PHILIPS

C7420 VIDEOPAC

Bedienungsanleitung für das C7420 Heimcomputermodul

INHALTSVERZEICHNIS

EINFÜHRUNG IN DAS C 7420 - HANDBUCH

Als Besitzer des neuen C 7420 Microsoft BASIC Interpreter Modules sind Sie nun in der Lage, alle Möglichkeiten, die Ihr G 7400 Spielcomputer bietet, voll auszunutzen. Zum Anfang werden Sie das grosse Vergnügen kennenlernen, das beim Schreiben von Computerprogrammen entsteht. Mit den Kenntnissen, die Sie dabei erlangen, werden Sie bald in der Lage sein, Ihre eigenen Videospiele zu gestalten, so einfach oder so komplex, wie Sie wollen. Hinzu kommt die Möglichkeit, mit Ihrer G 7400-/C 7420-Kombination alle Heimcomputer-Anwendungen auszuführen. Die Möglichkeiten sind fast unbegrenzt.

Um sicherzustellen, dass Sie alle Möglichkeiten Ihrer G 7400/C 7420-Kombination nutzen können, sollten Sie dieses Handbuch aufmerksam durchlesen. Es beginnt mit einer Einführung in die Welt der Computer, Begriffserklärung und Computerprogrammierung. Es beschreibt Ihnen dann, wie Sie Ihr C 7420-Modul mit Ihrer G 7400-Tastatur und einem Bildschirm benutzen und wie Sie Programme und andere Daten mit Hilfe eines Cassettenrecorders speichern und wieder abrufen können.

Die von Ihrer C 7420/G 7400-Kombination benutzte Computersprache ist Microsoft BASIC. Diese Computersprache wurde gewählt, weil Sie die vielleicht einfachste verfügbare Computersprache ist; und Microsoft BASIC ist als Standardversion weitgehend anerkannt. Alle Microsoft-Anweisungen sind in diesem Handbuch enthalten, mit Nummern versehen und leicht anhand praktischer Beispiele zu verfolgen.

Wir haben einige Aenderungen des normale Microsoft 8k-BASIC für die C 7420-Version des Microsoft-BASIC durchgeführt (diese wird in diesem Buch als BASIC-80 bezeichnet). Durch das Berücksichtigen dieser Aenderungen und das Benutzen der neuen Möglichkeiten können Sie die Vorteile dieser erweiterten BASIC-Programmierung ausnutzen.

Die Anweisungen sind allein einem übersichtlichen Kapitel zusammengefasst. Wenn Sie bestimmte Vorgänge nachschlagen wollen, können Sie sie leicht in diesem Kapitel finden. Weiterhin ist ein Anhang vorhanden, der alle Informationen über die Erzeugung realistischer Zeichnungen auf Ihrem Fernsehbildschirm gibt.

Wenn Sie dieses Handbuch aufmerksam lesen und die Anweisungen befolgen, werden Sie sehr bald vertraut sein mit den BASIC-Programmiertechniken und den beträchtlichen Möglichkeiten, eigene Videospiele zu entwickeln und zu spielen. So wird Ihnen ihre neue C 7420/G 7400-Kombination viele schöne Stunden der Unterhaltung bereiten.

TEIL 1

DER HEIMCOMPUTER

Kapitel 1

Im täglichen Keben strömen ständig Informationen auf uns ein. Zum Beispiel benutzen wir beim Spazierengehen unsere Augen, um nicht anzustossen. Nur einer der fünf menschlichen Sinne erlaubt es uns Informationen zu speichern. Die durch unsere Sinne erworbenen Daten werden durch das Gehirn verarbeitet und entweder in direkte Aktionen umgesetzt oder zum späteren Gebrauch gespeichert. Dies kann folgendermassen schematisch dargestellt werden:

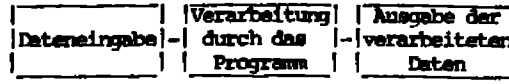
Der Computer arbeitet nach dem gleichen Prinzip. Das heisst, dass der Computer in der Lage sein muss Daten aufzunehmen, sie zu verarbeiten und wieder herauszugeben.

Das Datenangebot wird dem Computer nicht durch Sinne vermittelt, sondern durch spezielle Einrichtungen. Dieses kann durch Sensoren, Thermostaten, Fotozellen usw. oder über die Tastatur geschehen. Die Daten werden über eine der Schreibmaschine ähnliche Tastatur eingegeben. Die meisten Computer zeigen Daten auf einen Bildschirm an, einige sind jedoch in der Lage, die verarbeiteten Daten in Form einer Aktion umzusetzen, zum Beispiel durch den Arm eines Roboters.

Die Geräte, die der Computer zum Sammeln und Herausgeben von Informationen benutzt, nennt man Ein- oder Ausgabegeräte. Die Verarbeitung von Daten geschieht durch ein Bauteil innerhalb des Geräts, das man Prozessor nennt. (engl. Central Processing Unit/CPU). Die Fähigkeit, Daten zu verarbeiten, erlangt der Computer durch das Programm.

Ein Computerprogramm besteht aus einer Reihe von Befehlen, die geeignet sind, die gewünschte Ausgabe von den eingegebenen Daten zu erhalten. Wie solch ein Programm entsteht, ist im Teil 2 erklärt. Ausser mit dem Prozessor ist der Computer mit einem Speicher ausgerüstet, in dem Daten gespeichert und abgerufen werden können.

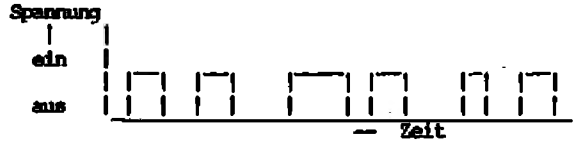
Die Funktion eines Computers kann auf folgende Weise schematisch dargestellt werden:



Dieses Schema zeigt, dass die Datenverarbeitung zwischen Eingabe und Ausgabe liegt. Da man viele verschiedene Programme in einen Computer eingeben kann, ist die Datenverarbeitung in vielen Bereichen anwendbar.

Kapitel 2

Die Zentraleinheit verarbeitet die eingegebenen Daten mit Hilfe eines Programms. Die Verarbeitung wird durch ein System elektronischer Schalter im "Mikroprozessor" durchgeführt. Die Schalter reagieren auf 2 unterschiedlich Spannungspotentiale, von denen eins die Zahl "0" und das andere die Zahl "1" darstellt. Eine Folge von hoher und niedriger Spannung kann so in einem Diagramm ausgedrückt werden.



Weil das Signal nur in der Lage ist, 2 Werte anzunehmen, nennt man es "Digital-Signal".

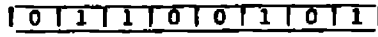
In den ersten Computern wurden diese Schaltungen aus Röhren und Widerständen gebildet. Diese wurden später durch Transistoren ersetzt, die viel kleiner und verläßlicher sind als Röhren. Integrierte Schaltungen kamen 1963 auf den Markt, besser bekannt als "Chip". Ein Chip ist ein winziges Stück Halbleitermaterial (wenige Quadrat-millimeter gross). Es besteht normalerweise aus Silizium, auf dem ein elektronischer Schaltkreisdurcheine spezielle Fertigungstechnik eingeprägt wurde.

Diese "Mikro-Chips" in dem Computer nehmen nur eingegebene Daten an, die in "1" und "0" umgesetzt sind. Eine Umsetzung ist bei der Ausgabe der verarbeiteten Daten ebenfalls erforderlich. Diese Umsetzung geschieht durch die Ein- und Ausgabegeräte.

Symbole, wie Buchstaben und Zahlen, werden im Prozessor in einem Binärcode dargestellt. Die am häufigsten verwendete Codierung heisst ASCII (American Standard Code for Information Interchange), in welchem z.B. der Buchstabe "A" als 01000001 dargestellt wird.

Heute ist es möglich, eine Zentraleinheit auf einem einzigen Chip unterzubringen. Einen solchen Chip nennt man "Mikroprozessor". Das C 7420 Modul enthält einen Z 80 A-Mikroprozessor. Mit dem C 7420 Modul wird der G 7400 zum richtigen "Home-Computer".

Die Zentraleinheit speichert die Daten in speziellen internen Speicherplätzen, die man "Register" nennt. Ein Register kann man sich vorstellen als 8 Kästchen, in denen je eine "1" oder eine "0" enthalten sein kann.



Es gibt verschiedene Register, wie:

Eingabe-Register - zur vorläufigen Speicherung von Daten aus dem Eingabegerät

Ausgabe-Register - zur vorläufigen Speicherung von Daten für das Ausgabegerät

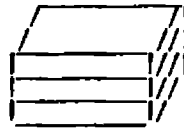
Arbeits-Register - zur Verarbeitung von Informationen, die als "1" oder "0" dargestellt sind

Das Arbeitsregister erhält die zu verarbeitenden Daten vom Speicher.

Kapitel 3

Der Speicher ist mit ähnlichen Schaltungen aufgebaut, wie die Zentraleinheit. Speicher können Informationen jedoch nur in Form von "0" und "1" aufnehmen.

Jeder kleine Kasten, der eine "1" oder "0" speichern kann, heisst "Bit". Eine Reihe von 8 Bit heisst "Byte". In dem ASCII-Code kann jeder Buch-stahe des Alphahets in einem, aus 8 Bit bestehenden Byte verschlüsselt werden. Eine Reihen von 4 Bytes nennt man "Wort". Jedes Computerwort hat einen hestimmten Speicherplatz.



Die meisten Speicher hestehen aus tausenden von Worten. Die Kapazität eines Speichers ist mit der Einheit "K", die für 1024 Bytes steht, definiert. Ein Speicher mit einer Kapazität von 16k kann demnach 16 x 1024 = 16.384 Zeichen Behalten.

Der Speicher kann dazu benutzt werden, Informationen zu speichern. Die Zentraleinheit verarheitet die Daten mit Hilfe des Programms. Dieses Programm wird ebenfalls im Speicher aufgenommen. Die Programmanweisungen werden in einem reservierten Teil des Speichers in Form von "1" oder "0" gespeichert. Das gespeicherte Programm ist somit in Maschine-sprache vorhanden.

Es gibt zwei Arten von Speichern: "Festspeicher" und "Arbeitsspeicher".

Festspeicher:
Dies ist ein Speicher, von dem die Zentraleinheit nur Informationen lesen, aber nicht speichern kann. Diesen Typ nennt man ROM (Read Only Memory). Der ROM-Speicher wird hauptsächlich benutzt, wenn Programme nicht geändert werden sollen. Diese Programme werden von der Herstellerfirma festgelegt und sind fest im ROM. Video-Spiele, wie sie in einem Video-Spiel-Computer benutzt werden, sind ein Beispiel dafür. Das C 7420 Modul ist mit einem 16K ROM-Speicher ausgestattet, der ein Uebersetzungsprogramm enthält, das Uebersetzungsbefehle in Maschinensprache übersetzt.

Arbeitsspeicher:
Die vielen Anwendungsmöglichkeiten eines Computers resultieren aus seiner Fähigkeit, viele verschiedene Programme zu verarbeiten. Der Computer braucht daher einen Speicher, der erlaubt, Daten zu speichern und abzurufen. Ein solcher Speicher, wird RAM genannt (Random Access Memory). Der C7420-Modul hat eine Kapazität von 2K für die Bildschirmanzeige und 14K RAM für die Speicherung von Benutzerprogrammen. Die Kombination des 16K ROM mit dem 16K RAM machen das C 7420 zu einem benutzers-freundlichen Computer. Ueber die Tastatur kann der Anwender sein Programm in BASIC-Computersprache eingeben. Der Interpreterim ROM sorgt automatisch dafür, dass das BASIC-80-Programm in die Maschinensprache umgesetzt wird.

Kapitel 4

Es wurde in Kapitel 1 angesprochen, dass jeder Computer Ein- und Ausgabegeräte braucht. So hat zum Beispiel der G 7400 eine Tastatur, mit der Daten eingegeben werden können. Für die Datenausgabe kann ein Bildschirm - Ihr Fernseher - an den G 7400 angeschlossen werden.

Als Ergänzung zur Standardtastatur und dem Bildschirm ist es möglich, einen Cassettenrecorder über das C 7420-Modul anzuschliessen. Mit Hilfe des Cassettenrecorders können Daten auf eine Cassette überschrieben oder Informationen von der Cassette in den Speicher eingelesen und abgerufen werden. Dieses bietet eine Menge Möglichkeiten. Zum Beispiel kann ein Programm, das über die Tastatur in den Speicher eingegeben wurde, mit Hilfe des Cassettenrecorders auf Cassette überspielt werden. Sollte dieses Programm zu einem späteren Zeitpunkt wieder benötigt werden, so kann der Computer es von der Cassette wieder einlesen und speichern. Das erspart die erneute Eingabe über die Tastatur.

Auf diese Weise kann der Anwender sich eine ganze Programmbibliothek auf Cassette anlegen und speichern. Dadurch wird die Cassette zu einem externen Speicher. Sie stellt eine Erweiterung des internen Speichers dar.

TEIL 2

SCHREIBEN EINES COMPUTERPROGRAMMS

Kapitel 1

Teil 1 ist hauptsächlich der "Hardware", unter der man alles Greifbare versteht, gewidmet, wie Tastatur, Bildschirm und C 7420 Computer-Modul.

Der Ausdruck "Software" wird zur Bezeichnung von Programmen benutzt, die im Computer gespeichert werden können. Programme sind Befehlsfolgen, die keine direkten Bestandteile des Computers sind.

Man nennt den Computer oft die "Magische Kiste", die in der Lage ist, alle Probleme zu lösen. Das ist natürlich ein Fehler. Der Computer ist ein Gerät, das nur die Befehle des Anwenders ausführen kann. Wir können den Begriff "Programm" nur in dem Fall anwenden, wenn eine Folge von Befehlen vorliegt, die auf ein vorgegebenes Ziel gerichtet ist. Zum Beispiel ein Programm, das aufgestellt wurde, um die Durchschnittszahl der eingegebenen Werte zu ermitteln. Nicht alle Programme müssen arithmetische Operationen enthalten. Daher ist zur Aufstellung eines Programms kein grosses mathematisches Wissen erforderlich.

Programme werden in solchen Programmiersprachen wie BASIC-80 geschrieben. Die Befehle in BASIC-80 werden in Teil 5 näher erklärt.

Um ein Programm zu schreiben, sitzt der Programmierer nicht nur hinter seinem Schreibtisch und schreibt eine Anzahl von Befehlen auf. Die Aufstellung eines Programmes teilt sich in folgende Schritte auf:

1 Entwurf des Programms
2 Umsetzen des Entwurfs in BASIC-80 Befehle
3 Eingeben und Testen des Programms

Diese Schritte werden in den folgenden Kapiteln ausführlich erklärt.

Kapitel 2

Ein Programm besteht aus einer Reihe von Befehlen, die zum Erreichen eines vorgegebenen Ziels notwendig sind.

Das bedeutet, dass der allererste benötigte Schritt für den Entwurf das Festlegen des Programm-Ziels ist, das die gewünschten Ergebnisse ausdrückt. Erst dann kann mit dem eigentlichen Programm begonnen werden. Ein Programmziel kann sein: Die Darstellung der Tabelle "X" auf dem Bildschirm. Hier ist "X" das einzugebende Zahl, wodurch die Tabelle bis zu zehnmal erweitert werden kann.

Ist einmal das Ziel klar definiert, kann man eine Methode wählen, mit der das Ziel erreichtwerden kann. Die im BASIC-80 verfügbaren Anweisungen sollten in Betracht gezogen werden. Die einfachsten Anweisungen und Befehle sind:

Arithmetische Anweisung (LET)
Mit dieser Anweisung kann addiert, subtrahiert, multipliziert, dividiert oder es können Gleichungen mit zwei oder mehr Speicherfelden aufgestellt werden.

Sprung-Anweisung (GOTO)
Mit dieser Anweisung kann innerhalb eines Programms an eine andere Stelle gesprungen werden. Eine Variation dieser Anweisung kann wieder an die alte Stelle des Programms zurückverzweigen, wenn der Verzweigungsteil abgeschlossen ist. Diese Verzweigung nennt man "Unterprogramm".

Vergleichsanweisung (IF)
Mit dieser Anweisung können zwei Speicherfelder miteinander verglichen werden. Es ist aber auch der Vergleich mit einer Konstanten möglich.

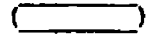
Druck-Anweisung (PRINT)
Diese Anweisung ist notwendig, um Daten auf dem Bildschirm auszugeben.

Eingabe-Anweisung (INPUT)
Diese Anweisung ermöglicht die Eingabe von Daten über die Tastatur.

