINDOW #1,13,30,10,14
```

Lassen Sie sich durch diesen kompliziert erscheinenden Befehl nicht verwirren. Wir schaffen hierdurch ein kleines Fenster auf dem Bildschirm, um das Menü einzugeben. Nach dem Wort

WINDOW geben wir zuerst die Nummer der Leitfunktion für dieses Fenster an (Zur Erinnerung: Bildschirm-Fenster sind #0 bis #7). Wird nichts angegeben, nimmt der Computer #0. Wir möchten vermeiden, die Leitfunktion #0 für unser kleines Fenster einzusetzen, da ansonsten alles vom Programm gedruckte übertragen wird. Stattdessen geben wir eine andere Leitfunktion (z.B. 1) an. Die vier anschließenden Zahlen geben Plazierung und Größe des Fensters (WINDOW) an. Wir sagen in unserem Beispiel, daß die linke Kante des Fensters bei Spalte 13 beginnt, die rechte Kante bei Spalte 3Ø endet, die obere Kante bei Reihe 1Ø beginnt und die untere Kante bei Reihe 14 endet.

Nunmehr lassen wir alle unsere Menü-Optionen in (WINDOW) #1 anzeigen, so daß wir Zeilen 4Ø bis 8Ø wie folgt aufzubereiten haben.

```
4Ø PRINT#1, "1. Daten eingeben"
5Ø PRINT#1, "2. Daten anzeigen"
6Ø PRINT#1, "3. Suchen"
7Ø PRINT#1, "4. Daten sichern"
8Ø PRINT#1, "5. Daten einlesen"
```

Außerdem wollen wir bestimmten (LOCATE), an welcher Stelle die INPUT-Menüangabe in Zeile 90 erscheint:

```
85  LOCATE 3,25
```

Um sicherzustellen, daß der Bildschirm stets gelöscht ist, wenn das Menü auf den Bildschirm übertragen wird, Eingabe von Zeile 11Ø wie folgt:

```
11Ø GOTO 34
```

Somit ist der Bildschirm jedesmal gelöscht, wenn eine Menüoption gewählt wurde; ergänzen Sie weiter:

```
95  CLS
```

Abschließend fügen wir die drei folgenden Zeilen hinzu, um vor Rückkehr zum Menü das Programm zu unterbrechen:

```
1Ø3 LOCATE 2, 25
1Ø5 PRINT "Menue anwählen mit beliebiger Taste"
1Ø7 IF INKEY$="" THEN 1Ø7
```

Zeile 1Ø3 teilt dem Computer mit, wo die in Zeile 1Ø5 angegebene Nachricht anzuzeigen ist. Das Programm bleibt solange auf Zeile 1Ø7 stehen, bis irgendeine Taste gemäß der Aufforderung aus Zeile 1Ø5 gedruckt wird.

Ist unser Programm damit beendet?

Ja, denn Sie haben jetzt jederzeit die Möglichkeit, Namen und Telefonnummern hinzuzufügen bzw. zu streichen, einen Ausdruck der Liste in alphabetischer Reihenfolge zu veranlassen, mit Ausdruck auf einem Drucker usw.

Wir möchten an dieser Stelle das Kapitel unserer Telefon-Liste in der Hoffnung verlassen, daß Sie eines in Bezug auf die Zusammenstellung eines Programmes gelernt haben. Straffen Sie nunmehr das Programm weiterhin durch Eingabe von:

RENUM

Speichern Sie es anschließend am Besten auf Diskette ab. Sie haben gerade bei Änderungen und Zugängen dann zukünftig immer die Möglichkeit, die Namens- und Telefonliste Ihrer Freunde mit geringem Aufwand immer aktuell zu halten. Und zwar ohne Radieren, Durchstreichen, neu von Hand schreiben usw., wie wir alle das von der manuellen Führung einer solchen Liste her kennen.

Abschließende Auflistung:

```
10 REM Telephonverzeichnis
20 DIM name$(100)
30 DIM tel$(100)
40 MODE 1
50 WINDOW #1,13,30,10,14
60 PRINT "1. Daten eingeben"
70 PRINT "2. Daten anzeigen"
80 PRINT "3. Suchen"
90 PRINT "4. Daten sichern"
100 PRINT "5. Datei einlesen"
110 LOCATE 3,25
120 INPUT "Menueauswahl mit [ENTER] eingeben";ms
130 CLS
140 ON ms GOSUB 190,290,350,430,500
```

Fortsetzung: Abschließende Auflistung

```
150 LOCATE 2,25
160 PRINT "Menue anwaehlen mit beliebiger Taste"
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 "Eingabe beenden nur mit [ENTER]"
230 INPUT;" Name";name$(x)
240 IF name$(x)="" THEN 270
250 INPUT " Telephonnummer";tel$(x)
260 NEXT
270 PRINT "Eingabe abgeschlossen"
280 RETURN
290 FOR x=1 TO 100
300 IF name$(x)="" THEN 330
310 PRINT name$(x);" ";tel$(x)
320 NEXT
330 PRINT "Verzeichnis zu Ende"
340 RETURN
350 INPUT "Suchbegriff";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)
460 NEXT
470 CLOSEOUT
480 PRINT "Daten gesichert"
490 RETURN
500 OPENIN "datei"
510 FOR x=1 TO 100
520 INPUT #9,name$(x),tel$(x)
530 NEXT
540 CLOSEIN
550 PRINT "Datei eingelesen"
560 RETURN
```

Für Ihre Notizen:

Kapitel 3

Komplette Aufstellung der Schneider CPC664-BASIC-Schlüsselwörter

WICHTIGER HINWEIS

Für die folgenden, allgemein dargestellten Befehle gelten diese grundsätzlichen Regeln:

1. Begriffe oder Zeichen, die ohne Klammern dargestellt werden, sind in genau dieser Form einzugeben:

   Beispiel: E N D

2. Eckige Klammern und der in ihnen enthaltende Begriff gelten als Platzhalter, dürfen also auf keinen Fall eingegeben werden.

   Beispiel: E D I T [Z e i l e n n u m m e r]

   Der Ausdruck [Zeilennummer] muß durch die gewünschte Zahl ersetzt werden, etwa:

   E D I T 1 Ø Ø

3. Darüber hinaus können Anweisungen in eckigen Klammern wahlweise entfallen:

   Beispiel: R U N [Z e i l e n n u m m e r]

   kann verwendet werden als

   R U N

   oder

   R U N 2 Ø Ø

   Auf die optionale Verwendung wird in den Erläuterungen jeweils hingewiesen.

4. RundeKlammern bleiben stets erhalten, und der in ihnen dargestellte Begriff ist unbedingt zu ersetzen.

   Beispiel: P R I N T C O S (numerischer Ausdruck)

   An die Stelle von numerischer Ausdruck tritt der konkrete Zahlenwert:

   P R I N T C O S (4 5)

SONDERZEICHEN

& or &H kennzeichnet die nachfolgende Konstante als sedezimal

&H kennzeichnet die nachfolgende Konstante als binär (dual)

kennzeichnet die nachfolgende Konstante als Ein-/Ausgabeeinheit (Leitfunktion stream)

DATENTYPEN

Textfolgen können 0 bis 255 Zeichen lang sein und werden durch Textausdruck (string expression) dargestellt. Texte können darüber hinaus mit einem "+" Zeichen aneinander gehängt werden. Dabei darf die Gesamtlänge natürlich 255 Zeichen nicht überschreiten.

Zahlenwerte können entweder ganzzahlig oder reell sein. Ein ganzzahliger Ausdruck kann die Werte von -32768 bis +32767 annehmen. Ein reeller Ausdruck hat eine Genauigkeit von etwas über neun Stellen und zwar innerhalb des Bereiches von ± 1,7E + 38, mit dem kleinsten Wert über Null von etwa 2,9E-39.

Ein numerischer Ausdruck hat immer einen Zahlenwert als Ergebnis. Dabei kann es sich um eine echte Zahlenangabe, eine Variable oder einen mathematischen Ausdruck handeln. Hierbei ist kurz gesagt alles, was keinen Textausdruck darstellt, erlaubt. Die Ein-/Ausgabeeinheit stellt einen numerischen Ausdruck dar, der z.B. den Bildschirm, den Drucker oder die Diskette identifiziert innerhalb der der Text der Ein-/Ausgabeeinheit (stream) gefordert wird.

Eine Liste von Einheiten (item) beschreibt einen Parameter, der eine Aufstellung von Dateneinheiten umfaßt, die durch Kommas getrennt werden. Die Liste kann hierbei eine bestimmte bzw. irgendeine Zahl von Dateneinheiten umfassen, die jeweils durch die maximale Zeilenlänge begrenzt sind.

Hierzu bestehen folgende Kennzeichnungen:

% Ganze Zahl

! Reelle Zahl (Standard)

$ Zeichenfolge

Bitte beachten Sie ferner, daß Schneider CPC664-BASIC-Befehle nachstehend in folgender Form aufgelistet sind:

SCHLÜSSELWORT	**Beispiel: ASC**
Syntax	ASC (<Textausdruck>)
Beispiel	PRINT ASC("x")
Beschreibung	Funktion: Gibt den numerischen Wert ...
Zugehörige Schlüsselwörter	CHR$

Bei Schlüsselwörtern handelt es sich entweder um:

KOMMANDOS:	Operationen, die direkt ausgeführt werden
FUNKTIONEN:	Operationen, die als Argumente in einen Ausdruck eingebracht werden
OPERATOREN:	Ausdruck, der nach mathematischen Regeln arbeitet

BASIC setzt alle in Kleinbuchstaben eingebenen Befehle in Großbuchstaben um, wenn ein Programm aufgezeichnet wird (LIST). Die in diesem Kapitel gezeigten Beispiele arbeiten mit der Großbuchstabenstellung, da in dieser Form bei Aufnahme das Programm erscheint. Demgegenüber sollten Sie jedoch Ihre Eingaben in Kleinschreibung vornehmen, da hierdurch Fehler leichter erkennbar sind und in kleinen Buchstaben bei (LIST) angezeigt werden.

SCHLÜSSELWÖRTER

ABS

ABS (<numerischer Ausdruck>)

```
PRINT ABS(-67.98)
 67.98
```

FUNKTION: Gibt den absoluten Wert des angegebenen Ausdrucks zurück, d.h. negative Zahlen werden positiv zurückgegeben.

Verwandte Befehle: SGN

AFTER

AFTER <ganzzahliger Ausdruck>[,<ganzzahliger Ausdruck>] GOSUB <Zeilennummer>

```
10 AFTER 250 GOSUB 60:CLS
20 PRINT "Rate einen Buchstaben in 5 Sekunden"
30 a$=INKEY$:IF flag=1 THEN END
40 IF a$ <> CHR$(INT(RND*26+97)) THEN 30
50 PRINT a$; "Richtig !! Du hast Gewonnen !!"
60 SOUND 1,478 : SOUND 1,358:END
70 PRINT "Du hast zu lange gebraucht. Ich habe gewonnen !!"
80 SOUND 1,2000:flag = 1:RETURN
run
```

KOMMANDO: Ruft in bestimmten Zeitabständen ein Unterprogramm auf. Der erste<ganzzahlige Ausdruck>beschreibt das Zeitintervall in Einheiten zu 0,02 Sekunden. Der zweite<ganzzahlige Ausdruck>(im Bereich zwischen 0 bis 3) gibt die Uhr (4 mögliche) an, die verwendet werden soll.

Jedes der Zeitwerke kann mit einem zugehörigen Unterprogramm arbeiten.

Weitere Informationen in Bezug auf Unterbrechungen sind in Teil 2 des Kapitels 8 "Wenn Sie Zeit haben..." enthalten.

Verwandte Befehle: EVERY, REMAIN, RETURN

A N D

A r g u m e n t A N D A r g u m e n t

```
If "Hans"<"Uwe" AND "Hund">"Maus" then print "richtig" else print "falsch"
Richtig
If "Uwe"<"Hans" AND "Maus">"Hund" then print "richtig" else print "falsch"
Falsch
If "Hans"<"Uwe" AND "Maus">"Hund" then print "richtig" else print "falsch"
Falsch
....
print 1 AND 1
1
print 0 AND 0
0
print 1 AND 0
0
```

OPERATOR: Bitweise Durchführung des Bool'schen Rechenvorgangs auf ganzen Zahlen. Das Ergebnis ist Ø, falls nicht beide Argument-Bits 1 sind.

Weitere Informationen bezüglich des logischen Aufbaus sind ebenfalls in Teil 2 des obengenannten Abschnittes enthalten.

Verwandte Befehle: O R , N O T , X O R

A S C

A S C (< T e x t a u s d r u c k >)

```
P R I N T  A S C ( " X " )
 8 8
```

FUNKTION: Gibt den numerischen Wert nur des ersten Zeichens des Textausdruckes zurück, falls ASCII-Zeichen verwendet werden.

Verwandte Befehle: C H R $

A T N

A T N (< numerischer Ausdruck >)

```
PRINT ATN(1)
 Ø.785398163
```

FUNKTION: Gibt den Wert zwischen -PI/2 und +PI/2 im Bogenmaß zurück, der sich aus dem Arcustangens des angegebenen <numerischen Ausdrucks> errechnet.

Verwandte Befehle: S I N , C O S , T A N , D E G , R A D

AUTO

A U T O [<Zeilennummer>] [,<Schrittweite>

```
AUTO 1ØØ,5Ø
```

KOMMANDO: Erzeugt automatisch Zeilennummern. <Zeilennummer> gibt die erste Zeilennummer an, mit der begonnen wird, wenn z.B. ein Programm verlängert werden soll. Die<Schrittweite> bestimmt die Differenz zur nächsten Zeilennummer. Wird einer der beiden Eingabewerte nicht angegeben, so wird 10 als Standardwert angenommen.

Falls eine Zeile mit derselben Zeilennummer schon vorhanden ist, werden die Inhalte der Zeile am Bildschirm angegeben und können erforderlichenfalls korrigiert werden. Wird [E N T E R] gedrückt, so wird die angezeigte Zeile in den Speicher übernommen.

Zur Unterbrechnung der automatischen Zeilennumerierung ist [E S C] zu drücken.

Verwandte Befehle: keine

BIN$

BIN$ (<vorzeichenloser ganzzahliger Ausdruck>[, <ganzzahliger Ausdruck>]

```
PRINT BIN$(64,8)
Ø1ØØØØØØ
```

FUNKTION: Gibt einen Text zurück, der den Wert des <vorzeichenlosen ganzzahligen Ausdrucks> in binärer (dualer) Form enthält. Falls der zweite <ganzzahlige Ausdruck> größer ist, als die Anzahl der sowieso benötigten Zeichen, wird der Text mit führenden Nullen versehen, bis zu der durch den zweiten <ganzzahligen Ausdruck> angegebenen Gesamtlänge. Ist die Anzahl zu klein, so wird der entstehende Ausdruck **nicht** auf die angegebene Zahl der Digits verkürzt, sondern die erforderlichen Digits werden ergänzt.

Der "vorzeichenlose ganzzahlige Ausdruck", der in eine binäre Form umzusetzen ist, muß einen Wert innerhalb des Bereiches -32768 bis 65535 besitzen.

Verwandte Befehle: DEC$, HEX$, STR$

BORDER

BORDER <Farbe>[,<Farbe>]

```
10 rem 729 Randfarben-Kombination!
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 " "; a;",";b
70 FOR t=1 TO 500
80 NEXT t,b,a
run
```

KOMMANDO: Die nicht beschreibbare Randfläche des Bildschirms wird in der angegebenen <Farbe> dargestellt. Wird eine zweite <Farbe> angegeben, so wechselt die Randfläche immer wieder ihr Aussehen zwischen den beiden Farben. Das Zeitintervall zwischen dem Farbwechsel wird durch das Kommando SPEED INK festgelegt. <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 <Adresse>[,<Liste von Parametern>]
CALL &BD19

KOMMANDO: Springt in ein Unterprogramm außerhalb des BASIC-Bereiches, das an der angegebenen Speicheradresse beginnt. Bei der Verwendung dieses Kommandos ist äußerste Vorsicht geboten. Es sollte nur von wirklich erfahrenen Programmierern angewendet werden. Der als Beispiel verwendete C A L L ist relativ harmlos, da er auf den nächsten Bildaufbau wartet (erst wenn der Elektronenstrahl das neue Bild ganz aufgebaut hat, wird es auch sichtbar). Damit können Bewegungen von Zeichen auf dem Bildschirm wirklichkeitsgetreu dargestellt werden.

Verwandte Befehle: U N T

CAT

C A T
C A T

KOMMANDO: CATalogisiert den Diskettenlnhalt. Zeigt in alphanumerischer Reihenfolge den vollen Namen aller auf der Diskettenseite befundenen Dateien mit dazugehöriger Länge (aufgerundet auf das nächst höhere Kbyte) an. Darüber hinaus wird der freie Speicherplatz auf der Diskette, Angaben über das benutzte Diskettenlaufwerk und den Benutzer angegeben.

Der C A T - Befehl hat keine Auswirkungen auf das sich augenblicklich im Speicher befindende Programm.

CHAIN

C H A I N <Dateiname>[,<Zeilennummernausdruck>]
C H A I N "testprog.bas", 35Ø

KOMMANDO: Läd ein Programm von der Diskette und ersetzt das augenblicklich geladene Programm. Das neue Programm wird entweder von Anfang an oder von der Zeile ausgeführt, die im <Zeilennummernausdruck> angegeben wurde.

Programme, die mit "SAVE<Programmname>,p" gespeichert worden sind, können mit dem C H A I N -Befehl geladen und ausge-

führt werden.

Verwandte Befehle: CHAIN MERGE, LOAD, MERGE

CHAIN MERGE

CHAIN MERGE <Dateiname>[,<Zeilennummernausdruck>]
[,DELETE<Zeilennummernbereich>]

CHAIN MERGE "Test.bas", 75Ø, DELETE 4ØØ-68Ø

KOMMANDO: Lädt ein Programm von der Diskette, übernimmt es in das laufende Programm des Speichers, beginnt anschließend mit dem sich ergebenden Programm, entweder vom Anfang oder von einer Zeile aus, die angegeben wurde.
Vor Durchführung der Funktion CHAIN MERGE ist es erforderlich Teile des ursprünglichen Programms zu löschen, wobei jedoch der Befehl DELETE <Zeilennummernbereich> eingesetzt werden kann.

Hierbei ist darauf zu achten, daß die für das alte Programm eingesetzten Zeilennummern, die sich im neuen Programm befinden, nach CHAIN MERGE durch die neuen Programmzeilen überschrieben werden.

Programme, die mit "SAVE<Dateiname>,p" gesichert worden sind können nicht mit "CHAIN MERGE" geladen oder ausgeführt werden

Verwandte Befehle: CHAIN, DELETE, LOAD, MERGE

CHR$

CHR$ (<Ganzzahliger Ausdruck>)

```
1Ø FOR x=32 TO 255
2Ø PRINT x; CHR$ (x),
3Ø NEXT
run
```

FUNKTION: Setzt einen ganzzahligen Ausdruck im Bereich von 0 bis 255 auf die entsprechenden Textzeichen (CHaRacter String) um und zwar unter Verwendung des Zeichensatzes Schneider CPC664 gemäß Abschnitt 3 des Kapitels "Wenn Sie Zeit haben..."

Beachten Sie, daß die Zeichen zwischen Ø und 31 Steuerzeichen sind. Die Steuerzeichen in obigem Beispiel können ausgedruckt werden, da sie im Bereich zwischen 32 und 255 liegen.

Verwandte Befehle: A S C

CINT

```
CINT (<numerischer Ausruck>)
```

```
10 n=1.999
20 PRINT CINT(n)
run
 2
```

FUNKTION: Gibt den gerundeten ganzzahligen Wert des <numerischen Ausdrucks> zurück, der im Wertbereich von -32768 bis +32767 liegt.

Verwandte Befehle: CREAL, FIX, INT, ROUND, UNT

CLEAR

```
CLEAR
CLEAR
```

KOMMANDO: Setzt alle Variablen auf Null und vergißt alle Dateien.

CLEAR INPUT

```
CLEAR INPUT
```

```
1Ø CLS
2Ø PRINT "Geben Sie jetzt Buchstaben ein!"
3Ø FOR t=1 TO 3000
4Ø NEXT
5Ø CLEAR INPUT
run
```

KOMMANDO: Löscht alle vorher eingegebenen Eingaben der Tastatur und diejenigen, die sich im Tastatur-Speicher befinden.

Um die Auswirkungen dieses Befehles festzustellen, starten Sie das oben genannte Programm (RUN), mit Eingabe von Buchstaben, soweit gefordert. Anschließend löschen Sie Zeile 5Ø des Programms und starten es erneut (RUN), um den Unterschied festzustellen.

Verwandte Befehle: INKEY, INKEY$, JOY

C L G

C L G[<F a r b s t i f t>]
C L G

KOMMANDO: Füllt den Graphikschirm mit der dem <F a r b s t i f t> (I N K) zugeordneten Farbe. Ist kein F a r b s t i f t angegeben, wird der beim letzten CLG-Kommando angegebene Farbstift verwendet, ansonsten der Farbstift 0.

Verwandte Befehle: C L S, G R A P H I C S P A P E R, I N K, O R I G I N

C L O S E I N

CLOSEIN

KOMMANDO: Schließt jede Eingabedatei vom Diskettenlaufwerk

Verwandte Befehle: E O F, O P E N I N

C L O S E O U T

CLOSEOUT

CLOSEOUT

KOMMANDO: Schließt die Ausgabe einer Datei auf dem Diskettenlaufwerk

Verwandte Befehle: OPENOUT

C L S

CLS[#<Ein-Ausgabeeinheit>]

```
10 PAPER # 2,3
20 CLS # 2
run
```

KOMMANDO: Setzt alle Variablen auf Null und vergißt alle Dateien. Ist kein Ausdruck für die Datenreihe angegeben, so wird die Bildschirmfolge # Ø gelöscht.

Verwandte Befehle: C L G, I N K, P A P E R, W I N D O W

C O N T

```
CONT
CONT
```

KOMMANDO: Setzt ein durch *B r e a k * (zweimaliges Drücken der [E S C] -Taste), S T O P oder E N D abgebrochenes Programm fort, falls es nicht geändert wurde.

Direkte Befehle können vor C O N T bzw. der Fortsetzung des Programms eingegeben werden.

Verwandte Befehle: S T O P

C O P Y C H R $

```
COPYCHR$ (#<Ein-/Ausgabegerät>)
```

```
10 CLS
20 PRINT"Oben links"
30 LOCATE 1,1
40 a$ = COPYCHR$(#0)
50 LOCATE 1,20
60 PRINT a$
run
```

FUNKTION: Kopiert ein Zeichen von der augenblicklichen Stellung innerhalb der Datenreihe (die anzugeben ist). Bei dem obengenannten Programm wird ein Zeichen von der Position 1,1 kopiert (oben links) und auf die Stelle 1,20 wiedergegeben (0).

Wird das gelesene Zeichen nicht erkannt, so wird eine Null ausgegeben.

Verwandte Befehle: L O C A T E

C O S

```
COS(<numerischer Ausdruck>)
```

```
DEG
PRINT COS (45)
run
```

FUNKTION: Berechnet den Kosinus des numerischen Ausdrucks.

Hierzu ist anzumerken, daß DEG und RAD dazu benutzt werden können, das Ergebnis der obigen Berechnung in Grad bzw. als Bogenmaß auszudrücken.

Verwandte Befehle: ATN, DEG, RAD, SIN

CREAL

CREAL (<numerischer Ausdruck>)

```
10 a=PI
20 PRINT CINT(a)
30 PRINT CREAL(a)
run
#
3.14159265
```

FUNKTION: Weist den numerischen Ausdruck in reeller Form aus.

Verwandte Befehle: CINT

CURSOR

CURSOR [<Systemschalter>][,<Bedienerschalter>]

```
10 CURSOR 1
20 PRINT"FRAGEN ?";
30 A$ = INKEY$:IF a$="" THEN 30
40 PRINT a$
50 CURSOR 0
run
```

KOMMANDO: Setzt den System- bzw. den Bedienerschalter zum Cursor auf Ein bzw. Aus. Bei den Parametern für Systemschalter und Bedienerschalter hat es sich entweder um 0 (Aus) bzw. 1 (Ein) zu handeln. Bei der obengenannten INKEY$-Steuerung, bei der der Cursor normalerweise nicht sichtbar ist, wurde der Cursor über den Systemschalter, Schaltstellung 1 (in Zeile 10) eingeschaltet.

Der Cursor wird angezeigt, wenn sowohl der Systemschalter, als auch der Bedienerschalter eingeschaltet (1) ist. Der Systemschalter wird automatisch für die Steuerung INPUT eingeschaltet, jedoch für INKEY$ ausgeschaltet.

Bei der Sichtbarmachung von Text auf dem Bildschirm wird empfohlen, den Cursor auszuschalten.

Auf eines der Schaltparameter kann verzichtet werden, nicht jedoch auf beide. Wird auf ein Schaltparameter verzichtet, so wird dieser spezielle Schaltzustand nicht geändert.
Verwandte Befehle: LOCATE

DATA

DATA <Liste von:<Konstanten>>

```
10 REM Namensliste
20 DIM Vorname$(4)
30 DIM Nachname$(4)
40 FOR n = 1 TO 4
50 READ Vorname$(n),Nachname$(n)
60 PRINT Vorname$(n);" ";Nachname$(n)
70 DATA Bernhard,Koch,Peter,Mayer,Wulf
80 DATA Huber,Fritz,Schiller
90 NEXT
run
```

KOMMANDO: Stellt Datenwerte dem Programm zur Verfügung. Es ist eine der meistgenützten Möglichkeiten in BASIC, Datenwerte in DATA-Kommandos zusammenzustellen, um sie bei Bedarf abzurufen. Die angegebenen Daten müssen mit der Art der Variablen übereinstimmen, die sie aufnehmen sollen. DATA-Kommandos können überall im Programm stehen.

Weitere Informationen hinsichtlich der Daten sind in Teil 2 des Kapitels 8 "Wenn Sie Zeit haben..." angegeben.

Verwandte Befehle: READ, RESTORE

DEC$

DEC$ (<numerischer Ausdruck>,<Formatschablone>)

```
PRINT DEC$(10↑7,"ff#########,.##")
£10,000,000.00
```

KOMMANDO: Richtet die Druckausgabe des <numerischen Ausdrucks> unter Verwendung der <Formatschablone> aus.

Dabei können ausschließlich folgende Formatzeichen verwendet werden:

+ - £ $ * # , . ↑

Der Einsatz dieser "Formatfeld-Kennzeichnungen" ist unter dem Schlüsselwort PRINT USING beschrieben.

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"Das Programm wurde vor ";
60 PRINT FNuhr;" Sekunden gestartet."
70 RETURN
run
```

KOMMANDO: BASIC ermöglicht es, innerhalb eines Programms sog. Funktionen zu definieren, die einen Wert zurückgeben. Die mit DEF FN-Kommandos erzeugten Funktionen gelten innerhalb des Programms und arbeiten auf dieselbe Art und Weise wie die in BASIC eingebauten Funktionen (z.B. COS).

Die Funktion kann überall im Programm verwendet werden. Die zu übergebenden Variablen müssen im Typ mit den in der Funktion verwendeten übereinstimmen. Außerdem sollte die Definition der Funktion im Programmablauf nur einmal durchlaufen werden.

Im obengenannten Beispiel kann festgestellt werden, wie der Wert der Funktion FNuhr ständig korrigiert wird, selbst wenn das Programm durch einmaliges Drücken der Taste [ESC] unterbrochen wird. Das gleiche gilt, wenn Sie nach zweimaligem Drücken der Taste [ESC] das Programm mit CONT fortsetzen. fortsetzen.

DEFINT

DEFINT <Buchstabenbereich(e)>

```
10 DEFINT z
20 zahl = 123.456
30 PRINT zahl
run
 123
```

KOMMANDO: Legt fest, daß alle Variablen, die mit den angegebenen <Buchstaben> beginnen, vom Variablentyp INT ganzzahlig (Integer) sind. Dabei werden Groß- und Kleinbuchstaben nicht unterschieden.

DEFINT a,b,c

DEFINT a-z

Verwandte Befehle: DEFREAL, DEFSTR

D E F R E A L

D E F R E A L <Buchstabenbereich(e)>

D E F R E A L x , a - f

KOMMANDO: Legt fest, daß alle Variablen, die mit den angegebenen <Buchstaben> beginnen, vom Variablentyp REAL (mit Dezimalpunkt) sind. Dabei werden Groß- und Kleinbuchstaben nicht unterschieden.

D E F R E A L a , b , c

D E F R E A L a - z

Verwandte Befehle: D E F I N T , D E F S T R

D E F S T R

D E F S T R <Buchstabenbereich(e)>

```
10 DEFSTR n
20 name = "Schneider"
30 PRINT name
run
Schneider
```

KOMMANDO: Legt fest, daß alle Variablen, die mit den angegebenen <Buchstaben> beginnen, vom Variablentyp STR Text sind. Dabei werden Groß- und Kleinbuchstaben nicht unterschieden.

D E F S T R a , b , c

D E F S T R a - z

Verwandte Befehle: DEFINT, DEFREAL

D E G

D E G
D E G

KOMMANDO: Schaltet auf Winkelgradmaß um. Standardmäßig werden

die in den Winkelfunktionen wie S I N und C O S verwendeten Werte als im Bogenmaß angegeben verwendet. Das Kommando schaltet nun die Interpretation auf Winkelgradmaß um, bis sie durch C L E A R oder R A D oder das Laden eines neuen Programms wieder aufgehoben wird.

Verwandte Befehle: ATN, COS, RAD, SIN, TAN

D E L E T E

D E L E T E <Zeichenbereich>

D E L E T E 1ØØ-2ØØ

KOMMANDO: Löscht alle Programmzeilen innerhalb des <Zeilenbereichs>. Da die gelöschten Zeilen nur wieder neu eingegeben werden können, sollte das Kommando mit Vorsicht benützt und auf Schreibfehler überprüft werden, bevor Sie den Befehl ausführen.

Auf die erste bzw. letzte Nummer innerhalb des Zeilenbereiches kann verzichtet werden, um anzugeben "...von Beginn des Programms" bzw. "bis zum Ende des Programms", d.h.:

DELETE -2ØØ

... bzw....

DELETE 5Ø-

...bzw....

DELETE

... wodurch das gesamte Programm gelöscht wird.

Verwandte Befehle: CHAIN MERGE, RENUM

D E R R

D E R R

```
LOAD xyz.abc

XYZ.ABC not found
Ready
PRINT DERR
146
```

FUNKTION: Verweist auf den letzten vom Diskettensystem zurückgesendeten Fehlercode. Der Wert von DERR kann dafür eingesetzt werden, um festzustellen, ob ein bestimmter Diskettenfehler aufgetreten ist. Siehe Aufstellung der Fehlernachrichten im

Kapitel 8 "Wenn Sie Zeit haben...".

Verwandte Befehle: ERL, ERR, ERROR, ON ERROR, GOTO, RESUME

D I

D I

```
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: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:RETURN
run
```

KOMMANDO: Verhindert alle Unterbrechungen, außer durch [E S C], bis sie durch das E I -Kommando bzw. das RETURN-Kommando eines Unterbrechungs-Unterprogramms wieder ermöglicht werden.

Hierbei ist darauf hinzuweisen, daß die Eingabe eines Unterbrechungs-Unterprogramms automatisch Unterbrechungen gleicher bzw. niedrigerer Priorität ausschaltet.

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 Hauptprogramm und das Unterprogramm jeweils den graphischen Bildschirm.

Verwandte Befehle: AFTER, EI, EVERY, REMAIN

D I M

D I M Liste von:<indizierten Variablen>

```
10 CLS
20 DIM name$(5),telnr$(5)
30 FOR n=1 TO 5
40 PRINT"Laufende Nr.: ";n
50 INPUT "Geben Sie den Namen ein";name$(n)
60 INPUT "Geben Sie die Telefon Nr. ein ";telnr$(n)
70 PRINT
80 NEXT
90 FOR n = 1 TO 5
100 PRINT n;name$(n),telnr$(n)
110 NEXT
run
```

KOMMANDO: Stellt Speicherbereich für Felder (Matrizen) bereit und legt den höchsten Tabellenwert (Index) fest. In BASIC muß die Indexobergrenze angegeben werden oder aber es wird der Standardwert 10 angenommen. Ist die Feldgröße einmal festgelegt, darf sie nicht geändert werden, da sonst ein Fehler auftritt. Eine <indizierte Variable> kann mit dem gleichen Variablennamen soviele Werte aufnehmen, wie durch die ganzzahligen Indexobergrenzen in der <Indexliste> angegeben sind. Man kann sich z.B. die erste Zahl in einer <Indexliste> vorstellen als die Stockwerke eines Parkhauses und die zweite dann als die Anzahl der Stellplätze je Stockwerk. Volles Verständnis für den Aufbau von Feldern ist eine wichtige Voraussetzung für fortgeschrittene BASIC-Programmierung. Die Größe eines Feldes ist einzig vom zur Verfügung stehenden Speicherbereich und der Vorstellungskraft des Programmierers begrenzt.

Hierbei ist zu beachten, daß der niedrigste Wert der indizierten Variablen (d.h. das erste verfügbare Element innerhalb des Feldes) Null ist.
Felder können mehr-dimensional aufgebaut sein, wobei jedes Element eines solchen Feldes durch seine Lage innerhalb des Rahmens festgelegt ist. Beispielsweise kann sich innerhalb eines Rahmens folgende Dimensionierung ergeben:

```
DIM position$ (20, 20, 20)
```

...somit würde ein Element des Feldes beispielsweise wie folgt benannt:

```
position$ (4,5,6)
```

Verwandte Befehle: ERASE

D R A W

D R A W<x-Koordinate>,<y-Koordinate>[,<Farbstift>][,
<Farbstiftmodus>]

```
10 MODE 1: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 eine Linie von der augenblicklichen Cursorposition zur angegebenen absoluten (x,y)-Position auf dem graphischen Bildschirm. Die Koordinatenangaben sind für alle drei Bildschirmmodi dieselben.

Der Wertbereich des <Farbstiftes> liegt zwischen Ø und 15.
Die wahlfreie Angabe des <Farbstiftmodus> bestimmt, wie die Farbe des gerade benutzten Farbstiftes mit den anderen Farben, die sich bereits auf dem Bildschirm befinden, zusammenwirkt.
Die vier <Farbstiftmodi> sind:

0 : Normal
1 : XOR (ausschließliche ODER)
2 : UND
3 : ODER

Verwandte Befehle: DRAWR, GRAPHICS PEN, MASK

D R A W R

D R A W R <x-Versatz>,<y-Versatz>[,<Farbstift>][,
<Farbstiftmodus>]

```
10 CLS:PRINT"Kommst du 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 von der augenblicklichen Cursor-Position zur angegebenen relativen (x,y)-Position, um x und y-Zeichenpositionen versetzt. Der Wertbereich des wahlfreien Parameters <Farbstift> liegt zwischen Ø und 15.
Die wahlfreie Angabe des <Farbstiftmodus> bestimmt, wie die Farbe des gerade benutzten Farbstiftes mit den anderen Farben,

die sich bereits auf dem Bildschirm befinden, zusammenwirkt. Die vier <Farbstiftmodi> sind:

0 : Normal
1 : XOR (ausschließlich ODER)
2 : UND
3 : ODER

Verwandte Befehle: DRAW GRAPHICS PEN, MASK

E D I T

E D I T <Zeilennummer>

E D I T 2Ø

KOMMANDO: Zeigt die in der Zeilennummer angegebene Programmzeile auf dem Bildschirm, zusammen mit dem betriebsbereiten Cursor an.

Verwandte Befehle: AUTO, LIST

E I

E I
E I

KOMMANDO: Um Unterbrechungen nach einem DI-Kommando wieder zu ermöglichen.

Werden Unterbrechungen durch ein Interrupt-Programm gesperrt, so erfolgt dann automatisch Freigabe, wenn BASIC auf die RETURN-Steuerung bei Abschluß des Unterprogramms trifft.

Weitere Informationen bezüglich der Unterbrechungen sind in Teil 2 des Kapitels "Wenn Sie Zeit haben ..." enthalten.

Verwandte Befehle: AFTER, DI, EVERY, REMAIN

E L S E

(Siehe IF)

E N D

E N D
E N D

KOMMANDO: Beenden des Programms. Ein Programm endet auch mit der letzten Befehlszeile. END schließt alle Dateien und schaltet in den Direkteingabemodus um. Noch gespeicherte Tonfolgen werden bis zu ihrem Ende weitergespielt.

Verwandte Befehle: STOP

E N T

E N T <Folgenummer>[,<Veränderungsfolgen>]

```
10 ENT 1,10,-50,10,10,50,10
20 SOUND 1,500,200,10,,1
run
```

KOMMANDO: Ermöglicht einen auszugebenden Ton in seiner Höhe zu variieren. Eine Ton<Veränderungfolge> bestimmt, wie die Tonhöhe variiert werden soll.

Die <Folgenummer> ist ein <ganzzahliger Ausdruck>, dessen Wert zwischen 1 und 15 liegen muß, und kennzeichnet die Tonveränderungsfolge. Falls die <Folgenummer> negativ ist, wird die <Veränderungsfolge> wiederholt.

Bis zu fünf <Veränderungsfolgen> können jeweils angegeben werden, in der Form <Schrittanzahl>, <Pausezeichen>, oder: <Tonperiode>, <Pausezeichen>.

Die erste Form beschreibt eine schrittweise Veränderung relativ zur momentanen Tonperiode, die zweite entspricht einem absoluten Setzen der Tonperiode. Dabei bedeutet: <Schrittanzahl> einen <ganzzahligen Ausdruck> zwischen 0 und 239, der die Anzahl der auszuführenden Einzelveränderungen darstellt, <Schrittweite> einen <ganzzahligen Ausdruck> zwischen -128 und +127, der die Differenz zwischen den jweiligen Tonperioden darstellt.

<Pausezeit> einen <ganzzahligen Ausdruck> zwischen 0 und 255, der die Wartezeit zwischen den jeweiligen Tonveränderungen darstellt. Dabei wird die Zeit in Einheiten zu 0.01 Sekunden gerechnet, außerdem wird für die Angabe 0 der Faktor 256 gerechnet.

<Tonperiode> einen <ganzzahligen Ausdruck> zwischen 0 und 4095, der den neuen Ton angibt.

Das SOUND-Kommando setzt die Start-Tonperiode und kann eine der 15 Folgenummern ansprechen. Wird keine oder eine nicht definierte Folgenummer angegeben, behält der angegebene Ton immer seine Originalhöhe.

Eine Tonveränderung hat keine Auswirkungen auf die Dauer des Tones. Sind noch Schritte in der Veränderungsfolge übrig, obwohl der Ton schon geendet hat, werden diese nicht mehr ausgeführt.

Eine zu wiederholende Tonveränderungsfolge wird immer wieder von vorne begonnen, so lange der Ton dauert.

Die im Kommando angegebenen Werte werden bei der Ausführung gespeichert und können später immer wieder abgerufen werden. Das Kommando muß also nicht wiederholt werden.

Jedesmal, wenn eine Tonveränderungsfolge mit der gleichen <Folgenummer> definiert wird, werden die alten Werte überschrieben. Findet eine solche Änderung statt, während ein SOUND-Kommando sie benutzt, so entstehen unbestimmte, aber vielleicht auch interessante Effekte.

Wird ein ENT-Kommando ohne <Veränderungsfolgen> angegeben, so werden die gespeicherten Werte der angesprochenen <Folgenummer> gelöscht.

Weitere Informationen bezüglich der Tonfolge sind in Teil 2 des Kapitels "Wenn Sie Zeit haben ..." behandelt.

Verwandte Befehle: ENV, SOUND

E N V

E N V <Folgenummer>[<Veränderungsfolgen>]

```
10 ENV 1,15,-1,10,15,1,10
20 SOUND 1,200,300,15,1
run
```

KOMMANDO: Ermöglicht einen auszugebenden Ton in seiner Lautstärke zu verändern und beschreibt, wie die Lautstärke verändert werden soll. Dabei kann die Schrittanzahl zwischen 0 und 127, die Schrittweite zwischen -128 und +127 und die Pausenzeit in Einheiten zu 0.01 Sekunden zwischen 1 und 256 liegen.

Die <Folgenummer> ist ein <ganzzahliger Ausdruck>, dessen Wert zwischen 1 und 15 liegen muß und kennzeichnet die Lautstärkeveränderungsfolge.

Bis zu fünf <Veränderungsfolgen> können jeweils angegeben werden in der Form: <Schrittanzahl>, <Schrittweite>, <Pausezeit>

oder = <Registerwert>, <Veränderungsperiode>.

Die erste Form beschreibt eine Veränderung mit Hilfe der Software. Dabei bedeutet:

<Schrittanzahl> einen <ganzzahligen Ausdruck> zwischen 0 und 127, der die Anzahl der auszuführenden Einzelveränderungen darstellt. <Schrittweite> einen <ganzzahligen Ausdruck> zwischen -128 und +127, der die Differenz zwischen den jeweiligen Lautstärken (Amplituden) darstellt oder aber, falls die <Schrittanzahl> gleich 0 ist, den absoluten Lautstärkewert.

<Pausezeit> einen <ganzzahligen Ausdruck> zwischen 0 und 255, der die Wartezeit zwischen den jeweiligen Lautstärkeveränderungen darstellt. Dabei wird die Zeit in Einheiten zu 0.01 Sekunden gerechnet. Außerdem wird für die Angabe 0 der Faktor 256 gerechnet.

Die zweite Form beschreibt eine Veränderung durch direkten Zugriff auf den Tongenerator. Dabei bedeutet <Registerwert> den in das Lautstärkeregister <Register 15 oktal> zu setzenden Wert. <Veränderungsperiode> die in die Veränderungsfolgenregister (Register 13 und 14 oktal) zu setzenden Werte.

Setzt man direkt die Hardwareregister, so wird die Lautstärkeveränderung sofort ausgeführt. Es ist daher ratsam, vor der nächsten Lautstärkeveränderung eine Tonpause einzulegen. Gibt es keine weitere Lautstärkeveränderung, so wird eine Pause von 2 Sekunden ausgeführt.

Das SOUND-Kommando setzt die Start-Lautstärke und kann eine der 15 Folgenummern ansprechen. Wird keine oder eine nicht definierte Folgenummer angegeben, bleibt die Lautstärke immer gleich. Wird zwar eine <Schrittanzahl>, aber eine <Schrittweite> 0 angegeben, so bleibt die Lautstärke gleich.

Die im Kommando angegebenen Werte werden bei der Ausführung gespeichert und können später immer wieder abgerufen werden. Das Kommando muß also nicht wiederholt werden.

Jedesmal, wenn eine Lautstärkeveränderungsfolge mit der gleichen <Folgenummer> definiert wird, werden die alten Werte überschrieben. Findet eine solche Änderung statt, während ein

SOUND-Kommando sie benützt, so entstehen unbestimmte, aber vielleicht auch interessante Effekte. Wird ein ENV-Kommando ohne <Veränderungsfolgen> angegeben, so werden die gespeicherten Werte der angesprochenen <Folgenummer> gelöscht.

Weitere Informationen in Bezug auf Töne sind in Teil 2 des Kapitels "Wenn Sie Zeit haben..." beschrieben.

Verwandte Befehle: ENT, SOUND

E O F

E O F

```
10 OPENIN "exl.bas"
20 WHILE NOT EOF
30 LINE INPUT #9,a$
40 PRINT a$
50 WEND
60 CLOSEIN
run
```

FUNKTION: Gibt den Wert -1 (wahr) zurück, wenn eine Eingabe von Diskette das Dateiende erreicht hat, ansonsten den Wert 0 (falsch).

Verwandte Befehle: OPENIN

E R A S E

ERASE <Liste von <Variablennamen>>

```
DIM a (100),b$(100)
ERASE a,b$
```

KOMMANDO: Löscht die angegebenen Variablen. Falls ein Feld nicht mehr benötigt wird, kann es gelöscht werden, um den Speicherplatz anderweitig verwenden zu können.

Verwandte Befehle: DIM

E R L

E R L

```
10 ON ERROR GOTO 30
20 GOTO 1000
30 PRINT "Der Fehler ist in Zeile: ";ERL
40 END
```

FUNKTION: Der Befehl E R L wird in Fehlerbehandlungsroutinen verwendet, um festzustellen, in welcher Zeile der letzte Fehler auftrat (in diesem Beispiel in Zeile 20).

Verwandte Befehle: DERR, ERR, ERROR, ON ERROR GOTO RESUME

E R R

E R R

```
GOTO 500
Line does not exist
Ready
PRINT ERR
  8
```

FUNKTION: Meldet die Nummer des zuletzt festgestellten Fehlers. (siehe Aufstellung der Fehlernachrichten in Kapitel "Wenn Sie einmal Zeit haben...")
Im obigen Beispiel tritt der Fehler mit der Nummer 8 "Die Zeile existiert nicht" auf.

Verwandte Befehle: DERR, ERL, ERROR, ON ERROR GOTO RESUME

E R R O R

E R R O R <ganzzahliger Ausdruck>

```
10 IF INKEY$="" THEN 10 ELSE ERROR 17
run
```

KOMMANDO: Führt eine Behandlung des im "ganzzahligen Ausdruck" genannten Fehlers durch. Eine Aufstellung der Fehlernachrichten 1 bis 32 befindet sich im Kapitel "Wenn Sie Zeit haben..."

BASIC behandelt den Fehler als echt ermittelten Fehler, springt in die Fehlerroutine und liefert darüber hinaus die zugehörigen Werte von ERR und ERL.

ERROR, in Verbindung mit einem ganzzahligen Ausdruck im Bereich 33 bis 255 kann verwendet werden, um normale Fehlernachrichten gemäß des folgenden Beispiels zu erstellen:

```
10 ON ERROR GOTO 100
20 INPUT "Gib ein Zeichen ein ";a$
30 IF LEN(a$)<>1 THEN ERROR 100
40 GOTO 20
100 IF ERR = 100 THEN 110 ELSE 130
110 PRINT CHR$(7)
120 PRINT "Ich sagte e i n  !!  Zeichen "
130 RESUME 20
run
```

Verwandte Befehle: ERL, ERR, ON ERROR GOTO, RESUME

E V E R Y

EVERY <Zeitperiode>,[<Zeitwerknummer>] GOSUB <Zeilennummer

```
10 EVERY 50,1 GOSUB 30
20 GOTO 20
30 SOUND 1,20
40 RETURN
run
```

KOMMANDO: Ruft ein BASIC-Unterprogramm in regelmäßigen Abständen auf. Der 1. Parameter gibt das Intervall (in Einheiten von 0,02 Sekunden) an, in den das Unterprogramm verzweigt werden soll.

Die Zeitwerknummer (im Bereich von 0 bis 3) gibt an, welches der 4 Verzögerungs-Zeitwerke einzusetzen sind. Zeitwerk 3 hat die höchste Priorität; Zeitwerk 0 (das Standardzeitwerk) hat die niedrigste Priorität.

Jedes der Zeitwerke kann mit einem zugehörigen Unterprogramm ausgerüstet sein.

Weitere Informationen bezüglich der Unterbrechungen sind in Teil 2 des Kapitels "Wenn Sie Zeit haben..." enthalten.

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 <Schreibstift>

```
10 MODE 1:sea=2:MOVE 0,200,3
20 DRAWR 100,-100 : DRAWR 220,0
30 MOVER 10,0:DRAWR 220,0
40 DRAWR 100,100:MOVE 0,0
50 FILL sea
run
```

KOMMANDO: Füllt einen beliebigen Bereich des graphischen Bildschirms aus. Die Fläche ist durch Linien begrenzt, die entweder in der Farbe des Schreibstiftes für graphischen Bildschirm oder in der Farbe, mit der die Figur ausgefüllt werden soll, gezeichnet worden sind (innerhalb des Bereiches von 0 bis 15).

Der Ausfüllvorgang beginnt ab der Cursor-Stellung des graphischen Bildschirms. Liegt dieser Punkt an einer Kante, so wird keine Figur ausgefüllt.

Verwandte Befehle: GRAPHICS PEN

FIX

FIX (<Numerischer Ausdruck>)

```
PRINT FIX(9.99999)
  9
```

FUNKTION: Gibt die ganze Zahl zurück, die durch Abschneiden der Dezimalstellen entsteht (also keine Rundung wie bei CINT).

Verwandte Befehle: CINT, INT, ROUND

F N

(Siehe DEF FN)

F O R

F O R <einfache Variable> = <Start> TO <Ende> [STEP <Schrittweite>]

```
10 FOR n = 2 TO 9 STEP 2
20 PRINT n
30 NEXT n
40 PRINT "Hast du das erwartet ?"
```

KOMMANDO: Wiederholt einen vorgegebenen Teil eines Programms so oft, wie durch die <Start>- und <Ende>-Werte angegeben wird. Falls keine <Schrittweite> angegeben ist, wird ein Wert 1 angenommen.

Die Schrittgröße kann als negativer numerischer Ausdruck angegeben werden; in diesem Falle muß der Wert des "Start"-Parameters größer sein als der des "Ende"-Parameters, da ansonsten die variable Steuerung nicht auslöst.

Bei "FOR NEXT"-Schleifen kann eine Verschachtelung auftreten, d.h. eine Schleife kann jeweils innerhalb einer anderen ausgeführt werden usw.

Bei der NEXT-Anweisung braucht der Variablenname nicht unbedingt angegeben werden, da BASIC automatisch bestimmt, welche FOR-Steuerung mit einem "anonymen" NEXT zu verbinden ist.

Verwandte Befehle: NEXT, STEP, TO

F R A M E

F R A M E

```
10 MODE 0
20 PRINT "FRAME ausgeschaltet"
30 TAG
40 MOVE 0,200
50 FOR x=0 TO 500 STEP 4
60 IF f=1 THEN frame
```

Fortsetzung:

```
70 MOVE x,200
80 PRINT" ";CHR$(143)
90 NEXT
100 IF f=1 THEN RUN
110 CLS
120 TAGOFF
130 PRINT "FRAME eingeschaltet"
140 f=1
150 GOTO 30
```

KOMMANDO: Synchronisiert das Schreiben graphischer Symbole auf dem Bildschirm mit dem Strahlrücklauf des Bildschirms. Der Gesamteffekt besteht hierbei darin, daß die Bewegung von Zeichen bzw. Graphiksymbolen auf dem Bildschirm weicher und ohne Flackereffekt erscheint.

Verwandte Befehle: TAG, TAGOFF

F R E

(<Numerischer Ausdruck>)

```
PRINT FRE(Ø)
PRINT FRE("")
```

FUNKTION: Gibt die Größe des von BASIC nicht benutzten Speicherplatzes zurück. FRE("") löst ein Sammeln auch kleinster freier Speicherplätze aus, bevor die Größe des freien Speicherplatzes zurückgegeben wird.

Verwandte Befehle: HIMEM, MEMORY

G O S U B

G O S U B <Zeilennummer>

G O S U B 2 1 Ø

KOMMANDO: Ruft ein BASIC-Unterprogramm, indem zur angegebenen <Zeilennummer> gesprungen wird.

Verwandte Befehle: RETURN

GOTO

GOTO <Zeilennummer>

```
GOTO 9Ø
```

KOMMANDO: Springt zur angegebenen <Zeilennummer>.

GRAPHICS PAPER

GRAPHICS PAPER <Schreibstift>

```
10 MODE 0
20 MASK 15
30 GRAPHICS PAPER 3
40 DRAW 640,0

run
```

KOMMANDO: Bestimmt die "Farbe" der graphischen Darstellung, d.h. die Hintergrundfläche auf dem Bildschirm. Beim Ausziehen von durchgehenden Linien wird diese Hintergrundfläche nicht sichtbar. Im obengenannten Beispiel ermöglicht der MASK-Befehl das Ausziehen einer unterbrochenen Linie sowie die Darstellung der Untergrundfarbe.

Die Untergrundfarbe der graphischen Symbole (im Bereich von 0 bis 15) wird für die "Darstellungsfläche" von Zeichen verwendet, die bei TAG- geschrieben werden; als Standardsymbol, wenn mit CLG das Graphikfenster gelöscht wird.

Verwandte Befehle: CLG, GRAPHICS PEN, INK, MASK, TAG, TAGOFF

GRAPHICS PEN

GRAPHICS PEN [<Schreibstift>] ,[<Betriebsart Hingrund>]

```
10 MODE 0
20 GRAPHICS PEN 15
30 MOVE 200,0
40 DRAW 200,400
50 MOVE 639,0
60 FILL 15
run
```

STEUERUNG: Bestimmt die "Farbe" (innerhalb des Bereiches zwischen 0 und 15) zum Zeichnen von Linien und für das Setzen

von Punkten. Die Hintergrundfarbe kann hierbei gesetzt werden entweder als

0 : Nicht transparenter Hintergrund
1 : Transparenter Hintergrund

(Der transparente Hintergrund beeinflußt die Farbe der mit TAG geschriebenen Zeichen sowie die der Zwischenräume in den gestrichelten Linien).

Es kann höchstens auf einen Parameter verzichtet werden. Hierbei wird die bestehende Einstellung nicht verändert.

Verwandte Befehle: GRAPHICS PAPER, INK, MASK, TAG, TAGOFF

HEX$

HEX$(<Vorzeichenloser Ganzzahliger Ausdruck>)[,
<Ganzzahliger Ausdruck>]

```
PRINT HEX$ (255,4)
ØØFF
```

FUNKTION: Gibt den Wert des <vorzeichenlosen ganzzahligen Ausdrucks>in hexadezimaler Form zurück. Der zweite <ganzzahlige Ausdruck> gibt die Anzahl der zurückzugebenden Zeichen an und muß zwischen 0 und 16 liegen.

Der vorzeichenlose ganze Ausdruck, der in hexadezimaler Form umzusetzen ist, muß zwischen -32767 und 65535 liegen.

Verwandte Befehle: BIN$, DEC$, STR$, UNT

HIMEM

```
PRINT HIMEM
42619
```

FUNKTION: Enthält die höchste von BASIC belegte Speicheradresse und kann in numerischen Ausdrücken in üblicher Weise benutzt werden.

Bevor man mit dem MEMORY-Kommando die oberste Speicheradresse neu festlegt, empfiehlt es sich, mit dem Kommando mm=HIMEM den bisherigen Wert sicherzustellen, um ihn später mit MEMORY mm wieder setzen zu können.

Verwandte Befehle: FRE, MEMORY, SYMBOL, SYMBOL AFTER

I F

I F <logischer Ausdruck>THEN<Befehle>[ELSE<Befehle>]

```
10 MODE 1
20 x=CINT(RND*100)
30 PRINT"Errate meine Zahl (0-100)"
40 INPUT n
50 IF n < x THEN PRINT n"Deine Zahl ist zu niedrig "
60 IF n > x THEN PRINT n"Deine Zahl ist zu gross"
70 IF n = x THEN 80 ELSE c=c+1:GOTO 40
80 PRINT "Bravo,  du hast meine Zahl erraten !!"
90 PRINT c+1;" Versuche !"
run
```

KOMMANDO: Je nach dem Ergebnis des <logischen Ausdrucks> wird entweder die eine oder die andere Anweisung ausgeführt. Ergibt der <logische Ausdruck > den Wert 'wahr', wird der THEN- oder GOTO-Teil ausgeführt. Im Falschfalle wird der ELSE- Teil ausgeführt oder, falls keiner angegeben ist, mit dem nächsten gefundenen Befehl weitergemacht. In IF-Kommando kann nicht länger als eine Zeile sein und darf als <Befehle>beliebige Kommandos beinhalten, außer ein weiteres IF-Kommando.

Verwandte Befehle: ELSE, GOTO, THEN

I N K

I N K <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"Ink ";p;", ";i
60 FOR t=1 TO 400:NEXT t,i,p
70 INK 0,1:INK 1,24:CLS
run
```

KOMMANDO: Ordnet einem bestimmten Farbstift Farben zu. Der Parameter <Farbe> beschreibt die Farbreferenz, bei der es sich um einen ganzen Ausdruck innerhalb des Bereiches 0 bis 15 für den Einsatz in dem zugehörigen PEN- bzw. PAPER-Befehl handeln muß; bei dem ersten Farbparameter muß es sich um einen

ganzen Ausdruck handeln, der einen Farbwert im Bereich 0 bis 26 besitzt. Wird eine zweite Farbe als Option angegeben, so wechselt der Farbstift zwischen zwei Farben und zwar in einer Geschwindigkeit, die von dem Befehl SPEED INK bestimmt wird.

Abhängig vom derzeitigen Bildschirmmodus ist eine bestimmte Zahl von Farben verfügbar; siehe Farbtabelle im Grundlagenkurs, Kapitel 8, Tabelle 1.

Verwandte Befehle: GRAPHICS PAPER, GRAPHICS PEN, PAPER, PEN, SPEED INK

I N K E Y

I N K E Y (<Ganzzahliger Ausdruck>)

```
10 IF INKEY (55) <> 32 THEN 10
20 PRINT "Sie haben -SHIFT- und V gedrueckt"
30 CLEAR INPUT
RUN
```

FUNKTION: Gibt zurück, welche Taste der Tastatur gedrückt wurde. Die Tastatur wird in Abständen von 0,02 Sekunden abgefragt. Die Funktion kann benutzt werden, um Ja/Nein-Abfragen auszuwerten, wenn es dabei nicht wichtig ist, zu wissen, ob große oder kleine Buchstaben eingegeben werden. Das oben angegebene Beispiel prüft, ob [SHIFT] und **V** gleichzeitig gedrückt wurden.

Eine Darstellung der Tastennummern ist im oberen rechten Diagramm des Diskettenlaufwerkes enthalten, ferner im Kapitel "Wenn Sie Zeit haben..."

Der Zustand von [SHIFT] und [CTRL] in Verbindung mit der im <ganzzahligen Ausdruck> angegebenen Taste wird wie folgt bezeichnet:

Rückgabewert	[SHIFT]	[CTRL]	Taste
-1	?	?	nicht gedrückt
0	nicht gedrückt	nicht gedrückt	gedrückt
32	gedrückt	nicht gedrückt	gedrückt
128	nicht gedrückt	gedrückt	gedrückt
160	gedrückt	gedrückt	gedrückt

Verwandte Befehle: CLEAR, INPUT, INKEY$, YOY

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 ausgewaehlt ":END
90 PRINT "Du hast Nein ausgewaehlt "
run
```

FUNKTION: Gibt den Wert der gedrückten Taste zurück, um Eingaben übernehmen zu können (da angezeigt wird, ob eine Taste gedrückt wurde). Wurde keine Taste gedrückt, so gibt INKEY$ einen "Nulltext" zurück. Im obengenannten Beispiel weisen die Zeilen 40 und 70 das Programm an, auf Zeile 30 zurückzukehren, nachdem die Tastatur abgefragt wurde.

Verwandte Befehle: CLEAR INPUT, INKEY

INP

INP (<Schnittstellenadresse>)

```
PRINT INP (&FF77)
 255
```

FUNKTION: Gibt den Eingabewert der angegebenen Eingabe-Schnittstelle (I/O port) zurück.

Verwandte Befehle: OUT, WAIT

INPUT

INPUT[#<Ein-/Ausgabenummer>,][;][<Ausgabetext>;]
<Liste von: [Variablen]>

```
10 CLS
20 INPUT "Geben Sie zwei durch Komma getrennte Zahlen ein";a
,b
30 IF a=b THEN PRINT "Die beiden Zahlen sind gleich"
40 IF a>b THEN PRINT a;"ist groesser als ";b
50 IF a<b THEN PRINT a;"ist kleiner als ";b
60 CLEAR:GOTO 20
run
```

KOMMANDO: Liest Daten vom mit <Ein-/Ausgabenummer> angesprochenen Eingabegerät. Ein Strichpunkt nach INPUT unterdrückt den Zeilenvorschub nach der Eingabe. Ein Strichpunkt nach dem <Ausgabetext> gibt als Eingabesignal ein Fragezeichen aus. Ein Komma unterdrück das Fragezeichen. Wird eine Eingabe gemacht, die nicht zum angegebenen Variablentyp paßt (z.B. der Buchstabe O statt die Zahl Ø für einen <numerischen Ausdruck>), antwortet BASIC mit

```
?REDO from start
```

und der Wiederholung des <Ausgabetextes> und wartet auf eine neue Eingabe.

Alle Eingaben müssen mit [ENTER] abgeschlossen werden. Der Strichpunkt nach <Ein-/Ausgabenummer> unterdrückt den Zeilenvorschub, d.h. der Cursor bleibt hinter dem letzten eingegebenen Zeichen in der Zeile stehen. Handelt es sich um eine Eingabe von Cassette, so wird der <Ausgabetext> unterdrückt. So kann dasselbe INPUT-Kommando durch Umsetzen der Variablen <Ein-/Ausgabenummer> mit unterschiedlichen Auswirkungen verwendet werden.

Je ein Datenausdruck wird für jede Variable in der <Liste von: <Variablen>> aus der Datei gelesen. Die Eingaben müssen mit den Variablentypen im INPUT Kommando übereinstimmen und die einzelnen Eingaben durch ein Komma, [ENTER], Leertaste oder Dateiende getrennt bzw. abgeschlossen werden. Kommas oder [ENTER]-Eingaben nach Leerzeichen werden ignoriert. Texte, die nicht in Anführungszeichen stehen, werden mit Komma oder Leertaste in Einzelwerte aufgeteilt.

Verwandte Befehle: LINE INPUT

INSTR

```
INSTR[(]<Ganzzahliger Ausdruck>,[)]<Textausdruck>,
      <Textausdruck>)
```

```
PRINT INSTR(2,"BANANA","AN")
```

FUNKTION: Gibt die Positionsnummer zurück, an der der zweite <Textausdruck> gefunden wurde. Die Suche beginnt am Anfang des ersten <Textausdrucks> oder bei der durch den <Ganzzahligen Ausdruck> angegebenen Position (im Bereich von 1 bis 255). Tritt der zweite <Textausdruck> nicht im ersten <Textausdruck> auf, so wird der Wert Ø zurückgegeben.

Verwandte Befehle: MID$, LEFT$, RIGHT$

I N T

I N T (<Numerischer Ausdruck>)

```
PRINT INT (-1.995)
-2
```

FUNKTION: Gibt den zum nächstkleineren gerundeten ganzzahligen Wert zurück. INT arbeitet wie das FIX-Kommando, nur daß für negative Werte um 1 kleinere Werte zurückgegeben werden, falls sie nicht schon ganzzahlig sind.

Verwandte Befehle: CINT, FIX, ROUND

J O Y

J O Y (<ganzzahliger Ausdruck>)

```
10 PRINT "Um das Programm zu beenden"
20 PRINT "bewege den Joystick"
30 IF JOY (0)<> 0 THEN END
40 GOTO 10
run
```

FUNKTION: Gibt den Wert der Eingabe vom durch den <Ganzzahligen Ausdruck> (entweder 0 oder 1) angegebenen Joystick zurück. Dabei werden die Bewegungen durch die Werte der einzelnen Bits weitergegeben.

Bit	Bedeutung	Dezimalwert
0:	Vorwärts	1
1:	Rückwärts	2
2:	Links	4
3:	Rechts	8
4:	Feuer 2	16
5:	Feuer 1	32

Wird beispielsweise der "Feuer"-Hauptknopf (Feuer 2) am ersten Joystick gedrückt, während der Griff des Joysticks nach links bewegt wird, so liefert die Funktion JOY (Ø) einen Dezimalwert von 20 entsprechend 16 (Feuer 2) + 4 (links).

Weitere Informationen bezüglich der Joystick sind in Kapitel "Wenn Sie Zeit haben..." enthalten.

Verwandte Befehle: CLEAR INPUT, INKEY

K E Y

K E Y <ganzzahliger Ausdruck>,<Textausdruck>

```
KEY 11, "border 13:paper Ø:pen 1:ink Ø,13:
ink 1,0: mode 2: list"+CHR$(13)
```

...anschließend ist die kleine [ENTER] - Taste zu drücken.

KOMMANDO: Belegt eine Funktionstaste mit Werten. Hierbei wird zwischen zweiunddreißig Tastatur-Erweiterungen im Bereich von 0 bis 31 unterschieden, die Tastenwerte zwischen 128 und 159 belegen.
Die Tasten 128 (0 auf numerischer Tastatur) bis 140([CTRL][ENTER] auf numerischen Tastaturen) sind standardmäßig den Ziffern 0 bis 9, der kleinen [ENTER] Taste, dem Dezimalpunkt und der Tastenkombination [CTRL] kleine [ENTER] Taste (mit der Funktion Run" für Cassettenbetrieb), wobei jedoch eine andere Zuordnung für einen anderen Textausdruck erfolgen kann. Bei den Erweiterungen 13 bis 31 (Tastenwerte 141 bis 159) handelt es sich um standardmäßige Leerketten, die jedoch unter Anwendung des im nächsten Beispiel beschriebenen KEY DEF-Befehls erweitert und Tasten zugeordnet werden können.

Die innerhalb der KEY-Steuerung angegebene Erweiterungsnummer kann sich im Bereich zwischen 0 und 31 bzw. zwischen 128 und 159 befinden, um die Tastenwerte auszudrücken (siehe Illustration in Bezug auf die Tasten in Kapitel "Wenn Sie Zeithaben...").

Die Summe aller in <Textausdruck> verwendeten Zeichen dürfen insgesamt 120 Zeichen nicht überschreiten. Der Versuch einer Erweiterung über diese Grenze hinaus vorzunehmen, führt zu einem Fehler vom Typ "Improper argument" (5).

Verwandter Befehl: KEY DEF

K E Y D E F

K E Y D E F <Tastennummer>,<Wiederholung>[,<Normalzeichen>[,<mit Shift>[,<mit CTRL>]]]

```
KEY 159, "dies ist die Tabulator-Taste"
KEY DEF 68, 1, 159
```

anschließend ist die Taste [TAB] zu drücken.

KOMMANDO: Definiert die von "Tastennummer" zurückgemeldeten KEY-Werte innerhalb des Bereiches zwischen 0 und 79 (eine Illustration der Tastennummern ist im Diagramm auf der oberen

rechten Seite des Diskettenlaufwerkes enthalten).

Die Parameter <Normalzeichen>, <mit Shift> und <mit CTRL> sollten hierbei die Werte enthalten, die zurückgemeldet werden, wenn die Taste entweder allein, bzw. in Verbindung mit [SHIFT] oder [CTRL] gedrückt wird. Jeder dieser Parameter ist hierbei wahlfrei.

Der Parameter "Wiederholung" ermöglicht Ihnen automatische Wiederholfunktion der Taste, Ein- bzw. Ausschaltung (1 bzw. Ø), wobei die Geschwindigkeit der automatischen Wiederholung über die SPEED KEY-Steuerung einstellbar ist.

Im obengenannten Beispiel wird die Taste 159 (entspricht der Erweiterung 31) zuerst einer Erweiterungskette zugeordnet, im Anschluß definiert der Befehl KEY DEF die Taste 68 (die Taste TAB) auf automatische Wiederholung (1), wobei der <Normalzeichen> -Wert 159 zurückgegeben wird, wenn die Taste allein gedrückt wird.

Im obengenannten Beispiel ergibt sich die normale Tastaturbelegung wieder durch:

KEY DEF 68,Ø,9

...wobei 9 den normalen ASCII-Wert für (TAB) ausdrückt.

Verwandte Befehle: KEY, SPEED KEY

L E F T $

L E F T $ (<Textausdruck>,<ganzzahliger Ausdruck>)

```
10 CLS
20 A$="Schneider"
30 B$=LEFT$(A$,3)
40 PRINT b$
run
```

FUNKTION: Gibt aus dem <Textausdruck> vom Beginn soviele Zeichen zurück, wie der <ganzzahlige Ausdruck> angibt. Ist der <Textausdruck> kürzer als die gewünschte Länge, so wird der gesamte <Textausdruck> zurückgegeben.

Verwandte Befehle: MID$, RIGHT$

LEN

LEN(<Textausdruck>)

```
10 A$ = "SCHNEIDER"
20 PRINT LEN(A$)
run
9
Ready
```

FUNKTION: Gibt die Anzahl aller Zeichen einschließlich Leerzeichen zurück, die der <Textausdruck> enthält.

LET

LET <logischer Ausdruck>

LET x = 100

KOMMANDO: Zuweisung eines Wertes an eine Variable. Veraltete Form aus früheren BASIC-Sprachen, bei der die Zuordnungen der Variablen zu überprüfen waren. Findet keine Verwendung innerhalb des Schneider-BASIC, mit Ausnahme des Nachweises der Kompatibilität mit den Programmen, die in früheren BASIC-Übungshandbüchern enthalten sind. Das Beispiel muß im Schneider-BASIC nur in der Form

x=100

geschrieben werden.

LINE INPUT

LINE INPUT[#<Ein-/Ausgabegerät>,][;][<Ausgabetext>;]<Textvariable>

```
10 LINE INPUT "Geben sie eine Textzeile ein ";a$
20 CLS
30 PRINT "Die Variable a$ enthaelt nun :"
40 PRINT a$
run
```

KOMMANDO: Ermöglicht die Eingabe einer gesamten Textzeile vom angegebenen Ein-/Ausgabegerät (standardmäßig 0, sofern nicht angegeben). Das Semikolon (;) nach <Ein-/Ausgabegerät> unterdrückt die Funktion Wagenrücklauf/Zeilenschaltung, die normalerweise bei Ausführung des Befehls erfolgt.

Als Trennelement kann entweder ein Semikolon oder ein Komma eingegeben werden. Ein Semikolon führt dazu, daß ein Fragezei-

chen angezeigt wird; ein Komma unterdrückt das Fragezeichen.

Die Funktion LINE INPUT von der Tastatur wird durch Drücken der Taste [ENTER] beendet.

LINE INPUT von der Diskette (bzw. von der Cassette), Ausgabe #9 wird durch Ausführung eines Wagenrücklaufes bestätigt bzw. durch die "Textvariable" mit Zuordnung auf 255 Zeichen, wobei der jeweils zuerst eintreffende Fall gültig ist.

Verwandte Befehle: INPUT

L I S T

L I S T [<Zeilenbereich>][,#<Ein-/Ausgabegerät>]

L I S T 10Ø-1ØØØ,#1

KOMMANDO: Listet die angegebenen Programmzeilen auf dem <Ein-/Ausgabegerät> auf. #Ø ist die Standard-Bildschirmausgabe, #8 der Drucker. Die Auflistung kann durch einmaliges Drücken der [ESC]-Taste unterbrochen und durch die Leertaste wieder fortgesetzt werden. Zweimaliges Drücken von [ESC] bricht die Ausgabe der Programme ab. Programme können in vorher definierten Bildschirmteilbereichen (WINDOW) aufgelistet werden, um z.B. Fehler suchen zu können, ohne den anderen Bildschirminhalt zu überschreiben. Die Auflistung vom Beginn des Programms bis zu einer bestimmten Zeile oder von einer bestimmten Zeile bis zum Programmende kann auch unter Weglassen der ersten bzw. letzten Zeilenzahl des <Zeilenbereichs> veranlaßt werden, z.B.