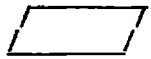
Schreib- und Lesebefehl (CSAVE-CLOAD)
Dieser Befehl erlaubt es, Daten vom Arbeitsspeicher auf eine Cassette zu schreiben oder von der Cassette in den Arbeitsspeicher zu laden. (Cassetten-SAVE und Cassetten-LOAD)

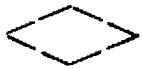
Natürlich gibt es viel mehr BASIC-80 Anweisungen als diese 6.

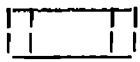
Beim Entwurf eines Programms wird ein Schema gewählt, bei dem die Anweisungen durch Symbole beschrieben werden können. Die Linien zwischen den Symbolen deuten die Reihenfolge der Anweisungen. Ein solches Schema nennt man daher "Flussdiagramm". Die wichtigsten Symbole eines Flussdiagramms sind:


Start und Ende

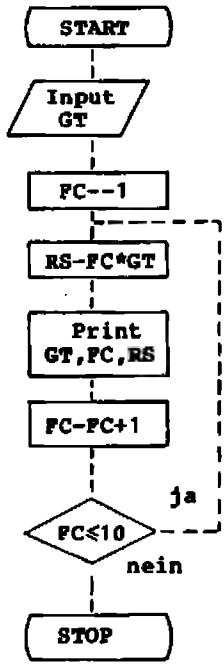

Operation


Ein- oder Ausgabe-Anweisungen


Vergleichsanweisung


Sprunganweisung zum Unterprogramm

Ein Flussdiagramm für ein vorgegebenes Ziel mit den o.g. Symbolen würde folgendermassen aussehen:



Programmstart

Tastatur eingabe akzeptiert

Faktor auf 1 setzen

Ergebnis wird: Faktor x Eingabe

Bildschirmanzeige: Eingabe, Faktor und Ergebnis

Factor um 1 erhöhen

Ist Faktor grösser als 10, Sprung zum Programmstop

Programmastop

Kapitel 3

Wenn das Konzept des Programms fertig ist, kann der Entwurf in der Programmiersprache BASIC-80 codiert werden.

Zur Vermeidung falscher Programme ist es ratsam, den Entwurf zu kontrollieren. Dieses geschieht am besten durch Einsetzen eines Beispiels, das man durch das Programm verfolgt. Das Ergebnis dieses Tests muss sich mit dem angestrebten Ziel decken.

Man codiert den Entwurf in BASIC-80 geschieht, indem man jedes Symbol aus dem Flussdiagramm in einen der verfügbaren BASIC-80-Befehle überträgt. Das Startsymbol wird in einen Befehl übersetzt, der den Bildschirm löscht. Im BASIC-80 ist jedem Befehl eine Zahl vorangestellt, die sogenannte "Zeilennummer". Die Zeilennummer kann frei gewählt werden; es ist aber darauf zu achten, dass die Zeilennummer der folgenden Zeile immer grösser sein muss, als die vorangegangene. Es ist daher empfehlenswert, die Zeilennummern in Schritten von 10 oder mehr steigen zu lassen, um die Möglichkeit von Einfügungen weiterer Befehle zwischen zwei Zeilen zu erhalten.

Alle verfügbaren BASIC-80 Befehle werden in Teil 5 behandelt.

Das Flussdiagramm auf Seite 18 würde folgendermassen in BASIC-80 codiert:

```
10 INIT 6
20 INPUT "Zahl";GT
30 LET FC=1
40 LET RS=FC*GT
50 PRINT GT,FC,RS
60 LET FC=FC+1
70 IF FC<=10 GOTO 40
80 END
```

Bei der Codierung werden alle Befehle aufgeschrieben, so dass das gesamte Programm niedergeschrieben ist.

Kapitel 4

Wenn das gesamte Konzept in BASIC-A0 Befehlen codiert ist, kann es über die Tastatur eingegeben werden.
Die RET-Taste muss nach jeder eingegebenen Zeile gedrückt werden. Wenn alle Zeilen eingegeben sind, kann durch Drücken der Taste "LIST" das gesamte eingegebene Programm über den Bildschirm aufgeblendet werden.

Trotz grösster Sorgfalt beim Entwurf und der Codierung des Programms ist es immer möglich, dass das Programm nicht einwandfrei funktioniert Das resultiert oft daraus, dass Fehler im Programm sind oder nicht korrekte Befehle im Programm benutzt wurden. Darum sollte das Programm sehr sorgfältig auf Fehler hin untersucht werden.

Vor dem Testen des Programms ist es eine ratsame Vorsichtsmassnahme, das eingegebene Programm mittels des "CSAVE-Befehls" auf Cassette zu überspielen. Dadurch erspart man sich eine erneute manuelle Eingabe im Falle einer auftretenden Störung. Das Programm kann mittels des "CLOAD-Befehls" von der Cassette in den Speicher geladen werden.

Man testet am besten mit den Daten, deren Ergebnisse bereits bekannt sind. Die häufigsten Fehler, die im Test aufgedeckt werden, sind erfahrungsgemäss:

Ein Befehl wurde nicht korrekt über die Tasten eingegeben,

In einem Befehl wurde ein Rechenfehler gemacht,

Ein "GOTO-Befehl" wird im Programm falsch verwendet,

Im Programm ist die Anweisung für die Bildschirmanzeige nicht korrekt.

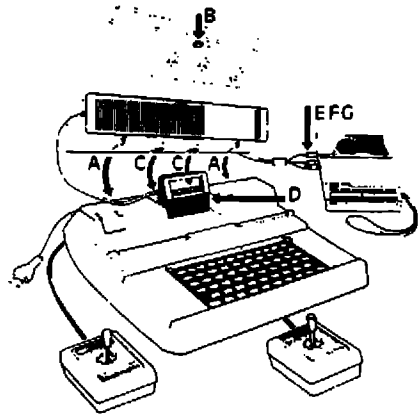
Ist das Programm in seiner Gesamtheit getestet, wird es wieder mit dem "CSAVE-Befehl" auf eine Cassette überspielt, so dass es zu einem späteren Zeitpunkt wieder in den Speicher des Computers eingelesen werden kann.

TEIL 3

EINBAU UND BEDIENUNG

Kapitel 1

Wenn Sie nach dieser Einbauanleitung vorgehen, ist das C 7420-Heimcomputer-Modul sehr einfach in den G 7400-Spielcomputer einzubauen:



1 Halten Sie das Modul so, dass es mit der dunkel geriffelten Front zu Ihnen zeigt. Das schwarze flexible Verbindungskabel mit dem Gehäusestecker ist nun auf der linken Seite.

2 Die Führungsnasen (A) werden in die Lüftungs-schlitze auf der Rückseite der Konsole eingesteckt, so dass der Modul die gesamte Breite der Konsole ausfüllt.

3 Das Modul vorsichtig nach vorne schieben, bis die Führungsnasen (A) die obere Kante der Lüftungsschlitze greifen.

4 Drücken Sie nun den Knopf (B) auf der Rück-seite des Moduls. Die Führungsnasen (C) bewegen sich aufwärts. Beachten Sie bitte, dass das Bild die umgekehrte Situation darstellt, um die einzelnen Teile besser identifizieren zu können.

5 Setzen Sie nun die Führungsnasen (C) in die mittleren Schlitze, und lassen Sie die Taste (B) wieder los. Die Führungsnasen werden die untere Kante der Lüftungsschlitze greifen und das Modul sicher in der Konsole verankern.

6 Die elektrische Verbindung zwischen dem Modul und der Konsole wird hergestellt, indem der Stecker (D) mit dem Aufkleber nach vorn in die Konsole eingesetzt wird.

Anschluss eines Audio-Recorders

Ein normaler Cassettenrecorder kann auch als externer Datenspeicher verwandt werden, die Benutzung eines automatischen Datenrecorders wird jedoch empfohlen.

Anschluss des Cassettenrecorders:

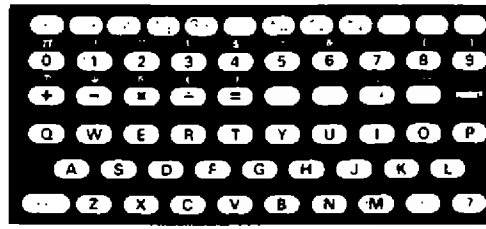
7 Schliessen Sie den roten Stecker (E) an der Mikrophonanschluss (Mic) des Recorders an.

8 Schliessen Sie den weissen Stecker (F) an den Kopfhörer-anschluss (Ear) des Recorders an.

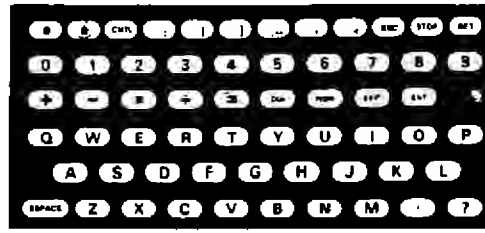
9 Schliessen Sie den schwarzen Stecker (G) an de Fernbedienungs-anschluss (Rem) an.

Hinweis:
Das C 7420-Heimcomputer-Modul erhält seinen Strom von der Konsole.

Tastatur A



Tastatur B



Es ist möglich, dass Sie die Tastatur B haben. In diesem Fall haben folgende Tasten die gleiche Funktion:

⬆	=	SHIFT
⇪	=	LOCK
STOP	=	BREAK
OUI	=	YES
NON	=	NO
EFF	=	CLEAR
ENT	=	ENTER
△	=	RESET
ESPACE	=	SPACE

Kapitel 2

Wenn das C 7420-Modul korrekt an den G 7400 Videospiel-Computer angeschlossen ist, kann die Arbeit mit BASIC-80 beginnen.

Verbindungen herstellen, Netzschalter einschalten, dann die "RESET"- und die "0"-Taste drücken.

Der Bildschirm färbt sich hellblau.

In der ersten Zeile wird folgendes in weissen -Zeichen ausgedruckt:

```
00001   X=1   Y=8     A0=140   A1=141
```

Und in schwarzen Zeichen wird angezeigt:

```
BASIC VIDEOMAC +
14075 bytes free
OK
```

Das blinkende schwarze Quadrat unter diesem Text nennt man "CURSOR" (Zeiger). Er zeigt den Punkt auf dem Bildschirm an, wo die gerade über die Tastatur eingegebenen Daten erscheinen.

Die Position des Cursors wird auf der obersten Linie angezeigt. Die Zahl hinter dem "X" zeigt die Zeile und die Zahl hinter dem "Y" die Spaltennummer an.

Hinter "A0" steht die Zahl 140 und hinter "A1" die Zahl 141. Die Zahl hinter "A0" bezieht sich auf die linke Aktiontaste des Handreglers. Die Zahl hinter "A1" bezieht sich auf die rechte Actiontaste des Handreglers. Diese Zahlen stellen Code-Symbole dar (siehe Teil 5, Anhang D). Die Zahl 140 steht für "Home". Das heisst, wenn die linke Aktiontaste gedrückt wird, geht der Cursor auf die 2. Zeile, Spalte 1. 27

Wenn ein BASIC-Programm eingelesen ist, bedeutet das, dass die ersten 21 Zeilen des Programms auf dem Bildschirm angezeigt werden. Die Zahl 141 steht für "RETURN"; das heisst, dass die rechte Aktiontaste die gleiche Funktion hat wie die "RET-Taste" der Eingabetastatur.

Die Zahl hinter "A0" kann verändert werden, indem die "BREAK-Taste" und die linke Aktiontaste gleichzeitig gedrückt werden. Die Zahl hinter "A1" kann ebenfalls durch Drücken der "BREAK"-Taste und gleichzeitigem Drücken der rechten Aktionstaste verändert werden. Nun kann ein Symbol-Code aus der Liste der "Symbol-Codes" (Teil 5, Anhang D) eingegeben werden. Die Aktionstaste hat nun dieselbe Funktion wie die bezeichnete Taste. Zum Beispiel "A0" mit dem Wert "97" gibt der linken Aktionstaste die gleiche Funktion wie die Taste "a".

Die Zahl "00001" auf der obersten Zeile zeigt an, von welcher Programmzeile die BASIC-Befehle angezeigt werden, nachdem man die linken Action-taste mit der CNTL-Taste gedrückt hat. Diese Zahl kann durch Drücken der "CNTL-Taste" und der Zahlentaste geändert werden. Zum Beispiel, wenn der Anwender die folgenden Tasten drückt: CNTL+1, CNTL+4, CNTL+6, CNTL+5, CNTL+0 wird die Zahl 14.650 in der obersten Zeile anzeigen. Das bedeutet, wenn die linke Aktiontaste gedrückt wird, werden die ersten 21 Zeilen, die auf die Zeilen-nummer 14.650 folgen, auf dem Bildschirm aufgeblendet.

System-Service-Zeile
Die oberste Zeile auf dem Bildschirm nennt man die "System-Service-Zeile". Diese Zeile kann mit der folgenden Anweisung ausser Kraft gesetzt werden:

```
PRINT CHR$(150)
```

Kapitel 3

Die folgenden Tasten stehen auf der Tastatur zur Verfügung:

: #] ¨ ; < 0-9 + - X ÷ = yes no a-z space . ?

In Verbindung mit der "SHIFT-Taste" können die folgenden Tasten (Grossbuchstaben) eingesetz werden:

* @ [^ / > # ! ,, £ $ % & , () ↑ ↓ ↖ ← → _ A-Z

Es gibt noch eine Anzahl weiterer Funktionstasten auf der Tastatur:

RET Diese Taste wird nach der Eingabe eines BASIC-Befehls gedrückt.

CLEAR Durch Drücken dieser Taste wird des Zeichen vor dem Cursor gelöscht.

BREAK Durch Drücken dieser Taste wird der Programmalauf unterbrochen.

SHIFT Wird diese Taste in Verbindung mit einer anderen Taste gedrückt, so werden Grossbuchstaben geschrieben.

LOCK Wenn die "LOCK-Taste" gedrückt wird, werden bei Betätigung der Buchstabentasten die Buchstaben als Grossbuchstaben geschrieben. Die "LOCK-Taste" nochmals gedrückt, er gibt wieder kleine Buchstaben.

ENTER Diese Taste hat die gleiche Funktion wie die RET-Taste, jedoch geht der Cursor nicht in die nächste Zeile, wenn die ENTER-Taste betätigt wurde.

CNTL Diese Taste hat nur dann eine Funktion, wenn sie in Verbindung mit einer anderen Taste gedrückt wird.

1 Wenn die "CNTL-Taste" gleichzeitig mit der "i-Taste" gedrückt wird, so werden alle Symbole rechts des Cursors nach rechts gerückt. Es entsteht eine Leerstelle.

2 Wenn die "CNTL-Taste" gleichzeitig mit der "CLEAR-Taste" gedrückt wird, wird der Bildschirm gelöscht. Der Cursor steht dann in Zeile 2 auf Position 1.

3 Wenn die "CNTL-Taste" gleichzeitig mit der "d-Taste" gedrückt wird, wird die Zeile ab der Cursor-Position gelöscht.

(Siehe Teil 4, Kapitel 1)

ESC Diese Taste hat keine Funktion.

Durch Dauerdruck auf eine Taste wird deren Funktion ständig wiederholt.

Cursor-Steuerung
Der Cursor kann durch Betätigung der Steuerungstaste oder des rechten Steuerhebels über den gesamten Bildschirm bewegt werden.

- Die ↓ -Taste führt den Cursor zur nächste Zeile auf dem Bildschirm. Dasselbe geschieht durch Ziehen des Steuerhebels in Richtung Bediener.

- Die ↑ -Taste führt den Cursor zur vorstehenden Zeile auf dem Bildschirm. Dasselbe geschieht, wenn der Steuerhebel vom Bediener weg nach vorne gedrückt wird.

- Durch eine Bewegung nach rechts des Steuerhebels oder durch Drücken der → -Taste wird der Cursor nach rechts bewegt.

- Durch eine Bewegung des Steuerhebels nach links oder durch Drücken der ← -Taste wird der Cursor nach links bewegt.

- Durch Drücken der ↖ -Taste wird der Cursor auf die Position 1 der 2. Zeile gestellt. Wenn dieses getan ist, werden die ersten 21 Zeilen des Programms auf dem Bildschirm angezeigt.

Durch Halten des Steuerhebels in der gewählten Position wird die Funktionsbewegung des Cursors ständig wiederholt.

Kapitel 4

Für BASIC-80 ist der Bildschirm in 23 Zeilen mit jeweils 40 Symbolen unterteilt. Die oberste Zeile ist reserviert als "System-Service-Zeile". Wenn die "LOCK-Taste" gedrückt wird, erscheint das Wort "CAPS" (Grossbuchstaben) in der System-Service-Zeile, welches anzeigt, dass die Alphabet-Tasten als Grossbuchstaben dargestellt werden.

Der Cursor kann nicht in der System-Service-Zeile stehen.

Bei eingabe eines BASIC-Programm kann der Cursor nicht in den ersten Position einer Zeile (X=0), den beiden obersten Zeilen (Y=0 und Y=1) und den beiden untersten Zeilen (Y=21 und Y=22) stehen. Trotzdem sind alle 23 Zeilen mit je 40 Positionen für das BASIC-Program verfügbar.

Kapitel 5

Durch den Cassettenrecorder können Programme auf die Cassette überspielt oder zurückgespielt werden.

Bei Verwendung einer Musikcassette spielt die Tonblenden- und Lautstärkeneinstellung beim Ueberspielen von und auf die Cassette eine wichtig Rolle. Ist die Tonblende ungünstig eingestellt, oder die Lautstärke zu hoch oder zu niedrig, kann der Computer die Cassette nicht einwandfrei einlesen.

Beim Lesen auf der Cassette muss die Tonblende des Recorders auf Maximum und auf mittlere Lautstärke eingestellt sein. Um die richtige Lautstärken-einstellung beim Einlesen einer Cassette, deren Programm nicht vom Anwender selbst geschrieben wurde, sollte man mit der kleinsten Stufe beginnen. Wird die Cassette nicht einwandfrei eingelesen, den Versuch mit etwas grösserer Lautstärke wiederholen. Jedes Programm auf einer Cassette sollte mit einem eigenen Namen versehen werden, den sogenannten Dateinamen. Dieser Name sollte bei jedem Laden des Programms verwendet werden. Dieser Dateiname erleichtert die Identifizierung und darf bis zu 6 Zeichen umfassen. Der Computer vervollständigt den Namen durch das "%-Zeichen". Wenn er weniger als 6 Zeichen umfasst, zum Beispiel der Dateiname "TEST" wird vom Computer als eine Datei mit Namen TEST% gelesen.

Folgende Schritte sind beim Uebertragen auf eine Cassette zu beachten:

1 Unbenutzte Cassetten einlegen und zurückspulen.
2 Wiedergabe- und Aufnahmetaste am Cassettenrecorder drücken. (PLAY- und RECORD-Taste)
3 Mit der G 7400-Tastatur
  CSAVE
  eingeben und RET-Taste drücken.

4 Wenn auf dem Bildschirm OK erscheint, folgende Eingabe machen:
CSAVE "Dateiname"
und RET-Taste drücken.
5 Ist das Programm auf die Cassette überspielt, wird auf dem Bildschirm OK angezeigt.
6 Cassettenrecorder stoppen und Band zurückspulen.

Die folgenden Schritte sollten beachtet werden, wenn ein Programm von einer Cassette eingelesen wird:

1 Cassette zum Anfang zurückspulen.
2 Wiedergabetaste am Cassettenrecorder drücken.
3 Mit der G 7400-Tastatur
CLOAD "Dateiname"
eingeben und RET-Taste drücken.
4 Wenn der Computer das Programm gefunden hat, wird auf dem Bildschirm
FOUND Dateiname
angezeigt.
5 Wenn das Programm eingelesen ist, wird OK auf dem Bildschirm angezeigt.
6 Cassettenrecorder stoppen und Band zurückspulen.

Es ist möglich, mehr als ein Programm auf eine Cas sette zu überspielen. Wenn dies erforderlich ist, die Cassette nicht auf den Anfang zurückspulen, sondern auf eine unbenutzte Position bringen. In diesem Fall ist es nicht notwendig, den Befehl CSAVEL zu benutzen. Dieser Befehl hilft, den Einfädelstreifen der Cassette zu überbrücken. Findet der Computer ein anderes Programm beim Einlesen einer Cassette, wird auf dem Bildschirm

SKIP Dateiname

angezeigt.
Der Computer liest dann die Cassette weiter, bis das gesuchte Programm gefunden ist. Dieses wird dann in den Speicher eingelesen.

Fehlermeldungen
Beim Einlesen einer Cassette kann der Computer folgende Fehlermeldungen aufblenden:

BAD FILE
Dies bedeutet, dass der Computer das gewünschte Programm nicht lesen kann. Dies kann durch Verschmutzung des Aufnahme- und Wiedergabekopfes des Cassettenrecorders geschehen. Es ist daher ratsam, ein Programm mehrfach auf eine Cassette zu übertragen.

SPEED
Wenn die Lautstärke oder Tonblende des Cassettenrecorders nicht richtig eingestellt sind, kann der Computer folgende Fehlermeldungen anzeigen:

SPEED —
SPEED -
SPEED +
SPEED ++

Der Computer wird in jedem Fall versuchen, Programm zu lesen; es kann jedoch vorkommen, dass die Daten unvollständig von ihm gelesen werden. Wenn diese Meldung erscheint, muss das Programm mit veränderter Lautstärkeneinstellung neu eingelesen werden. Diese Meldung kann auch durch zu geringe Batteriespannung hervorgerufen werden. Im diesem Falle sollten die Batterien gegen neue aus-getauscht werden.

BAD LABEL - REPOSITION TAPE
Diese Meldung kann erscheinen, wenn der Computer das gewünschte Programm nicht findet. Der Computer liest jedoch die Cassette weiter. Die Cassette sollte dann durch Betätigen der Stop-Taste des Recorders gestoppt und zum Anfang zurückgespult werden. Dann die Laut-stärke erhöhen und den Wiedergabeknopf drücken. Der Computer wird erneut versuchen, das Programm zu lesen. Cassettenrecorder mit REM-Busche werden automatisch mit den BASIC-Befehlen CSAVE und CLOAD gestartet und gestopt.

Cassettenrecorder ohne REM-Busche
Cassettenrecorder, die nicht mit einer REM-Busche ausgestattet sind, müssen manuell gestartet und gestoppt werden. Die Cassetten von diesen Recordern werden sofort nach Drücken der Wiedergabetaste gelesen und nehmen sofort nach Drücken der Wiedergabe- und Aufnahmetaste auf. Der Anwender sollte daher die Wiedergabetaste (PLAY) des Recorders erst drücken, nachdem er einen CLOAD-Befehl gegeben hat. Vor Eingabe des CSAVE-Befehl sollte die Wiedergabe- und Aufnahmetaste gedrückt werden. (PLAY- u. REC-Taste)

Cassettentypen

Es ist empfehlenswert, für die Datenspeicherung FE-Cassetten C60 zu verwenden. Eine spezielle Daten-cassette kann ebenso verwandt werden. Dieser Cassetentyp hat keinen Einfädestreifen am Anfang. Das macht den CSAVE-Befehl überflüssig.

TEIL 4

PROGRAMMIEREN MIT BASIC-80

Kapitel 1

Wie man mit BASIC-80 beginnt ist im Teil 2, Kapitel 3, beschrieben. Wenn BASIC-80 auf dem Bildschirm "OK" anzeigt, bedeutet dies, dass der Computer einen Befehl vom Anwender erwartet. Er ist nun auf seiner Kommandoebene. Von dieser Kommandoebene aus kann BASIC-80 in zwei Arten benutzt werden: direkt und indirekt.

In der direkten Anwendung werden die Befehle sofort ausgeführt, aber nicht gespeichert. Man nennt dies ein Kommando.

Versuchen Sie folgende Eingabe über die Tastatur:

```
PRINT 5+2
```

Sollten Sie einen Eingabefehler gemacht haben, benutzen Sie zum Löschen die "CLEAR-Taste", und das Symbol auf der linken Seite des Cursors wird gelöscht. Auf diese Art können so viele Symbole wie nötig gelöscht werden. Die "RET-Taste" muss nach Eingabe dieses Befehls gedrückt werden Alternativ kann auch die rechte Aktiontaste gedrückt werden. Vergessen Sie nicht, dass die "RET-Taste" nach jeder Befehlseingabe gedrückt werden muss. BASIC-80 folgt dem eingegebenen Befehl und antwortet mit:

```
7
OK auf dem Bildschirm
```

BASIC-80 ist nun wieder auf Kommandoebene. Wird die "SHIFT"- und die "X-Taste" gleichzeitig gedrückt, so wird der Bildschirm vollständig gelöscht, das heiss, dass der Computerspeicher leer ist. Als Alternative zum Betätigen der "SHIFT"-und der "X-Taste" kann die linke Actiontaste betätigt werden.

In der indirekten Anwendung steht den Befehlen eine Zeilennummer voran. Die eingegebenen Befehle werden nicht ausgeführt, aber in der Folge ihrer Zeilennummern im Computer gespeichert.

Versuchen Sie folgende Eingabe über die Tastatur:

```
10 PRINT 5+2
```

Es geschieht nichts, wenn die "RET-Taste" oder die rechte Aktiontaste gedrückt werden. Wenn man aber die "SHIFT"- und die "X-Taste", oder die linke Aktiontaste drückt, wird die Bildschirmanzeige gelöscht und der BASIC-80-Ausdruck erscheint:

```
00010 PRINT 5+2
```

Dieses zeigt an, dass die eingegebenen Befehle vom Speicher übernommen wurden. Wenn der Cursor um eine oder mehrere Zeilen nach unten bewegt, durch Ziehen des rechten Steuerhebels zum Körper hin und folgendes eingibt:
RUN
Danach die RET-Taste drückt, so wird der gespeicherte Befehl ausgeführt. BASIC-80 zeigt an:

```
7
OK
```

Die gespeicherten Befehle werden zum Programm. Jede BASIC-80-Programmzeile beginnt mit einer Zeilennummer, die zwischen 0 und 65529 liegen muss.

Aendern eines Programms

Eine Programmzeile kann geändert werden, indem man eine neue Programmzeile mit der gleichen Zeilen-nummer eingibt. Zusätzliche Programmzeilen können eingefügt werden, indem man sie mit einer noch nicht benutzten Zeilennummer versieht. Eine Programmzeile kann ebenfalls durch Betätigung der rechten Aktiontaste geändert werden.

Versuchen Sie folgende Eingabe:

```
20 PRINT 6-3
```

und drücken Sie die rechte Aktiontaste. Nun drücken Sie die linke Aktiontaste. Der Bildschirm wird gelöscht und es erscheint die BASIC-80-Anzeige:

```
00010 PRINT 5+2
00020 PRINT 6-3
```

Führen Sie nun, mittels des rechten Steuerhebels den Cursor auf die 6 in der Programmzeile 20. Drücken Sie die Taste "4" und die rechte Aktiontaste. Die Programmzeile 20 ist nun geändert auf "PRINT 4-3", und der Cursor steht auf der Position 1 der nächsten Zeile. Auf diese Art und Weise können weitere Programmzeilen hinzugefügt werden.

Nun versuchen Sie dieses:
Betätigen Sie die linke Aktiontaste.
In BASIC-80 wird ausgedruckt:

```
00010 PRINT 5+2
00020 PRINT 4-3
```

Steuern Sie nun den Cursor mit dem rechten Steuerhebel auf die 1 der Zeilennummer der ersten Zeile. Drücken Sie die Taste "3" und steuern Sie den Cursor auf die 5 in der gleichen Zeile. Drücken Sie die Taste "8" und dann die rechte Aktiontaste. Nun ist die Programmzeile 30 dem Programm hinzugefügt worden. Zum Ueberprüfen, dass dies geschehen ist, muss die linke Aktiontaste gedrückt werden.

Der Bildschirm ist gelöscht, und es erscheint die BASIC-80-Anzeige:

```
00010 PRINT 5+2
00020 PRINT 4-3
00030 PRINT 8+2
```

Nach Betätigen der rechten Aktiontaste muss sichergestellt werden, dass der Cursor auf der ersten Position der nächsten Zeile steht. Geschieht dies nicht, so ist die Aktiontaste nicht richtig bestätigt worden. In Kleinbuchstaben eingegebene Befehle werden durch BASIC-80 in Grossbuchstaben geändert. Denken Sie stets daran, dass die rechte Aktiontaste die gleiche Funktion hat wie die "RET-Taste" und die linke Aktionstaste genauso funktioniert wie die " ↖ -Taste".

Eingabe von Befehlen

Es stehen eine Reihe von Möglichkeiten zur Zeitersparnis bei der Eingabe von Befehlen zur Verfügung. Die "?-Taste" kann an stelle von "PRINT" benutzt werden. BASIC-80 setzt das "?" in "PRINT" um.

Versuchen Sie folgende Eingabe auf einer unbenutzten Zeile:

```
100 ? 2+9
```

Vergessen Sie die "RET-Taste" nicht. Betätigen Sie die linke Aktiontaste und beobachten Sie, was BASIC-80 mit der Programmzeile 100 macht.

Es gibt andere Befehle, die durch die Benutzung der "CNTL-Taste" ebenfalls vereinfacht werden. Diese Taste funktioniert nur dann, wenn sie zusammen mit einer anderen Taste betätigt wird. (Siehe Teil 3, Kapitel 3). Was BASIC-80 ausdruckt, wenn die "CNTL-Taste" gleichzeitig mit einer anderen Taste betätigt wird, ist im folgenden Bild dargestellt.

CNTL-Taste + A-Taste	BASIC druckt **CLOAD** ab
CNTL-Taste + B-Taste	BASIC druckt **CSAVE** ab
CNTL-Taste + C-Taste	BASIC druckt **CLEAR** ab
CNTL-Taste + E-Taste	BASIC druckt **EDIT** ab
CNTL-Taste + F-Taste	BASIC druckt **FOR I=1 TO** ab

CNTL-Taste + G-Taste	BASIC druckt **GOTO** ab
CNTL-Taste + H-Taste	BASIC druckt **GOSUB** ab
CNTL-Taste + J-Taste	BASIC druckt **RETURN** ab
CNTL-Taste + K-Taste	BASIC druckt **DIM** ab
CNTL-Taste + L-Taste	BASIC druckt **LIME** ab
CNTL-Taste + M-Taste	BASIC druckt **INPUT** ab
CNTL-Taste + N-Taste	BASIC druckt **NEXT I** ab
CNTL-Taste + O-Taste	BASIC druckt **DATA** ab
CNTL-Taste + P-Taste	BASIC druckt **POKE** ab
CNTL-Taste + Q-Taste	BASIC druckt **RESTORE** ab
CNTL-Taste + R-Taste	BASIC druckt **RUN** ab
CNTL-Taste + S-Taste	BASIC druckt **SOUND** ab
CNTL-Taste + T-Taste	BASIC druckt **CHR$(** ab
CNTL-Taste + U-Taste	BASIC druckt **LEFT$(** ab
CNTL-Taste + V-Taste	BASIC druckt **RIGHT$(** ab
CNTL-Taste + W-Taste	BASIC druckt **MID$(** ab
CNTL-Taste + X-Taste	BASIC druckt **CURSORX** ab
CNTL-Taste + Y-Taste	BASIC druckt **CURSORY** ab
CNTL-Taste + Z-Taste	BASIC druckt **SCREEN** ab

Fehlermeldungen
Wenn BASIC-80 einen Fehler im auszuführenden Programm feststellt, wird eine Fehlermeldung ausgegeben und das Programm stoppt. Eine Liste der Fehlermeldungen und Fehlerschlüssel finden Sie im Teil 5, Anhang B.

Programmzeilen
Es können mehr als ein Befehl in einer Programmzeile untergebracht werden, diese müssen aber durch das Symbol ":" voneinander getrennt werden. Zum Beispiel:

```
10 ? 3+5:? 10+3
```

In jeder Zeile stehen 39 Symbole, einschliesslich der 5-stelligen Zeilennummer zur Verfügung.
Alle Programmzeilen können durch Benutzung des Befehls "NEW" im Speicher gelöscht werden.

Kapitel 2

BASIC-80 arbeitet mit fünf Rechenarten, nämlich Addition (+), Subtraktion (-), Multiplikation (*), Division (/) und Potenzen (^). Diese Rechenarten können in einer Programmzeile kombiniert werden. Zum Beispiel:

```
? 40/5+3
```

ergibt das Resultat 11. Der Vorgang arithmetischer Operationen ist wie folgt:

1 Potenzen - z.B. 12^3,
2 Die Erzeugung negativer Zahlen - z.B. - 3,
3 Multiplikation und Division - z.B. 3*2, 4/2
4 Addition und Subtraktion - z.B. 3+2, 3-2

Die Rangfolge kann durch Benutzung von Klammern () geändert werden. Zum Beispiel:

```
? 40/(5+3)
```

ergibt das Resultat 5.

Versuchen Sie nun die folgende Eingabe:

```
? 1231.45+1.456
```

Nach Betätigen der "RET-Taste" erscheint folgender BASIC-Ausdruck:

```
1232.91
```

Beachten Sie, dass die Dezimalstellen nicht durch ein Komma (,), sondern durch einen Punkt angezeigt werden. (Britische und US-Schreibweise) Das Ergebnis des letzten Beispiels unterscheidet sich von dem, was man erwartet hat. BASIC-80 kann mit maximal 7-stelligen Zahlen arbeiten, es wurden jedoch nur 6 Stellen ausgedruckt. Um dieses zu erreichen, wurde das Ergebnis aufgerundet.