LIST-2ØØ oder LIST 3Ø-

L O A D

L O A D <Dateiname>[<Adressenausdruck>]

L O A D "discfile.xyz",&2AF8

KOMMANDO: Liest ein BASIC-Programm von der Diskette in den Speicher. Sind bereits andere Programme geladen, so werden diese hierdurch überschrieben. Durch Angabe des <Adressenausdrucks> wird eine Binärdatei an die entsprechende Stelle des Speichers geladen.
Ein geschütztes BASIC-Programm kann unter Verwendung der LOAD-Steuerung **nicht** geladen werden, da es sofort vom Speicher gelöscht wird; anstelle dessen sind die Steuerungen RUN bzw.CHAIN

einzusetzen.

Verwandte Befehle: CHAIN, CHAIN MERGE, MERGE, RUN, SAVE

L O C A T E

L O C A T E [#<Ein-/Ausgabegerät>,]<x-Koordinate>, <y-Koordinate>

```
10 MODE 1
20 FOR n=1 TO 20
30 LOCATE n,n
40 PRINT CHR$(143);"   Position"
50 LOCATE 30,n:PRINT n;",";n
60 NEXT
run
```

KOMMANDO: Positioniert den Cursor auf die angegebene x/y-Koordinate, die relativ zum Ursprung des Bildschirmteilbereichs (WINDOW) ausgewertet wird. Die linke obere Ecke eines Bildschirmteilbereichs hat die Koordinate 1/1. Ist kein <Ein-/Ausgabegerät> angegeben, so wird #Ø angenommen.

Verwandte Befehle: WINDOW

L O G

L O G (<numerischer Ausdruck>)

```
P R I N T  L O G (9999)
9.21024037
```

FUNKTION: Berechnet den natürlichen Logarithmus des <numerischen Ausdrucks>, der größer als Null sein muß.

Verwandte Befehle: EXP, LOG 1Ø

L O G 1Ø

L O G 1Ø (numerischer Ausdruck)

```
P R I N T  L O G 1Ø (9999)
3.99995657
```

FUNKTION: Gibt den Wert des Logarithmus zur Basis 1Ø des <numerischen Ausdrucks> als reelle Zahl (mit Dezimalpunkt) zurück.

Verwandte Befehle: EXP, LOG

LOWER$

LOWER$(<Textausdruck>)

```
A$="SCHNEIDER":PRINT LOWER$(A$)
schneider
```

FUNKTION: Gibt den angegebenen <Textausdruck> von Großbuchstaben - oder gemischter Schreibweise umgewandelt in Kleinbuchstabenschreibweise zurück. Ein nützliches Hilfsmittel, um Eingaben auszuwerten, die in unterschiedlicher Schreibweise eingegeben sein können.

Verwandte Befehle: UPPER$

MASK

MASK [<Ganzer Ausdruck>][,<Erstpunkteinstellung>]

```
10 CLS:TAG
20 MASK 1:MOVE 0,250:DRAWR 240,0
30 PRINT "(binaer 00000001 in mask)"
40 MASK 3:MOVE 0,200:DRAWR 240,0
50 PRINT "(binaer 00000011 in mask)"
60 MASK 7:MOVE 0,150:DRAWR 240,0
70 PRINT "(binaer 00000111 in mask)"
80 MASK 15:MOVE 0,100:DRAWR 240,0
90 PRINT "(binaer 00001111 in mask)"
run
```

KOMMANDO: Einstellung der "Maske" bzw. der zur Auszeichnung von Linien zu verwendenden Schablone. Der Binärwert des <ganzzahligen Ausdrucks> im Bereich zwischen 0 und 255 setzt die Bits in jeder benachbarten Gruppe von 8 Pixel auf ON (1) bzw. OFF (0).

Die "Erstpunkteinstellung" bestimmt, ob der Erstpunkt der Zeile zu plotten (1) bzw. nicht zu plotten (0) ist.

Hierbei kann auf einen der Parameter verzichtet werden, nicht aber auf beide. Wird auf einen Parameter verzichtet, so wird diese bestimmte Einstellung nicht verändert.

Verwandte Befehle: DRAW, DRAWR, GRAPHICS PAPER, GRAPHICS PEN

M A X

M A X (<Liste von:<numerischen Ausdrücke>)

```
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> zurück.

Verwandte Befehle: MIN

M E M O R Y

M E M O R Y <Adressenausdruck>

M E M O R Y &2ØAA

KOMMANDO: Setzt die Obergrenze des BASIC zur Verfügung stehenden Speichers (siehe Beschreibung des Kommandos HIMEM). Um den Speicherbereich zu überprüfen verwendet man das FRE -Kommando.

Verwandte Befehle: FRE, HIMEM, SYMBOL, SYMBOL AFTER

M E R G E

M E R G E [<Dateiname>]

M E R G E "P L A N"

KOMMANDO: Mischt ein Programm von Diskette in das Programm im Speicher. Der Inhalt der Datei wird dem Programm im Speicher hinzugefügt. Falls ein Programm in das andere gemischt werden soll, ohne Zeilen des schon im Speicher befindlichen zu überschreiben, so sollte letzteres in einen Zeilenbereich umnummeriert (RENUM) werden, der über dem des einzulesenden liegt.

Alle Variablen, benutzerdefinierte Funktionen und offene Dateien werden gelöscht, ein ON ERROR GOTO wird abgeschaltet, ein RESTORE wird durchgeführt und die DEFINT, DEFREAL- und DEFSTR- Festlegungen verlieren ihre Wirkung. Geschützte Dateien können nicht mit dem MERGE-Kommando gelesen werden.

Verwandte Befehle: CHAIN, CHAIN MERGE, LOAD

M I D $ (Funktion)

M I D $ (<Textausdruck>, <Startposition>[,<Unterlänge des Textausdruckes>)

```
10 MODE 1:ZONE 3
20 a$="ENCYCLOPAEDIA"
30 PRINT "Zeige mir wie man buchstabiert."
40 PRINT"OK ....";a$
50 FOR n=1 TO LEN(A$)
60 PRINT MID$(a$,n,1)
70 FOR t = 1 TO 700:NEXT t,n
80 PRINT:PRINT
90 INPUT "Gib mir ein anderes Wort ein.";a$
100 GOTO 50
MID$
```

FUNKTION: Gibt ab der mit <Startposition> angegebenen Stelle aus dem <Textausdruck> den Text zurück. Mit dem Parameter <Unterlänge des Textausdruckes> wird die Anzahl der zu extrahierenden Zeichen angegeben (Standard: Bis zum Ende des <Textausdrucks>).

Ist die "Startposition" größer als die Gesamtlänge des "Textausdruckes" so wird eine Leerkette zurückgegeben. Der Bereich der "Startposition" liegt zwischen 1 und 255. Der Bereich der "Länge der Unterkette" liegt zwischen 0 und 255.

Verwandte Befehle: LEFT$, RIGHT$

M I D $ (Kommando)

M I D $ (<Variable>, <Einfügungsposition> < neue Textlänge> = "neuer Textausdruck]

M I D $ (< Variable>, <Einfügungsposition> ,<neue Textlänge>)="neuer Text"

```
10 a$ = "hallo"
20 MID$(a$,3,2) = "XX"
30 PRINT a$
run
haXXo
```

KOMMANDO: Fügt den "neuen Text" in die von der< Variablen > angegebenen Texte ein. Der Zeichenumfang wird mit der< neuen Textlänge > bestimmt und überschreibt den bisherigen Text beginnen mit der < Einfügungsposition>.

Sie haben sicher bemerkt, daß bei Verwendung von MID$ als KOMMANDO eine< Variable> wie z.B. a$ zu verwenden ist, und nicht nur eine Textkonstante wie z.B. "Hallo".

M I N

M I N (<Liste von <numerischen Ausdrücke>>)

```
PRINT MIN(3,6,2.999,8,9)
 2.999
```

FUNKTION: Gibt den kleinsten Wert aus der Liste der <numerischen Ausdrücke> zurück.

Verwandte Befehle: MAX

M O D

<Argument>MOD <Argument>

```
PRINT 1Ø MOD 3
  1
PRINT 1Ø MOD 5
  Ø
```

FUNKTION: Weist nach Division des ersten< Arguments> durch das zweite< Argument> den 'Rest' aus; unterdrückt dabei Dezimalstellen.

M O D E

M O D E <ganzzahliger Ausdruck>

```
10 m=1
20 PRINT "Dies ist MODE :";m
30 PRINT "Druecke eine beliebige Taste"
40 IF INKEY$ ="" THEN 40
50 m=m+1:IF m>2 THEN m=0
60 MODE m
70 GOTO 20
```

KOMMANDO: Ändert den Bildschirmmodus (0,1 bzw. 2).

Alle Text- und Graphik-Fenster werden auf den gesamten Bildschirm zurückgesetzt und Text und Graphik-Cursoren auf ihre jeweiligen Ursprünge zurückgeführt.

Verwandte Befehle: ORIGIN, WINDOW

MOVE

MOVE <X-Koordinate>, <Y-Koordinate>[,[<INK>][,<INKMODUS>]]

```
10 MODE 1:TAG
20 x=RND*800-100:y=RND*430
30 MOVE x,y
40 PRINT "Ich habe mich hierher bewegt"
50 GOTO 20
```

KOMMANDO: Positioniert den Graphik-Cursor auf die angegebene absolute x/y-Koordinate. Der Parameter für INK kann hierbei dafür eingesetzt werden, die Farbe des Graphik-Schreibstiftes innerhalb des Bereiches zwischen 0 und 15 zu ändern (siehe INK).

Durch Wahl der Betriebsart für <INK> können Sie bestimmen, mit welcher der bereits auf dem Graphikschirm befindlichen Farbe weitergearbeitet werden soll. Die 4 Betriebsarten hierzu lauten:

0: Normal
1: XOR (ausschließlich ODER)
2: UND
3: ODER

Verwandte Befehle: MOVER, ORIGIN, XPOS, YPOS

M O V E R

M O V E R <x-Versatz>,<y-Versatz>[,[<INK>][, <INKMODUS>]]

```
10 MODE 1:TAG:MOVE 0,16
20 PRINT"Das Leben geht";
30 FOR n=1 TO 10
40 MOVER -32,16
50 PRINT"rauf";:NEXT:PRINT "und";
60 FOR n=1 TO 10
70 MOVER -64,-16
80 PRINT "runter";:NEXT
run
```

KOMMANDO: Positioniert den Graphik-Cursor von seiner augenblicklichen Position auf eine x/y-Koordinate, die sich aus dem relativen x/y-Versatz ergibt.

Der Parameter < I N K> bestimmt hierbei die Farbe innerhalb der Graphikdarstellung (Werte 0 bis 15)

Durch Wahl der Betriebsart für < I N K > können Sie bestimmen, mit welcher der bereits auf dem Graphikschirm befindlichen Farbe weitergearbeitet werden soll. Die 4 Betriebsarten hierzu lauten:

0: Normal
1: XOR
2: UND
3: ODER

Verwandte Befehle: MOVE, ORIGIN, XPOS, YPOS

N E W

N E W

KOMMANDO: Löscht das sich im Speicher befindende Programm und seine Variablen. Das Kommando entspricht fast einem ganzen Rücksetzen des Rechners oder dem Ausschalten. Es ist deshalb mit Vorsicht zu benützen. Die Tastenfunktionen bleiben hierbei erhalten, ohne Veränderung der Anzeigewerte wie z.B. MODE, PEN, PAPER, INK usw. Der Bildschirm wird nicht gelöscht.

N E X T

N E X T [<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 "PEN ";a;" BORDER ";b"
60 FOR c=1 TO 500
70 NEXT c,b,a
run
```

KOMMANDO: Legt das Ende einer FOR-Schleife fest. Das NEXT-Kommando kann ohne oder mit dem Variablennamen angegeben werden, der auf das entsprechende FOR-Kommando verweist.

Aus dem obigen Beispiel ist zu entnehmen, daß die <Liste der Variablen> in umgekehrter Reihenfolge zu ihren entsprechenden FOR-Befehlen eingegeben werden muß. Dadurch wird erreicht, daß sich 'verschachtelte' Schleifen nicht überlappen.

Verwandte Befehle: FOR, RESUME, STEP, TO

N O T

N O T <Argument>

```
If NOT "Hans"<"Uwe" then print "richtig" else print "falsch"
Falsch
If NOT "Hund">"Maus" then print "richtig" else print "falsch"
Richtig
....
Print Not -1
0
Print Not 0
-1
```

FUNKTION: Führt bitweise Vorgänge bei ganzen Zahlen aus, wobei jedes Bit im Argument umgekehrt wird.

Bezüglich weiterer Informationen zur Logik wird auf Absatz 2 des Kapitels "Wenn Sie Zeit haben" verwiesen.

Verwandte Befehle: AND, OR, XOR,

ON BREAK CONT

ON BREAK CONT

```
10 ON BREAK CONT
20 PRINT "Dieses Programm wird fortgesetzt,wenn Sie versuchen,"
30 PRINT "es mit Hilfe der -ESC- Taste zu stoppen."
40 FOR t = 1 TO 1000:NEXT:GOTO 20
run
```

KOMMANDO: Löscht die Funktion der Taste [ESC] zur Unterbrechung des Programms und setzt stattdessen die Ausführung fort (CONT). Bei diesem Befehl ist darauf zu achten, daß das Programm weiterläuft, bis der Computer völlig zurückgestellt ist; demzufolge ist ein solches Programm abzuspeichern, bevor mit RUN begonnen wird.

ON BREAK CONT läßt sich innerhalb eines Programms durch ON BREAK STOP außer Betrieb setzen.

Verwandte Befehle: ON BREAK GOSUB, ON BREAK STOP

ON BREAK GOSUB

ON BREAK GOSUB <Zeilennummer>

```
10 ON BREAK GOSUB 40
20 PRINT "Programm arbeitet"
30 GOTO 20
40 CLS
50 PRINT "Zweimal - ESC - ruft das Unterprogramm auf"
60 FOR t = 1 TO 2000:NEXT
70 RETURN
run
```

KOMMANDO: Springt in ein Unterprogramm auf <Zeilennummer>, wenn zweimal die [ESC]-Taste 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 "Druecke -ESC- ";
50 PRINT "zweimal druecken ruft GOSUB - Routine auf"
60 FOR t = 1 TO 2000:NEXT
70 ON BREAK STOP
80 RETURN
run
```

KOMMANDO: Löscht die Steuerung ON BREAK CONT und ON BREAK GOSUB, so daß zweimaliges Drücken der Taste [ESC] zur Unterbrechung des Programms führen. Innerhalb des obengenannten Beispiels wird der Befehl ON BREAK GOSUB nur einmal ausgeführt, da er durch Zeile 65 im Unterprogramm ON BREAK STOP 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 festgestellt wird, "
30 PRINT "dann liste das Programm auf."
40 FOR t=1 TO 4000:NEXT
50 GOTO 100
60 PRINT "Der Fehler wurde festgestellt in Zeile: ";
70 PRINT ERL:PRINT:LIST
run
```

KOMMANDO: Springt beim Auftreten eines Programmfehlers in die entsprechende Zeile.

Der Befehl ON ERROR GOTO Ø unterbricht die Fehlersuche und führt zur normalen Fehlerbearbeitung durch BASIC.

Verwandte Befehle: DERR, ERL, ERR, ERROR, RESUME

ON GOSUB
ON GOTO

ON <ganzzahliger Ausdruck>GOSUB<Liste von Zeilennummern>>
ON <ganzzahliger Ausdruck>GOTO<Liste von Zeilennummern>>

```
10 ON tag GOSUB 100,200,300,400,500
10 ON rate GOTO 1000,2000,3000,4000
```

KOMMANDO: Springt in Abhängigkeit vom <ganzzahligen Ausruck> in ein Unterprogramm (GOSUB) oder zu einer Zeile im Programm (GOTO). Falls der Wert des <ganzzahligen Ausdrucks> 1 ist, wird zur ersten Zeilennummer, bei 2 zur zweiten usw. gesprungen. In obigem Beispiel wird bei tag = 1 zum Unterprogramm bei Zeile 100, bei tag = 2 zum Unterprogramm bei Zeile 200 usw. gesprungen.

Der Wert des <ganzzahligen Ausdrucks> muß eine positive Zahl ohne Dezimalstellen im Bereich von 0 bis 255 sein.

Verwandte Befehle: GOTO, GOSUB, RETURN

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: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 an der angegebenen Zeile, falls bei der Tonausgabe in der Warteschlange des angegebenen <Kanals> Platz frei ist. <Kanal> ist ein <ganzzahliger Ausdruck> mit folgenden Werten und Bedingungen:

1 für Kanal A
2 für Kanal B
4 für Kanal C

Verwandte Befehle: SOUND, SQ

Weitere Informationen bezüglich der Tonfolgen sind in Teil 2 des Kapitels "Wenn Sie Zeit haben..." enthalten.

O P E N I N

O P E N I N <Dateiname>

```
10 REM Eroeffnet eine Eingabe-Datei von der Diskette
20 OPENIN "Dateinam":INPUT #9,a,a$
30 CLOSEIN:PRINT "Die 2 Werte sind:"
40 PRINT:PRINT a,a$
run
```

KOMMANDO: Innerhalb eines laufenden Programmes wird eine Eingangsdtei von der Diskette eröffnet.
Bei der Eingangsdatei zu OPEN muß es sich um eine ASCII-Datei handeln.

Das obige Beispiel trifft nur dann zu, wenn Sie die im nächsten Beispiel genannte Datei aufgebaut haben (unter OPENOUT).

Verwandte Befehle: CLOSEIN, EOF

O P E N O U T

O P E N O U T <Dateiname>

```
10 REM Eroeffnet eine Ausgabe-Datei auf  Diskette
20 INPUT "Geben Sie eine Zahl ein : ";a
30 INPUT "Geben Sie einen Text ein : ";a$
40 OPENOUT "Dateiname"    .
50 WRITE #9,a,a$
60 CLOSEOUT:PRINT "Die Datei ist auf Diskette gesichert"
run
```

KOMMANDO: Eröffnet eine Ausgangsdatei auf Diskette.

Verwandte Befehle: CLOSEOUT

O R

<Argument> OR <Argument>

```
If "Hans"<"Uwe" OR "Hund">"Maus" then print "richtig" else print "falsch"
Richtig
If "Uwe"<"Hans" OR "Maus">"Hund" then print "richtig" else print "falsch"
Falsch
If "Hans"<"Uwe" OR "Maus">"Hund" then print "richtig" else print "falsch"
Richtig
....
Print 1 OR 1
1
Print 0 OR 0
0
```

FUNKTION: Ermöglicht eine bitweise Bool'sche Rechenoperation für ganze Zahlen. Als Ergebnis erscheint 1, außer wenn beide Argumentbits auf Null stehen.

Verwandte Befehle: AND, NOT, XOR

O R I G I N

O R I G I N <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 Grafik Fenster ";
70 FRAME:NEXT:GOTO 40
run
```

KOMMANDO: Setzt den Startpunkt des Graphik-Cursors Ø,Ø auf den durch die x und y-Koordinaten festgelegte Position.

Die Bereiche für den Graphik-Bildschirm (WINDOW) können auch durch Angabe der letzten vier freien Parameter bestimmt werden. Liegen die für den Graphik-Bildschirmbereich angegebenen Koordinaten-Punkte außerhalb des Bildbereichs, so wird der Bildschirm als Begrenzung gewählt.

Verwandte Befehle: CLG

OUT

OUT <Interfaceadresse>,<ganzzahliger Ausdruck>

OUT &F8F4,10

KOMMANDO: Schickt den Wert des <ganzzahligen Ausdrucks>, der zwischen 0 und 255 liegen muß, zur Ausgabe an das mit seiner Adresse angesprochene Interface.

Verwandte Befehle: INP, WAIT

PAPER

PAPER [#<Ein-Ausgabegerät>,]<Farbstift>

```
10 MODE 0
20 FOR p=0 TO 15
30 PAPER p:CLS
40 PEN 15-p
50 LOCATE 6,12:PRINT "paper"p
60 FOR t=1 TO 500:NEXT t
70 NEXT p
run
```

KOMMANDO: Legt fest, welcher Farbstift zum Ausfüllen des beschreibbaren Bildschirmbereichs verwendet werden soll. Wird ein Zeichen auf den Bildschirm geschrieben, so wird der zum Zeichen gehörende Hintergrund mit der dem <Farbstift> (INK) zugeordneten Farbe des PAPER-Kommandos gefüllt, es sei denn, es wurde im 'Transparentmode' ausgegeben.

Die Anzahl der zu verwendenden <Farbstifte> hängt vom ausgewählten Bildschirmmodus ab. Falls der angegebene <Farbstift> nicht zulässig ist, wird ein beliebiger verwendbarer ausgewählt.

Verwandte Befehle: GRAPHICS PAPER, INK, PEN

P E E K

P E E K (<Adressenausdruck>)

```
10 MODE 2
20 INK 1,0:INK 0,12:BORDER 12
30 INPUT "Startadresse des Speicherbereichs";anfang
40 INPUT "Endadresse des Speicherbereichs";ende
50 FOR n = anfang TO ende
60 value$ = HEX$(PEEK(n),2)
70 PRINT value$
80 PRINT "an der Adresse ";HEX$(n,4)
90 NEXT
run
```

FUNKTION: Gibt den Inhalt der angegebenen Speicheradresse zurück. Mit obigem Beispiel kann man sich den Inhalt des RAMs CPC664 anschauen. Damit ist es möglich, im RAM-Bereich zu lesen, der vom unteren (von &ØØØØ bis &3FFFF) und vom oberen (von &CØØØ bis &FFFF) Bereich des ROMs überlagert wird, nicht im ROM selbst.

Verwandte Befehle: POKE

P E N

P E N [#<Ein-/Ausgabegerät>] [,<Farbstift>] [,<Hintergrundmodus>]

```
10 MODE 0:PAPER 0:INK 0,13
20 FOR p=1 TO 15
30 PEN p:PRINT SPACE$ (47);"Farbstiftnummer ";p
40 FOR t=1 TO 500:NEXT t,p:GOTO 20
run
```

KOMMANDO: Ordnet dem Schreib- bzw. Zeichenstift den angegebenen <Farbstift> zu (innerhalb des Beriches von 0 bis 15). Ist kein <Ein-/Ausgabegerät> angegeben, so wird standardmäßig die #Ø, der Bildschirm angenommen. Der Parameter <Hintergrundmodus> kann entweder auf Transparent (1) bzw. nicht Transparent (0) gesetzt werden.

Hierbei kann auf einen der beiden Parameter verzichtet werden, nicht jedoch auf beide; wird ein Parameter ausgelassen, so wird die Einstellung nicht verändert.

Verwandte Befehle: PAPER

P I

P I

```
PRINT PI
 3.14159265
```

```
10 REM Perspektivisches Zeichnen
20 MODE 2
30 RAD
40 INK 1,0
50 INK 0,12
60 BORDER 9
70 FOR n=1 TO 200
80 ORIGIN 420,0
90 DRAW 0,200
100 REM Zeichne Winkel vom Fluchtpunkt
110 DRAW 30*n*SIN(n*PI/4),(SIN(PI/2))*n*SIN(n)
120 NEXT
130 MOVE 0,200
140 DRAWR 0,50
150 DRAWR 40,0
160 WINDOW 1,45,1,10
170 PRINT "Schreiben sie das Programm selbst weiter!"
run
```

FUNKTION: Gibt den Wert von PI zurück, dem Verhältnis zwischen Umfang und Durchmesser des Kreises, nämlich 3.141592653468251. Die Funktion wird in den meisten graphischen Programmen verwendet.

Verwandte Befehle: DEG, RAD

P L O T

P L O T <x-Koordinate>,<y-Koordinate>[,<Farbstift>]
[,Farbstiftmodus]

```
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 Einzelpunkt auf den Graphik-Bildschirm, an der absoluten, durch die x-, y-Koordinaten bestimmte Stellung. Die Farbe des Punktes kann hierbei angegeben werden (im Bereich zwischen 0 bis 15).

Der <Farbstiftmodus> legt fest, in welcher Form die Farbe mit der bereits auf dem Graphik-Bildschirm befindlichen zusammenwirkt. Die 4 Farb-Betriebsarten sind:

0: Normal
1: XOR (ausschließliches ODER)
2: UND
3: ODER

Verwandte Befehle: GRAPHICS PEN, PLOTR

P L O T R

P L O T R <x-Versatz>,<y-Versatz>[,<Farbstift>]
[,<Farbstiftmodus>]

```
10 REM Benutze die Cursor Tasten, um Linien zu zeichnen
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: Plottet jeden Einzelpunkt von der momentanen Cursor-Position bis zu der sich aus dem Versatz ergebenden relativen x/y-Koordinate. Zusätzlich kann über den Parameter <Farbstift> die Farbe bestimmt werden, in der der Punkt geplottet werden soll (im Bereich zwischen 0 und 15).

Der <Farbstiftmodus> legt fest, in welcher Form die Farbe mit der bereits auf dem Graphik-Bildschirm befindlichen zusammenwirkt. Die 4 Betriebsarten für Farbstift sind:

0: Normal
1: XOR (ausschließliches ODER)
2: UND
3: ODER

Verwandte Befehle: GRAPHICS PEN, PLOT

POKE

POKE <Adressenausdruck>,<ganzzahliger Ausdruck>

```
10 FOR m=49152 TO 65535
20 POKE m,100
30 NEXT
run
```

POKE &ØØFF,1Ø

KOMMANDO: Schreibt den <ganzzahligen Ausdruck>, dessen Wert zwischen 0 und 255 liegen muß, in die angegebene Speicherstelle. Bei der Verwendung des POKE-Kommandos ist Vorsicht geboten, da das System ggf. überschrieben werden kann.

Verwandte Befehle: PEEK

P O S

P O S (#<Ein-/Ausgabegerät>)

```
10 MODE 1:BORDER 0:LOCATE 8,2
20 PRINT"Benuetze die Cursor-Tasten links/rechts"
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,"Horizontal Position =";
100 PRINT #1,POS(#0):GOTO 50
run
```

FUNKTION: Gibt die augenblickliche Position des angegebenen <Ein-/Ausgabegerätes> zurück. Das <Ein-/Ausgabegerät> muß immer angegeben werden, sonst wird ein Fehler (Syntax error) gemeldet. Bildschirm: Es wird die momentane x-Position des Textcursors relativ zum angegebenen Bildschirmteilbereich (WINDOW) zurückgegeben. Dabei ist die linke obere Ecke des Bildschirmteilbereiches immer 1,1.

POS (#8): Es wird die momentane Druckposition des Schreibkopfes zurückgegeben, wobei 1 die erste Druckstelle innerhalb der Zeile bedeutet.

POS (#9) meldet die Anzahl der druckbaren Zeichen, die seit dem letzten [ENTER] -Befehl auf die Diskette geschrieben worden sind.
Druckbare Zeichen sind alle Zeichen, deren ASCII-Wert größer & 1F(31) sind.
Die Position untermittelbar nach dem [ENTER] -Befehl ist 1.

Verwandte Befehle: VPOS, WINDOW

PRINT

PRINT [#<Ein-/Ausgabegerät>][<Ausgabeliste>]

```
10 a$ = "Kurzer Text"
20 b$ = "Dies ist ein langer Text"
30 PRINT a$;b$
40 PRINT a$,b$
50 PRINT
60 PRINT b$;b$
70 PRINT b$,b$
run
```

KOMMANDO: Druckt die <Ausgabeliste> auf dem entsprechenden <Ein-/Ausgabegerät> (#0, Bildschirm, wenn kein <Ein-/Ausgabegerät> angegeben worden ist).

Hierbei ist darauf zu achten, daß bei Auftreten eines Semikolons ; BASIC zuerst prüft, ob die folgende Druckstelle innerhalb der gleichen Zeile untergebracht werden kann. Falls dies nicht der Fall ist, wird sie auf einer neuen Zeile gedruckt, unabhängig vom Semikolon.

Ferner ist zu beachten, daß in Fällen, in denen ein Komma , eingesetzt wird, um den Computer darauf hinzuweisen, daß die folgende Druckstelle der nächsten Druckzone gedruckt wird, BASIC zuerst prüft, um festzustellen, ob die vorhergehende Druckstelle nicht die Länge der laufenden Zone überschritten hat.

Ist dies der Fall, so wird die folgende Druckstelle innerhalb einer anderen Zone ausgedruckt.

PRINT SPC

PRINT TAB

PRINT (#<Ein-/Ausgabegerät>, <Ausgabeliste> ;
SPC (<ganzzahliger Ausdruck>) "Ausgabeliste"

PRINT (#<Ein-/Ausgabegerät>, <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 (<ganzzahliger Ausdruck>) führt zur Ausgabe von Leerstellen entsprechend der durch den <ganzzahligen Ausdruck> angegebenen Anzahl. Ist die Anzahl kleiner Null, so erfolgt keine Ausgabe, ist sie größer als die zulässige Breite des Ausgabegerätes, so wird sie auf die maximal mögliche Anzahl gekürzt. **SPC** muß nicht durch ein <Trennzeichen> abgeschlossen werden, da immer ein ; angenommen wird.

TAB (<ganzzahliger Ausdruck>) gibt Leerzeichen bis zu der durch den <ganzzahligen Ausdruck> angegebenen Ausgabeposition (Spalte innerhalb einer Zeile). Ist der <ganzzahlige Ausdruck> negativ, wird 1 angenommen, ist er größer als die maximal mögliche Breite des Ausgabegeräts, wo wird wie bei **SPC** verkleinert. Ist die gewünschte Ausgabeposition größer als die augenblickliche, so werden bis zur neuen Leerzeichen ausgegeben, ist sie kleiner, wo wird bis zum Zeilenanfang zurückgegangen und dann werden von dort aus Leerzeichen ausgegeben, bis die gewünschte Position erreicht ist. Als <Trennzeichen> wird immer ein ; angenommen.

P R I N T U S I N G "<FORMAT>"

Für numerische Ausgaben:

Formatzeichen	Beschreibung	Beispiel
#	Jedes # ergibt eine Ausgabestelle	####
.	Position des Dezimalpunkts	#.#
+	Gibt für positive Zahlen das Vorzeichen mit aus	
	entweder zu Beginn	+##
	oder am Ende	##+

Formatzeichen	Beschreibung	Beispiel
-	Gibt für negative Zahlen das Vorzeichen am Ende aus	##-
**	Maximal 2 führende Druckstellen werden mit * statt mit Leerzeichen ausgegeben	**###.##
$$	Vor dem ersten Zeichen wird ein $ ausgegeben	$$##.##
£ £	Vor dem ersten Zeichen wir ein £ ausgegeben	£ £ ##.##
**£	Kombination aus ** und £ £	**£#.##
**$	Kombination aus ** und $$	**$#.##
,	Alle drei Stellen vor dem Dezimalpunkt wird eine Zahlengruppe durch ein Komma getrennt	#,###.#
↑ ↑ ↑ ↑	Ausgabe in Exponentialformat. Dabei wird die Zahl so ausgerichtet, daß die erste Stelle nie Null ist.	#.## ↑ ↑ ↑ ↑

Für Textausgaben:

!	Nur das erste Zeichen des Textes wird ausgegeben	!
\<Leerzeichen>\	Soviele Zeichen des Textes wie Leerzeichen angegeben sind, werden ausgegeben	\ \
&	Der gesamte Textausdruck wird ausgegeben	&

Die Musterschablone für eine Zahl darf 20 Stellen nicht überschreiten. Die Zahlen werden auf die Anzahl der gedruckten Stellen abgerundet.

Ist die Formatschablone für den Eingabeausdruck zu klein, z.B.:

```
PRINT USING "####"; 12345678
```

... so wird das Druckergebnis **nicht** auf die Schablone hin abgekürzt, sondern stattdessen voll gedruckt, mit vorangestelltem %Zeichen zur Angabe des "Formatfehlers".

```
10 CLS:a$="abcdefghijklmnopqrst"
20 PRINT "Eingabezeichen = ";a$
30 PRINT:PRINT "! Spezifikation = ";
40 PRINT USING "!";a$
50 PRINT:PRINT "² Leerstellen² Spezifikation = ";
60 PRINT USING "²        ²";a$
70 PRINT:PRINT "& Spezifikation = ";
80 PRINT USING "&";a$
90 GOTO 90
run
```

Die Schablone für einen Textausdruck darf 255 Stellen nicht überschreiten.

Sowohl die numerischen als auch die Textformat-Schablonen können durch Variable für Zeichenreihen dargestellt werden, z.B.:

```
10 a$="######,.##"
20 b$="!"
30 PRINT USING a$;12345.6789;
40 PRINT USING b$;"DMARK
run
```

Weitere Informationen in Bezug auf die Druckformatierung sind in Teil 2 des Kapitels "Wenn Sie Zeit haben..." enthalten.

Verwandte Befehle: SPC, TAB, USING

R A D

R A D

R A D

KOMMANDO: Schaltet auf Bogenmaßauswertungs-Modus um. Die Umschaltung über BASIC erfolgt dann, wenn der Computer eingeschaltet bzw. zurückgestellt wird, oder wenn die Kommandos NEW, CLEAR bzw. LOAD, RUN usw. ausgegeben werden.

Verwandte Befehle: ATN, COS, DEG, SIN, TAN

R A N D O M I Z E

R A N D O M I Z E [<numerischer Ausdruck>]

```
RANDOMIZE 123.456
PRINT RND
 Ø. 258852139
```

KOMMANDO: Setzt den Anfangswert für den Zufallszahlengenerator. In BASIC werden Zufallszahlen in Abhängigkeit von einem gegebenen Startwert erzeugt. Die Zahlenfolge ist immer dieselbe. Mit dem RANDOMIZE-Kommando wird entweder ein neuer Startwert mit dem <numerischen Ausdruck> direkt gesetzt oder die Eingabe des Startwertes erzeugt eine Zufallszahlenfolge, die fast unwiederholbar ist.

Verwandte Befehle: RND

R E A D

R E A D <Liste von:<Variablen>>

```
10 FOR n=1 TO 8
20 READ a$,c
30 PRINT a$;" ";:SOUND 1,c:NEXT
40 DATA Sie,478,hoeren,426,die,379,8,358
50 DATA Noten,319,der,284,Ton,253,leiter,239
run
```

KOMMANDO: Liest Daten aus DATA-Anweisungen entsprechend der angegebenen <Liste von <Variablen>>. Das RESTORE-Kommando setzt den Zeiger für die zu lesenden Daten wieder auf den ersten DATA-Wert.

Verwandte Befehle: DATA, RESTORE

RELEASE

RELEASE <Kanäle>

```
10 SOUND 65,1000,100
20 PRINT "Druecke `R` um einen Ton zu erzeugen"
30 IF INKEY(50) = -1 THEN 30
40 RELEASE 1
run
```

KOMMANDO: Hebt den eventuellen Wartezustand der angegebenen Ton-<Kanäle> auf. Der Wert, um die <Kanäle> ansprechen zu können, ist Bitabhängig:

1: Freigabe Kanal A
2: Freigabe Kanal B
3: Freigabe Kanal A und B
4: Freigabe Kanal C
5: Freigabe Kanal A und C
6: Freigabe Kanal B und C
7: Freigabe Kanal A, B und C

Verwandte Befehle: SOUND

REM

REM <Rest der Zeile>

```
10 REM Der hier angegebene Text dient der
20 REM Programmdokumentation
```

KOMMANDO: Fügt Kommentarzeilen in das Programm ein. Jedes nach dem

REM-Kommando angegebene Zeichen wird als Kommentar betrachtet (auch der :). Ein einfaches Anführungszeichen (') in einer Zeile hat die Bedeutung: REM, es sei denn, es ist in einem <Textausdruck> oder einer DATA-Anweisung angegeben.

REMAIN

REMAIN (<ganzzahliger Ausdruck>)

```
10 AFTER 500,1 GOSUB 40
20 AFTER 100,2 GOSUB 50
30 PRINT "Das Programm laeuft":GOTO 30
40 REM Diese Gosub Routine wird nur bei der Unterbrechung aufgeru
50 PRINT:PRINT"Timer 1 ";
60 PRINT"Unterbrechung durch REMAIN"
70 PRINT"Verbrauchte Zeit "
80 PRINT REMAIN(1)
run
```

FUNKTION: Gibt die verbleibende Restzeit des durch den <ganzzahligen Ausdruck> (zwischen 0 und 3) angesprochenen Weckers zurück und schaltet ihn zusätzlich ab. Falls der angesprochene Wecker nicht eingeschaltet war, wird 0 zurückgegeben.

Weitere Informationen bezüglich der Unterbrechungen sind in Teil 2 des Kapitels "Wenn Sie Zeit haben..." enthalten.

Verwandte Befehle: AFTER, EVERY

RENUM

RENUM [<neue Zeilennummer>][,[<alte Zeilennummer>] [,<Schrittweite>]]

```
10 CLS
20 REM Dies ist die Zeile 123
30 REM Dies ist die Zeile 124
40 REM Dies ist die Zeile 125
RENUM 123,20,1
LIST
```

KOMMANDO: Änderung der Numerierung der Programmzeilen (RENUM)

Der Parameter <alte Zeilennummer> nennt die laufende bestehende Zeilennummer, bei der die Neunumerierung zu beginnen hat. Fehlt die alte Zeilennummer, so beginnt die Neunumierung ab Beginn des Programms.

Die neue<Zeilennummer>gibt den Startwert der neuen Numerierung an. Fehlt die<neue Zeilennummer> so beginnt das neu numerierte Programm mit Zeile 10.

Die Schrittweite bestimmt die Abstände zwischen den Zeilennummern. Fehlt die Schrittweite , so gilt als Standardwert für den numerischen Schritt 10.

RENUM aktualisiert alle GOSUB, GOTO- und andere Zeilenkommandos. Demgegenüber werden jedoch Zeilennummer-Referenzen innerhalb von Kettenausdrücken wie z.B. bei KEY-Kommandos, nicht verändert; ferner bleiben unverändert Zeilenreferenzen innerhalb von REM-Angaben bzw. der Zeilennummernausdruck innerhalb einer CHAIN- bzw. CHAIN MERGE-Steuerung.

Zeilennummern können innerhalb des Bereiches von 1 bis 65535 liegen.

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 restored Daten kann man
50 DATA immer wieder lesen
run
```

KOMMANDO: Setzt den Zeiger für die zu lesenden Data-Anweisungen auf die erste im Programm auftretende oder auf die mit der <Zeilennummer> angegebene.

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 "Springe nach einem Fehler hierher"
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 angebenen <Zeilennummer> fortzusetzen, falls es aufgrund eines ON ERROR GOTO-Sprungs unterbrochen wurde. Ist die Zeilennummer zur Wiederaufnahme (RESUME) hier nicht angegeben, dann beginnt das Programm erneut ab der gleichen Zeile, in der der Fehler zum erstenmal auftrat.

```
7Ø 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 die -ENTER-  Taste"
30 INPUT "1";a
40 INPUT "2";a
50 inpot "3";a REM Syntax Fehler
60 INPUT "4";a
70 INPUT "5";a
80 END
90 PRINT "Fehler Nr.: ";ERR;"in Zeile ";ERL
100 RESUME NEXT
run
```

KOMMANDO: Veranlaßt das Programm, normal fortzufahren, falls es aufgrund eines ON ERROR GOTO-Sprungs wegen eines Fehlers unterbrochen wurde.

RESUME NEXT übernimmt die Wiederaufnahme der Ausführung an der Zeile, in der der Fehler erstmals erfaßt wurde.

Verwandte Befehle: DERR, ERL, ERR, ERROR, ON ERROR GOTO, RESUME

RETURN

RETURN

```
10 GOSUB 50:PRINT"Nach dem GOSUB Befehl":END
20 FOR n = 1 TO 20
30 PRINT"Unterprogramm"
40 NEXT:PRINT
50 FOR n = 1 TO 20
60 PRINT"Unterprogramm"
70 NEXT:PRINT
80 RETURN
run
```

KOMMANDO: Rücksprung aus einem Unterprogramm. BASIC kehrt aus einem Unterprogramm zu dem Befehl zurück, der dem GOSUB-Befehl folgt und macht dort weiter.

Verwandte Befehle: GOSUB, ON x GOSUB, ON SQ GOSUB, AFTER n GOSUB, EVERY n GOSUB, ON BREAK GOSUB

RIGHT$

RIGHT$ (<Textausdruck>,<ganzzahliger Ausdruck>)

```
10 CLS
20 a$ = "Schneider"
30 b$ = RIGHT$(a$,3)
40 PRINT b$
run
[Bildschirm wird gelöscht]
DER
Ready
```

FUNKTION: Gibt die mit dem <ganzzahligen Ausdruck> angegebene Anzahl der Zeichen vom Ende des <Textausdrucks> her gerechnet (zwischen Ø und 255) zurück. Falls der <Textausdruck> kürzer ist, als die gewünschte Anzahl, so wird der gesamte <Textausdruck> zurückgegeben.

Verwandte Befehle: MID$, LEFT$

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 zurück, wenn der numerische Ausdruck einen positiven Wert besitzt bzw. nicht angegeben hat.

Besitzt der <numerische Ausdruck> den Wert Null, so liefert RND die gleiche Zufallszahl zurück, die beim letztenmal erzeugt wurde.

Ist der <numerische Ausdruck> negativ, so wird eine neue Zufallszahl gestartet.

Verwandte Befehle: RANDOMIZE

ROUND

ROUND (<numerischer Ausdruck>[,<ganzzahliger Ausdruck>])

```
10 FOR n=4 TO -4 STEP -1
20 PRINT ROUND (1234.5678,n)
30 PRINT "Die gerundete Stellenzahl ist : ";n
40 NEXT
run
```

KOMMANDO: Gibt den gerundeten Wert des <numerischen Ausdrucks> zurück. Dabei wird entweder nach Dezimalstellen oder nach Potenzen von 10 gerundet, je nach dem <ganzzahligen Ausdruck>. Falls der <ganzzahlige Ausdruck> negativ ist, wird der <numerische Ausdruck> so zu einem ganzzahligen Wert gerundet, daß so viele Stellen vor dem Dezimalpunkt durch eine Null ersetzt werden, wie der Absolutwert des <ganzzahligen Ausdrucks> angibt. Die Stellen hinter dem Dezimalpunkt entfallen.

Verwandte Befehle: ABS, CINT, FIX, INT

RUN

RUN <Textausdruck>

RUN "Welcome 1.bas"

KOMMANDO: Lädt ein BASIC- bzw. Binärprogramm von der Diskette und startet es. Hierbei wird jedes vorher geladene BASIC-Programm aus dem Speicher gelöscht.

Geschützte BASIC-Programme können auf diese Art direkt bearbeitet werden.

Verwandte Befehle: LOAD

RUN

RUN [<Zeilennummer>]

RUN 200

KOMMANDO: Beginnt mit der Ausführung des laufenden BASIC-Programms, ausgehend von der angegebenen <Zeilennummer> bzw. vom Beginn des Programms, wenn kein Parameter angegeben ist.

Mit RUN wird der Wert aller laufenden Programm-Variablen auf Null gesetzt.

Geschützte Programme können hierbei in dieser Form nach dem Laden **nicht** bearbeitet werden.

Verwandte Befehle: CONT, END, STOP

SAVE

SAVE <Dateiname>[,<Dateityp>][,<Binärdateiparameter>]

SAVE "Programm.xyz"

... sichert das Programm als normale, ungeschützte BASIC-Datei.

SAVE "Programm.xyz", P

... sichert das Programm als geschützte BASIC-Datei.

SAVE "Programm.xyz", A

... sichert das Programm als ASCII-Datei.

```
SAVE "Programm.xyz",B,8000,3000,8001
```

... sichert das **Programm in der Binärbetriebsart.**
In diesem Beispiel wird der Bereich des Speichers mit Beginn auf Adresse 8000 geschützt, wobei die Länge der Datei 3000 Bytes beträgt; hierbei liegt der Eingangspunkt der Adresse auf 8001.

KOMMANDO: Schreibt das im Speicher befindliche Programm auf die Diskette. Eine Binärdatei belegt eine Speicherfläche und wird auf Diskette abgespeichert. Die Binär-Parameter sind hierbei:

<Start Adresse>,<Dateilänge>,[,<Eingangspunkt>]

Der Bildschirmspeicher kann als Binärdatei abgespeichert werden. Dieser Vorgang ist als Zwischenspeicherung (screen dump) bekannt und kann unter Einsatz folgender Steuerung ausgeführt werden:

```
SAVE "Bild",B,&C000,&4000
```

Mit anschließender Übernahme auf den Bildschirm:

```
LOAD "Bild"
```

Verwandte Befehle: CHAIN, CHAIN MERGE, LOAD, MERGE, RUN

S G N

```
SGN (<numerischer Ausdruck>)
```

```
10 FOR n=200 TO -200 STEP -20
20 PRINT"SGN gibt den Wert ";:PRINT SGN(n);"zurueck wenn n = ";n
30 NEXT
run
```

FUNKTION: Gibt -1 zurück, falls der <numerische Ausdruck> kleiner 0 ist, 0, falls der <numerische Ausdruck> gleich 0 ist und 1, falls der <numerische Ausdruck> größer ist.

Verwandte Befehle: ABS

S I N

S I N (<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: Gibt den Wert des Sinus des <numerischen Ausdrucks> zurück, der als im Bogenmaß angegeben verstanden wird, solange nicht durch das DEG-Kommando auf Gradmaßinterpretation umgeschaltet wird.

Verwandte Befehle: ATN, COS, DEG, RAD, TAN

S O U N D

S O U N D <Kanal-Status>,<Tonperiode>[,<Dauer>
[,<Lautstärke>[,<Lautstärkenvariation>[,
<Tonvariation>[,<Geräuschfolge>]]]]]

```
10 FOR z=0 TO 4095
20 SOUND 1,z,1,12
30 NEXT
run
```

KOMMANDO: Erzeugt einen Ton. Der Befehl setzt sich aus folgenden Parametern zusammen:

K a n a l S t a t u s

Wertebereich 1 ... 255

Standardwert, wenn nicht belegt: keiner (Eingabe zwingend).

Mit dem CPC664 können gleichzeitig 3 verschiedene Töne gespielt werden, und zwar über die 3 vorhandenen SOUND-Kanäle A, B und C. Die Eingabe erfolgt als dezimale Ganzzahl, aber wenn die Dezimalzahl in eine 8 Bit Darstellung umgewandelt wird, aktivieren die gesetzten Bits folgende Kommandos:

DEZIMAL	BIT	KOMMANDO
1	0 LSB	sende SOUND nach Kanal A
2	1	sende SOUND nach Kanal B
4	2	sende SOUND nach Kanal C
8	3	Rendezvous mit Kanal A
16	4	Rendezvous mit Kanal B
32	5	Rendezvous mit Kanal C
64	6	hold
128	7 MSB	flush

LSB = niederwertigstes Bit (Least Significant Bit)
MSB = höherwertigstes Bit (Most Significant Bit)

Beispiele für ganzzahlige Eingabe:

2 = sende den folgenden SOUND nach Kanal B

5 = sende den folgenden SOUND nach Kanal A und C

98 = 64+32+2

= sende den folgenden SOUND nach Kanal B, Rendezvous mit Kanal C und halte.

Wichtig: Wenn ein Rendezvous auf zwei oder mehreren Kanälen arrangiert werden soll, muß der <Kanal Status> eines jeden Kanals zur richtigen Zeit markiert werden.

Beim Setzen eines Haltepunktes stoppt der Ton und die Tonschlange dahinter wird solange eingefroren, bis ein RELEASE-Kommando gegeben wird, oder sie mit einem späteren SOUND Kommando geleert wird (flush).

Wenn das Flush-Bit für einen Kanal gesetzt ist, wird der SOUND, der dabei mitgegeben wird, sofort ausgeführt. Die Tonschlange (Queue) bleibt dabei leer und der Kanalkopf inaktiv. Jegliche Töne, die zu diesem Zeitpunkt gerade gespielt werden, werden abgebrochen.

T o n P e r i o d e

Wertebereich 0 ... 4095

Standardwert, wenn nicht belegt: keiner (Eingabe zwingend).

Legt eine Periode und damit die Frequenz des Tons, der gespielt werden soll, fest. Die Frequenz bekommt man über folgende Formel: Frequenz = 125000/Periode.

Wenn Sie 0 angeben, wird keine Frequenz gesetzt. Das ist dann sinnvoll, wenn lediglich Geräusche erzeugt werden sollen.

D a u e r

Jeder ganzzahlige Wert im Bereich -32768 ... +32767 ist erlaubt.

Standardwert, wenn nicht belegt: 20.

Bei Werten größer als 0 bedeutet der eingegebene Wert die Dauer in 1/100stel Sekunden. Wenn der Wert 0 ist, wird die Dauer durch die Hüllkurve der Lautstärke gesteuert.

Wenn die Zahl negativ ist, wird die Lautstärken-Hüllkurve dem positivem Wert der Zahl entsprechend oft wiederholt.

L a u t s t ä r k e

Ein ganzzahliger Wert im Bereich 0 ... 15 (0 ... 7, wenn für die Lautstärke keine Hüllkurve angegeben ist).

Standardwert, wenn nicht belegt: 12 (4 ohne Lautstärken-Hüllkurve).

Das ist die Lautstärke am Anfang des Tons, die durch die Lautstärken-Hüllkurve verändert werden kann. 0 = keine Lautstärke.

L a u t s t ä r k e n - H ü l l k u r v e

Ein ganzzahliger Wert im Bereich 0 ... 15.

Standardwert, wenn nicht belegt: 0.

Der Wert spezifiziert eine vorgegebene Hüllkurve, die mit ENV definiert wird. Eine feste Definition besteht für Lautstärken-Hüllkurve 0. Diese kann nicht geändert werden und ist auf 2 Sekunden auf der <Lautstärke-Skala> festgelegt.

T o n - H ü l l k u r v e

Ein ganzzahliger Wert im Bereich 0 ... 15.

Standardwert, wenn nicht belegt: 0.

Der Wert spezifiziert eine vorgegebene Hüllkurve, die mit ENT definiert wird. Eine feste Definition besteht für Ton-Hüllkurve 0, die nicht geändert werden kann und auf die 'Ton-Periode' festgesetzt ist.

G e r ä u s c h - P e r i o d e

Ein ganzzahliger Wert im Bereich 0 ... 15.

Standardwert, wenn nicht belegt: 0.

Legt das Geräusch (noise) fest, das dem SOUND hinzugefügt wird. Wenn nichts oder der Wert 0 angegeben wird, dann wird kein Geräusch hinzugefügt. Es kann immer nur eine Geräusch-Periode definiert werden. Das heißt, daß alle Kanäle mit dem gleichen Geräusch versorgt werden.

Weitere Informationen bezüglich der Tonfolgen sind in Teil 2 des Kapitels "Wenn Sie Zeit haben ..." enthalten.

Verwandte Befehle: ENT, ENV, ON SQ GOSUB, RELEASE, SQ

SPACE$

SPACE$ (<ganzzahliger Ausdruck>)

```
10 MODE 1
20 PRINT"Fuege 9 Leerstellen zwischen mir";
30 PRINT SPACE$(9);
40 PRINT"und dir ein!"
run
```

FUNKTION: Gibt einen <Textausdruck> mit der durch den <ganzzahligen Ausdruck> angegebenen Anzahl von Leerzeichen zurück.

Verwandte Befehle: SPC, STRING$,TAB

SPC

(siehe PRINT SPC)

SPEED INK

SPEED INK <ganzzahliger Ausdruck>,<ganzzahliger Ausdruck>

```
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 Länge der Farbdauer bei Farbwechseln für

INK und BORDER. Das INK- und das BORDER-Kommando erlauben es, je zwei sich abwechselnde Farben anzugeben. Der erste <ganzzahlige Ausdruck> gibt die Dauer der ersten und der zweite <ganzzahlige Ausdruck> der zweiten Farbe an. Die Zeiten zwischen den Farbwechseln werden in Einheiten zu 0.02 Sekunden gemessen. Gehen Sie vorsichtig mit dem Auswählen von Farben und Farbwechseln um, da Sie hypnotische Effekte erzielen können.

Verwandte Befehle: BORDER, INK

SPEED KEY

SPEED KEY <Startverzögerung>,<Wiederholungs-periode>

```
10 CLS:FOR k=7 TO 1 STEP -2
20 PRINT "Gib deinen Namen ein und druecke dann -ENTER-"
30 SPEED KEY k,k
40 LINE INPUT a$:NEXT
50 PRINT "Du hast einen lustigen Namen"
run
```

KOMMANDO: Legt den Wert für die automatische Wiederholung eines Buchstabens fest, wenn Sie eine Taste länger gedrückt halten. Der Parameter <Startverzögerung> gibt die Zeit, in Einheiten von 0.02 (Fünfzigstel) Sekunden an, bevor die automatische Wiederholung beginnt. Der Parameter für die< Wiederholungsperiode> gibt das Intervall zwischen jeder auto repeat-Funktion einer Taste an.

SKEED KEY arbeitet nur für Tasten, die standardmäßig mit "auto repeat" ausgerüstet sind, bzw. die auf diese Funktion über die KEY-DEF-Steuerung besetzt wurden.

Sollen sehr kleine Werte für die <Startverzögerung> eingesetzt werden, so empfiehlt es sich, eine der Schreibtasten dahingehend vorzuprogrammieren, daß die Tastatur auf die Standardeinstellung SPEED KEY von 30,2 zurückgeschaltet wird. Dieses kann durch folgende Steuerung erfolgen:

KEY Ø, "SPEED KEY 3Ø,2" +CHR$(13)

...hierdurch wird SPEED KEY auf die Standardwerte zurückgesetzt, wenn die Ø-Taste auf der Tastatur gedrückt wird.

Verwandte Befehle: KEY DEF

SPEED WRITE

SPEED WRITE <ganzzahliger Ausdruck>

SPEED WRITE 1

KOMMANDO: Setzt die Schreibgeschwindigkeit für die Ausgabe auf Cassette (sofern angeschlossen). Es kann entweder mit 2000 Baud (<Ganzzahliger Ausdruck> = 1) oder 1000 Baud (<ganzzahliger Ausdruck> = 0) geschrieben werden. Wird eine Datei von Cassette gelesen, schaltet der CPC664 automatisch auf die für die Datei richtige Lesegeschwindigkeit um.

Werden Cassetten minderer Qualität verwendet, sollte zur Erhöhung der Datensicherheit mit 1000 Baud geschrieben werden.

Der SPEED WRITE-Befehl hat keine Auswirkung auf den Betrieb der Diskette.

Verwandte Befehle: OPENOUT, SAVE

SQ

SQ (<Kanal>)

```
10 SOUND 65,100,100
20 PRINT SQ(1)
run
```

FUNKTION: Gibt die Anzahl der freien Plätze in der Tonwarteschlange und den Status der Warteschlange für den angegebenen <Kanal> zurück. Dabei bedeutet 1 den Kanal A, 2 den Kanal B und 3 den Kanal C. Die Funktion gibt zurück, ob der angegebene <Kanal> aktiv ist und falls nicht, warum der eventuelle erste Eintrag der Warteschlange in einem Wartezustand ist. Die Rückgabe erfolgt in einzelnen Bits:

Bits 0,1,2: Anzahl der freien Einträge in der Warteschlange

Bits 3,4,5: ggf. den Rendezvousstatus des ersten Eintrags der Warteschlange (Warten auf anderen Kanal)

Bit 6: wird gesetzt, wenn die Warteschlange sich in einem Haltezustand befindet.

Bit 7: wird gesetzt, wenn der Kanal aktiv ist.

Weitere Informationen sind in Teil 2 des Kapitels "Wenn Sie Zeit haben..." enthalten.

Verwandte Befehle: ON SQ GOSUB, SOUND

S Q R

S Q R (<numerischer Ausdruck>)

```
PRINT SQR(9)
  3
```

FUNKTION: Gibt den Wert der Quadratwurzel aus dem <numerischen Ausdruck> zurück.

S T E P

(Siehe FOR)

S T O P

S T O P

```
10 FOR n=1 TO 30:PRINT n:NEXT
20 STOP
30 FOR n=31 TO 60:PRINT n:NEXT
run
```

KOMMANDO: Stoppt ein Programm und läßt BASIC in einem Unterbrechungszustand, so daß das Programm mit dem CONT-Befehl wieder fortgesetzt werden kann. Das Kommando kann dazu benutzt werden, um ein Programm zu Testzwecken an einer bestimmten Stelle zu unterbrechen.

Verwandte Befehle: CONT, END

S T R $

S T R $ (<numerischer Ausdruck>)

```
10 a=&FF : REM 255 hex
20 b=&X1111 : REM 15 binaer
30 c$=" ***"
40 PRINT c$+STR$(a+b)+c$
run
```

FUNKTION: Gibt den Dezimalwert des angegebenen <numerischen Ausdrucks> als Textausdruck in derselben Form zurück, wie er im PRINT-Kommando ausgegeben wird.

Verwandte Befehle: BIN$, DEC$, HE$, VAL

S T R I N G $

S T R I N G $ (<Länge>, <Textzeichen>)

```
PRINT STRING$ (40, "*"
****************************************
```

FUNKTION: Gibt das angegebene< Textzeichen> so oft zurück, wie der < ganzzahlige Ausdruck> angibt (im Bereich zwischen 0 und 255). Hierbei ist zu beachten, daß das obige Beispiel wie folgt eingegeben werden kann:

```
PRINT STRING$ (40, 42)
****************************************
```

...wobei sich das Textzeichen 42 auf den ASCII-Wert des Zeichens* bezieht, d.h. PRINT STRING$ (40, CHR$ (42)) entspricht.

Verwandte Befehle: SPACE$

S W A P

(Siehe WINDOW SWAP)

S Y M B O L

S Y M B O L <Zeichennummer>,<Liste von: <Beschreibungswerten>>

```
10 MODE 1:SYMBOL AFTER 105
20 Reihe1 = 255:REM binaer 11111111
30 Reihe2 = 129:REM binaer 10000001
40 Reihe3 = 189:REM binaer 10111101
50 Reihe4 = 153:REM binaer 10011001
60 Reihe5 = 153:REM binaer 10011001
70 Reihe6 = 189:REM binaer 10111101
80 Reihe7 = 129:REM binaer 10000001
90 Reihe8 = 255:REM binaer 11111111
100 PRINT "Zeile 110 belegt den Buchstaben `i`(105
) neu"
110 PRINT "druecke ein paarmal auf diese Taste und
 liste dann das Programm"
120 SYMBOL 105,Reihe1,Reihe2,Reihe3,Reihe4,Reihe5,
Reihe6,Reihe7,Reihe8
run
```

KOMMANDO: Belegt das durch <Zeilennummer> angesprochene Zeichen mit einem neuen Darstellungswert. Die Neubelegung muß vorher mit SYMBOL AFTER x ermöglicht werden, wobei x gleich bzw. unter der Zeichenzahl liegt, die neu definiert werden soll.

Anschließend wird die Steuerung SYMBOL ausgeführt, gefolgt durch die Zeichennummer x.

Unabhängig davon, ob der Wert von x ein Zeichen definiert, das sich direkt auf der Tastatur eingeben läßt, können die neu definierten Zeichen mit folgendem Befehl am Bildschirm ausgedruckt werden:

P R I N T C H R $ (x)

Nach dem SYMBOL x folgen bis zu 8 Parameter, die die 8 einzelnen Horizontalreihen des Zeichens, von oben beginnend, definieren. Jeder Parameter kann sich innerhalb des Bereiches von 0 bis 255 befinden. Die Binärdarstellung aller 8 Parameter bestimmt das Bild dieser einzelnen Reihe im abgeschlossenen Zeichen.

Handelt es sich beispielsweise bei dem ersten der 8 Parameter um 1, so weist die obere Zeichenreihe eine Binärdarstellung von 00000001 auf. Erscheint 1, so wird der Abschnitt des Zeichens in der Farbe PEN gedruckt; erscheint 0, so ist der Abschnitt des Zeichens nicht sichtbar, da ja in der Hintergrundfarbe (PAPER) dargestellt wird. Demzufolge besitzt die obere Reihe dieses neu definierten Zeichens einen Punkt in der oberen rechten Ecke. Bei Fortsetzung diesen Beispiels werden die 7 anderen

Parameter angegeben als 3, 7, 15, 31, 63, Ø, Ø - wobei sich für alle 8 Parameter folgende Binärdarstellung ergibt:

Parameter (Reihe) 1: 00000001 binär: (Dezimal 1)
Parameter (Reihe) 2: 00000011 binär: (Dezimal 3)
Parameter (Reihe) 3: 00000111 binär: (Dezimal 3)
Parameter (Reihe) 4: 00001111 binär: (Dezimal 15)
Parameter (Reihe) 5: 00011111 binär: (Dezimal 31)
Parameter (Reihe) 6: 00111111 binär: (Dezimal 63)
Parameter (Reihe) 7: 00000000 binär: (Dezimal 0)
Parameter (Reihe) 8: 00000000 binär: (Dezimal 0)

Ausgehend von der Binärdarstellung der obengenannten 8 Parameter muß es möglich sein, festzustellen, welche Form das neue Zeichen annehmen wird. Wir werden nunmehr diese Parameter der Zeichennummer 255 unter Einsatz folgender Steuerung zuordnen:

```
SYMBOL 255,1,3,7,15,31,63,Ø,Ø
```

Hierbei enthalten die beiden letzten Parameter (Wert Ø) keine Informationen, die wir benötigen, wir geben deshalb nur ein:

```
SYMBOL 255, 1,3,7,15,31,63
```

Sie haben auch die Möglichkeit, die Parameter in Binärform einzugeben, so daß keine Umsetzung des dezimal dargestellten Symbols erforderlich ist. Achten Sie auf den &X-Binärprefix! Z.B.:

```
SYMBOL 255,&XØØØØØØØ1, &XØØØØØØ11, &XØØØØØ111,
&XØØØØ1111, &XØØØ11111, &XØØ111111
```

Nunmehr zur Darstellung des Zeichens:

```
PRINT CHR$(255)
```

Werden die obengenannten Parameter einem schreibfähigen Zeichen auf der Tastatur zugeordnet, so erscheint das neue Zeichen immer dann, wenn die entsprechende Taste gedrückt wird bzw. wenn das vorhergehende Zeichen gedrückt worden wäre. Demzufolge weist BASIC dieses neue Zeichen nicht als unverständlich zurück, sondern betrachtet es als Äquivalent zu dem ursprünglichen Zeichen.

Verwandte Befehle: HIMEM, MEMORY, SYMBOL AFTER

SYMBOL AFTER

SYMBOL AFTER <Ganzzahliger Ausdruck>

```
10 CLS
20 SYMBOL AFTER 115
30 PRINT "Zeile 40 definiert die `s` - Taste um";
40 SYMBOL 115,0,56,64,64,48,8,8,112
50 PRINT"das Zeichen `s` ist jetzt "
60 PRINT "Mache diesen Vorgang rueckgaengig ";
70 PRINT "indem du SYMBOL AFTER 240 eingibst"
run
```

KOMMANDO: Setzt den Bereich, in dem benutzereigene Zeichen beschrieben werden können. Der Standardwert ist 240, so daß 16 Zeichen beschrieben werden können. Falls der <ganzzahlige Ausdruck> den Wert 32 hat, können alle Zeichen mit den Nummern 32 bis 255 umgesetzt werden.

SYMBOL AFTER 256 erlaubt keine Neudefinition der Zeichen.

Wird der SYMBOL-AFTER-Befehl ausgeführt, so werden alle benutzereigenen Zeichenbeschreibungen auf die Standardbelegung zurückgesetzt.

SYMBOL-AFTER arbeitet dann **nicht**, wenn der Aufruf erfolgt, nach dem Wert von HIMEM unter Verwendung des MEMORY-Befehls verändert wurde. Das gleiche gilt, wenn eine Datei durch OPENIN bzw. OPENOUT eröffnet wurde. Unter diesen Umständen erscheint "Improper argument", Fehler (5).

Weitere Informationen bezüglich der benutzereigenen Zeichen sind in Teil 2 des Kapitels "Wenn Sie Zeit haben..." enthalten.

Verwandte Befehle: HIMEM, MEMORY, SYMBOL

TAB

(Siehe PRINT TAB)

T A G

T A G [#<Ein-/Ausgabegerät>]

```
10 INPUT "Gib deinen Namen ein ";a$:CLS
20 PRINT "Dein Name laeuft jetzt ueber den Bildschirm ";a$
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: Ermöglicht mit dem PRINT-Kommando auf Graphikcursorpositionen auszugeben. Damit können Sie Texte und Symbole mit Graphik beliebig mischen. Das <Ein-/Ausgabegerät> ist standardmäßig #Ø, also der Bildschirm. Die linke obere Ecke des Textzeichens wird auf die Graphikcursorposition ausgegeben. Nicht abdruckbare Kontrollzeichen, wie z.B. neue Zeile (LINE FEED) werden als Sonderzeichen ausgegeben, falls hinter dem PRINT-Kommando kein ; steht.

Innerhalb der Standardausgabe (#Ø) schaltet BASIC TAG ab, wenn in die direkte Betriebsart zurückgeschaltet wird.

Verwandte Befehle: TAGOFF

T A G O F F

T A G O F F [#<Ein-/Ausgabegerät>]

```
10 MODE 2:TAG:REM Text auf Grafik - Cursor Position
20 Jahr = 1985:FOR x=1 TO 640 STEP 70
30 MOVE x,400:DRAWR 0,-350
40 jahr = jahr + 1:PRINT jahr;:NEXT
50 TAGOFF :REM Text auf Grafik - Cursor Position aus
60 LOCATE 34,25:PRINT "Jaehrliche Darstellung"
70 GOTO 70
run
```

KOMMANDO: Schaltet die PRINT-Kommando-Ausgabe auf Graphikcursorpositionen ab. Danach gibt das PRINT-Kommando ab der Textcursorposition weiter aus, die vor dem TAG-Kommando erreicht war.

Verwandte Befehle: TAG

TAN

TAN (<numerischer Ausdruck>)

```
PRINT TAN (45)
 1.61977519
```

FUNKTION: Gibt den Wert des TANgens des <numerischen Ausdrucks> zurück, der sich innerhalb des Bereiches -200000 bis +200000 befinden muß.

Hierbei kann DEG und RAD eingesetzt werden, um das Ergebnis der obigen Berechnung auf Bogen- bzw. Gradmaß umzurechnen (standardmäßig Bogenmaß).

Verwandte Befehle: ATN, COS, DEG, RAD, SIN

TEST

TEST (<x-Koordinante>, <y-Koordinate>)

```
10 CLS
20 PRINT "Du benutzt PEN - Nummer : ";
30 PRINT TEST (10,386)
40 PRINT "Aendere jetzt PEN und MODE "
50 PRINT ".... dann starte das Programm erneut."
run
```

FUNKTION: Gibt die Farbstiftnummer zurück, die an der angegebenen absoluten x/y-Koordinate 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 "pen ";TESTR(0,-16):NEXT
run
```

FUNKTION: Gibt die Farbstiftnummer zurück, die an der durch den x/y-Versatz zur augenblicklichen Graphikcursorposition relativen x/y-Koordinate 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:datum=INT(TIME/300)
60 WHILE stunde<13
70 WHILE minute<60
80 WHILE tick<60
90 tick=(INT(TIME/300)-datum)+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
```

KOMMANDO: Gibt die Zeit zurück, die seit dem Einschalten vergangen ist, abzüglich der für das Lesen von und Schreiben auf Cassette verbrauchten Zeit. Die Zeit wird in Einheiten zu 1/300 einer Sekunde gemessen.

Verwandte Befehle: AFTER, EVERY, END, WHILE

TO

(Siehe FOR)

TROFF

TRON

```
TROFF
TRON

10 TROFF:PRINT:PRINT"Programmverfolgung aus"
20 FOR n=1 TO 8
30 PRINT"Programm laeuft":NEXT
40 IF f=1 THEN END
50 TRON:PRINT:PRINT"Programmverfolgung ein"
60 f=1:GOTO 20
run
```

KOMMANDO: TRON ermöglicht es, die Ausführung eines Programms zu verfolgen, da nach dem Tron-Kommando vor der Ausführung die jeweilige Zeilennummer in eckigen Klammern[ ]ausgegeben wird.

TRON schalter die Programmverfolgung ON
TROFF schaltet die Programmverfolgung OFF.

Diese Möglichkeit der Programmverfolgung ist vor allen Dingen dann hilfreich, wenn ein Fehler auftritt, und Sie den davorliegenden Ablauf prüfen wollen.

UNT

UNT (<Adressenausdruck>)

```
PRINT UNT (&FF66)
-154
```

FUNKTION: UNT liefert einen ganzzahligen Wert im Bereich von -32768 bis 32767 (d.h.: das entsprechende Zweier-Komplement des vorzeichenfreien Wertes des <Adressenausdrucks>.

Verwandte Befehle: CINT, FIX, INT, ROUND

UPPER$

UPPER$ (<Textausdruck>)

```
PRINT UPPER$("schneider")
 SCHNEIDER
```

FUNKTION: Gibt den <Textausdruck> umgewandelt in lauter Großbuchstaben zurück.

Die Funktion eignet sich zur Verarbeitung von Eingängen, die in gemischter Form (Großbuchstaben/Kleinbuchstaben) eintreffen können.

Verwandte Befehle: LOWER$

USING

(Siehe PRINT USING)

VAL

VAL (<Textausdruck>)

```
10 CLS:PRINT"Ich kann rechnen !"
20 PRINT:PRINT "Druecke eine Taste zwischen 1 und 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 12
60 PRINT n;"X";x;"=";n*x
70 NEXT:GOTO 20
run
```

FUNKTION: Gibt den numerischen Wert (VAL) des ersten Zeichens im angegebenen <Textausdruck> einschließlich aller negativen Vorzeichen und Dezimalpunkte wieder.

Handelt es sich bei dem ersten Zeichen nicht um eine Zahl, so wird Ø zurückgemeldet. Handelt es sich bei dem ersten Zeichen um ein negatives Zeichen bzw. einen Dezimalpunkt, mit anschließendem nicht numerischen Zeichen, so wird der Fehler "Type mismatch" (13) angezeigt.