Zahlen zwischen 0.01 und 999.999 werden in gewohnter Weise ausgedruckt. Zahlen ausserhalb dieses Bereiches werden in wissenschaftlicher Schreibweise dargestellt. Sehr kleine oder sehr grosse Zahlen werden dann in Form von 10er-Potenzen dargestellt. Versuchen Sie folgende Eingabe:

```
? 234000000
```

Nach Betätigen der "RET-Taste" wird in BASIC-80 auf dem Bildschirm erscheinen:

```
2.34E+8
```

Dies ist das Gleiche wie 2.34* 10^8.

Kapitel 3

Der Speicher eines Computers kann nicht nur Programmzeilen, sondern auch andere Daten enthalten. Diese können sich aufteilen in numerische Daten, mit denen arithmetische Rechnungen durchgeführt werden und alphanumerische Daten. Diese beide Gruppen können in Konstanten (unveränderbare Daten) und Variablen unterteilt werden.

Ein Beispiel für eine numerische Konstante ist die Zahl 26, -13 oder 12.5. Eine alphanumerische Konstante ist beispielsweise "HELLO" oder "GOODBYE" oder "12345". Alphanumerische Konstanten unterscheiden sich von den numerischen Konstanten durch das Zeichen", das am Anfang und am Ende der Konstante stehen muss. Arithmetische Funktionen können nicht mit der alphanumerischen Konstanten "12345" durchgeführt werden.
Zum Beispiel:

```
? 6
```

stellt eine numerische Konstante dar; mit

```
? "HELLO"
```

wird eine alphanumerische Konstante ausgedrückt.

Eine Variable ist ein Teil des Speichers, der alphanumerischen oder numerischen Daten speichern kann. Jede Variable wird vom Programmierer mit einem Namen versehen. Der Name der Variablen drückt den Wert dieses speziellen Speicherteils aus.
Zum Beispiel:

```
AB = 13.5
BC$ = "HELLO"
```

Dies bedeutet, der Variablen AB ist der Wert von 13.5 und der Variablen BC$ der Wert "HELLO" zugeordnet worden. Das Zeichen $ hinter dem Namen der Variablen bedeutet, dass es sich um eine alphanumerische Variable handelt.

Eine numerische Variable kann maximal 7-stellig sein, eine alphanumerische Variable kann bis zu 30 Symbole enthalten. Es können so viele numerische oder alphanumerische Variablen verwandt werden, wie das Programm erfordert, jedoch maximal bis zum Erreichen der Speicherkapazitätsgrenze. Der Befehl "LET" wird benutzt, um der Variablen einen Wert zuzuweisen. Zum Beispiel:

```
10 LET A=10
20 LET B$="HELLO"
```

Das Wort LET kann wahlweise benutzt werden.
Wird eine dieser Variablen in einer arithmetischen Rechnung eingesetzt, so wird diese Rechnung mit dem Wert der Variablen durchgeführt. Zum Beispiel:

```
30 C=A*5
```

Wenn dieser Befehl ausgeführt wird, wird der Wert der Variablen "A" mit 5 multipliziert. Das Ergebnis wird dann als Variable "C" gespeichert, d.h. wenn "A" = 10 ist, wird "C" = 50. Arithmetische Operationen können nicht mit alphanumerischen Variablen durchgeführt werden. Alphanumerische Daten können jedoch verbunden werden. Zum Beispiel:

```
40 D$=B$+", GOODBYE"
```

Bei der Ausführung wird der Textvariable "B$" eine Konstante ", GOODBYE" hinzugefügt und das Ergebnis unter D$ gespeichert. Der Wert von D$ ist dann "HELLO, GOODBYE". Eine Variable kann auch mit dem Befehl "INPUT" einen Wert erhalten. Mit diesem Befehl erwartet der Computer, dass über die Tastatur Daten eingegeben werden. Nach Betätigen der "RET-Taste" werden die eingegebenen Daten der Variablen - die dem "INPUT"-Befehl folgt - zugeordnet. Zum Beispiel:

```
50 INPUT A
60 INPUT B$
```

Während in der Programmzeile 50 nur numerische Daten akzeptiert werden, werden in Programmzeile 60 alphanumerische Daten akzeptiert.

Namen der Variablen

Der Programmierer kann die Namen der Variablen bestimmen. Es gibt jedoch eine Anzahl von Bedingungen, die der Name einer Variablen erfüllen muss. Der Name muss immer mit einem Buchstaben des Alphabets (A-Z) beginnen, der Rest der Zeichen kann eine Mischung aus Buchstaben und Zahlen sein. Besteht ein Name aus mehr als 2 Zeichen, so werden jedoch nur die ersten beiden vom BASIC-80 erkannt.

Zum Beispiel: Die Namen "PO" und "POST" beziehen sich auf dieselbe Variable. Die Variable darf keinen Namen haben, der mit einem BASIC-Befehl identisch ist. So kann zum Beispiel der Name "TX" nicht genommen werden, da "TX" ein BASIC-Befehl ist. (siehe Teil 5, Anhang F)

Kapitel 4

Wenn in einem Programm eine Anzahl von konstanten Datenwerten benutzt werden, ist es ratsam, diese Werte mit Hilfe des "DATA-Befehls" in Tabellenform aufzulisten. Die Datenelemente werden mit dem "READ-Befehl" aus der Tabelle gelesen. Versuchen Sie folgende Eingabe:

```
NEW
10 READ X$:? X$
20 GOTO 10
30 DATA JAN,FEB,MAR
```

Bitte beachten Sie die verschiedenen Datenelemente im "DATA-Befehl", die durch Komma (,) voneinander getrennt sind.

Geben Sie nun ein:

```
RUN
```

Der Computer wird anzeigen:

```
JAN
FEB
MAR
? OD error in 10
```

Der "READ-Befehl" nimmt das erste folgende Datum aus der "DATA-Befehl-Tabelle". Mit dem GOTO-Befehl" geht das Programm zurück zur Programmzeile 10. Der Fehler trat auf, weil an vierter Stelle des "DATA-Befehls" keine Daten mehr zu lesen waren. Der "RESTORE-Befehl" fordert den "READ-Befehl" auf, die Daten des "DATA-Befehls" beginnend mit dem ersten Datenelement, zu lesen.

Versuchen Sie folgende Eingabe:

```
15 RESTORE
RUN
```

Das Programm beginnt nun "JAN" unendlich oft zu drucken. Unterbrechen Sie das Programm durch Drücken der "BREAK-Taste". Eine Tabelle für die Variablen kann auch mit dem "DIM-Befehl" aufgestellt werden. Zum Beispiel:

```
DIM A(20)
```

Das bedeutet, dass 21 Speicherplätze, 0-20 einschliesslich, für den Variablenamen "A" reserviert sind. Wenn diese Variable im Programm verwendet wird, reicht der Variablenname nicht mehr aus. Jeder der 21 Plätze muss bezeichnet werden. Dies geschieht dadurch, dass eine Zahl in Klammern hinter dem Namen der Variablen steht.
Versuchen Sie nun folgende Eingabe:

```
NEW
10 DIM A(8)
20 A(3)=15:B=0
30 ? A(8)
40 END
RUN
```

Der Computer wird "0" auf dem Bildschirm ausgeben. Das ist der Wert von A (0). Aendern Sie die Programmzeile 20 in:

```
20 A(3)=15:B=3
```

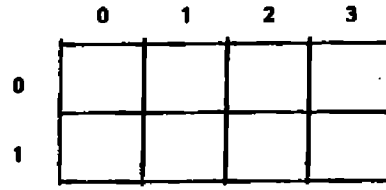
und lassen Sie das Programm noch einmal laufen. Der Computer wird nun "15" drucken, oder in anderen Worten den Wert von A (3).
Es ist auch möglich, eine alphanumerische Tabelle zu drucken. Zum Beispiel:

```
DIM B$(20)
```

Zweidimensionale Variable
BASIC erlaubt es, zweidimensionale Variablen zu reservieren. Nehmen Sie zum Beispiel den Befehl:

```
DIM A(1,3)
```

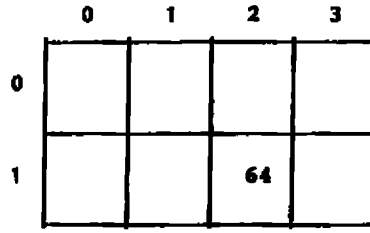
welcher den folgenden Speicherplätze reserviert.



Ist die Variable "A" in einem Programm zugeordnet worden, müssen die beiden Dimensionen zusammen mit dem Variablennamen festgelegt werden. Zum Beispiel der Befehl:

```
LET A(1,2)=64
```

Die Zahl 64 wird in Zeile 1, Spalte 2 des reservierten Feldes plaziert.



Kapitel 5

Der Verlauf eines Programms kann durch den Befehl "IF" variiert werden.
Geben Sie folgendes Beispiel ein:

```
NEW
10 INPUT A:INPUT B
20 IF A = B THEN ? "A = B"
30 IF A > B THEN ? "A > B"
40 IF A < B THEN ? "A < B"
50 END
```

In diesem Beispiel werden die Werte der zwei Variablen miteinander verglichen. Der Text auf dem Bildschirm entspricht den Werten der beiden Variablen. Der "IF-Befehl" fragt, ob eine Bedingung zutrifft. Trifft dies zu, wird der Befehl nach dem Wort "THEN" ausgeführt. Wenn nicht, wird der Befehl nach dem Wort "THEN" nicht ausgeführt.
Die folgenden Bedingungen können mit dem "IF-Befehl" festgelegt werden:

```
<   kleiner als
>   grösser als
=   gleich
< > ungleich
> = grösser oder gleich
< = kleiner oder gleich
```

Steht der "GOTO-Befehl" hinter dem Wort "THEN", muss das Wort "THEN" nicht weiter definiert werden. Versuchen Sie folgendes Beispiel:

```
NEW
10 CT=0
20 ? "PHILIPS"
30 CT=CT+1
40 IF CT<10 GOTO 20
50 END
RUN
```

Das Ergebnis dieses Programms ist, dass "PHILIPS" zehnmal nacheinander auf dem Bildschirm erscheint.
Dieses Beispiel kann durch die Benutzung des "FOR...NEXT-Befehls" vereinfacht werden.

```
NEW
10 FOR CT=1 TO 10
20 ? "PHILIPS"
30 NEXT CT
40 END
RUN
```

Dieses Programm produziert das gleiche Ergebnis wie das vorangegangene Beispiel. In Programmzeile 10 hat "CT" den Wert 1. Folglich wird "PHILIPS" in Programmzeile 20 ausgedruckt. In Programmzeile 30 wird "CT" um 1 erhöht. Wenn "CT" grösser als 10 ist, wird Zeile 40 ausgeführt. Wenn "CT" kleiner als 10 ist, wird der ganze Vorgang, beginnend mit Zeile 20, wiederholt.
Durch "NEXT" wird "CT" um den Wert 1 erhöht. Dieser Wert kann durch Gebrauch der "STEP-Möglichkeit" in dem "FOR-Befehl" verändert werden.
Aendern Sie Zeile 10 des letzten Beispiels in:

```
10 FOR CT=1 TO 10 STEP .5
```

und starten Sie das Programm.

Programm mit einem Menü
Programme, die mehrere Funktionen erfüllen können, haben oft am Anfang ein Anzeigen-Menü. Das gewünschte Programm wird gewählt, nachdem eine Zahl eingegeben wurde. Geben Sie folgendes Programm ein:

```
NEW
10 ? "WAHL-MENU"
20 ? "1 = Programmzeile 100"
30 ? "2 = Programmzeile 150"
40 ? "3 = Programmende"
50 INPUT "Ihre Wahl";A
60 IF A<1 GOTO 50
70 IF A>3 GOTO 50
80 ON A GOTO 100,150,200
100 ? " Dies ist Zeile 100":GOTO 10
150 ? " Dies ist Zeile 150":GOTO 10
200 END
RUN
```

In diesem Beispiel ist die Programmzeile 80 die interessanteste. Wenn der Wert der Variablen A gleich 1 ist, fährt das Programm in Programmzeile 100 fort. Wenn A gleich 2 ist, springt das Programm auf Zeile 150. Ist A gleich 3, wird mit der Zeile 200 fortgefahren.

Kapitel 6

Stellen Sie sich vor, dass folgendes Programm gespeichert wurde:

```
10 INPUT A:INPUT B
20 C=A*100/B:? A;
30 ? "IST";C;"PROZENT VON";
40 ? B
50 C=B*100/A:? B;
60 ? "IST";C;"PROZENT VON";
70 ? A
80 END
```

Programmzeile 60 ist die gleiche wie Programmzeile 30. Diese Programmzeile kann in einer separaten Programmzeile plaziert und in den Zeilen 30 und 60 mit dem Befehl "GOSUB" abgerufen werden. Das vorangegangene Beispiel würde dann so aussehen:

```
10 INPUT A:INPUT B
20 C=A*100/B:? A;
30 GOSUB 90
40 ? B
50 C=B*100/A:? B;
60 GOSUB 90
70 ? A
80 END
90 ? "IST";C;"PROZENT VON";
100 RETURN
```

In Zeile 30 springt BASIC auf Zeile 90 und macht von dort weiter. Bekommt es in Zeile 100 den "RETURN-Befehl", geht es zurück auf Zeile 40. BASIC springt ebenso von Zeile 60 auf Zeile 90. Bekommt es jedoch in Zeile 100 den "RETURN-Befehl", springt das BASIC zurück auf Zeile 70.

Die Befehle in den Programmierzeilen 90 und 100 nennt man Unterprogramme. Erreicht BASIC mittels "GOSUB-Befehl" die Zeile 90, speichert es, in welcher Zeile der Sprungbefehl stand; so kehrt es durch den "RETURN-Befehl" in die Programmzeile zurück, die dem "GOSUB-Befehl folgt.

Programmzeile 80 ist für das vorherige Beispiel sehr wichtig. Würde diese Zeile vergessen, würde Zeile 70 von Zeile 90 gefolgt. Zeile 100 würde dann einen "RG-Fehler" (Rückkehr ohne GOSUB) anzeigen.

Kapitel 7

Wenn das BASIC-80 gestartet wird, ist der Bildschirm hellblau. Die System-Service-Zeile hat weisse Buchstaben, die restlichen sind schwarz. Die Farbe des Bildschirms kann durch "INPUT-Befehle" verändert werden. Versuchen Sie folgendes Programm:

```
NEW
10 FOR I=1 to 5
20 FOR J=0 to 7
30 INIT J
40 NEXT J
50 NEXT I
60 INIT 6:END
RUN
```

Die Bildschirmfarbe nennt mann die Hintergrundfarbe. Die Farbe der Symbole auf dem Bildschirm nennt man die Vordergrundfarbe, die nach dem Start von BASIC-80 schwarz ist.
Die Vordergrundfarbe der Buchstaben kann mit dem "TX-Befehl" geändert werden. Geben Sie das folgende Programm ein und beobachten Sie die Vorder- und Hintergrundfarbe:

```
NEW
10 INIT 2:TX 7,0,0
20 ? " Dies ist weiss auf grün "
30 END
RUN
```

Der "TX-Befehl" bietet mehrere Möglichkeiten für das Ausdrucken von Buchstaben. Versuchen Sie diese Möglichkeiten und sehen Sie, was passiert. Doppelte hohe und doppelt breite Symbole benötigen zwei Positionen auf dem Bildschirm.
Zum Beispiel:

```
10 TX 0,2,0:? "W "
20 TX 0,3,0:? "H ";? "H "
30 TX 0,0,0:END
```

In Zeile 10 wird ein breiter Buchstabe "W" und in Zeile 20 ein hoher Buchstabe "H" auf dem Bildschirm erscheinen. Für einen breiten Buchstaben müssen 2 Zeichen gedruckt werden. Für einen hohen Buchstaben müssen 2 Zeilen mit dem gleichen Zeichen gedruckt werden. Zeile 30 sorgt dafür, dass die Buchstaben in normalen Grössen auf dem Bildschirm dargestellt werden. Um die Ausgangsfarben wieder zu erhalten, müssen die "CNTL-" und die "CLEAR-Taste" gleichzeitig gedrückt werden.