Verwandte Befehle: STR$

V P O S

V P O S (# <Ein-/Ausgabegerät>)

```
10 MODE 1:BORDER 0:LOCATE 2,2
20 PRINT"Benutze die Cursor - Tasten oben/unten "
30 WINDOW 39,39,1,25
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 die vertikale Position (=Zeilennummer) des Textcursors des angegebenen <Ein-/Ausgabegeräts> zurück. Das <Ein-(Ausgabegerät> muß in jedem Fall angegeben werden.

Verwandte Befehle: POS, WINDOW

W A I T

W A I T

W A I T <Interfaceadresse>,<UND-Maske>[,<Exclusive-ODER-Maske>]

W A I T &FF34,2Ø,25

KOMMANDO: Wartet, bis über das angesprochene Interface (Ein-/Ausgabeschnittstelle) ein Wert zwischen 0 und 255 eingelesen ist und weitergegeben wird. Der eingelesene Wert wird ggf. mit der Exklusive-ODER-Maske verknüpft und wenn das Ergebnis nicht 0 ist, noch mit der UND-Maske. BASIC verbleibt solange in einem Wartezustand, bis die Eingabe erfolgt ist. Falls Sie das obige Beispiel eingeben, müssen Sie den Rechner komplett zurücksetzen, um weiterarbeiten zu können.

Diese Steuerung ist nur von erfahrenen Programmierern vorzunehmen.

Verwandte Befehle: INP, OUT

W E N D

W E N D

```
10 MODE 1:REM Basic Uhrzeitprogramm
20 INPUT "Geben Sie die aktuelle Uhrzeit ein       (Stunden,
Minuten,Sekunden)";stunde,minute,sekunde
30 CLS:datum = INT(TIME/300)
40 WHILE stunde<24
50 WHILE minute<60
60 WHILE ticken<60
70 ticken = (INT(TIME/300)-datum)+sek
80 LOCATE 70,4
90 PRINT #0,USING "## ";stunde,minute,ticken
100 WEND
110 ticken = 0
120 sekunde =0
130 minute = minute + 1
140 GOTO 30
150 WEND
160 minute = 0
170 stunde = stunde+1
180 WEND
190 stunde = 1
200 GOTO 40
run
```

KOMMANDO: Schließt eine WHILE-Schleife ab. Die WHILE/WEND-Schleife führt Teile des Programms solange aus, bis die entsprechende Bedingung eintritt. Das Beispiel hier verwendet die WHILE/WEND-Schleife, um zu zeigen, wie elegant ein Programm durch dieses Leistungsmerkmal gestaltet werden kann.

Verwandte Befehle: TIME, WHILE

W H I L E

W H I L E <Logischer Ausdruck>

```
10 CLS:PRINT"10 Sekunden Warten !":t=TIME
20 WHILE TIME <t+3000
30 SOUND 1,0,100,15
40 WEND:SOUND 129,40,30,15
run
```

KOMMANDO: Setzt den Anfang und startet den Ablauf einer Pro-

grammschleife, die solange wiederholt wird, wie der <Logische Ausdruck> zutrifft. Das WHILE-Kommando bildet den Anfang der Schleife und definiert die Bedingung. Das WEND-Kommando schließt den Schleifenbereich ab.

Verwandte Befehle: TIME, WEND

WIDTH

WIDTH<ganzzahliger Ausdruck>

WIDTH 86

KOMMANDO: Begrenzt die Anzahl der Zeichen pro Zeile für die Druckerausgabe. BASIC fügt bei längeren Zeilen automatisch einen Wagenrücklauf und Vorschub auf eine neue Zeile ein.

Der Computer geht von einem Standardwert von 132 aus, wenn keine WIDTH-Steuerung angegeben wird.

Die Steuerung WIDTH 255 hebt die zusätzliche Steuerung für Wagenrücklauf/Zeilenvorschub auf und ermöglicht somit einen Druck mit Zeilenabschluß durch den Drucker. Hierbei ist anzumerken, daß die Funktion Wagenrücklauf/Zeilenvorschub erneut durch eine PRINT-Steuerung ausgelöst werden kann, die nicht mit einem Semikolon bzw. Komma abgeschlossen ist.

Verwandte Befehle: POS

WINDOW

WINDOW [#<Ein-/Ausgabegerät>,]<Links>,<Rechts>,
<Oben>,<Unten>

```
10 MODE 0:BORDER 0
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: Legt einen Bildschirmteilbereich als eigen ansprechbares <Ein-/Ausgabegerät> (stream) fest.

Ist das <Ein-/Ausgabegerät> nicht angegeben, so wählt BASIC als Standard #Ø.

Weitere Informationen bezüglich der Fenster (WINDOW) sind in Teil 2 des Kapitels "Wenn Sie Zeit haben..." enthalten.

Verwandte Befehle: WINDOW SWAP

WINDOW SWAP

WINDOW SWAP <Ein-/Ausgabegerät>,<Ein-/Ausgabegerät>

```
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 Nr. 0 "
50 CLS #1:PRINT #1,"Fenster Nr. 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 Bildschirmteilbereich (WINDOW). So können z.B. die auf dem <Ein-/Ausgabegerät> #Ø üblichen BASIC-Meldungen in einen anderen Bildschirmteilbereich (WINDOW) gelegt werden, um sie von normalen Programmausgaben deutlich unterscheiden zu können.

Es müssen beide <Ein-/Ausgabegeräte> angegeben werden, die in diesem Falle nicht durch ein vorangestelltes #-Symbol eingeleitet werden dürfen.

Das Kommando kann dazu eingesetzt werden, von BASIC erzeugte Nachrichten umzuleiten, die normalerweise auf dem Bildschirm (#Ø) ausgegeben werden.

Verwandte Befehle: WINDOW

WRITE

WRITE [# <Ein-/Ausgabegerät>,][<Ausgabeliste>]

```
10 REM Schreibt Variable auf Diskette
20 INPUT "Gib eine Variablennummer ein ";a
30 INPUT "Gib eine Textvariable ein ";a$
40 OPENOUT "daten"
50 WRITE #9,a,a$
60 CLOSEOUT:PRINT "Die Daten sind auf Diskette gesichert !"
run
```

KOMMANDO: Gibt die in der <Ausgabeliste> angegebenen Werte durch Kommas getrennt aus, wobei Texte in Doppelanführungszeichen eingeschlossen werden.

Innerhalb dieses Beispieles werden die von Ihnen eingegebenen Werte der Variablen auf die Ausgabeneinheit #9 (Disketten-Ausgabe) geschrieben.

(Um die Werte dieser Variablen von der Diskette abzurufen, muß ein Programm wird das nachstehende eingesetzt werden.)

Verwandte Befehle: INPUT, LINE INPUT

X O R

<Argument> XOR <Argument>

```
If "Hans"<"Uwe" XOR "Hund">"Maus" then print "richtig" else print "falsch"
Falsch

If "Uwe"<"Hans" XOR "Maus">"Hund" then print "richtig" else print "falsch"
Falsch

If "Hans"<"Uwe" XOR "Maus">"Hund" then print "richtig" else print "falsch"
Richtig
....

Print 1 XOR 1
0

Print 0 XOR 0
0

Print 1 XOR 0
1
```

FUNKTION: Führt eine bitweise Bool'sche Operation auf ganzen Zahlen aus. Das Ergebnis ist hierbei 1, außer wenn beide Argumentbits gleich sind - ausschließliches ODER.

Verwandte Befehle: AND, OR, NOT

X P O S

X P O S

```
10 MODE 1:DRAW 320,200
20 PRINT "X POSition des Grafik-Cursors";
30 PRINT XPOS
run
```

FUNKTION: Gibt die augenblickliche x-Koordinate des Graphikcursors zurück.

Verwandte Befehle: MOVE, MOVER, ORIGIN, YPOS

Y P O S

Y P O S

```
10 MODE 1:DRAW 320,200
20 PRINT "Y POSition des Grafik-Cursors";
30 PRINT YPOS
run
```

FUNKTION: Gibt die augenblickliche y-Koordinate des Graphik-cursors zurück.

Verwandte Befehle: MOVE, MOVER, ORIGIN, XPOS

Z O N E

Z O N E <G a n z z a h l i g e r A u s d r u c k>

```
10 CLS:FOR z=2 TO 20
20 ZONE z
30 PRINT "X", "X ZONE ="; z:NEXT
run
```

KOMMANDO: Ändert die Breite der Druckzone (als PRINT-Befehle unter Verwendung eines Kommas zwischen Druckteilen angegeben). Der Standardwert beträgt 13 für die Druckzone (Spalten), kann jedoch nach Angabe des <ganzzahligen Ausdrucks> im Bereich zwischen 1 und 255 geändert werden. Der Zonenwert wird von NEW-, LOAD-, CHAIN- und RUN "<Dateiname>"-Kommandos zurückgesetzt.

Verwandte Befehle: PRINT, WIDTH

Für Ihre Notizen:

Kapitel 4

Einsatz von Disketten und Kassetten

Dieses Kapitel beschreibt die Anfertigung von Arbeits-Disketten und führt Sie in die Möglichkeiten von CP/M und seinen Dienstprogrammen ein.

Behandelte Themen sind:

- **Anfertigen einer Sicherheits-System/Dienstprogramm-Diskette**
- **Anfertigen einer Arbeits-System/Dienstprogramm-Diskette**
- **Arbeiten mit einer BASIC-Diskette**
- **Installation einer schlüsselfertigen Schneider BASIC-Applikation**
- **Installation einer schlüsselfertigen CP/M-Applikation**

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, und wie mit DISCCOPY (ein Laufwerk) und COPYDISC (zwei Laufwerke) exakte Disketten-Kopien angefertigt werden können. Das folgende Kapitel beschreibt die Anfertigung von Disketten für Ihre Programme.

Sicherungs-Disketten

Es ist empfehlenswert, von der mit dem CPC664 ausgelieferten System-Diskette eine Kopie anzufertigen und das Original an einem sicheren Ort zu verwahren.

Die ausgelieferte Diskette hat zwei Seiten, eine für System- und Dienstprogramme, die andere für Dr.LOGO. Natürlich hat jede Diskette zwei Seiten, die beliebig verwendet werden können.

Eine neue Diskette kann mit DISCCOPY und COPYDISC formatiert und mit Programm-Kopien belegt werden.

Eine Arbeits-Diskette für System- und Dienstprogramme

Neben der Anfertigung einer Kopie der System/Dienstprogramm- und Dr.LOGO-Diskette ist es empfehlenswert, eine Arbeits-Dienstprogramm-Diskette zu erstellen, die nur einige der häufiger benötigten Programme von der System/Dienstprogramm-Diskette enthält. Dadurch steht mehr Speicherplatz für Ihre Programme zur Verfügung. Sollten Sie noch zusätzliche Dienstprogramme auf dieser Diskette benötigen, so können sie jederzeit von der Kopie Ihrer System/Dienstprogramm-Diskette geholt werden. Alle häufiger benötigten Programme stehen Ihnen dadurch sofort zur Verfügung.

Erstellen Sie zuerst mit CP/M und FORMAT eine neue Leer-Diskette. Verwenden Sie dann das CP/M-Programm FILECOPY, um die gewünschten Programme auf die neue Diskette kopieren zu können. Die Erfahrung wird Ihnen zeigen, welche Programme schneller greifbar sein müssen. In diesem Beispiel wurden folgende Programme ausgewählt:

```
AMSDOS.COM
FILECOPY.COM
DISCCOPY.COM
```

Legen Sie die CP/M-Systemdiskette ein. Geben Sie nun ICPM ein und wenn auf dem Bildschirm A>erscheint, tippen Sie:

```
FILECOPY  FILECOPY.COM
```

Die Quellen-Diskette ist die Diskette, von der Sie Programme oder Daten kopieren wollen; die Zieldiskette ist die Diskette, die Sie neu erstellen wollen, d.h. auf die Sie die Programme

oder Daten kopieren wollen.
Daraufhin kopieren Sie die beiden anderen Programme mit den Befehlen

```
FILECOPY AMSDOS.COM
FILECOPY DISCCOPY.COM
```

Nach der Erstellung einer Arbeits-System/Dienstprogramm-Diskette mit ausgesuchten Programmen kann die zweite Seite dieser Diskette oder beide Seiten weiterer Disketten durch DISCCOPY oder COPYDISC vorbereitet werden. Beachten Sie jedoch, daß Sie die am Ende dieses Handbuches enthaltenen Lizenzabkommen nicht verletzen.

BASIC-Diskette

Die Spuren 0 und 1 der System-Diskette sind grundsätzlich für CP/M reserviert und können nicht durch Sie belegt werden. Wenn eine Diskette jedoch vollständig mit BASIC-Spielen, Programmen oder Daten belegt werden soll und Sie an dieser Diskette weder CP/M noch CP/M-Dienstprogramme benötigen, kann sie auch ohne CP/M-Systemspuren formatiert werden. In diesem Falle handelt es sich um eine reine Daten-Diskette.

Datendisketten sind mit dem Befehl:

```
FORMAT D
```

zu formatieren.

Um Programme auf diese Diskette zu kopieren, muß FILECOPY (vorher von der System-Diskette laden!) verwendet oder LOAD und SAVE unter BASIC benutzt werden. Bei einem System mit zwei Diskettenlaufwerken kann das CP/M-Dienstprogramm PIP verwendet werden.

Die Programme COPYDISC und DISCCOPY formatieren die Ziel-Diskette auf den gleichen Stand wie die Quellen-Diskette.

Fertige Schneider BASIC-Disketten

Wenn Sie für den CPC664 ein in Schneider BASIC geschriebenes Anwender-Programm kaufen, ist es in der Regel beim Einschalten betriebsbereit. Es ist also sozusagen "schlüsselfertig" (der Begriff stammt aus der Zeit, als Kleincomputer noch mit abschließbaren Netzschaltern ausgerüstet waren). Sie müssen

dieses Programm also nur noch auf eine geeignete Arbeits-Diskette bringen.

Schlüsselfertiges BASIC auf der ausgelieferten Diskette

Kopieren sie mit COPYDISC oder DISCCOPY die System/Dienstprogramm-Diskette, verwahren Sie die System-Diskette an einem sicheren Ort und verwenden Sie die Kopie. Folgen Sie dann den Instruktionen zum Ablauf des Programms. Wenn Sie zusätzliche Programme von der System/Dienstprogramm-Diskette benötigen, kopieren Sie diese mit FILECOPY.

Schlüsselfertiges BASIC auf der Arbeits-Diskette

In diesem Falle kopieren Sie die gelieferten Programme auf die bestehende Diskette. Geben Sie FILECOPY *.* ein und folgen Sie den Instruktionen. Antworten Sie auf die Frage

Ambigous file name: Confirm individual files (Y/N)?

mit N.

Das FILECOPY-Programm informiert Sie über die kopierten Dateinamen. Nun können Sie die neuen Anwender-Programme Ihrer Arbeits-Diskette starten.

Fertige CP/M-Disketten

Das Betriebssystem CP/M ermöglicht Ihnen das Laden und den Ablauf einer umfangreichen Software-Bibliothek, die bereits für CP/M unterstützende Personal Computer erstellt worden ist. Da die grundsätzlichen Erfordernisse für die Anwendung dieser Programme bereits erfüllt sind, ist für ihre Benutzung auf dem CPC664 nur noch die Bereitstellung einer geeigneten Diskette und möglicherweise eine Mitteilung über die besondere Methode, mit welcher der CPC664 den Bildschirm ansteuert, erforderlich.

Eine zur Bearbeitung einer bestimmten Anwendung auf einer Diskette befindliche Anzahl von Programmen wird als Programm-Paket bezeichnet. Diese Pakete sind normalerweise für die Verwendung auf verschiedenen Computern ausgelegt, die eine jeweils unterschiedliche Bildschirmgröße und Cursor-Bearbeitung aufweisen. Manchmal sind die käuflichen Pakete bereits für den CPC664

installiert bzw. durch ein Menü alternativer Installationen auf ihn vorbereitet. In diesem Fall befolgen Sie bitte die Instruktionen des entsprechenden Software-Pakets. Wenn das Paket keine CPC664-Variante zur Verfügung stellt, zeigt Ihnen der Abschnitt **"Konfigurieren eines CP/M-Programms"** die an den CPC664-Bildschirm zu sendenden Befehle, um die für das Paket erforderlichen Effekte erzeugen zu können. Normalerweise ist in die Installation oder System-Anpassung die Eingabe eines entsprechenden Codes einbezogen. Folgen Sie auch hier den Anweisungen des Software-Pakets.

Die erworbene Software muß sich auf einer für den CPC664 geeigneten Diskette befinden. Häufig verwenden Computer unterschiedliche Disketten-Formate. Andererseits bedeutet gleiches Disketten-Format noch nicht, daß die auf ihnen gespeicherten Informationen kompatibel sind. Verwenden Sie daher möglichst nur 3 Zoll Schneider CPC664-Versionen.

Erstellung einer schlüsselfertigen CP/M-System-Diskette

Es wird empfohlen, die Original-System-Diskette mit dem neuen Software-Paket an einem sicheren Ort zu verwahren und die Programme auf eine andere Diskette zu kopieren. Obgleich die obigen Anweisungen für ein System mit einem Disketten-Laufwerk gelten, ist es im allgemeinen ebenso einfach, sie auf Systeme mit zwei Disketten-Laufwerken anzuwenden (indem das zweite Laufwerk ignoriert wird).

Formatieren Sie zuerst eine neue System-Diskette. Kopieren Sie dann alle Programme Ihrer Software-Paket-Diskette mit Hilfe von FILECOPY der System/Dienstprogramm-Diskette.

Geben Sie FILECOPY *.* ein und folgen Sie den Instruktionen auf dem Bildschirm. Antworten Sie auf die Frage

Ambigous filename: Confirm individual files (Y/N)?

mit N.

Das FILECOPY-Programm nennt Ihnen die Namen der kopierten Dateien.
Nach Abschluß des FILECOPY-Programmes besitzen Sie eine Arbeitskopie der schlüsselfertigen Diskette. Wenn Sie irgendwelche Dienstprogramme benötigen, kopieren Sie diese mit FILECOPY von Ihrer System/Dienstprogramm-Diskette.

Konfiguration eines CP/M-Programms

Der CPC664 unterstützt eine große Anzahl Steuercodes, die für die Kunden-Anpassung eines unter CP/M ablauffähigen Software-Paketes geeignet sind.

Die meisten Software-Pakete erfordern die Ausgabe von Meldungen auf einen beliebigen Bildschirm-Bereich, sowie die Eingabe in einem bestimmten Bildschirm-Bereich, und arbeiten in der Regel mit Cursor-Steuerung.

Wenn Ihr Paket bereits für den CPC664 aufbereitet ist, müssen Sie nichts weiter unternehmen.

Konfiguration der Ausgabe

Die Paket-Installation besteht normalerweise aus dem Ablauf eines speziellen Programms (oftmals mit dem Namen INSTAL), das einige Fragen über die Parameter des CPC664-Bildschirms stellt. Die Antworten sollten der folgenden Tabelle entnommen werden.

HEX-Wert	Dezimal-Wert	Funktion
&07	7	Pieps-Ton
&08	8	Bewege den Cursor eine Position zurück
&09	9	Bewege den Cursor eine Position vor
&0A	10	Bewege den Cursor eine Zeile abwärts
&0B	11	Bewege den Cursor eine Zeile aufwärts
&0C	12	Bewege den Cursor auf die Ausgangsposition und lösche den Bildschirm (home)
&0D	13	Bewege den Cursor auf die linke Fensterkante der momentanen Zeile
&10	16	Lösche das Zeichen auf der Cursor-Position
&11	17	Lösche von der linken Fensterkante bis einschließlich Cursor-Position

HEX-Wert	Dezimal-Wert	Funktion
&12	18	Lösche von der momentanen Cursor-Position (einschl.) bis zur rechten Fensterkante
&13	19	Lösche vom Fensteranfang bis einschließlich momentaner Cursor-Position
&14	20	Lösche von der momentanen Cursur-Position (einschl.) bis zum Fensterende
&18	24	Kehre EIN/AUS des inversen VIDEO um
&1E	30	Bringe den Cursor auf die Ausgangs-Position (home)
&1F <s><z>	31 <s><z>	Bewege den Cursor auf die gegebene Position im momentanen Fenster <s> ist die Spalte(von 1 bis 80), <z> ist die Zeile (von 1 bis 25).

Konfiguration der Eingabe

Die Programme des Software-Pakets können u.U. die Tastatur abfragen. Mit Ausnahme der Cursor-Tasten geben die meisten Tasten der CPC664-Tastatur Standardwerte an. Unter Verwendung von SETUP (siehe Teil 2 des Kapitels 5 "Schneider und CP/M") können die von der Tastatur erzeugten Codes redefiniert werden. Es sollten jedoch möglichst die Standard-Werte verwendet werden, indem die verschiedenen Pakete entsprechend konfiguriert werden.

Die mit "WP-Wert" bezeichnete Spalte der folgenden Tabelle zeigt typische Werte, die mit SETUP für die Tastatur eingestellt werden können. Dies ist beispielsweise für eine Textverarbeitung sinnvoll, wenn Curcor-Codes sowohl von der Cursor-Tasten als auch von anderen Tastatur-Tasten mitgeliefert werden und das Textverarbeitungs-Paket nur einen einheitlichen Code für jede Funktion verarbeiten kann.

Die Installationsprozedur für ein Paket besteht üblicherweise aus der Beantwortung einiger Fragen über die Parameter der CPC664-Tastatur. Die Antworten sollten der folgenden Tabelle entnommen werden.

Taste	HEX-Wert	Dezimal-Wert	Tasten-Nummer	WP-Wert (dezimal)
Cursor auf	&F0	240	0	5
Cursor rechts	&F3	243	1	4
Cursor ab	&F1	241	2	24
Cursor links	&F2	242	8	19
Clr	&10	16	16	7
Return	&0D	13	18	13
Space	&20	32	47	32
Escape	&FC	252	66	27
Tab	&09	9	68	9
Del	&7F	127	79	127

Start eines fertigen CP/M-Pakets

Normalerweise muß nach dem Erscheinen des A>-Prompts nur noch der Programm-Name des Software-Pakets eingegeben werden. Sie geben beispielsweise zum Ablauf eines Lohn-Programms mit dem Namen GEHALT.COM nur noch GEHALT ein.

Autostart eines fertigen CP/M-Pakets

1. Disketten

Das Betriebssystem CP/M ist unter Verwendung einer besonderen Diskette in der Lage, bestimmte Programme nach dem Aufruf von CP/M automatisch zu starten. Dies wird durch eine Option im SETUP-Programm ermöglicht (siehe Teil 2 des Kapitels "Schneider und CP/M").

2. Cassetten

Sofern Sie beabsichtigen, eine Cassetten-Einheit an das System anzuschließen (in der in Teil 2 des Grundlagenkurses beschriebenen Form), schalten Sie den Computer über den Befehl |TAPE auf Cassetten-Betrieb. In diesem Fall erscheinen verschiedene Software-Nachrichten in Fragestellungs-Funktion auf dem Bildschirm, die während des normalen Diskettenbetriebes nicht sichtbar sind.

Im Gegensatz zu Disketten weisen Dateinamen auf der Cassette nicht so strikte Regeln in Bezug auf ihre Form auf. Sie können bis zu 16 Zeichen lang sein, können Zwischenräume und Satzzeichen einschließen. In einigen Fällen kann auf alle diese letzteren verzichtet werden.

Die folgende Aufstellung beschreibt die ABWEICHUNGEN beim Betrieb und jeweils auf diese BASIC-Kommandos bezogen. Die Beschreibung der Steuerungen selbst sind im Kapitel "Komplette

Aufstellung der Schneider CPC664 BASIC-Schlüsselwörter" enthalten.

CAT

Sie werden angewiesen:

Press PLAY then any key:

...woraufhin Sie die Taste [PLAY] auf Ihrer Cassetteneinheit drücken sollten und anschließend eine der grauen Tasten auf dem Computer. Das Band in der Cassette läuft an und der Computer zeigt jede auf der Cassette gespeicherte Datei an. Der Dateiname wird in Großbuchstaben angezeigt, gefolgt von der Anzahl der Blöcke des jeweiligen Programms, einem Zeichen, das die Dateiart angibt und dem Wort ok wenn die Datei lesbar und damit ladbar ist. Die Zeichen für die Dateiart sind:

$ Standard BASIC Programm
% geschütztes BASIC Programm
* ASCII Datei
& Datei in Binärform

Die CAT-Funktion beeinträchtigt nicht das laufende Programm im Speicher des Computers.

Wurde eine Cassetten-Datei ohne Angabe eines Namens aufgenommen, so zeigt CAT:

Unnamed file (nicht bezeichnete Datei)

Die Funktion CAT wird mit dem Drücken der Taste [ESC] beendet.

Lesefehler

Werden die obengenannten Nachrichten in Bezug auf Dateilesen nicht gezeigt bzw. erhalten Sie folgende Meldung:

Read error a oder Read error b

Hierdurch wird auf folgende mögliche Fehler hingewiesen:

1. Ihre Cassetteneinheit ist nicht einwandfrei an den Anschluß TAPE des Computers angeschlossen (siehe Teil 2 des Grundlagenkurses).
2. Die Steuerung VOLUME bzw. LEVEL auf Ihrer Cassetteneinheit ist nicht einwandfrei abgeglichen.

3. Schlechte Bandqualität bzw. beschädigtes Band.
4. Das Band wurde einem Magnetfeld ausgesetzt, bedingt durch einen Lautsprecher, Fernseher, o.ä.
5. Der Lesevorgang wurde für eine Cassette ausgelöst, die sich nicht für einen Einsatz innerhalb der Schneider Computersysteme eignet. An dieser Stelle ist anzumerken, daß Sie die gesamte für Schneider-CPC664 bestehende Software einsetzen können, darüber hinaus auch die Mehrzahl der für Schneider CPC464 vorhandenen Programme.

CHAIN
CHAIN MERGE
LOAD
MERGE
RUN

Falls Sie die erste geeignete Datei auf die Cassette aufnehmen möchten, brauchen Sie keinen Dateinamen anzugeben. Beispiel:

```
CHAIN ""
CHAIN "", 1ØØ

CHAIN MERGE ""
CHAIN MERGE "", 1ØØ
CHAIN MERGE "",1ØØ, DELETE 3Ø-7Ø

LOAD ""
LOAD "", &1F4Ø

MERGE""
```

RUN "" (Hierzu ist anzumerken, daß es zur Auslösung des Befehls ausreicht, zuerst die Taste [CTRL] zu drücken und anschließend die kleine [ENTER]-Taste auf dem Tastenfeld. Einsatz bei Cassetten-Software, anschließend Eingabe |TAPE.)

Anschließend werden Sie aufgefordert:

```
Press PLAY then any key:
```

...hierauf sollte die Taste [PLAY] auf Ihrer Cassetteneinheit gedrückt werden, anschließend eine der grauen Tasten auf dem Computer. Das Band in der Cassette beginnt zu laufen und der

Computer lädt die Datei.

Am Bildschirm erscheint folgende Meldung:

Loading Dateiname block 1

...anschließend wird die Anzahl der Blocknummern angezeigt, bis die Datei geladen ist.

Lautet das erste Zeichen des Dateinamens ! dann wird die obige Meldung unterdrückt und Sie werden nicht aufgefordert, "irgendeine Taste zu drücken", um die Aufnahme der Datei zu veranlassen. (Sie haben hierbei sicherzustellen, daß die Taste PLAY auf Ihrer Cassetteneinheit gedrückt ist.) Verwenden Ihre Programme das ! und es soll ebenso für die Diskette eingesetzt werden, so wird das Ausrufungszeichen ! während des Diskettenbetriebes ignoriert (wenn der Dateiname gelesen ist).

Hierzu ist anzumerken, daß das Ausrufungszeichen (!) keine der Zeichenstellen auf der Cassette bzw. in der Dateibezeichnung der Diskette belegt.

Brechen Sie den Ladevorgang durch zweimaliges Drücken der Taste [ESC] ab, so erscheint auf dem Bildschirm die Fehlermeldung:

Broken in

Wurde die Datei nicht einwandfrei übernommen, so lesen Sie den Absatz "Lesefehler" zu Beginn dieses Abschnittes.

ACHTUNG: Die interne Disketten-Schnittstelle nimmt einen kleinen Teil des Speichers ein, der in einigen Fällen von Herstellern der Cassetten-Software für den CPC464 eingesetzt wurde. Diese Cassetten arbeiten nicht einwandfrei mit dem CPC664 im Cassettenbetrieb.

EOF
POS (#9)

Diese Funktionen arbeiten sowohl auf Cassette als auch auf Diskette.

INPUT #9
LINE INPUT #9
OPENIN and CLOSEIN

Hierzu brauchen Sie nicht den Dateinamen anzugeben, wenn Sie die erste entsprechende Datei auf der Cassette übernehmen wollen. Steuerbeispiel:

```
OPENIN ""
```

Sie werden daraufhin angewiesen:

```
Press PLAY then any key:
```

woraufhin Sie die Taste [PLAY] auf Ihrer Cassetteneinheit drücken sollten, anschließend eine der grauen Tasten auf dem Computer.

Somit läuft das Band in der Cassette an und der Computer übernimmt die ersten zwei Kbytes der Datei in einen als "File buffer" bezeichneten Abschnitt des Speichers.

Die Eingabe erfolgt vom "file buffer", bis dieser leer ist, wonach der Computer quittiert:

```
Press PLAY then any key:
Loading <Dateiname> block 1
```

Jetzt wird blockweise die gesamte Datei geladen.

Ist das erste Zeichen des Dateinamens in der OPENIN-Steuerung ein Ausrufungszeichen (!), so wird die obengenannte Meldung unterdrückt und Sie brauchen keine Tasten zu drücken, um eine Datei zu lesen. (Hierbei ist jedoch darauf zu achten, daß die PLAY-Taste auf Ihrer Cassetteneinheit gedrückt ist.) Hierzu ist anzumerken, daß das Ausrufungszeichen ! keine Zeichenstelle innerhalb der Cassette bzw. des Dateinamens der Diskette belegt. Brechen Sie den Ladevorgang durch zweimaliges Drücken der Taste [ESC] ab, so erscheint auf dem Bildschirm die Fehlermeldung:

```
Broken in
```

Wird die Datei nicht einwandfrei geladen, so lesen Sie den Absatz "Lesefehler" zu Beginn dieses Abschnittes.

LIST #9
OPENOUT and CLOSEOUT
PRINT #9
WRITE #9

Sie brauchen den Dateinamen dann nicht anzugeben, wenn die Datei ohne Bezeichnung abzuspeichern ist. Beispiel:

```
OPENOUT ""
```

Die ersten 2 Kybtes der auf Cassette abzuspeichernden Datei werden zuerst in einen mit "file buffer" bezeichneten Abschnitt des Speichers eingeschrieben. Ist der "file buffer" gefüllt, so erfolgt folgende Angabe:

```
Press REC and PLAY then any key:
```

...woraufhin die Tasten [RECORD] und [PLAY] auf Ihrer Cassetteneinheit zu drücken sind, anschließend eine der grauen Tasten auf dem Computer. Das Band in der Cassette beginnt sich zu drehen und der Computer speichert die Inhalte des file buffers ab. Anschließend füllt der Computer den file buffer wieder mit den nächsten 2 Kbytes der Datei auf und meldet erneut:

```
Press REC and PLAY then any key:
```

...und speichert die nächsten 2 Kbytes auf die Cassette ab.

Ist der file buffer teilweise gefüllt und erscheint der Befehl CLOSEOUT, so übernimmt der Computer die verbleibenden Inhalte des file buffers auf Cassette und antwortet:

```
Press REC and PLAY then any key:
```

Der Bildschirm zeigt folgende Meldung:

```
Saving<Dateiname>block 1
```

Ist das erste Zeichen des Dateinamens in der OPENIN-Steuerung ein Ausrufungszeichen (!), so wird die obengenannte Meldung unterdrückt und Sie brauchen keine Tasten zu drücken, um eine Datei zu sichern. (Achten Sie darauf, daß die [RECORD] - und [PLAY] -Tasten auf Ihrer Cassetteneinheit gedrückt sind.)

Hierzu ist anzumerken, daß das Ausrufungszeichen ! keine Zeichenstelle innerhalb der Cassette bzw. des Dateinamens der Diskette belegt. Brechen Sie den Sicherungsvorgang durch zweimaliges Drücken der Taste [ESC] ab, so erscheint auf dem Bildschirm die Fehlermeldung:

```
Broken in
```

Erfolgreiche Sicherung

Für die erfolgreiche Sicherung einer Datei sind folgende Prüfungen erforderlich:

1. Prüfen, ob Ihre Cassetteneinheit einwandfrei in den Computer-Anschluß (TAPE) eingesteckt ist (siehe Teil 2 des Grundlagenkurses).
2. Prüfen, ob "RECORD" bzw. "LEVEL"-Steuerung auf Ihrer Cassetteneinheit einwandfrei abgestimmt ist.
3. Prüfen, ob Sie gegebenenfalls Cassetten geringerer Qualität einsetzen bzw. C120.
4. Sicherstellen, daß Ihre Cassetten nicht magnetischen Feldern ausgesetzt sind durch die Nähe eines Lautsprechers, Fernsehers usw.
5. Vor Löschen eines Programms aus dem Speicher sicherstellen (CAT), ob das Programm entsprechend abgespeichert wurd.
6. Sicherstellen, daß Ihre Cassetteneinheit regelmäßig gewartet und die Tonköpfe regelmäßig gereinigt werden.

SAVE

Die Angabe des Dateinamens entfällt dann, wenn Sie das Programm als nicht bezeichnete Datei abspeichern wollen. Beispiel:

```
SAVE""
```

Sie werden daraufhin angewiesen:

```
Press REC and PLAY then any key:
```

...anschließend sollten Sie die Tasten [RECORD] und [PLAY] Ihrer Cassetteneinheit drücken, danach eine der grauen Tasten auf dem Computer. Das Band der Cassette beginnt zu laufen und der Computer speichert das Programm ab.

Der Bildschirm zeigt die System-Nachricht:

Saving <Dateiname> block 1

...anschließend werden die einzelnen Blocknummern angezeigt, bis die Datei abgespeichert ist.

Ist das erste Zeichen des Dateinamens ein Ausrufungszeichen (!), so werden die obengenannten Nachrichten unterdrückt und Sie brauchen keine Tasten zu drücken, um eine Datei zu speichern. (Hierbei ist jedoch darauf zu achten, daß die Tasten [RECORD] und [PLAY] auf Ihrer Cassetteneinheit gedrückt sind.) Es ist anzumerken, daß das Ausrufungszeichen keine Zeichenstelle innerhalb des Cassetten- bzw. Dateinamens der Diskette belegt. Brechen Sie den Sicherungsvorgang durch zweimaliges Drücken der Taste [ESC] ab, so erscheint auf dem Bildschirm die Fehlermeldung:

Broken in

Siehe "Erfolgreiches Abspeichern" zu Beginn dieses Abschnittes.

SPEED WRITE

Diese Steuerung arbeitet nur mit der Cassette. Sie können mit diesem Befehl die Geschwindigkeit des Abspeicherns oder Ladens verändern.

SPEED WRITE Ø speichert mit 1000 Baud (Bits pro Sekunde),

SPEED WRITE 1 verdoppelt die Aufzeichnungsrate auf 2000 Baud.

Fehlermeldungen

Die Fehlermeldungen 7, 21, 24, 25, 27 und 32 (siehe Kapitel "Wenn Sie Zeit haben...") können während des Cassettenbetriebs auftreten.

Schneider externe Steuerungen

Der Eingabe- und Ausgabehinweis für Cassette bzw. Diskette erfolgt durch nachstehende Steuerungen:

|TAPE (welches unterteilt wird in |TAPE.IN und |TAPE.OUT)
|DISC (welches unterteilt wird in |DISC.IN und |DISC.OUT)

Alle externen Steuerungen

|A
|B
|CPM
|DIR
|DRIVE
|ERA
|REN
|USER

werden für Diskette ausgeführt, unabhängig davon, ob Cassettenbetrieb gewählt wurde.

KAPITEL 5
AMSDOS und CP/M

Teil 1: AMSDOS

Dieses Kapitel beschreibt AMSDOS, seine verfügbaren Befehle und seine Anwendung. Es detailliert das Format der Dateinamen und bietet eine Übersicht über die Befehle. Jeder Schritt wird durch entsprechende Beispiele verdeutlicht.

Folgende Themen werden behandelt:

- ★ Einführung in AMSDOS
- ★ Disketten-Inhaltsverzeichnis
- ★ Disketten-Wechsel
- ★ Format der Dateinamen
- ★ AMSDOS Datei-Vorspann
- ★ Wild Cards
- ★ Beispiel-Programm mit AMSDOS-Befehlen
- ★ Übersicht der AMSDOS-Befehle
- ★ Datei-Bearbeitung
- ★ Übersicht der Fehlermeldungen

Einleitung

AMSDOS erweitert das mit Ihrem Computer gelieferte BASIC durch eine Anzahl externer Befehle, die jeweils durch das vorausgestellte Balkensymbol gekennzeichnet werden.

AMSDOS ermöglicht es dem Benutzer, die Disketten nach Belieben zu wechseln, so lange keine Dateien in Bearbeitung sind – andernfalls wird eine Fehlermeldung ausgegeben und es kann zu Datenverlusten kommen, wenn gerade in die offene Datei geschrieben wurde.

Disketten-Inhaltsverzeichnis

Jede Diskette ist in zwei Bereiche unterteilt, einen für das Inhaltsverzeichnis und einen für Daten. Das Inhaltsverzeichnis beinhaltet eine Liste aller Dateinamen und einen „Belegungsplan", der angibt, wo sich die Dateien auf der Diskette befinden. AMSDOS und CP/M können die Größe einer Datei nach Einsicht der Inhaltsverzeichnis-Einträge 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 ihr Inhaltsverzeichnis-Eintrag geprüft und der Speicherort auf der Diskette bestimmt. Beim Anlegen einer neuen Datei wird der freie Speicherplatz reduziert, beim Löschen einer Datei entsteht freier Speicherplatz. Das Inhaltsverzeichnis arbeitet mit 1 KByte-Einheiten und kann bis zu 64 unterschiedliche Einträge verwalten. Große Dateien haben einen Eintrag für jeden 16 KByte-Block. Der Anwender merkt hiervon jedoch nichts.

Diskettenwechsel

Unter AMSDOS kann eine Diskette immer dann gewechselt oder aus dem Laufwerk entfernt werden, wenn auf das Laufwerk nicht zugegriffen wird und weder Eingabe- noch Ausgabe-Dateien dieses Laufwerks eröffnet sind. Im Gegensatz zu CP/M ist ein Log-In der Disketten (Warmstart) nicht erforderlich.

Der Diskettenwechsel während eines Schreibzugriffs kann die Daten auf der Diskette verfälschen. Sind beim Wechsel der Diskette Dateien nicht geschlossen, so wird eine Fehlermeldung ausgegeben, nachdem AMSDOS diesen Zustand erkannt hat. Alle bis dahin geschriebenen Daten sind verloren, und der letzte Inhaltsverzeichnis-Eintrag wird nicht auf die Diskette geschrieben. AMSDOS kann jedoch den Disketten-Wechsel nur dann erkennen, wenn das Inhaltsverzeichnis gelesen wird, was nach jedem 16 KByte-Block der Datei und immer dann, wenn eine Datei eröffnet oder geschlossen wird, geschieht. Dadurch sind beim Diskettenwechsel mit nicht geschlossenen Dateien bis zu maximal 16 KByte Daten zerstörbar.

AMSDOS-Dateinamen und -Dateitypen

Die Namensgebung von Disketten-Dateien ist derart ausgelegt, daß der Dateityp erkennbar ist. Diese Namens-Konvention zwingt den Computer nicht, die Datei auf eine bestimmte Art und Weise zu behandeln. Einige Programme akzeptieren eine Datei jedoch nur, wenn der korrekte Namenstyp angegeben ist. BASIC akzeptiert jeden Namenstyp, sucht jedoch 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 8 Zeichen lang sein, der zweite bis zu 3 Zeichen. So sind beispielsweise `"SCREEN.BIN"`, `"WELCOMED.BAS"` und `"FORMAT.COM"` zulässige Dateinamen. Der zweite Teil des Dateinamens ist der sogenannte Dateityp. Dateinamen und Dateitypen können aus Buchstaben und Ziffern zusammengesetzt sein, dürfen jedoch keine eingeschlossenen Leerzeichen enthalten. Einige mögliche Dateitypen sind:

.<Leerzeichen>	Nicht spezifizierter Typ. Kann eine Datei, erzeugt durch `OPENOUT "RESULTS"` oder über ein durch AMSDOS (mittels `SAVE "PROGRAM", A`) gesichertes BASIC-Programm, sein.
`.BAS`	BASIC-Programm, durch AMSDOS mittels `SAVE "PROGRAM"` oder `SAVE "PROGRAM", P` oder `SAVE "PROGRAM.BAS", A` gesichert.
`.BIN`	Programm oder Speicherbereich, durch AMSDOS mittels `SAVE "MEMORY", B,` <binäre Parameter> gesichert.
`.BAK`	Alte Version einer Datei, wobei AMSDOS oder ein Dienstprogramm eine neuere Version dieser Datei mit einem bereits bestehenden Namen gesichert hat. Hierdurch wird die Abspeicherung einer Sicherheits-Kopie der vorhergehenden Version ermöglicht.
`.COM`	Befehls-Datei. Alle CP/M-Dienstprogramme sind von diesem Dateityp.

AMSDOS-Vorspann

AMSDOS sichert Dateien gemeinsam mit einem geeigneten Typ-Identifikator. Wenn Sie die oben genannten Standard-Typen beibehalten wollen, brauchen Sie keine Typen spezifizieren. AMSDOS BASIC-Programmdateien, geschützte BASIC-Programmdateien und binäre Dateien werden auf der Diskette mit einem Vorspann aufgezeichnet, so daß der AMSDOS `LOAD`-Befehl sie wiedererkennen und entsprechend bearbeiten kann. Der Programmierer braucht beim `LOAD`-Befehl den Typ (BASIC, geschütztes BASIC, binär oder ASCII) nicht anzugeben. Diese Information wird dem Vorspann entnommen. Wenn für eine Datei kein Vorspann gefunden wird, geht AMSDOS davon aus, daß es sich um eine ASCII-Datei (Text) handelt.

Wenn AMSDOS eine Datei ohne spezifizierten Dateityp laden soll, sucht es nach einer Datei vom Typ `.` <Leerzeichen>. Existiert diese Datei nicht auf der Diskette, so sucht AMSDOS nach einer Datei vom Typ `.BAS` und dann `.BIN`. Dadurch kann ein Dateiname gekürzt werden, d.h. in den meisten Fällen muß kein Dateityp spezifiziert werden.

Eine durch den Befehl `OPENOUT` eröffnete und daraufhin mit den BASIC-Befehlen `WRITE`, `PRINT` oder `LIST` beschriebene Disketten-Datei besitzt keinen Vorspann und ist vom Typ ASCII (Text). Der Disketten-Befehl `OPENIN` sucht nach Dateien mit nicht spezifiziertem Datei-Typ in der gleichen Reihenfolge wie der `LOAD`-Befehl.

Dateinamen in Systemen mit zwei Laufwerken

In einem System mit zwei Laufwerken können Dateien auf beiden Laufwerken bestehen. Der Anwender muß das Laufwerk spezifizieren, da der Computer die Datei nicht automatisch in beiden Laufwerken sucht.

Sie können zur Auswahl eines Laufwerks die Befehle `|A`, `|B` oder `|DRIVE` und dann den normalen Dateinamen verwenden. Alternativ kann die standardmäßige Laufwerk-Zuweisung durch die Spezifikation mit vorgestelltem `A:` oder `B:` ersetzt werden. So sichert beispielsweise

```
|B
SAVE "PROG.BAS"
|A
```

und

```
|A
SAVE "B:PROG.BAS"
```

das Programm auf dem zweiten Laufwerk B.

Auf ähnliche Weise können Sie die voreingestellte USER-Zuweisung überschreiben, indem Sie eine USER-Nummer (im Bereich 0–15) vor den Dateinamen stellen. Zum Beispiel:

```
LOAD "15:PROG.BAS"
```

... und ...

```
SAVE "15:PROG.BAS"
```

würden das Programm vom Disketten-Teil der USER-Nummer 15 laden und sichern, ganz gleich, welche USER-Nummer voreingestellt war (siehe |USER-Kommando).

Schließlich ist es auch möglich beide Voreinstellungen für USER und DRIVE (in dieser Reihenfolge zu überschreiben, indem man beides dem Dateinamen voranstellt, z.B.:

```
RUN "15B:PROG.BAS"
```

Wild-Cards

Es ist häufig erforderlich, einige Disketten-Operationen (katalogisieren, kopieren, löschen usw.) auf mehrere Disketten-Dateien zu beziehen. Wenn ein Dateiname für eine bestimmte Operation spezifiziert ist, sucht die Software exakt den angegebenen Namen im Disketten-Inhaltsverzeichnis. Dort wo der Befehl es zuläßt, kann die Operation auch auf mehrere Dateien bezogen werden, indem einige Zeichen beliebig („don't care") sein dürfen. Dieses beliebige Zeichen muß an der entsprechenden Stelle durch das Zeichen ? ersetzt werden. Wenn der gesamte Dateiname oder Dateityp beliebig ist, können die dann mehrfach anzuwendenden ? durch ein einziges * abgekürzt werden. So ist beispielsweise FRED.* eine Kurzform für FRED.??? und F*.BAS für F???????.BAS. Der Ausdruck *.* bedeutet „alle Dateien".

Beispiel:

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
FINISH.BAS	FINISH.BAS		FINISH.BAS

Beispiele zu AMSDOS-Befehlen in Programmen

Wir empfehlen Ihnen zum besseren Verständnis der AMSDOS-Befehle, die folgenden Beispiele durchzuarbeiten. Bearbeiten Sie diese Programme jedoch nicht mit eingelegter Original-System/Dienstprogramm-Diskette, da Sie die Diskette beschreiben. Verwenden Sie stattdessen eine Arbeitskopie.

Sichern von Variablen und Erstellen eines Bildschirm-Abzugs

Die Dateitypen `.DAT` und `.SRN` deuten auf den Datei-Inhalt hin. Die Datei `PARAM.DAT` ist eine ASCII-Datei ohne Vorspann, `FLAGDUMP.SRN` ist eine AMSDOS-Binär-Datei mit Vorspann. Die Programme befinden sich auf der System/Dienstprogramm-Diskette im ungeschützten ASCII-Format.

Das erste Beispiel (`EX1.BAS`) zeichnet die britische Nationalflagge (Union Jack) und sichert den gesamten Bildschirm-Inhalt auf Diskette. Geben Sie zum Starten `RUN "EX1"` ein. AMSDOS sucht automatisch den Dateityp und ergänzt `.BAS`. Die Informationen zum Bildschirm-Abzug, wie Bildschirm-Modus, Paletten-Farbe und Name der diese Informationen enthaltenden Datei werden in einer Parameter-Datei gespeichert. Dieses Beispiel illustriert die Anwendung von Daten-Dateien zum Speichern (mittels `WRITE`) von Programm-Variablen (`dumpfile$`) und Konstanten (`1`) zur Verwendung durch andere Programme.

```
10 dumpdatei$="flagge.srn"
20 MODE 1:BORDER 0
30 DIM farbe(2)
40 FOR i=0 TO 2
50 READ farbe(i): REM Holen der Farben aus den DATA-Sta
tements
60 INK i,farbe(i)
70 NEXT
80 ON ERROR GOTO 430
90 OPENIN "param.dat" REM testen ob Datei vorhanden ist
100 CLOSEIN:ON ERROR GOTO 0
110 IF errnum=32 AND DERR=146 THEN CLS:GOTO 160 ' Datei
 gibt es nicht
120 CURSOR 1:PRINT "Wollen Sie die alte Version der Dat
ei   ueberschreiben? (J/N) ";
130 a$=INKEY$:ON INSTR(" JN",UPPER$(a$)) GOTO 130,150,1
40:GOTO 130
140 PRINT a$:PRINT "Programm beendet":END
150 PRINT a$:CURSOR 0
160 OPENOUT "param.dat"
170 WRITE #9,dumpdatei$,1: REM Absichern Dateiname und
Modus
180 FOR i=0 TO 2
190 WRITE #9,farbe(i): REM Absichern der 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
```

Das zweite Beispiel (EX2.BAS) ist ein Programm zur Darstellung eines Bildschirm-Abzugs 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 Bildschirm-Abzug muß auf einer bekannten Position im Speicher gesichert werden, da die Darstellung sonst verschoben wird. Dies wird durch das speichernde Programm sichergestellt, indem ein MODE-Befehl ausgeführt und der Bildschirm nicht „gescrollt" wird.

```
10 DIM farbe(15): REM Felder fuer 16 Farben
20 OPENIN "param.dat"
30 INPUT #9, dateiname$,bildschirmmodus
40 i=0
50 WHILE NOT EOF
60 INPUT #9,farbe(i)
70 INK i,farbe(i)
80 i=i+1
90 WEND
100 CLOSEIN
110 MODE bildschirmmodus:BORDER 0
120 LOAD dateiname$
```

Zusammenfassung der externen AMSDOS-Befehle

|A

```
|A
```

Setzt das Standard-Laufwerk auf Laufwerk A.
Äquivalent zu `|DRIVE` mit dem Parameter A.
(Das Hauptlaufwerk im Computer ist Drive A).

|B

```
|B
```

Setzt das Standard-Laufwerk auf Laufwerk B.
Äquivalent zu `|DRIVE` mit dem Parameter B.
(Das Hauptlaufwerk im Computer ist Drive A).

|CPM

```
|CPM
```

Umschalten auf alternative Disketten-Umgebung durch Laden des Betriebssystems von einer System-Diskette. Das mit dem Computer ausgelieferte Betriebssystem ist CP/M 2.2. Das Kommando funktioniert nicht wenn sich im Laufwerk des Computers keine Systemdiskette mit CP/M befindet.

|DIR

`|DIR [,`<String – Ausdruck>`]`

```
|DIR,"*.BAS"
```

Anzeigen des Disketten-Inhaltsverzeichnisses sowie des freien Speicherplatzes. Wenn der <String-Ausdruck> fehlt, wird `*.*` angenommen.

|DISC

```
|DISC
```

Entspricht den beiden Befehlen `|DISC.IN` und `|DISC.OUT`.

|DISC.IN

```
|DISC.IN
```

Verwendet die Diskette als Datei-Eingabe-Medium.

|DISC.OUT

```
|DISC.OUT
```

Verwendet die Diskette als Datei-Ausgabe-Medium.

|DRIVE

```
|DRIVE,<String-Ausdruck>
```

```
    |DRIVE,"A"
```

Setze das Standard-Laufwerk. Dieser Befehl arbeitet nur, wenn AMSDOS die Diskette im angegebenen Laufwerk lesen kann.

|ERA

```
|ERA,<String-Ausdruck>
```

```
    |ERA,"*.BAK"
```

Alle Dateien, die mit dem Dateinamen übereinstimmen und nicht nur lesbar sind, werden gelöscht. Wild cards werden unterstützt.

|REN

```
|REN,<String-Ausdruck>,<String-Ausdruck>
```

```
|REN,"NEUNAME.BAS","ALTNAME.BAS"
```

Gib einer Datei einen neuen Namen. Eine Datei mit dem neuen Namen darf noch nicht existieren. Wild cards sind nicht erlaubt.

|TAPE

```
|TAPE
```

Entspricht den beiden Befehlen |TAPE.IN und |TAPE.OUT.
Nicht benötigt, wenn ein externer Cassettenrecorder angeschlossen ist.

|TAPE.IN

```
|TAPE.IN
```

Verwendet die Cassette als Datei-Eingabe-Medium. (Bei Anschluß eines externen Cassettenrecorders).

|TAPE.OUT

```
|TAPE.OUT
```

Verwendet die Cassette als Datei-Ausgabe-Medium. (Bei Anschluß eines externen Cassettenrecorders).

|USER

```
|USER,<Integer-Ausdruck>
|USER,3
```

Legt fest, für welchen der bis zu 16 verschiedenen Abschnitte des Inhaltsverzeichnisses (von 0 bis 15) die Disk-Funktionen (z.B. CAT, LOAD, |DIR, usw.) ausgeführt werden 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","0:BEISPIEL.BAS" eine Datei von USER Nummer 0 auf USER Nummer 15, ohne daß der Name der Datei (BEISPIEL.BAS) geändert wird.

Kopieren von Dateien

AMSDOS-Dateien mit Vorspann

Diese Dateien können in der CP/M-Umgebung mittels `PIP` oder `FILECOPY` (siehe Teil 2 dieses Kapitels) kopiert werden. Jede durch AMSDOS und mit Vorspann erzeugte Datei ist vollständig von Diskette auf Diskette, von Diskette auf Cassette und umgekehrt kopierbar. Im allgemeinen kann jedoch der Inhalt der Datei durch CP/M-Programme nicht bearbeitet werden.

ASCII-Dateien

Durch AMSDOS erzeugte Dateien ohne Vorspann sind in der Regel ASCII-Dateien und sowohl kopierfähig, als auch durch CP/M-Programme bearbeitbar. Insbesondere sollte die Möglichkeit freigehalten werden, ASCII-Programm-Dateien, ASCII-Daten-Dateien und ASCII-Text-Dateien frei zwischen AMSDOS- und CP/M-Programmen austauschen zu können.

Nur lesbare Dateien

Die Katalogisierungs-Funktion unter CP/M bietet die Möglichkeit, Dateien als nur lesbar und/oder unsichtbar zu deklarieren. Die Attribute sind nur in einer CP/M-Umgebung möglich, werden jedoch durch AMSDOS unterstützt. Weitere Einzelheiten finden sich in Teil 2 dieses Kapitels über CP/M (siehe `STAT`).

Prozeduren zum Kopieren von Dateien

Die folgenden Tabellen zeigen die Datei-Kopier-Möglichkeiten zwischen Cassette (falls angeschlossen) und Diskette. Es wird davon ausgegangen, daß kein weiteres Diskettenlaufwerk angeschlossen ist. Geschützte BASIC-Programme und Binär-Dateien (wie z.B. Maschinen-Code von Video-Spielen) sind nicht kopierfähig, es sei denn, die Lade-Adressen sind bekannt. Weitere Hinweise zu den Programmen `FILECOPY`, `CLOAD` und `CSAVE` befinden sich in Teil 2 dieses Kapitels, das auch Informationen zum CP/M-Dienstprogramm `PIP` enthält.
Das Kopieren von Dateien einer Diskette auf eine andere wird in einem System mit zwei Laufwerken durch Zuhilfenahme des CP/M-Dienstprogramms PIP vereinfacht.

Kopierziel:	Kopierquelle: *Schneider BASIC auf Cassette**	*ASCII-Daten auf Cassette**	*Binär-Daten auf Cassette**

*Schneider BASIC auf Cassette**	`|TAPE`
	`LOAD 'FILE'`
	<Cassetten-Wechsel>
	`SAVE "FILE"`
	`|DISC`		

*Binär auf Cassette**			`H=HIMEM`
			`|TAPE`
			`MEMORY<a>-1`
			`LOAD "FILE"`
			<Cassetten-Wechsel>
			`SAVE "FILE"`
			`,B,<a>,<l> [,<s>]`
			`|DISC`
			`MEMORY H`
			<Note 2>

*ASCII auf Cassette**	`|TAPE`
	`LOAD "FILE"`
	<Cassetten-Wechsel>
	`SAVE "FILE",A`
	`|DISC`
		`|CPM`
		`CLOAD`
		`"FILE",`
		`TEMP`
		<Cassetten-Wechsel>
		`CSAVE TEMP`
		`,"FILE"`
		`ERA TEMP`
		`AMSDOS`
		<Note 1>	

*Schneider BASIC auf Diskette**	`|TAPE`
	`LOAD "FILE"`
	`|DISC`
	`SAVE "FILE"`		

ASCII auf Diskette	`|TAPE`
	`LOAD "FILE"`
	`|DISC`
	`SAVE "FILE",A`
		`|CPM`
		`CLOAD "FILE"`
		`AMSDOS`	

*Binär auf Diskette**			`H=HIMEM`
			`|TAPE`
			`MEMORY<a>-1`
			`LOAD "FILE"`
			`|DISC`
			`SAVE "FILE"`
			`,B,<a>,<l> [,<s>]`
			`MEMORY H`
			<Note 2>

*Datei mit Vorspann

<Note 1> Erfordert freien Disketten-Speicherplatz für die temporäre Datei „TEMP".
<Note 2> <a> ist die Anfangsadresse der Datei, <l> die Länge und <s> optionell die Startadresse.

Kopierziel:	Kopierquelle: *Schneider BASIC auf Diskette**	*ASCII-Daten auf Diskette**	*AMSDOS Binär-Daten auf Diskette**	*Alle anderen auf Diskette*

*Schneider BASIC auf Cassette**	`LOAD"FILE"`
	`|TAPE`
	`SAVE"FILE"`
	`|DISC`			

*Binär auf Cassette**			`H=HIMEM`
			`MEMORY<a>-1`
			`LOAD"FILE"`
			`|TAPE`
			`SAVE"FILE"`
			`B,<a>,<l>[,<s>]`
			`|DISC`
			`MEMORY H`
			<Note 2>	

*ASCII auf Cassette**	`LOAD"FILE"`
	`|TAPE`
	`SAVE"FILE",A`
	`|DISC`
	`|CPM`
	`CSAVE FILE`
	`AMSDOS`
		`|CPM`
		`CSAVE FILE`
		`AMSDOS`
				`|CPM`
				`CSAVE FILE`
				`AMSDOS`
				<Note 3>

*Schneider BASIC auf Diskette**	`LOAD"FILE"`
	<Disketten-Wechsel>
	`SAVE"FILE"`
	– oder –
	`|CPM`
	`FILECOPY FILE`
	<Anweisungen befolgen>
	`AMSDOS`			

ASII auf Diskette	`LOAD"FILE"`
	`SAVE"FILE",A`
		`|CPM`
		`FILECOPY FILE`
		<Anweisungen befolgen>
		`AMSDOS`		

*AMSDOS Binär auf Diskette**			`|CPM`
			`FILECOPY FILE`
			<Anweisungen befolgen>
			`AMSDOS`	

Alle anderen auf Diskette				`|CPM`
				`FILECOPY FILE`
				<Anweisungen befolgen>
				`AMSDOS`

* Datei mit Vorspann

<Note 2> <a> ist die Anfangsadresse der Datei, <l> die Länge und <s> optionell die Startadresse.

<Note 3> Ziel-Datei kann nicht direkt durch BASIC verwendet werden. Diese Option ist als günstiges Transport- oder Backup-Medium nützlich. Die Datei kann mit `CLOAD"FILE"` zurück auf eine Diskette kopiert werden.

Übersicht der AMSDOS-Fehlermeldungen

Wenn AMSDOS irgendeinen Befehl nicht ausführen kann, wird eine Fehlermeldung ausgegeben. Gibt es Probleme mit der Hardware, so folgt der Fehlermeldung die Frage:

`Retry, Ignore or Cancel?`

`R (Retry)` verursacht eine Wiederholung der Aktion, möglicherweise nachdem der Anwender die entsprechenden Vorsichtsmaßnahmen durchgeführt hat.

`I (Ignore)` verursacht eine Fortsetzung der Bearbeitung, als wäre kein Problem aufgetreten. Dies führt häufig zu unerwarteten und möglicherweise unbequemen Folgen.

`C (Cancel)` verursacht die Aufhebung der Aktion, was häufig zu einer erneuten Fehlermeldung führt.
Weitere Hinweise finden sich in Teil 6 des Kapitels „Zum Nachschlagen".

`Unknown command`

Der Befehl ist nicht richtig geschrieben.

`Bad command`

Der Befehl kann aus bestimmten Gründen nicht ausgeführt werden (Syntax- oder Hardware-Fehler).

`<Dateiname> already exists`

Versuch eine Datei mit einem bereits existierenden Namen zu versehen.

`<Dateiname> not found`

Datei existiert nicht.

`Drive <Laufwerk>: directory full`

Kein Platz für neue Einträge in das Disketten-Inhaltsverzeichnis vorhanden.

`Drive <Laufwerk>: disc full`

Kein Platz für neue Daten auf der Diskette vorhanden.

`Drive <Laufwerk>: disc changed, closing <Dateiname>`

Diskette wurde gewechselt, ohne auf ihr gespeicherte Dateien zu schließen.

`<Dateiname> 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" gesetzt werden.

`Drive <Laufwerk>: disc missing`

Keine Diskette im Laufwerk oder Diskette nicht korrekt eingelegt bzw. drehbar. Die Diskette sollte entnommen und erneut eingelegt werden. Anschließend `R` eingeben.

`Drive <Laufwerk>: disc is write protected`

Es wurde versucht, auf eine Diskette mit offenem Schreibschutz-Loch zu schreiben. Sie entnehmen die Diskette, schließen das Schreibschutz-Loch, legen die Diskette erneut ein und drücken `R`.

`Drive <Laufwerk>: read fail`

Hardware-Fehler beim Lesen der Diskette. Entnehmen Sie die Diskette und legen Sie sie erneut ein. Daraufhin `R` drücken.

`Drive <Laufwerk>: write fail`

Hardware-Fehler beim Beschreiben der Diskette. Entnehmen Sie die Diskette und legen Sie sie erneut ein. Daraufhin `R` drücken.

`Failed to load CP/M`

Lesefehler beim Laden von CP/M mit dem `|CPM`-Befehl oder Verwendung einer ungültigen System-Diskette ohne CP/M.

Teil 2: CP/M

In diesem Kapitel behandelte Themen sind:

- ★ Booten von CP/M
- ★ Konfigurations-Sektor
- ★ Direkter Konsol-Modus
- ★ Transiente Programme
- ★ Peripherie-Verwaltung

Mit Ihrem CPC664 wird CP/M 2.2 geliefert. CP/M ist ein Disketten-Betriebssystem. Es ist ein spezielles Programm, welches Ihnen die volle Nutzung des CPC664 Systems ermöglicht. Da CP/M für sehr viele unterschiedliche Computer angeboten wird, sind entsprechend viele Anwender-Programmpakete verfügbar, die Ihnen vielfältige Möglichkeiten zur Wissenserweiterung und Vergrößerung Ihres Erfahrungshorizonts bieten.

Nähere Einzelheiten zu CP/M, einschließlich Anleitungen zum Entwickeln eigener Programme auf dem CPC664 sind in SW 159 „CP/M-Handbuch" enthalten.

Einleitung

Das CP/M-Betriebssystem bietet eine Benutzer-Schnittstelle zur Disketten-Hardware, d.h. eine Möglichkeit mit dem Computer zu kommunizieren sowie Dateien und Peripherie zu manipulieren.

Die wichtigste Schnittstelle ist der sogenannte „Direkte Konsol-Modus", der durch die Prompts `A>` oder `B>` gekennzeichnet ist. Es stehen bestimmte „eingebaute" Befehle zur Verfügung, aber die meisten Funktionen erhalten Sie durch das Laden und Starten „transienter Programme". Sie werden als „transient" bezeichnet, da sie während der Anwendung Bestandteil des Computers (und nicht auf Diskette) sind.

Das Schneider Disketten-System gibt sowohl die Standard CP/M-Fehlermeldungen als auch spezielle Hardware-Fehlermeldungen aus (siehe AMSDOS-Fehlermeldungen in Teil 1 dieses Kapitels).

CP/M-System-Spuren

Der Haupt-Bestandteil von CP/M befindet sich auf den äußersten beiden Spuren der Diskette. Der Computer lädt CP/M in zwei Schritten von diesen Spuren in den Speicher.

Der AMSDOS-Befehl `|CPM` lädt zuerst den ersten Sektor von Spur 0 in den CPC464. Auf einer System-Diskette ist dieser Sektor als Programm ausgelegt, welches dann die verbleibenden System-Spuren in den Speicher lädt. Es werden verschiedene Tests ausgeführt, um zu überprüfen, ob die System-Spuren gültige CP/M-Software enthalten. Weiterhin wird die Speicher-Ladeadresse errechnet.

Konfigurations-Sektor

Während der Aktivierung von CP/M bzw. beim Ladevorgang werden verschiedene System-Parameter von einem speziellen Sektor innerhalb der System-Spur geladen. Diese Parameter beinhalten die „System bereit" (Sign-On)-Meldung, sowie spezielle Tastatur-Codes usw. Das `SETUP`-Dienstprogramm ermöglicht die Anpassung des Konfigurations-Sektors an Ihre speziellen Anforderungen.

Kontrol-Codes von der Konsole

In der CP/M-Umgebung dienen verschiedene Tasten-Funktionen zur Steuerung des Programm-Ablaufs. Diese Tasten-Funktionen ersetzen die Wirkung der `[ESC]`ape-Taste im Schneider-BASIC, obgleich einige Anwender-Pakete die `[ESC]`ape-Taste funktionell erweitern.

`[CTRL] S` stoppt die Bildschirm-Ausgabe unter CP/M. Geben Sie ein beliebiges Zeichen zur Fortsetzung ein.

`[CTRL] C` übergibt am Anfang einer Zeile die Steuerung an den direkten Konsolen-Modus. Viele Dienst- und Anwender-Programme erkennen diese Eingabe als Programm-Abbruch.

`[CTRL] P` Umschaltung der Druckerausgabe, d.h. Ein- und Ausschaltung der Bildschirm-Ausgabe auf den Drucker.

`[CTRL] Z` Textende.

Einloggen einer Diskette

Wenn keine besonderen Vorkehrungen durch das CP/M-Programm getroffen worden sind (wie z.B. durch `FILECOPY`), erlaubt CP/M solange keine Schreibzugriffe auf Diskette, bis sie dem System durch ein „Log-In" bekannt ist. Der Format-Typ der Disketten (SYSTEM, DATEN oder IBM) ist ebenfalls nur nach dem „Log-In" bestimmbar. Diese Aufbereitung findet für das Laufwerk im Computer (Laufwerk A) immer dann statt, wenn CP/M in den direkten Konsolen-Modus zurückkehrt oder nach dem Erscheinen von `A>` bzw. `B>` `[CTRL]` C eingegeben wird. Für ein weiteres Laufwerk (Laufwerk B) erfolgt das Log-In nach dem ersten Zugriff, sofern für Laufwerk A bereits ein Log-In durchgeführt ist.

Wenn eine Diskette ohne vorheriges Log-In beschrieben wird, erscheint die Fehlermeldung

```
Bdos Error on <Laufwerk>: R/O.
```

Zum Fortsetzen drücken Sie bitte eine beliebige Taste. Wenn die ausgewechselte Diskette ebenfalls ein anderes Format besitzt, tritt ein Lese- oder Schreibfehler auf. Zum Fortsetzen drücken Sie bitte C.

Direkte Konsol-Befehle

Es gibt fünf direkte Konsol-Befehle (residente Befehle), die nach dem Erscheinen von A> bzw. B> eingegeben werden können. Es sind dies SAVE, DIR, ERA, REN und TYPE. Der erste von ihnen – SAVE – ist nur für spezielle Anwendungen vorgesehen.

CP/M-Fehlermeldungen sind in der Regel kurz und bestehen gewöhnlich aus einer Wiederholung des fehlerhaften Befehls oder Dateinamens mit nachfolgendem Fragezeichen (?). Geben Sie den Befehl daraufhin mit den entsprechenden Korrekturen erneut ein.

Dateinamen

Viele Befehle übernehmen Dateinamen als Parameter. Dort wo Dateinamen auftreten, können auch Wild-Cards enthalten sein (siehe Abschnitt „Wild Cards" im Teil 1 dieses Kapitels). Alle Dateinamen werden in Großbuchstaben umgewandelt. Direkte Konsol-Befehle und die meisten Dienstprogramme erfordern keinen Einschluß der Dateinamen in Anführungszeichen (""). Nur der Cassetten-Dateiname der Dienstprogramme CLOAD und CSAVE benötigt Anführungszeichen.

Bei Verwendung eines zweiten Laufwerkes können die Dateinamen ein vorgestelltes A: oder B: aufweisen, damit CP/M das entsprechende Laufwerk verwendet.

Umschaltung des Standard-Laufwerks

Wenn Sie ein zweites Disketten-Laufwerk angeschlossen haben, kann das Standard-Laufwerk zwischen Laufwerk A und B umgeschaltet werden, indem nach Erscheinen von B> bzw. A> die Zeichen A: oder B: eingegeben werden. Die Prompt-Zeichen A> bzw. B> informieren Sie nur über die momentanen Standard-Laufwerke. Das einem Dateinamen vorgestellte A: oder B: ändert nur die Laufwerk-Auswahl für die nachgestellte Operation, setzt jedoch die Auswahl des Standard-Laufwerks nicht zurück.

DIR-Befehl

DIR listet das Inhaltsverzeichnis der Diskette auf. Die Dateinamen sind in keiner bestimmten Reihenfolge angeordnet, die Anordnung des Dateinamens in der Darstellung des Inhaltsverzeichnisses ist jedoch ein Hinweis auf die Position des Datei-Eintrags im Disketten-Inhaltsverzeichnis. Wild-Cards werden unterstützt.

`DIR`	listet alle Dateien.
`DIR B:`	listet alle Dateien von Laufwerk B.
`DIR *.BAS`	listet alle Dateien vom Typ .BAS.
`DIR B: *.BAS`	listet alle Dateien vom Typ .BAS von Laufwerk B.
`DIR PIP.COM`	listet nur die Datei PIP.COM, sofern vorhanden.

ERA-Befehl

ERA dient dem Löschen von Dateien im Inhaltsverzeichnis. Es wird nur der Eintrag im Inhaltsverzeichnis gelöscht, die Daten selbst verbleiben in den Daten-Bereichen der Diskette, bis der Speicherplatz durch eine andere Datei belegt wird.

Auf die Information kann jedoch nicht mehr zugegriffen werden. Dateinamen mit Wild-Cards werden unterstützt. Wenn der Dateiname *.* spezifiziert wird, erwartet ERA eine Bestätigung dafür, daß alle Dateien gelöscht werden sollen. ERA listet die gelöschten Dateinamen nicht auf. Wenn eine zu löschende Datei erkannt wird, die nur lesbar ist (siehe STAT), bricht die Befehlsbearbeitung ab.

`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.

REN-Befehl

REN ermöglicht die Umbenennung einer bestehenden Datei. Der neue Dateiname wird zuerst spezifiziert, gefolgt von = und dem alten, bestehenden Dateinamen. Wenn der neue Dateiname bereits besteht, wird eine Fehlermeldung ausgegeben. Wild-Cards im Dateinamen werden nicht unterstützt.

`REN HELLO.BAS=HALLO.BAS`	Benenne den Namen einer Datei von
`REN B:HELLO.BAS=HALLO.BAS`	HELLO.BAS in HALLO.BAS um.

TYPE-Befehl

TYPE stellt die spezifizierte Datei auf dem Bildschirm dar. Wenn die Datei keine ASCII-Textdatei ist, können unvorhersehbare und unerwünschte Nebenwirkungen auftreten.

`TYPE EX1.BAS`	Darstellung der Programm-Datei EX1.BAS.

Transiente Befehle

Um anspruchsvollere Datei-Bearbeitungen, als durch die direkten Konsol-Befehle ermöglicht, durchzuführen, müssen die angebotenen Dienstprogramme eingesetzt werden. Diese sind durch Eingabe des Programm-Namens (und eventueller Parameter) aufrufbar.

Diese Programme sind in verschiedene Kategorien einteilbar. Die komplette Dokumentation dieser Programme ist sehr umfangreich und befindet sich im CP/M-Handbuch SW 159.

Die Befehle SYSGEN, BOOTGEN, FILECOPY, COPYDISC, DISCCOPY, CHKDISK, DISCCHK, FORMAT, SETUP, CSAVE, CLOAD und AMSDOS-Befehle arbeiten ausschließlich auf dem CPC-System. Sie haben keine Funktion auf anderen CP/M-Systemen, obgleich andere Hersteller ähnliche Dienstprogramme (oft mit gleichem Namen) für ihre Hardware bereitstellen.

Peripherie-Verwaltung

Das Dienstprogramm PIP (Peripheral Interchange Program) ermöglicht Ihnen den Transfer von Informationen zwischen dem Computer und seiner Peripherie. Die allgemeine Form des Befehls ist

PIP <Ziel> = <Quelle>.

<Ziel> und <Quelle> können entweder Dateinamen (in <Quelle> sind Wild-Cards erlaubt) oder Peripherie-Einheiten sein.

Es können die folgenden Peripherie-Einheiten verwendet werden:

Als Quelle:	Als Ziel:
CON: Tastatur	CON: Bildschirm
RDR: serielles Interface	PUN: serielles Interface
	LST: Drucker

Beispiele:

PIP B:=A:*.COM	kopiere alle .COM-Dateien von Laufwerk A: nach Laufwerk B:
PIP SAV.BAS=EX1.BAS	anfertigen einer Kopie von EX1.BAS mit dem Namen SAV.BAS
PIP CON:=EX1.BAS	gib die Datei EX1.BAS auf dem Bildschirm aus (ähnlich dem Befehl TYPE EX1.BAS)
PIP LST:=EX1.BAS	gib die Datei EX1.BAS auf dem Drucker aus
PIP TYPEIN.TXT=CON:	Übernimm die Tastatur-Eingabe und speichere sie in der Datei TYPEIN.TXT ab. Diese Operation wird mit dem Steuercode [CTRL] Z abgeschlossen. Nach jedem [ENTER] für eine neue Zeile muß [CTRL] J (ASCII-Zeichen für Zeilenvorschub) eingegeben werden.

PIP kann nicht zum Kopieren einer Datei von einer Diskette auf eine andere in einem System mit einem Laufwerk verwendet werden. Hierfür sollte FILECOPY benutzt werden.

Kopieren von Dateien und Disketten

Kopieren einzelner Dateien

Das Dienstprogramm FILECOPY ermöglicht das Kopieren von Dateien einer Diskette auf eine andere auf dem Einzellaufwerk Ihres Computers. Hierzu müssen die Disketten gewechselt und die Anweisungen auf dem Bildschirm befolgt werden. Wenn Dateinamen mit Wild-Cards verwendet werden, fragt FILECOPY, ob tatsächlich jede Datei einzeln kopiert werden soll.

Das Programm informiert Sie über jeden kopierten Dateinamen.

FILECOPY *.BAS	Kopiere alle Dateien vom Typ .BAS
FILECOPY EX1.BAS	Kopiere die Datei EX1.BAS

Kopieren und Testen der gesamten Diskette

`DISCCOPY` (für das normale System mit einem Laufwerk) und `COPYDISK` (für Systeme mit zwei Laufwerken) ermöglichen die Anfertigung von Sicherheits-Kopien einer kompletten Diskette. Hierzu werden Anweisungen auf dem Bildschirm ausgegeben. Wenn die Ziel-Diskette nicht formatiert ist oder nicht das Format der Quellen-Diskette besitzt, wird die Diskette beim Kopieren automatisch korrekt formatiert. Die Dienstprogramme `DISCCHK` und `CHKDISC` ermöglichen den Vergleich zweier Disketten, wodurch die Übereinstimmung dieser beiden Disketten sichergestellt werden kann.

Cassetten-Dateien

Zwei Dienstprogramme erlauben den Datei-Transfer zwischen Cassette und Diskette. Mit Ausnahme spezieller Anwendungen ist es recht unwahrscheinlich, daß jemals andere als ASCII-Dateien transferiert werden.

`CLOAD` kann zwei Parameter übernehmen. Der erste ist der Quellen (-Cassetten) -Dateiname (in Anführungszeichen) und der zweite der Ziel (-Disketten) -Dateiname. Wenn der Ziel-Dateiname ausgelassen wird, erhält die Disketten-Datei den Namen der Cassetten-Datei. Ist der Quellen-Dateiname ausgelassen, so liest `CLOAD` die erste auf der Cassette gefundene Datei. Die normalen Cassetten-Meldungen werden unterdrückt, wenn das erste Zeichen des Cassetten-Dateinamens ein Ausrufungszeichen (!) ist.

Beispiel-Befehl:

```
CLOAD "MY LETTER" MYLETTER.TXT
```

`CSAVE` kann drei Parameter übernehmen. Der erste ist der Quellen (-Disketten) -Dateiname, der zweite der Ziel (-Cassetten) -Dateiname (in Anführungszeichen). Wenn der Ziel-Dateiname ausgelassen wird, erhält die Cassetten-Datei den Namen der Disketten-Datei. Die normalen Cassetten-Meldungen werden unterdrückt, wenn das erste Zeichen des Cassetten-Dateinamens ein Ausrufungszeichen (!) ist. Sind beide Dateinamen spezifiziert, kann ein dritter Parameter zur Spezifikation der Band-Schreibgeschwindigkeit (0 für 1000 Baud, 1 für 2000 Baud) verwendet werden.

Beispiel-Befehle:

```
CSAVE OUTPUT.TXT "OUTPUT TEXT" 1
CSAVE DATAFILE
```

System-Verwaltung

STAT

`STAT` ermöglicht eine anspruchsvollere Verwaltung des Inhaltsverzeichnisses und der Peripherie. Für die Dateinamen gelten alle bisherigen Regeln, einschließlich der für die Verwendung von Wild-Cards. Es stehen folgende Möglichkeiten zur Verfügung:

Anzeigen des Disketten-Status und freien Speicherplatzes:

```
STAT
STAT A:
STAT B:
```

Anzeigen der erweiterten Inhaltsverzeichnis-Information für eine bestimmte Datei:

```
STAT *.COM
STAT EX1.BAS
```

Setzen des „Nur lesbar“ -Status einer Datei, so daß sie nicht versehentlich gelöscht oder überschrieben werden kann:

```
STAT *.COM $R/O
STAT EX1.BAS $R/O
```

Setzen des „Schreib/Lese“-Status einer Datei (Umkehrung des „Nur lesbar“-Status):

```
STAT *.COM $R/W
STAT EX1.BAS $R/W
```

Setzen des „System“-Status einer Datei, so daß sie für die Auflistung des Inhaltsverzeichnisses bzw. Datei-Kopierprogramme unsichtbar wird und für alle anderen Anwendungen sichtbar bleibt:

```
STAT *.COM $SYS
STAT SECRET.BAS $SYS
```

Setzen des „Inhaltsverzeichnis“-Status einer Datei (Umkehrung des „System“-Status):

```
STAT *.COM $DIR
STAT SECRET.BAS $DIR
```

Setup

Mit diesem Dienstprogramm können die charakteristischen Eigenschaften der CPC664-Tastatur, des Disketten-Laufwerks und der seriellen Schnittstelle neu definiert, sowie verschiedene Funktionen beim ersten Laden von CP/M aufgerufen werden. Nach Abschluß der Bearbeitung wird der Konfigurations-Sektor auf den neuesten Stand gebracht. Das Programm ist Menü-gesteuert. Wenn ein bestimmter Bildschirminhalt keine Änderungen erfordert, kann auf den nächsten umgeschaltet werden, indem auf die Frage

```
Is this correct (Y/N):=
```

mit `Y` geantwortet wird.

Das Programm kann mit `[CTRL]C` abgebrochen werden. Wenn alle Änderungen durchgeführt sind, erscheint

```
Do you want to update your system disc (Y/N):=
```

auf dem Bildschirm, wodurch Sie den bestehenden Konfigurations-Sektor (bei Eingabe von `N`) unverändert lassen können.