Bildschirmspeicher
Das C 7420 hat einen Speicherteil für die Ausgabe von Daten auf den Bildschirm. Diesen Speicherteil nennt man "Video Bildkarte". Mit dem "STORE-Befehl" können die Symbole in diesem Teil des Speichers gespeichert werden. Das Gegenteil von "STORE" ist der "DISPLAY-Befehl". Die Befehle "SCREEN" oder "LINE" werden eingesetzt, um die gespeicherten Symbole auf dem Bildschirm auszudrucken.
Geben Sie folgendes Beispiel ein:

```
NEW
10 IMIT 3:TX 0,0,0
20 FOR I=1 TO 20
30 ? " PHILIPS"
40 NEXT I
50 END
RUN
```

Merken Sie sich die Zeit, die der Computer braucht, um dieses Programm vollständig zu fahren. Wenn am Bildschirm "OK" erscheint, drücken Sie die "CNTL-" und die "CLEAR-Taste". Der Bildschirm ist nun gelöscht. Betätigen Sie nun die linke Action-taste, um das Programm auf dem Bildschirm auszudrucken.
Aendern Sie nun die Programmzeilen 10 und 40 in:

```
10 IMIT 3:TX 0,0,0:STORE
20 NEXT I:SCREEN:DISPLAY
```

Wenn das Programm mit RUN gestartet ist, werden alle Daten gleichzeitig ausgedruckt.

Graphische Symbole

BASIC-80 kann graphische Symbole genauso gut darstellen, wie alphanumerische. Der "GR-Befehl" dient diesem Zweck und kann ebenfalls die Vorder-und Hintergrundfarben ändern. Der "TX-Befehl" stellt sicher, dass alphanumerische Symbole nochmals auf dem Bildschirm dargestellt werden können. Die darstellbaren Symbole sind in Teil 5, Anhang D aufgelistet. Lassen Sie mit BASIC-80 folgendes Programm laufen und beobachten Sie die am Bildschirm dargestellten Symbole.

```
NEW
10 INIT 6
20 TX 0,0,0
30 ? "ABCDE"
40 GR 0,6,0
50 ? "ABCDE"
60 TX 0,0,0
70 END
RUN
```

Sie können auch selbst Symbole gestalten (siehe Anlage E, Teil 5).

Programmzeilen werden in schwarz gedruckt. Programmzeilen, die einen "REM-Befehl" beinhalten, werden in weiss dargestellt. Der "REM-Befehl" wird benutzt, um Anmerkungen zwischen den Programmzeilen einzusetzen; BASIC-80 führt den "REM-Befehl" nicht aus, beachtet ihn aber als Kommentar.

Kapitel 8

BASIC-80 ist auch mit einer Anzahl von Funktionen ausgestattet. Diese Funktionen werden in Rechengängen, unter Einbeziehung von Konstanten oder Variablen, eingesetzt. Diese funktionalen Konstanten oder Variablen nennt man "ARGUMENT". Funktionen werden in Verbindung mit Argumenten in den Befehlen benutzt. Zum Beispiel:

```
10 PRINT SIN(5)
```

oder

```
10 x=5
20 PRINT SIN(x)
```

Alle verfügbaren Funktionen sind in Teil 5, Kapitel 2 beschrieben. Diese Funktionen können folgendermassen unterteilt werden:

Mathematische Funktionen	ABS,ATN,COS,EXP,INT,LOG,RND,SGN,SIN,SQR,TAN
Druck-Funktionen	POS,SPC,TAB
Alphanumerische Funktionen	ASC,LEFT$,LEN,MID$,RIGHT$,STR$,VAL
Verbleibende Funktionen	ACTION,CHR$,FRE, PEEK,STICKX,STICKY,USR

Handsteuerhebel
Die Funktionen "ACTION", "STICKX" und "STICKY" dienen dazu, den Steuerhebel und die Steuerknöpfe zu aktivieren. Geben Sie folgendes Programm ein:

```
NEW
10 INIT 4:? CHR$(150);:STORE
20 XP=0 :YP=10:CURSORX XP:CURSORY YP
30 GR 1,4,0:? "]"; Line YP,YP
40 X=ACTION(1):Y=STICKY(1)
50 IF X=1 THEN GOSUB 100
60 IF Y=255 THEN GOSUB 200
```

```
70 IF Y=1 THEN GOSUB 300
80 GOTO 40
100 BRIGHT 1:SOUND 7
110 FOR XP=1 TO 39
120 CURSORX XP : ? "L"; LINE YP,YP
130 CURSORX XP : ? "@";
140 NEXT XP
150 LINE YP,YP:BRIGHT 0:XP=0:RETURN
200 CURSORX XP:? "@";
210 YP=YP-1:IF YP < 0 THEN YP=0
220 CURSORX XP : CURSORY YP
230 ? "]";: LINE YP,YP+1
240 RETURN
300 CURSORX XP:? "@";
310 YP=YP+1: IF YP > 22 THEN YP=22
320 CURSORX XP:CURSORY YP
330 ? "]";:LINE YP-1,YP
340 RETURN
RUN
```

Mit diesem Programm wird der Bildschirm völlig blau, mit einem roten Objekt auf der linken Seite. Wenn die Aktiontaste auf der rechten Seite gedrückt wird, feuert das Objekt einen Schuss ab. Wenn der rechte Steuerhebel zum Anwender hin bzw. nach unten gezogen wird, bewegt sich das Objekt an den unteren Bildschirmrand. Das Objekt bewegt sich zum oberen Bildrand, wenn der Steuerhebel vom Anwender weg bzw. nach oben geschoben wird.

Wie ist dieses Programm zusammengesetzt?
In Programmzeile 10 wählte man die blaue Hintergrundfarbe des Bildschirms durch den "INIT 4" Befehl. Der Befehl "PRINT CHR$ (150)" sorgt dafür, dass die System-Service-Zeile auf dem Bildschirm nicht sichtbar ist. Durch den "STORE-Befehl" wird alles, was dargestellt werden soll, gespeichert.
Mit der Programmzeile 20 wird der Cursor auf die Position 0 in Zeile 10 geschickt.
In Programmzeile 30 wird der Graphik-Modus durch die Anweisung "GR" aktiviert, der dafür sorgt, dass die Daten rot auf blauem Untergrund dargestellt werden. Das gewünschte Zeichen wird mit Hilfe des "PRINT-Befehls" auf der aktuellen Cursor-Position plaziert und durch den "LINE-Befehl" aufgeblendet.

In Programmzeile 40 werden die Daten, die die Position des Steuerhebels (Joystick) und der rechten Aktiontaste repräsentieren, den Variablen "X" und "Y" zugeordnet.
In Programmzeile 50 wird die Variable "X" mit dem Wert "1" eingesetzt. Wenn "X" gleich "1" ist, wird die rechte Aktiontaste gedrückt, und das Unter-programm der Zeilen 100 bis 150 einschliesslich werden ausgeführt.
In den Programmzeilen 60 und 70, wenn "Y" = 255 ist, bewegt man den rechten Steuerhebel zum Bediener hin, bzw. man zieht ihn nach oben, und das Unterprogramm der Zeilen 200 bis einschliesslich 240 wird ausgeführt. Wenn "Y" = 1 ist, bewegt man den rechten Steuer-hebel zum Bediener hin, bzw. nach unten und das Unterprogramm der Zeilen 300 bis einschliesslich 340 wird ausgeführt.
Eine ausführliche Beschreibung der Befehle finden Sie in Teil 5.

Kapitel 9

Der allgemeine Gebrauch eines Cassettenrecorders in Verbindung mit dem C 7420-Modul ist in Teil 3, Kapitel 5 beschrieben. Die von BASIC-80 gebotenen Möglichkeiten für die Datenspeicherung auf Cassette sind in diesem Kapitel beschrieben.
Um diese leichte Handhabung zu nutzen, verbinden Sie den Cassettenrecorder richtig mit dem C 7420-Modul und legen eine Leercassette ein.

Speichern eines BASIC-80-Programms auf Cassette
Drücken Sie die Aufnahme- und Wiedergabetaste (REC- u. PLAY-Taste) des Recorders, und geben Sie über die Tastatur ein:

```
CSAVE
```

Der Vorlaufstreifen der Cassette wird nach Drücken der "RET-Taste" vorgespult und BASIC antwortet auf dem Bildschirm mit "OK".
Geben Sie nun das folgende Programm ein:

```
NEW
10 REM Test
20 ? "Programmzeile 20"
30 ? "Programmzeile 30"
40 ? "Programmzeile 40"
50 END
CSAVE "TEST",20
```

Das Programm wird nun auf die Cassette überspielt. Die Zahl "20" in der CSAVE-Zeile zeigt die Programmzeile, von der an das noch einzulesende Programm gesichert werden soll. Daher überprüfen Sie zuerst mit folgenden Schritten ob das Programm einwandfrei auf
Das Programm wird nun auf die Cassette überspielt. Die Zahl "20" in der CSAVE-Zeile zeigt die Programmzeile, von der an das noch einzulesende Programm gesichert werden soll. Daher überprüfen Sie zuerst mit folgenden Schritten ob das Programm einwandfrei auf die Cassette überspielt wurde: Cassettenrecorder stoppen (STOP-Taste des Recorders drücken); Cassette zurückspulen und die Wiedergabetaste des Recorders drücken. Geben Sie nun ein:

```
CLOAD?
```

Nach Drücken der "RET-Taste" wird die Cassette eingelesen, und BASIC-80 schreibt am Bildschirm aus:

```
FOUND TEST
```

Beachten Sie die Lautstärkenregelung! Das Programm auf der Cassette wird mit dem Programm im Speicher verglichen. Ist das Programm nicht einwandfrei auf die Cassette überspielt worden, gibt BASIC-80 die Fehlermeldung"BAD FILE" aus. In diesem Fall muss der gesamte Ueberspielvorgang wiederholt werden. Wenn das Programm einwandfrei überschrieben wurde, druckt BASIC-80 "OK" aus.

Stoppen Sie den Cassettenrecorder, spulen Sie die Cassette zurück und drücken Sie die Wiedergabetaste. Geben Sie an der Tastatur ein:

```
NEW
```

Drücken Sie die "RET-Taste" und dann die linke Aktiontaste. Der Bildschirm ist gelöscht und der Speicher leer. Geben Sie ein:

```
CLOAD "TEST"
```

Nach Drücken der "RET-Taste" wird die Cassette eingelesen. BASIC-80 druckt aus:

```
FOUND TEST
```

Wenn das Programm eingelesen ist, startet das Programm in Programmzeile 20. Wird dasselbe Programm von der Cassette eingelesen mit

```
CLOAD "TEST",40
```

startet das Programm automatisch in Programmzeile 40, nachdem es eingelesen ist. Wird der Befehl

```
CLOAD "TEST",0
```

eingegeben, startet das Programm nach dem Einlesen nicht automatisch. BASIC-80 ist nun auf Befehlsebene. Cassettenprogramme können zu dem bereits gespeicherten Programm hinzugefügt werden, durch den Befehl "CLOADA". Spulen Sie die Cassette zurück, drücken Sie die Wiedergabetaste des Recorders und geben Sie an der Tastatur ein:

```
NEW
3 REM TEST PROGRAM
5 ? "Zeilen 3-7 bereits vorhanden"
7 CLOADA "TEST"
RUN
```

Sofort wird die Cassette eingelesen, das gespeicherte Programm wird, beginnend mit Programmzeile 20, durchlaufen. Drückt man die linke Aktionstaste, kann man feststellen, dass das Programm eingelesen ist und hinter dem bestehenden Programm plaziert wurde.

Bildschirm-Zeichnungen auf Cassette

BASIC-80 macht es möglich, Zeichnungen vom Bildschirm auf Cassette aufzunehmen und später wieder einzulesen mit dem Befehl "CSAVES". Spulen Sie die Cassette zurück, drücken Sie dann am Recorder die Aufnahme- und die Wiedergabetaste und geben Sie über die Tastatur ein:

```
NEW
10 INIT 2:STORE:TX 7,1,0
20 CURSORY 10:CURSORX 10
30 ? " Bildschirmbild auf Band";
40 CURSORY 11:CURSORX 10
50 ? " Bildschirmbild auf Band";
60 SCREEN:CSAVEL:CSAVES "PICTOR"
70 INIT 6
80 END
RUN
```

Die Video-Bild-Karte (siehe Kapitel 7) wird durch Programmzeile 60 auf die Cassette überspielt. Zum Ueberspielen des Programms von der Cassette in den Speicher sollten folgende Schritte durchgeführt werden: Cassette zurückspulen, die Wiedergabetaste am Recorder drücken und an der Tastatur eingeben:

```
NEW
10 CLOAD "PICTOR"
20 SCREEN:DISPLAY
30 END
RUN
```

Wenn der Befehl "CLOAD-PICTURE" gegeben ist, wird die Video-Bild-Karte mit den Daten auf Cassette gespeichert.
Achtung: Die Daten werden mit dem "SCREEN-Befehl" in Zeile 20 ausgedruckt.
Drücken Sie die "CNTL-Taste" und die "CLEAR-Taste".

Numerische Daten auf Cassette
Die Werte numerischer Variablen können auf Cassette in TABELLENFORM überspielt werden. Zum Beispiel:

```
NEW
10 DIM AA(20)
20 FOR I=0 TO 20
30 INPUT AA(I)
40 NEXT I
50 CSAVE:CSAVE* "NUM" AA
60 END
RUN
```

Achtung: in Programmzeile 50 ist der Name der Variablen hinter dem Dateinamen "NUM" definiert. Wenn diese numerischen Daten wieder eingelesen werden, muss zuerst die Tabelle mit dem "DIM-Befehl" definiert werden. Zum Beispiel:

```
NEW
10 DIM AA(20)
20 CLOAD "NUM" AA
30 FOR I=0 TO 20
40 ? AA(I)
50 NEXT I
60 END
RUN
```

Alphanumerische Daten auf Cassette
Die Werte alphanumerischer Variablen können mit dem "CSAVEX-Befehl" auf Cassette übertragen werden Zum Beispiel:

```
NEW
10 A$="ABCDEFGHIJKLMNO"
20 B$="ALFN":CSAVEL:CSAVEX B$ A$
30 END
RUN
```

Bevor diese Daten eingelesen werden können, muss zuerst die alphanumerische Variable, mit der gleichen Anzahl von Zeichen, mit der sie auch gespeichert wurde, definiert werden. Zum Beispiel:

```
10 A$="AAAAAAAAAAAAAAA"
20 B$="ALFN":CLOAD B$,A$
30 END
RUN
```

Stellen Sie sicher, dass die Variable A$ jeweils beim Ueberspielen von und zur Cassette 15 Zeichen enthält. Ueberprüfen Sie den Wert der Variablen A$ nach dem zweiten Durchlauf dieses Beispiels durch Drücken der linken Aktiontaste. Alphanumerische Daten können auch in der gleichen Weise wie die numerischen Daten auf Cassette überspielt werden. Ein Dateiname kann in den Namen der alphanumerischen Variablen einbezogen werden.
Diese Variable kann sich dannin den Befehlen CLOAD und CSAVE beziehen. Siehe Programmzeile 20 der beiden vorgegangenen Beispielen.