```
Do you want to restart CP/M (Y/N):=
```

ermöglicht Ihnen bei Eingabe von `Y` das Laden und Austesten der neuen Konfiguration.

Um einen Konfigurations-Sektor auf eine andere Diskette zu kopieren, verwenden Sie bitte `BOOTGEN` (siehe unten) oder laden `SETUP` von der Quellen-Diskette, beantworten alle Fragen mit `Y` und legen vor Beantwortung der vorletzten Frage 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,[,\,],>,-` verwendet 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 Automatik-Start eines bestimmten Programms. Für die `[ENTER]`-Taste sind die beiden Zeichen `↑M` zu verwenden. Zum Automatik-Start von `STAT` sollte der Befehls-Puffer `STAT↑M` enthalten.

`Sign-on String`

Dies ist die beim ersten Laden von CP/M am oberen Bildschirmrand erscheinende Meldung. Für einen Wagenrücklauf und Zeilenvorschub ist `↑J↑M` zu verwenden. Der erste Teil der Standard-Meldung setzt die geeigneten Bildschirm- und Bildschirmrand-Farben für den 80 Zeichen/Zeile-Modus und sollte exakt kopiert werden.

`Keyboard translations`

Hierdurch können den Tasten neue ASCII-Werte zugeordnet werden, ähnlich dem Schneider BASIC-Befehl `"KEY DEF"`. Die erforderlichen Parameter sind die Tasten-Nummer und der zuzuordnende ASCII-Wert. Eine Darstellung der Tasten-Nummern finden Sie ganz rechts auf Ihrem Computer, oder im Kapitel „Zum Nachschlagen".

`Keyboard expansions`

Gleicher Effekt wie beim BASIC-Befehl `"KEY"`.

AMSDOS

Dieses Programm beendet die Steuerung durch CP/M und übergibt sie an Schneider BASIC, unter dem die AMSDOS-Disketten-Befehle ablauffähig sind.

Disketten erstellen

FORMAT

Der Computer unterstützt drei Disketten-Formate. Eines dieser Formate hat zwei Varianten.

Das gebräuchlichste Format ist das System-Format und wird durch den Standard-Befehl `FORMAT` erzeugt. Die System-Spuren werden von der Diskette, die das Programm `FORMAT.COM` enthält, gelesen und automatisch auf die Ziel-Diskette geschrieben.

Die anderen Formate werden durch Hinzufügen eines einzelnen, als Parameter dienenden Zeichens an den Befehl erzeugt (durch Leerzeichen getrennt):

Für Daten-Format:	`FORMAT D`
Für IBM-Format:	`FORMAT I`
Für Vendor-Format:	`FORMAT V`

** WARNUNG **

Das Lizenzabkommen für Ihre CP/M-Diskette (die eine elektromagnetisch aufgezeichnete Serien-Nummer enthält) erlaubt nur die Verwendung auf einem einzigen Computer-System. Dies heißt, die Weitergabe einer Kopie der CP/M-Diskette mit Ihrer Serien-Nummer an andere Personen ist verboten. Da jede von Ihnen erzeugte System-Diskette Ihr CP/M enthält, dürfen Sie die System-Format-Diskette weder verkaufen bzw. tauschen noch irgendwie anders in Umlauf bringen. Stattdessen müssen Sie eine Vendor-Diskette formatieren, die mit dem System-Format, mit Ausnahme der leeren System-Spuren, identisch ist, und die entsprechende Software mit FILECOPY oder PIP kopieren.

Stellen Sie sicher, daß die auf diese Weise kopierte Software nicht ebenfalls urheberrechtlich geschützt oder Bestandteil eines Lizenzabkommens ist.

Wenn Sie Software auf einer Diskette im Vendor-Format erwerben, sollten Sie – der bequemen Handhabung wegen – diese entweder mit FILECOPY oder PIP auf eine System-Diskette kopieren oder die Diskette durch Hinzukopieren Ihres CP/Ms in eine System-Diskette umwandeln. Dies kann mit den Befehlen BOOTGEN und SYSGEN durchgeführt werden.

LESEN SIE BITTE AUFMERKSAM DEN ABSCHNITT ÜBER DAS END-ABNEHMERLIZENZABKOMMEN IN DIESEM HANDBUCH

MOVCPM

Manchmal ist die Erstellung einer CP/M-Version notwendig, die nicht in den Standard-Speicherbereich geladen werden soll. Dies ist dann der Fall, wenn Sie bestimmte Speicherbereiche für andere Zwecke verwenden und CP/M auf einen unteren Speicherbereich kopieren wollen. CP/M kann auf jeden Speicherplatz in 256 Byte-Schritten ausgerichtet werden. Die Position ist durch einen Größen-Parameter im Bereich von 64 bis 179 spezifiziert. Dieser Parameter gibt die Anzahl der für CP/M und transiente Programme verfügbaren 256 Byte-Bereiche an.

Das neu ausgerichtete CP/M kann nun entweder mit `SYSGEN` auf die System-Diskette kopiert oder mit einem durch das `MOVCPM`-Programm angebotenen Befehl gesichert werden. Die Syntax des Befehls ist `MOVCPM <Größe> *`. `MOVCPM 178*` erzeugt eine gegenüber Standard-CP/M um 256 Bytes nach unten verschobene Version. Standard-CP/M ist mit der maximal möglichen Größe 179 erzeugt.

SYSGEN

SYSGEN schreibt das Ergebnis eines `MOVCPM`-Befehls auf die System-Spur einer System- oder Vendor-Diskette. Es gibt drei Optionen:

`SYSGEN *` schreibt das durch einen unmittelbar vorhergehenden `MOVCPM`-Befehl erzeugte CP/M.

`SYSGEN <Dateiname>` liest die spezifizierte Datei, die vermutlich nach einem `MOVCPM`-Befehl gesichert wurde, und sichert sie auf den System-Spuren (z.B. `SYSGEN CPM44.COM`).

SYSGEN ohne Parameter erfragt die Quellen- und Ziel-Diskette und kopiert daraufhin die System-Spuren. Diese Option sollte angewendet werden, um eine Vendor-Diskette zur System-Diskette aufzurüsten.

BOOTGEN

BOOTGEN erfragt die Quellen- und Ziel-Diskette und kopiert Sektor 1, Spur 0 (mit dem Lader) sowie den Konfigurations-Sektor von einer Diskette auf eine andere. BOOTGEN sollte zur Aufrüstung einer Vendor-Diskette zur System-Diskette oder zum Schreiben eines neu erstellten Konfigurations-Sektors auf mehrere Disketten verwendet werden.

Fortgeschrittene Programmierung

Die folgenden Programme dienen speziellen Anwendungen. Es wird empfohlen, im CP/M-Handbuch SW 159 oder ähnlichen Unterlagen nachzuschlagen.

ASM	8080 Assembler.
DDT	8080 Assembler-Code Testhife.
DUMP	Dienstprogramm für hexadezimale Datei-Ausgabe.
ED	Einfacher Text-Editor.
SUBMIT	Stapel-Verarbeitung (Batch) aus dem Konsol-Modus.
XSUB	Stapel-Verarbeitung (Batch) durch transiente Programme.

KAPITEL 6
Einführung in LOGO

Dieser Abschnitt führt anhand von Beispielen in die Programmiersprache LOGO ein und bietet eine Übersicht über die verfügbaren Befehle. Er sollte jedoch nicht als erschöpfendes Lehrbuch oder Nachschlagewerk betrachtet werden, hierfür steht ein LOGO-Führer (SW 160) zur Verfügung.

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, unabhängig davon, ob Sie zuvor programmiert haben oder nicht. LOGO ist eine leistungsfähige Programmiersprache, die aufgrund ihrer leichten Erlernbarkeit sehr schnell populär wurde.

Sie benutzen Prozeduren als Bausteine für die Erstellung von LOGO-Programmen. Dr. Logo besteht selbst 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 Schildkröte mit einfachen Befehlen über den Bildschirm geführt werden.

Dr. LOGO

Dr. Logo ist eine besondere LOGO-Version, die an den Schneider Personal Computer angepaßt wurde, um das Programmieren zu erleichtern. Es wurden umfangreiche Erweiterungen eingebaut, um die großartigen Sound-Möglichkeiten des CPC664 auszunutzen. Die Programm-Bearbeitung wurde durch das Einbeziehen des Cursortasten-Feldes erleichtert.

Start von Dr. LOGO

Um mit Dr. Logo zu arbeiten, legen Sie zunächst eine Kopie der Seite 2 Ihrer Master-Diskette in das Laufwerk. In Teil 10 des Grundlagen-Kurses wird die Duplizierung einer Master-Diskette beschrieben.

Drücken Sie daraufhin [**CTRL**], [**SHIFT**] und [**ESCAPE**] gleichzeitig, um den Computer zurückzusetzen.

Geben Sie `|CPM` [**ENTER**] ein und LOGO wird automatisch geladen.

Die folgende Dr. Logo-Startmeldung erscheint dann auf dem Bildschirm:

```
Welcome to

Amstrad LOGO V1.1
Copyright (c) 1983/4, Digital Research
Pacific Grove, California

Dr.Logo is a trademark of Digital Research

Serial No. 6002-1232-123456


Please Wait
```

Der Begrüßungstext verschwindet daraufhin, und es erscheint eine Prompt-Meldung (**?**) auf dem Bildschirm. Das Fragezeichen weist darauf hin, daß Dr. Logo eine Eingabe von der Tastatur erwartet.

Erste Versuche

Tippen Sie (in Kleinbuchstaben)

`fd 60` [**ENTER**]

ein, und es erscheint eine Schildkröte, die 60 Einheiten vorwärts marschiert und dabei von ihrem Startpunkt bis zum Endpunkt eine Linie zeichnet. Der Bildschirm wird daraufhin gelöscht und in einen Graphik- und einen Text-Bereich (am unteren Rand des Bildschirms) unterteilt. Letzterer enthält die Prompt-Meldung.

Dr. Logo wird des öfteren den Bildschirm-Bereich neu organisieren, um je nach Ihren Bedürfnissen einen größeren Text- oder Graphik-Bereich einzuteilen.
Geben Sie nun

```
rt 90 [ENTER]
```

ein, woraufhin sich die Schildkröte um 90 Grad nach rechts dreht. Ab jetzt setzen wir voraus, daß Sie nach jeder Eingabe selbständig **[ENTER]** drücken.
Wenn Sie jetzt

```
fd 60
```

eingeben, wird eine zweite Linie im rechten Winkel zur ersten Linie gezeichnet.

Experimentieren Sie mit den einfachen Anweisungen `fd`, `bk` (Abk. für back), `rt` und `lt` (Abk. für 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 zu erledigen. Sie werden Ihre ersten Prozeduren wahrscheinlich schreiben, indem Sie bereits bestehende Dr. Logo-Prozeduren 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`, die den Bildschirm löscht und die Schildkröte an ihren Ausgangspunkt zurücksetzt.

Schreiben einer einfachen Prozedur

Es ist leicht einzusehen, daß die Anweisungen

```
fd 60 rt 90
```

ein Quadrat mit der Seitenlänge 60 ergeben, wenn sie 4 mal wiederholt werden.

Der gleiche Effekt kann durch eine 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, geben Sie

```
to quadrat
repeat 4 [fd 60 rt 90]
end
```

ein. Dr. Logo wird ab jetzt 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 vorhandenen 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. Sie werden bemerkt haben, daß der Variablen `:seite` ein Doppelpunkt vorausgeht, der eine Variable von einem Befehl unterscheidbar macht.

Wenn wir die Prozedur `verschiedenequadrate :seite` benutzen, brauchen wir für die Variable `:seite` einen Wert. Der Befehl `verschiedene quadrate 150` wird also ein Quadrat mit einer Seitenlänge von 150 Einheiten zeichnen.

Versuchen Sie zwei Prozeduren zusammenzufügen und sehen Sie, was geschieht. Geben Sie dazu beispielsweise

```
cs verschiedenequadrate 100 rt 45
verschiedenequadrate 150
```

ein, und die Schildkröte zeichnet 2 verschieden große Quadrate, die im Winkel von 45 Grad gegeneinander versetzt sind.

Beachten Sie, wie Sie Dr. Logo darauf aufmerksam macht, daß eine Folge von Befehlen auf mehr als eine Bildschirmzeile aufgeteilt wurde.

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 neue Prozedur mit dem Namen dreieck:

```
to dreieck
repeat 3 [fd :ecke rt 120]
end
```

Wir können die Prozedur testen, indem wir

```
make "ecke 100
dreieck
```

eingeben. Wenn wir den Wert von `:ecke` wissen wollen, brauchen wir nach der Prompt-Meldung (`?`) nur `:ecke` eingeben und Dr. Logo gibt den Wert auf dem Bildschirm aus.

Zum Abschluß soll die Variable `:ecke` in einer Prozedur verwendet werden, um ein Muster zu zeichnen. Beachten Sie, wie der Wert von `:ecke` durch das Aufaddieren auf den ursprünglichen Wert ansteigt. Dadurch wird der Wert der Variablen sowie das Muster immer größer.

```
to muster
dreieck lt 60 dreieck rt 60
make "ecke :ecke + 4
muster
end
make "ecke 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.
Die verwendbaren Editiertasten sind:

- die Cursortasten [↑] [↓] [←] [→], die den Cursor jeweils ein Zeichen bzw. eine Zeile weiterbewegen.
- die obigen Cursortasten gleichzeitig mit [**CTRL**] gedrückt, wodurch der Cursor eine Seite vor oder zurück bzw. an den linken oder rechten Rand einer Zeile bewegt werden kann.
- [**CLR**] löscht das Zeichen, auf dem der Cursor steht, [**DEL**] löscht das Zeichen links vom Cursor.
- [**ENTER**] teilt Dr. Logo mit, daß Sie mit der Editierung einer Zeile fertig sind bzw. eine neue Zeile der Prozedur eingegeben werden soll.
- [**ESC**] bedeutet Abbruch, [**COPY**] teilt Dr. Logo mit, daß die Bearbeitung einer Prozedur abgeschlossen ist.

Wenn Sie die Befehle oder neue Prozeduren eingeben wollen, editieren Sie einfach den Text auf dem Bildschirm. Alle oben nicht aufgeführten Zeichen werden direkt auf der Cursorposition in den Text eingefügt.

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 den vorher beschriebenen Befehlen die Prozedur ändern. Versuchen Sie die Prozedurmuster zu ändern, indem Sie

```
ed "muster
```

eingeben und mit den Editiertasten experimentieren.

Nach Abschluß der Eingaben und Drücken von [**ESC**] bricht Dr. Logo die Bearbeitung ab und übergibt die Original-Version der Prozedur. Geben Sie jetzt

```
ed "muster
```

ein, ändern Sie `4` in `8`, steigen Sie mit [**COPY**] aus, lassen Sie die Prozedur nochmals ablaufen und sehen Sie sich den geänderten Bildschirmausdruck an. Vergessen Sie nicht, den Anfangswert von `:ecke` zu setzen.

Bedienungshinweise

Der Arbeitsbereich von Dr. Logo ist in Knoten (Nodes) unterteilt. Sie können sich durch Eingabe von `nodes` die Anzahl der freien Knoten anzeigen lassen. Gelegentlich, wenn fast alle Knoten belegt sind, reorganisiert Dr. Logo den Speicherbereich. Die Schildkröte pausiert dann. Sie können Dr. Logo direkt auffordern, den Speicherplatz zu reorganisieren, indem Sie `recycle` eingeben. Hiermit können Sie die Bearbeitung fortsetzen, falls Dr. Logo gemeldet hat, daß keine Knoten mehr frei sind.

Bevor Sie Dr. Logo starten, sollten Sie sicherstellen, daß hinreichend Speicherplatz auf der Diskette frei ist, um Ihre Prozeduren zu sichern. Sie können den freien Speicherplatz unter AMSDOS mit dem `CAT`-Befehl abfragen (siehe Teil 7 des Grundlagenkurses).

Schauen Sie sich das nächste Kapitel an und probieren Sie einige der 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 beenden wollen, geben Sie `bye` ein.

Verzeichnis der Dr. LOGO-Grundprozeduren

Dieser Abschnitt faßt die Dr. Logo-Grundprozeduren (teilweise mit Beispiel) alphabetisch geordnet zusammen.

Wörter- und Listen-Verarbeitung

(In den folgenden Beispielen werden die Prompt-Zeichen `?` und `>` angezeigt).

ascii

Gibt den ASCII-Wert des ersten Zeichens eines eingegebenen Wortes aus.

```
?ascii "G
71
?ascii "g
103
```

bf

(but first) Gibt alle Zeichen, bis auf das erste des eingegebenen Ausdrucks, aus.

```
?bf "klang
lang
?bf [1 2 3]
[2 3]
```

bl

(but last) Gibt alle Zeichen, bis auf das letzte des eingegebenen Ausdrucks, aus.

```
?bl "klang
klan
?bl [1 2 3 4]
[1 2 3]
```

char

Gibt das dem eingegebenen ASCII-Wert entsprechende Zeichen aus.

```
?char 83
S
```

count

Gibt die Anzahl der Elemente eines eingegebenen Ausdrucks aus.

```
?count "sechs
5
?count [0 1 2 3]
4
```

emptyp

Gibt TRUE aus, wenn der eingegebene Ausdruck ein leeres Wort oder eine leere Liste ist; andernfalls wird FALSE ausgegeben.

```
?emptyp "
TRUE
?emptyp []
TRUE
?emptyp [x]
FALSE
?make "x []
?emptyp :x
TRUE
```

first

Gibt das erste Element eines eingegebenen Ausdrucks aus.

```
?first "zebra
z
?first [1 2 3]
1
```

fput

(firstput) Gibt einen neuen Ausdruck aus, indem der erste eingegebene Ausdruck das erste Element des zweiten Ausdrucks wird.

```
?fput "k "lang
klang
?fput 1 [2 3]
[1 2 3]
```

item

Gibt das angegebene Element eines Ausdrucks aus.

```
?item 4 "klang
n
```

list

Gibt die eingegebenen Ausdrücke als Liste aus; bewahrt die äußeren Klammern (vergleichen Sie mit `se`).

```
?(list 1 2 3 4)
[1 2 3 4]
?list "grosse [arme]
[grosse [arme]]
?(list)
[]
```

se

(sentence) Gibt die eingegebenen Ausdrücke als Liste aus; entfernt die äußeren Klammern (vergleichen Sie mit `list`).

```
?make "befehls_liste rl
repeat 4 [fd 50 rt 90]
?run (se "cs :befehls_liste "ht
```

Beachten Sie, daß man den Unterstrich zwischen `befehls` und `liste` durch Drükken von [**SHIFT**] `0` bekommt.

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

cos

Gibt den Cosinus des in Grad eingegebenen Winkels aus.

```
?cos 60
0.500000000017049
```

int

Gibt den ganzzahligen Anteil einer eingegebenen Zahl aus.

```
?int 4/3
1
```

random

Gibt eine positive Zufallszahl aus, die kleiner als die eingegebene Zahl ist.

```
?random 20
```

sin

Gibt den Sinus des in Grad eingegebenen Winkels aus.

```
?sin 30
0.500000000017049
```

+

Gibt die Summe der eingegebenen Zahlen aus.

```
?+ 2 2
4
? 2+2
4
```

−

Gibt die Differenz der 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 eingegebenen Zahlen aus.

```
?/ 25 5
5
?25/5
5
```

4.10.3 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 Audrücke falsch sind.

```
?or "TRUE "FALSE
TRUE
?or (3=4)(1=1)
TRUE
```

=

Gibt TRUE aus, wenn zwei eingegebene Ausdrücke gleich sind; anderenfalls wird FALSE ausgegeben.

```
?= "LOGO "LOGO
TRUE
?1=2
FALSE
```

>

Gibt TRUE aus, wenn die erste Eingabe größer als die zweite ist; anderenfalls wird FALSE ausgegeben.

```
?> 19 20
FALSE
?20>19
TRUE
```

<

Gibt TRUE aus, wenn die erste Eingabe kleiner als die zweite ist; anderenfalls wird FALSE ausgegeben.

```
?< 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 eingegebenen Variablen den nachfolgenden Wert zu.

```
?make "seite 50
? :seite
50
```

Prozeduren

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
?po "x
X is 3
```

pots

(print out titles) Zeigt die Namen und Bezeichnungen aller Prozeduren im Arbeitsspeicher auf dem Bildschirm an.

```
?pots
```

to

Kennzeichnet den Beginn einer Prozedur – Definition.

```
?to quadrat
>repeat 4 [fd 50 rt 90]
>end
quadrat defined
```

Editier-Befehle

ed

(edit) Laden der angegebenen Prozedur(en) und/oder Variable(n) in den Bildschirm-Editierbereich.

```
?ed "quadrat
```

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
```

pr

(print) Stellt die Eingabe(n) auf dem Text-Bildschirm dar, entfernt die äußeren Klammern der Listen und führt nach der letzten Eingabe einen Zeilenrücksprung aus (vergleichbar mit `show` und `type`).

```
?pr [a b c]
a b c
```

setsplit

Setzt die Anzahl der Zeilen im geteilten Bildschirmbereich.

```
?setsplit 10
```

show

Stellt die Eingabe(n) auf dem Text-Bildschirm dar, behält die äußeren Klammern der Listen bei und führt nach der letzten Eingabe einen Zeilenrücksprung aus (vergleichen Sie mit `pr` und `type`).

```
?show [a b c]
[a b c]
```

ts

(text screen) Wählt den vollen Text-Bildschirm aus.

```
?ts
```

type

Stellt die Eingabe(n) auf dem Bildschirm dar, entfernt die äußeren Klammern der Listen und führt nach der letzten Eingabe keinen Zeilenrücksprung aus (vergleichen Sie mit `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 das Koordinaten-System von Schneider BASIC verwendet.
Mit anderen Worten, alle Bildschirmpositionen werden zum nächsten geradzahligen Bildschirmpunkt hin gerundet. Rot, Grün und Blau können Werte von 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), ausgerichtet nach 0 (Nord). Der Zeichenstift ist aktiviert.

```
?rt 90 fd 50
?cs
```

dot

Zeichnet einen Punkt, der durch die Eingabekoordinaten – Liste festgelegt wird, in der momentanen Pen-Farbe.

```
?dot [50 10]
```

fence

Errichtet einen Rahmen, der die Bewegungen der Schildkröte auf den sichtbaren Graphik-Bildschirm begrenzt. `w i n d o w` löscht die Grenze wieder.

```
?fence
?fd 300
Turtle out of bounds
```

fs

(full screen) Wählt den gesamten Graphik-Bildschirm aus.

```
?fs
```

pal

(palette) Gibt die einem Pen zugeordneten Farbwerte für Rot, Grün und Blau aus.

```
?pal 2
[0 2 2]
```

setpal

(set palette) Setzt die Pen-Farbpalette. Damit werden die Farbwerte für Rot, Grün und Blau für einen Pen eingegeben.

```
?setpal 3 [1 1 2]
?pal 3
[1 1 2]
```

sf

(screen facts) Gibt Informationen über den Graphik-Bildschirm aus. Das <Format> dieser Informationen ist
[<Hintergrundfarbe> <Bildschirm-Status> <Fenstergröße> <Fenster-Status> <Scrunch>]
wobei <Hintergrundfarbe> den Hintergrund-Pen bezeichnet (immer 0). <Bildschirm-Status> beinhaltet entweder `s s` (geteilter Bildschirm), `f s` (voller Bildschirm) oder `t s` (Text-Bildschirm). <Fenstergröße> gibt die Anzahl der Text-Zeilen im Bildschirm-Textfenster an. <Fenster-Status> zeigt den `w i n d o w`-, `w r a p`- oder `f e n c e`-Modus an. <Scrunch> ist immer mit 1 belegt.

```
?sf
[0 SS 5 FENCE 1]
```

ss

(split screen) Gibt ein Textfenster auf dem Graphik-Bildschirm aus.

```
?ss
```

window

Erlaubt der Schildkröte (nach einem `wrap` – oder `fence` – Ausdruck) außerhalb des sichtbaren Graphik-Bildschirms zu zeichnen.

```
?fence fd 300
Turtle out of bounds
?window
?fd 300
```

wrap

Läßt die Schildkröte auf der gegenüberliegenden Seite des Graphik-Bildschirms erscheinen, 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
```

ht

(hide turtle) Macht die Schildkröte unsichtbar. Geschwindigkeit und Zeichnung sind sichtbar.

```
?ht
?cs fd 50
?st
```

lt

(left) Dreht die Schildkröte um den in Grad angegebenen Winkel 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) Setzt den Zeichenstift (Pen) der Schildkröte auf die Hintergrund-Farbe 0; 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 beendet das Zeichnen.

```
?fd 30
?pu
?fd 30
?pd fd 30
```

px

(pen reverse) Veranlaßt die Schildkröte, die Farbe der vorher gezeichneten Pixel auf Ihrem Weg ins Reverse oder logische Farb-Komplement zu wandeln.

```
?fd 20 pu fd 20
?pd setpc 3 fd 20
?px
?bk 80
?fd 80
?pd bk 100
```

rt

Dreht die Schildkröte um den in Grad angegebenen Winkel nach rechts.

```
?rt 90
```

seth

(set heading) Dreht die Schildkröte in die absolute Richtung, die durch den in Grad angegebenen *Winkel* festgelegt wird. Positive Werte drehen die Schildkröte in Uhrzeigerrichtung, negative Werte drehen die Schildkröte entgegen der Uhrzeigerrichtung.

```
?seth 90
```

setpc

(set pen colour) Setzt die Farbe des Zeichenstiftes (Pen) der Schildkröte auf die angegebene Zeichenstiftnummer. 0 ist die Hintergrund-Farbe.

```
?setpc 1
```

setpos

(set position) Setzt die Schildkröte auf die Position, die durch die Eingabekoordinaten festgelegt wurde.

```
?setpos [30 20]
```

st

(show turtle) Macht die Schildkröte sichtbar, wenn Sie vorher unsichtbar war.

```
?ht
?fd 50
?st
```

tf

(turtle facts) Gibt Informationen über die Schildkröte aus. Das <Format> dieser Information ist
[<xcor> <ycor> <Richtung> <Pen-Status> <Pen-Farben #n> <sichtbar>]
wobei <xcor> die x-Koordinate und <ycor> die Y-Koordinate der Schildkröte angibt. <Richtung> gibt die Ausrichtung der Schildkröte (Kompaß-Richtung) an. <sichtbar> ist TRUE, wenn die Schildkröte sichtbar ist. <Pen-Status> kann den Zustand PD (Pen aktiv), PE (Pen löschen), PX (Pen reverse) oder PU (Pen inaktiv) annehmen. <Pen-Farben #n> ist die Farbnummer des Zeichenstiftes (Pen).

```
setpos [15 30]
?rt 60
?setpc 3
?pe
?ht
?tf
[15 30 60 PE 3 FALSE]
```

Speicherplatz-Verwaltung

er

(erase) Löscht die angegebene(n) Prozedur(en) im Speicher.

```
?er "quadrat
```

ern

(erase name) Löscht die angegebene(n) Variable(n) im Speicher.

```
?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
```

recycle

Reorganisiert den Arbeitsspeicher und stellt soviele Knoten wie möglich zur Verfügung.

```
?recycle
?nodes
```

Eigenschaftslisten

glist

(get list) Zeigt eine Liste aller im Arbeitsbereich gespeicherten Ausdrücke an, welche die eingegebene Eigenschaft in ihrer Eigenschaftsliste enthalten.

```
?glist ".DEF
```

gprop

(get property) Zeigt das dem spezifizierten Ausdruck und der eingegebenen Eigenschaft zugeordnete Eigenschaftswort an.

```
?make "hoehe "72
?gprop "hoehe ".APV
72
```

plist

(property list) Zeigt die Eigenschaftsliste des eingegebenen Ausdrucks an.

```
?plist "hoehe
[.APV 72]
```

pprop

(put property) Übergibt die beiden eingegebenen Eigenschaften an die Eigenschaftsliste des spezifizierten Ausdrucks.

```
?pprop "auto ".APV "farbe
? :auto
farbe
```

remprop

(remove property) Entfernt die eingegebene Eigenschaft aus der Eigenschaftsliste des spezifizierten Ausdrucks.

```
?remprop "auto ".APV
```

Disketten-Dateien

dir

(directory) Gibt eine Liste mit den Dr. Logo-Dateinamen der Standard-Diskette oder der angegebenen Diskette aus. Wild-cards werden akzeptiert.

```
?dir
[STEINE STERNE MUSTER]
?dir "b:
[MITTEL HILFSPRO ADRESSEN]
?dir "?TE?????
[STEINE STERNE]
```

(* Wild cards werden von Dr. LOGO nicht unterstützt).

load

Liest die angegebene Datei von der Diskette in den Arbeitsspeicher.

```
?load "STERNE
?load "b:HILFSPRO
```

save

Schreibt den Inhalt des Arbeitsspeichers unter dem angegebenen Dateinamen auf die Diskette.

```
?save "HILFSPRO
```

Tastatur, Joystick

buttonp

(button pressed) Gibt TRUE aus, wenn der Knopf des angegebenen Joysticks gedrückt ist. Die zwei möglichen Zustände sind 0 oder 1.

```
?to feuer
>label "schleife
>if (buttonp 0) [pr[feuer 0!]]
>if (buttonp 1) [pr]feuer 1!]]
>go "schleife
>end
```

Die Position des Joysticks wird durch `paddle` ermittelt.

keyp

Gibt TRUE aus, wenn ein Zeichen der Tastatur gedrückt wurde und darauf wartet, eingelesen 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
```

Die Feuer-Knöpfe werden durch `buttonp` getestet.

rc

(read character) Gibt das erste Zeichen aus, das auf der Tastatur gedrückt wurde.

```
?make "taste rc
```

(Drücken Sie 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 "befehls_liste rl
repeat 4[fd 50 rt 90]
?:befehls_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 Schneider-Version von Dr. Logo und sind ähnlich den Befehlen in Schneider-BASIC. Lesen Sie zur näheren Information im Teil 9 des Grundlagenkurses.

`sound` übergibt einen Ton an die Ton-Warteschlange.
Das Format ist
[<Kanal-Status> <Tonperiode> <Dauer> <Lautstärke> <Lautstärken-Hüllkurve> <Ton-Hüllkurve> <Geräuschfolge>]
Die nach <Dauer> folgenden Parameter sind wahlweise.

```
?sound [1 20 50]
```

env

Setzen einer Lautstärken-Hüllkurve. Das Format ist
[<Hüllkurven-Nummer> <Hüllkurven-Abschnittslänge(n)>]

```
?env [1 100 2 20]
?sound [1 200 300 5 1]
```

ent

Setzen einer Ton-Hüllkurve. Das Format ist
[<Hüllkurven-Nummer> <Hüllkurven-Abschnittslänge(n)>]

```
?ent [1 100 2 20]
?sound [1 200 300 5 1 1]
```

release

Gibt einen Ton-Kanal frei, der durch einen Sound-Befehl gesperrt wurde. Die freizugebenden Kanäle werden folgendermaßen gekennzeichnet:

Eingabewert	Freigegebener Kanal
0	Keiner
1	A
2	B
3	B und A
4	C
5	C und A
6	C und B
7	C und B und A

```
?release 1
```

Ablauf-Steuerung

bye

Beendet die Bearbeitung unter Dr. Logo.

```
?bye
```

co

Beendet eine Unterbrechung, die durch `pause`, **[CTRL] Z** oder ERRACT verursacht wurde.

```
?co
```

go

Führt die Zeile innerhalb der momentanen Prozedur aus, deren Marke (label) dem eingegebenen Wort entspricht.

```
>go "schleife
```

if

Führt eine von zwei möglichen Befehlslisten in Abhängigkeit vom eingegebenen Ausdruck aus. Die eingegebene Befehlsliste darf nur Literale enthalten und muß in Klammern gesetzt werden.

```
>if (a>b)[pr [a ist groesser]]
>[pr [b ist groesser]]
```

label

Kennzeichnet die Zeile, die nach dem go-Befehl mit dem eingegebenen Wort ausgeführt werden soll.

```
>label "schleife
```

op

(output) Der Befehl beendet die Ausführung der Prozedur und gibt die Eingabe als Wert zurück.

```
?op [Ergebnis]
```

repeat

Führt die angegebene Befehlsliste sooft aus, wie die Zahl angibt.

```
?repeat 4[fd 50 rt 90]
```

run

Führt die eingegebene Befehlsliste aus.

```
?make "befehls_liste[fd 40 rt 90]
?run :befehls_liste
```

stop

Beendet die Ausführung der momentanen Prozedur und kehrt zu TOPLEVEL (mit der Prompt-Meldung **?**) oder der aufrufenden Prozedur zurück.

```
?stop
```

wait

Unterbricht die Ausführung der Prozedur für eine bestimmte Zeit. Unterbrechungszeit = Eingabewert * 0,22 Sekunden.

```
?wait 20
```

Ausnahmefälle

catch

Erkennt Fehler und besondere Umstände, die während der Ausführung der eingegebenen Befehlslisten auftreten.

```
>catch "error[+[][]]
>pr [Hier bin ich]
Hier bin ich
```

error

Gibt eine Liste aus, welche die zuletzt aufgetretenen Fehler enthält.

```
>catch "error[do.until.error]
>show error
```

pause

Unterbricht die Ausführung der momentanen Prozedur und stellt den Interpreter oder Editor zur Verfügung.

```
>if :groesse>5 [pause]
```

throw

Führt die Zeile aus, die durch den eingegebenen Namen des vorherigen catch-Ausdrucks gekennzeichnet ist.

```
?throw "TOPLEVEL
```

System-Grundprozeduren

`.contents`	Gibt den Inhalt des Dr. Logo Symbol-Speichers aus.
`.deposit`	Legt den zweiten Eingabewert unter der absoluten Speicheradresse, die durch den ersten Eingabewert festgelegt wurde, ab.
`.examine`	Gibt den Inhalt der angegebenen absoluten Speicheradresse aus.

System-Variablen

ERRACT

Ist ERRACT TRUE, bewirkt sie beim Auftreten eines Fehlers eine Pause und kehrt daraufhin nach TOP LEVEL zurück.

FALSE

System-Wert.

REDEFP

Ist REDEFP TRUE, können Grundprozeduren neu definiert werden.

TOPLEVEL

`throw "TOPLEVEL` beendet alle momentan bearbeiteten Prozeduren.

TRUE

System-Wert.

System-Eigenschaften

.APV

Associated property value; der Wert einer globalen Variablen.

.DEF

Definition einer Prozedur.

.PRM

Identifiziert eine Grundprozedur (Primitive).

Für Ihre Notizen:

Kapitel 7 Zu Ihrer Information ... ...Wenn Sie etwas Zeit haben.

Dieser Abschnitt befaßt sich mit einer Vielzahl von Informationen, die für Sie sehr nützlich sind, während Sie sich mit diesem Computer vertraut machen.

Behandelte Themen:

- ★ Cursorposition und Kontrollzeichen
- ★ Interrupt-Funktionen
- ★ ASCII-Zeichensatz und graphische Zeichen
- ★ Tastenbezeichnungen
- ★ Klangeffekte (Sound)
- ★ Fehlermeldungen
- ★ BASIC-Schlüsselwörter
- ★ Planer
- ★ Anschlüsse
- ★ Drucker
- ★ Joysticks
- ★ Disketten-Organisation
- ★ Speicher

Weitere Informationen über BASIC und Firmware für den CPC664 finden Sie in den Schneider Handbüchern SW 945 bzw. SW 946.

Teil 1:
Cursorpositionen und Kontrollzeichen

In Anwenderprogrammen kann es vorkommen, daß der Textcursor außerhalb des augenblicklichen Ausgabebereiches (`WINDOW`) positioniert wird. Verschiedene Ausgabeoperationen führen dazu, daß der Cursor vor ihrer Ausführung auf eine zulässige Position zurückgebracht wird. Dies sind

- Schreiben eines Zeichens
- Ausgeben des Cursors
- Ausführung der Kontrollzeichen, die in der auf den nachfolgenden Seiten angegebenen Liste mit einem Stern gekennzeichnet sind.

Die Rückführung des Cursors auf eine zulässige Position geschieht folgendermaßen:

a) Steht der Cursor außerhalb des rechten Rands, so wird er auf das erste Zeichen der nächsten Zeile gesetzt.

b) Steht der Cursor außerhalb des linken Rands, so wird er auf das letzte Zeichen der vorherigen Zeile gesetzt.

c) Steht der Cursor oberhalb des Ausgabebereichs, so wird der ganze Bereich um eine Zeile nach unten gerollt und der Cursor in die erste Zeile gesetzt.

d) Steht der Cursor unterhalb des Ausgabebereiches, so wird der ganze Bereich um eine Zeile nach oben gerollt und der Cursor in die letzte Zeile gesetzt.

Die Prüfungen und Korrekturen finden in der oben angegebenen Reihenfolge statt. Die zulässigen Cursorpositionen können den Wert 0 haben oder negativ sein, was bedeutet, daß sie links oder oberhalb des Ausgabebereichs liegen.

Zeichenwerte im Bereich von 0 bis 31 (siehe Anhang III) erzeugen kein abdruckbares Zeichen, wenn sie ausgegeben werden, werden aber als Kontrollzeichen interpretiert (sie können bei unvorsichtiger Anwendung den Rechner zum Stillstand bringen). Einige Zeichen ändern die Bedeutung des oder der nachfolgenden Zeichen, da diese dann als Parameter verstanden werden.

Kontrollzeichen, die auf dem graphischen Bildschirm ausgegeben werden, erscheinen als Sonderzeichen (Siehe `TAG`-Kommando in Kapitel 8). Andere Steuerzeichen, wie z. B. (`&07'Klingel'`), werden bei Eingabe über Tastatur (`[CTRL]G`) als Sonderzeichen ausgegeben, führen dagegen ihre zugeordnete Funktion bei der Ausgabe über ein `PRINT`-Kommando aus (`PRINT CHR$(&07`).

Die mit * gekennzeichneten Kontrollzeichen führen den Cursor vor ihrer Ausführung auf eine zulässige Position zurück, – können ihn aber danach in einer unzulässigen zurücklassen. Die Kontrollcodes und Ihre Bedeutung sind in der nachfolgenden Liste zuerst mit ihrem hexadezimalen (`&XX`) und dann mit ihrem Dezimalwert beschrieben.

Zusätzliche Kontrollzeichen-Kommandos: Nicht immer über die Tastatur mit der [CTRL]-Taste eingebbar

Wert	Name	Parameter	Bedeutung
&00 0	NUL		Kein Einfluß. Ignoriert.
&01 1	SOH	0...255	Drucken des als Parameterwert angegebenen Symbols. So können Symbole mit den Werten 0 bis 31 ausgedruckt werden.
&02 2	STX		Textcursor ausschalten.
&03 3	ETX		Textcursor einschalten. Anmerkung: BASIC verhindert das Überschreiben des Cursors; wird nur beim Warten auf die Eingabe über Tastatur aufgehoben.
&04 4	EOT	0...2	Bildschirmmode setzen. Parameterauswertung: MOD 4. Gleichwertig mit MODE Kommando.
&05 5	ENQ	0...255	Ausgeben des als Parameter angegebenen Symbols auf die Graphikcursorposition.
&06 6	ACK		Textbildschirm aktivieren (siehe &15,NAK, nächste Seite).
&07 7	BEL		Klingel ertönen lassen. Die Tonausgabewarteschlangen werden geleert.
&08 8	* BS		Cursor ein Zeichen zurücksetzen.
&09 9	* TAB		Cursor ein Zeichen vorsetzen.
&0A 10	* LF		Cursor eine Zeile nach unten setzen.
&0B 11	* VT		Cursor eine Zeile nach oben setzen.
&0C 12	FF		Bildschirm löschen und Cursor in die linke obere Ecke setzen. Gleichwertig mit CLS-Kommando.
&0D 13	* CR		Cursor auf die erste Schreibstelle der aktuellen Schreibzeile setzen.

&0E	14		SO	0...15	Farbstift für den beschreibbaren Bildschirmbereich setzen. Parameterauswertung: MOD 16. Gleichwertig mit PAPER-Kommando.
&0F	15		SI	0...15	Farbstift für die Schreib- und Zeichenausgabe setzen. Parameterauswertung MOD 16. Gleichwertig mit PEN-Kommando.
&10	16	*	DLE		Zeichen auf Cursorposition löschen. Schreibstelle mit Schreibflächenfarbe füllen.
&11	17	*	DC1		Zeile von Beginn bis einschließlich der Cursorposition löschen. Schreibstellen mit aktueller Schreibflächenfarbe füllen.
&12	18	*	DC2		Zeile einschließlich der Cursorposition bis zum Ende löschen. Schreibstellen mit aktueller Schreibflächenfarbe auffüllen.
&13	19	*	DC3		Beginn der Schreibfläche bis einschließlich der Cursorposition löschen. Schreibstellen mit aktueller Schreibflächenfarbe füllen.
&14	20	*	DC4		Einschließlich der Cursorposition bis zum Ende der Schreibfläche löschen. Schreibstellen mit aktueller Schreibflächenfarbe füllen.
&15	21		NAK		Textbildschirm abschalten. Bildschirm reagiert solange nicht mehr, bis ein ACK(06) ausgegeben wird.
&16	22		SYN	0...1	Parameterauswertung: MOD 2; Transparentmodus mit 0 ausschalten, 1 = einschalten.
&17	23		ETB	0...3	Parameterauswertung: MOD 4; Graphikfarbstiftmodus setzen: 0 = Normal 1 = 'XOR''' 2 = 'AND''' 3 = 'XOR'''
&18	24		CAN		Papier- und Schreibstiftfarben austauschen.
&19	25		EM	0...255 0...255 0...255 0...255 0...255 0...255 0...255 0...255	Matrix für benutzereigene Zeichen setzen. Gleichwertig mit Symbol-Kommando. Benötigt 9 Parameter. Der erste gibt an, welches Zeichen nun zu setzen ist. Die restlichen acht beschreiben die Matrix; das erste (linke) Bit im ersten Byte beschreibt den obersten linken Punkt in der Zeichendarstellung; das letzte (rechte) Bit im letzten Byte stellt den untersten rechten Punkt in der Zeichendarstellung dar.

&1A 26	SUB	1...80 1...80 1...80 1...80	Bildschirmteilbereich (WINDOW) setzen. Gleichwertig mit WINDOW-Kommando. Benötigt 4 Parameter. Die ersten zwei definieren die linken und rechten Ränder, wobei der kleinere Wert den linken und der größere Wert den rechten Rand beschreibt. Die letzten beiden Parameter definieren die oberen und unteren Ränder; der kleinere den oberen und der größere den unteren Rand.
&1B 27	ESC		Kein Einfluß. Ignoriert.
&1C 28	FS	0...15 0...31 0...31	Farbenpaar für Farbstift setzen. Gleichwertig mit INK-Kommando. Der erste Parameter (MOD 16) definiert den Farbstift. Die beiden nächsten definieren die Farben (MOD 32).
&1D 29	GS	0...31 0...31	Farbenpaar für den nicht beschreibbaren Bildschirmrand. Gleichwertig mit BORDER-Kommando. Die zwei Parameter (MOD 32) definieren die beiden Farben.
1E RS	RS		Cursor in die obere linke Ecke des Bildschirmteilbereichs setzen.
&1F 31	US	1...80	Cursor auf die angegebene Position setzen. Gleichwertig mit LOCATE-Kommando. Der erste Parameter definiert die Spalte und der zweite die Zeile.

Die Steuerung des CPC664 wird von einem anspruchsvollen Echtzeit-(real-time)-Betriebssystem übernommen. Das Betriebssystem steuert den 'Verkehrsfluß' durch den Computer, von der Eingabe bis zur Ausgabe.

Es stellt hauptsächlich das Bindeglied zwischen der Hardware und dem BASIC-Interpreter dar, z. B. bei der Funktion der wechselnden Schreibstiftfarben übergibt BASIC die Parameter und das Betriebssystem übernimmt die Weiterbehandlung: Ein Teil analysiert was zu tun ist, der andere überwacht den zeitlichen Ablauf.

Das Betriebssysem einer Maschine wird als sogenannte 'Firmware' bezeichnet und besteht aus Maschinencode-Routinen, die BASIC mit seinen Kommandos aufruft.

Während es über BASIC (außer mit dem Kommando ON BREAK GOSUB) fast unmöglich ist, den CPC664 zu blockieren, ist es relativ leicht, durch Zugriffe auf das Betriebssystem eine Situation zu erzeugen, die Sie nur durch vollständiges Zurücksetzen des Rechners wieder aufheben können. Dabei gehen natürlich das ganze Programm und alle Daten verloren!

Falls Sie versuchen sollten, mit POKE-Kommandos die internen Speicherinhalte zu verändern oder mit CALL-Aufrufen die Systemroutinen aufzurufen, sichern Sie vorher Ihr Programm, da es zerstört werden könnte.

Eine ausführliche Beschreibung der Betriebssystem-Firmware des CPC664 steht in einem zusätzlichen Handbuch, das weit über den Umfang dieser Einführung hinausgeht.

Teil 2: Unterbrechungen

Der CPC664 benützt die Unterbrechungsmöglichkeiten des Z80 in großem Maß, so daß ein Betriebssystem mit Multitasking-Funktionen (mehrere Anweisungen werden gleichzeitig bearbeitet) gegeben ist. Ein Beispiel dafür sind die AFTER- und EVERY-Kommandos, die in Kapitel 8 beschrieben sind. Die Priorität der Unterbrechungen hat folgende Reihenfolge:

Break [ESC] [ESC]
Zeitgeber 3
Zeitgeber 2 (und die drei Tonkanalwarteschlangen)
Zeitgeber 1
Zeitgeber 0

Unterbrechungen sollen erst dann eingebaut werden, wenn man die möglichen Zustände der verschiedenen Variablen zum Zeitpunkt der Unterbrechung gut durchdacht hat. Die Unterbrechungsbehandlungsroutine sollte ungewollte Veränderungen der Variablen des Hauptprogramms vermeiden.

Die Tonwarteschlangen haben unabhängige Unterbrechungen mit gleicher Priorität. Sobald eine solche Unterbrechung eingetreten ist, wird sie von keiner anderen nochmals unterbrochen, so daß in diesen Unterbrechungsbehandlungsroutinen anders als bei Zeitunterbrechungen die gleichen Variablen benutzt werden können.

Sind Unterbrechungen für eine Tonwarteschlange zulässig, so tritt sofort eine Unterbrechung ein, wenn die Tonwarteschlange 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. Sobald eine solche Unterbrechung eintritt, werden weitere tonabhängige Unterbrechungen gesperrt, so daß die Behandlungsroutine die Unterbrechungsmöglichkeit aktivieren muß, falls weitere Unterbrechungen gewünscht sind. Wird ein Ton ausgelöst oder der Zustand der Warteschlange getestet, werden ebenfalls weitere Unterbrechungen gesperrt.

Die Priorität der [ESC] [ESC]-Folge über allen anderen Unterbrechungen bewirkt, daß ein BASIC-Progrmm angehalten werden kann, ohne daß es verloren geht, solange nicht durch andere Anweisungen verhindert wurde, den Ablauf des Programms zu verändern.

Teil 3: ASCII und graphische Zeichen

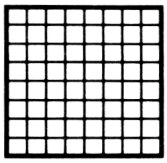
ASCII

Die nachstehende Tabelle zeigt eine Zusammenstellung des Standard-ASCII-Zeichensatzes mit dezimaler, oktaler und hexadezimaler Schreibweise, zusammen mit dem entsprechenden ASCII-Code. Jedes der CPC664-Zeichen ist auch im Detail auf den folgenden Seiten abgebildet.

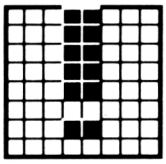
DEC	OCTAL	HEX	ASCII characters
0	000	00	NUL ([CTRL]@)
1	001	01	SOH ([CTRL]A)
2	002	02	STX ([CTRL]B)
3	003	03	ETX ([CTRL]C)
4	004	04	EOT ([CTRL]D)
5	005	05	ENQ ([CTRL]E)
6	006	06	ACK ([CTRL]F)
7	007	07	BEL ([CTRL]G)
8	010	08	BS ([CTRL]H)
9	011	09	HT ([CTRL]I)
10	012	OA	LF ([CTRL]J)
11	013	OB	VT ([CTRL]K)
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
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
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	~

Maschinen-spezifischer Zeichensatz

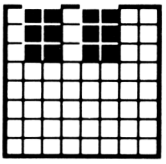
Die hier abgebildeten Zeichen sind in einer 8 x 8 Matrix dargestellt, wie sie auf dem CPC664-Bildschirm erscheinen. Anwendereigene Zeichen können für spezielle Effekte zusammengestellt werden und bündig aneinanderstoßen. Siehe anwendereigene Zeichen im Abschnitt „Wenn Sie Zeit haben"...



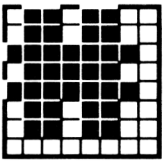
32 &H20
&X00100000



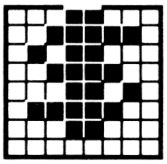
33
&H21
&X00100001



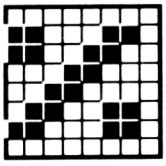
34
&H22
&X00100010



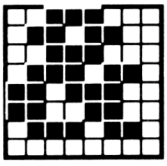
35
&H23
&X00100011



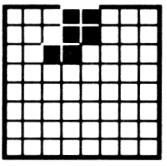
36
&H24
&X00100100



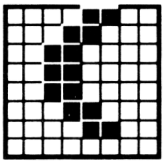
37
&H25
&X00100101



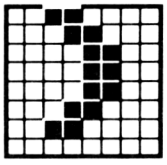
38
&H26
&X00100110



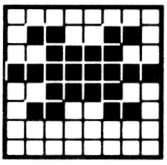
39
&H27
&X00100111



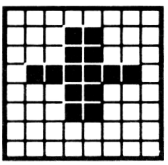
40
&H28
&X00101000



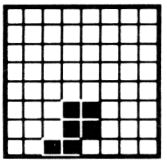
41
&H29
&X00101001



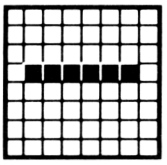
42
&H2A
&X00101010



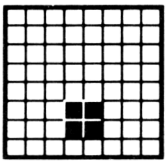
43
&H2B
&X00101011



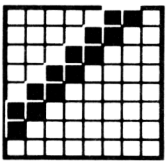
44
&H2C
&X00101100



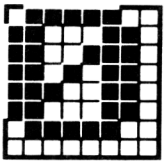
45
&H2D
&X00101101



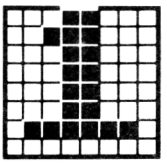
46
&H2E
&X00101110



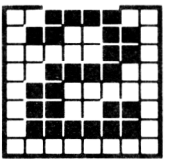
47
&H2F
&X00101111



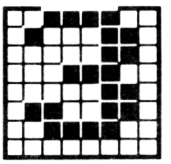
48
&H30
&X00110000



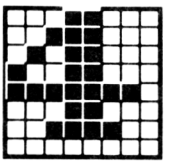
49
&H31
&X00110001



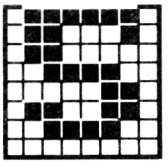
50
&H32
&X00110010



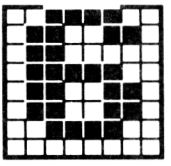
51
&H33
&X00110011



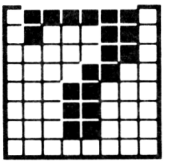
52
&H34
&X00110100



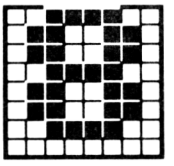
53
&H35
&X00110101



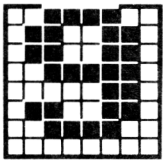
54
&H36
&X00110110



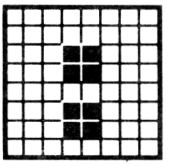
55
&H37
&X00110111



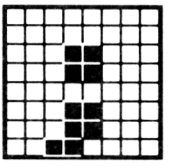
56
&H38
&X00111000



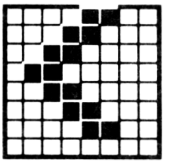
57
&H39
&X00111001



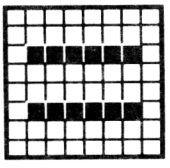
58
&H3A
&X00111010



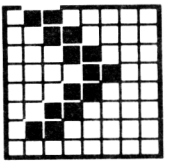
59
&H3B
&X00111011



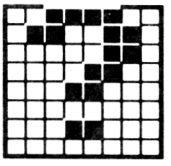
60
&H3C
&X00111100



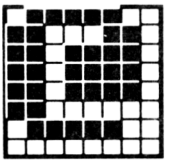
61
&H3D
&X00111101



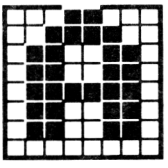
62
&H3E
&X00111110



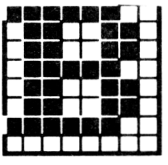
63
&H3F
&X00111111



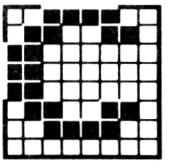
64
&H40
&X01000000



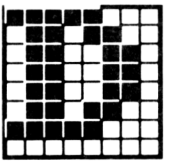
65
&H41
&X01000001



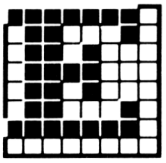
66
&H42
&X01000010



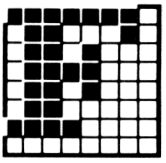
67
&H43
&X01000011



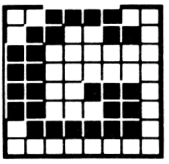
68
&H44
&X01000100



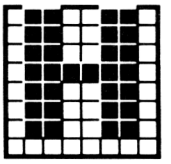
69
&H45
&X01000101



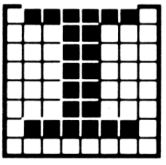
70
&H46
&X01000110



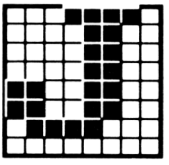
71
&H47
&X01000111



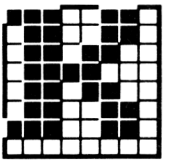
72
&H48
&X01001000



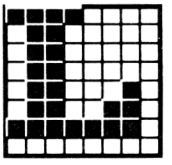
73
&H49
&X01001001



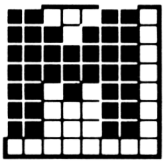
74
&H4A
&X01001010



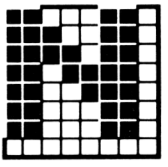
75
&H4B
&X01001011



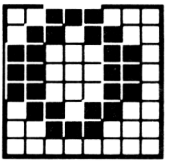
76
&H4C
&X01001100



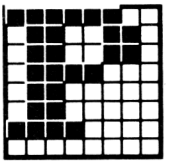
77
&H4D
&X01001101



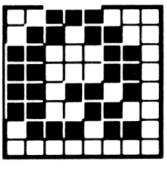
78
&H4E
&X01001110



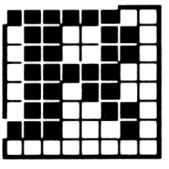
79
&H4F
&X01001111



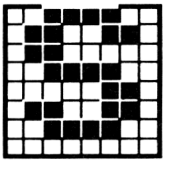
80
&H50
&X01010000



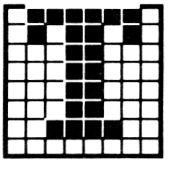
81
&H51
&X01010001



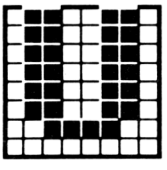
82
&H52
&X01010010



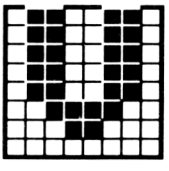
83
&H53
&X01010011



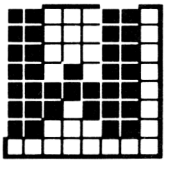
84
&H54
&X01010100



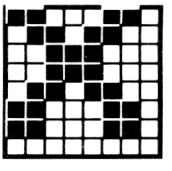
85
&H55
&X01010101



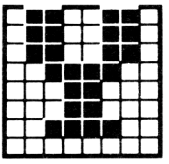
86
&H56
&X01010110



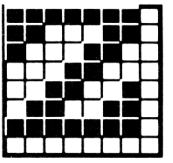
87
&H57
&X01010111



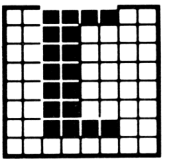
88
&H58
&X01011000



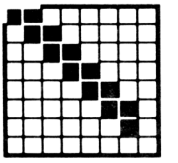
89
&H59
&X01011001



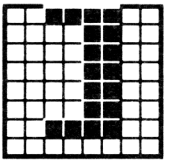
90
&H5A
&X01011010



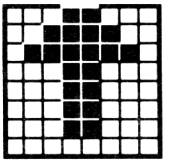
91
&H5B
&X01011011



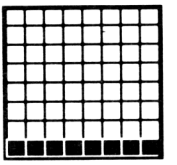
92
&H5C
&X01011100



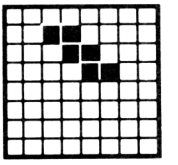
93
&H5D
&X01011101



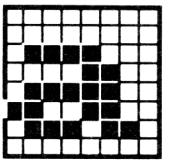
94
&H5E
&X01011110



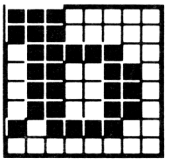
95
&H5F
&X01011111



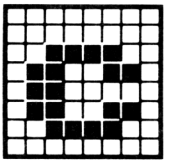
96
&H60
&X01100000



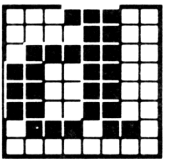
97
&H61
&X01100001



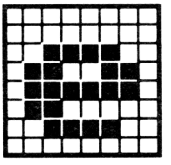
98
&H62
&X01100010



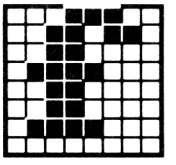
99
&H63
&X01100011



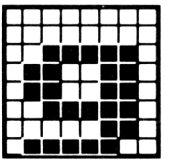
100
&H64
&X01100100



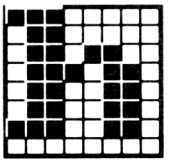
101
&H65
&X01100101



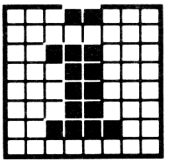
102
&H66
&X01100110



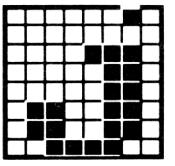
103
&H67
&X01100111



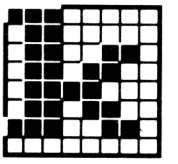
104
&H68
&X01101000



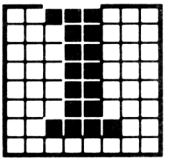
105
&H69
&X01101001



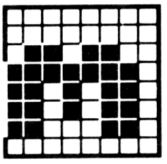
106
&H6A
&X01101010



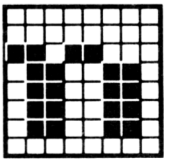
107
&H6B
&X01101011



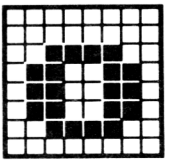
108
&H6C
&X01101100



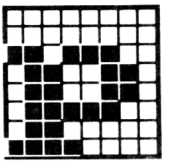
109
&H6D
&X01101101



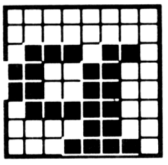
110
&H6E
&X01101110



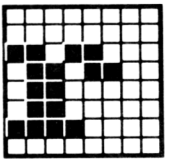
111
&H6F
&X01101111



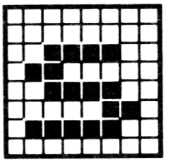
112
&H70
&X01110000



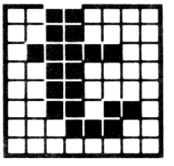
113
&H71
&X01110001



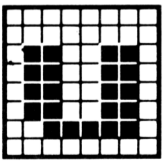
114
&H72
&X01110010



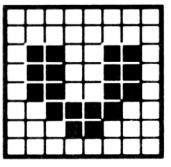
115
&H73
&X01110011



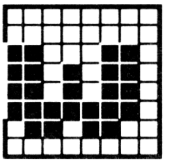
116
&H74
&X01110100



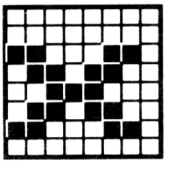
117
&H75
&X01110101



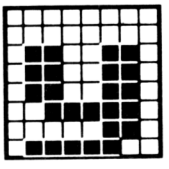
118
&H76
&X01110110



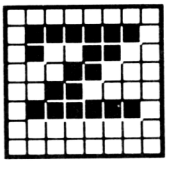
119
&H77
&X01110111



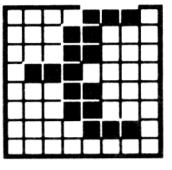
120
&H78
&X01111000



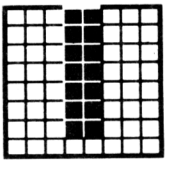
121
&H79
&X01111001



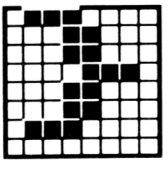
122
&H7A
&X01111010



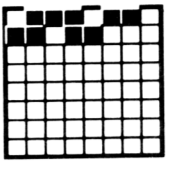
123
&H7B
&X01111011



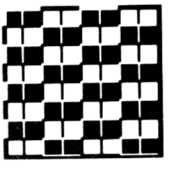
124
&H7C
&X01111100



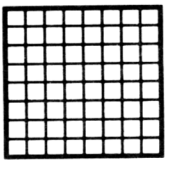
125
&H7D
&X01111101



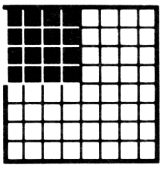
126
&H7E
&X01111110



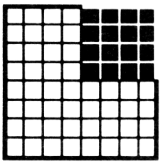
127
&H7F
&X01111111



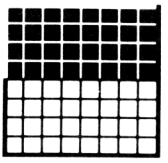
128
&H80
&X10000000



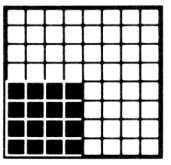
129
&H81
&X10000001



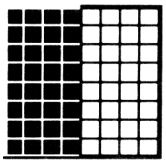
130
&H82
&X10000010



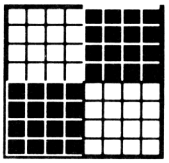
131
&H83
&X10000011



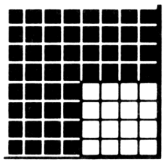
132
&H84
&X10000100



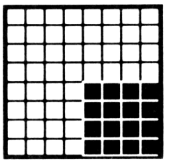
133
&H85
&X10000101



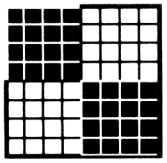
134
&H86
&X10000110



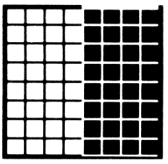
135
&H87
&X10000111



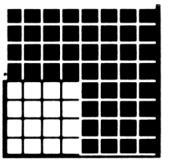
136
&H88
&X10001000



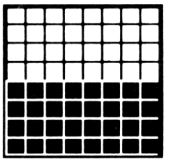
137
&H89
&X10001001



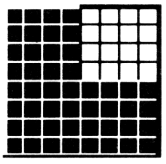
138
&H8A
&X10001010



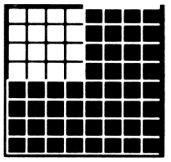
139
&H8B
&X10001011



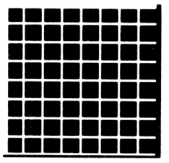
140
&H8C
&X10001100



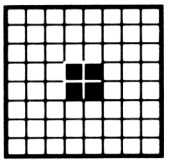
141
&H8D
&X10001101



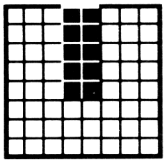
142
&H8E
&X10001110



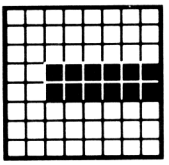
143
&H8F
&X10001111



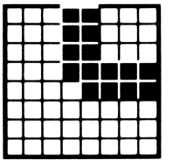
144
&H90
&X10010000



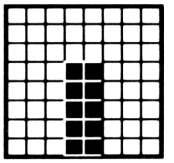
145
&H91
&X10010001



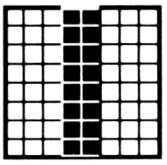
146
&H92
&X10010010



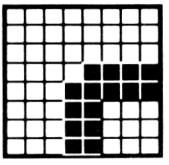
147
&H93
&X10010011



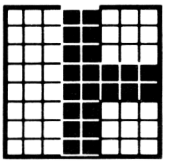
148
&H94
&X10010100



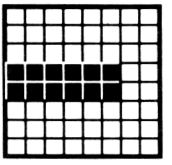
149
&H95
&X10010101



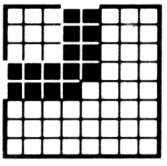
150
&H96
&X10010110



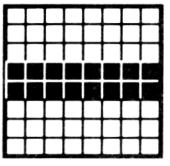
151
&H97
&X10010111



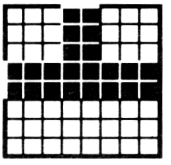
152
&H98
&X10011000



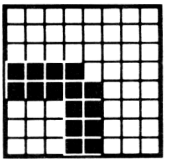
153
&H99
&X10011001



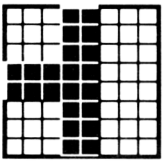
154
&H9A
&X10011010



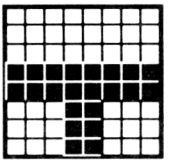
155
&H9B
&X10011011



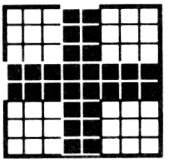
156
&H9C
&X10011100



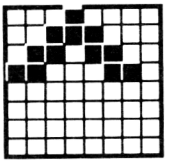
157
&H9D
&X10011101



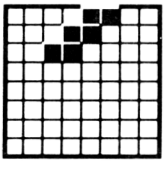
158
&H9E
&X10011110



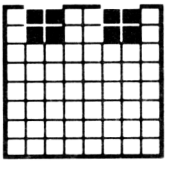
159
&H9F
&X10011111



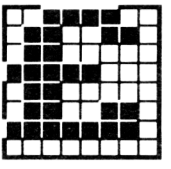
160
&HA0
&X10100000



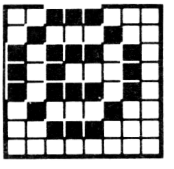
161
&HA1
&X10100001



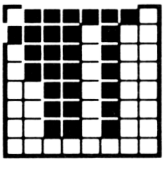
162
&HA2
&X10100010



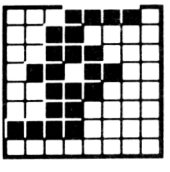
163
&HA3
&X10100011



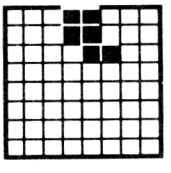
164
&HA4
&X10100100



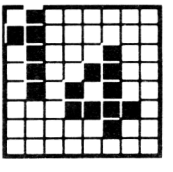
165
&HA5
&X10100101



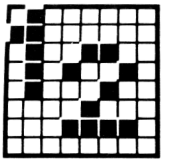
166
&HA6
&X10100110



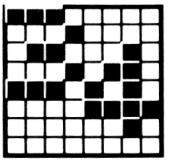
167
&HA7
&X10100111



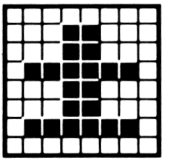
168
&HA8
&X10101000



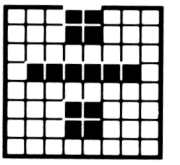
169
&HA9
&X10101001



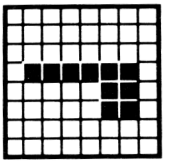
170
&HAA
&X10101010



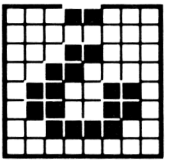
171
&HAB
&X10101011



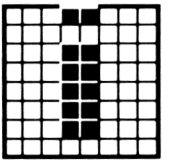
172
&HAC
&X10101100



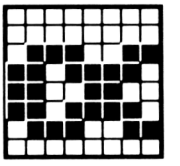
173
&HAD
&X10101101



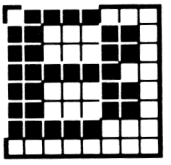
174
&HAE
&X10101110



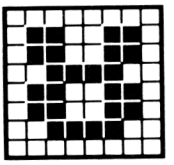
175
&HAF
&X10101111



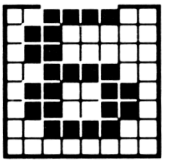
176
&HB0
&X10110000



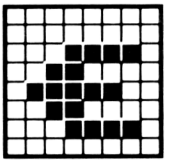
177
&HB1
&X10110001



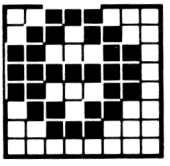
178
&HB2
&X10110010



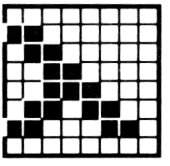
179
&HB3
&X10110011



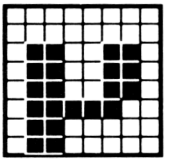
180
&HB4
&X10110100



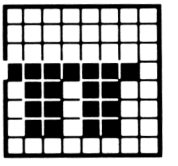
181
&HB5
&X10110101



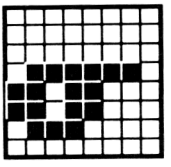
182
&HB6
&X10110110



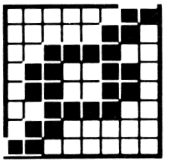
183
&HB7
&X10110111



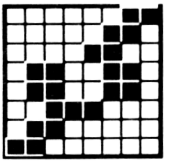
184
&HB8
&X10111000



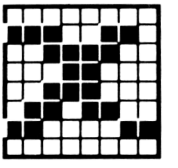
185
&HB9
&X10111001



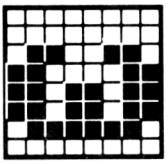
186
&HBA
&X10111010



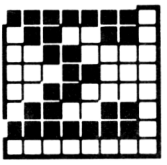
187
&HBB
&X10111011



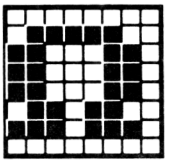
188
&HBC
&X10111100



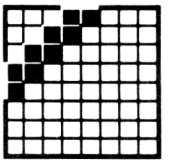
189
&HBD
&X10111101



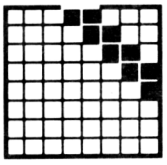
190
&HBE
&X10111110



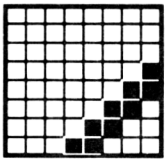
191
&HBF
&X10111111



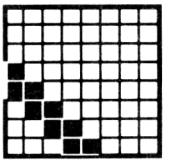
192
&HC0
&X11000000



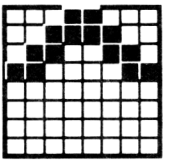
193
&HC1
&X11000001



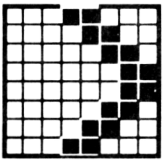
194
&HC2
&X11000010



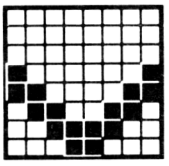
195
&HC3
&X11000011



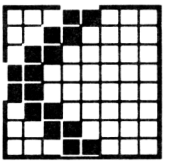
196
&HC4
&X11000100



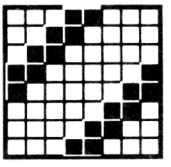
197
&HC5
&X11000101



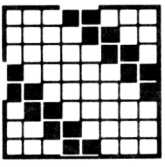
198
&HC6
&X11000110



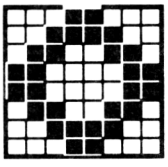
199
&HC7
&X11000111



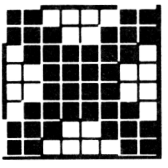
200
&HC8
&X11001000



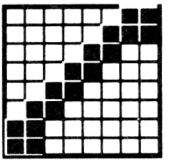
201
&HC9
&X11001001



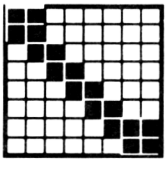
202
&HCA
&X11001010



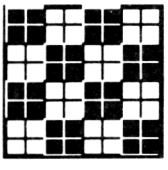
203
&HCB
&X11001011



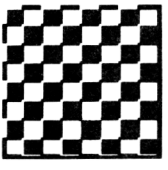
204
&HCC
&X11001100



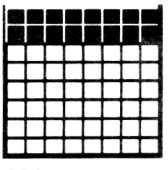
205
&HCD
&X11001101



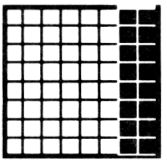
206
&HCE
&X11001110



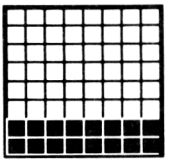
207
&HCF
&X11001111



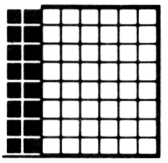
208
&HD0
&X11010000



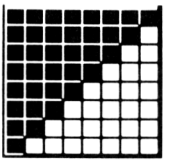
209
&HD1
&X11010001



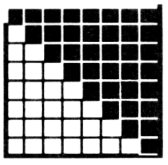
210
&HD2
&X11010010



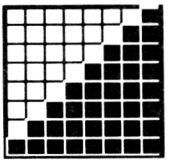
211
&HD3
&X11010011



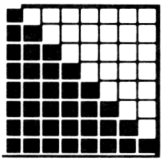
212
&HD4
&X11010100



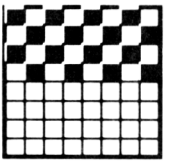
213
&HD5
&X11010101



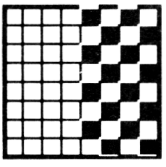
214
&HD6
&X11010110



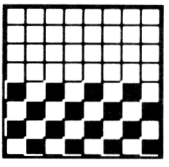
215
&HD7
&X11010111



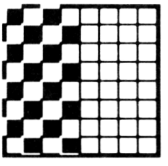
216
&HD8
&X11011000



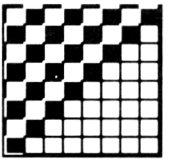
217
&HD9
&X11011001



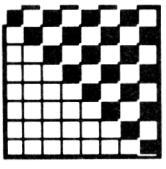
218
&HDA
&X11011010



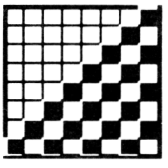
219
&HDB
&X11011011



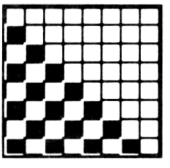
220
&HDC
&X11011100



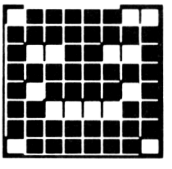
221
&HDD
&X11011101



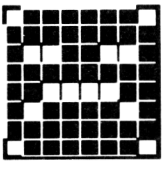
222
&HDE
&X11011110



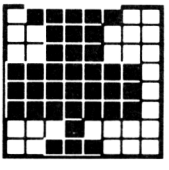
223
&HDF
&X11011111



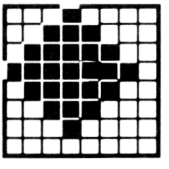
224
&HE0
&X11100000



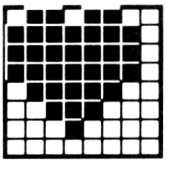
225
&HE1
&X11100001



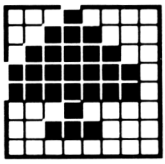
226
&HE2
&X11100010



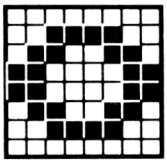
227
&HE3
&X11100011



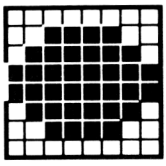
228
&HE4
&X11100100



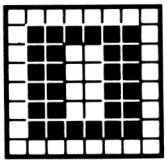
229
&HE5
&X11100101



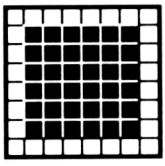
230
&HE6
&X11100110



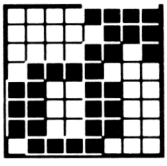
231
&HE7
&X11100111



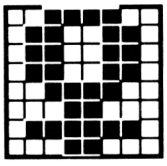
232
&HE8
&X11101000



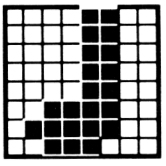
233
&HE9
&X11101001



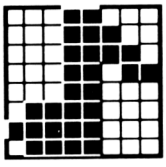
234
&HEA
&X11101010



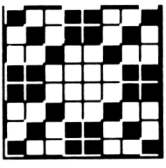
235
&HEB
&X11101011



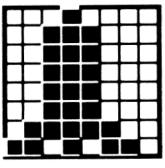
236
&HEC
&X11101100



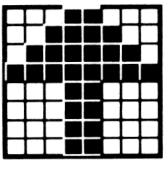
237
&HED
&X11101101



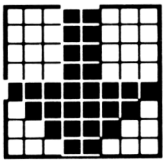
238
&HEE
&X11101110



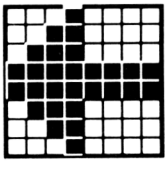
239
&HEF
&X11101111



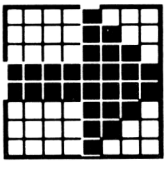
240
&HF0
&X11110000



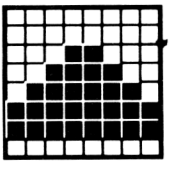
241
&HF1
&X11110001



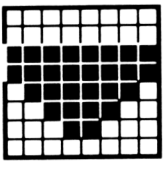
242
&HF2
&X11110010



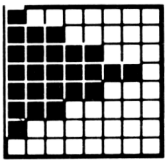
243
&HF3
&X11110011



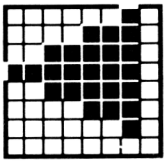
244
&HF4
&X11110100



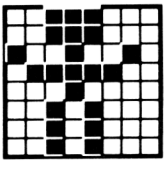
245
&HF5
&X11110101



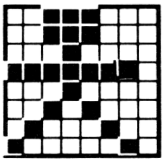
246
&HF6
&X11110110



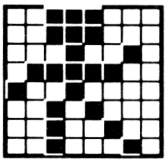
247
&HF7
&X11110111



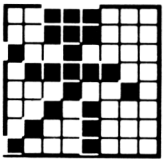
248
&HF8
&X11111000



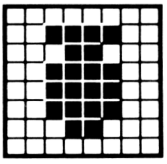
249
&HF9
&X11111001



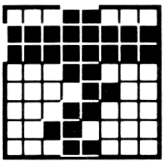
250
&HFA
&X11111010



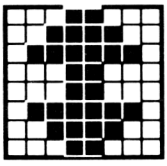
251
&HFB
&X11111011



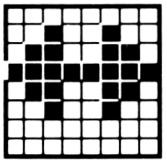
252
&HFC
&X11111100



253
&HFD
&X11111101



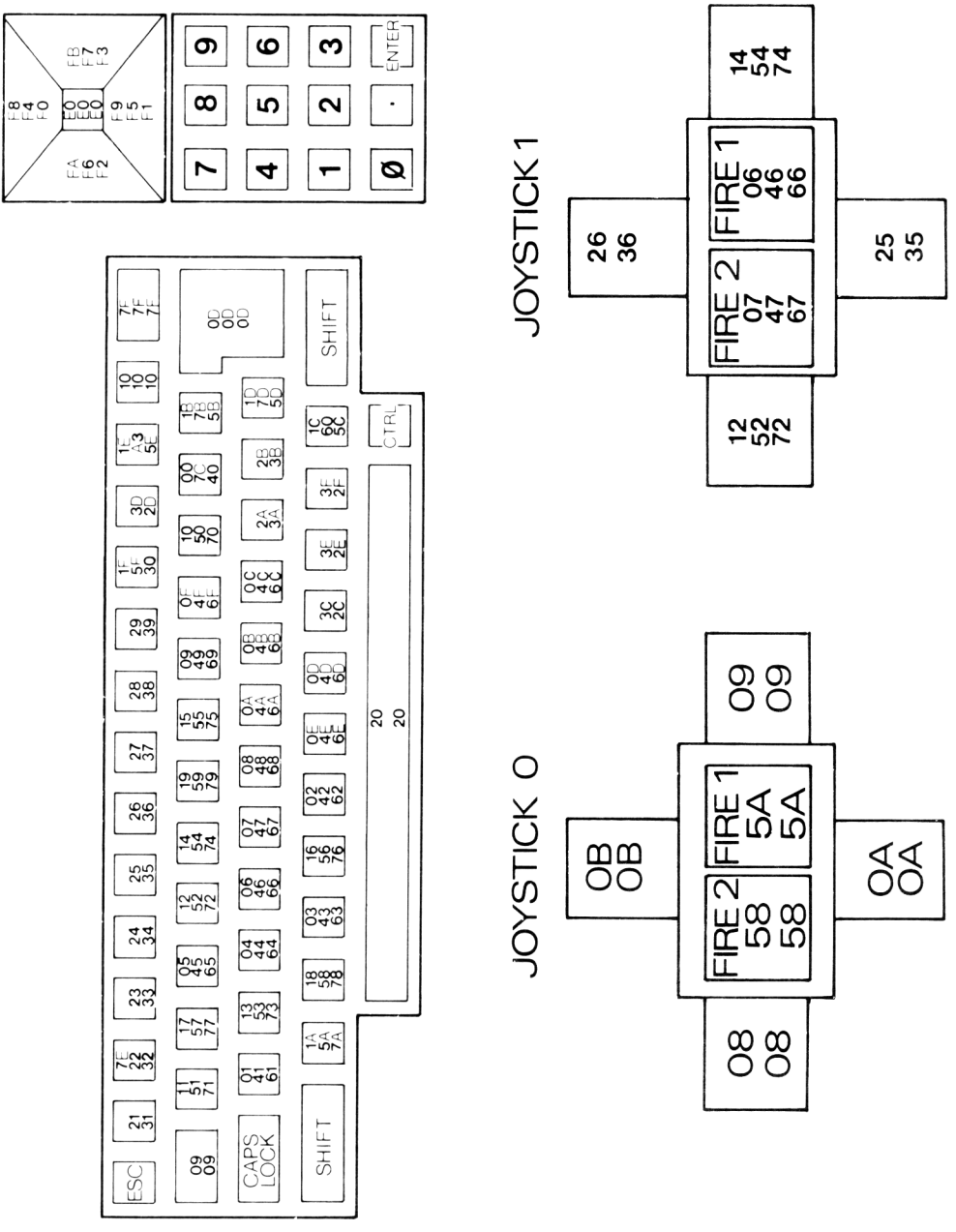
254
&HFE
&X11111110



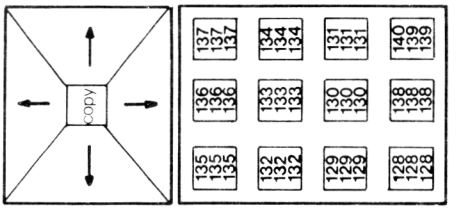
255
&HFF
&X11111111

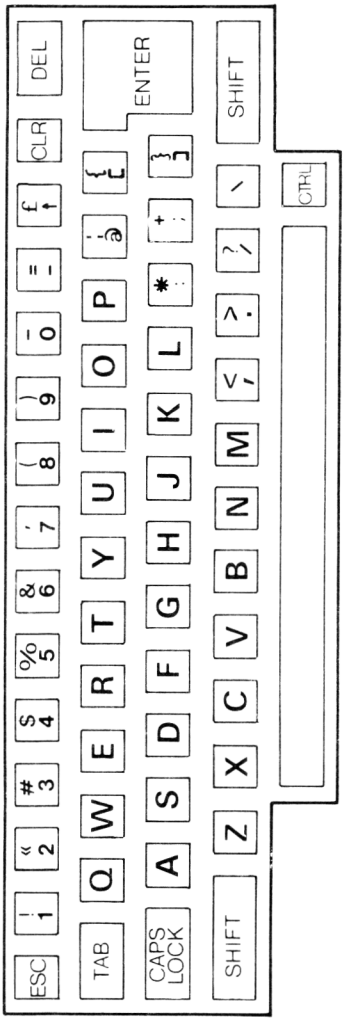
Teil 4: Tastenbezeichnungen

Standard-ASCII-Werte



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)	[ENTER]	&0D
12(140)	RUN"[ENTER]	&52 &55 &4E &22 &0D

Anmerkung: Erweiterungszeichen 13 bis 31 (141 bis 159) haben standardmäßig einen Nullwert. Sie erhalten Werte durch das BASIC-Kommando KEY und werden über das Kommando KEY DEF den entsprechenden Tasten zugeordnet.

Nummernzuordnung zu Tasten und Joystick-Signalen

Teil 5: Sound

Noten und Tonperioden

Die folgende Tabelle beschreibt die empfohlenen Tonperioden für die Noten aller acht Oktaven.
Die wirklich erzeugte Frequenz ist nicht die genau gewünschte Frequenz, da die Periodenangabe ganzzahlig sein muß.
Der RELATIVE FEHLER ist das prozentuale Verhältnis der Differenz von genauer und erzeugter Frequenzen zur genauen Frequenz.

NOTE	FREQUENZ	PERIODE	RELATIVER FEHLER	
C	32.703	1911	-0.007 %	
C#	34.648	1804	+0.007 %	
D	36.708	1703	-0.007 %	
D#	38.891	1607	-0.004 %	**Octave -2**
E	41.203	1517	+0.009 %	
F	43.654	1432	-0.016 %	
F#	46.249	1351	-0.009 %	
G	48.999	1276	-0.002 %	
G#	51.913	1204	+0.005 %	
A	55.000	1136	+0.012 %	
A#	58.270	1073	-0.008 %	
B	61.735	1012	+0.011 %	

NOTE	FREQUENZ	PERIODE	RELATIVER FEHLER	
C	65.406	956	-0.007 %	
C#	69.296	902	+0.007 %	
D	73.416	851	+0.022 %	
D#	77.782	804	-0.004 %	
E	82.407	758	+0.009 %	
F	87.307	716	+0.019 %	
F#	92.499	676	-0.028 %	**Octave -3**
G	97.999	638	+0.037 %	
G#	103.826	602	+0.005 %	
A	110.000	568	-0.032 %	
A#	116.541	536	+0.039 %	
B	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	478	-0.037 %	
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 %	
B	246.942	253	-0.038 %	

NOTE	FREQUENZ	PERIODE	RELATIVER FEHLER	
C	261.626	239	+0.046 %	Middle C
C#	277.183	225	+0.007 %	
D	293.665	213	+0.081 %	
D#	311.127	201	+0.058 %	
E	329.628	190	-0.057 %	
F	349.228	179	+0.019 %	Oktave 0
F#	369.994	169	+0.046 %	
G	391.995	159	-0.037 %	
G#	415.305	150	+0.005 %	
A	440.000	142	-0.032 %	
International A				
A#	466.164	134	-0.055 %	
B	493.883	127	-0.038 %	

NOTE	FREQUENZ	PERIODE	RELATIVER FEHLER	
C	523.251	119	+0.046 %	
C#	554.365	113	-0.215 %	
D	587.330	106	+0.081 %	
D#	622.254	100	+0.058 %	
E	659.255	95	+0.206 %	
F	698.457	89	+0.019 %	Oktave 1
F#	739.989	84	+0.046 %	
G	783.991	80	-0.277 %	
G#	830.609	75	-0.328 %	
A	880.000	71	-0.032 %	
A#	932.328	67	-0.055 %	
B	987.767	63	+0.356 %	

NOTE	FREQUENZ	PERIODE	RELATIVER FEHLER	
C	1046.502	60	-0.374 %	
C#	1108.731	56	+0.229 %	
D	1174.659	53	-0.390 %	
D#	1244.508	50	-0.441 %	
E	1318.510	47	+0.206 %	
F	1396.913	45	-0.543 %	Oktave 2
F#	1479.978	42	-0.548 %	
G	1567.982	40	+0.350 %	
G#	1661.219	38	-0.328 %	
A	1760.000	36	-0.032 %	
A#	1864.655	34	-0.055 %	
B	1975.533	32	-0.435 %	

NOTE	FREQUENZ	PERIODE	RELATIVER FEHLER	
C	2093.004	30	+0.462 %	
C#	2217.461	28	-0.662 %	
D	2349.318	27	-0.390 %	
D#	2489.016	25	-0.441 %	
E	2637.021	24	-0.855 %	
F	2793.826	22	+0.574 %	Oktave 3
F#	2959.955	21	-0.548 %	
G	3135.963	20	+0.350 %	
G#	3322.438	19	+0.992 %	
A	3520.000	18	+1.357 %	
A#	3729.310	17	+1.417 %	
B	3951.066	16	+1.134 %	

NOTE	FREQUENZ	PERIODE	RELATIVER FEHLER	
C	4186.009	15	+0.462 %	
C#	4434.922	14	-0.662 %	
D	4698.636	13	+1.469 %	
D#	4978.032	13	-0.441 %	
E	5274.041	12	+1.246 %	
F	5587.652	11	-1.685 %	Oktave 4
F#	5919.911	11	-0.548 %	
G	6271.927	10	+0.350 %	
G#	6644.875	9	+0.002 %	
A	7040.000	9	+1.357 %	
A#	7458.621	8	+1.417 %	
B	7902.133	8	+1.134 %	

Diese Werte wurden vom Internationalen A aus wie folgt errechnet:

```
Frequenz = 440 * (2 (OKTAVE + (10 - N) /12))
 Periode = Round (62500 / Frequenz)
```

mit N gleich 1 fuer C, 2 fuer C#, 3 fuer D, usw.

Teil 6: Fehlermeldungen

1 `Unexpected NEXT` – unerwartetes NEXT

Ein `NEXT`-Kommando wurde gefunden, obwohl keine `FOR`-Schleife vorhanden war, oder die Steuervariable im `NEXT`-Kommando paßt nicht zu der im `FOR`.

2 `Syntax Error` – Syntaxfehler

BASIC versteht die angegebene 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 Mehrzweckmeldung. Der Wert eines Arguments einer Funktion oder ein Kommandoparameter ist falsch.

6 `Overflow` – Überlauf

Das Ergebnis einer arithmetischen Operation ist 'übergelaufen'. Es kann ein Gleitkommawertüberlauf sein, weil das Ergebnis größer als `1.7E-38` ist; oder es wurde versucht, eine Gleitkommazahl in eine 16 Bit große Ganzzahl mit Vorzeichen umzuwandeln.

7 `Memory full` – Speicher voll

Das Programm oder seine Variablen sind einfach zu voll, oder das Programm ist zu sehr verschachtelt (geschachtelte `GOSUB`s, `WHILE`s und `FOR`s).
Ein `Memory`-Kommando kann diese Meldung auslösen, falls der BASIC zur Verfügung gestellte Speicherplatz zu klein ist oder viel zu groß gewählt wurde. Beachten sie, daß eine geöffnete Cassettendatei 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 (Index) 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: Reeller oder Ganzzahliger Division, Divisionsreste-Bestimmung oder Potenzierung auftreten.

12 `Invalid direct command` – Falsches Direktkommando

Das zuletzt versuchte Kommando ist im direkten Modus nicht erlaubt.

13 `Type mismatch` – Typenunstimmigkeit

Ein numerischer Wert wurde angegeben, obwohl ein Textzeichen verlangt wurde oder umgekehrt, oder es wurde in einem `READ` oder `INPUT` eine Zahl des falschen Typs eingegeben.

14 `String space full` – Textzeichenbereich voll

Es wurden zuviele Zeichenketten erstellt, so daß selbst nach dem Auffüllen aller freien Speicherplätze im Programm kein Platz mehr ist.

15 `String too long` – Zeichenkette zu lang

Eine Zeichenkette hat mehr als 255 Zeichen. Kann auftreten, wenn mehrere Zeichenketten aneinandergehängt werden (addiert).

16 `String expression too complex` – Zeichenkettenausdruck zu komplex

Zeichenkettenausdrücke können viele Zwischenketten erzeugen. Wenn deren Anzahl eine vernünftige Grenze überschreitet, gibt BASIC mit dieser Meldung auf.

17 `Cannot CONTinue` – `CONT` funktioniert nicht

Das aktuelle Programm kann aus irgendeinem Grund mit `CONT` nicht wieder gestartet werden. Beachten Sie, daß `CONT` nur für das Wiederstarten eines Programms nach einem `STOP`-Kommando, `[ESC][ESC]` oder Fehler ist, und jegliche zwischenzeitige Programmänderung ein Wiederstarten unmöglich macht.

18 `Unknown user function` – unbekannte Anwender-Funktion

Für die gerufene Funktion wurde kein `DEF FN`-Kommando gegeben.

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

Eine Zeile ohne Zeilennummer wurde gefunden, während ein Programm von Cassette geladen wurde.

22 `Operand missing` – Operand fehlt

BASIC hat eine unvollständige Anweisung gefunden.

23 `Line too long` – Zeile zu lang

Eine Zeile wird im internen BASIC-Format zu groß

24 `EOF met` – `EOF` angetroffen

Es wurde versucht, eine Cassetten-Eingabedatei weiter zu lesen, obwohl deren Dateiende (end of file) schon erreicht ist.

25 `File type error` – Dateityp fehlerhaft

Die gelesene Cassettendatei hat einen unzulässigen Typ. `OPENIN` ist nur für `ASCII`-Textdateien erlaubt. `LOAD`, `RUN`, usw. sind nur für Dateien erlaubt, die mit `SAVE` erstellt wurden.

26 `NEXT missing` – `NEXT` fehlt

Ein `FOR`- Kommando wurde angegeben, ein dazu passendes `NEXT` kann nicht gefunden werden.

27 `File already open` – Datei schon geöffnet

Ein `OPENIN`- oder `OPENOUT`-Kommando wurde gegeben, obwohl die früher schon eröffnete Datei noch nicht wieder geschlossen wurde.

28 `Unknown command` – Kommando unbekannt

Ein externes Kommando ist im BASIC unbekannt.

29 `WEND missing` – `WEND` fehlt

Ein `WHILE` Kommando wurde angegeben, ein dazu passendes `WEND` kann nicht gefunden werden.

30 `Unexpected WEND` – unerwartetes `WEND`

Ein `WEND` wurde außerhalb einer `WHILE` Schleife gefunden, oder ein `WEND` paßt nicht zu der aktuellen `WHILE` Schleife.

`31 File not open = Datei nicht geoeffnet`

(Siehe späteren Abschnitt mit dem Titel „Diskettenfehler")

`32 Broken in`

(Siehe späteren Abschnitt mit dem Titel „Diskettenfehler").

Diskettenfehler

Während der Bearbeitung beliebiger Dateien können mehrere Fehler auftreten. BASIC bearbeitet alle diese Fehler unter Fehler Nr. 32, wobei sich jedoch eine nähere Aussage über die Funktion DERR bei Antreffen dieser Fehlernummer treffen läßt.
Hierbei ergeben sich die nachstehend aufgezeigten Werte:

ERR kann ebenfalls als 31 zurückgemeldet werden, wenn ein Zugriff versucht wurde, jedoch keine Datei geöffnet war. Die normale Form des einsatzes von ERR und DERR wäre die Einbeziehung eines **ON ERROR GOTO**-Befehls, der über eine kurze Routine prüfen läßt, ob ERR den Wert von **31** bzw. **32** hat; handelt es sich um **32**, so kann DERR nach weiteren Informationen über die Art des Fehlers abgefragt werden. Beispielsweise

```
10 ON ERROR GOTO 1000
20 OPENOUT "myfile.asc"
30 WRITE #9,"test-data"
50 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, nehmen Sie doch eine neue Daten-
     Diskette":END
1040 IF amsdoserr=&X01001000 THEN PRINT "legen Sie
     eine Diskette in das Laufwerk, und druecken
     Sie eine Taste":WHILE INKEY$<>"":WEND:RESUME
1050 END
```

AMSDOS Fehler	DERR-Wert	Fehlerursache
0	0 bzw. 22	(ESC) wurde gedrückt.
14	142(128+14)	Kanal hat keinen passenden Status.
15	143(128+15)	Ende der Datei erreicht.
16	144(128+16)	Fehlerhaftes Kommando, normalerweise nicht zutreffender Dateiname.
17	145(128+17)	Datei besteht bereits.
18	146(128+18)	Datei besteht noch nicht.
19	147(128+19)	Keine Inhaltsverzeichnis-Einträge mehr frei.
20	148(128+20)	Diskette voll.
21	149(128+21)	Diskette bei offenen Dateien entnommen.
22	150(128+22)	Nur lesbare Datei.
26	154(128+26)	Dateiende ermittelt (Soft end)

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 Bits haben jeweils 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 durch AMSDOS angegeben wurde.

Teil 7: BASIC Schlüsselwörter

Im Anschluß erfolgt eine Aufstellung aller Schneider BASIC CPC664-Schlüsselwörter. Als solche sind diese Schlüsselwörter 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

HEX$, HIMEM

IF, INK, INKEY, INKEY$, INP, INPUT, INSTR, INT

JOY

KEY, KEY DEF

LEFT$, LEN, LET, LINE, LIST, LOAD, LOCATE, LOG, LOG10, LOWER$

MAX, MEMORY, MERGE, MID$, MIN, MOD, MODE, MOVE, MOVER, MASK

NEXT, NEW, NOT

ON, ON BREAK, ON ERROR GOTO, 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

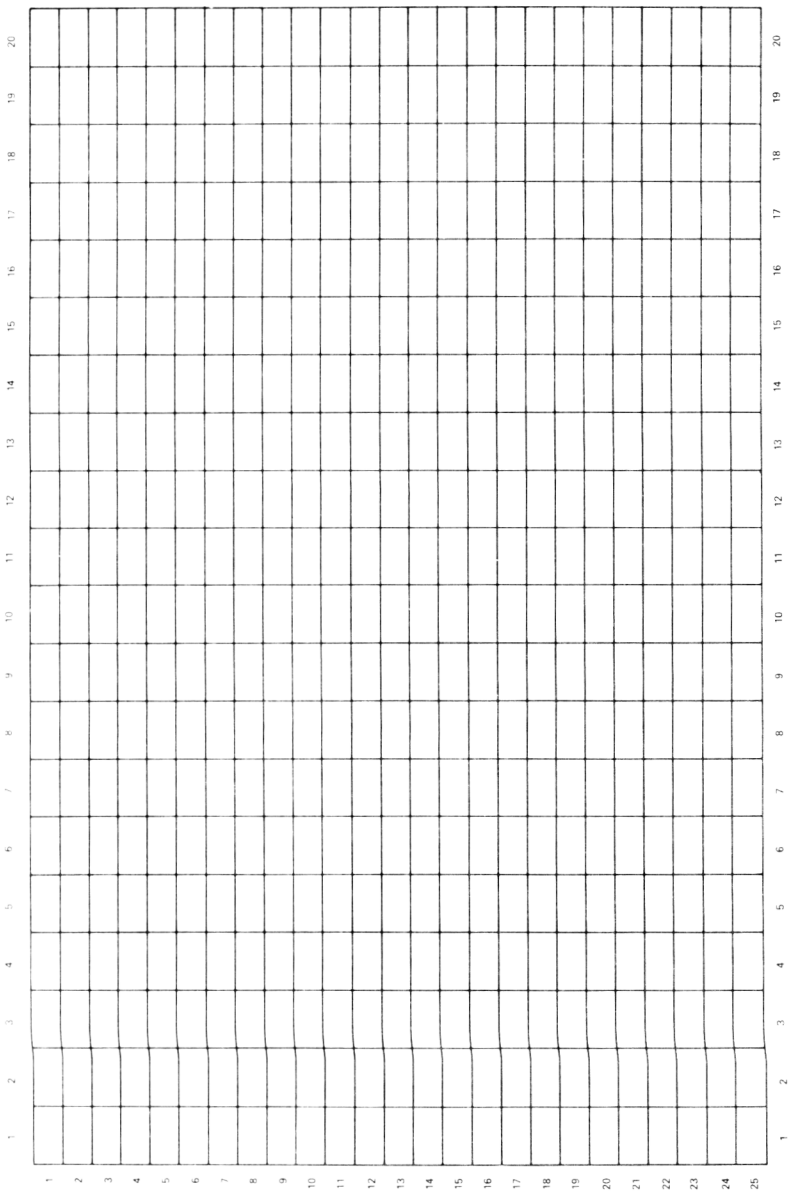
XOR, XPOS

YPOS

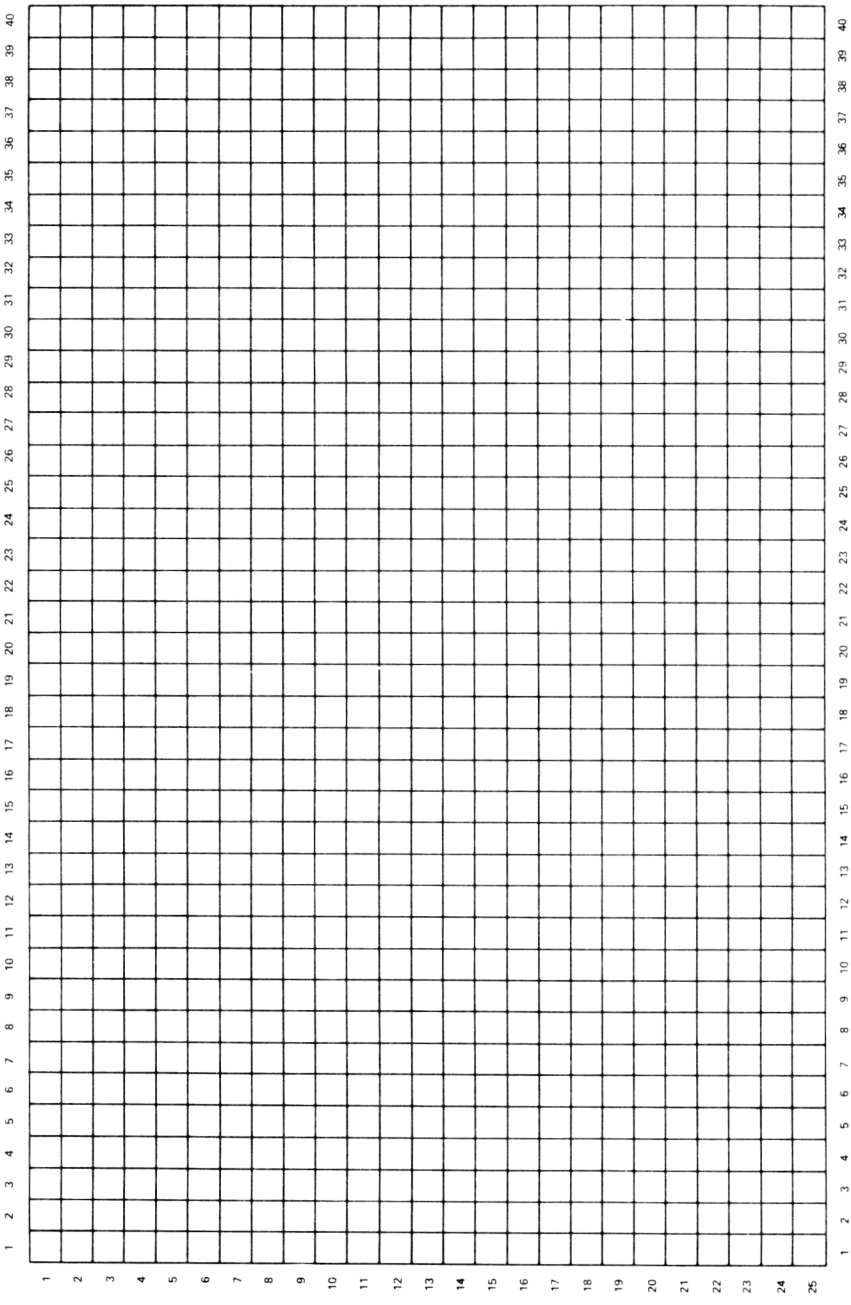
ZONE

Teil 8: Planer

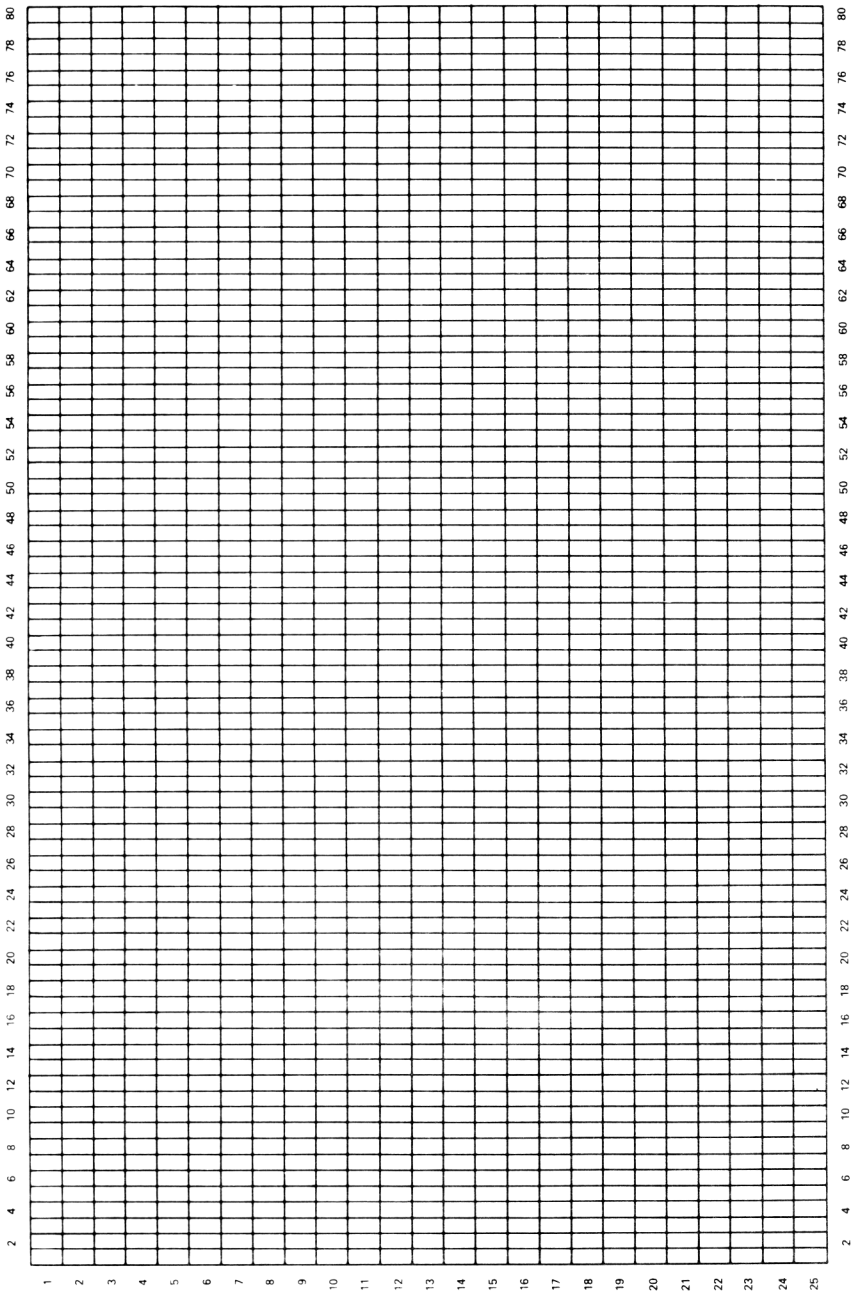
Text- und Fenster-Planer – MODUS 0 (20 Spalten)



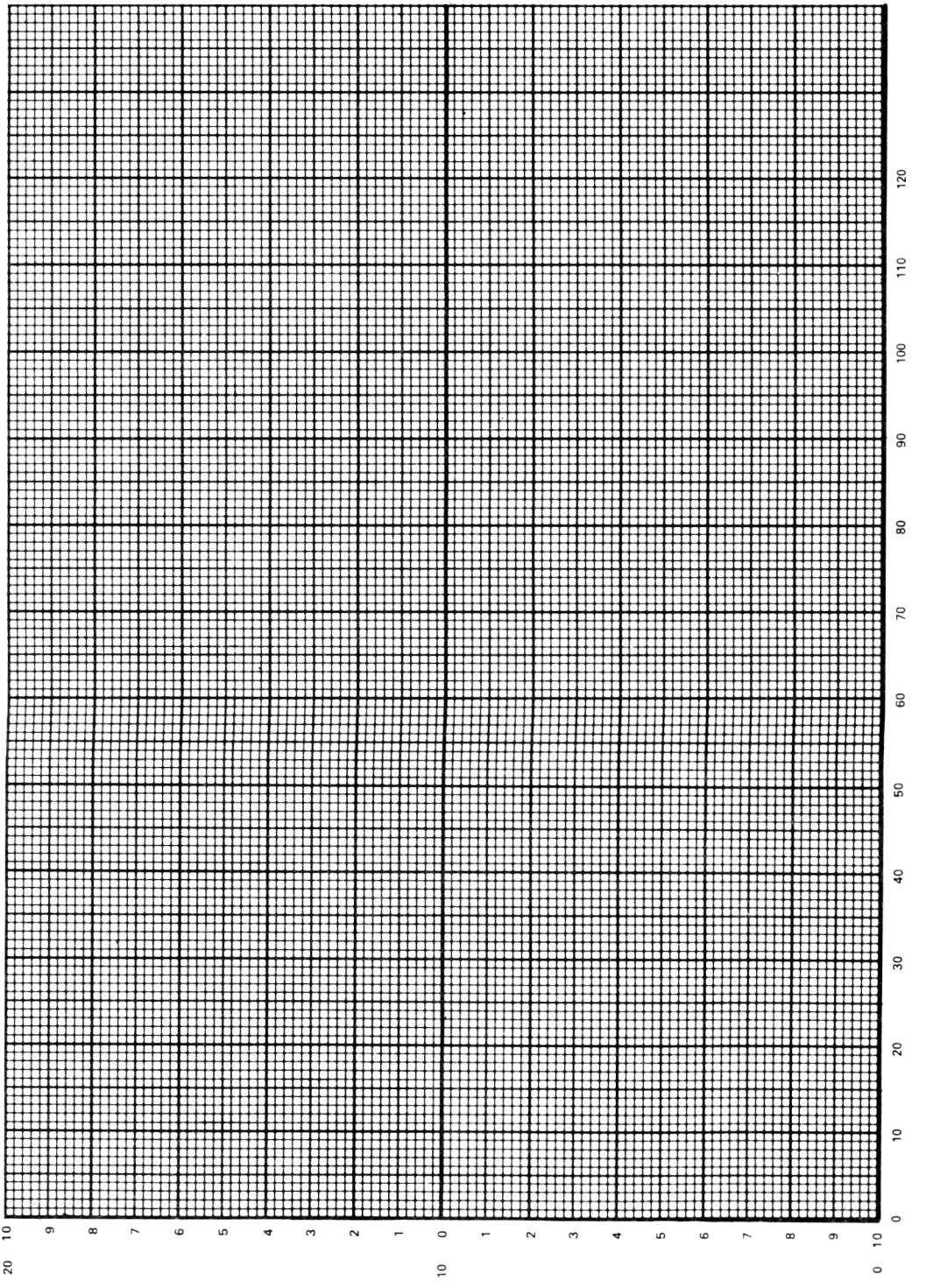
TEXT und WINDOW Planer
MODE 1 40 Spalten



TEXT und WINDOW Planer
MODE 2 80 Spalten

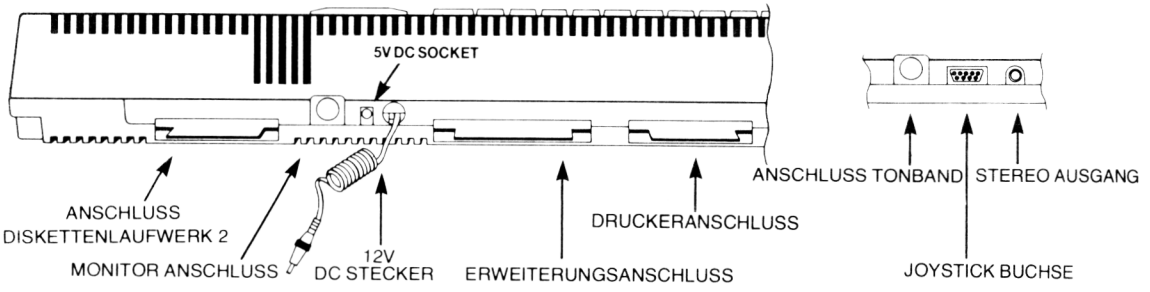


Tonvariations- und Musik-Planer

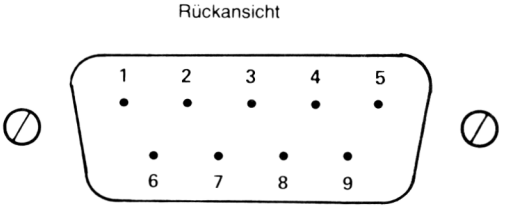


Teil 9: Anschlüsse

CPC664 Eingangs-/Ausgangs-Anschlüsse

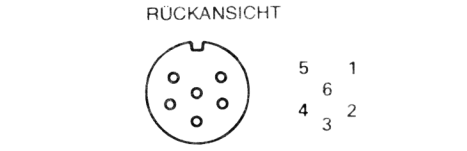


Anschluß für Joystick



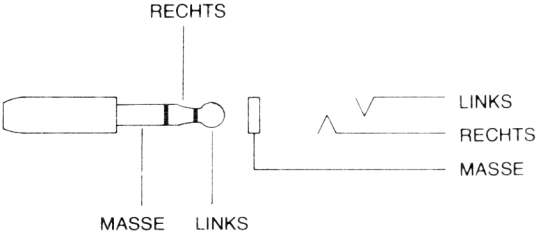
Stift 1	oben	Stift 6	Feuer 2
Stift 2	unten	Sitft 7	Feuer 1
Stift 3	links	Stift 8	Common
Stift 4	rechts	Stift 9	COM 2
Stift 5	frei		

Monitor-Anschluß



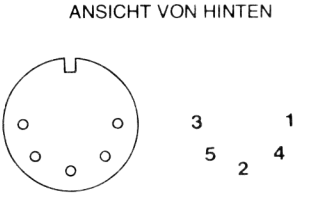
Stift 1	rot	Stift 4	Sync
Stift 2	grün	Stift 5	Erde
Stift 3	blau	Stift 6	Luminanz

Stereo-Anschluß



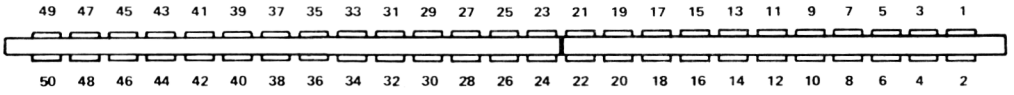
Stift 1	Linker Kanal
Stift 2	Rechter Kanal
Stift 3	Masse

Bandanschluß



Stift 1	Fernsteuerung	Stift 4	Dateneingang
Stift 2	Masse	Stift 5	Datenausgang
Stift 3	Fernsteuerung		

Erweiterungsanschluß



ANSICHT VON HINTEN

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 des Disketten-Laufwerkes 2

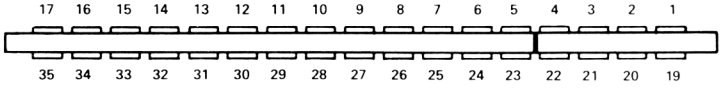


ANSICHT VON HINTEN

PIN 1	$\overline{\text{READY}}$	PIN 18	$\overline{\text{GND}}$
PIN 2	$\overline{\text{GND}}$	PIN 19	$\overline{\text{MOTOR ON}}$
PIN 3	$\overline{\text{SIDE 1 SELECT}}$	PIN 20	GND
PIN 4	$\overline{\text{GND}}$	PIN 21	N/C
PIN 5	$\overline{\text{READ DATA}}$	PIN 22	$\overline{\text{GND}}$
PIN 6	$\overline{\text{GND}}$	PIN 23	$\overline{\text{DRIVE SELECT 1}}$
PIN 7	$\overline{\text{WRITE PROTECT}}$	PIN 24	GND
PIN 8	$\overline{\text{GND}}$	PIN 25	N/C
PIN 9	$\overline{\text{TRACK 0}}$	PIN 26	$\overline{\text{GND}}$
PIN 10	$\overline{\text{GND}}$	PIN 27	$\overline{\text{INDEX}}$
PIN 11	$\overline{\text{WRITE GATE}}$	PIN 28	GND
PIN 12	$\overline{\text{GND}}$	PIN 29	N/C
PIN 13	$\overline{\text{WRITE DATA}}$	PIN 30	GND
PIN 14	$\overline{\text{GND}}$	PIN 31	N/C
PIN 15	$\overline{\text{STEP}}$	PIN 32	GND
PIN 16	$\overline{\text{GND}}$	PIN 33	N/C
PIN 17	$\overline{\text{DIRECTION SELECT}}$	PIN 34	GND

Druckeranschluß

ANSICHT VON HINTEN



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 CPC664-System ermöglicht den Anschluß und Einsatz eines Industrienorm-Druckers mit „Centronics-Schnittstelle".

Das Druckerkabel ist in einfacher Form als eins zu eins-Verbindung zwischen dem DRUCKER-Anschluß auf der Rückseite des Computers und dem Anschlußelement des Parallel-Druckers aufgebaut.

Einzelheiten in Bezug auf die Belegung der Schnittstelle sind in Abschnitt 9 dieses Kapitels behandelt.

Demzufolge sollte das Kabel Stift 1 des Computers mit Stift 1 des Druckers, Stift 19 des Computers mit Stift 19 des Druckers usw. verbinden. Hierbei ist jedoch darauf zu achten, daß die Stifte 18 und 36 des Druckers **nicht** mit dem Computer verbunden werden.

Ferner ist anzumerken, daß sich 17 Anschlüsse auf der oberen Reihe des Anschlußelementes des Computers (PRINTER) befinden und die untere Stiftreihe von 19 an aufwärts numeriert ist. Damit ist gewährleistet, daß jeder Stift bzw. Leitung zum jeweils gleich numerierten Stift bzw. Leitung auf der Computer-Anschlußseite und auf der Drucker-Anschlußseite geht.

Der Computer setzt das BUSY-Signal (Stift 11) ein, um zu prüfen, ob der Drucker bereit ist; der Computer wartet, wenn sich der Drucker im OFF-LINE-Zustand befindet.

Sie brauchen hierbei keine besonderen Benutzerbefehle einzugeben.

Die Drucker-Schnittstelle im CPC664 ist so ausgelegt, daß Matrixdrucker, Typenraddrucker mit entsprechender Schnittstelle, Graphikplotter und Farb-Tintenstrahldrucker angeschlossen werden können. In jedem Falle wird die erforderliche Kompatibilität durch die genormte parallele Schnittstelle gewährleistet.

Die auf die Belange des Benutzers abgestimmte Software im Schneider NLQ401 unterstützt den Punktgraphikbetrieb, das Ausdrucken kompletter Bildschirmabzüge.

Der „NLQ401" ist weiterhin in der Lage, den kompletten deutschen Zeichensatz zu drucken.

Teil 11: Joystick

Die im Schneider CPC664 eingebaute Software unterstützt einen oder zwei Joysticks, die wie ein Teil der Tastatur behandelt werden und deshalb mit INKEY sowie INKEY$ abgefragt werden können.

Hierzu ist anzumerken, daß in der Mehrzahl der Fälle der Haupt-„Feuer"-Knopf auf einem Joystick als „Feuer 2" für den CPC664 bezeichnet wird.

Die Funktionen JOY (0) und JOY (1) ermöglichen eine direkte Prüfung des ersten bzw. zweiten Joysticks. Die Funktion liefert ein bitsignifikantes Ergebnis, das die Bewegung des Joysticks zum Zeitpunkt der letzten Tastatur-Abprüfung wiedergibt.
Die anschließende Tabelle zeigt die von beiden Joysticks gelieferten Werte. Den JOY-Werten sind solche für den Einsatz in Befehlen nachgestellt, die Schlüsselnummern als Parameter aufweisen (d.h. INKEY und KEY DEF).

STATUS	JOY-STEUERUNG		SCHLÜSSELWERTE		
	Bit-Einst.	Anz. Wert	Erster Joystick	Zweiter Joystick	Entsprechende Taste
Auf	0	1	72	48	'6'
Ab	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'

Hierbei ist darauf hinzuweisen, daß bei der Rückübertragung der Werte vom zweiten Joystick der Computer nicht unterscheiden kann, ob diese Werte vom Joystick bzw. von der entsprechenden Taste ausgehen (in der letzten Spalte der vorhergehenden Tabelle angegeben). Hieraus folgt, daß die Tastatur als Ersatz für den zweiten Joystick angesehen werden kann.

Teil 12: Disketten-Organisation

Das BIOS unterstützt drei Disketten-Formate:
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 Erkennung möglich zu machen, besitzt jedes Format einheitliche Sektor-Nummern.

3 Zoll-Disketten sind doppelseitig verwendbar. Der Computer hat immer den Datenbestand einer Diskettenseite im Zugriff. Zum Bearbeiten der Rückseite drehen Sie die Diskette um und führen sie wieder in das Laufwerk ein. Die beiden Seiten der Diskette können unterschiedliche Formate besitzen.

Für alle Formate gemeinsam:

Single sided (die beiden Seiten einer 3 Zoll-Diskette werden gesondert behandelt).
512 byte physikalische Sektorgröße
40 Spuren, von 0 bis 39 numeriert
1024 Byte CP/M-Blockgröße
64 Inhaltsverzeichnis-Einträge.

Systemformat

9 Sektoren pro Spur &41 bis &49.
2 reservierte Spuren.

Das Systemformat ist das wichtigste Format, da CP/M nur von einer Diskette mit Systemformat geladen werden kann (Kalt- und Warmstart).
Die reservierten Spuren werden wie folgt verwendet:

Spur 0, Sektor &41	: Boot-Sektor	
Spur 0, Sektor &42	: Konfigurationssektor	
Spur 0, Sektoren &43 bis &47	: unbenutzt	
Spur 0, Sektoren &48 bis &49	: }	CCP und BDOS.
Spur 1, Sektoren &41 bis &49	: }	

Hinweis: Das VENDOR-Format ist eine spezielle Version des Systemformats, das keine Software auf den beiden reservierten Spuren enthält.

DATA-ONLY-Format

9 Sektoren &C1 bis &C9 pro Spur
Keine reservierten Spuren

Dieses Format dient zukünftigen Erweiterungen und ist nicht für die Anwendung mit CP/M vorgesehen, da mit diesem Format kein „Warmstart" möglich ist. Wenn jedoch ausschließlich AMSDOS verwendet wird, steht mit diesem Format (DATA ONLY) etwas mehr Disketten-Speicherplatz zur Verfügung.

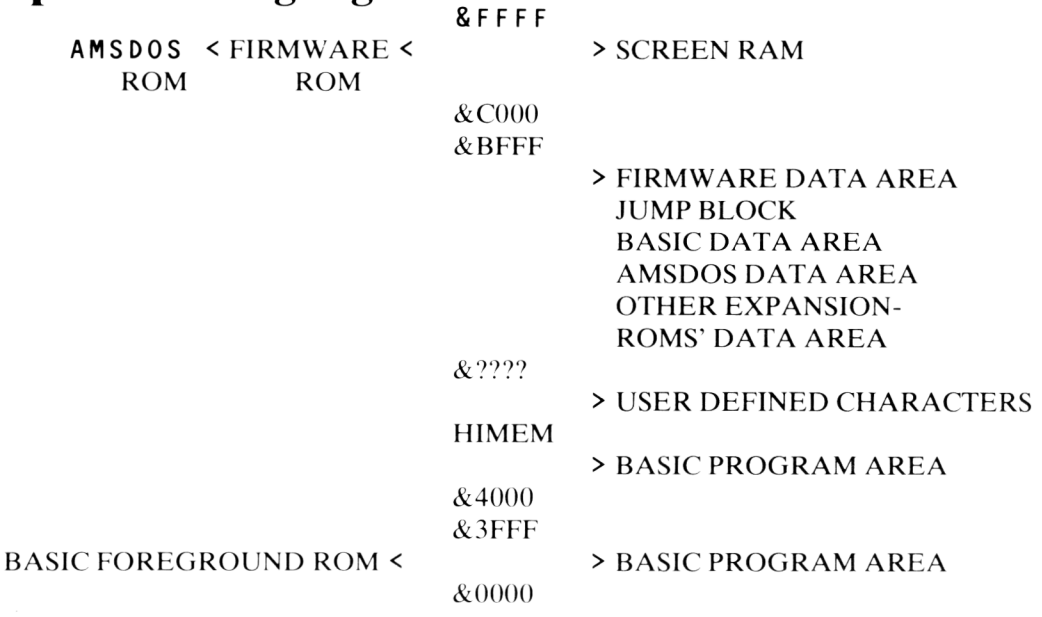
IBM-Format

8 Sektoren &1 bis &8 pro Spur
1 reservierte Spur.

Dieses Format ist logisch das gleiche Format, wie das durch CP/M auf dem IBM PC verwendete Single-Side-Format. Es dient speziellen Anwendungen und wird für andere Zwecke nicht empfohlen.

Teil 13: Speicher

Speicherbelegung



...wobei die Adresse &???? von den Erweiterungs-ROMs abhängt. (Hierbei ist anzumerken, daß &???? &A6FC ausdrückt, wenn keine externen Erweiterungs-ROMs vorgesehen sind).

Zusätzliche Ein-/Ausgabe

Die Mehrzahl der Eingabe-/Ausgabeadressen wird vom Computer belegt, insbesondere sollten Adressen unter &7FFF nicht belegt werden.

Es kann so vorgegangen werden, daß der Adressenteil A0–A7 den Typ der externen Eingabe-/Ausgabe-Vorrichtung bezeichnen soll, und daß die Adressenleitungen A8 und A9 decodiert werden können, um Register innerhalb der Eingabe-/Ausgabe-Vorrichtung zu wählen. Von den verbleibenden Adressenleitungen ist lediglich A10 zu decodieren (als low) während die Leitungen A11 bis A15 als high gelten. Somit kann jede Vorrichtung mit Registern ausgerüstet werden, die adressiert werden wie z.B. &F8??, &F9??, &FA??, wobei ?? den Bereich DC bis DF für Kommunikations-Schnittstellen und EO bis FE für andere Benutzer-Peripheriegeräte angibt.

Hierbei ist zu beachten, daß die Befehle des Z80 einzusetzen sind, die das B-Register auf die obere Hälte des Adressenbus (A15–A8) setzt.

Zusätzliche ROMs

Anstelle der vorhandenen ROMs können zusätzliche ROMs eingesetzt werden. Die Adressenzuordnung und Bank select Logik sind in einem Modul enthalten, das an die Erweiterungssammelleitung angeschlossen ist, wobei jedoch alle erforderlichen Signale zur Erweiterungs-Sammelleitung übertragen werden.

Für Ihre Notizen:

Kapitel 8
„Wenn Sie Zeit haben . . .“

Dieses Kapitel befaßt sich ganz zwanglos mit einigen Hintergrund-Informationen für den Umgang mit dem Computer im Allgemeinen und mit dem CPC664 im Besonderen. Sie müssen dieses Kapitel nicht vor dem Arbeiten mit dem Rechner durcharbeiten, jedoch hilft es Ihnen etwas bei Ihrem Bemühen, zu entdecken, was sich "hinter den Karten versteckt".

TEIL 1: GANZ ALLGEMEIN GESPROCHEN ...

Packen wir den Stier bei den Hörnern!

Selbst dann, wenn Sie ihr CPC664-System mit dem Ziel erwarben, anspruchsvolle Rechnerspiele zu betreiben, wären Sie wahrscheinlich über die vielen Aspekte erstaunt, die der Computer unter dem Decknamen "Hardware" noch für Sie bereithält.

Unter Hardware wird das verstanden, was Sie anheben und herumtragen können, die Haupttastatur des Computers, den Monitor, die Anschlußleitungen usw. Genaugenommen wird unter Hardware ungefähr alles das verstanden, was nicht spezifisch "Software" ist - Programme, Handbücher, bzw. Informationen auf Diskette bzw. Cassette.

Übrigens, einige Merkmale, die der Computer bietet, sind auf die Hardware zurückzuführen - wie z.B. der Farbbildschirm auf dem Fernsehgerät (bzw. Monitor). Hier liegt es wiederum an der Software, diese Möglichkeiten der Hardware einzusetzen, um Sonderzeichen und Formen auf dem Bildschirm entstehen zu lassen.

Die Hardware leitet somit den Elektronenstrahl auf der Leuchtfläche innerhalb des Fernsehschirmes und sorgt für ein "Aufleuchten" - die Software fügt Befehle und Intelligenz in der Form hinzu, daß sie die Hardware anweist, wann und in welcher Form diese Funktionen zu erfüllen sind. Hinzu kommen Zeitfunktionen, Steuerung und Taktung, um ein startendes Raumschiff oder etwas naheliegenderes wie z.B. einen Brief erscheinen zu lassen, den Sie an der Tastatur eingeben.

Worin unterscheiden sich nun Computer qualitativ?

Hardware, ohne Software, ist undenkbar. Das gleiche gilt für eine Software ohne Hardware, - der Computer erhält erst dann Wert, wenn beide zusammenwirken, um verschiedene Aufgaben zu erfüllen. Mit diesen wenigen und sehr grundsätzlichen Erwägungen lassen sich die Funktionsbereiche von Hardware und Software grob abstecken.

Im weiteren Verlauf erläutern wir nunmehr generell vereinbarte Bezugspunkte für Personal Computer:

1. Die Auflösung des Schirms - der kleinste Bereich des Bildschirms, auf den zugegriffen werden kann

Hierbei handelt es sich um eine Kombination von Faktoren, einschließlich der Zahl der verschiedenen Farben, die dem Programmierer zur Verfügung stehen, um die Zahl der bestimmten Flächen, die auf dem Bildschirm aufgelöst werden können (d.h. die Pixel) und die Anzahl der Textzeichen, die auf einer Schirmbildoberfläche dargestellt werden können.

Sie werden feststellen, daß Ihr CPC664-System innerhalb einer vergleichbaren Preisgruppe einem Vergleich mit anderen Geräten sehr gut standhält.

2. Der BASIC-Interpreter

Nahezu jeder Heimcomputer umfaßt einen BASIC-Interpreter, der es dem Benutzer ermöglicht, Aufbauprogramme unter Einsatz der Hardware-Möglichkeiten zu starten. Die eingebaute Programmsprache (BASIC), die zusammen mit Ihrer Maschine geliefert wird, stellt selbst ein Programm dar - ein immens kompliziertes und verwickeltes Programm, das in über einer Million Mann-Jahren Erfahrung entwickelt wurde, seit BASIC in den USA "erfunden" wurde. Der für Anfänger gedachte Mehrzweck-Symbolbefehlscode ist sicher die in der Welt am weitesten verbreitete Computersprache und, wie jede Sprache, umfaßt sie darüber hinaus noch eine Vielzahl lokaler "Dialekte".

Die innerhalb des CPC664-Systems verwendete Version ist eine der anpassungsfähigsten Programmsprachen von BASIC und arbeitet ebenfalls zur Steuerung von Programmen, die für den Betrieb unter Einsatz von CP/M-Arbeitsdisketten geschrieben wurden. Diese Version stellt eine sehr schnelle Verwirklichung von BASIC dar

- mit anderen Worten, sie führt ihre Rechnungen sehr schnell aus - und wenn Sie vielleicht auch nicht sehr überrascht sein mögen, daß ein Computer 0,05 Einheiten einer Sekunde benötigt, um 3 mal 5 zu multiplizieren und die Antwort darzustellen, während andere vielleicht 0,075 Sekunden benötigen, um die gleiche Aufgabe zu erfüllen - wobei ein Programm graphische Darstellungen auf den Bildschirm zeichnet, die viele Tausende von einfachen repetitiven Rechnungen auszuführen, wird die Differenz zwischen 0,05 und 0,075 einer Sekunde zu einer beträchtlichen Leistungsdifferenz.

Sie werden oft den Ausdruck "Maschinencode" hören. Ein Maschinencode ist die ursprüngliche Form des Befehlscodes, der an den Prozessor weitergegeben werden kann. Es benötigt weniger Zeit, nach der Frage zu forschen, andererseits wird das Ergebnis 5 bis 15 mal schneller als bei einer vergleichbaren Operation geliefert, die über den BASIC-Interpreter läuft. Andererseits kann es 5 bis 50 mal länger dauern, ein entsprechendes Programm als Maschinencode auszudrücken, verglichen mit der Durchführung unter Benutzung von BASIC.

Die Programmsprache BASIC in Ihrem Schneider-Computer gehört zu den schnellsten und leistungsfähigsten innerhalb irgendeines Heimcomputer-Systems und vereinigt eine solche Vielzahl von Merkmalen, die es dem erfahrenen BASIC-Programmierer ermöglichen, einige der systembedingten Nachteile eines auf hohem Stand stehenden Sprach-Interpreters zu beseitigen, um überraschende dynamische visuelle und musikalische Effekte zu erzeugen.

3. Erweiterungsfähigkeit

Bei den meisten Computern wird auf die erforderlichen zusätzlichen Hardware-Einrichtungen verwiesen wie z.B.: Drucker, Joysticks, zusätzliche Diskettenlaufwerke. Erstaunlicherweise fordern einige der erfolgreichsten Home Computer die Hinzunahme von zusätzlichen "Erweiterungsschnittstellen", bevor noch ein einfacher Drucker bzw. Joystick-Steuereinheit hinzugefügt werden kann.

Der Käufer muß nicht immer an seine zukünftigen Bedürfnisse denken, da eine Maschine, die bereits einen entsprechend vorbereiteten Paralleldruckeranschluß enthält (mit Centronics kompatibel), darüber hinaus einen Joystickanschluß für Spiele, sich als echt günstiger erweisen kann.

Der CPC664 Computer enthält eine eingebaute Drucker-Schnittstelle Centronics, eine Schnittstelle für ein zusätzliches Diskettenlaufwerk, einen Datenanschlußsockel für Cassette (+ Motorsteuerung), Anschlußmöglichkeiten für bis zu zwei Joysticks, einen Stereo-Tonausgang und eine angepaßte Erweiterungsschiene, die verwendet werden kann, um serielle Schnittstellen (RS232), MODEMs, Sprach-Synthesiser, Light-Pen, usw. anschließen zu können.

4. Toneffekte

Die akustischen Merkmale eines Computers entscheiden darüber, ob sich die Tonwiedergabe wie eine bewegte Blechkonserve anhört - oder ob sie in der Lage ist, eine entsprechende Wiedergabe eines elektronischen Musikinstrumentes zu gewährleisten.

Der CPC664-Computer arbeitet mit einem Tongenerator mit 3 Kanälen, der Tonumfang beträgt 8 Oktaven. Über die Beeinflußung von Amplitude und Hüllkurve wird eine äußerst ansprechende Musikqualität erreicht. Das 3-Kanal-System odnet einen Kanal der linken, den zweiten der rechten Seite zu, der dritte Kanal scheint akustisch aus der Mitte zu kommen.

Mit diesen Möglichkeiten können ausgefeilte Geräuscheffekte erzielt werden, ja sogar Kulisseneffekte, die simulatan zur Bewegung am Bildschirm verlaufen.

Letztendlich müssen Sie als Benutzer selbst entscheiden, welche der Eigenschaften Ihres Computers für Sie am wichtigsten ist. Wir können nur hoffen, daß Sie alles ausprobieren, damit Sie Ihren Computer optimal nutzen können.

Warum ist es nicht möglich, daß...

Von Computer-Neulingen wird, in Anbetracht der Leistungsfähigkeit der modernen Technologie, oft die Frage gestellt, warum ein so weit entwickeltes Gerät wie der CPC664, nicht in der Lage ist, die Art von Bildern auf dem Bildschirm zu bringen, die dem Standard eines jeden Fernsehgerätes entsprechen. Es wird die Frage gestellt, warum ein Computer z.B. kein Bild mit einer natürlich über den Bildschirm laufenden Person darstellen kann, warum alle Figuren wie Strichfiguren mit "Charly Chaplin-Bewegungen" aussehen.

Die Antwort ist sowohl einfach wie komplex. Die einfache Antwort ist, daß Sie nicht dazu verleitet werden sollen, zu glauben, daß der Bildschirm so raffiniert ist, wie der Schirm Ihres Fernsehgerätes. Ein Fernseher arbeitet mit "linearen" Informationen, die scheinbar eine unendlich feine Abstufung (Auflösung) zwischen hell und dunkel sowie über alle Nuancen des Farbspektrums ermöglichen. Das bedeutet, daß so ein Bildschirm rund 20 mal soviel Speicherkapazität benötigt, wie umgerechnet ein Heimcomputer für seine Video-Ausgabe braucht.

Das ist nur ein Teil des Problems, daß solche Bildbewegungen verlangen, daß diese enorme Speicherkapazität sehr schnell verarbeitet werden muß (ca. 50 mal pro Sekunde). Das erreichen nur Maschinen, die einige tausendmal teurer sind wie ein Heimcomputer - wenigstens in der nahen Zukunft!

Bis die Kosten für Hochgeschwindigkeitsspeicher drastisch fallen, und das werden sie irgendwann, müssen Kleincomputer mit vergleichbar weniger Speicher auskommen, um den Bildschirm zu steuern. Das bringt niedrigere Auflösung und linkischere Bewegungen. Überlegte Hardware-Planung und gute Programmierung können viel dazu beitragen, diese Situation zu mildern, aber wir sind noch sehr weiter davon entfernt, auf billigen Computern realistische Bewegungen und lebendige Bilder darzustellen, wenigstens so wie ein Zeichentrickfilm.

Diese Tastatur ist ein so gewohntes Bild...

Warum kann ich nicht einfach zum Computer gehen und eine Seite einfachen Text direkt in die Maschine eingeben?

Lassen Sie sich nicht dadurch verleiten, daß der Computer wie eine Schreibmaschine mit einer elektronischen Anzeige aussieht. Der Bildschirm ist kein Stück elektronisches Papier, d.h. er gibt Ihnen einfach nur die Möglichkeit, mit der Programmiersprache (und den Programmen) im Speicher der Maschine Kontakt aufzunehmen.

Solange Sie ihm nichts anderes sagen, versucht der Computer sämtliche Zeichen, die Sie über die Tastatur eingeben, als Programmieranweisungen zu interpretieren. Wenn Sie auf ENTER drücken, schaut sich der Computer das an, was Sie eingetippt haben, um zu sehen, ob es einen Sinn für sein eingebautes BASIC gibt, wenn nicht, so weist er die "Eingabe" mit dem Kommentar zurück:

```
Syntax error
```

Wenn Sie aber gerade ein Text-Verarbeitungsprogramm aktiv im Speicher haben, dürfen Sie beliebig Wörter eintippen, auf (ENTER) drücken und weitermachen, als ob Sie auf einem elektronischen Stück Papier in einer elektronischen Schreibmaschine arbeiten würden. Aber dazu müssen Sie erst ein Text-Verarbeitungsprogramm in den Computerspeicher von einem Datenträger (Disketten/Cassetten) geladen haben. Der Computer "scheint" mehrere Gerätekomponenten zu vereinen, die zu Hause und im Büro bekannt sind - der fernsehartige Bildschirm und die Tastatur. Sie haben jedoch daran zu denken, daß diese Ähnlichkeiten nur oberflächlicher Natur sind und daß der Computer eine Kombination vertraut aussehender Hardware darstellt, allerdings mit seiner ganz eigenen Persönlichkeit!

Wer hat Angst vor der Fachsprache?

Wie in allen "spezialisierten" Industrien hat die 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", Fachsprache und Spezialbegriffen verstecken - die meisten Berufe und Arbeitsgebiete haben ihre eigene "Sprache", wogegen wir alle schon einmal zu kämpfen hatten.

Der größte Unterschied ist, wie bei den Juristen, daß die benutzten Wörter nicht so schwierig sind, nur die Art, wie sie gebraucht werden. Die Leute, die mit der Datenverarbeitung groß geworden sind, versuchen diese Begriffe unkompliziert anzuwenden, um so die Komplexität zu verkleinern.

Lassen Sie sich nicht durch die "einfache Sprache" in der Datenverarbeitung in die Irre führen, sie ist kein literarisches Thema, sondern eine präzise Wissenschaft. Abgesehen von der "Syntax" der Sprache, ist ihre Struktur sehr einfach und absolut nicht verwirrend oder widersprüchlich. Lehrer der Datenverarbeitung haben es bisher noch nicht versucht, eine Art Standard zur genauen Analysierung eines Programms zu entwickeln.

Trotzdem können Programme analysiert werden. Obwohl die Bedeutung eines Programms nicht auf Anhieb klar sein kann, sticht sofort ins Auge, ob das Programm elegant oder unübersichtlich ist. Heute wird wesentlich mehr Nachdruck auf ordentlichen Programmaufbau gelegt, als es während des Anfangsdurcheinanders

der Micro-Revolution üblich war.

Die Datenverarbeitung wird schnell von vielen jungen Leuten verstanden, da sie für die Genauigkeit und Einfachheit der Ideen einen Sinn haben, ebenso wie für die Art, wie sie weitergegeben werden. Sie werden kaum einen zehnjährigen Anwalt finden - aber Sie finden sehr viele zehnjährige Programmierer!

Grundlagen des BASIC

Fast alle Heimcomputer bevorzugen als Programmiersprache BASIC, womit Programme in fast normaler Sprache geschrieben werden können. BASIC hat nunmehr keine besondere Bedeutung mehr hinsichtlich des Schwierigkeitsgrades, 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 Programmiersprache, die den Umfang der erlaubten Kommandos interpretiert und dann Operationen an Daten während des Programmablaufes ausführt. Während der menschliche Wortschatz 5 bis 8000 Worte umfaßt (zuzüglich der verschiedenen Anwendungen von Verben usw.), kommt BASIC mit ca. 200 Worten aus. Programme, die mit BASIC geschrieben werden, unterliegen den starren Regeln für die Anwendung dieser Worte. Die Syntax ist sehr präzise und jeder Versuch, eine andere Schreib- oder Sprachweise zu verwenden, endet mit der kalten und nüchternen Meldung:

```
Syntax error
```

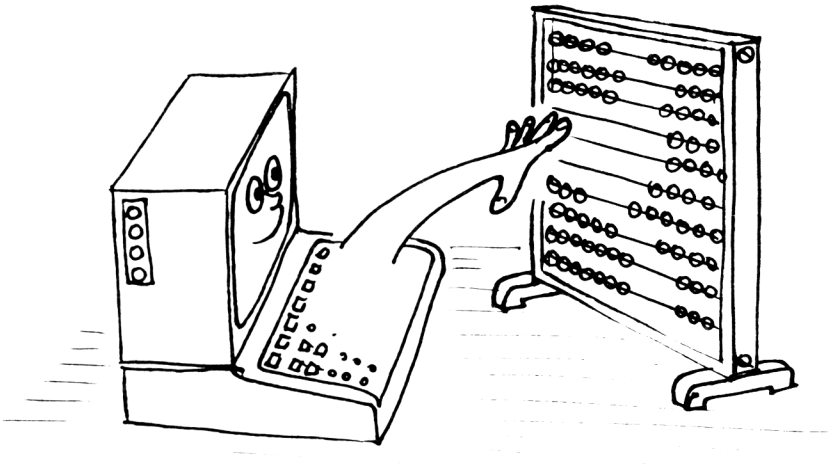
Dies ist nicht so einschränkend, wie es auf den ersten Blick erscheint, da die BASIC Sprache (Syntax) hauptsächlich für die Bearbeitung numerischer Daten gedacht ist. Die Worte sind im wesentlichen nur Erweiterungen der bekannten mathematischen Operationen wie +/- usw. Das Wichtigste für einen Anfänger ist es, zu akzeptieren, daß der Computer nur mit numerischen Daten arbeiten kann, da jede Information nur in der Form von numerischen Daten an die CPU-Einheit (Central Processor Unit) übergeben werden kann.

Zahlen bitte!

Würde ein Computer dazu benutzt, die kompletten Werke von Shakespeare zu speichern, dann wäre kein einzige Buchstabe oder kein einziges Wort im ganzen System zu finden. Jedes Stück der Information wird erst in eine Zahl umgewandelt, mit welcher der Computer arbeiten kann.

BASIC interpretiert die Worte als Zahlen, die der Computer dann mit Addition, Subtraktion und Eigenschaften der Bool'schen Logik behandeln kann, so daß er die Daten vergleichen und einige Teile davon auswählen kann, um z.B. zu prüfen, ob eine Zahl größer als die andere ist oder um einen bestimmten Ablauf einzuleiten, falls die eine ODER andere Zahl dies verlangt.

Mittels des Programms wird jede Aufgabe vom Computer in eine Serie von einfachen Ja/Nein-Operationen zerlegt.



Wenn Ihnen dieser Prozeß umständlich vorkommt, dann haben Sie recht und Sie haben die erste und wichtigste Wahrheit der Datenverarbeitung entdeckt. Ein Computer ist ein Werkzeug zur Ausführung allereinfachster, immer wiederkehrender Aufgaben - sehr schnell und absolut präzise. Also, BASIC interpretiert Anweisungen in einem Programm und wandelt sie in eine Sprache um, welche die CPU (Zentrale Bearbeitungseinheit) behandeln kann. Der Computer versteht nur zwei Zustände: "Ja" und "Nein", die in binärer Schreibweise mit "1" oder "0" dargestellt werden. Es gibt in der Bool'schen Logik nur "wahr" oder "falsch", kein "gegebenenfalls" oder "vielleicht"!

Das Umschalten zwischen diesen beiden bestimmten Zuständen ist die wesentlichste Bedeutung von "digital"; in der Natur bewegen

sich Prozesse nur allmählich von einem "stabilen" Zustand zum anderen in einer glatten Weise. Mit anderen Worten, der Übergang erfolgt hierbei auf einem langen Weg zwischen den beiden Zuständen, in einer reinen digitalen Umgebung nimmt dieser Übergang keine Zeit in Anspruch - wobei jedoch physikalische Grenzen der Mikroelektronik eine minimale Verzögerung verursachen, die sogenannte Laufzeitverzögerung. Die Summe vieler dieser Laufzeitverzögerungen ist der Grund, warum es etwas Zeit erfordert, bis die Information bearbeitet ist und eine Antwort ausgegeben wird.

In diesem Falle muß der Computer eine bestimmte Zeit warten, bis eine Aufgabe fertig ist und er deren Ergebnis für weitere Aufgaben weiterverwenden kann.

Eine digitaler Prozeß kennt nur schwarz oder weiß, wobei der Übergang in Grautönen bedeutungslos ist; demgegenüber führt der lineare bzw. "analoge" Übergang über alle grauen Töne.

Wenn die letztendliche Antwort nur 0 oder 1 sein kann, so kann es keine "fast" richtige Antwort geben. Daß es manchmal so aussieht, als ob der Computer Fehler bei der Behandlung von numerischen Daten macht, hängt damit zusammen, daß der Computer in der Größe der Darstellung einer Zahl begrenzt ist. Numerische Daten, die diese Grenzen überschreiten, werden "abgeschnitten", was zu Rundungsfehlern führen kann. Dadurch kann 999,999,999 dann 1,000,000,000 werden.

Erhebt sich die Frage, wie in einer Welt, in der es nur 0 oder 1 gibt, über 1 hinausgerechnet werden kann?

Bits und Bytes

Wir haben uns daran gewöhnt, mit Zahlen im Dezimal-System umzugehen, wobei der Bezugspunkt die Zahl 10 ist, d.h. Werte werden mit zehn Ziffern im Bereich von 0 bis 9 dargestellt (0 bis 9 ist der Schreibweise von 1 bis 10 vorzuziehen). Das Zahlensystem mit Zahlen im Bereich von 0 bis 1 ist das Binär-System und die Einheiten, womit das System arbeitet, heißen Bits, kurz für BInary DigiT".

Der Zusammenhang zwischen Bits und der dezimalen Schreibweise ist einfach zu verstehen:

Es ist normal, die maximale Anzahl von Binärziffern im Gebrauch anzugeben und führende Nullen zu schreiben, so daß immer die ganze Anzahl der Bits verwendet wird:

d.h. dezimal 7 wird:

00111 Binär

...in 5-Bit-Schreibweise.

Im Binärsystem können die Ziffern als Anzeigen in Spalten verstanden werden, die aussagen, ob die entsprechenden Potenzen von 2 vorhanden (1) sind oder nicht (0).

$2^0 = 1$

$2^1 = 2 = 2 = 2(2^0)$

$2^2 = 4 = 2 \times 2 = 2(2^1)$

$2^3 = 8 = 2 \times 2 \times 2 = 2(2^2)$

$2^4 = 16 = 2 \times 2 \times 2 \times 2 = (2^3)$

So daß die Spalten wie folgt aussehen können:

2^4		2^3		2^2		2^1		2^0		
1		0		0		1		1		
(16	+	0	+	0	+	2	+	1)	=	19

Eine Kurzschreibweise, um Binärziffern darstellen zu können, ist der Begriff "Byte" für 8 Bits. Die größe 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 Dateien in Vielfachen von 8 Bits. 256 ist keine so große Zahl, so daß zur Adressierung des Speichers zwei Bytes verwendet werden. Der Speicher kann als Bereich mit einer horizontalen und vertikalen Adresse betrachtet werden, in dem jedes Element folgendermaßen angesprochen werden kann:

	0	1	2	3	4	5	6	7	8	9
0										
1										
2										
3										
4										
5				1		1				
6										
7										
8										
9										

Dieser Bereich kann bis zu (10 x 10) Informationseinheiten aufnahmen, deren Adressen jeweils im Bereich von 0 bis 9 liegen. Auf der Position 3,5 ist "1" gespeichert, auch auf 5,5.

Ein Bereich mit 256 x 256 Einheiten umfaßt 65,536 einzelne Positionen, mit mit 8-Bit-Adressen für die waagrechten und senkrechten Achsen angesprochen werden können. Mit unserem "0" und "1" können wir nun 65,536 Elemente adressieren.

Die nächste Ebene der Kurzschreibweise ist Kilobyte (k-Byte oder einfach "K"), was 1024 Bytes bedeutet. 1024 ist das dem dezimalen Begriff "Kilo" nächstgelegene binäre Vielfache und erklärt, warum ein Computer mit "64K" Speicher in Wirklichkeit mit 65.536 Bytes (64x1024) Speicher ausgestattet ist.

Glücklicherweise führt der BASIC Interpreter alle notwendigen Konvertierungen für Sie durch, so daß Sie ein fähiger Programmierer werden können, ohne das Binärsystem vollständig verstehen zu müssen. Sollten Sie das Binärsystem voll begriffen haben, wird es Ihnen helfen, schneller die "mysteriösen" bzw. signifikanten Zahlen zu verstehen, die unvermeidlich auftauchen werden, wenn Sie sich durch die Wissenschaft der Datenverarbeitung durcharbeiten.

Es lohnt sich, die Mühe aufzuwenden, um das Binärsystem zu verstehen und die verschiedenen Zahlen wie 255, 1024 usw. zu erkennen, da es sehr unwahrscheinlich ist, daß sich diese Grundlagen der Computerarbeitsweise in absehbarer Zeit ändern werden. Die Sicherheit und Einfachheit, die durch die Arbeit mit nur zwei Zuständen entsteht, wird über die sich steigernde Komplexität siegen, die mit jedem anderen Zahlensystem gegeben wäre.

Jedoch...

Die binäre Schreibweise ist denkbar einfach und elegant, aber sie ist aufwendig zu schreiben und verleitet zu Fehlern beim Lesen. Das Binärsystem ist mit anderen Zahlensystemen verwandt, die als Kurzschreibweise für Programmierer dienen. Ein solches Zahlensystem, das im Zusammenhang mit Mikrocomputern verwendet wird, heißt HEX (Kurzform von Hexadezimal).

Dabei basieren die Zahlen auf der Zahl 16 (0 bis 15) und 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 Vier-Bit-Zahl ist: 1111 binär. Der erste Block enthält die Anzahl kompletter "15er"-Einheiten, der zweite Block enthält den "Divisionsrest" - wobei hier die ganze Eleganz des Binärsystems zum Tragen kommt.

Hier noch einmal die Tabelle zur Einführung in die Binärschreibweise:

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 11010110 (&D6 hex) kann in zwei 4-Bit-Zahlen (als Halbbytes bezeichnet), unterteilt werden. In diesem Handbuch werden Hex-Zahlen immer mit dem Symbol "&" gekennzeichnet, z.B. &D6, und dieses Zahlensystem wird auch von den meisten Programmierern in einer Assembler-Sprache benutzt. Die Assembler-Sprache bietet den engsten Kontakt zum Maschinencode überhaupt, da hierbei einfache Buchstabenkurzzeichen anstelle der echten Maschinencodezahlen benutzt werden.

Wenn Sie mit HEX arbeiten, müssen Sie die erste Ziffer zur Bestimmung der Anzahl der 16 im endgültigen Ergebnis verwenden, dann die zweite Ziffer dazu addieren, um den endgültigen Dezimalwert auszurechnen. Es gibt eine starke Neigung, eine Zahl wie &D6 als 13+6 oder 136 zu betrachten. Aber richtig ist (13x16)+(6) = 214.

Es ist genauso wie beim Dezimalsystem, in dem man z.B. für 89 liest: (8x10)+(9). Die Multiplikation mit 10 kommt Ihnen einfacher vor, weil Sie viel mehr Übung damit haben als mit 16.

Wenn Sie so weit gekommen sind, ohne allzu sehr verwirrt zu sein, dann haben Sie einen guten Start gemacht, um die Grundlagen des Computers zu begreifen. Und wenn Sie sich fragen,

wozu das alles - dann haben Sie recht. Ein Computer ist eine Maschine, die mit sehr einfachen Ideen und Konzepten arbeitet, aber das sehr schnell (Millionen von Malen je Sekunde) und mit einer enormen Kapazität, sich beide zu merken, welche Daten eingegeben wurden, sowie die Abertausend von Zwischensummen, die auf dem Weg zum endgültigen Ergebnis entstehen.

Um die Theorie über Ihren Computer weiter zu vertiefen, stehen Tausende von Büchern zur Verfügung. Einige werden Sie nur verwirren, andere dagegen werden Sie sehr viel weiter bringen und Ihnen enthüllen, wie schlicht und einfach die grundlegenden Zusammenhänge zwischen Zahlensystemen und deren Behandlung in Ihrem Computer sind.

Teil 2: Weitere Informationen, insbesondere über das CPC664-System

Dieser Abschnitt behandelt noch einige Maschinen-spezifische Aspekte des CPC664-Systems. Unterstützende Informationen zu diesen Sachgebieten sind im Grundlagenkurs und im Kapitel "Aufstellung der Schneider CPC664-BASIC-Schlüsselwörter" enthalten.

In diesem Kapitel finden Sie:

- Zeichensatz
- ASCII
- Variable
- Logik
- Anwendereigene Zeichen
- Druckformatierung
- Fenster
- Interrupts
- Daten
- Toneffekte
- Graphische Symbole
- Maschinenspezifische Hardware

Ein Bit eines Zeichens...

Wenn Sie an Ihrer Tastatur CPC664 Eingaben vornehmen, so sollten Sie es nicht einfach als gegeben betrachten, daß auf dem Bildschirm erkennbare Schriftzeichen und Buchstaben usw. erscheinen. Darüber hinaus haben wir bereits die Tatsache diskutiert, daß es sich bei Ihrem Computer nicht einfach um eine Schreibmaschine handelt.

Was sich tatsächlich ereignet, ist ein Ergebnis davon, daß Sie eine Kombination elektrischer Schalter betätigen.

Die beim Betätigen dieser Schalter erzeugten elektrischen Signale werden von der ausrüstungsinternen Schaltung in eine Punktfolge auf dem Schirm umgesetzt. Wir erkennen diese Folge von Punkten als Buchstabe, Nummer bzw. als ein anderes Zeichen innerhalb des Zeichensatzes des CPC664-Systems.

Einige der Zeichen, die Sie sehen, können nicht direkt über die Tastatur eingegeben werden, sondern nur mit Hilfe der PRINT CHR$ (Zahl)-Funktion gedruckt werden. Das kommt daher, daß alle Informationen in einem Computer in sogenannten Bytes

gespeichert werden. Wie in Teil 1 dieses Kapitels behandelt, bietet ein Byte die Möglichkeit, 256 verschiedene Kombinationen von Zeichen darzustellen. Da der Computer zumindest ein ganzes Byte für jedes gespeicherte Zeichen einzusetzen hat (unabhängig davon, ob wir Wert darauf legen oder nicht - es handelt sich um kleinste Bezeichnung, die das CPC664-System aufnimmt), können also 256 unterschiedliche Zeichen gebildet werden, von denen aber auf den üblichen Tastaturen und Druckern nur ca. 96 verwendet werden, - somit bleiben 160 Möglichkeiten ungenutzt.

Der Standardbereich des Zeichenvorrates ist als "Untermenge" bekannt. Er wird überall in der Computerwelt als das "ASCII-System" bezeichnet, ein Ausdruck, abgeleitet aus "American Standard Code for Information Interchange". Es handelt sich hierbei im wesentlichen um ein System, das die Übertragung von Daten von einem Computer zum anderen in erkennbarer Form sicherstellt. Wir empfehlen Ihnen daher, den Abschnitt "Wenn Sie Zeit haben" aufzusuchen, der den gesamten ASCII-Zeichensatz wiedergibt, in Verbindung mit allen weiteren Zeichen, die es auf dem CPC664 gibt, mit Angabe Ihrer jeweiligen Codenummer.

Wie können wir uns hier Einblick verschaffen...

Auf der Grundlage der vorangegangenen Erläuterungen, werden Sie wahrscheinlich mit folgendem Programm vertraut sein:

```
10 FOR n=32 TO 255
20 PRINT CHR$(n);
30 NEXT
```

...somit läßt der Computer das entsprechende Zeichen auf dem Bildschirm erscheinen. Lassen Sie uns nunmehr zum Grund dieses kleinen Programms vordringen:

Zuerst ist festzustellen, daß der Computer nicht für den Druckvorgang (PRINT) angewiesen wurde: "absdefghijklmn...usw."; anstelle dessen wurde er aufgefordert, CHR$(n) zu drucken (PRINT). n erscheint als bequeme Kurzform zur Bezeichnung einer "Variablen". Eine Variable ist eine "Informationszeile" im Computerm deren Wert je nach Programmieranweisung verändert werden kann. (Die Wahl des Buchstaben n für Variable ist willkürlich - es kann sich hierbei um jeden anderen Buchstaben handeln, so lange dieser kein Schlüsselwort bezeichnet.)

Womit können Sie ausdrücken, worin eine Veränderliche besteht?

Eine Zahl wie 5 hat einen festen Wert, der zwischen 4 und 6 liegt, die ist also keine Variable. Das Zeichen n hat auch einen festen Wert - es ist ein Buchstabe aus dem Alphabet.

Woraus erkennt nun 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 Nachricht geantwortet: Syntax error - da er nicht die Befehlsfolge FOR "n"=32 TO 255 erkennt.

Einfach durch die Tatsache bedingt, daß wir n ohne Anführungszeichen verwenden, haben wir dem Computer mitgeteilt, daß n eine Variable ist. Die Anweisung FOR erwartet nach dem Schlüsselwort eine Variable und akzeptiert damit die nachfolgende Buchstabenkombination als Variable, wenn sie den Regeln entspricht.

Wir haben dem Computer fernerhin mitgeteilt, daß n = 32 TO 255 ist. Somit haben wir den Bereich der Variablen festgelegt. Es handelt sich in der Tat um eine Folge, die mit 32 beginnt und mit 255 endet.

Nachdem diese Variable definiert ist, muß dem Computer gesagt werden, was mit ihr geschehen soll - Zeile 20 übernimmt diese Funktion:

```
20 PRINT CHR$(n);
```

Hiermit wird angegeben, daß unabhängig vom laufenden numerischen Wert von n der Computer in seinem Speicher nachzuprüfen hat, welcher Zeichenwert diesem Wert entspricht und das Zeichen am Bildschirm ausdruckt.

Der Strichpunkt am Zeilenende verhindert, daß nach jedem Zeichen, das angezeigt wird, eine neue Zeile angefangen wird. (Ansonsten wird jedes neue Zeichen in der ersten Spalte einer neuen Zeile ausgedruckt.)

Zeile 30 weist den Computer an, daß nach Ausführung der Aufgabe mit dem ersten Wert von n in der Folge (bei dem es sich um 32 handelt) er auf die Zeile zurückzukehren hat, wo sich FOR befindet, wonach die Anweisung mit dem nächst höheren Wert (NEXT) der Variablen n durchgeführt wird. Diesen Vorgang nennt man Schleife (loop), eine sehr wichtige Technik beim Programmieren. Diese Schleifen vereinfachen viele Vorgänge und Sie werden diese Technik sicher häufig einsetzen, wenn Sie die Vor-

züge erkannt haben.

Wenn diese FOR-NEXT-Schleife ihren Grenzwert (255) erreicht hat, wird der Vorgang unterbrochen und der Computer prüft nach der nächsten Zeile nach Zeile 30 - da jedoch hier keine weitere Anweisung mehr vorliegt, kehrt er zur Direktbetriebsart zurück und meldet sich mit Ready. Hiermit wird angezeigt, daß der Computer bereit ist, weitere Anweisungen zu empfangen oder aber es kann das Programm (ENTER) erneut eingeführt und bis zum Abschluß des Programms wiederholt werden.

Das Programm wird im Speicher sicher abgespeichert und verbleibt hier, bis Sie dem Computer andere Anweisungen erteilen bzw. Letzteren abschalten. Dieses Programm zeigt Wesentliches in Bezug auf das Arbeiten mit Rechnern - d.h. alles was der Computer intern tut, beruht auf Zahlen. Der Computer hat seinen Zeichenvorrat ausgedruckt, indem er jeweils eine Zahl für das gewünschte Zeichen benutzt hat. Wenn Sie die Taste A drücken, erkennt der Computer anhand des entsprechenden Wertes der Tasten in seinen Tabellen, daß diese Taste A darstellt und gibt ein entsprechendes Zeichen aus. Die jeweilige Datenstelle ist durch einen Zahlencode definiert, der durch Eingabe an der Tastatur aktiviert wird. Jedes Zeichen hat eine entsprechende Nummer, die in Teil 3 des Kapitels "Wenn Sie Zeit haben..." aufgeführt sind.

In gleicher Weise hat auch der dargestellte Buchstabe nichts mit dem "Schreiben" des Buchstabens am Bildschirm zu tun, auch hier handelt es sich wiederum nur um Nummern.

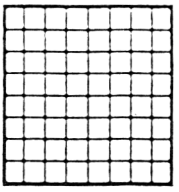
So lautet beispielsweise der ASCII-Code für den Buchstaben "A" 97. Der Computer versteht auch 97 nicht (dieser fiese Kerl), sondern setzt diese Zahl in den sogenannten Maschinencode um. Zu Beginn dieses Kapitels werden die diesem Maschinenaspekt zugrundeliegenden Prinzipien erläutert.

Eingangs erscheint die Umsetzung von dezimalen Zahlen aus unserem täglichen Gebrauch in den Maschinencode schwer verständlich, da wir an unser Zahlensystem gewöhnt sind. In Werten zu denken, die auf dem Prinzip der Zehnereinheiten beruhen, ist so natürlich, wie wir Messer und Gabel mit verschiedenen Händen benutzen.

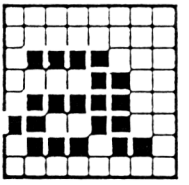
Die gleiche Selbstverständlichkeit sollten wir auch beim Umgang mit hexadezimalen Zahlen erreichen, dann wird uns manches leichter fallen und verständlicher werden. Sollte es Ihnen noch an Sicherheit in Bezug auf die binären und hexadezimalen Numerierungssysteme fehlen, so empfehlen wir Ihnen, gründlich Teil 1 dieses Kapitels noch einmal durchzuarbeiten (wenn Sie es nicht

schon getan haben).

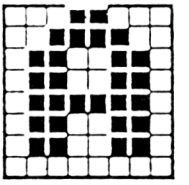
Der Computer übersetzt ein eingetipptes A in eine Zahlendarstellung, die er versteht und entnimmt dann einer Tabelle eine andere Folge von Ziffern, die das Zeichen definiert. Das Zeichen, das man am Bildschirm sieht, wird durch einen Datenblock erzeugt, der im Speicher als numerische Matrix abgelegt ist:



Raster für
ein Leerzeichen



Kleines A



Grosses A

Die Elemente dieser Matrix sind Spalten und Zeilen mit Punkten. Das Zeichen wird durch ein entsprechendes Punktmuster dargestellt, das im Computer gespeichert ist. 8 Zeilen und 8 Spalten bilden beim CPC664 ein Zeichen " (dargestellt durch je ein Byte). Wenn Sie ein bestimmtes Zeichen nicht finden, aus dem Satz von 255, so können Sie Ihre eigenen Zeichen selbst durch das im weiteren Verlauf dieses Abschnittes beschriebene Schlüsselwort SYMBOL definieren.

Die vom Benutzer selbst definierten Zeichen können aus Kombinationen von 0 bis 64 Punkten dargestellt werden; hierbei in beliebiger Form geordnet - so daß die "komplette" Buchstabenfolge, die alle möglichen Kombinationen dieses Rasters benutzt, noch viel mehr verschiedene Zeichen umfassen kann. Sie können somit Blöcke von Zeichen zusammenfassen, um noch einen größeren Zeichenblock zu bilden, wobei die Anzahl der Möglichkeiten der somit selbst definierten Zeichen allein durch Ihre Zeit und Ihren Einfallsreichtum begrenzt werden.

Logische Ausdrücke

Ein grundlegender Unterschied zwischen einer Rechenmaschine und einem Computer ist die Fähigkeit des Computers, logische Ausdrücke wie IF THEN bearbeiten zu können. Die Daten, die mit diesen logischen Ausdrücken verarbeitet werden, werden intern als Bit-Muster dargestellt und Bit für Bit einzeln abgearbeitet.

Die Art und Weise, wie dies geschieht, ist in dieser kurzen Definition schwierig zu beschreiben. Und zwar logisch, mit einfachen Worten, ohne die Anwesenheit präziser Definitionen.

Die zwei Hälften eines logischen Ausdrucks werden Argumente genannt. Ein logischer Ausdruck umfaßt:

"Argument" "Vergleichsoperator" "Argument"

wobei:

"Argument" ist: NICHT "Argument"
oder: "numerischer Ausdruck"
oder: "Bezugsbedingung"
oder: ("logischer Ausdruck")

Beide Argumente für einen Vergleichsoperator müssen ganze Zahlen sein. ERROR 6 wird angezeigt, wenn dies nicht der Fall ist.

Die logischen Operatoren in der Reihenfolge ihrer Bearbeitung und ihre Auswirkung auf jedes Bit sind:

AND Ergebnis ist 0, wenn nicht beide Argumentbits 1 sind

OR Ergebnis ist 1, wenn nicht beide Argumentbits 0 sind

XOR Ergebnis ist 1, wenn nicht beide Argumentbits gleich sind.

AND ist der meist benutzte Operator und bedeutet nicht dasselbe wie "Addieren".

```
PRINT 10 UND 10
```

führt zum Ergebnis 10.

```
PRINT 10 UND 12
```

führt zum Ergebnis 8.

```
PRINT 10 AND 1000
```

führt wieder zum Ergebnis 8.

Dies kommt daher, weil beide Zahlen, 10 und 1000, in ihre Binärdarstellung umgewandelt wurden.

```
      1010
1111101000
```

Dann wird bei einer AND-Operation Bit für Bit verglichen. Und wenn das Bit oben AND, die untere Reihe 1 liefert, so lautet die Antwort 1:

```
0000001000
```

...somit unser Ergebnis 8. Der logische Operator AND wird eingesetzt, um zu erfassen, wenn zwei Bedingungen gleichzeitig vorliegen. Hier ergibt sich ein selbst erklärendes Beispiel:

```
10 INPUT "Geben Sie den heutigen Tag ein";day
20 INPUT "Geben Sie den Monat ein"; Monat
30 IF day=25 AND month =12 THEN 50
40 CLS:GOTO 10
50 PRINT "Fröhliche Weihnachten!"
```

OR vergleicht ebenfalls Bits. Das Ergebnis ist immer 1, außer wenn beide Bits 0 sind. In diesem Fall ist das Ergebnis 0. Nehmen wir die gleichen Zahlen wie beim AND-Beispiel:

```
PRINT 1000 OR 10
 1002
```

Bitmuster:

```
       1010
1111 10 1000
```

Das Ergebnis ist:

```
111110  1010
```

Und im Programmbeispiel:

```
10 CLS
20 INPUT "The number of the month";month
30 IF month= 12 OR month=1 OR month=2 THEN 50
40 GOTO 10
50 PRINT "It must be winter!"
```

Der NOT-Operator setzt jedes Bit in Argument um (0 wird 1 und umgekehrt):

```
10 CLS
20 INPUT"Geben sie den laufenden Monat ein ";monat
30 IF NOT (monat=6 OR monat=7 OR monat=8) THEN 50
40 GOTO 10
50 PRINT"Es ist Sommer !"
run
```

Bedenken Sie, daß Sie mehrere logische Bedingungen miteinander verknüpfen können, um zu einer klaren Entscheidung zu kommen (bis zur maximalen Zeilenlänge):

```
10 INPUT"Geben sie den heutigen Tag ein ";day
20 INPUT"Geben sie den Monat ein ";monat
30 IF NOT (monat=12 OR monat=1) AND day=29 THEN 50
40 CLS:GOTO 10
50 PRINT "Dieses Jahr ist ein Schaltjahr !"
run
```

Das Ergebnis eines Vergleichsausdrucks ist entweder -1 oder Ø. Die Bitdarstellung für -1: Alle Bits der Ganzzahl = 1; für 0 sind alle Bits 0. Das Ergebnis einer logischen Vergleichsoperation bei zwei solchen Argumenten ergibt -1 für "wahr" (True) oder 0 für "falsch" (False).

Prüfen Sie dies, indem Sie dem obigen Programm Zeile 60 hinzufügen:

```
60 PRINT NOT (monat=12 OR monat=1)
70 PRINT (monat=12 OR monat=1)
```

...und wenn das Programm läuft, geben Sie 29 für den Tag und 2 für den Monat ein. Dann wird die Meldung aus Zeile 50 angezeigt und die Ergebnisse des Vergleichs werden in Zeile 60 und 70 darunter angezeigt.

Schließlich ist das Ergebnis von XOR (exklusives OR) wahr, wenn beide Argumente unterschiedlich sind.

Die folgende Tabelle zeigt alle Möglichkeiten, die bei logischen Vergleichen existieren; dies ist eine gute Demonstration dessen, was bei bitweisen logischen Operationen passiert:

Argument A	1010
Argument B	0110
UND-Ergebnis	0010
ODER-Ergebnis	1110
XOR-Ergebnis	1100

Benutzereigene Zeichen

Eine der ersten Anwendungen der Binärziffern, mit der Sie wahrscheinlich zu arbeiten haben, besteht in der Zuordnung der Buchstaben bei Einsatz des SYMBOL-Kommandos. Wird das Zeichen auf einer 8/8-Matrix vorbereitet, dann kann jede der 8 Reihen in einen Binärwert umgesetzt werden, indem man 1 für jedes Pixel einsetzt, das zu färben ist und eine Null für jedes, das unsichtbar zu bleiben hat, d.h. in der Farbe des Bildschirms zu erscheinen hat. Diese acht Ziffern werden anschließend zum SYMBOL-Kommando weitergeleitet. Um beispielsweise einen Hausumriß zu zeichnen:

```
   *     = 00001000 = &08 =                8        = 8
  ****   = 00111100 = &3C =          32+16+8+4      = 60
 *    *  = 01000010 = &42 =      64              +2 = 66
* *  * * = 10100101 = &A5 = 128   +32      +4   +1 = 165
*      * = 10000001 = &81 = 128                 +1 = 129
* ** * * = 10110101 = &B5 = 128   +32+16  +4   +1 = 181
* **   * = 10110001 = &B1 = 128   +32+16       +1 = 177
******** = 11111111 = &FF = 128+64+32+16+8+4+2+1 = 255
```

lautet die Steuerung:

```
SYMBOL 240, 8,60,66,165,129,181,177,255
```

...bzw.:

```
SYMBOL 240,&08, &3C, &42, &45, &81, &B5, &B1, &DD
```

...bzw.:

```
SYMBOL 240,&X00001000, &X00111100, &X01000010, &X10100101,
           &X10000001, &X10110101, &X10110001, &X11111111
```

Für die Eingabe des benutzereigenen Zeichens ist einzugeben:

```
PRINT CHR$(240)
```

Um abschließend Zeichenblöcke miteinander zu verbinden, können Sie eingeben:

```
semi$=CHR$(240)+CHR$(240)
PRINT semi$
```

bzw.:

```
terrace$=STRING$(15,240)
PRINT terrace$.
```

Druckaufbereitung

Der PRINT-Befehl kann auf verschiedene Weise eingesetzt werden.

1. Die zu druckende Zahl, Zeichenfolge oder Variable steht direkt - nach einer Leerstelle - hinter dem Befehl

```
PRINT 3
 3

PRINT "Hallo"
Hallo

a=5
PRINT a
 5

a$="Test"
PRINT a$
Test
```

2. Mehrere Druckausgaben können in einer PRINT-Anweisung zusammengefaßt werden. TAB, SPC, ein Semikolon oder ein Komma grenzen sie gegeneinander ab. Das Semikolon bewirkt eine fortlaufende Ausgabe. Wenn Sie ein Komma verwenden, sieht es anders aus:

Die Begriffe werden dann in Felder ausgegeben, die jeweils 13 Zeichen lang sind. Ist der Begriff kürzer als 13 Stellen, wird der Rest des Feldes mit Leerstellen aufgefüllt.
Die Feldgröße kann mit dem Befehl ZONE verändert werden.

```
PRINT 3;-4;5
 3 -4 5
PRINT "Hallo";"Freund"
Hallo Freund
PRINT "Hallo","Freund"
Hallo          Freund
PRINT 3,-4,5
 3             -4            5
ZONE 4
PRINT 3,-4,5
 3 -4     5
```

Anmerkung: Jede Zahl wird mit einer nachfolgenden Leerstelle ausgegeben. Bei negativen Zahlen steht das Minuszeichen voran, bei positiven Zahlen eine Leerstelle.
Sämtlichen Ausdrücken folgt ein Zwischenraum. Strings werden genau in der Form gedruckt, in der sie zwischen "Anführungszeichen" erscheinen.

Die Funktion SPC nimmt einen numerischen Ausdruck als Parameter an und veranlaßt eine Zahl von Leerräumen entsprechend der Angabe durch den Ausdruck. Ist der Wert negativ, so wird Null angenommen, ist er größer als die Breite des laufenden Stromes (Window), dann wird die Strombreite angenommen.

```
PRINT SPC(5)"hi"
     hi
x=3
PRINT SPC(x*3)"hi"
         hi
```

Die Funktion TAB ist sehr vergleichbar, mit der Ausnahme, daß die erforderliche Zahl von Leerstellen gedruckt wird, so daß der zu druckende Ausdruck in der angegebenen Spalte erscheint.

Der Strom, in dem der gesamte Druckausdruck erscheint, ist Fenster 0, es sei denn, daß ein Leitsymbol (#) eingefügt wurde, bevor die Ausdruckfolge gedruckt wird. Weitere Ströme können für den Ausgang zu anderen Fenstern eingesetzt werden. Die Ströme 8 und 9 stellen besondere Fälle dar - die gesamte Druckfolge auf Strom 8 erscheint auf dem Drucker (sofern angeschlossen). Strom 9 leitet den Ausgang zu einer Disketten-(bzw. Cassetten)-Datei. Hierzu ist anzumerken, daß das WRITE-Kommando für diesen Zweck anstelle des PRINT-Kommandos einzusetzen ist.

```
PRINT "hello"
hello                              -Fenster 0

PRINT # 0,"hello"
hello                              -ebenfalls Fenster 0

PRINT # 4,"hello"
hello                              -Fenster 4
                                   (am oberen Schirmrand)

PRINT # 8,"hello"
hello                              -auf dem Drucker
                                   (sofern angeschlossen)
```

TAB und SPC eignen sich für einfache Druckformate, wobei jedoch zur Angabe eines detaillierten Formats das PRINT USING-Kommando zusammen mit einer geeigneten Formatschablone eingesetzt werden kann. Eine Formatschablone besteht aus einem Stringausdruck einschließlich Sonderzeichen, von denen jedes einen bestimmten Formattyp darstellt.

Diese als "Formatfeldmarkierungen" bezeichneten Zeichen sind in der Beschreibung des Schlüsselwortes PRINT USING zu Beginn dieses Handbuches enthalten.

Durch Einsatz der folgenden Beispiele wird ihr Einsatz klarer:

Eingangs werden die für das Drucken der Stränge zur Verfügung stehenden Formate beschrieben:

"\ \" veranlaßt den Druck der Zahl von Zeichen eines gegebenen String entsprechend der Anzahl der Zwischenräume + innerhalb der Schablone.

```
PRINT USING "\   \";"test string"
test s
```

"!" kann eingesetzt werden, um das erste Stringzeichen zu drucken.

```
PRINT USING"!"; "test string"
t
```

Hierbei ist jedoch das gebräuchlichste Stringzeichen das "&". Dieses Zeichen kann verwendet werden, um erforderlichenfalls Strangzeichen von BASIC zu überschreiben. Normalerweise veranlaßt BASIC den Start des Druckvorgangs jeder Folge auf einer neuen Zeile, falls die Folge zu lang ist, um auf der laufenden Zeile untergebracht werden zu können. Das Kommando PRINT USING "&"; kann verwendet werden, um diese Funktion aufzuheben.

(Einsatz von BORDER Ø, so daß Sie die Kanten des Bildschirms sehen können.)

```
MODE 1: LOCATE 39,1:PRINT "too long"
                                              Zeile 1
too long                                      Zeile 2

MODE 1: LOCATE 39,1:PRINT USING "&";"
too long"
                                   to         Zeile 1
                                              Zeile 2
o long
```

Für das Drucken von Zahlen steht eine große Anzahl von Schablonen zur Verfügung. Die einfachste ist wahrscheinlich die Schablone PRINT USING " # ", wobei jeweils ein Digit für jedes " " gedruckt wird, das innerhalb der Schablone erscheint.

```
PRINT USING "          ;123
   123
```

Die Position des Dezimalpunktes kann durch Einsatz von "." eingeschlossen werden.

```
PRINT USING "     .     ";12.45
12.45ØØØ
```

Die Digits vor dem Dezimalpunkt können als Zweige zusammengefaßt und jeweils durch Kommas abgetrennt werden, wenn "," vor dem Dezimalpunkt in der Schablone enthalten ist.

```
PRINT USING "      ,.      ";123456.78
 123,456.78ØØ
```

In das Format können darüber hinaus Dollar- und Pfundzeichen eingefügt werden, d.h. ein Währungszeichen, das stets direkt vor dem ersten Digit einer Nummer gedruckt wird, selbst dann, wenn das gesamte Format hierdurch nicht ausgefüllt wird. Dies erfolgt durch Einsatz von "$$" und "££" innerhalb der Schablone.

```
PRINT USING "$$    ";7
    $7

PRINT USING "$$    ";351
$351

PRINT USING "£ £     ,.   ",1234.567
£1,234.57
```

Beachten Sie die Abrundung des Ergebnisses.

Der Raum vor dem Ergebnis kann mit aufeinanderfolgenden Sternchen durch Einsatz von "**" innerhalb der Schablone ausgefüllt werden.

```
PRINT USING "**     .   "; 12.22
****12.2
```

Diese Eingabe kann mit den Währungssymbolen kombiniert werden (wobei lediglich ein Währungszeichen eingesetzt wird), d.h. "**$...usw." bzw. "**£...usw."

Ein "+" zu Beginn der Schablone gibt an, daß stets das Vorzeichen des Ergebnisses vor dem ersten Digit gedruckt wird. Ein "+" am Ende der Schablone druckt ein folgendes Zeichen.

Ein "-" kann nur am Ende des Formats stehen und gibt an, daß ein abschließendes Minuszeichen zu drucken ist, wenn die Zahl negativ ist.

```
PRINT USING "+    ";12
+12

PRINT USING "+    ";-12
-12

PRINT USING "  +";12
12+

PRINT USING "  -";-12
12-

PRINT USING "  -";12
12
```

Eine Formatschablone " " kann verwendet werden, um eine Zahl in exponentieller Form zu drucken.

```
PRINT USING "   .        ";123.45
12.35E+01
```

Bei Verwendung von Druckformaten für Zahlen ist festzustellen, daß, falls eine Zahl für die angegebene Schablone zu lang ist, ein %Zeichen-Symbol vor dem Ergebnis gedruckt wird, um anzuzeigen, daß dieser Fall eintrat; das Ergebnis wird hierbei **nicht** auf die angegebene Schablone gekürzt.

```
print using "    "; 123456
%123456
```

Funktion der sogenannten Text-"Fenster" (window)

Die BASIC-Beschreibung des CPC664-Systems bietet ein verständliches Verfahren zur Aufstellung von maximal acht Text-Fenstern. Sämtliche Bildschirm-Steuerkommandos können anschließend auf irgendeines dieser Fenster geleitet werden.

Die Steuerung, die eingesetzt wird, um ein Fenster anzusteuern, ist einfach genug: FENSTER (WINDOW). Dieser Steuerung folgen 5 Werte. Der erste ist freibleibend und wird eingesetzt, um anzugeben, welches Fenster definiert werden soll - falls keine Definition erfolgt, wird Fenster 0 angenommen, wobei alle normalen BASIC-Bestätigungen und Nachrichten (z.B. Ready) in Fenster 0 erzeugt werden. Das Symbol (#) vor dieser Nummer dient

zur Angabe, daß es sich um ein bestimmtes Ausgabegerät handelt. Die folgenden vier Nummern bezeichnen die linke, rechte, obere und untere Grenze des Fensters. Diese Werte bezeichnen Spalten- und Reihennummern, so daß sich diese im Bereich von 1 bis 80 für links/rechts und 1 bis 25 für oben/unten bewegen müssen.

Das folgende Beispiel definiert WINDOW (stream) Nummer 4 für den Start in Spalte 7 (links) mit Weiterführung zu Spalte 31 (rechts), mit Beginn in Reihe 6 (oben), mit Weiterführung zu Linie 18 (unten).
Stellen Sie den Computer zurück und geben Sie anschließend ein:

```
WINDOW #4,7,31,6,18
```

Hierbei werden Sie feststellen, daß nach dieser Steuerung kein Ereignis folgt, geben Sie trotzdem weiterhin ein:

```
INK 3,9
PAPER #4,3
CLS #4
```

Hierdurch wird ein großes grünes Rechteck auf dem Bildschirm bewirkt, wobei es sich um Fenster Nummer 4 handelt. Weiter oben wurde bereits darauf hingewiesen, daß PAPER und CLS in Verbindung mit irgendeinem der acht Fenster durch Einschluß eines stream-Symbols eingesetzt werden können; ein Auslassen führt die Steuerung zum Betrieb eines Fensters 0 - das Standardfenster.

Jedes der folgenden Kommandos kann mit einem stream-Symbol arbeiten, um das Fenster zu kennzeichnen, in dem das Kommando auszuführen ist.

```
CLS                    POS
COPYCHR$               PRINT
INPUT                  TAG
LINE INPUT             TAGOFF
LIST                   VPOS
LOCATE                 WINDOW
PAPER                  WRITE
PEN
```

Das neue grüne Fenster, das Sie auf dem Bildschirm erscheinen ließen, hat bestimmte Anteile des vorhergehenden Textes abgedunkelt (auf Fenster-Nummer 0 geschrieben).

Anschließend kann Text zu jedem Fenster übertragen werden und zwar durch Einschluß eines stream-Symbols innerhalb eines PRINT-Kommandos:

```
PRINT #4, "hello there"
```

Diese Wörter erscheinen am oberen Teil des grünen Rechtecks nicht auf der folgenden Zeile, die dann belegt worden wäre, wenn...

```
PRINT "hello There"
```

...verwendet worden wäre.

Beim Einschreiben in die frühere Steuerung haben Sie festgestellt, daß Teile des grünen Fensters durch den Text überschrieben wurden.

Falls Sie alle normalen BASIC-Befehle in Fenster 4 erscheinen lassen wollen, dann kann eine Umschaltung mit dem Standardfenster (0) durch Einsatz des Kommandos WINDOW SWAP erfolgen:

```
WINDOW SWAP Ø,4
```

Das "Ready", das diesem Kommando folgt, wird auf dem grünen Fenster erscheinen. Der Cursor wird anschließend direkt daneben-gesetzt. Versuchen Sie nunmehr folgende Eingabe:

```
PRINT #4, "hello there"
```

...und die Worte "hello there" erscheinen direkt neben der WINDOW-SWAP-Steuerung im früheren Fenster Ø, das nunmehr als Fenster 4 zu bezeichnen ist. Hieraus kann ebenfalls entnommen werden, daß die laufende Druckstellung in jedem Fenster gespeichert wird, so daß selbst nach einem WINDOW SWAP ein Textausdruck erfolgt und zwar innerhalb des neuen Stromes 4 im Gegensatz zu einem Start oben. Versuchen Sie folgende Eingabe:

```
LOCATE 4,20,1
PRINT "Das ist window Ø"
PRINT #4,"Das ist window 4"
```

Die Mitteilung "window Ø" erscheint auf der Zeile nach PRINT, während die Nachricht "Fenster 4" im mittleren Bereich der oberen Zeile des ganzen Schirms erscheint.

Bevor ein WINDOW-Kommando ausgegeben wurde, bedecken alle acht Fenster den gesamten Bildschirm. Dies gilt auch nach einem MODE-Kommando - so daß nach Einsatz der Fenster der Cursor in

einem sehr kleinen Fenster verbleibt; geben Sie nunmehr in MODE 1 wie folgt ein:

```
MODE 1
WINDOW 20,21,7,18
MO
DE
 1
```

Kümmern Sie sich nicht um die Aufteilung des Ausdruckes "MODE", er wird weiterhin arbeiten und vergessen Sie hierbei nicht einen Zwischenraum zwischen MODE und 1.

Nachdem Sie nunmehr einige Kenntnis über die Funktionsweise der Fenster besitzen - versuchen Sie die Eingabe des folgenden kurzen Programms:

```
10 MODE 0
20 FOR n=0 TO 7
30 WINDOW n,n+1,n+6,n+1,n+6
40 PAPER n,n+4
50 CLS n
60 FOR c=1 TO 200:NEXT c
70 NEXT n
```

Hierdurch werden acht sich überlappende Fenster gesetzt, wobei jedes eine andere Hintergrundfarbe aufweist.
Nach Durchlauf des Programms und Erscheinen von "Ready" drücken Sie (ENTER) einige Male, um festzustellen, wie das Rollen des Fensters 0 die Farbblöcke auf dem Bildschirm beeinflußt. Obwohl jedoch diese farbigen Blöcke rollen, bewegen sich die Standorte der anderen Fenster nicht. Versuchen Sie folgende Eingabe:

```
CLS #4
```

...und Sie werden feststellen, daß sich das 4. Fenster noch in der gleichen Position befindet - der neue farbige Block hat hierbei die benachbarten abgedunkelt. Beobachten Sie interessenhalber folgende Unterschiede bei Eingabe von:

```
LIST
LIST#4
LIST#3
```

Ein weiteres Merkmal des WINDOW-Kommandos - durch das abschliessende Programm in diesem Abschnitt demonstriert - besteht darin, daß es ohne Bedeutung bleibt, wenn Sie die Abmessungen des

linken und rechten Fensters in umgekehrter Reihenfolge angeben. Hieraus folgt, daß in den Fällen, in denen der Wert des ersten Parameters größer ist als der zweite, BASIC automatisch die Abmessungen in die entsprechende Reihenfolge bringt. Gleiches bezieht sich auf die oberen und unteren Abmessungen der Fenster.

```
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
```

Falls ich unterbrechen könnte...

Falls Sie es noch nicht bemerkt haben sollten, eine der wichtigsten Neuheiten des CPC664-Systems ist die Fähigkeit, Unterbrechungen in BASIC zu behandeln, d.h. daß Schneider BASIC in der Lage ist, innerhalb eines Programms mehrere unterschiedliche Handhabungen gleichzeitig auszuführen. Solch eine Fähigkeit wird manchmal "Multitasking" genannt und kann mit den neuen Befehlen AFTER und EVERY angesprochen werden.

Diese Fähigkeit zeigt sich auch deutlich in der Behandlung von Tönen durch die Warteschlangen und Rendevouztechnik.

Alle zeitabhängigen Ereignisse werden von der zentralen Uhr gesteuert, ein quarzgesteuerter Zeitgeber im Rechner. Sie überwacht den zeitlichen Ablauf und stimmt alle Handlungen im Rechner aufeinander ab, wie z.B. das Aufbauen des Bildes auf dem Bildschirm oder das Taktgeben im Prozessor. Wann immer eine Funktion der Hardware zeitabhängig ist, wird sie von der zentralen Uhr gesteuert.

Die Software ist mit den Kommandos AFTER und EVERY abgedeckt. Wie bei Schneider BASIC üblich, sind diese Kommandos so benutzerfreundlich, daß sie auch das tun, was der Name verspricht.

So wird z.B. nach (AFTER) der Zeit, die Sie in der Kommandozeile vorgegeben haben, in das gewünschte Unterprogramm gesprungen und dieses ausgeführt.

Der CPC664 besitzt eine Echtzeituhr. Das AFTER-Kommando ermöglicht, daß Unterprogramme erst nach einer geplanten Zeit ausgeführt werden können. Vier unabhängige Zeitgeber stehen zur Verfügung, die je ein eigenes Unterprogramm ansprechen können.

Sobald die vorgegebene Zeit abgelaufen ist, wird das Unterprogramm automatisch aufgerufen, gerade so, als ob ein GOSUB-Kommando an der Stelle im Programm angegeben wäre, an der es sich im Ablauf gerade befindet. Am Ende des Unterprogramms, das mit einem normalen RETURN-Kommando abgeschlossen wird, kehrt das Programm zu der Stelle zurück, wo es vorher unterbrochen wurde.

Das EVERY-Kommando beauftragt ein BASIC-Programm, Unterprogramme wiederholt zu regelmäßigen Zeitpunkten aufzurufen. Auch hier stehen vier Zeitgeber zur Verfügung, denen jeweils ein Unterprogramm zugeordnet werden kann.

Die Zeitgeber haben verschiedene Unterbrechnungs-Prioritäten. Der Zeitgeber 3 hat die höchste Priorität, der Zeitgeber 0 die niedrigste (siehe Kapitel "Wenn Sie Zeit haben...").

```
10 MODE 1:n=14:x=RND*400
20 AFTER x GOSUB 80
30 EVERY 25,2 GOSUB 160
40 EVERY 10,1 GOSUB 170
50 PRINT "Testen Sie Ihre Reflexe"
60 PRINT "Druecken Sie die Leertaste.";
70 IF flag=1 THEN END ELSE 70
80 z=REMAIN(2)
90 IF INKEY(47)=-1 THEN 110
100 SOUND 1,900:PRINT "Betrueger!":GOTO 150
110 SOUND 129,20:PRINT "JETZT":t-TIME
120 IF INKEY(47)=-1 THEN 120
130 PRINT "Sie brauchten";
140 PRINT (TIME-t)/300;"Sekunden"
150 CLEAR INPUT:flag=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-Kommandos können zu jeder Zeit angegeben werden, wobei die zugehörigen Unterprogramme und Zeitgeber jeweils neu gesetzt werden. Die Zeitgeber sind dieselben, die auch im AFTER- und EVERY-Kommando verwendet werden, so daß ein AFTER-Kommando ein vorheriges EVERY-Kommando für einen bestimmten Zeitgeber überschreibt und umgekehrt.

Die DI und EI-Kommandos sperren und geben Zeitgeberunterbrechungen frei, während die Steuerungen zwischen Letzteren ausgeführt werden. Hierdurch wird bewirkt, daß eine Unterbrechung höherer Priorität verzögert wird, während eine Unterbrechung niedriger Priorität bearbeitet wird. Die REMAIN-Funktion sperrt und führt die verbleibende Zählung für einen der vier Verzögerungszeitwerke aus.

Einsatz der Daten

Innerhalb eines Programms, das in der Startphase stets den gleichen Satz an Informationen fordert, wäre es zweckmäßiger, einen Weg zu finden, alle Werte eingeben zu können, ohne daß der Benutzer diese jeweils neu eingeben muß. Diese Möglichkeit wird durch die READ- und DATA-Kommandos gegeben. Der Ausdruck READ ist dem Ausdruck INPUT insofern sehr ähnlich, daß er verwendet werden kann, Variablen Werte zuzuordnen. Er unterscheidet sich jedoch in der Tatsache, daß Werte von DATA-Anweisungen gelesen werden, im Gegensatz zu einer Eingabe von der Tastatur aus. Die folgenden beiden Beispiele zeigen dies auf:

```
10 INPUT "Gib 3 Zahlen durch Komma getrennt ein.";a,b,c
20 PRINT "Die Zahlen waren :";a;"und";b;"und";c
run
```

```
10 READ a,b,c
20 PRINT "Die Zahlen waren :";a;"und";b;"und";c
30 READ DATA 12,14,21
run
```

In der gleichen Weise, in der verschiedene Sachgruppen innerhalb einer INPUT-Anweisung durch Kommas getrennt werden, verhält es sich mit Sachgruppen innerhalb einer DATA-Anweisung.

In Ergänzung der numerischen Werte können darüber hinaus konstante Zeichenketten in DATA-Anweisungen geführt werden:

```
10 DIM a$(6)
20 FOR i = 0 TO 6
30 READ a$(i)
40 NEXT
50 FOR i = 0 TO 6
60 PRINT a$(i);" ";
70 NEXT
80 DATA Montag,Dienstag,Mittwoch,Donnerstag,Freitag,Samstag,
Sonntag
run
```

Sie können hierbei feststellen, daß abgesehen davon, daß DATA Zeichenketten enthält, die Zeichenketten nicht durch Anführungszeichen eingeschlossen werden. Der Einsatz der Anführungszeichen in DATA-Anweisungen zur Trennung (separater) Zeichenketten ist freigestellt, in der gleichen Weise als wenn eine Zeichenkette als Beantwortung einer INPUT-Anweisung aufgestellt wird. Innerhalb eines Falles erweist es sich jedoch als zweckmäßig, Anführungszeichen einzusetzen und zwar dann, wenn die Zeichenkette DATA selbst Kommas enthält. Werden Zeichenketten nicht durch Anführungszeichen unter diesen Bedingungen begrenzt, so arbeitet die READ-Anweisung mit Kommas zur Begrenzung der Zeichenketten innerhalb der DATA-Anweisung.

```
10 READ a$
20 WHILE a$ <> "*"
30 PRINT a$
40 READ a$
50 WEND
60 DATA Der,Computer,liest,die,Daten,einzeln
70 DATA "Diese Daten werden als ganzes eingelesen."
80 DATA *
run
```

Die Zeichenkette in Zeile 60 enthält Kommas, so daß jeder Teil getrennt gelesen (READ) und gedruckt wird. Die Zeichenkette in Zeile 70 wird jedoch durch Anführungszeichen begrenzt und, wie beabsichtigt, als Ganzes gedruckt.

Das obige Beispiel zeigt die Tatsache, daß Daten über eine Reihe von Zeilen verteilt werden können. READ setzt die Zeilen in Nummernfolgen um (60,70,80 usw.). Eine weitere Tatsache, die

nicht gleich augenscheinlich wird, liegt darin, daß DATA-Anweisungen beliebig innerhalb eines Programms eingesetzt werden können; vor oder nach der READ-Anweisung, die die Information aufnimmt.

Enthält ein Programm mehr als eine READ-Anweisung, dann setzt die zweite READ-Anweisung an dem Punkt fort, an dem die erste READ-Anweisung unterbrochen wird:

```
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 sind i.O." ELSE
PRINT "Hier ist ein Fehler in den Daten !!"
run
```

Versuchen Sie eine Korrektur der Zeile 10, so daß eine der ersten 5 Zahlen falsch ist und lassen Sie anschließend das Programm erneut anlaufen. Dieses Verfahren der Hinzunahme eines weiteren Wertes am Ende der DATA-Anweisungen, die die Summe aller anderen Werte ausdrückt, ist ein gutes Verfahren zur Fehlerermittlung innerhalb von DATA, insbesondere dann, wenn es sich um eine große Anzahl von DATA-Zeilen handelt - in diesem Falle bekannt als "Prüfsumme".

Fordert ein Programm gemischte Daten (Zeichenketten und Nummern), so ist es zulässig, Zeichenkette und Nummernelemente in READ- und DATA-Anweisungen zu kombinieren, so lange die Teile einwandfrei gelesen werden. Enthält beispielsweise DATA Folgen von zwei Nummern, mit anschließender Zeichenkette - dann wäre es nur von Nutzen, eine READ-Anweisung einzusetzen, die durch zwei numerische Variablen und anschließend eine Zeichenkettenvariable gefolgt würde:

```
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,karl,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
```

Hierzu alternativ kann gefordert werden, die verschiedenen Datentypen zu trennen:

```
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,karl,erich,peter,alfons
100 FOR i=1 TO 5
110 PRINT s$(i),":";a(i)*b(i)
120 NEXT
```

Wird die FOR-Schleife in Zeile 20 geändert auf:

```
2Ø FOR 1=1 TO 4
```

...dann liefern die ersten beiden Versuche, die Zeichenketten in Zeile 60 zu lesen, "9", anschließend "1". Bei diesen Werten handelt es sich selbstverständlich um gültige Zeichenketten, jedoch war das Ergebnis nicht in dieser Form geplant. Eine Möglichkeit, durch die das Programm einwandfrei arbeiten könnte, würde in der Einbeziehung folgender Steuerungen bestehen:

```
15 RESTORE 8Ø
45 RESTORE 9Ø
```

Durch die RESTORE-Anweisung wird der DATA-Lesezeiger auf die angegebene Zeile geführt und kann demzufolge innerhalb einer bedingten Anweisung dazu dienen, einen bestimmten Datenblock abhängig von einem bestimmten Kriterium zu lesen. So beispielsweise in einem auf mehreren Ebenen durchgeführten Spiel, das eine Anzahl verschiedener Bilder umfaßt, könnte DATA für jeden Schirm entsprechend einer bestimmten Variablen aufgenommen werden - beispielsweise "LEVEL". Im folgenden wird nur ein Beispiel eines solchen Programms dargestellt:

```
1000 REM Unterprogramm zum Zeichnen des Bildschirmes
1010 IF level=1 THEN RESTORE 2010
1020 IF level=2 THEN RESTORE 2510
1030 IF level=3 THEN RESTORE 3010
1040 FOR y=1 TO 25
1050 FOR x=1 TO 40
1060 READ char
1070 LOCATE x,y: PRINT CHR$(char);
1080 NEXT x,y
2000 REM DATA fuer Bildschirm 1
2010 DATA 200,190,244,244,218.....etc
2500 REM DATA fuer Bildschirm 2
2510 DATA 100,103,245,243,251.....etc
3000 REM DATA fuer Bildschirm 3
3010 DATA 190,191,192,193,194.....etc
```

Ein weiteres Einsatzbeispiel von DATA, READ und RESTORE kann in einem Programm liegen, das eine Melodie wiedergibt. Hierbei können Tonperiodenwerte aus DATA-Anweisungen gelesen werden und die Funktion RESTORE eingesetzt werden, um einen Abschnitt zu wiederholen, in dem der Zeiger auf den Anfang eines bestimmten Teiles der DATEN der Musik geführt wird.

```
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
```

Musik-Einführung

Von allen Merkmalen des CPC664-System sind vielleicht die Ton- und Hüllkurven-Steuerungen die auf den ersten Blick entmutigensten - jedoch braucht dies nicht der Fall zu sein. Mit etwas Praxis sollten Sie in der Lage sein, eine Vielzahl verschiedener Geräusche zu erzeugen, Sie sollten sogar in der Lage sein, eine komplette Melodie mit Harmonien zu spielen.

Lassen Sie uns zuerst die 4 Teile einer SOUND-Steuerung untersuchen: Hierbei handelt es sich um Kanalnummer, Tonperiode, Dauer einer Note und Lautstärke. Sie werden sich hierbei fragen, welchen Bereich jede Nummer einnimmt.

Wir möchten zuerst noch den ersten Teil (Kanalnummer) für eine bestimmte Zeit auslassen, da dieser Bereich recht kompliziert ist. Der zweite Teil (Tonperiode) kann jede ganze Zahl zwischen 0 und 4095 einnehmen, wobei jedoch nur einige dieser Werte tatsächlich erkennbare Töne vom Musikalischen her gesehen, liefern. Diese sind in Teil 5 des Abschnitts "Wenn Sie Zeit haben..." aufgeführt. So ermöglicht beispielsweise die Nummer 478 das mittlere C, die 506 die Note B unter dem mittleren C und die Werte 479-505 führen zu einem Ton, jedoch nicht zu einem, der der Klaviertastatur entspricht. Ist die Tonperiode Null, so wird kein Ton erzeugt - diese Erkenntnis ist dann von Nutzen, wenn später der Begriff "Rauschen" erläutert wird.

Der dritte Teil der Tonsteuerung bestimmt die Dauer der Note in Einheiten einer hunderstel Sekunde. Der Wert kann sich im allgemeinen im Bereich zwischen 1 und 32767 befinden. Ist der Wert jedoch Null, so wird die Länge der Note durch die eingesetzte "Hüllkurve" (nähere Erklärung im weiteren Verlauf) bestimmt. Ist der Wert negativ, so wird angegeben, wie oft die Hüllkurve eingesetzt werden soll; somit würde -3 bedeuten, "die Hüllkurve dreimal zu wiederholen" (Erläuterung im weiteren Teil).

Der vierte Teil der Steuerung betrifft die Lautstärke. Diese kann zwischen 0 und 15 schwanken (wird sie nicht angegeben, wird 12 angenommen). Mit dem bisher einfachen Geräusch blieb das Volumen für die gesamte Zeit der Note konstant. Wird eine "Lautstärken-Hüllkurve" zur Veränderung eingesetzt, so wird der Volumenteil der SOUND-Steuerung als Startwert der Note angesetzt.

Gehen wir nunmehr zu dem Teil der Steuerung über, der die Kanalnummer betrifft. Wahrscheinlich werden Sie wissen, daß es sich hierbei um eine "bit-signifikante" Nummer handelt, wes-

halb Sie etwas mehr über Binärzahlen wissen sollten, um diesen Abschnitt voll zu verstehen (siehe Teil 1 dieses Kapitels).

Ein Ton kann auf einem von drei möglichen Kanälen entstehen, wenn der Computer mit einem Stereoverstärker verbunden ist; in diesem Falle befindet sich ein Kanal links, ein anderer Kanal rechts und der dritte praktisch im mittleren Bereich. Um auszuwählen, auf welchem Kanal bzw. Kanälen eine Note hörbar sein soll, werden folgende Nummern eingesetzt:

1 Kanal A
2 Kanal B
4 Kanal C

Um für den Klang mehr als einen Kanal zu benutzen, sind die Nummern der gewünschten Kanäle hinzuzufügen. Beispielsweise Kanal A und C arbeiten mit 1+4=5.

```
TON 5,284
```

Sie werden erstaunt sein, warum Kanal C der Nummer 4 und nicht 3 zugeordnet ist, wie Sie annehmen könnten. Der Grund liegt darin, daß jeder dieser Nummern ein Wert von zwei zugeordnet ist (1=2 0,2=2 1, 4=2 2), so daß in der Kombination eine Binärzahl entsteht. Wenn Sie sich eine drei Digit-Binärzahl vorstellen, so kann jedes der drei Digits entweder 0 bzw. 1 sein, wobei dieses Schema für die Kanalnummer angewendet wird, um anzuzeigen, ob der entsprechende Kanal zu- bzw. abgeschaltet wird. Ausgehend von dem obigen Beispiel:

5 in Dezimal entspricht 1*4+Ø*2+1*1 bzw. 101 in binärer Schreibweise. Hieraus folgt, daß bei Benennung jeder Spalte dieser Binärzahlen mit C, B und A folgende Form entsteht:

```
C B A
1 0 1
```

Mit anderen Worten, ist Kanal C eingeschaltet, Kanal B ausgeschaltet und Kanal A eingeschaltet. Wurde der Ton auf den Kanälen A und B übertragen, so werden sich hieraus ergeben:

```
C B A
0 1 1
```

wobei die Binärzahl 011 die gleiche ist wie 0*4+1*2+1*1 = 3.

Somit ergibt sich folgende SOUND-Steuerung:

```
SOUND 3,248
```

Hierbei handelt es sich selbstverständlich um den gleichen Wert als den, den Sie erhalten würden, wenn Sie die Nummern der Kanäle addieren würden (A=1, B=2, C=4). Um hierfür A und B einzusetzen, ergibt sich die Kanalnummer durch 1+2=3.

Auch wenn Sie nicht gleich verstehen werden, bleiben Sie unbesorgt. Solange Sie erkennen werden, daß eine Kombination von Kanälen dadurch gewählt werden kann, daß die Zahlen für jeden der Kanäle addiert werden können, reicht es vorerst für das Verständnis aus.

Leider existieren mehr Werte als innerhalb einer Kanalnummer eingesetzt werden können. Die Werte 8,16 und 32 werden mit dem Ziel eingesetzt, anzugeben, daß der Ton sich mit einem anderen Kanal mischen sollte (A, B, C). Sie werden sich über den Ausdruck "mischen" bzw. "Rendevouz" wundern. Bisher wurden die Töne, die wir erzeugt haben, zu direkten angegebenen Kanälen übertragen. Unternehmen Sie einen Versuch:

```
SOUND 1,248,2000
SOUND 1,90,200
```

Wenn Sie nicht gerade ein sehr langsamer Schreiber sind, werden Sie festgestellt haben, daß Sie in der Lage waren, in Sekunden diese Steuerungen einzugeben, bevor die erste abgeschlossen war. Dieser Umstand ist dadurch zu erklären, daß das Tonsystem in der Lage ist, bis zu 5 Tonsteuerungen in jedem Kanal in einer "Warteschlange" zu halten. Falls wir einen Ton über Kanal A übertragen wollen, anschließend zwei Töne gleichzeitig über A und B, benötigen wir ein Mittel, dem Computer mitzuteilen, daß der Ton auf B erst dann auszulösen ist, wenn die zweite Note auf A übertragungsbereit ist, d.h. wir müssen veranlassen können, daß ein Kanal auf den anderen wartet. Dieser Vorgang wurde als Rendevouz bezeichnet, wobei hierfür zwei Verfahren bestehen.

```
SOUND 1,200,1000
SOUND 3,90,200
```

Hiernach wird die zweite Note nach A und B geleitet, wobei kein Start erfolgen kann, bis die Note auf A nicht abgeklungen ist. Die Begrenzung durch Einsatz dieses Verfahrens (eine Notenkombination warten zu lassen, bis alle Kanäle, auf der sie

übertragen werden sollen, frei sind) besteht darin, daß der gleiche Ton zu jedem der Kanäle geleitet wird (in diesem Falle 90, 200 mit Richtung auf A und B). Hierzu besteht ein weiteres alternatives Verfahren:

```
SOUND 1,200,2000
SOUND 1+16,90,200
SOUND 2+8,140,400
```

Hier wird der zweite Ton auf A mit dem Ton auf B abgestimmt (ferner der Ton auf B mit der Note auf A). Der Vorteil hierbei liegt auf der Hand: obwohl die zweite Note auf A von der Note auf B abwich, waren beide so miteinander verbunden, daß keine beginnen konnte, bevor beide Kanäle nicht frei waren - in diesem Falle handelt es sich um ein Rendevouz. Noch einmal zu erwähnen ist, daß es sich hierbei um bit-signifikante Werte handelt.

$8=2\uparrow 3, 16=2\uparrow 4$, und $32=2\uparrow 5$

...somit kann die Kanalnummer als Binärzahl aufgefaßt werden, und zwar jeweils aufgeteilt wie folgt:

Rendezvous C	Rendezvous B	Rendezvous A	Spiel auf C	Spiel auf B	Spiel auf A
Zu 32	Zu 16	Zu 8	Zu 4	Zu 2	Zu 1

Somit wird einer Noteauf C, mit Abstimmung auf 4 folgende Zuordnung zugewiesen:

0	0	1	1	0	0

Hierbei handelt es sich um die Binärzahl 1100, die dem Ausdruck 8 + 4 = 12 entspricht.

Demzufolge teilt eine Kanalnummer 12 dem Computer mit, eine Note auf Kanal C zu übertragen und auf eine Note zu warten, die in dieser Form auf Kanal A abgestimmt wird.

Wird 64 ($2\uparrow 6$) der Kanalnummer hinzugefügt, so wird hierdurch angezeigt, daß die Note zu halten ist. Dies bedeutet somit, daß die Note erst dann erklingt, wenn die Steuerung RELEASE eingesetzt wird.

Wird schließlich 128 (2↑7) der Nummer hinzugefügt, dann wird die Warteschlange des angegebenen Kanals freigegeben. Lösen Sie demzufolge einen Ton aus, der auf einem bestimmten Kanal zu lange andauert, so besteht ein schneller Weg der Unterbrechung darin, den Kanal freizugeben.

```
SOUND 1,248,30000        (der Ton wird für 5 Minuten
                          gehalten)
SOUND 1+128,0            (hierdurch wird kurz unter-
                          brochen)
```

Innerhalb der direkten Betriebsart besteht ein noch schnellerer Weg der Unterbrechung langer Töne darin, die Taste DEL zu Beginn einer Zeile zu drücken; der kurze Warnton bezieht sich auf alle Tonkanäle.

Nachdem wir nunmehr in der Lage sind, ohne weiteres einen Ton auf die von uns zu wählenden drei Kanäle zu übertragen (erforderlichenfalls mit der Rendezvous-Funktion) wäre es schön, einen Ton zu erzeugen, der etwas über die Qualität dieses unmelodiösen Pieptons hinausgeht, den die einfache SOUND-Steuerung liefert. Der Weg hierfür liegt in der Verwendung einer Hüllkurve - eines Verlaufes, der definiert, wie der Ton lauter und leiser wird, während der kurzen Zeit, während der er andauert. Ein durch ein Instrument erzeugter Ton zeigt ein erstes Ansteigen, wenn die Lautstärke sehr schnell zunimmt. Anschließend fällt die Lautstärke des Tones auf einen niedrigeren Pegel, der eine gewisse Zeit beibehalten wird, nachdem der Ton auf Null abklingt. Auf diese Weise kann eine Hüllkurve für die Töne erzeugt werden, die über die SOUND-Steuerung ausgelöst wird. Hierfür dient die zugehörige ENV-Steuerung. Betrachten wir zuerst ein einfaches Beispiel:

```
ENV 1,5,3,4,5,-3,8
SOUND  1,284,0,0,1
```

Die ENV-Steuerung hat der entsprechenden SOUND-Steuerung vorauszugehen. Um diese Hüllkurve innerhalb einer SOUND-Steuerung einzusetzen, wird ihre Nummer als 5. Teil in die SOUND-Steuerung übernommen, in diesem Falle handelt es sich bei der Hüllkurve um die Nummer 1. Die erste Nummer innerhalb einer ENV-Steuerung ist die Nummer der definierenden Hüllkurve. Die ENV-Anweisung enthält eine Information über die Dauer der Note sowie über deren Lautstärke; demzufolge werden die Anteile für Dauer und Lautstärke der SOUND-Steuerung auf Null gesetzt.

Die obengenannte Hüllkurve bewirkt ein Ansteigen des Tones in 5 Schritten, wobei jeder Schritt das Volumen um 3 erhöht und jeder Schritt 4 Hundertstel einer Sekunde andauert. Anschließend wird die Lautstärke in 5 Schritten reduziert, wobei jeder Schritt eine Verminderung der Lautstärke um -3 bewirkt, und jeder Schritt 8 Hundertstel einer Sekunde lang ist. Mit anderen Worten, bezeichnet die erste Nummer die entsprechende Hüllkurve, gefolgt von zwei Gruppen von drei Nummern, wobei in jeder dieser Gruppen die erste Nummer anzeigt, um wieviele Schritte die Lautstärke zu erhöhen bzw. abzusenken ist. Die zweite Nummer zeigt den Betrag der Lautstärke nach oben bzw. nach unten und zwar innerhalb jedes dieser Schritte, wobei die dritte Nummer die Länge jedes Volumenschrittes bestimmt.

Die Gesamtlänge der Zeit für jeden Abschnitt entspricht dem ersten Wert (Anzahl der Schritte) multipliziert mit dem dritten Wert (Pausenzeit). Der Gesamtanstieg bzw. Abfall der Lautstärke entspricht der Anzahl der Schritte, multipliziert mit der Schrittgröße. Die Gesamtlänge einer Hüllkurve mit mehr als einem Abschnitt entspricht der Summe der Längen jedes Abschnittes.

Selbstverständlich muß die Anfangslautstärke nicht immer den Wert 0 besitzen (wie durch die SOUND-Steuerung eingestellt). Innerhalb des obigen Beispiels wurde ein Tone rzeugt, der zuerst an- und dann anschließend abschwoll. Im folgenden Beispiel erfolgt zuerst ein Abschwellen, anschließend ein Anschwellen:

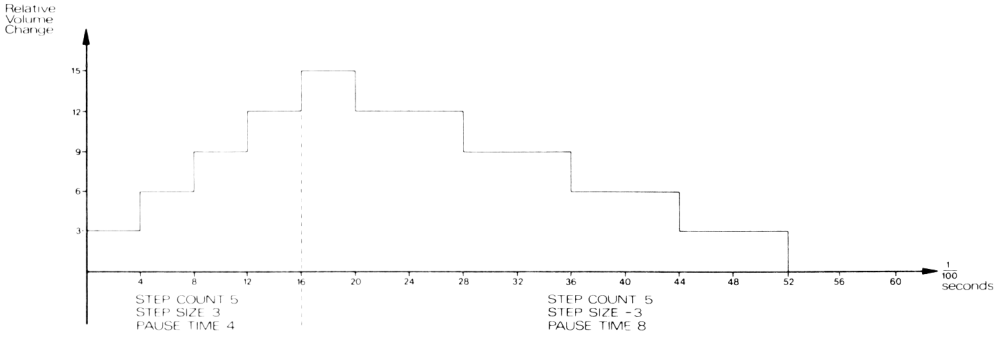
```
ENV 2,5,-2,1,20,0,1,10,1,1
SOUND 1,248,0,15,2
```

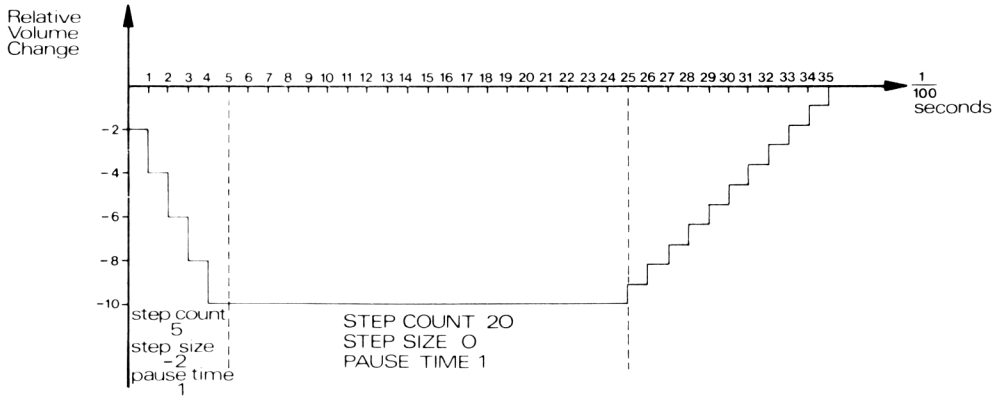
Diese Hüllkurve wird durch die Zahl 2 angegeben und besitzt drei Abschnitte. Im ersten die Lautstärke in 5 Schritten von -2 reduziert. D.h. es erfolgt ein Abschwellen von jeweils 2 bei jedem Schritt, wobei 5 Schritte zu durchlaufen sind. Die Länge jedes dieser Schritte beträgt eine hundertstel Sekunde. Das im zweiten Abschnitt umfaßt 20 Schritte, jedoch mit der 0-Lautstärke-Reduzierung bzw. Ansteigen (d.h. konstantes Volumen) pro Schritt. Auch hier beträgt die Länge jedes dieser Schritte ein Hundertstel einer Sekunde. Schließlich besitzt der dritte Abschnitt 10 Schritte, mit einem jeweiligen Anstieg der Lautstärke um 1, wobei wiederum jeder Schritt 1 Länge von einer hundertstel Sekunde besitzt.

Das SOUND-Kommando besitzt ein Startvolumen von 15, wobei nach dem ersten aBschnitt das Volumen auf 5 reduziert wird, dieses während 20 hundertstel Sekunden konstant gehalten und anschließend auf 15 im endgültigen Abschnitt der Hüllkurve erhöht wird.

Es mag sich mitunter als schwierig erweisen, die form dieser Hüllkurven darzustellen. Oftmals erweist es sich als hilfreich, diese auf papier aufzuzeichnen und die Werte hiervon für die ENV-Steuerung abzunehmen.

Die folgende Darstellung zeigt die Form der bisher definierten zwei Hüllkurven.

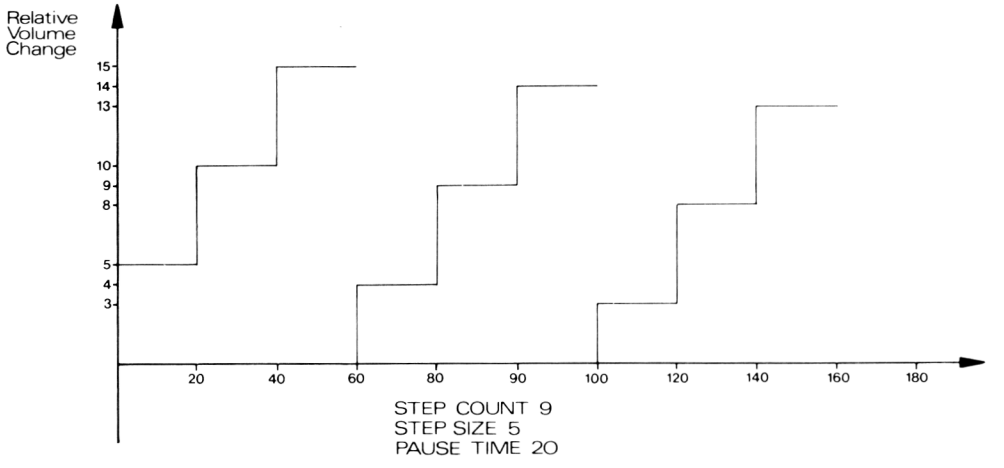




Die Maximalzahl der Abschnittein jeder Hüllkurve ist 5, wobei jeder Abschnitt 3 Werte besitzt, so daß die ENV-Steuerung bis zu 16 Teile umfassen kann (einschließlich des ersten, der definiert, welche der 15 Hüllkurven (1 bis 15) aufgestellt wird. Bewirken die aufwärts- bzw. abwärts gerichteten Schritte ein Überschreiten von 15 bzw. Unterschreiten von 0 des Volumens, so wird der Wert mitgeteilt; demzufolge gilt der Schritt über 15 gleich 0 und der Schritt unter 0 gleich 15:

```
ENV 3,9,5,20
SOUND 1,248,0,0,3
```

Diese einfache Hüllkurve liefert 9 Schritte, wobei jeder Schritt die Lautstärke um 5 erhöht und jeder Schritt während 20 Hundertstel einer Sekunde andauert. Demzufolge ergibt sich nach den ersten drei Schritten eine Lautstärke von 15, so daß der nächste Schritt eine Abrundung auf 4, auf 9 usw. bewirkt. Das Diagramm der folgenden Abbildung zeigt somit die Auswirkung:



Der Bereich der Werte für die Schrittanzahl liegt zwischen 0 und 127. Die Schrittgröße läßt sich zwischen -128 und +127 variieren (negative Werte vermindern die Lautstärke) und die Pausenzeit (d.h. die Zeit zwischen den Schritten) kann sich beliebig innerhalb des Bereiches zwischen 0 und 255 befinden.

Nachdem wir nunmehr in der Lage sind, eine Lautstärken-Hüllkurve bestimmter Formen zu erzeugen, wird es uns wahrscheinlich auch interessieren, einen Tonverlauf in der Art eines Vibrators zu erzeugen - bei dem der Ton um den mittleren Bereich der Note schwingt.

Der Weg hierzu entspricht stark dem der Definition der Lautstärken-Hüllkurven. Die Definition der Lautstärken-Hüllkurven erfolgt unter Einbeziehung der ENT-Steuerung. Beispielsweise:

```
ENT 1,5,1,1,5,-1,1
SOUND 1,248,10,15,,1
```

Die Übernahme einer Ton-Hüllkurve in die SOUND-Steuerung erfolgt durch Eingabe der Nummer der Ton-Hüllkurve als 6. Teil der SOUND-Steuerung. Auch in diesem Falle muß sich die ENT-Steuerung vor der SOUND-Steuerung befinden.

Das erste Beispiel von ENT gibt an, daß innerhalb der Ton-Hüllkurvennummer 1 fünf Schritte enthalten sind, wobei jeder Schritt die Tonperiode um 1 erhöht, und jeder Schritt die Länge von einer hundertstel Sekunde besitzt. Der zweite Abschnitt der Hüllkurve unterscheidet wiederum 5 Schritte, wobei diesmal jeder Schritt die Länge einer hundertstel Sekunde besitzt. Somit ergibt sich eine Gesamtlänge von 5 + 5 = 10. Sie werden sich erinnern, daß diese Länge in der SOUND-Steuerung angegeben war, da eine Ton-Hüllkurve nicht die Zeitlänge bestimmt, die eine Note andauert (desgleichen verhält es sich bei einer Lautstärken-Hüllkurve).

Ist die innerhalb einer SOUND-Steuerung eingesetzte Länge kürzer als die Länge der Ton-Hüllkurve, dann wird der letzte Teil der Ton-Hüllkurve unterdrückt. Ist dieser größer, dann wird der letzte Teil der Note mit konstantem Ton anhalten. Gleiches gilt dann, wenn eine Lautstärken-Hüllkurve eingesetzt wird, um die Notenlänge zu bestimmen.

(Sie werden das Fehlen jeder Zahl innerhalb der Lautstärken-Hüllkurve als Teil der SOUND-Steuerung vermissen. Aus diesem Grund haben wir keine Lautstärken-Hüllkurve für diesen Ton gebildet.)

Die Mehrzahl der Tonhüllkurven ist wahrscheinlich kürzer als die erwartete Länge der Note. Aus diesem Grunde kann eine Tonhüllkurve während der gesamten Zeit wiederholt werden, während der die Note andauert. Dies erfolgt durch Angabe einer negativen Hüllkurvennummer, das positive Äquivalent ist hierbei innerhalb der SOUND-Steuerung einzusetzen:

```
ENT -5,4,1,1,4,-1,1
SOUND 1,248,100,12,,5
```

Hierdurch entsteht für die Note ein Vibrato-Effekt. Bei der Festlegung der Ton-Hüllkurven ist es normalerweise anzuraten, diese symmetrisch um die erste Tonperiode variieren zu lassen, so daß bei Wiederholung der Note kein Abklingen von der Erstfrequenz erfolgt. Versuchen Sie diesen Vorgang:

```
ENT -6,3,1,1
SOUND 1,248,90,12,6
```

Sie werden feststellen, daß die Frequenz dieses Tones sehr stark abfällt, da in jedem Falle die Hüllkurve wiederholt wird und die Tonperiode um 3 erhöht wird, wobei dieser Vorgang 30 mal stattfindet (90/3). Diese Wirkung kann sich als sehr nützlich für das "Verändern" des Tones von Noten und Sirenen erweisen:

```
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
```

Die Anzahl der möglichen Ton-Hüllkurven beträgt 15 (im Bereich von 1 bis 15), mit einem negativen Wert, der angibt, daß die Hüllkurve zu wiederholen ist. Die Anzahl der Schritte (der erste Wert in jeder Gruppe von 3) kann sich beliebig zwischen 0 und 239 befinden. In der Lautstärken-Hüllkurve kann sich die Schrittgröße zwischen -128 und 127 ändern, wobei die Pausenzeit zwischen 0 und 255 liegen sollte. Auch hier kann eine ENT-Steuerung, wie eine ENV-Steuerung über maximal 5 Abschnitte (jeweils mit 3 Werten) verfügen.

Als letzter Teil, der einer SOUND-Steuerung beigemessen werden kann, ist ein siebenter Wert zu nennen, der den Rauschpegel angibt, der für den Ton gelten soll. Bei der Hinzunahme eines Rauschpegels zu einem Ton sollte daran gedacht werden, daß nur

ein Rauschkanal zur Verfügung steht, so daß jeder darauffolgende Rauschperiodenwert die jeweils vorhergehenden überdeckt.

Ein Rauschpegel kann einem Ton mit dem Ziel hinzugefügt werden, einen anderen Klang zu erzeugen bzw. kann dieser getrennt eingesetzt werden, und zwar durch Einstellung der Tonperiode (zweiter Teil) der SOUND-Steuerung auf 0, so daß nur der Rauschanteil vorhanden ist. Auf diese Weise entstehen Töne in der Art der Perkussion. Versuchen Sie diese Möglichkeit:

```
ENT -3,2,1,1,2,-1,1
ENV 9,15,1,1,15,-1,1
FOR a=1 TO 10: SOUND 1,4000,0,0,9,3,15:NEXT
```

Diese Folge könnte die Basis eines Rauschanteiles bilden. Sie werden hierbei feststellen, daß in diesem Falle Hüllkurven und Rauschanteil kombiniert werden.

Die Anteile für Dauer und Lautstärke der SOUND-Steuerung werden beide auf 0 gesetzt, und zwar entsprechend ihrer Festlegung durch die Lautstärken-Hüllkurve.

Da wir nunmehr in der Lage sind, die Steuerungen SOUND, ENV und ENT voll auszunutzen, können wir uns mit weiteren anderen zugehörigen Funktionen und Kommandos befassen.

Bei der Beschreibung der Kanalnummer der SOUND-Steuerung werden Sie sich erinnern, daß durch die Hinzunahme von 64 der Ton in der Form "gehalten" wurde, daß er sich in einer Warteschlange befand, ohne Wiedergabe bis zur Auslösung. Die Freigabe eines Tones erfolgt hierbei durch Einsatz der RELEASE-Steuerung. Dem Ausdruck RELEASE folgt eine bit-signifikante Nummer, in der jedes Bit eingesetzt wird, um auf einen von drei möglichen Kanälen hinzuweisen, die ausgelöst werden können. Auch hier geht es weniger darum, volles Verständnis zu erreichen, so lange Sie davon ausgehen können, daß:

4 Kanal C bedeutet
2 Kanal B bedeutet
1 Kanal A bedeutet

...und daß eine Kombination von Kanälen dadurch freigegeben wird, daß Werte für jeden der Kanäle hinzugefügt werden.

Um somit gehaltene Töne auf allen drei Kanälen freizugeben, ist wie folgt zu verfahren:

```
RELEASE 7
```

...wobei 7 = 1 + 2 +4. Werden keine Kanäle gehalten, so wird die RELEASE-Steuerung unterdrückt.
Prüfen Sie folgendes Muster:

```
SOUND 1+64,90
SOUND 2+64,140
SOUND 4+64,215
RELEASE 3: FOR t=1 TO 1000: NEXT: RELEASE $
```

Die Geräusche, die Sie von diesen SOUND-Befehlen erwarten könnten, werden erst dann erzeugt, wenn die erste RELEASE-Steuerung einsetzt, die die Tonfolgen auf den Kanälen A und B freigibt. Nach Ablauf der Verzügerung erfolgt Freigabe von Tonvolumens auf Kanal C.

Darüber hinaus gibt es noch ein weiteres Verfahren, mit Hilfe dessen mehr als ein Ton abgestimmt werden kann.
Wird ein Ton einer Warteschlange hinzugefügt, die das Haltebit (hinzugefügte 64) enthält, dann wird nicht nur dieser Ton gehalten, sondern alle darauffolgenden Töne in die Warteschlange überführt. Werden mehr als 4 weitere Töne zur Warteschlange überführt, dann unterbricht die Maschine, bis die Warteschlange freigegeben ist, gegebenenfalls durch Einsatz eines Unterprogramms, das nach einer feststehenden Zeitperiode unter Einsatz von AFTER bzw. EVERY aufgerufen wird. Hierbei handelt es sich jedoch nicht um ein besonders gutes Verfahren für den Einsatz des Tonsystems, da das Programm, das die Tonsteuerungen enthält, von Zeit zu Zeit unterbricht, wenn sich die Tonwarteschlangen füllen. Gleiches gilt für einen Satz längerer Tonfolgen, die in sehr schneller Folge hinzugefügt werden.

Versuchen Sie folgende Schaltung:

```
10 FOR a=1 TO 8
20 SOUND 1,100*a,200
30 NEXT
40 PRINT "hallo"
run
```

Sie werden feststellen, daß der Ausdruck "hallo" nicht sofort erscheint, sondern erst nach den ersten drei Tönen. Das hängt damit zusammen, daß eine Programmausführung erst dann fortge-

setzt werden kann, wenn innerhalb der Warteschlange ein freier Raum gegeben ist.

BASIC enthält einen Unterbrechungsmechanismus in der Art des in den AFTER- und EVERY-Kommandos enthaltenen sowie in ON BREAK GOSUB. Hierdurch kann ein Ton-Unterprogramm angegeben werden, das nur dann aufgerufen wird, wenn ein freier Raum innerhalb der erforderlichen Warteschlange erscheint.

Setzen Sie hierzu folgendes ein:

```
10 a=0
20 ON SQ(1) GOSUB 1000
30 PRINT a;
40 GOTO 30
1000 a=a+1
1010 SOUND 1,a,200
1020 IF a< 200 THEN ON SQ(1) GOSUB 1000
1030 RETURN
run
```

Sie werden feststellen, daß das Programm nie unterbricht. Das SOUND-Kommando wird nur dann aufgerufen, wenn die Warteschlange des Kanals A (1) eine Unterbrechung aufweist. Diese Bedingung wird durch das ON SQ(1)-GOSUB-Kommando in Zeile 20 erfaßt.

Das Kommando startet eine Unterbrechung, die das Ton-Unterprogramm aufruft, wenn eine Unterbrechung in der angegebenen Warteschlange erscheint. Nach Einsatz von ON SQ GOSUB ist erneut zu initialisieren und zwar über Leitung 1020 im Ton-Unterprogramm. In diesem Beispiel führt das Ton-Unterprogramm nur dann eine Neu-Initialisierung durch, wenn der Wert von "a" unter 200 liegt.

Innerhalb eines kompletten Programms, bei dem sich bewegliche Objekte am Bildschirm befinden können, darüber hinaus Summen und ähnliche Ausdrücke, kann ein Hintergrundton ständig gehalten werden, wobei ein Unterprogramm für jede Note nur dann aufgerufen wird, wenn sich eine Unterbrechung in der Warteschlange befindet. Hierdurch wird sichergestellt, daß das Programm niemals unterbrochen wird, während es auf eine Unterbrechung wartet. Wurden die Tonwerte für diese Klangfolge von DATA-Anweisungen gelesen, so läßt sich die Tonroutine so einstellen, daß eine Neu-Initialisierung gestoppt wird, bevor die Datenmenge erschöpft ist.

Bei der Zahl innerhalb der Klammern des ON SQ () GOSUB-Kommandos kann es sich um 1,2 oder 4 handeln, davon abhängig, welche Kanalwarteschlange auf eine Unterbrechung geprüft wird.

Darüber hinaus besteht eine Funktion SQ (), die innerhalb eines Programmes eingesetzt werden kann, um den laufenden Zustand irgendwelcher Tonkanäle zu erfassen. Die Zahl innerhalb der Klammern nach SQ lautet auch hier 1,2 bzw. 4, um anzugeben, für welchen Kanal die Information bestimmt ist. Die Funktion überträgt eine bit-signifikante Nummer und erfordert wiederum ein Verstehen der Binärzahlen zur Entschlüsselung. Die Bits des übertragenen Wertes besitzen folgende Bedeutung:

BIT	DEZIMAL	BEDEUTUNG
0,1,2	1,2,4	Anzahl der freien Stellen in der Warteschlange
3	8	Note am Kopf der Schlange für Rendezvous mit Kanal A
4	16	Note am Kopf der Schlange für Rendezvous mit Kanal B
5	32	Note am Kopf der Schlange für Rendezvous mit Kanal C
6	64	Obere Note in Warteschlange mit Haltebit (Schlange wird gehalten)
7	128	Eine Note erklingt laufend

Wenden Sie dieses einfache Beispiel an:

```
10 SOUND 2,200
20 x=SQ(2)
30 PRINT BIN$(x)
run
```

Hierdurch wird die Binärzahl 10000100 gedruckt, in der Bit 7 gesetzt ist, das anzeigt, daß der Kanal permanent läuft, wenn die SQ-Funktion eingesetzt wurde. Die letzten drei Digits 100, nach Umsetzung in Dezimalwert, geben den Wert 4, der anzeigt, daß 4 freie Eingänge in der Warteschlange angetroffen wurden. Diese Funktion kann zur Prüfung des Status einer Warteschlange an einem bestimmten Punkt innerhalb des Programms eingesetzt werden, während ON SQ () GOSUB prüft und auf den Status einer Warteschlange an einem unbestimmten Punkt reagiert.

Bisher haben sich alle Beispiele auf ein bzw. zwei Töne bezogen. Die Bearbeitung einer ganzen Gruppe unbestimmter Noten, beispielsweise in einem Musikstück, kann durch Erfassen der erforderlichen Noten in einer DATA-Anweisung erfolgen, von wo aus diese in ein SOUND-Kommando eingelesen werden können:

```
10 FOR oktave=-1 TO 2
20 FOR x=1 TO 7: REM Noten pro Oktave
30 READ note
40 SOUND 1,note/2^oktave
50 NEXT
60 RESTORE
70 NEXT
80 DATA 426,379,358,319,284,253,239
```

Das abschließende Musterprogramm in diesem Abschnitt arbeitet im wesentlichen nach diesem Grundprinzip. Ton und Rhythmus auf den Kanälen A und B verwenden das Rendezvous-Verfahren für den Takt. Das Beispiel zeigt einen der Wege, bei denen DATA in der Form formatiert werden kann, Note, Oktave, Länge und Rendezvous-Information zu vereinen:

```
10 REM Zeile 190 fuer den Sopran Notenschluessel
20 REM Zeile 200 fuer den Bass Notenschluessel
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-B6A2Cr1f1g1f1g1a
b1A1-b2C2g2A2g2f1g1a2g2f6e2c2e2c2g2e2c1-B1A2g2f4e4d8c
3f1c2d4-b2fr2-B2A2g2f6e2gr4C4-B1a1f1-b1g2c2-b4a4g4fr6
A2-B4-B2Ar4-B2A2g2f6e2g4C4-B1A1f1-B1g2C2-B4A4g8f.
```

```
190 DATA r4f4f8f4e4c4fr8f4e2f2e4d2e2d8c8c6e2f4g4g8e4f3f
1c4dr8g4cr4e4c6f2d4c4c8fr8-e4dr8g8c4e4c6f2d4c4c8f.
200 REM Ton fuer 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$,ch1%)
360 ch1%=ch1%+1
370 GOTO 300
380 REM Ton fuer Kanal B
390 p2$=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
```

Einsatz der graphischen Möglichkeiten

Dieser Abschnitt beschreibt die zur Verfügung stehenden graphischen Möglichkeiten. Das erste Beispiel führt uns nach und nach in die Hauptmerkmale ein.

Zu Beginn unterteilen wir den Bildschirm in ein Textfenster (Zeile 40) und in ein Graphikfenster (Zeile 30), mit Wahl der Betriebsart (MODE) und einer Anzahl Leuchtfarben innerhalb des Abschnittes (Zeile 20):

```
10 REM mask and tag in window
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
```

Während des Laufes dieses Programms (RUN) sehen Sie eine Farbmarkierung auf halber Strecke an der rechten Seite des Bildschirms. Diese Markierung wurde auf Farbnummer 2 eingestellt (Magenta/Zyan) über Zeile 50, wobei der Ursprung der Koordination zur linken unteren Ecke des Bildschirms versetzt wurde.

Die MODE-Steuerung hat den Graphik-Cursor auf den Ursprung der Koordinaten (X=0, Y=0) gesetzt, so daß wir eine Diagonale durch die Markierung mit Linie 60 ziehen können.

```
60 DRAW 200,200,3
```

Lassen wir das neue Programm anlaufen, um die Auswirkung zu sehen. Anschließend geben wir hinzu:

```
80 MOVE 0,2:FILL 3
```

Zeile 80 setzt den Graphik-Cursor gerade innerhalb einer der beiden Hälften des Symbols und füllt dieses mit Farbe 3 aus. Die Grenzlinie der Farbausfüllung ist die Kante des Graphikfensters (in diesem Falle auch die Kante des Symbols), wobei alle Eingaben des Graphikstiftes (3) bzw. darüber hinaus in dieser Farbe erscheinen (ebenfalls 3).

Lassen wir nunmehr das sich ergebende Programm anlaufen.

Zur Überprüfung der "Farb"-Kanten ist nachstehende Linie 70 hinzuzufügen. Hierbei geht es nur darum, daß es sich bei der

Farbe um die gleiche wie die der diagonalen Linie handelt, die die Hälfte des Symbols begrenzt.

```
70 GRAPHICS PEN 1
run
```

Anweisung von Zeile 80 zur FÜLLUNG mit Farbe 1, anschließendes Anlaufen zur Überprüfung dieses letztgenannten Punktes. Anschließend Rückführung zum Original (FILL 3).

Anschließend sind die Zeilen 100 bis 140 hinzuzufügen, die ein Viereck zeichnen:

```
100 MOVE 20,20
110 DRAW 180,20
120 DRAW 180,180
130 DRAW 20,180
140 DRAW 20,20
run
```

Das Viereck wird durch Zeile 70 in Farbe 1 gezeichnet. Wird Zeile 70 weggelassen, so müssen wir ein ",1" hinzufügen, entweder als drittes Parameter der MOVE-Steuerung in Zeile 100 oder als drittes Parameter der DRAW-Steuerung in Zeile 110, um den Computer anzuweisen, die Graphik-Schreibstifte zu wechseln.

Punktreihe

Die Linien müssen nicht durchgezogen sein, sondern können unterbrochen dargestellt werden. Die MASK-Steuerung ermöglicht uns die Angabe der Größe der Puntke. Der Verlauf wiederholt sich alle 8 Pixel und auf jeder folgenden Zeile wird das Punktschema fortgesetzt, das auf der letzten Zeile unterbrochen wurde. Eine neue MASK-Steuerung (wahrscheinlich mit dem gleichen Parameter wie die laufend) führt zur Rücksetzung der Punktlogik zum Startpunkt des 8-Pixel-Musters.

Die Punktschablone stellt gegenwärtig eine aus einem Byte bestehende Binärzahl dar, in der die Bits die jeweilige Schreibstiftfarbe angeben. In unserem Beispiel benutzen wir eine konstante Binäre (durch "&X" aufgerufen) mit der angegeben wird, daß wir 4 Pixel in der Mitte von jeweils einer 8 Pixel umfassenden Gruppe zeichnen wollen, wobei jeweils die äußeren nicht gezeichnet werden. Hierdurch ergibt sich eine gestrichelte Linie mit vier Ein- und vier Ausschaltpixel. Hierzu ist

folgende Form hinzuzufügen:

```
90 MASK &X00111100
run
```

Unterbrechen wir jedoch einen Moment!
Dieses Programm liefert uns keine genaue Fortsetzung der Punktlinie an den Kanten wie erwartet. Der Grund hierfür liegt darin, daß jeder Kantenpunkt gegenwärtig doppelt ausgelöst wird, einmal als letztes Pixel der Zeile und erneut als erstes Pixel der nächsten Zeile. Ein schwerfälliger Weg ist folgende Eingabe:

```
115 MOVE 180, 22
125 MOVE 178, 180
135 MOVE 20, 178
run
```

...woraus sich der gewünschte Effekt ergibt. Jedoch gibt es einen einfacheren Weg, der darin besteht, ein zweites Parameter ",0" am Ende der MASK-Steuerung hinzuzufügen, das den Computer anweist, den ersten Punkt jeder Zeile **nicht** auszuführen. Korrigieren Sie Zeile 90 wie nachstehend:

```
90 MASK &00111100,0
```

...und löschen Sie die Zeilen durch folgende Eingabe:

```
115
125
135
```

Wenn wir erneut anlaufen lassen, so zeigt sich in diesem Falle das gestrichelte Kästchen als symmetrisch. Lautet das zweite MASK-Parameter ",1", so wird die Steuerung zurückgesetzt, um ganze Linien zu zeichnen, einschließlich des ersten Pixel.

Betrachten Sie nunmehr das Bild zwischen den Strichen der Zeile. Hier befindet sich etwas im unteren rechten Dreieck, das sich nicht im oberen linken Dreieck befindet. Hierbei handelt es sich um das Graphik-Papier, das auf Farbe 2 über die Steuerung CLG 2 in Zeile 50 gesetzt wurde, jedoch nicht sichtbar ist im oberen linken Dreieck, da es die gleiche Farbe wie der Hitnergrund hat. Veränderung der Zeile 50 wie folgt:

```
50 CLG 2: GRAPHICS PAPER 0
```

...und erneuter Anlauf des Programms. Das Papier ist nunmehr klarer um das Kästchen zu sehen.

Es ist ferner möglich, das Graphikpapier unsichtbar zu gestalten, bzw. wie wir es nennen: "transparent". Dies bedeutet, daß eine gestrichelte Linie über einem bestehenden Bild die Zwischenräume zwischen den Punkten ausfüllt. Die graphische Zeichnung wird transparent durch Hinzunahme eines ",1"-Parameters zur Steuerung des graphischen Schreibstiftes. (Es kann umgeschaltet werden zu nicht-transparent bzw. überdeckend durch ein Parameter ",0"). Veränderung der Linie 70 wie folgt:

```
70 GRAPHICS PAPER 1,1
run
```

...und Feststellung des Ergebnisses.

In der gleichen Form wie das Zeichnen von Linien (bzw. das Setzen von Punkten) ist es möglich, normale Schriftzeichen auf die Lage des Graphik-Cursors zu schreiben. Hierbei liegt der Vorteil darin, daß wir die Buchstaben mit wesentlich größerer Genauigkeit setzen können (innerhalb eines Pixels im Gegensatz zu 8 Pixel) und daß wir darüber hinaus Buchstaben mit der jeweiligen Farbe einsetzen können (im Weiteren erläutert).

Für das Einschreiben von Buchstaben an die Stelle des Graphik-Cursors ist einfach der Graphik-Cursor an der linken Kante an die Stelle des Buchstabens zu setzen, anschließend die Steuerung TAG (bzw. TAG #1 usw. für weitere Textströme) einzusetzen, gefolgt durch normale PRINT-Steuerungen.

Somit wird der Graphik-Cursor automatisch um 8 Pixel weitergeführt, nachdem der jeweilige Buchstabe geschrieben ist. Hinzunahme von:

```
160 MOVE 64, 108
170 TAG
180 PRINT "SALLY"
190 TAGOFF
run
```

(Jeder von BASIC übertragene Befehl wird zum Text-Bildschirm übertragen, unabhängig vom Zustand des Schalters TAG/TAGOFF, wobei jedoch empfeholen wird, die TAG-Zuordnung zu löschen, sobald der TAG-befehl ausgeführt ist.)

Was sagen jedoch die Pfeile hinter der Bezeichnung aus?
In diesem Falle handelt es sich um Symbole für Wagenrücklauf (CHR$(13) bzw. Zeilenvorschub CHR$(10). Die Graphik-Software übersetzt alle der 32 ASCII-Zeichen in ihre jeweils druckbaren Versionen (als stünde jeweils ein CHR$ (1) ; davor, wenn diese zum Textbildschirm übertragen werden. Der Grund hierfür liegt darin, daß die Mehrzahl der ersten 32 Steuercode nur für den Textbildschirm von Bedeutung sind. Gleiches erfolgt beim Übergang zur rechten Seite des Graphikfensters, wo ebenfalls keine Einhüllung stattfindet.
Die Symbole für Wagenrücklauf und Zeilenvorschub können durch das normale Abschlußverfahren des PRINT-Befehls unter Einschluß eines Semikolons unterdrückt werden, d.h.:

```
180 PRINT "SALLY";
run
```

Eine Textübertragung zum Graphik-Bildschirm unter Einsatz von TAG richtet sich nach den gleichen GRAPHIK-Schreibstift-Steuerungen wie beim Ziehen von Linien. Somit wird der Name mit GRAPHIK-SCHREIBSTIFT 1 geschrieben und ist transparent.
Die Steuerung:

```
150 GRAPHICS PEN 1,0
run
```

...schaltet zurück auf überdeckend, während:

```
150 GRAPHICS PEN 0,1
run
```

...transparent mit Farbe Nummer 0 schreibt.

Löschen Sie nunmehr erneut Zeile 150 und RUN. Farbnummer 1 + Transparentmodus (vorher in Zeile 70 gesetzt) wird für den Graphikstift wieder aktiviert.

Transparente Zeichen

Durch Einsatz der vorhandenen Steuercodes ist es ebenfalls möglich, Zeichen auf den Textbildschirm transparent aufzubringen. Hierzu ist hinzuzufügen:

```
200 PRINT #8,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 Strom #2 auf Transparent-Modus. Hierbei kann festgestellt werden, wie die Unterstreichung die Sternchen überdeckt. Hierdurch wird aufgezeigt, daß es möglich ist, zusammengesetzte Zeichen aufzubauen, selbst in mehreren Farben. Zeile 230 schaltet den Transparentmodus ab und setzt den Strom #2 zurück auf die Betriebsart 'überdeckend'.

Farb-Betriebsarten

Durch Einsatz einer Graphik-Betriebsart für Farbe ist es möglich, die Farbe der laufenden Zeichnung mit der bereits auf dem Bildschirm befindlichen zu kombinieren. Die endgültige Farbe für jedes Pixel wird durch Bildung der logischen Kombination berechnet und zwar der früheren Farbe für den Pixel mit der Schreibstiftfarbe (Schreibstift bzw. Bildschirm). Die hierfür vorgesehenen logischen Kombinationen lauten XOR, AND und OR. Die Farb-Betriebsart kann eingestellt werden entweder als vierter Parameter von DRAW/DRAWR, PLOT/PLOTR und MOVE/MOVER bzw. durch DRÜCKEN eines Symbols CHR$(23); CHR$ (Betriebsart). In jedem Falle setzt ein Wert von 1 die Einstellung XOR, 2 von AND und 3 von OR. Ein Wert von 0 führt auf die Standardbetriebsart zurück, innerhalb der die Graphikfarbe in der jeweils vorhandenen geführt wird.

Das folgende Beispiel zeigt den Einsatz der XOR-Kombination. XOR wird oftmals in sogenannten Schildkröten-Graphiken verwendet, da diese Steuerung die Eigenschaft hat, das Originalbild zweimal darzustellen. Somit wird die Zeichnungsroutine für das Viereck zweimal ausgeführt (innerhalb der Zeilen 110 und 130) und der Druckvorgang TAG wird ebenfalls zweimal ausgeführt (in den Zeilen 170 und 190). Die FRAME-Steuerungen bewirken eine ausreichende Verzögerung, um den Effekt sichtbar

zu machen. Vergleichen Sie in Zeile 90 den Einsatz der Steuerungen ohne das erste Parameter. Für diese Steuerungen gilt dies ohne weiteres, wobei Sie lediglich die laufenden Steuerungen der ersten Parameters unverändert zu übernehmen brauchen.

Das dritte Parameter (,1) der MOVE-Steuerung in Zeile 220 setzt den Graphik-Stift auf 1, und löscht die in Zeile 60 gesetzte "3". Die Betriebsart XOR wird als viertes Parameter der DRAWR-Steuerung in Zeile 230 gesetzt. Auch hier sind die fehlenden Parameter festzustellen.

Ein Kriterium des ersten Punktzeichnungseffektes kann festgestellt werden, wenn die MASK-Steuerung in Zeile 90 gelöscht wird. Die Kanten des Vierecks verschwinden, da sie zweimal gezeichnet wurden (am Ende jeder Zeile und am Beginn der nächsten) und werden demzufolge durch den XOR-Vorgang gelöscht.

```
10 REM Inks mit XOR-Verknuepfung
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
```

Bewegungsabläufe bei graphischen Darstellungen

Bewegungseindrücke können durch Umschaltung zu bestimmten Farben hervorgerufen werden. Obwohl hierbei die Inhalte des Bildschirmspeichers unverändert bleiben, besteht der Eindruck einer Bewegung. Ein solches Beispiel befindet sich im "Begrüßungs"-Programm Ihres Diskettensystems (geben Sie RUN "disc" ein, um diese Vorführung zu verfolgen). Eine einfache Palettenschaltung dieses Beispiels reicht jedoch dann nicht aus, wenn sich bewegte Abläufe überlappen sollen. Das folgende Beispiel arbeitet mit einer ODER-Schaltung von Farben zur Einschreibung der Nummern 1 bis 4 auf dem Bildschirm. (Die Form wird durch Abtastung des gedruckten Schriftzeichens in der unteren linken Ecke bestimmt, wobei alles, was die Abtastung liefert, in großer Blockgraphik dargestellt wird.)

Die Einschreibung der Farben erfolgt unter Verwendung von 1,2,4 und 8 über die eingeschaltete OR-Betriebsart - in diesem Falle mit einer Steuerzeichenfolge; siehe Zeile 50.

Die Zeilen ab 160 drehen diese Palette entsprechend einer mathematischen Formel, die als Blockgraphik-Nummer dargestellt wird. Die Einstellung der Farben erfolgt durch Prüfung der einzelnen Farbe mit Bestimmung, inwieweit diese die Binärkomponente enthält, die erforderlich ist. Wurde beispielsweise die Nummer 3 auf Farbe 4 gesetzt, so haben wir allen Farben, deren Zahl eine Binärziffer 4 enthält, eine sichtbare Farbe zuzuordnen. Diese Farben sind:

4 (0100),5(0101),6(0110),7(0111),12(1100),13(1101),
14(1110),15(1111)

Innerhalb einer praktischen Anwendung sind alle Farben, die wir benötigen, zu jedem Bewegungszustand zu berechnen, wobei die Zeilen 180 bis 200 durch eine raschere Programmfolge ersetzt werden.

```
10 REM Ueberlappende Bewegungen
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
```

Möglichkeiten der Farbschattierung

In den obigen Beispielen haben wir gesehen, wie durch die geschriebenen Graphiken der Farben 1,2,4 und 8 ein Bewegungseffekt durch Farbwechsel erzeugt werden kann. Bei Verwendung der gleichen Farben, jedoch bei verschiedener Anordnung der Farben läßt sich ein völlig abweichender Effekt erzielen. Dieser Effekt ist als "Farbabschattierung" bekannt und wird in nachstehendem Beispiel demonstriert:

```
10 REM Gebirge
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 1,9:NEXT
60 FOR i=8 TO 15:INK 1,20: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
```

Zur Erläuterung dieser Funktion müssen wir erneut die Farbnummern binär darstellen. Beginnend mit der höchsten Farbnummer (15) könnten alle Farben mit dem vorhandenen Bit "8" (15 bis 8) auf Zyan gesetzt werden. Anschließend werden alle Farben mit dem vorhandenen Bit "4" (7 bis 4) auf grün gesetzt. Die Farben 2 und 3, jeweils mit dem Bit "2" werden auf rot gesetzt und schließlich die Farbe 1 auf helles weiß, wobei die Farbe 0 blau bleibt.

Auf dem Bildschirm werden die Farben mit einem ODER-Gatter verarbeitet - siehe Zeilen 230 und 240. Die auf dem Bildschirm innerhalb eines bestimmten Pixels sichtbare Farbe wird durch das höchstwertige Bit des Ergebnisses an diesem Punkt bestimmt. Demzufolge wird ein Bild innerhalb einer "höherwertigen" Ebene immer ein Bild in einer weniger bedeutenden Ebene abdunkeln, wobei jedoch der Hintergrund ausgespart bleibt und wieder sichtbar wird, wenn das höherwertige Bild ausgeblendet wird. Die Möglichkeit der Ausblendung eines Bildes ist ein Einsatz unter Verwendung der Betriebsart AND mit den Farbnummern 7,11,13 bzw. 14, wodurch die ursprünglichen Farben von 8,4,2 und 1 ausgeblendet werden (siehe Zeilen 280 und 290).

Jedes der vier obengenannten Programme zeigt besondere Aspekte der Graphik des CPC664-Systems. Die Graphik ist wahrscheinlich der interessantere Aspekt des Umgangs mit dem Computer, wobei der erreichte Erfolg von Ihrem Einfallsreichtum, dem des Programmierers abhängt. Es wäre unmöglich, alle graphischen Möglichkeiten innerhalb dieses Handbuches zusammenzufassen, wobei wir sicher sind, daß Sie Ihre Erfahrung fortsetzen werden, um noch viele weitere graphische Steuerungen und Funktionen herauszufinden.

Die Anlage selbst

Um dieses Kapitel abzuschließen, möchten wir noch einige Erläuterungen bezüglich des Computers anschließen.

Der CPC664 beruht auf dem Z80-Mikroprozessor. Die Einheit umfaßt darüber hinaus ein 32 K-ROM, das in zwei Abschnitten adressiert wird, wobei die obere Hälfte den BASIC-Interpreter enthält, der innerhalb des Speicherbereiches von &C000 bis &FFFF arbeitet. Die untere Hälfte ist eingeteilt zwischen &0000 und &4000 und enthält die Firmware, die alle normalen Aufgaben wir z.B. den Antrieb der Maschine, die Erfassung der

Tastatur und die Steuerung des Bildschirms usw. übernimmt.

Der CPC664 umfaßt darüber hinaus eine ULA-Einheit, durch die die 8-Bit-CPU-Einheit mehr als 64 K Speicherraum adressieren kann. Aus diesem Grunde arbeiten der Bildschirm und BASIC ROM mit den gleichen Adressen. Die oberen 16K des RAM werden vom 6845-Video-Steuerchip dazu verwendet, die Pixel Farbinformation zu speichern, die das Bild entstehen läßt. Die Tastatur wird über die Parallelschnittstelle 8255 abgetastet, die darüber hinaus den Soundchip AY-3-8912 steuert.

Das Druckergatter arbeitet mit einer 8-Bit-Verriegelung.

Hierbei stehen 7 Datenbits zur Verfügung, während das 8. Bit eingesetzt wird, um das Strobe-Signal zu verriegeln.

Am Erweiterungsgatter werden alle Standard-Bussignale Z80 zur Verbindungseinheit geführt, jedoch ohne Zwischenspeicherung.

Für Ihre Notizen:

ANHANG 1

ENDABNEHMER – LIZENZABKOMMEN ZWISCHEN DIGITAL RESEARCH UND SCHNEIDER

Hinweis für den Anwender:

Lesen Sie diesen Abschnitt sorgfältig durch bevor Sie das Disketten-Paket öffnen.

Durch das Öffnen des Disketten-Pakets dokumentieren Sie die Zustimmung und Kenntnis des Inhalts und der Bedingungen dieses Endabnehmer-Lizenzabkommens.

1. DEFINITIONEN

1.1 DRI steht für DIGITAL RESEARCH (CALIFORNIA) INC., P.O. Box 579, Pacific Grove, California 93950, dem Inhaber des Copyright für dieses Programm.

1.2 Maschine steht für den bestimmten, einzelnen Mikrocomputer, auf dem Sie dieses 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 DRI Ihnen ausliefert, unabhängig von der speziellen Verwendung und Modifikation Ihrerseits.

1.4 Schneider steht für SCHNEIDER COMPUTER DIVISION, Silvastr. 1, 8939 Türkheim.

Die Verantwortung für die Programm-Auswahl, ihre Verwendung, sowie die Installation liegt beim Käufer.

2. LIZENZ

Sie dürfen

2.1 das Programm auf einer einzigen (dieser) Maschine betreiben.

2.2 das Programm auf dieser Maschine für Sicherheitskopien oder Änderungen in beliebige maschinenlesbare oder gedruckte Form übertragen, wenn es Ihren privaten, persönlichen Zwecken dient. Die Anfertigung von bis zu drei Programm-Kopien ist erlaubt. Verschiedene Programme verbieten oder beschränken diese Anfertigung von Kopien. Sie sind durch den Hinweis „Copy protected" gekennzeichnet. Das Kopieren und Vervielfältigen der Dokumentation und zugehörigem gedruckten Material ist verboten. Das Disassemblieren des Programmcodes ist verboten.

2.3 das Programm auf dieser Maschine mit anderen Programmen betreiben, sofern es Ihren privaten, persönlichen Anwendungen dient. Jeder mit einem anderen Programm betriebene Bestandteil dieses Programms unterliegt ebenfalls diesem Lizenzabkommen.

2.4 das Programm abtreten und einem Dritten lizenzieren, sofern Sie den Namen und die Adresse an DRI melden und der Dritte

a) den Inhalt und die Bedingungen dieses Lizenzabkommens akzeptiert

b) für DRI eine Kopie der Registrierkarte unterzeichnet und übersendet

c) die Lizenzgebühr entrichtet.

Wenn Sie dieses Programm übertragen, müssen Sie entweder gleichzeitig alle Kopien, einschließlich Original, in gedruckter oder maschinenlesbarer Form mit übergeben oder alle nicht übergebenen Kopien zerstören. Dies gilt auch für sämtliche modifizierten und auf andere Programme übertragenen Module.

Der Text des Copyrights muß auf jeder Kopie, Modifikation und jedem auf andere Programme übertragenen Modul entsprechend reproduziert werden.

JEDE DISKETTE TRÄGT EINE SERIENNUMMER. DAS PROGRAMM, EINE KOPIE, MODIFIKATION ODER EIN MODUL DESSELBEN DARF, WENN IM LIZENZABKOMMEN NICHT AUSDRÜCKLICH GENEHMIGT, WEDER GANZ NOCH TEILWEISE FÜR EINEN DRITTEN ZUGÄNGLICH GEMACHT WERDEN. BEI MISSBRAUCH 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 gemeinsam mit sämtlichen Kopien, Modifikationen und in irgendeiner Form übertragenen Modulen zerstören. Die Lizenzrechte erlöschen ebenfalls aufgrund außerhalb dieses Abkommens dargelegter Vertragsbedingungen bzw. bei Verletzung der Übereinkunft. Durch den Erwerb dieses Programms stimmen Sie den hier aufgeführten Vertragsbedingungen zu.

4. EINGESCHRÄNKTE GARANTIE

SCHNEIDER UND DRI ÜBERNEHMEN KEINE GARANTIE FÜR DIE EIGNUNG DES PROGRAMMS FÜR BESTIMMTE ANWENDUNGEN. DAS RISIKO BEZÜGLICH QUALITÄT UND LEISTUNGSFÄHIGKEIT LIEGT BEIM KÄUFER. SOLLTE DAS PROGRAMM DEFEKT SEIN, ÜBERNIMMT DER KÄUFER DIE NOTWENDIGEN KOSTEN FÜR REPARATUR UND SERVICE.

Weder Schneider noch DRI übernehmen die Garantie dafür, daß die durch das Programm angebotenen Funktionen den Anforderungen des Benutzers genügen bzw. fehlerfrei sind. Schneider gewährleistet, daß sich die Programm-Diskette in einem einwandfreien Zustand hinsichtlich Material und Verarbeitung befindet. Die Garantiezeit für diese Eigenschaften beträgt 90 Tage ab Lieferdatum.

5. HAFTUNG

Schneider haftet nur für Disketten, die der eingeschränkten Gewährleistung nicht entsprechen.

DRI UND SCHNEIDER HAFTEN NICHT FÜR SCHÄDEN, DIE IN IRGENDEINER FORM BEI DER BENUTZUNG DIESES PRODUKTS ENTSTEHEN.

6. REGISTRIERKARTEN

Die Programme werden von Zeit zu Zeit durch DRI überarbeitet. Diese Updates werden nur an Kunden ausgeliefert, deren unterzeichnete Registrierkarte bei DRI oder einem authorisierten Händler vorliegt.

DRI ist jedoch nicht verpflichtet, die Programme zu überarbeiten und entsprechende Updates auszuliefern.

7. ALLGEMEINES

Das Programm darf nur im Rahmen dieser Übereinkunft lizenziert, abgetreten oder übertragen werden. Jeder Versuch, diese Rechte auf andere Weise zu lizenzieren, abzutreten oder zu übertragen, zieht keine Verpflichtungen des Herstellers nach sich.

Fragen, die sich aus dieser Übereinkunft ergeben, können direkt an

Digital Research Inc.
P.O. BOX 579
Pacific Grove
California 93950

gerichtet werden.

DIESES LIZENZABKOMMEN DARF NICHT DURCH KAUF, WERBUNG ODER ANDERE REPRÄSENTATIONEN MODIFIZIERT WERDEN, SONDERN NUR DURCH EINEN SCHRIFTLICHEN VERTRAG MIT EINEM AUTHORISIERTEN HÄNDLER ODER SCHNEIDER.
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 SCHNEIDER KEINE ZUSÄTZLICHEN SCHRIFTLICHEN ODER MÜNDLICHEN ABSPRACHEN GIBT.

Für Ihre Notizen:

ANHANG 2
Fachwörterverzeichnis

Einige häufig verwendete Begriffe aus der Welt der Datenverarbeitung und deren Erklärung für die Benutzer des CPC664

Abenteuerspiele (Adventure Game)

Für einige das Höchste, für andere schlicht langweilig. Ein Computerspiel mit viel Text, bei dem der Spieler eingeladen ist, an einer Reihe von Zufallsereignissen teilzunehmen und sich in einem Irrgarten oder Labyrinth zurechtzufinden.

Abschneiden (Truncated)

Eine Zahl oder ein Text werden gekürzt, indem die führenden oder hintenstehenden Zeichen entfernt werden. Diese Funktion kann mit Absicht benutzt werden bei der Rundung einer Zahl. Unabsichtlich angestoßen gehen die zusätzlichen Zeichen verloren, da für die Zahl oder Zeichenkette nicht mehr Platz zur Verfügung steht.

Akkumulator (Accumulator)

Eine Speicherstelle im Microprozessor-Schaltkreis, im Herz des Microcomputers, in der Daten während ihrer Verarbeitung vorübergehend zwischengepeichert werden. Er wird ständig in der Maschinencode-Programmierung verwendet - BASIC Anwender interessiert es nicht, daß es ihn gibt!

Akkustikkoppler (Acoustic coupler)

Auch bekannt als akustisches Modem. Eine elektronische Zusatzeinrichtung, die einen Telefonhörer mit einem Computer verbindet und ihn 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 usw., um Software auszutauschen oder nur Daten und Informationen zu empfangen.

Algorithmus (Algorithm)

Ein großartig klingender Name für eine komplizierte Formel oder Rechenaufgabe. Eine Folge logischer und arithmetischer Schritte um eine ganu definierte Aufgabe mit dem Computer zu erledigen.

Adresse (Address)

Die Zahl in einer Anweisung, die die Lage einer 'Zelle' im Speicher des Computers identifiziert. Über ihre Adresse kann eine bestimmte Speicherstelle gefunden und ihr Inhalt 'gelesen' werden. Wenn es sich um ein RAM handelt, kann sowohl gelesen als auch - nach einer Änderung - zurückgeschrieben werden.

Alphanumerisch (Alphanumeric)

Eigenschaft, die die Zeichenmenge aus Buchstaben und Ziffern und Sonderzeichen (auch graphischen Symbolen) bezeichnet.

ALU

Abkürzung für Arithmetic Logic Unit (arithmetisch-logische-Einheit). Der Teil des Mikroprozessors, der die arithmetischen und logischen Operationen ausführt. Nur wichtig beim Programmieren im Maschinencode.

Ambiguous File Name

Dateiname mit einem oder mehreren Wildcard-Zeichen. Diese Dateinamen beziehen sich auf mehr als eine bestimmte Datei.

AMSDOS

Armstrad Disc Operation System.
Dieses Programm ermöglicht BASIC den Zugriff auf Disketten.

A/D Analog /Digital

Ein Zustand, bei dem Veränderungen zwischen einem Anfangs- und einem Endpunkt stufenlos sind, und nicht in einzelnen Schritten. Computer arbeiten digital, so daß die in der natürlichen Welt meist analog vorkommenden Ereignisse erst in digitale Einheiten umgeformt werden müssen, bevor ein Computer etwas damit anfangen kann.

Animation

Zeichentrickfilme sind die bekannteste Art von Animation. Bei der Computer Animation werden graphische Darstellungen bewegt, so daß ein Gefühl von 'echter' Bewegung vermittelt wird.

Anweisung (Statement)

Ein Befehl oder eine Folge von Befehlen in einem Programm.

Anwendungsprogramm (Applications program)

Ein Programm für eine spezielle Aufgabe, im Gegensatz zu einem allgemein anwendbaren Software-'Tool', wie z.B. Assembler, Sortierprogramm, Drucksteuerprogramm, usw.

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, z.B. in dem Ausdruck x+y=z, sind x und y Argumente.

Array

Eine zweidimensionale Matrix (Gitter), in der Daten gespeichert und mit 'waagrechten' und 'senkrechten' Koordinaten adressiert werden.

ASCII

American Standard Code for Information Interchange. Eine weit verbreitete Art der Darstellung von Nummern, Buchstaben und sonstigen Zeichen, die von der Tastatur eingegeben werden können, bzw. durch andere Kommandos erstellt werden. Die Codes des CPC664 sind in Anhang III aufgeführt.

Assembler

Maschinennahe Programmiersprache mit symbolischer Schreibweise für Befehle.

Auflösung (Resolution)

Die Möglichkeit festzustellen, wo ein Element einer Bildschirmausgabe endet und wo das nächste beginnt. Manchmal auch für die Fähigkeit des Computers, Errechnungen mit sehr großen Zahlen durchzuführen.

Ausdruck (Expression)

Eine einfache oder komplizierte Formel, im Programm zur Berechnung von Daten - die Formel beschreibt meist die Art von Daten, die bearbeitet werden kann.

Ausgabe (Output)

Alles was vom Computer als Ergebnis einer Errechnung kommt.

Austesten (Debugging)

Fehlersuche im Programm durch eine Kombination aus Versuchen und wissenschaftlichen Methoden.

Automatenspiel (Arcade Game)

Computerspiele besonderer Art mit viel 'Aktion', in denen Eindringlinge aus dem Weltall angreifen und besiegt werden, hungrige Monster-Figuren durch Irrgärten jagen und auffressen und der Spieler seine Figuren steuern muß, daß sie dem unangenehmen 'Tod' entgehen. Sie machen Spaß und trainieren die Reflexe, bieten aber nur wenig für den, der sich ernsthaft mit dem Computer beschäftigt.

Backup

Sicherheitskopie, 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 (ein allgemeiner symbolischer Befehlscode für Anfänger). Eine interpretative Programmiersprache, die in fast allen Heimcomputern benutzt wird. BASIC ist leicht zu erlernen und einfach zu benutzen, da Programme während ihrer Entwicklung zusammengestückelt und getestet werden können, und nicht wie bei Compilern gleich das komplette Programm vorliegen muß.

Basis (Base)

Basis im Zahlensystem. Das Binärsystem hat Basis 2, das Dezimalsystem hat die Basis 10 und das Hexadezimale (sedezimale) System hat die Basis 16. Siehe Anhang II für weitere Informationen.

Baud

Ein Bit pro Sekunde. Die Maßeinheit für die Übertragungsgeschwindigkeit bei der Datenübertragung.

BCD

Binär codiertes Dezimalsystem. Ein System zur Codierung von Dezimalzahlen, in dem jede Zahl mit vier binären Ziffern 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)

Eine Anforderung oder ein Kommando an den Computer, eine bestimmte Operation auszuführen. Eine Sammlung oder Folge von Anweisungen ergeben zusammen ein Programm.
Die primären logischen und mathematischen Prozesse, die ein Microprozessor durchführt. Jeder Befehl einer höheren Programmiersprache (auch Assemblerbefehle) müssen in einzelne Befehle aufgegliedert werden, die von einer Computer-CPU verstanden werden können. Ein einziges Kommando kann u.U. viele Befehle der CPU auslösen.

Bemerkung (Remark)

Eine nicht ausführbare Programmanweisung (REM), im Programm zu dokumentieren, was an dieser Stelle getan wird oder auch zur Kennzeichnung des Programmentwicklungsstandes.

Benutzerdefinierte Taste (User defined key - UDK)

Der CPC664 hat 32 Tasten, die redefiniert werden können, um verschiedene Aufgaben zu erledigen. Jede dieser Tasten kann bis zu 32 Zeichen aufnehmen.

Betriebssystem (Operating system)

Software, welche die Verwaltung des Speichers und die Koordinierung der Aktionen des Computers steuert.

Bildschirmeditor (Screen Editor)

Ein Aufbereitungsprogramm für Texte oder Programme, bei dem der Cursor an jede Stelle des Bildschirms positioniert werden kann, um die dort stehenden Zeichen zu ändern.

Binär (Binary)

(Siehe Basis) Das Zahlensystem mit der Basis 2, in dem alle Zahlen nur mit den beiden Ziffern 0 und 1 dargestellt werden.

Binärzahl (Binary number)

Eine Zahl in binärer Schreibweise. Bei der Programmierung des CPC664 werden sie mit einem vorangestellten &X angegeben (z.B. bedeutet &X0101 in dezimaler Schreibweise 5).

BIOS

Basic Input/Output System. Dies ist der Hardwareabhängige Bestandteil von CP/M, der für einen speziellen Computer-Typ entwickelt ist. Ein- und Ausgabefunktionen über Bildschirm, Tastatur, Diskette usw. werden durch BIOS bearbeitet.

Bit

Kurzform von Binary digiT (Binärziffer). Eine Binärziffer kann im binären Zahlensystem nur den Wert 0 oder 1 annehmen.

Bitsignifikant

Jedem der acht Bits eines Bytes ist eine gesonderte Bedeutung zugeordnet. Zur Auswertung muß jedes Bit getestet werden.

Boolesche Algebra (Boolean algebra)

Logische Verknüpfung von Operanden, die nur zwei Ergebnisarten (wahr, unwahr) kennt, die üblicherweise mit 0,1 dargestellt werden.

Booten

Laden eines Betriebssystems in den Speicher. Beim CP/M-Start unter BASIC wird automatisch ein kleines Boot-Programm von der Diskette geladen, welches wiederum 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 der Diskette geladen werden müssen.

Bus

Eine Gruppe von Verbindungen entweder im Computer selbst oder als Verbindung zu Peripheriegeräten, womit Informationen über den Zustand der CPU, der RAM oder andere Hardwareeigenschaften übertragen werden. Der CPC664 Bus ist auf der größeren der beiden Steckerleisten auf der Computerrückseite ansprechbar.

Byte

Eine Gruppe von acht Bits, welche die kleinste adressierbare Einheit im Speicher ist. Siehe Anhang II für weitere Informationen.

CAD

Computer Aided Design (Computer Unterstützter Entwurf). Normalerweise ein Zusammenspiel von Rechenleistung und Computergraphik als elektronisches Reißbrett, obwohl jede Art von Errechnungen für 'Entwürfe' als CAD gelten können.

CAE

Computer Aided Education (Computer unterstützte Schulung). Noch ein Schlagwort der Computerwelt. Hiermit ist das ganze Thema der Schulbildung mit Hilfe von Computern gemeint. CAI (Computer unterstützter Unterricht) und CAL (Computer unterstütztes Lernen) sind zwei Aspekte von CAE.

Cassette

Außer dem bekannten Magnetband versteht man hierunter auch alles sonst, was irgendwie 'verpackt' kommt, ROM Software usw.

CCP

Console Command Processor. Dieses CP/M-Modul interpretiert und bearbeitet die über Tastatur erfolgten Anwender-Eingaben. In der Regel sind diese Eingaben Befehle, die durch CCP geladen und ausgeführt werden.

Chip

Ein irreführender, jedoch umgangssprachlicher Begriff für eine integrierte Schaltung. Der Chip ist in Wirklichkeit eine dünne Scheibe aus speziell hergestelltem Silizium, auf der die Schaltung untergebracht ist.

Code

Außer in seiner normalen Bedeutung wird dieses Wort von Programmierern oft als Abkürzung von 'Maschinencode' benutzt.

Compiler

Ein komplexes Programm, das andere in einer höheren Programmiersprache als BASIC geschriebene Programme in das Befehlsformat des Microprozessor umwandelt. Dadurch wird eine höhere Ablaufgeschwindigkeit ermöglicht.

Computer der fünften Generation (Fifth generation computers)

Meist sehr große Computer, die unter Anwendung sogenannter künstlicher Intelligenz selbst programmiert werden können.

Computergenerationen (Computer generations)

Klassifizierung der großen technischen Meilensteine der Computerentwicklung. Die Gruppierungen, die sich daraus ergeben, werden als Generationen bezeichnet.

CP/M

Der de facto Standard von Digital Research für 8-Bit-Computer mit Plattenbetriebssystemen.

CPU

Central Processing Unit (Zentrale Prozessoreinheit). Das Herz eines jeden Computersystems, in dem Anweisungen interpretiert und deren Ausführung ausgelöst werden. In einem Microcomputer ist die CPU der Microprozessor selbst.

Cursor

Ein beweglicher Zeiger, der anzeigt, wo das nächste Zeichen auf dem Bildschirm erscheint.

Cursor Kontrolltasten (Cursor control keys)

Tasten, die den Cursor auf dem Bildschirm bewegen und auf denen Pfeile abgebildet sind. Sie werden oft zur Kontrolle von Bewegungen in Automatenspielen benutzt.

Datei (File)

Eine Datensammlung, die meistens auf Cassette oder Platte gespeichert ist, obwohl manche Computer auch RAM-Dateien unterstützten.

Dateiname (File name)

Name einer Datei. Bei Dr. Logo besteht ein Dateiname aus bis zu 8 alphabetischen oder numerischen Zeichen. Unter CP/M sind drei zusätzliche Zeichen für den Dateityp zugelassen, die durch einen Punkt vom File-Namen getrennt werden.

Datenaufnahme (Data capture)

Die Art, wie Daten von einem externen Gerät aufgenommen werden, das in beliebiger Weise an den Computer angeschlossen ist.

Datenfluß (Stream)

Der Weg, der bei der Ausgabe vom Computer benutzt wird, z.B. der Bildschirm, die Cassette oder der Drucker.

Datenbasis (Database)

Ein Bereich mit beliebigen Datenarten in verschiedenen adressierbaren Formaten.

Datenübertragung (Downfload)

Die Übertragung von Information von einem Computer zum anderen, wobei der Empfänger als 'downloading' Maschine bezeichnet wird. Der andere Computer ist dann die 'uploading' Maschine.

Default

Der ohne Anwender-Eingabe angenommene Standard-Wert. So wird beispielsweise beim Aufruf von CP/M Laufwerk A als Standard (Default)-Laufwerk angenommen.

Dezimale Schreibweise (Decimal system)

Auch bekannt als Dezimalsystem, ein Zahlensystem zur Basis 10 mit Ziffern von 0 bis 9, mit dem Einser, Zehner, Hunderter, Tausender usw. dargestellt werden.

Diagnostik (Diagnostic)

Die Meldung, die vom Computer automatisch ausgegeben wird, um mit der dazu gehörenden Erklärung auf einen Fehler im Programm hinzuweisen.

Dienstprogramm

Programm auf Diskette, mit deren Hilfe der Anwender bestimmte Aufgaben bewältigen kann (siehe transiente Programme).

Fehler (Bug)

Sie können ganz verschieden auftreten, als unerwarteter Programmablauf verursacht durch eine falsche Anwendung des Programms (z.B. ändert sich die Bildschirmfarbe, wenn gleichzeitig vier Tasten gedrückt werden), oder als Kette von Fehlern, die komplett und unwiderruflich ein Programm abstürzen läßt und sämtliche Daten im Speicher löscht.

Festkommazahl (Fixed-point number)

Eine Zahl, deren Dezimalpunkt an einer fest definierten Position steht.

File

Sammlung von Daten, die im allgemeinen auf Kassette oder Diskette gespeichert sind.

File-Name

Name einer Datei. Bei Dr. LOGO besteht ein File-Name aus bis zu 8 alphabetischen oder numerischen Zeichen. Unter CP/M sind drei zusätzliche Zeichen für den Dateityp zugelassen, die durch einen Punkt vom File-Namen getrennt werden.

Firmware

Software im ROM - ein Bindeglied zwischen reiner Software und reiner Hardware.

Flußdiagramm (Flowchart)

Eine systematische Darstellung von Programmablauf und logischen Prozessen, womit die Reihenfolge der Ereignisse während der Ausführung des Programms verfolgt werden kann.

Forth

Eine sehr schnelle Programmiersprache, mit Geschwindigkeit und Komplexität zwischen einer höheren Sprache und Maschinencode. Nicht für Anfänger geeignet.

Funktionstaste (Funktion key)

Eine Taste der Tastatur, der eine besondere Funktion zugeordnet ist, die zusätzlich oder an Stelle der eigentlich zugeordneten Funktion dieser Taste verwendet werden kann. Der CPC664 hat einige Tasten, die als Funktionstasten definiert werden können, wobei ein einzelner Tastendruck bis zu 32 Textzeichen auslösen kann, z.B. oft benutzte Anweisungen oder Befehle für Peripheriegeräte wie Drucker oder Modems.

Ganzzahl (Integer number)

Eine Zahl ohne Bruchteil, d.h. ohne Ziffern rechts vom Dezimalpunkt. Dagegen ist eine reelle Zahl eine Ganzzahl zuzüglich dem Bruchteil.

Gatter (Gate)

Logische Gatter erlauben die Weitergabe von Daten, falls bestimmte Bedingungen erfüllt sind. Es gibt viele Arten (OR, AND, XOR usw.). Siehe auch Boolesche Algebra.

Geräusch (Noise)

Der Tongenerator des CPC664 ermöglicht dem Anwender mit dem SOUND-Kommando Geräuscheffekte zu erzeugen, wie z.B. Explosionen.

Gleitkommazahl (Floating-point number)

Eine reelle Zahl, deren Dezimalpunkt an einer gewünschten Position steht. Diese Methode bietet sich für die Darstellung sehr großer Zahlen an.

Graphik (Graphics)

Der Teil von der Bildschirmausgabe, der nicht unmittelbar mit der Textzeichenausgabe zu tun hat, z.B. das Zeichnen von Linien, Kreisen usw. Mit einem geeigneten Drucker kann auch eine Ausgabe auf Papier erfolgen.

Graphikcursor (graphics cursor)

Ähnlich dem Textcursor, jedoch für den Graphikbildschirm. Zwar ist er beim CPC664 unsichtbar, aber notwendig für das Zeichnen von Graphiken. Nicht mit graphischen Zeichen verwechseln, (Anlage III), die ein Teil des Zeichenvorrats sind und beim Textcursor ausgegeben werden.

Graphik mit der Schildkröte

Die nach dem Bewegen der Schildkröte auf dem Bildschirm erzeugte Graphikdarstellung. Die Schildkröte zeichnet ihre Spur auf dem Bildschirm und erzeugt damit Zeichnungen.

Graphikmodus (Graphics mode)

Frühere Microcomputer mußten in den jeweiligen Modus zur Verarbeitung von entweder Text oder Graphik umgeschaltet werden. Die modernen Personal Computer können beide Funktionen gleichzeitig behandeln.

Graphiktablett (Graphics tablett)

Ein Eingabegerät, das die Koordinatenpunkte eines Bildes oder einer Zeichnung zur Verarbeitung an den Computer weitergibt. Eine Art Analog/Digitalwandler.

Graphisches Zeichnen (Graphics charakter)

Besondere Zeichen oder Formen, die für die Erstellung von Bildern sehr nützlich sind. Der CPC664 hat einen vollständigen Satz davon, dieser ist in Anlage III beschrieben.

Halbbyte (Nibble)

Ein vier Bit großer binärer Ausdruck. Jede der hexadezimalen Ziffern im Ausdruck &F6 stellt ein Halbbyte dar.

Hard Copy

Papierausdruck von einem Programm oder sonstigen Texten - oder von Graphiken. Der vorübergehende Ausdruck am Bildschirm heißt Softcopy.

Hardware

Die elektronischen und mechanischen Teile eines Computersystems, alles das weder Software noch Firmware ist.

Hexadezimaldarstellung (hexadecimal notation)

Zahlen zur Basis 16, siehe Anhang II. Im CPC664 durch ein vorgestelltes & bzw. &H gekennzeichnet.

Hex-File

ASCII-Darstellung einer Befehls- oder Maschinenkode-Datei.

Höchstwertiges Bit (MSB - Most Significant Bit)

Das in einer Binärzahl am weitesten links stehende Bit.

Höhere Programmiersprache (Higher level language)

Programmiersprache, die der normalen Sprechweise ziemlich nahe kommt, so daß sie sehr leicht interpretiert werden kann. Langsamer als Programme in Maschinencode, jedoch dafür viel einfacher zu verstehen, wie z.B. BASIC.

IEEE-488

Eine der Standardschnittstellen für den Anschluß von Geräten an einen Microcomputer. Ähnlich der Centronics Parallelschnittstelle, jedoch nicht völlig kompatibel.

Informationstechnik (Information technology)

Alles was mit der Anwendung von Elektronik bei Nachrichtenverarbeitung und Kommunikationen zu tun hat: Textverarbeitung, Datenfernverarbeitung, PRESTEL, usw.

Initialisieren (Initialise)

Ein System einschalten, oder Vorbelegung von Variablen vor dem eigentlichen Hauptprogramm - z.B. Variablen als Ganzzahlen zu deklarieren.

Integer

Ganzzahliger Wert ohne Dezimalpunkt.

Integrierte Schaltung (Integrated circuit)

Eine Sammlung winzig kleiner elektronischer Schaltungseinheiten auf einem einzigen Siliziumplättchen. Siehe auch 'Chip'.

Intelligentes Datensichtgerät (Intelligent terminal)

Ein Datensichtgerät bei dem außer der Behandlung von Eingabe und Ausgabe für den Computer auch eine lokale Bearbeitungsmöglichkeit unterstützt wird, auch wenn das Gerät 'offline' ist.

Interaktiv (Interactive)

Dialog zwischen Anwender und Programm, wobei der Anwender aufgefordert wird, etwas einzugeben. Es kann z.B. die Steuerung eines Raumschiffs in einem Automatenspiel als auch das Beantworten von Fragen eines Schulungsprogramms sein. Die Aktion des Anwenders beeinflußt sofort den weiteren Ablauf des Programms.

Interface

Schnittstelle, die zwei unabhängigen Systemen, wie beispielsweise dem CPC664 und einem Disketten-Laufwerk, die gegenseitige Kommunikation ermöglicht.

Interpreter

Eine Erweiterung zur Analogie von Computer Befehlsvorrat und Sprache. Der Interpreter ist eine Systemsoftware, die eine höhere Programmiersprache in die Maschinensprache umwandelt, z.B. konvertiert er BASIC in die interne Sprache des Computers.

Iteration

Eines der Elemente der Datenverarbeitung. Ein Computer führt alle Funktionen durch, indem er sie in einfache kleine Schritte aufteilt, die von der CPU verstanden werden. Der Computer muß dazu zwischen vielen kleinen Schritten hin- und herwechseln, bis eine gewünschte Bedingung erfüllt ist.

K

Die Abkürzung für 'kilo' - 1000. In der Datenverarbeitung aber ist es auch die Abkürzung für Kilobyte - mit einem Wert von 1024, der aus der Sicht des Binärsystems 2 hoch 10 entspricht. Siehe Anhang II.

Kaltstart

Booten und Initialisieren eines Betriebssystems. Der ICPM-Befehl bewirkt beispielsweise den Kaltstart von CP/M.

Knoten

Speichereinheit im LOGO-Arbeitsbereich, der 4 Bytes in Anspruch nimmt.

Kommando (Command)

Eine programmierte Anweisung.

Konsolenmodus

Direkter CP/M-Modus. Auf dem Bildschirm erscheint A>. Das System erwartet daraufhin die Eingabe eines CP/M- oder Dienstprogramm-Befehls.

Kopplung (Handshaking)

Eine Folge von elektronischen Signalen, die den Austausch von Daten zwischen einem Computer und einem peripheren Gerät bzw. zwei Computern anstoßen, prüfen und synchronisieren.

Korruption

Die unbeabsichtigte und irreversible Zerstörung bzw. Veränderung des Inhalts einer Disketten-Datei oder eines Speichers.

Künstliche Intelligenz (Artificial Intelligence)

Eine Struktur innerhalb des Programmaufbaus, die es dem Programm ermöglicht, aus vorheriger Erfahrung zu lernen.

Laderoutine (Booting or Bootstrapping)

Programme und Betriebssysteme werden nicht von selbst geladen, sondern mit einem Ladeprogramm, einer kleinen Routine meist im ROM, die den Ladevorgang auf eine bestimmte Speicheradresse ausstößt.

Leistungstest (Benchmark)

Eine Standardaufgabe für die Ermittlung von Geschwindigkeit, Leistungsfähigkeit und Genauigkeit verschiedener Computer, z.B. die Wurzel aus 99.999 zum Quadrat.

Lichtstift (Light pen)

Eine andere Eingabemöglichkeit, bei der ein Stift oder ein 'Zauberstab' benutzt wird.

Lisp

Die Abkürzung für LIST Processor Language, eine weitere höhere Programmiersprache.

LIST

Sie können jederzeit mit Hilfe der Anweisung LIST das im Speicher befindliche Programm auf dem Bildschirm anzeigen.

Logik (Logic)

Die elektronischen Komponenten, welche die elementaren logischen Operationen und Funktionen ausführen, aus denen jede Operation eines Computers letztendlich aufgebaut ist.

Logisches Gerät

Darstellung eines Geräts, das sich von der physikalischen Erscheinungsform desselben unterscheiden kann. So kann beispielsweise das logische CP/M-Gerät LIST dem physikalischen Gerät Centronics-Kanal oder VDU zugewiesen werden.

Logo

Eine einfach erlernbare graphikorientierte höhere Programmiersprache, die in Schulen oft im Programmierunterricht verwendet wird.

LSI

Hochgradige Integration (Large Scale Integration). Die Entwicklung von integrierten Schaltungen, bei der immer mehr Funktionen auf immer kleiner werdenden Stückchen Silizium untergebracht werden.

Maschinell lesbar (Maschine readable)

Daten in einer Form, die ohne zusätzliche Eingabe über z.B. die Tastatur in den Computer eingegeben werden können.

Maschinencode (Machine Code)

Die Programmiersprache, die der Microprozessor direkt versteht, da alle ihre Kommandos als Bitmuster dargestellt werden.

Matrix (Matrix)

Die Anordnung von Punkten, welche den Bereich eines Schriftzeichens auf dem Bildschirm oder dem Druckkopf eines Punktmatrixdruckers formt. Wird auch in der Mathematik und Informatik zur Beschreibung von Feldern verwendet.

Maus (Mouse)

Ein auf Rollen gleitendes Eingabegerät. Die Maus wird auf dem Tisch hin und her bewegt, um meist den Cursor auf dem Bildschirm zu bewegen. Ursprünglich wurde die Maus dazu entwickelt, um die Angst vor Tastaturen zu nehmen und Software anwenderfreundlicher zu gestalten.

Mensch-Maschinen-Schnittstelle (Man-machine interface)

Der Berührungspunkt zwischen Computer und Anwender: Tastatur, Bildschirm, usw.

Menü (Menü)

Eine Auswahlliste der verschiedenen Funktionen, die ein Programm ausführen kann.

Microprozessor (Microprozessor)

Eine integrierte Schaltung im Herzen eines Microcomputers, welche die Maschinencode-Befehle ausführt, die vom BASIC-Interpreter angeboten werden, um die verschiedenen Ausgabegeräte und sonstigen Möglichkeiten zu kontrollieren.

Modem

Ein MOdulator/DEModulator, der den Anschluß eines Computers an eine Telefonleitung oder sonstige serielle Datenübertragungseinrichtung, einschließlich Glasfaserkabel ermöglicht. Siehe auch Akkustikkoppler.

Monitor

Der Bildschirm eines Computerterminals als auch ein Maschinensprachenprogramm, das Zugriffe zu fundamentalen Maschinensprachenoperationen des Computers ermöglicht.

Netz (Network)

Die Verbindung von zwei oder mehr Computern, entweder verdrahtet oder über Modem, um Daten und Informationen untereinander auszutauschen.

Niedere Programmiersprache (Low level language)

Z.B. die Assembler Sprache. Eine Programmiersprache, bei der jede Anweisung mit einer Maschinencode-Anweisung korrespondiert.

Niederwertigstes Bit (Least significant bit)

In einer Binärzahl das am weitesten rechts stehende Bit (siehe Anhang II).

Node

Eine Speichereinheit innerhalb des LOGO-Arbeitsbereiches. Normalerweise nimmt ein Node vier Bytes Speicherraum ein.

Numerisches Tastenfeld (Numeric keypad)

Ein Teil der Tastatur, in dem Zahlentasten zusammengruppiert sind, um numerische Daten einfacher eintippen zu können. Beim CPC 664 können diese Tasten zusätzlich auch als Funktionstasten definiert werden.

OCR

Optical Character Recognition - optische Zeichenerklärung. Ein System, bei dem gedruckte oder handgeschriebene Zeichen durch ein optisches Lesegerät erfaßt und direkt in Daten umgewandelt werden, die für den Computer lesbar sind.

Oktal (Octal)

Das Zahlensystem zur Basis 8, in dem jede Zahl (0-7) mit drei Bits dargestellt wird.

Off line

Ein peripheres Gerät - normalerweise ein Datensichtgerät oder ein Drucker - das nicht betriebsbereit an die Hauptprozessoreinheit angeschlossen ist, auf das also nicht zugegriffen werden kann.

On line

Das Gegenteil von Off line.

Operator

Der Teil eines arithmetischen Ausdrucks, der eine Zahl mit einer anderen verknüpft, wie z.B. + , - usw.

Paddle

Ein anderer Name für Joystick (Steuerknüppel). Auch als 'game paddle' bekannt.

Page Zero

Bezieht sich auf einen Speicherbereich in der CP/M-Umgebung, der sich im Adressbereich zwischen 0000Hex und 0100Hex befindet und wichtige System-Parameter enthält.

Paperware

Unter Paperware wird das gedruckte Begleitmaterial zu einem Produkt (z.B. dieses Handbuch) zusammengefaßt.

Parallele Schnittstelle (Parallel interface)

Die CPC664 Druckerschnittstelle unterstützt einen parallelen Drucker, d.h. jeder Datenausgang des Bus ist mit einem entsprechenden Dateneingang des Druckers verbunden. Daten werden über eine parallele Schnittstelle schneller ausgetauscht als über eine serielle Schnittstelle, da bei serieller Verarbeitung die Daten erst in eine Bytedarstellung mit Synchronisierungsinformation umgeformt werden.

Pascal

Eine strukturierte höhere Programmiersprache, die compiliert werden muß, bevor ein Programm ablaufen kann, aber dann sehr schnell abläuft. Sehr oft die nächste Sprache, die ein eifriger BASIC-Schüler erlernt.

PEEK

Die BASIC-Funktion, die es ermöglicht, den Wert einer bestimmten Stelle des Computerspeichers zu lesen.

Peripheres Gerät (Peripheral)

Drucker, Modems, Joysticks, Plattengeräte - alles was an einen Computer angeschlossen werden kann, um seine Fähigkeiten zu erweitern.

Physikalisches Gerät

Ein reales, existierendes Hardware-Gerät. Physikalische Geräte können durch ein logisches Gerät repräsentiert werden.

Pixel

Der kleinste Bereich des Bildschirms auf den durch die Hardware bedingt zugegriffen werden kann.

Platte (Disk)

Eine flache, dünne, runde Scheibe aus Kunststoff, die auf einer oder beiden Seiten mit einer magnetisierbaren Beschichtung versehen ist und zur Speicherung von Daten verwendet wird. Die Magnetplatte ist immer in einem Schutzgehäuse untergebracht, das teilweise mit einer Öffnung für den Schreib-/Lesekopf versehen ist. Siehe auch Floppy Disk und Winchester.

Plattengerät (Disk drive)

Die Einheit, die Informationen auf die magnetisierbare Oberfläche der sich drehenden Platte schreibt oder dort vorhandene Informationen wieder liest.

Plotter

Ein Ausgabegerät zum Erstellen von Zeichnungen, das Schreibstifte und nicht einen Druckkopf wie ein Drucker benutzt.

POKE

Die Anweisung in BASIC, womit ein Wert an eine bestimmte Stelle des Speichers geschrieben werden kann. Siehe Kapitel 8.

Port

Eine besonders adressierbare Adresse auf einer Schnittstelle (Interface) für die Eingabe oder Ausgabe von Daten.

Portabilität (Portability)

Die Fähigkeit ein und dieselbe Software auf Computern verschiedener Hersteller ablaufen zu lassen - meist durch kompatible Betriebssysteme erreichbar, wie CP/M von Digital Research.

Primitive

Bereitgestellte Prozeduren, Funktionen oder Befehle unter Dr. LOGO.

Programm

Eine Kombination von Befehlen, die den Computer veranlassen, eine Aufgabe zu erfüllen. Hierbei kann es sich um eine einfache Maschinencode-'Routine' oder um komplette Anwendungsprogramme, wie z.B. Textverarbeitung handeln.

Programmentwicklung (Software engineering)

Ein Ausdruck für Programmierung mittels Strukturierung und Planung statt mit willkürlichen Techniken.

Programmfehler (Bug)

Ein Problem, das von einem 'unerwarteten Merkmal' unter Einschluß einiger unerklärbarer Aspekte des Programmeinsatzes (wenn Sie beispielsweise vier Tasten gleichzeitig drücken, ändert der Bildschirm seine Farbe) bis zu einer Folge reicht, die komplett und nicht umkehrbar ein Computerprogramm stört und sämtliche Daten im Speicher löscht.

Programmiersprache (Programming language)

Programmfolge unter strikter Einhaltung von Regeln für den Einsatz von Wörtern, Zahlen sowie die einbezogenen Folgen.

PROM (Programmable Read Only Memory)

Programmierbarer, nur lesbarer Speicher. Eine integrierte Speicherschaltung, die, einmal beschrieben, nicht mehr gelöscht werden kann. (Siehe auch EPROM).

Prompt

Kurze Meldung oder Zeichenfolge, die den Anwender zur Eingabe einer Anweisung auffordert. Das CP/M prompt ist beispielsweise >, das Dr. Logo-prompt ein ?.

Prozedur

Eine Reihe von Ausdrücken oder Programm-Befehlen, die eine bestimmte Aufgabe bewältigen.

Puffer (Buffer)

Ein (Zwischen-) Speicher, der vorübergehend Daten aufnimmt, die von einer Funktionseinheit zu einer anderen übertragen werden; z.B. vom Cassettenrecorder zur Zentralprozessoreinheit (CPU) und zum Haupt-RAM. Der Puffer reguliert den Datenfluß zwischen Einheiten, die mit verschiedenen Geschwindigkeiten arbeiten, wie z.B. einem Modem oder einem Drucker.

Punktmatrix (Dot matrix)

Ein rechteckiges Punktraster, auf dem ein Zeichen durch Auswahl einzelner Punkte dargestellt werden kann.

RAM

Random Access Memory. Eine Speichereinheit, die mit der internen Schaltung des Computers während einer normalen Programmausführung sowohl gelesen als auch beschrieben werden kann.

Raster

Eine Art des 'Schreibens' auf dem Bildschirm, wobei die Bilder aus einer Anzahl waagerechter Abtastlinien aufgebaut werden.

Read only R/O

Ein der Diskette, einer Disketten-Datei oder einem Disketten-Laufwerk zugeordnetes Attribut, welches das Schreiben oder Verändern von Daten verhindert.

Read write R/W

Ein der Diskette, einer Disketten-Datei oder einem Disketten-Laufwerk zugeordnetes Attribut, welches das Lesen und Schreiben von Daten ermöglicht.

Record

Eine Gruppe Bytes in einer Datei. CP/M benötigt 128 Byte-Records.

Reelle Zahl (Real number)

Eine Zahl mit ganzzahligem und Bruchteil, also mit Werten zu beiden Seiten des Dezimalpunktes. Eine Variable kann trotzdem eine reelle Zahl sein, obwohl sie manchmal im Programm nur den ganzzahligen Teil besitzt.

Register

Eine Speichereinheit in der CPU, die dazu benutzt wird, Daten kurzzeitig zwischenzuspeichern.

Rekursion

Eine Serie von wiederholten Schritten innerhalb eines Programms oder einer Routine, deren Ergebnisse bei jeder Wiederholung in Verbindung zu denen des vorangegangenen Durchlaufs stehen.

Reserviertes Wort (Reserved word)

Ein Wort mit besonderer Bedeutung in der Programmiersprache, das nicht anders als vordefiniert verwendet werden kann. Z.B. wird BASIC das Wort NEW nicht als Variablenname annehmen, da es schon für einen anderen Zweck reserviert ist.

RF Modulator

Die Art, in der Videosignale des Computers umgeschlüsselt werden, so daß sie über den Antenneneingang eines Fernsehers 'gesendet' werden können.

ROM

Nur lesbare Speicher (Read Only Memory). Halbleiterspeicher, der einmal beschrieben, weder gelöscht noch überschrieben werden kann.

RPN

Eine umgekehrte polnische Notation (Reverse Polish notation). Eine Methode, die manche Hersteller bei der Errechnung von arithmetischen Ausdrücken in ihren Computern verwenden, wobei die Operatoren hinter den Werten angeordnet werden, zu denen sie gehören (+,-,*,/).

RS232C

Ein spezieller Standard für serielle Datenübertragungsschnittstellen. Geräte an beiden Enden der Übertragungsstrecke müssen an die spezifischen Bedingungen der benutzten RS232-Schnittstelle angepaßt sein. Auch die Centronics-(parallel)Schnittstelle ist ja engen Regeln unterworfen.

Schildkröte (Turtle)

Graphiksymbol, ähnlich einem Aufwärtspfeil, das bei Dr. LOGO als Graphik-Cursor dient.

Schildkröten-Schritt (Turtle step)

Kürzeste Entfernung, die eine Schildkröte zurücklegen kann. In der Regel ist dies ein Pixel.

Schleife (Loop)

Ein Prozess im Programm, der sooft wiederholt wird, bis eine bestimmte Bedingung erfüllt ist.

Schlüsselfertig (Turn key)

Begriff, der den Automatik-Start eines Programms nach dem System- bezeichnet. Die Dr. LOGO-Diskette ist beispielsweise ein schlüsselfertiges Programm.

Schlüsselwort (Keyword)

Ein Wort, dessen Anwendung in einem Computerprogramm oder einer Sprache für eine spezifische Funktion oder eines Kommandos reserviert ist.

Schnittstelle (Interface)

Der Ein- und Ausgang eines Computers sowohl in elektrischer als auch im menschlicher Hinsicht. Als Schnittstelle am CPC664 gelten die Tastatur (Eingabe), der Bildschirm (Ausgabe) genauso wie der Schnittstellenbus auf der Computerrückseite für periphere Geräte.

Schreibmaschinentastatur

Begriff für eine Computertastatur, die in ihrer Tastenanordnung wie eine Schreibmaschine aufgebaut ist.

Schreibschutz

Eine Sicherheitsmaßnahme zum Schutz vor dem Beschreiben einer Diskette oder Disketten-Datei. Eine schreibgeschützte Diskette oder Datei ist nur lesbar.

Scrolling

Die Art, wie die Zeilen des Bildschirms nach oben 'gerollt' werden, um Platz in der untersten Zeile für neue Ein- oder Ausgaben zu machen.

Sektor

Datenblock auf einer Diskette. Die Sektorgröße des Schneider Diskettensystems beträgt 512 Bytes.

Serielle Schnittstelle (Serial interface)

Dieser Begriff bezieht sich fast immer auf eine RS232-Schnittstelle, obwohl noch andere serielle Schnittstellen für sequentielle Datenübertragung existieren.

Simulation

Eine Technik, lebensnahe Aktionen mit Hilfe des Computers nachzuahmen, wie z.B. Flug- und Fahrsimulation usw.

Single sided

Bezeichnet eine Diskette, die nur eine Seite zur Datenspeicherung besitzt.

Soft Key

Siehe Benutzerdefinierte Taste (User defined key-UDK)

Software

Programme

Speicher (Memory)

Der Bereich eines Computers, in dem Informationen und Daten geordnet zwischengelagert werden, so daß auf alles wieder zugegriffen werden kann; bekannt entweder als RAM (random access memory - wahlfreier Zugriffsspeicher), der sowohl beschrieben als auch gelesen werden kann, oder als ROM (read only memory - nur lesbarer Speicher), der nur gelesen aber nicht beschrieben werden kann. Magnetplatten und -bänder sind Beispiele für 'Massenspeicher', wobei dieser Begriff heute öfter für den Speicherbereich verwendet wird, der direkt von der CPU angesprochen wird.

Speicherbelegungsplan (Memory map)

Die Beschreibung der Speichernutzung mit den jeweiligen Adressen und der Speicherzuordnung für die verschiedenen Funktionen, wie z.B. für den Bildschirm, das Magnetbandbehandlungssystem usw.

Spracherkennung (Speech recognition)

Die Umwandlung gesprochener Wörter in Anweisungen, die der Rechner erkennen kann.

Sprachsynthese (Speech synthesis)

Die Erzeugung von simulierter Sprache mittels Hard- und Software.

Sprite

Ein Zeichen, von spezieller Hard- oder Software erzeugt, das sich frei auf dem Bildschirm bewegt und zufällig erscheinen oder verschwinden kann.

Spuren

Spuren sind konzentrische Ringe auf einer Diskette. Jede Spur besteht aus mehreren Sektoren. Beim Formatieren wird ein bestimmter Bereich der Spuren und Sektoren beschrieben.

Stapel (Stack)

Ein Bereich im Speicher zum Ablegen von Informationen, woraus aber nur der jeweils letztgeschriebene Eintrag wieder geholt werden kann.

Steckmodul (Cartridge)

Eine besonders verpackte Speichereinheit mit integrierten Schaltungen, die Software enthält und durch dafür eingebaute Stecksockel direkt an den Computer angeschlossen werden kann. Diese Software wird schneller geladen und läuft auch schneller ab als Software von Cassetten, ist jedoch wesentlich teuerer.

Steuerknüppel (Joystick)

Ein Eingabegerät, das die Funktionen der Cursorsteuertasten ersetzt, um Spiele schneller zu machen.

Strichcode (Bar code)

Beispiele finden Sie auf vielen Lebensmittelverpackungen. Ein Code aus Linien unterschiedlicher Stärke, die mit einer Lesevorrichtung (z.B. Schwachstromleser) erfaßt und ausgewertet werden.

Strukturierte Programmierung (Structured programming)

Eine logische und vorbedachte Art der Programmierung, deren Ergebnis Programme sind, die mit klar definierten Schritten von 'oben nach unten' ablaufen.

Syntaxfehler (Syntax error)

Wenn der Ablauf eines Programms durch eine falsche Benutzung von Schlüsselwörtern und Variablen abgebrochen wird, gibt BASIC diese Meldung für den Anwender aus. Siehe Anhang VIII.

Systemspuren

Für das CP/M-System reservierte Disketten-Spuren.

Tabellenkalkulation (Spreadsheet)

Ein Programm, das in Zeilen und Spalten eingetragene Zahlen arithmetisch behandeln kann. Wenn ein Eintrag geändert wird, werden alle damit verbundenen Kalkulationen neu errechnet und ein neues Ergebnis errechnet.

Tastatur (Keyboard)

Eine Zusammenstellung von alphanumerischen Tastenschaltern, um Kommandos und andere Informationen in den Computer eingeben zu können.

Teilprogramm (Routine)

Ein Teil des Programms, in dem ein zusammengehörender Teil der Programmlogik bearbeitet wird. Ein solches Teilprogramm kann im Hauptprogramm untergebracht oder auch ein eigenes Unterprogramm sein, das in verschiedenen anderen Programmen mitverwendet werden kann. Z.B. kann man ein Programm, in dem die Uhrzeit aus der Systemuhr in eine normale 12-Stundenuhr aufbereitet wird, als Teilprogramm gestalten.

Terminal

Ein peripheres Gerät, bestehend aus einer Tastatur und entweder einem Bildschirm oder einem Fernschreiber als Ausgabesystem.

Textfolge (String)

Die Arten von Daten, die aus einer Folge von Zeichen bestehen und nicht als numerische Werte betrachtet werden. Auch wenn sie aus nur numerischen Zeichen bestehen, so werden sie nicht als numerisch betrachtet, solange sie nicht durch ein entsprechendes Umwandlungskommando einer numerischen Variablen zugewiesen worden sind.

Tongenerator (Sound generator)

Der Teil des Computers (es kann sowohl Hard- als auch Software sein), der Töne und Geräusche erzeugt.

TPA

Transient Program Area. Speicherbereich ab Adresse 0100Hex, in dem CP/M Anwender-Programme und -Daten gespeichert sind.

Transientes Programm

Ein CP/M-Dienstprogramm (z.B. FILECOPY), das in den TPA geladen und durch Eingabe seines Namens gestartet werden kann.

Trennzeichen (Separator)

Ein Trennzeichen hat die gleiche Wirkung wie ein Begrenzer (Delimiter): Die Grenze zwischen reservierten Wörtern und sonstigen Elementen des Programms oder der Daten wird hiermit markiert.

Typenrad-Drucker (Daisy-wheel printer)

Ein Drucker mit hoher Schreibqualität. Zeichen werden durch Anschlag eines Buchstabens auf einem Rädchen gegen ein Farbband erzeugt.

Überschreiben (Overwrite)

Löschen eines Bereichs im Speicher durch Überschreiben seines Inhalts mit neuen Daten.

Uhr (Clock)

Ein Zeitsystem im Computer als Basis für den Gleichlauf der internen Funktionen. Eine Echtzeituhr ist eine, die Stunde, Datum usw. verwaltet.

Umgekehrte polnische Notation (Reverse Polish notation - RPN)

Eine Methode, die manche Hersteller bei der Errechnung von arithmetischen Ausdrücken in ihren Computern verwenden, wobei die Operatoren hinter den Werten angeordnet werden, zu denen sie gehören.

Unintelligentes Datensichtgerät (Dumb terminal)

Ein Datensichtgerät, das nur als Mittel zur Dateneingabe bzw. -ausgabe dient, ohne die Daten irgendwie zu bearbeiten. Es gibt noch einfachere Sichtgeräte, die nicht einmal einen Generator haben, so daß alle Abbildungen am Bildschirm reine Videodarstellungen sind.

Unterprogramm (Subroutine)

Siehe Teilprogramm (Routine)

Variable

Ein Datenfeld in einem Programm, das mit einem Namen angesprochen werden und dessen Wert während des Programmablaufs verändert werden kann.

Vorzeichenlose Zahl (Unsigned number)

Eine Zahl ohne ein vorangestelltes Kennzeichen, das aussagt ob die Zahl positiv oder negativ ist.

Wahrheitstabelle (Truth table)

Das Ergebnis einer logischen Operation ist entweder 'wahr' oder 'falsch'. Der Computer stellt dies entweder mit 1 oder 0 dar. Eine Liste aller möglichen Ergebnisse einer logischen Operation (z.B. IF A> B THEN C) heißt Wahrheitstabelle.

Warmstart

Ein Warmstart wird unter CP/M durch Betätigen der [CTRL]C-Taste ausgelöst. Er reinitialisiert das Disketten-Subsystem und übergibt die Steuerung an CP/M, woraufhin Befehle eingegeben werden können.

Wiederauffrischen (Refresh)

Das Aktualisieren von Daten auf einem Bildschirm oder im Speicher. Die schon vorhandenen Daten müssen dabei nicht unbedingt zerstört werden, sondern können auch nur wieder aufgefrischt werden.

Wildcard-Zeichen (Wildcard character)

Eines der Zeichen * oder **?**. Dr. LOGO unterstützt nur das Zeichen **?**. Das Wildcard-Zeichen * kann eine beliebige Anzahl **?** ersetzen. Ein 'Ambiguous File Name' ist der Name einer Datei, auf die über Wildcard-Zeichen Bezug genommen wird. Jedes in einem Dateinamen erscheinende **?** steht für ein numerisches oder alphabetisches Zeichen.

Zeichen (Character)

Ein Buchstabe, eine Nummer oder ein graphisches Symbol, das im Computer dargestellt und ausgegeben werden kann (siehe Anhang III).

Zeichenkette (Character String)

Eine Folge von Zeichen, die als Einheit gespeichert oder behandelt werden können, z.B. ein Wort oder eine Kette von Worten.

Zeichenvorrat (Character set)

Alle Buchstaben, Nummern und graphische Symbole, die in einem Computer oder auf einem Drucker zur Verfügung stehen. Daß ein Zeichen im Computer zur Verfügung steht, bedeutet nicht, daß es auch auf einem Drucker vorhanden ist.

Zeichenzelle (Character cell)

Die Punktematrix auf dem Bildschirm, in der ein Zeichen durch unterschiedliche Farbgebung der einzelnen Punkte sichtbar gemacht wird.

Zeilennummer (Line number)

BASIC und einige andere Sprachen benutzen die Zeilennummer als Ordnungskriterium innerhalb des Programms.

Zufallszahl (Random number)

Eine vom Computer generierte Zahl, die weder wiederholbar noch voraussagbar ist.

Anhang 3

Einige Programme für Sie ...

Tennis

Dieser 'Urvater' aller Computer-Spiele macht auch heute noch großen Spaß. Sie können alleine gegen den Computer oder zu zweit spielen. Wählen Sie außerdem noch zwischen der Tastatur oder einem Joystick für die Bedienung.

```
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 14,1:PRINT USING"#### ";score1;
130 LOCATE 34,1: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 22,1:PR
INT "Spieler 2 :    0";
```

Fortsetzung nächste Seite

```
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$=" " THEN xb=xb+dx:yb=yb+dy:RETURN
530 IF ch$="¦" 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)"
---------"
```

Fortsetzung nächste Seite

```
720 PEN 3:PRINT:PRINT TAB(15)"Steuerung"
730 PRINT:PRINT:PEN 1
740 PRINT" Spieler 1        Spieler 2     Richtung!":PRI
NT
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> or <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. Tastatur oder Joysticks

```
10 ' ELEKTRO ZAUN 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 9
280 PRINT:PRINT "   Kontrolltasten"
290 PRINT:PRINT:PRINT:PRINT"Eingabe: J/T [ENTER]"
300 LOCATE 4,10:PRINT"Joystick";TAB(4);"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
```

Fortsetzung nächste Seite

```
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:FR
AME
680 PAPER #4,0:CLS #4
690 GOTO 570
700 '
710 '
720 PAPER #6,5:WINDOW #6,4,17,p2wp,p2wp:CLS #6:FRAME:FR
AME
730 PAPER #6,0:CLS#6
740 GOTO 620
750 '
760 '
```

Fortsetzung nächste Seite

```
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$(p1sc);
:IF p1sc=9 THEN 860
780 IF p2sa AND NOT(p1sa) AND pwpe THEN p2sc=p2sc+1:SOU
ND 132,100,10,0,1,0:PRINT #2,a$(p2sc);:IF p2sc=9 THEN 8
60
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 9
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$(32)
1120 NEXT n2,n
1130 '
1140 '
1150 b$=" ELEKTRO ZAUN"
```

Fortsetzung nächste Seite

```
1160 c$=CHR$(32)+CHR$(164)+" Alexander Martin"
1170 ENV 1,=9,2000:ENT -1,6,3,1
1180 ENV 2,127,0,0,127,0,0,127,0,0,127,0,0,127,0,0
1190 ENV 3,=9,9000
1200 '
1210 '
1220 BORDER 0
1230 INK 0,12:PEN #4,1:PEN #6,2:PEN #1,1:PEN #2,2:PAPER
 #1,3:PAPER #2,3:PAPER #0,6
1240 RETURN 'Damit wurden die Konstanten gesetzt
1250 '
1260 '
1270 INK 0,12:INK 1,2:INK 2,6:INK 3,13:INK 4,20:INK 5,1
7:INK 6,20
1280 WINDOW #3,3,3,6,25:WINDOW #5,18,18,6,25
1290 WINDOW #1,3,5,1,5:WINDOW #2,16,18,1,5:WINDOW #7,1,
20,1,5:PAPER #7,3
1300 CLS:CLS#7:PRINT#1,a$(0);:PRINT#2,a$(0);:p1sc=0:p2s
c=0:p1wp=5:p2wp=24:p1dir=1:p2dir=1
1310 GOSUB 570:GOSUB 620
1320 SOUND 1,1000,0,12,2:SOUND 2,900,0,12,2
1330 p1sa=0:p2sa=0:finished=0
1335 PEN 9
1340 RETURN 'Bild wieder erstellt
1350 '
1360 '
1370 CLS
1380 PEN 7
1390 FOR n=1 TO LEN(b$)
1400 LOCATE 2+n,10
1410 FOR n2=LEN(b$) TO n STEP-1
1420 PRINT MID$(b$,n2,1)
1430 LOCATE 2+n,10
1440 SOUND 135,100,0,13,3,1,20
1450 NEXT n2,n
1460 SOUND 135,100,0,13,3,1,20
1470 PEN 9:PRINT:PRINT:PRINT:PRINT c$
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"
```

Fortsetzung nächste Seite

```
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
```

Unser '17 und 4' oder auch 'Pontoon' genannt

Lassen Sie Ihrer Leidenschaft freien Lauf! Für einen Spieler gegen den Computer. Nur Tastatur.

```
10 'PONTOON
20 'copyright (c) AMSOFT 1984
30 '
40 INK 0,21:INK 1,0:INK 3,6
50 ENV 1,7,2,1
60 MODE 1
70 yc=2:cc=2
80 asse=0
90 casse=0
100 lasse=0
110 s=0
120 t=0
130 DIM farbe$(4)
140 farbe$(1)=CHR$(226)
150 farbe$(2)=CHR$(229)
160 farbe$(3)=CHR$(228)
170 farbe$(4)=CHR$(227)
180 CLS
190 DIM stapel(52)
200 FOR x=1 TO 52
210 stapel(x)=0
220 NEXT x
230 '
240 ' Jedem Spieler zwei Karten geben
250 '
260 LOCATE 8,3
270 PRINT "SPIELER";SPC(14)"BANK"
280 LOCATE 7,5
290 GOSUB 930
300 s=s+f
310 IF f=11 THEN asse=asse+1
320 LOCATE 27,5
330 GOSUB 930
340 t=t+f
350 IF f=11 THEN casse=casse+1
360 LOCATE 7,6
370 GOSUB 930
380 s=s+f
390 IF s=21 THEN 850
400 IF s=11 THEN asse=asse+1
410 IF s=22 THEN asse=asse-1
```

Fortsetzung nächste Seite

```
420 IF s=22 THEN s=12
430 '
440 'Eingabe: Karten geben oder auflegen
450 '
460 LOCATE 8,12: PRINT "GEBEN (g) oder AUFLEGEN (a)":FO
R z=1 TO 1000:NEXT:LOCATE 8,12:PRINT SPC(27):FOR z=1 TO
 500:NEXT
470 x$=LOWER$(INKEY$):IF x$<>"a" AND x$<>"g" THEN 460
480 IF x$="a" THEN LOCATE 10,12:PRINT:GOTO 680
490 LOCATE 7,yc+5
500 yc=yc+1
510 GOSUB 930
520 s=s+f
530 IF f=11 THEN asse=asse+1
540 '
550 'Punkte und Asse ueberpruefen
560 '
570 IF s<22 THEN 460
580 IF asse=0 THEN 620
590 asse=asse-1
600 s=s-10
610 GOTO 570
620 LOCATE 12,19
630 PRINT "SIE HABEN VERLOREN"
640 PRINT:LOCATE 11,21:PRINT "EIN NEUES SPIEL (J/N)"
650 x$=LOWER$(INKEY$):IF x$<>"j" AND x$<>"n" THEN 650
660 IF x$="j" THEN RUN
670 END
680 LOCATE 27,6
690 GOSUB 930
700 t=t+f
710 IF f=11 THEN casse=casse+1
720 IF t=17 OR t=18 OR t=19 OR t=20 OR t=21 THEN GOTO 8
80
730 cc=cc+1
740 LOCATE 27,cc+4
750 GOSUB 930
760 t=t+f
770 IF f=11 THEN casse=casse+1
780 IF t>6 AND t<12 AND lasse>0 THEN t=t+10:GOTO 880
790 IF t<21 THEN 720
800 IF t=21 THEN 880
810 IF casse=0 THEN 850
820 casse=casse-1:lasse=lasse+1
830 t=t-10
```

Fortsetzung nächste Seite

```
840 GOTO 790
850 LOCATE 12,19
860 PRINT"SIE HABEN GEWONNEN"
870 GOTO 640
880 LOCATE 11,19
890 IF t<s THEN 850
900 PRINT "DIE BANK HAT GEWONNEN"
910 GOTO 640
920 '
930 'KARTE GEBEN
940 '
950 karte=INT(RND(1)*52+1)
960 IF stapel(karte)=1 THEN GOTO 950
970 stapel(karte)=1
980 f=karte-13*INT(karte/13)
990 IF f=0 THEN f=13
1000 IF f=1 OR f>10 THEN GOTO 1090
1010 FOR z=1 TO 1000:z=z+1:NEXT
1020 IF f=10 THEN PRINT f;" ";
1030 IF f<10 THEN PRINT " "f" ";
1040 IF f>10 THEN f=10
1050 IF karte>26 THEN PEN 3 ELSE PEN 1
1060 SOUND 1,0,0,0,1,,1:PRINT farbe$(INT((karte-1)/13)+
1)
1070 PEN 1
1080 RETURN
1090 FOR z=1 TO 1000:z=z+1:NEXT
1100 IF f=11 THEN PRINT "  J  ";
1110 IF f=12 THEN PRINT "  Q  ";
1120 IF f=13 THEN PRINT "  K  ";
1130 IF f<>1 THEN GOTO 1040
1140 f=11
1150 PRINT "  A  ";
1160 GOTO 1060
```

Bomber

Eine Variation eines bekannten Themas!
Für einen Spieler, gegen den Computer. Tastatur oder Joystick

```
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 10,10:I
NK 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$(242):x=2:y=2
:drop=0:a=2:b=2
70 GOSUB 480
80 CLS
90 PEN 2:LOCATE 1,15:INPUT"Schwierigkeitsgrad eingeben
mit ENTER:  0-5 (Profi - Anfaenger) ",skill
100 IF skill<0 OR skill>9 GOTO 90
110 skill=skill+10
120 LOCATE 1,15:PRINT CHR$(18);:LOCATE 1,15:INPUT"Gesch
windigkeit eingeben mit ENTER:      0-100 (Schnell - Lan
gsam) ",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:PRINT C
HR$(143)+CHR$(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 12,25:
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
```

Fortsetzung nächste Seite

```
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$=" " AND drop=0 THEN drop=1:b=y+2:a=
x
380 IF x=21 THEN drop=0
390 IF drop=1 THEN LOCATE a,b:PRINT CHR$(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 3,4000,10,
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 'Instruktionen
470 '
480 LOCATE 1,2:PEN 1:PRINT "Sie steuern  ein  Flugzeug
 ueber  einerverwuesteten Stadt und muessen  alle Ge-ba
eude zerstoeren  um landen  und tankenzu koennen. Ihr F
lugzeug bewegt sich vonlinks nach rechts ueber den Bild
schirm.";:PRINT
490 PRINT:PRINT"Wenn Sie  die  rechte  Seite  erreichen
,erscheint Ihr Flugzeug EINE ZEILE TIEFERam linken Rand
 des  Bildschirms erneut. Durch Druecken der LEERTASTE
koennen Siejeweils eine Bombe aus Ihrem unbegrenzt-en V
orrat abwerfen.";:
500 PRINT:PRINT"Jedesmal wenn Sie landen werden entwede
rdie Gebaeude  hoeher oder Ihre Geschwin-digkeit steigt
.";:PRINT:PRINT:PRINT"SOBALD SIE EINE BOMBE ABGEWORFEN
 HABEN,KOENNEN SIE  KEINE WEITERE  ABWERFEN BISDIE VORH
ERIGE EXPLODIERT IST !!!!";
510 PEN 2:LOCATE 7,24:PRINT"START mit beliebiger Taste.
";
520 a$=INKEY$:IF a$="" GOTO 520
530 RETURN
540 '
550 'Zusammenstoss
560 '
```

Fortsetzung nächste Seite

```
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$(8);:FO
R tm=0 TO 50:NEXT:NEXT:PEN 2
590 CLS:LOCATE 5,5:PRINT"Punkte";score;
600 IF score>hiscore THEN hiscore=score:LOCATE 3,8:PRIN
T "NEUER REKORD!!";
610 score=0:LOCATE 3,12:PRINT"Neustart mit N";
620 a$=INKEY$:IF a$="n" OR a$="N" GOTO 630 ELSE GOTO 62
0
630 PEN 1:MODE 1:x=2:y=2:a=2:b=2:GOTO 90
640 '
650 'Bombardierte Gebaeude
660 '
670 LOCATE a,b-1:PRINT" "+CHR$(8);:PEN 4:FOR tr=1 TO IN
T(RND(1)*3)+1:score=score+5:SOUND 3,4000,10,12,0,0,10:L
OCATE 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
```

Amthello

Ein Spiel für Denker. Versuchen Sie, die Felder Ihres Gegners einzukreisen und einzunehmen, ohne jedoch die Ihren zu gefährden. Für einen Spieler, gegen den Computer. Nur Tastatur.

```
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 #1,6,1: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,"  BITTE 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%,o%)=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%
```

Fortsetzung nächste Seite

```
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 420
420 LOCATE #1,1,3:INPUT #1,"   WELCHE SPALTE WAEHLEN SI
E ";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 QUADRAT IST
BEREITS BELEGT !"
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 ERHALTEN SIE ";vrx%
;" QUADRAT(E)"
```

Fortsetzung nächste Seite

```
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 PASSE
N.                  SPIEL AUS"
930 FOR t=1 TO 1000:NEXT t:GOTO 1550
940 CLS #1:LOCATE #1,12,2:PRINT #1,"ICH MUSS PASSEN"
950 GOSUB 2720
960 IF pas%=1 GOTO 970 ELSE 990
970 CLS #1:PRINT #1," SACKGASSE ! SIE MUESSEN AUCH PASS
EN.                 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%=o% 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
```

Fortsetzung nächste Seite

```
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 ERHALTE ICH ";vry%
;" QUADRAT(E)"
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
```

Fortsetzung nächste Seite

```
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 #1,3,1:PRINT #1,"SIE HABEN";cq%;"QUA
DRATE; ICH HABE";cn%
1660 LOCATE #1,7,3:PRINT #1,"ICH HABE GEWONNEN.....!!!
!"
1670 END
1680 CLS #1:LOCATE #1,3,1:PRINT #1,"SIE HABEN";cq%;"QUA
DRATE; ICH HABE";cn%
1690 LOCATE #1,3,3: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
```

Fortsetzung nächste Seite

```
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%(i%-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
```

Fortsetzung nächste Seite

```
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
```

Fortsetzung nächste Seite

```
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
```

Fortsetzung nächste Seite

```
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
```

Raffles

Brechen Sie ein in seiner Lordschaft's Schloß und machen Sie reiche Beute. Vielen Hindernissen werden Sie begegnen und - vergessen Sie nicht seine Dogge! Für einen Spieler gegen den Computer. Tastatur 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 15,1: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
```

Fortsetzung nächste Seite

```
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 5,20
770 PEN 5:PRINT">";
780 PEN 1:PRINT"Fluchtwege";
```

Fortsetzung nächste Seite

```
790 PEN 5:PRINT"<";:PEN 1
800 LOCATE 8,2: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$(2),1);" T
ueren":PRINT
850 PEN 3:PRINT switch$(1,0);switch$(2,0);" Licht (AUS)
"
860 PEN 3:PRINT switch$(1,1);switch$(2,1);" Licht (EIN)
":PRINT
870 PEN 4:PRINT LEFT$(windw$(1),1);LEFT$(windw$(2),1);"
 Fenster":PRINT
880 PEN 5:PRINT gem$;"  Wertvolle Juwelen"
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 Tastatur"
930 dummy=REMAIN(1)
940 AFTER delay*4,1 GOSUB 340
950 RETURN
960 '
970 'Generieren 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
```

Fortsetzung nächste Seite

```
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"Raum: ";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 'Anzeige Juwelen/Gegenstaende
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
```

Fortsetzung nächste Seite

```
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 2,3:PRINT"Sie sind entwischt";
1920 LOCATE 9,5:PRINT"mit";
1930 IF gems=stolen THEN LOCATE 6,7:PRINT"ALLEN"; ELSE
LOCATE 8,7
1940 PRINT USING " ##";stolen;
1950 PEN 5:LOCATE 7,9: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
```

Fortsetzung nächste Seite

```
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 555,150+(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"Crash ";: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 'Tueren zeichnen
2190 '
2200 READ no,dr$
2210 IF dr$="V" THEN dr=2 ELSE dr=1
2220 PEN 2
2230 pic$=door$(dr):GOSUB 2500
2240 GOTO 1280
2250 '
2260 'Fenster zeichnen
2270 '
2280 READ no,wi$
2290 IF wi$="V" THEN wi=2 ELSE wi=1
2300 PEN 4
2310 pic$=windw$(wi):GOSUB 2500
2320 GOTO 1280
2330 '
2340 'Linien zeichnen
2350 '
2360 READ x1,y1,x2,y2
2370 MOVE x1,y1,,0
2380 DRAW x1,y2,,0:DRAW x2,y2,,0
2390 DRAW x2,y1,,0:DRAW x1,y1,,0
2400 GOTO 1280
2410 '
2420 'Schalter zeichnen
2430 '
2440 READ no,sw$
```

Fortsetzung nächste Seite

```
2450 IF sw$="L" THEN sw=1 ELSE sw=2
2460 PEN 3
2470 pic$=switch$(sw,0):GOSUB 2500
2480 GOTO 1280
2490 '
2500 'Zeichen ausgeben
2510 '
2520 READ x,y:LOCATE x,y:PRINT pic$;
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
```

Fortsetzung nächste Seite

```
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

Hinweise: Alle Angaben dieses Anhangs beziehen sich auf das entsprechende Kapitel mit Seitenzahl, z.B. 1.42 = Kapitel 1, Seite 42

A |A 1.48 1.85 5.7
Abbruch von CP/M-Befehlen 1.90
Ablauf-Steuerung (LOGO) 6.22
ABS 3.3
Addition 1.38
AFTER 3.3 8.33
AMSDOS 1.85 5.1 5.22
AMSDOS-Befehle 5.5
AMSDOS-Fehlermeldungen 5.13 7.30
AMSDOS-Dateinamen 5.2 5.3
AND 3.4 8.20
AND (LOGO) 6.10
Anmeldebild 1.4 1.5
Animation 8.53
Anschluß des Computers 1.2
Anschluß von Peripherie 1.7 7.36 7.37 7.38 7.39
Anzeigenleuchte 1.16
Arithmetische Operationen 1.38
Arithmetische Operationen (LOGO) 6.9
Arrays 2.3 3.18 3.25 7.26
ASC 3.5
ASCII 6.6 7.7 7.20
ASCII-Zeichen 7.7 7.8 8.16
ASCII-Datei 1.49 3.71 7.28
ASM 5.24
ATN 3.5

Ausnahmefälle (LOGO) 6.23
AUTO 2.10 3.5

B |B 1.48 1.85 5.7
BASIC 1.25 3.1 7.31 8.7
BASIC-Diskette 4.3
Benutzer definierte Tasten 3.37 7.20 7.21 7.22
Benutzer definierte Zeichen 3.79 8.23
BF 6.6
Bildfang 1.4
BIN$ 3.6
Binary-Dateien 1.49 3.71
Binär-Zahlen 8.9 8.10
Bit 8.9
BK 6.15
BL 6.7
Blinkende Farben 1.57
BOOTGEN 5.24
BORDER 1.53 3.6 7.5
BREAK 3.7
BUTTON P (LOGO) 6.20
BYE (LOGO) 6.6 6.22
Byte 8.9

C CALL 3.7
CAPS LOCK-Taste 1.18
Cassetten-Operationen 1.7 1.86 4.9 5.22
CAT 1.47 3.7
CAT (Cassette) 4.9
CATCH (LOGO) 6.23
CHAIN 3.7
CHAIN (Cassette) 4.10

CHAIN MERGE 3.8
CHAIN MERGE (Cassette) 4.10
CHAR (LOGO) 6.7
Charakters 1.61 8.15
CHKDISC 1.90 5.20
CHR$ 1.61 3.8 8.17
CINT 3.9
CLEAN (LOGO) 6.13
CLEAR 3.9
CLEAR INPUT 3.9
CLG 3.10
CLOAD 5.20
CLOSEIN 2.12 3.10
CLOSEIN (Cassette) 4.12
CLOSEOUT 2.10 3.10
CLOSEOUT (Cassette) 4.13
CLR-Taste 1.19
CLS 1.25 3.10 7.3
CO (LOGO) 6.22
CON 5.19
Configurieren einer Diskette 4.6
CONT 3.11 7.27
CONTENTS (LOGO) 6.24
CONTRAST-Regelung 1.4
Copy Cursor-Methode 1.32
COPYCHR$ 3.11
COPYDISC 1.89 5.20
COPY-Taste 1.32
COS 3.11
COS (LOGO) 6.9
COUNT (LOGO) 6.7

CP/M 5.15
|CPM 1.44 5.7
CP/M-Dateinamen 5.17
CP/M System-Diskette 1.44 1.87 4.2
CP/M System-Spuren 5.15
CREAL 3.12
CS (LOGO) 6.13
CSAVE 5.22
CT (LOGO) 6.12
CURSOR 3.12 7.3
Cursor-Taste 1.17
D DATA 3.12 3.2 7.26 8.35
Data only format 5.25 7.43
Dateien kopieren 1.48 5.19
Dateinamen 1.48 5.2 5.3 5.17
Dateien Kopien (Tabelle) 5.11 5.12
Datentypen 5.2
Dateitypen 5.2
DC-Anschluß 1.2 1.3
DDT 5.24
DEC$ 3.13
DEF (LOGO) 6.25
DEF FN 3.14 7.27
DEFINT 3.14
DEFREAL 3.15
DEFSTR 3.15
DEG 3.15
DEL-Taste 1.17
DELETE 3.16
DEPOSIT (LOGO) 6.24

DERR 3.16 7.29 7.30
DI 3.17 8.35
DIM 2.3 3.18 7.26
DIR (CP/M) 1.45 5.19
DIR (LOGO) 6.19
|DIR 5.7
Direkte Konsol-Befehle 5.17
Disketten 1.12 1.43 4.1
Disketten einlegen 1.12
Disketten Inhaltsverzeichnis 1.47 5.1 7.30
Diskettenlaufwerk (zusätzlich) 1.9 7.38
Diskettenorganisation 7.42
Disketten kopieren 1.87 5.20
Diskettentest 1.89 5.20
Diskettenwechsel 5.2
|DISC 1.87 5.7
DISCCHK 1.89 5.20
DISCCOPY 1.88 5.20
DISC DRIVE 2-Buchse 1.9 7.38
Disketten-Dateien Befehle (LOGO) 6.19
Disketten erstellen (CP/M) 5.22
Diskettenfehler 7.29
|DISC IN 1.87 5.8
|DISC OUT 1.87 5.8
Division 1.39
DOT (LOGO) 6.14
Dotted lines 3.43 8.57
DRAW 1.65 3.19
DRAWR 3.19
|DRIVE 5.8

Dr. LOGO 6.1
DUMP 5.24
E ED 5.24
EDIT 1.31 3.20
Editieren 1.31
Editieren (LOGO) 6.5 6.12
EI 3.20 8.35
Eigenschaftslisten 6.18
Einloggen einer Diskette 5.16
Einschalten des Computers 1.4 1.5
EJECT-Taste 1.16
ELSE 1.33 3.20
EMPTYP (LOGO) 6.7
END 3.21
END (LOGO) 6.11
ENT 1.81 3.21 8.48
ENT (LOGO) 6.21
ENTER-Taste 1.17 1.20
ENV 1.80 3.22 8.44
ENV (LOGO) 6.21
Envelope Planbogen 7.35
EOF 3.24 4.11 7.27
ER (LOGO) 6.18
ERA 5.18
|ERA 5.8
ERASE 3.24
ERL 3.25
ERN (LOGO) 6.18
ERR 3.25 7.29
ERRACT 6.24

ERROR 3.25
ERROR (LOGO) 6.23
Erweiterungszeichen 7.21
ESC-Taste 1.20
EVERY 3.26 8.33
EX1 und EX2 Programme 5.5
EXAMINE 6.24
EXP 3.27
Expansion characters 3.37 7.21
EXPANSION Buchse 1.11 7.38
Exponential-Rechnung 1.40
Externe Befehle 5.7
F FALSE (LOGO) 6.24
Farben 1.51
FD (LOGO) 6.15
Fehlernummer 7.26
Fehlermeldungen (AMSDOS) 5.13 7.29 7.30
Fehlermedlungen 7.26
FENCE (LOGO) 6.14
FILECOPY 4.2
FILL 1.70 3.27 8.57
FIRST (LOGO) 6.7
FIX 3.27
Flush-Sound-Kanalstatus 3.73 8.44
FN 3.28
FOR 1.34 3.28 7.26 7.28 8.16
FORMAT 1.45 5.22
Format (print) 3.60 8.24
Format (disc) 1.45 7.42
FPUT 6.8
FRAME 1.64 3.28

FRE 3.29
FS (LOGO) 6.14
G Geschützte Dateien 1.43 3.71
Geräusch 1.78
GLIST (LOGO) 6.18
GO (LOGO) 6.22
GOSUB 1.35 3.30 7.26
GOTO 1.27 3.30
Graphik 1.61 8.56
Graphik-Bildschirm-Befehle (LOGO) 6.13
GRAPHICS PAPER 3.30 8.59
GRAPHICS PEN 3.30 8.57
GPROP (LOGO) 6.18
Gemischte Kalkulation 1.41
H Hardware 7.44 7.45 8.66
Headers 5.11 5.12
HEX$ 3.31
Hexadezimale Zahlen 8.12
HIMEM 3.31
Helligkeitsregler 1.4
Hold-Sound-Kanalstatus 3.73 8.44
HT (LOGO) 6.15
I IBM-Format 5.22 7.43
IF 1.33 3.32
IF (LOGO) 6.22
Inbetriebnahme 1.1
INK 1.55 3.22 7.5
INKEY 3.33 7.41
INKEY$ 2.12 3.34 7.41
Ink-Modus 7.4 8.61
INP 3.34

INPUT 1.28 1.29 2.2 3.34 7.27
INPUT (Cassette) 4.12
INSTR 2.5 3.35
INT 3.36
INT (LOGO) 6.9
Interrupts 7.6 8.33
I/O 7.36 7.44
ITEM (LOGO) 6.8
J JOY 3.36 7.41
Joystick-Befehle (LOGO) 6.20
Joysticks 1.7 7.41
JOYSTICK-Anschluß 1.7 7.36
K Kanal-Status 1.76 3.73 8.38
KEY 3.73 7.21
KEY DEF 3.37 7.21 7.41
Keyboard 1.17 7.20 7.21 7.22
KEYP (LOGO) 6.20
Knoten (LOGO) 6.6
Konfiguration 5-Sektor 5.16
Konsol-Befehle 5.18
Kontroll-Codes Konsole (CP/M) 5.16
Kontrollzeichen 7.3 8.61
Kopieren einer Diskette 1.75 1.88 1.89 4.1
Kopfhörer 1.80
Kreis 1.67
Kubikwurzel 1.40
L LABEL 6.22
Lautsprecher (extern) 1.9
Lautstärke 1.77
Lautstärken-Hüllkurve 1.78 3.23 8.38

Lautstärke-Regler 1.65 1.9
LEFT$ 3.38
LEN 2.10 3.39
LET 3.39
LINE INPUT 3.39
LINE INPUT (Cassette) 4.12
LIST 1.27 3.40
LIST (LOGO) 6.8
Listen- und Wörter-Verarbeitung (LOGO) 6.6
LOAD 1.48 3.40
LOAD (Cassette) 4.10
LOAD (LOGO) 6.19
LOCAL 6.11
LOCATE 1.62 4.41 7.5
LOG 3.41
LOG 10 ..... 3.41
Logische Ausdrücke 8.20
Logische Operationen (LOGO) 6.10
LOGO 6.1
LST 5.19
LT 6.15
LOWER$ 3.42
M Machine code 7.5
MAKE 6.11
MASK 3.42 8.58
MAX 3.43
MEMORY 3.43 7.26
Memory map 7.44
Menü 2.5 3.51
MERGE 3.43

MERGE (Cassette) 4.10
MID$ 3.44
MIN 3.45
MOD 1.39 3.45
MODE 1.52 3.45 7.3
Modulator/Stromversorgungseinheit (MP2) 1.3 1.5 1.65
Monitor 1.2
MONITOR-Anschluß 1.2 1.3 7.39
MOVCPM 5.23
MOVE 1.66 3.46
MOVER 3.47
Multiplikation 1.38
Musik-Planbogen 7.35
Modulo 1.39
N NEW 3.47
NEXT 1.34 3.48 7.26 7.28 8.18
NODES (LOGO) 6.18
NOT 3.48 8.22
NOT (LOGO) 6.10
Noten und Tonperioden 7.24
O ON BREAK CONT 3.49
ON BREAK GOSUB 3.49
ON BREAK STOP 3.50
ON ERROR GOTO 3.50 7.27 7.29
ON GOSUB 2.11 3.51
ON GOTO 3.51
ON SQ GOSUB 3.51 8.52
OP (LOGO) 6.23
OPENIN 2.12 3.52 7.28
OPENIN (Cassette) 4.12

OPENOUT 2.10 3.52 7.28
OPENOUT (Cassette) 4.13
Operators 8.20
OR 3.53 8.20 8.23
OR (LOGO) 6.10
ORIGIN 1.69 3.53
OUT 3.54
P PADDLE (LOGO) 6.20
PAL (LOGO) 6.14
PAPER 1.54 3.54 7.4
PAUSE (LOGO) 6.24
PD (LOGO) 6.16
PE (LOGO) 6.16
PEEK 3.55
PEN 1.54 3.55
Peripherie-Verwaltung 5.19
Peripherie 1.7
PI 3.56
PIP 5.19
Planners 7.32 7.33 7.34 7.35
PLIST (LOGO) 6.19
PLOT 1.64 3.57
PLOTR 3.57
PO (LOGO) 6.11
POKE 3.58
POS 3.59 4.11
Potenzrechnung 1.40
POTS (LOGO) 6.12
POWER switch 1.5
PPROP (LOGO) 6.19

PR (LOGO) 6.12
Primitives 6.6
PRM 6.25
PRINT 1.26 3.60 8.24
Print-Formate 3.61 8.24
PRINT SPC 3.60 8.25
PRINT TAB 3.60 8.26
PRINT USING 3.61 8.26
PRINTER-Anschluß 1.8 7.41
Printers 1.8 7.40 7.42 7.43
Programme laden 1.20
Prozeduren 6.11 6.3
Protected files 1.49 3.71
PU (LOGO) 6.16
PUN 5.19
PX (LOGO) 6.16
Q Quadratwurzel 1.40
Quersummenprüfung 8.37
Queue (sound) 3.78 7.7 8.52
R RAD 3.64
RANDOM (LOGO) 6.9
RANDOMIZE 3.64
Random numbers 3.70
RC (LOGO) 6.20
RDR 5.19
READ 3.64 7.27 8.35
Read errors 4.9
Read Dateien 5.10 5.21 7.31
RECYCLE (LOGO) 6.18 6.6
REDEFP (LOGO) 6.24

RELEASE 3.65 8.50
RELEASE (LOGO) 6.22
REM 1.35 3.65
REMAIN 3.66 8.34
REMPROP (LOGO) 6.19
REN 5.18
|REN 5.8
Rendezvous Sound-Kanalstatus 3.73 8.43
RENUM 3.66
REPEAT (LOGO) 6.23
RESTORE 3.67 8.38
RESUME 3.68 7.27
RESUME NEXT 3.68
RETURN 1.35 3.69 7.26
RIGHT$ 3.69
RL (LOGO) 6.20
RND 3.70
RQ (LOGO) 6.20
ROINTIME.DEM Programm 1.22
ROUND 3.70
RT (LOGO) 6.16
RUN 1.26 1.45 3.71
RUN (Cassette) 4.10
RUN (LOGO) 6.23
S SAVE 1.43 1.43 3.71
SAVE (Cassette) 4.14
SAVE (CP/M) 5.17
SAVE (LOGO) 6.19
Schlüsselwörter 3.3
Schreib-/Leseschutz 1.13

Sichern auf Cassette 4.13
Sichern von Variablen 2.10 3.93 5.5
Screen dump (Bildschirm-Abzug) 1.45 3.71 5.5
SE (LOGO) 6.8
SETH (LOGO) 6.17
SETPAL (LOGO) 6.14
SETPC (LOGO) 6.17
SETPOS (LOGO) 6.17
SETSPLIT (LOGO) 6.12
SETUP 5.21
SF (LOGO) 6.14
SGN 3.72
SHIFT-Taste 1.18
SHOW (LOGO) 6.13
Sideways ROMS 7.45
SIN 3.73
SIN (LOGO) 6.9
Software 1.22
Sonderzeichen 3.2
SOUND 1.76 3.73 7.23 8.40
SOUND (LOGO) 6.21
Sound-Befehle (LOGO) 6.21
Sound Laustärken-Hüllkurvenplaner 7.35
SPACE$ 3.76
SPC 3.76 8.26
SPEED INK 3.76
SPEED KEY 3.77
SPEED WRITE 3.78 4.15
Speech-synthesiser /Sprach-Synthesier 1.10
Speicherplatzverwaltung (LOGO) 6.18

Speicherbelegung 7.44
Sprites 8.64
SQ 3.78 8.45
SQR 1.40 3.79
SS (LOGO) 6.15
ST (LOGO) 6.17
Starten 1.1
STAT 5.21
STEP 3.79
Stereo 1.10 1.76 7.37
STEREO-Anschluß 1.10 1.76 7.37
STOP 3.79
STOP (LOGO) 6.23
STR$ 3.80
STRING$ 3.80
String Variable 1.29 7.27
Subtraktion 1.38
SUBMIT 5.24
SWAP 3.80
SYMBOL 3.81 7.5 8.23
SYMBOL AFTER 3.83
Syntax Fehler 1.20 7.26
SYSGEN 5.23
System-Diskette 1.44 1.85 4.2
System-Format 5.22 7.42
System-Verwaltung 5.20
System-Grundprozeduren (LOGO) 6.24
System-Eigenschaften (LOGO) 6.25
System-Spuren (CP/M) 5.17
System-Variable (LOGO) 6.24

TAB 3.83 8.26
TAG 3.84 8.59
TAGOFF 3.84 8.59
TAN 3.85
Tastenbelegung 7.20 7.21 7.22
|TAPE 1.87 5.9
|TAPE IN 1.87 5.9
|TAPE OUT 1.87 5.9
TAPE-Anschluß 1.7 7.37
TEST 3.85
TESTR 3.85
Text-Bildschirm-befehle (LOGO) 6.12
Text/windwow Planer 3.32 7.33 7.34
TF (LOGO) 6.17
THEN 1.33 3.86
THROW (LOGO) 6.24
TIME 3.86
TO 3.87
TO (LOGO) 6.12
Ton-Hüllkurve 1.78 3.22 8.47
TOPLEVEL (LOGO) 6.24
Transiente Befehle 5.18
Transparent writing 7.5 8.61
TROFF 3.87
TRON 3.87
TRUE (LOGO) 6.25
TS (LOGO) 6.13
Turnkey BASIC Disketten 4.3
Turnkey CP/M Disketten 4.4
Turtle Graphik-Befehle (LOGO) 6.18

TV-Anschluß 1.3 1.5
TYPE-Befehl (CP/M) 5.18
TYPE (LOGO) 6.13
Tonperiode 1.76 7.23
Tonplaner 7.35
U UNT 3.87
UPPER$ 3.88
|USER 5.9
USING 3.88 8.26
V VAL 3.88
Variable 1.29 7.31 8.16
Variable (LOGO) 6.4 6.11
Variable (sichern) 2.9 3.93 5.5
Vendor format 5.22 7.42
Verstärker Extern 1.9 7.37
Vibrato 8.48
Volume envelope 1.80 3.21 8.44
VPOS 3.89
W WAIT 3.89
WAIT (LOGO) 6.23
Welcome Programm 1.23
WEND 3.90 7.28
WHILE 3.90 7.26 7.28
WIDTH 3.91
Wild cards 5.4
WINDOW 2.11 3.92 7.5 8.29
WINDOW (LOGO) 6.15
Window Planer 7.32 7.33 7.34
WINDOWSWAP 3.92 8.30
WORD 6.8
WORDP 6.8

WRAP 6.15
WRITE 3.93 2.9 8.26
Write protection 1.13
X XOR 3.93 8.20 8.23
XPOS 3.94
XSUB 5.24
Y YPOS 3.95
Z Zufallszahl 3.70
ZONE 3.95 8.25



Document numérisé avec amour par

AMSTRAD CPC

MÉMOIRE ÉCRITE

https://acpc.me/