MUSTERPROGRAMM 1

Kapitel 10

Dieses Programm dient als Beispiel für das Aufstellen eines arithmetischen Programms in BASIC-80.
Der Zweck dieses Programms ist es, die jährliche Verzinsung eines Hypothekendarlehens auf monatlicher Basis zu berechnen. Zusätzlich wird der Steuervorteil über die Gesamtlaufzeit des Vertrages berechnet.

```
00010 REM **************
00020 REM Hauptprogramm
00030 REM **************
00040 GOSUB 1000
00050 GOSUB 2000
00060 GOSUB 3000
00070 GOSUB 4000
00080 GOSUB 5000
00090 IF OK$="J" GOTO 40
00100 IF OK$="j" GOTO 40
00110 INIT 6
00120 END
01000 REM ***********************
01010 REM Eingabe der Variablen
01020 REM ***********************
01030 INIT 7: STORE
01040 PRINT "***********************"
01050 PRINT "*                     *"
01060 PRINT "*Hypothekenprogramm*"
01070 PRINT "*                     *"
01080 PRINT "***********************"
01090 PRINT:SCREEN:DISPLAY
01100 INPUT "Hypothekenkapital ";KA
01110 IF KA<=0 GOTO 1100
01120 RE=KA:XC=KA
01130 INPUT "Beginn im Monat(MM)";MA
01140 IF (MA<1)OR(MA>12) GOTO 1130
01150 INPUT "Beginn im Jahr (JJJJ)";JA
01160 IF (JA<1)OR(JA>9999) GOTO 1150
01170 XA=JA
01180 INPUT "Laufzeit in Jahren";JL
```

```
01190 IF JL<>0 GOTO 1180
01200 ML=JL*12:XB=JL
01210 INPUT "Vertragslaufzeit(JJ)";CT
01220 IF (CT<=0)OR(CT>JL) GOTO 1210
01230 INPUT "jahreszinsen %";PJ
01240 IF PJ<=0 GOTO 1230
01250 MR=PJ/1200
01260 PRINT
01290 INPUT "Daten korrekt (J/N)";OK$
01300 IF OK$="J" GOTO 1330
01310 IF OK$="j" GOTO 1330
01320 GOTO 1090
01330 PRINT "Einen Moment bitte";
01340 RETURN
02000 REM **********
02010 REM berechnung
02020 REM **********
02030 C=0:A=1+MR
02040 FOR I=1 TO ML
02050 AA=A^I:B=1/AA:C=C+B
02060 NEXT I
02070 D=1/C:PRINT
02080 PRINT "Jährlicher Faktor" ;D
02090 RETURN
03000 REM ************************
03010 REM Jährlichen Zahlungsrate
03020 REM ************************
03030 E=KA*D
03040 PRINT "Monatliche Zahlungsrate";
03050 Z=E:GOSUB 14000:AN=Z
03060 PRINT AN
03070 INPUT "Bitte RET-Taste drücken";T$
03080 RETURN
04000 REM ********************
04010 REM Bildschirm anzeige
04020 REM ********************
04030 GT=0:FT=0
04040 GOSUB 6000
04050 JL=JL-1
04060 IF JL<=0 GOTO 4150
04070 P=12:CN=CT
04080 IF (MA=1)AND(CN=1) GOTO 4170
04090 CN=CN-1
04100 IF CN<=0 GOTO 4150
04110 FOR J=1 TO JL
04120 IF J=CN THEN J=JL
```

```
04130 GOSUB 7000
04140 NEXT J
04150 IF MA=1 GOTO 4170
04160 GOSUB 8000
04170 RETURN
05000 REM *********
05010 REM Abschluss
05020 REM *********
05030 PRINT:PRINT "weitere"
05040 INPUT "Berechnungen(J/N) ";OK$
05050 RETURN
06000 REM ***********
06010 REM Erstes Jahr
06020 REM ***********
06030 JA=JA-1:P=(12-MA)+1
06040 GOSUB 7000
06050 RETURN
07000 REM ********************
07010 REM Progression im Jahr
07020 REM ********************
07030 FOR K=1 TO P
07040 GOSUB 10000
07050 NEXT K
07060 GOSUB 11000
07070 GOSUB 12000
07080 RETURN
08000 REM **************
08010 REM Abschlussjahr
08020 REM **************
08030 P=MA-1
08040 FOR K=1 TO P
08050 GOSUB 10000
08060 NEXT K
08070 IF XE>CT GOTO 8090
08080 GT=RA:RE=0
08090 GOSUB 11000
08100 GOSUB 12000
08110 RETURN
10000 REM ************************
10010 REM Bestimmung der Zahlung
10020 REM ************************
10030 F=RE*MR:G=E-F:O=RE-G:RE=O
10040 FT=FT+F:GT=GT+G
10050 RETURN
11000 REM *************************
11010 REM Drucken der Ueberschrift
11020 REM *************************
```

```
11030 INIT 7:STORE
11040 PRINT "Hypothekenkapital  ";XC
11050 PRINT "Anfang     (MM-JJJJ)";MA
11060 PRINT "-";XA
11070 PRINT "Laufzeit in Jahren ";XB
11080 PRINT "Vertragslaufzeit   ";CT
11090 PRINT "Zinsen % pro Jahr  ";ZJ
11110 PRINT "Monatliche Zahlung ";AN
11120 PRINT
11130 RETURN
12000 REM *******************************
12010 REM Ausdruck eines vertragsjahres
12020 REM *******************************
12030 JA=JA+1
12040 IF (J=JL)AND(XB=CT) GOTO 12060
12050 GOTO 12090
12060 IF MA=1 GOTO 12080
12070 GOTO 12090
12080 GT=GT+RE:FT=FT-RE:RE=0
12090 B2=B1
12110 ST=B3:B2=B1-FT
12130 L=ST-B3
12140 PRINT "Jahr";TAB(23);JA
12150 PRINT "Schuld"; TAB(23);KA
12160 Z=GT:GOSUB 14000
12170 PRINT "Zahlung geleistet";
12180 PRINT TAB(23);Z
12190 Z=RE:GOSUB 14000
12200 PRINT "Schuldsaldo NCE";TAB(23);Z
12210 Z=FT:GOSUB 14000
12220 PRINT "Zinszahlung";TAB(23);Z
12250 M=L/P:N=E-M
12260 Z=N:GOSUB 14000
12280 PRINT:SCREEN:DISPLAY
12290 INPUT "Bitte RET-Taste drücken";T$
12300 GT=0:FT=0:KA=RE
12310 RETURN
14000 REM ********
14010 REM Abrunden
14020 REM ********
14030 Z=Z*100:Z=SGN(Z)*INT(ABS(Z))
14040 Z=Z/100
14050 RETURN
```

Kapitel 11

In diesem Programm wird der Cassettenrecorder zum Speichern und Ueberspielen von Daten auf Cassette benutzt. Zu diesem Zweck werden Daten über monatliche Einkünfte und Ausgaben eingegeben. Diese Daten können auf Cassette gespeichert werden, so dass sie beim nächsten Benutzen des Programms wieder eingelesen werden können.

Die Daten werden unter dem Dateinamen "MUDATA" auf Cassette gespeichert. Vergessen Sie nicht, ein leere Cassette einzulegen, wenn die Daten überspielt werden. Bevor die Daten wieder eingelesen werden können, muss die Cassette zum Anfang zurückgespult werden.

```
00010 REM Bandreservierung
00020 DIM AB(24)
00030 REM **************
00040 REM Hauptprogramm
00050 REM **************
00060 INIT 6:STORE
00070 PRINT"**********************
00080 PRINT"*                    *
00090 PRINT"*Cassettenprogramm*
00100 PRINT"*                    *
00110 PRINT"**********************
00120 PRINT:PRINT
00130 PRINT "1=Dateneingabe"
00140 PRINT
00150 PRINT "2=Lesen der Daten";
00160 PRINT "von der Cassette":PRINT
00170 PRINT "3=Graphische Darstellung"
00180 PRINT
00190 PRINT "4=Ueberspielen auf";
00200 PRINT "Cassette":PRINT
00210 PRINT "5=Programmende"
00220 PRINT:PRINT:SCREEN:DISPLAY
00230 INPUT "Sie können wählen";K
00240 IF K=5 THEN END
00250 IF (K<1)OR(K>4) GOTO 280
```

```
00260 ON K GOSUB 1000,2000,3000,4000
00270 GOTO 60
00280 PRINT "Wählen Sie bitte richtig"
00290 GOTO 230
01000 REM ************
01010 REM Dateneingabe
01020 REM ************
01030 INIT 6:STORE
01040 A=0:B=0
01050 FOR I=1 TO 12:A=A+AB(I):NEXT I
01060 FOR I=13 TO 24:B=B+AB(I):NEXT I
01070 PRINT TAB(5);"Einkommen %";
01080 PRINT TAB(22);"Ausgaben %"
01090 PRINT TAB(5);"———————— ——";
01100 PRINT TAB(22);"———————— ——"
01110 RESTORE
01120 FOR I=1 TO 12
01130 READ C$:PRINT C$;
01140 PRINT TAB(5);AB(I);TAB(15);
01150 GOSUB 5000
01160 I=I+12
01170 PRINT TAB(22);AB(I);TAB(31);
01180 GOSUB 6000
01190 I=I-12
01200 NEXT I
01210 PRINT TAB(5);"———————— ——";
01220 PRINT TAB(22);"———————— ——"
01230 PRINT "TOT";TAB (5);A;TAB(15);
01240 PRINT "100%";TAB(22);B;TAB(31);
01250 PRINT "100%":SCREEN:DISPLAY
01260 INPUT "Monatscode (0=Ende)";M
01270 MC=INT(M)
01280 IF MC>0 THEN RETURN
01290 IF (MC<1)OR(MC>12) GOTO 1260
01300 INPUT "Einkommen in DM";IK
01310 INPUT "Ausgaben in DM";UI
01320 INPUT "Stimmt (J/N)";OK$
01330 IF OK$="J" GOTO 1360
01340 IF OK$="j" GOTO 1360
01350 GOTO 1260
01360 AB(MC)=IK:MC=MC+12:AB(MC)=UI
01370 GOTO 1030
01500 DATA JAN,FEB,MAR,APR,MAI,JUN
01510 DATA JUL,AUG,SEP,OKT,NOV,DEC
02000 REM ************************
02010 REM Lesen von der Cassette
02020 REM ************************
02030 INIT 6
```

```
02040 PRINT "Datencassette einlegen"
02050 PRINT "und zum Anfang"
02060 PRINT "zurückspulen"
02080 PRINT
02090 PRINT "Starttaste am"
02100 PRINT "Cassettenrecorder drücken"
02110 PRINT "und RET-Taste am G7400 drücken"
02120 INPUT T$:POKE -30762,3
02130 CLOAD* "NUDATA" AB
02140 PRINT
02150 PRINT "Einlesen beendet"
02160 FOR I=1 TO 3000:NEXT I
02170 RETURN
03000 REM ************************
03010 REM Graphische Darstellung
03020 REM ************************
03030 INIT 3:PRINT " Einen Moment bitte"
03040 STORE:PRINT:T=4250
03050 FOR I=1 TO 16
03060 T=T-250
03070 IF T=500 THEN PRINT TAB(1);
03080 GOSUB 9000
03090 GOSUB 8000
03100 NEXT I
03110 PRINT TAB(6);
03120 FOR I=1 TO 26
03130 PRINT CHR$(126);
03140 NEXT I:PRINT
03150 PRINT TAB(7);
03160 PRINT "J F M A M J J A S O N D"
03170 PRINT
03180 TX 4,0,0:PRINT "^";
03190 TX 0,0,0:PRINT "=Einkommen";
03200 TX 1,0,0:PRINT "^";
03210 TY 0,0,0:PRINT "=Ausgaben";
03220 SCREEN:DISPLAY
03230 INPUT " Bitte RET-Taste drücken ";OK$
03240 RETURN
04000 REM ************************
04010 REM Auf Cassette Uebertragen
04020 REM ************************
04030 INIT 6
04040 PRINT "Leercassette einlegen"
04050 PRINT "und";
04060 PRINT "zum Anfang"
04070 PRINT "zurückspulen"
04080 PRINT
04090 PRINT "Aufnahme- und Wiedergabe-"
```

```
04100 PRINT "Taste am Recorder"
04110 PRINT "und die RET-Taste";
04120 PRINT "am G7400 drücken"
04130 INPUT T$
04150 CSAVEL
04150 CSAVE* "NUDATA" AB
04160 PRINT
04170 PRINT "Uebertragung beendet"
04180 PRINT "Die Cassette ist nun eine"
04190 PRINT "Datencassette"
04200 FOR I=1 TO 3000:NEXT I
04210 RETURN
05000 REM ****************************
05010 REM % Berechnung des Einkommens
05020 REM ****************************
05030 IF A=0 THEN Y=0:GOTO 5050
05040 Z=AB(I)*100/A:Y=INT(Z)
05050 PRINT Y;"%";
05060 RETURN
06000 REM ****************************
06010 REM % Berechnung der Ausgaben
06020 REM ****************************
06030 IF B=0 THEN Y=0:GOTO 6050
06040 Z=AB(I)*100/B:Y=INT(Z)
06050 PRINT Y;"%"
06060 RETURN
08000 REM *******
08010 REM Spalten
08020 REM *******
08030 PRINT TAB(6);CHR$(123);
08040 FOR J=1 TO 12
08050 TX 4,0,0
08060 GOSUB 8500
08070 J=J+12
08080 TX 1,0,0
08090 GOSUB 8500
08100 J=J-12:NEXT J
08110 TX 0,0,0:PRINT
08520 RETURN
08500 IF T<>250 GOTO 8520
08510 IF AB(J)>0 GOTO 8540
08520 IF AB(J)>T GOTO 8540
08530 PRINT "^";: GOTO 8550
08540 PRINT "^";
08550 RETURN
09000 REM **********************
09010 REM Vertikale Limitierung
09020 REM **********************
```

```
09030 IF T= 500 THEN PRINT T;
09040 IF T=1000 THEN PRINT T;
09050 IF T=1500 THEN PRINT T;
09060 IF T=2000 THEN PRINT T;
09070 IF T=2500 THEN PRINT T;
09080 IF T=3000 THEN PRINT T;
09090 IF T=3500 THEN PRINT T;
09100 IF T=4000 THEN PRINT T;
09110 RETURN
```

MUSTERPROGRAMM 3

Kapitel 12

Dieses Programm dient als Beispiel für die Darstellung von Zeichnungen und beweglichen Bildern im C7420 BASIC-80. Das Programm enthält spezielle Zeichen, die der Anwender mit Anhang E, Teil 6 definieren kann.

```
00010 REM **************
00020 REM Hauptprogramm
00030 REM **************
00040 GOSUB 3000
00050 GOSUB 4000
00060 GOSUB 1000
00080 GOSUB 5000
00090 GOSUB 6000
00100 GOTO 60
00200 REM **************
00210 REM Hilfsroutinen
00220 REM **************
00300 CURSORY 3:CURSORX Y
00310 PRINT LEFT$(A$,Z):LINE 3,3
00320 RETURN
00350 CURSORY 3:CURSORX Y
00360 PRINT LEFT$(E$,Z);:LINE 3,3
00370 RETURN
00400 CURSORY 3:CURSORX Y
00410 PRINT RIGHT$(F$,Z-1);:LINE 3,3
00420 CURSORY 6:CURSORX Y
00430 PRINT RIGHT$(D$,Z);
00440 CURSORY 5:CURSORX Y
00450 PRINT RIGHT$(C$,Z);
00460 CURSORY 4:CURSORX Y
00470 PRINT RIGHT$(B$,Z);
00480 CURSORY 3:CURSORX Y
00490 PRINT RIGHT$(A$,Z);:LINE 3,6
00495 IF X=18 THEN RETURN
00500 CURSORY 3:CURSORX Y
00510 PRINT RIGHT$(E$,Z);:LINE 3,3
```

```
00520 CURSORY 6:CURSORX Y
00530 PRINT RIGHT$(I$,Z);
00540 CURSORY 5:CURSORX Y
00550 PRINT RIGHT$(H$,Z);
00560 CURSORY 4:CURSORX Y
00570 PRINT RIGHT$(G$,Z);
00580 CURSORY 3:CURSORX Y
00590 PRINT RIGHT$(J$,Z);:LINE 3,6
00600 RETURN
00700 CURSORY 3:CURSORX Y-1
00710 PRINT LEFT$(F$,Z);:LINE 3,3
00720 CURSORY 6:CURSORX Y
00730 PRINT LEFT$(D$,Z);
00740 CURSORY 5:CURSORX Y
00750 PRINT LEFT$(C$,Z);
00760 CURSORY 4:CURSORX Y
00770 PRINT LEFT$(B$,Z);
00780 CURSORY 3:CURSORX Y
00790 PRINT LEFT$(A$,Z);:LINE 3,6
00800 CURSORY 3:CURSORX Y
00810 PRINT LEFT$(E$,Z);:LINE 3,3
00820 CURSORY 6:CURSORX Y
00830 PRINT LEFT$(I$,Z);
00840 CURSORY 5:CURSORX Y
00850 PRINT LEFT$(H$,Z);
00860 CURSORY 4:CURSORX Y
00870 PRINT LEFT$(G$,Z);
00880 CURSORY 3:CURSORX Y
00890 PRINT LEFT$(J$,Z);:LINE 3,6
00900 RETURN
01000 REM **********************
01010 REM Bildschirmaufteilung
01020 REM **********************
01030 INIT 6:PAGE:STORE:GR 2,2,0
01040 FOR I=17 TO 23
01050 CURSORY I:CURSORX 0
01060 PRINT CHR$(127);
01070 NEXT I:SCREEN
01080 DISPLAY:PRINT CHR$(150);
01090 RETURN
03000 REM ***************
03010 REM Spezialsymbole
03020 REM ***************
03030 CURSORY 22:CURSORX 39:PRINT " ";
03040 SETEG 032,"00000000000000000000
03050 SETEG 064,"FFFFFFFFFFFFFFFFFFFF
03100 SETEG 065,"FFFF0000000000000000
03110 SETEG 066,"FFFF0000000000FFFF
```

```
03120 SETEG 067,"FFFF030303030303FFFF
03130 SETEG 068,"FFFF000000000000F0FC
03140 SETEG 069,"00000102040810204080
03150 SETEG 070,"0000000000000000003F
03160 SETEG 071,"010307070F1F3F7FFFFF
03170 SETEG 072,"FFFFFF0000000C1E3F3F
03180 SETEG 073,"FFFFFF0F070321606060
03190 SETEG 074,"00C080F0F8FCFEFF7F3F
03200 SETEG 075,"0000000000000000C0C0
03210 SETEG 076,"0102040810204000000
03220 SETEG 077,"F07F3F1F0F0703010000
03230 SETEG 078,"00FFFFFFFFFFFFFF0000
03240 SETEG 079,"3FFFFFFFFFFFFFFF0000
03250 SETEG 080,"FFFFFFFFFFFFFFFFFF7F
03260 SETEG 081,"1E0C4F4FFFFFFFFFFFFF
03270 SETEG 082,"60C08000FFFFFFFFFFFF
03280 SETEG 083,"1F1F1F3FFFFFFFFFFFFF
03290 SETEG 084,"E0F0F0F0F0F0C0800000
03300 SETEG 085,"3F1F0F0700000000FFFF
03310 SETEG 086,"FFFFFFFF0303030303FF
03320 SETEG 087,"FFFFFFFF00000000FFFF
03330 SETEG 088,"FFFFFFFECDCCCCCCFFFF
03340 SETEG 089,"F0CD000000000030FFCF0
03350 SETEG 090,"0000000000C000000000
03400 SETEG 049,"0303000000000000FFFF
03410 SETEG 050,"00000000000000000F0FC
03420 SETEG 051,"00000000000000000F0F
03430 SETEG 052,"3F3F000000000000FFFF
03440 SETEG 053,"F0F0000000000000FFFF
03450 SETEG 054,"0000000000000000080C0
03500 SETEG 097,"0F0F0000000000000000
03510 SETEG 098,"FFFF0000000000000F0F
03520 SETEG 099,"FFFF303030303030FFFF
03530 SETEG 100,"FFFF0000000000080C0
03540 SETEG 101,"F0F0000000000000000
03550 SETEG 102,"00000000000001020408C
03560 SETEG 103,"0000102040800000000
03570 SETEG 104,"00000000103070F3FFF
03580 SETEG 105,"1F3F7FFFFFFFFFFFFF
03590 SETEG 106,"FFFFFFF0F0F0F0F1F1F3
03600 SETEG 107,"FFFFFF000000C0E0F1F3
03610 SETEG 108,"F0FCFEFF7F3F1F8F8703
03620 SETEG 109,"00000000080C0E0F0F8FC
03630 SETEG 110,"0000000001020400000
03640 SETEG 111,"1F274381000000000000
03650 SETEG 112,"00FFFFFFFF7F3F1F0000
03660 SETEG 113,"03FFFFFFFFFFFFFF0000
03670 SETEG 114,"FFFFFFFFFFFFFFFFF0F07
```

```
03680 SETEG 115,"F1F0F1F1FFFFFFFFFFFF
03690 SETEG 116,"E6CCF8F0FFFFFFFFFFFF
03700 SETEG 117,"01010103FFFFFFFFFFFF
03710 SETEG 118,"FEFFFFFFFFFFFCF8F0CC
03720 SETEG 119,"03010000000000000F0F
03730 SETEG 120,"FFFFFF7F0000000000FFFF
03740 SETEG 121,"FFFFFFFF30303030FFFF
03750 SETEG 122,"FFFFFFFF0CCCCCCCFFFF
03760 SETEG 058,"FFFCF0CC000000000FFFF
03770 SETEG 059,"00000000000C3CF0CC000
03900 RETURN
04000 REM *************
04010 REM Rubeschrauber
04020 REM *************
04100 A$="    AAAABCB2      "
04110 B$=" E    FG@@HLJK    "
04120 C$="LMBBBBOP@@ORST    "
04130 D$="        UVWXYZ    "
04140 E$="          lCBDAAAA "
04200 F$="          34cBdAAAe"
04210 G$=" fg    hi@jklm    "
04220 H$="nopMMMqr@@stuv    "
04230 I$="        wxyWz:;    "
04240 J$="    aAAAbBc56      "
04900 RETURN
05000 REM ***************
05010 REM Poster drucken
05020 REM ***************
05030 GR 0,6,0:STORE:CURSORY 16
05040 CURSORX 25:PRINT "U"
05050 CURSORX 33:PRINT "j";:SCREEN
05060 CURSORY 13:CURSORX 22:DELIM 7,7,0
05070 CURSORX 37:DELIM 6,6,0
05080 CURSORY 14:GOSUB 5500
05090 CURSORY 15:GOSUB 5500
05100 CURSORY 16:CURSORX 7:DELIM 7,7,0
05110 CURSORX 15:DELIM 6,6,0
05120 CURSORY 14:CURSORY 23:TX 0,0,0
05130 PRINT "  VIDEOPAC+  ";
05140 CURSORY 15:CURSORX 8
05150 PRINT "C-7420";:SCREEN
05160 RETURN
05500 CURSORX  7:DELIM 7,7,0
05510 CURSORX 15:DELIM 6,6,0
05520 CURSORX 22:DELIM 7,7,0
05530 CURSORX 37:DELIM 6,6,0
05540 RETURN
06000 REM ************************
```

```
06010 REM Fliegender Hubschrauber
06020 REM ************************
06030 EG 1,6,0
06040 FOR X=1 TO 18
06050 Y=0:Z=X:GOSUB 400
06060 NEXT X
06070 GOSUB 11000:GOSUB 720
06080 EG 1,6,0
06090 FOR X=1 TO 39
06100 Y=X:Z=40-X:GOSUB 700
06110 NEXT X
06120 RETURN
11000 REM *****************
11010 REM Poster aufnehmen
11020 REM *****************
11040 FOR I=7 TO 13:J=1
11050 EG 1,6,0:GOSUB 350
11060 J=J+1:IF J < 5 GOTO 11110
11070 TX 0,0,0:CURSORY I
11080 CURSORX  9:PRINT "I";
11090 CURSORX 12:Print "I";
11100 LINE I,I
11110 EG 1,6,0:GOSUB 300
11120 IF J < 5 GOTO 11050
11130 NEXT I
11200 FOR I=13 TO 15
11210 EG 1,6,0:GOSUB 350
11220 CURSORY I:CURSORX 7:DELIM 7,7,0
11230 CURSORX 15:DELIM 6,6,0
11240 EG 1,6,0 : GOSUB 300
11250 NEXT I
11270 CURSORY 16:CURSORX 7
11280 TX 6,0,0:PRINT " ";
11290 GR 0,6,0:PRINT "U";
11300 TX 6,0,0:PRINT "     ";
11310 GR 0,6,0:PRINT "j";
11320 TX 6,0,0:PRINT " ";
11330 EG 1,6,0:GOSUB 350
11340 TX 0,4,1:CURSORY 14
11350 CURSORX 8:PRINT "C-7420";
11360 CURSORX 23:PRINT "  Videopac+  ";
11370 TX 7,0,0
11380 CURSORY 13:CURSORX 8
11390 PRINT "       ";
11400 CURSORY 15:CURSORX 8
11410 PRINT "       ";
11420 LINE 13,16
11430 EG 1,6,0:GOSUB 300
```

```
11500 FOR I=12 TO 7 STEP -1:J=1
11510 EG 1,6,0:GOSUB 350
11520 J=J+1:IFJ<5 GOTO 11570
11530 TX 0,0,0:CURSORY I
11540 CURSORX 9:PRINT " ";
11550 CURSORX 12:PRINT " ";
11560 LINE I,I
11570 EG 1,6,0:GOSUB 300
11580 IF J<5 GOTO 11510
11590 NEXT I
11900 RETURN
```

TEIL

BASIC-ANLEITUNG

Kapitel 1

Alle BASIC-80 Befehle und Kommandos sind in diesem Kapitel beschrieben. Jede Beschreibung ist nach folgenden Schema gegliedert :

Format Zeigt die korrekte Form des Befehls (siehe nachstehende Format-bezeichung).

Anwendung Zeigt, wozu der Befehl benutzt wird.

Anmerkung Beschreibt detailliert, wie der Befehl angewandt wird.

Beispiel Zeigt Musterprogramme oder Programmteile, die den Gebrauch des Befehls demonstrieren.

Formatbezeichnung
Wo immer das Format für ein Kommando oder Befehl gegeben ist, sind die folgenden Regeln anzuwenden:

1 Teile in Grossbuchstaben müssen, wie gezeigt, eingegeben werden

2 Teile in Kleinbuchstaben, eingeschlossen in spitzen und Klammern (<>) müssen durch den Anwender eingesetzt werden.

3 Teile in eckigen Klammern ([]) sind freigestellt.

4 Alle Satzzeichen, ausser spitzen und eckigen Klammern, (zum Beispeil Kommata, runde Klammern, Semikolon, Bindestrich, Gleichheitszeichen) müssen dort eingesetzt werden, wo sie gezeigt werden.

5 Teile in elliptischen Klammern (.....) können so oft wie möglich, bis zur Länge einer Zeile, wiederholt werden.

1.1 BRIGHT

Format BRIGHT <I>

Anwendung Änderung der Helligkeit des Bildschirms.

Anmerkung I muss eine ganze zahl sein
0 = halbe Intensität
1 = volle Intensität

Beispiel
```
10 IF ACTION(1)=1 THEN BRIGHT 1
20 BRIGHT 0
```

Wenn die rechte Actiontaste gedrückt wird, erreicht die Helligkeit des Bildschirms die grösstmögliche Intensität.

1.2 CLEAR

Format CLEAR [<I>,<J>]

Anwendung Alle numerischen und alle String variablen werden auf Null gesetzt, und die Zeichenfolge wie auch das Speicherende werden auf SPACE gesetzt.

Anmerkung I muss ein ganzer Ausdruck sein und setzt die Zeichenfolge auf SPACE (löscht sie). Die Vorgabe liegt bei 50 Bytes.
J ist ein Speicherplatz, welcher wenn festgelegt, den höchsten verfäugbaren Speicherplatz im BASIC-80 setzt.
J muss kleiner als 32768 sein, Speicherplätze über 32768 können benutzt werden, wenn J eine negative Zahl zugeordnet wird.
Der Wert von J errechnet sich, indem man 65536 von der gewünschten Adresse abzieht. Ein Beispiel, um das Speicherende auf die Adresse 45000 zu setzen:
J = 45000 - 65536 = 20536

Beispiel **CLEAR 2000,-20480**

löscht die Grösse der Zeichenfolge auf 2000 Byte; die höchste von BASIC-80 benutzte Speicheradresse ist 45056.

1.3 CLOA

Format

```
CLOAD [<"Dateiname">][,<I>]
CLOAD <"Dateiname"> <String Variablen-name>
CLOAD* <"Dateiname"> <Array-name>
CLOADA [<"Dateiname">][,<I>]
CLOAD? [<"Dateiname">]
```

Anwendung

Einen Datensatz von einer Cassette einzulesen.

Anmerkung

"Dateiname" ist der Ausdruck, der beim Ueberspielen (mit CSAVE) auf die Cassette festgelegt wird. Wenn der "Dateiname" (filename) fehlt, wird das erste abgelegte Programm auf der Cassette eingelesen.
"I" ist die Zeilenummer, von der aus das Programm nach dem Speichervofgang gestartet wird. Enthält die Ablage ein Programm, so führt CLOAD erst einen "NEW"-Befehl aus, bevor das Programm gelesen wird. Ist die Datei eine in Maschinencode geschriebene Datei, so werden die Daten mit denselben Speicheradressen wie beim Ueberspielen (mit CSAVE) auf die Cassette gespeichert. Ist die Datei eine Bildschirmdatei, werden die Daten in der Bildschermspeicherkarte gespeichert. Enthält die Datei eine Zeichenfolge, werden die Daten in eine Stringvariable geladen und mit einem "Stringvariablennamen" versehen, wenn die Daten (mit CSAVE) überspielt wurden. Stellen Sie sicher, dass die Variablen vor dem Speichern reserviert worden sind.
CLOAD* speichert die Daten in einer Ordnung, die mit einem "Ordnungsnamen" bezeichnet wurden, als sie auf Cassette gespeichert wurden. Diese "Ordnung" kann eine numerische oder eine Textordnung sein. Stellen Sie sicher, dass die Ordnung dimensioniert wurde, bevor sie gespeichert wird.
CLOADA bringt das zu speichernde Programm hinter das aktuelle Programm im Arbeitsspeicher.
CLOAD? vergleicht das gespeicherte Programm mit dem Programm auf der Cassette. Sind die beiden Programme gleich, gibt BASIC ein "OK" aus; stimmt es nicht überein, erscheint ein "BAD FILE".

Beispiel

```
CLOAD "PROGA",500
```

Das Programm "PROGA" wird gespeichert und mit der Ausführung bei Zeile 500 begonnen.

1.4 CONT

Format CONT

Anwendung Dient zur Fortsetzung der Programmausführung nach dem Betätigen der "BREAK"-Taste oder ein nachdem ein "STOP"- oder "END"-Befehl ausgeführt worden ist.

Anmerkung Die Weiterführung des Programms findet an der Stelle statt, an der die Unterbrechung eintrat. Wenn die Programmunterbrechung durch eine INPUT-Anweisung erfolgt, wird die Ausführung nach der gewünschten Eingabe fortgesetzt.

"CONT" wird normalerweise in Verbindung mit "STOP" zur Fehlerkorrektur benutzt. Wenn die Ausführung gestoppt ist, können die dazwischenliegenden Werte mit Hilfe direkter Befehle überprüft und geändert werden. Das Programm kann mit "CONT" oder dem direkten Befehl "GOTO", hierbei wird in der angegebenen Zeile wieder begonnen, fortgesetzt.

Beispiel Siehe Beispiel 'STOP'

1.5 CU

Format CURSORX <I>

Anwendung Um den CURSOR an eine definierte Stelle in der augenblicklich angezeigten Zeile auf dem Bildschirm zu senden.

Anmerkung "I" ist eine Spaltennummer und muss mit einer ga Zahl zwischen 0 und 39 bezeichnet sein.

Beispiel CURSORX 5

Der CURSOR wandert auf die Position 5 der augenblicklich angezeigten Zeile.

1.6 CURSORY

Format CURSORY <I>

Anwendung Um den Cursor an eine bestimmte Position in der momentan angezeigten Spalte zu senden.

Anmerkung "I" ist eine Zeilennummer und muss mit einer ganzen Zahl zwischen 0 und 22 bezeichnet werden.

Beispiel CURSORY 10

Der Cursor wandert auf die Position 10 der momentan angezeigten Spalte.

1.7 CSA

Form CSAVE <"Dateiname">[,<I>]
CSAVEM <"Dateiname">,<S>,<T>[,<I>]
CSAVEL
CSAVES <"Dateiname">
CSAVE* <"Dateiname"> <Array-name>
CSAVEX <"Dateiname"> <String Variablen-name>

Anwendung Um eine Datei auf eine Cassette zu speichern.

Anmerkung "CSAVE" speichert das momentan im Speicher befindliche Programm auf Cassette. "Dateiname" ist eine "String" bezeichnung "I" ist die Zeilennummer, von der aus das Programm nach dem Einlesen startet.
"CSAVEX" speichert eine bestimmte Springvariable auf Cassette.
"CSAVE*" speichert ein bestimmtes Feld auf Cassette. Das Feld kann numerisch sein oder Strings enthalten. Die Elemente eines mehrdimensionalen Feldes (array) werden am schnellsten über den Index geändert.
"CSAVEM" speichert den momentanen Speicherinhalt in Maschinencode, "S" ist die Adresse des ersten zu speichernden Bytes,und "T" die Anzahl der zu speichernden Bytes.
"CSAVES" überträgt die aktuelle Bildschirmspeicherkarte auf Cassette.
"CSAVEL" überspringt das Einfädelstück der Cassette.

Beispiel CSAVE "PROGA"

Speichert das momentan im Speicher befindliche Programm unter dem Dateinamen "PROGA" auf Cassette.

1.8 DATA

Format DATA <Listung der Konstanten>

Anwendung Um die numerischen und Zeichenkonstanten zu speichern, die mit dem READ-Befehl eingelesen werden sollen (siehe READ).

Anmerkung DATA-Anweisungen sind nicht ausführbar und können irgendwo im Programm plaziert werden. Eine DATA-Anweisung kann beliebig viele Konstanten auf einer Zeile beinhalten (durch Kommata voneinander getrennt), und beliebig viele DATA-Anweisungen können im Programm benutzt werden. Durch den READ-Befehl werden die Daten geordnet durch Zeilennummern, nacheinander gelesen, so dass man sich die hierin enthaltenen Daten als eine fortlaufende Liste von Posten vorstellen kann, unabhängig davon, wieviele Posten auf einer Zeile im Programm angeordnet sind.

Die Liste der Konstanten kann numerische Konstanten enthalten (Numerische Ausdrücke sind in der liste nicht erlaubt.) Die Zeichenkonstanten im DATA-Befehl müssen in doppelte Anführungszeichen ("...") gezetzt werden, wenn Sie Kommata, Doppelpunkte, Ausrufungszeichen oder Leertasten enthalten. Anders werden Anführungszeichen nicht verwandt.

Die Typ der Variablen (numerisch oder Zeichen), im READ-Befehl vorgegeben, muss mit dem im DATA-Befehl übereinstimmen.

DATA-Befehle können vom ersten Element beginnend, durch den RESTORE-Befehl wieder eingelesen werden.

Beispiel Siehe Beispiel "READ-Befehl"

1.9 DEF

Format

DEF FN<Name>[(<Parameter-Liste>)]=
<Funktionsdefinition>

Anwendung

Um eine vom Anwender geschriebene Funktion zu bezeichnen und zu definieren.

Anmerkung

<Name> muss ein gültiger Variablenname sein. Dieser Name, FN vorangestellt, wird er Name der Funktion. <Parameter-Liste> ist ein Variablename innerhalb der <Funktionsdefinition> der bei Aufruf der Funktion zu ersetzen ist.

Funktionsdefinition ist ein Ausdruck, der für die Operationen der Funktion steht. Er ist auf eine Variablenamen, die in diesem Ausdruck erscheinen, dienen nur zur Definition der Funktion; Programmvariable mit dem gleichen Namen werden nicht beeinflusst. Ein Variablenname, der in einer Funktionsdefinition gebraucht wird, kann, muss aber nicht, in der Parameter-liste erscheinen. Ist der Name in der Parameter-Liste enthalten, so wird der Parameterwert bei Aufruf der Funktion versorgt. Andernfalls wird der aktuelle Wert der Variablen benutzt.

Vom Anwender können keine String-Funktionen bestimmt werden. Wenn ein Typ im Funktionsnamen spezifiziert ist,wird der Wert der Aussage diesem Typ zugeordnet, bevor er zum Aufrufbefehl zurückgestellt wird. Wenn das Argument mit dem im Funktionsnamen spezifizierten Typ nicht übereinstimmt, erscheint der Fehlermeldung "TYPE MISMATCH". Ein "DEF FN"-Befehl muss durchgeführt werden, bevor die Funktion, die ihn definiert, aufgerufen werden kann. Wird eine Funktion vor ihrer Definition aufgerufen, erscheint die Fehlermeldung "UNDEFINED USER FUNCTION". "DEF FN" darf nicht im Direkt-Modus benutzt werden.

Beispiel

```
410 DEF FNAB(X)=X^3
420 T=FNAB(5)
```

Zeile 410 definiert durch FN die Funktion AB. In Zeile 420 wird die Funktion aufgerufen.

1.10 DELIM

Format DELIM <I,J,K>

Anwendung Zeichen auf dem Bildschirm andere Farben zu geben.

Anmerkung I ändert die Vordergrundfarbe und muss eine ganze Zahl beinhalten.

0 = schwarz
1 = rot
2 = grün
3 = gelb
4 = blau
5 = purpur
6 = hellblau
7 = weiss

J ändert die Hintergrundfarbe von der Cursorposition an bis zum Ende der Zeile und muss eine ganze Zahl beinhalten.

0 = schwarz
1 = rot
2 = grün
3 = gelb
4 = blau
5 = purpur
6 = hellblau
7 = weiss

K muss eine ganze Zahl sein

0 = normal
1 = gesperrt gedr.
2 = gesperrt gedr.
3 = gesperrt gedr.
4 = unterstrichen
5 = unterstrichen und gesperrt
6 = unterstrichen und gesperrt
7 = unterstrichen und gesperrt

Die DELIM-Anweisungsetzt immer den Zeichencode 127 auf die aktuelle Cursorposition.
Dieser Befehl kann nur im Text-Modus benutzt werden. Die Textzeichen von der DELIM-Position ab bis zum Ende der Zeile erhalten die speziellen DELIM-Zeichen.

Beispiel

```
10 FOR LN=0 TO 9
20 CURSORY LN:CURSORX 0
30 DELIM 4,4,0
40 NEXT LN
```

Die ersten 10 Zeilen werden blau

1.11 DIM

Format DIM<Liste der beschreibenden Variablen>

Anwendung Dient zur Festlegung der Maximalgröss von array-Variablen, ihrem Subscript (Index) und der Zuweisung des entsprechenden Speicherplatzes.

Anmerkung Wird eine array-Variable ohne DIM-Anweisung benutzt, so wird der Maximalgrösse seines Subscript (en) 10 sein. Wird ein Subscript grösser als das angegebene Maximum benutzt, so erscheint die Fehlermeldung: "SUBSCRIPT OUT OF RANGE". Der kleinste Wert für ein Subscript ist immer 0.

Der DIM-Befehl setzt alle Elemente der beschriebenen Arrays auf den Anfangswert 0.

Beispiel

```
10 DIM A(20)
20 FOR I=0 TO 20
30 READ A(I)
40 NEXT I
```

1.12 DISPLAY

Format DISPLAY

Anwendung Alle gedruckten Zeichen werden sofort auf dem Bildschirm angezeigt.

Anmerkung Die gegenteilige Funktion zu "DISPLAY" ist "STORE". Die Anweisung für das Ausdrucken von Zeichen ist "DISPLAY".

Beispiel Siehe "STORE"

1.13 END

Format END

Anwendung Die Programmausführung zu beenden und auf Befehlsebene zurückzukehren.

Anmerkung END-Befehle können an jeder Stelle des Programms eingesetzt werden, um die Ausführung zu beenden. Im Gegensatz zum STOP-Befehl wird keine "BREAK" Meldung gebracht. Ein END-Befehl am Ende eines Programms ist möglich. Nach der Ausführung des END-Befehls ist BASIC-80 wieder auf Befehlsebene.

Beispiel

```
520 IF K>1000 THEN END
```

1.14 FOR...NEXT

Format

FOR <Variable> = x TO y [STEP z]
.
.
.
NEXT <Variable>
(x, y, z sind numerische Ausdrücke)

Anwendung

Eine Reihe von Anweisungen kann in einer Schleife eine vorgegebene Anzahl von Durchläufen machen.

Anmerkung

<Variable> wird als Zähler benutzt. Der erste numerische Ausdruck(x) ist der Anfangswert für den Zähler. Die auf den FOR-Befehl folgenden Programmzeilen werden ausgeführt, bis der NEXT-Befehl erreicht ist. Dann wird der Zähler unbedingt durch STEP bestimmte Schrittweite erhöht. Ein Vergleich wird durchgeführt, um festzustellen, ob der Wert des Zählers nun grösser ist als der Endwert (y). Ist dies nicht der Fall, geht BASIC-80 zurück zu der en Zeile, die auf den FOR-Befehl folgt, und der durchlauf beginnt von neuem.
Ist der Zahlenwert grösser, wird das Programm mit der auf den NEXT-Befehl folgenden Zeile fortgesetzt. Dies ist eine 'FOR.... NEXT'-Schleife. Wird STEP nicht festgelegt, so ist das Schrittmass 1.
Ist STEP negativ, so ist der Endwert des Zählers auf einen kleineren Wert als den Anfangswert zu setzen. Der Zähler zählt bei jedem Schleifendurchlauf rückwärts. Die Schleife ist beendet, wenn der Zähler einen Wert erreicht, der kleiner als der Endwert ist.

Die Schleife wird übersprungen, wenn der Anfangswertes und das Produkt aus Anzahl der Schleifendurchgänge mal Schrittweite grösser als der Endwert wird.

Verschachelte Schleifen

FOR...NEXT-Schleifen können verschachelt werden, d.h., eine FOR...NEXT-Schleife kann innerhalb einen anderen FOR...NEXT-Schleife auftreten. Wenn Schleifen ineinander verschachtelt werden, muss jede Schleife einen eigenen Variablen namen für den Zähler haben. Der NEXT-Befehl für die innere Schleife muss vor der äusseren Schleife erscheinen.

Wird der NEXT-Befehl vor dem zugehörigen FOR-Befehl eingesetzt, dann wird eine Fehlermeldung "NEXT WITHOUT FOR" angezeigt, und das Programm wird unterbrochen.

Beispiel 1

```
10 K=10
20 FOR I=1 TO K STEP 2
30 PRINT I;
40 K=K+10
50 PRINT K
60 NEXT I
RUN
1  20
3  30
5  40
7  50
9  60
OK
```

Beispiel 2

```
10 J=0
20 FOR I=1 TO J
30 PRINT I
40 NEXT I
```

In diesem Beispiel wird die Schleife nicht ausgeführt, weil der Anfangswert den Endwert übersteigt.

Beispiel 3

```
10 I=5
20 FOR I = 1 TO I+5
30 PRINT I;
40 NEXT I
RUN
 1 2 3 4 5 6
OK
```

In diesem Beispiel wird die Schleife 6 mal durchlaufen. Der Anfangswert der Schleife wird immer vor den Endwert gesetzt.

1.15 GOSUB ... RETURN

Format

```
GOSUB <Zeilennummer>
.
.
.
RETURN
```

Anwendung Verzweigen zu einem Unterprogramm und das Wiederzurückkehren.

Anmerkung <Zeilennummer>ist die erste Zeile des Unterprogramms. Ein Unterprogramm kann beliebig oft in einem Programm und auch von einem anderen Unterprogramm aus aufgerufen werden. Die Verschachtelung von Unterprogrammen wird nur durch die Speicherkapazität begrenzt.

Der RETURN-Befehl in einem Unterprogramm bewirkt die Verzweigung zurück auf die Programmzeile, die dem GOSUB-Befehl folgt. Ein Unterprogramm kann mehr als einen RETURN-Befehl enthalten, wenn die Logik eine Rückkehr von verschiedenen Punkten des Unterprogramms vorschreibt. Unterprogramme können an jeder Stelle des Programms auftreten, es ist aber empfehlenswert, das Unterprogramm deutlich vom Hauptprogramm zu trennen. Um einen unerwünschten Eintritt in ein Unterprogramm zu verhindern, kann ein STOP-, END- oder GOTO-Befehl eingesetzt werden, um das Programm zu umgehen.

Beispiel

```
10 GOSUB 40
20 PRINT"ZURUCK VOM UNTERPROGRAMM"
30 END
40 PRINT"UNTERPROGRAMM";
50 PRINT" IN";
60 PRINT" FORTFUEHRUNG"
70 RETURN
RUN
UNTERPROGRAMM IN FORTFUEHRUNG
ZURUCK VOM UNTERPROGRAMM
OK
```

1.16 GOTO

Format GOTO <Zeilennummer>

Anwendung Um uneingeschränkt aus dem laufenden Programm zu einer bestimmten Zeilennummer zu verzweigen.

Anmerkung Wenn die <Zeilennummer> einen ausführbaren Befehl erhält, wird dieser und die folgenden, ausgeführt.
Ist es ein nicht ausführbarer Befehl, wird das Programm mit dem nächsten ausführbaren Befehl nach <Zeilennummer>fortgesetzt.

Beispiel

```
10 READ R
20 PRINT "R =";R,
30 A=3.14*R^2
40 PRINT "FLACHE =";A
50 GOTO 10
60 DATA 5,7,12
RUN
R = 5                FLACHE = 78.5
R = 7                FLACHE = 153.86
R = 12               FLACHE = 452.16
OD ERROR IN 10
OK
```

1.17 GR

Format GR <I,J,K>

Anwendung Bildschirmfarbe im Graphik-Modus festlegen.

Anmerkung I ändert die Farbe des Vordergrundes und muss eine ganze Zahl sein.

0=schwarz
1=rot
2=grün
3=gelb
4=blau
5=purpur
6=hellblau
7=weiss

J ändert die Farbe des Hintergrundes von der Position des Cursors an bis zum Ende der Zeile und muss eine ganze Zahl enthalten.

0=schwarz
1=rot
2=grün
3=gelb
4=blau
5=purpur
6=hellblau
7=weiss

K muss ein ganzer Ausdruck sein.

0 = stehende Anzeige
1 = blinkende Anzeige

Beispiel

```
10 CURSORX 5:CURSORY 10
20 GR 1,4,0
```

In Zeile 10 wird die Hintergrundfarbe von Position 5 bis 39 in blau geändert. Jedes Zeichen in dieser Zeile, von Position 5 an, wird als rotes stehendes Graphikzeichen dargestellt.

1.18 IF...THEN und IF...GOTO

Format IF <Ausdruck> THEN <Ausdruck>

Format IF <Ausdruck> GOTO <Zeilennummer>

Anwendung Zum Treffen einer programmabhängigen Entscheidun die auf dem Ergebnis gestützt, durch den Ausdruck erwartet wird.

Anmerkung Wenn das Ergebnis von < Ausdruck > wahr (true) ist, wird der THEN- oder der GOTO-Befehl ausgeführt. THEN kann eine Verzweigung zu einer Zeilenummer oder eine oder mehrere ausführende Befehle enthalten. Auf GOTO folgt immer eine Zeilenummer. Wenn das Ergebnis von < Ausdruck > nicht wahr (false) ist, werden THEN oder GOTO nicht beachtet. Die ausführung des Programms wird mit der nächsten durchführbaren Zeile fortgesetzt.

Beispiel

```
100 IF (I<20) AND (I>10) GOTO 300
110 IF (I<5) OR (I>25) GOTO 500
120 PRINT "AUSSERHALB DES BEREICHS"
```

In diesem Beispiel wird geprüft, ob I grösser als 10 und kleiner als 20 ist. Wenn der Wert I in diesem Bereich liegt, springt das Programm auf Zeile 300. Liegt I nicht in diesem Bereich, geht die Programmausführung bei Zeile 110 weiter. Der Test in Zeile 110 prüft, ob I kleiner als 5 oder grösser als 25 ist. In diesem Fall wird das Programm mit Zeile 500 weitergeführt.

1.19 INIT

Format INIT <I>

Anwendung Um die Hintergrundfarbe auf dem Bildschirm zu bestimmen.

Anwendung Der Bildschirm wird gelöscht und nimmt die gewählte Hintergrundfarbe an.
I ist der Hintergrundfarbnummer und muss ganzzahlig eingegeben werden.

0 = schwarz
1 = rot
2 = grün
3 = gelb
4 = blau
5 = purpur
6 = hellblau
7 = weiss

Der Cursor wandert auf Zeile 2, Position 1. Mit dem INIT-Befehl werden folgende Befehle gleichzeitig ausgeführt.
TX 7,0,0;SCROLL,DISPLAY,BRIGHTO
Durch Drücken der RESET- oder CNTL+ CLEAR-Tasten gleichzeitig wird der Befehl INIT 6 ausgeführt.

Beispiel INIT 0

Der Bildschirm wird schwarz.

1.20 INPUT

Format INPUT [<"Fragetext">]; <Liste der Variablen>

Anwendung Der Befehl erlaubt die Dateneingabe über die Tastatur während der Programmausführung.

Anmerkung Wird der INPUT-Befehl gegeben, wird die Programmausführung unterbrochen und ein Fragezeichen ausgedruckt, das anzeigt, dass das Programm auf eine Dateneingabe wartet. Wenn < "Fragetext" > eingesetzt ist, wird dieser vor dem Fragezeichen ausgedruckt. Die notwendigen Daten werden dann über die Tastatur eingegeben.

Die eingegebenen Daten werden den Variablen < Liste der Variablen > zugeordnet. Die Anzahl der eingegebenen Daten muss mit der Anzahl der Variablen übereinstimmen. Die einzelnen Daten werden durch Kommata voneinander getrennt.

Die Variablenname der Liste kann numerisch oder ein Zeichenvariablename sein. (einschl. indizierte Variablen). Jeder eingegebene Daten-Typ muss mit dem durch den Variablennamen gestgelegten Typ übereinstimmen. (Zeichenfolgen hinter INPUT müssen nicht in Anführungszeichen gesetzt werden).

Werden in Verbindung mit INPUT zu viele oder zu wenige Daten oder als falschen Typ (numerische an Stelle von Zeichen etc.) eingegeben, wird die Fehlermeldung "?REDO FROM START" ausgedruckt. Es findet keine Zuordnung der Daten statt, wenn keine akzeptablen Werte eingegeben werden.

INPUT kann nicht im Direkt-Modus eingesetzt werden.

Um eine numerische Variable auf Null zu setzen, muss eine Null eingegeben werden. Zeichenvariable werden durch Eingabe eines Fragezeichens gelöscht (auf blank gesetzt).
Ein "?" ist in "Fragetext" nicht erlaubt.

Beispiel 1

```
10 INPUT X
20 PRINT X;"zum QUADRAT IST";X^2
30 END
RUN
? 5
(Die 5 wurde vom Anwender als Antwort auf das
Fragezeichen gegeben.)
 5 zum QUADRAT IST 25
OK
```

Beispiel 2

```
10 PI=3.14
20 INPUT "WELCHER RADIUS";R
30 A=PI*R^2
40 PRINT "DIE KREISFLACHE IST";A
50 PRINT
60 GOTO 20
RUN
WELCHER RADIUS? 7.4 (Eingabe durch Anwender 7.4)
DIE KREISFLACHE IST 171.946

WELCHER RADIUS?
```

1.21 LET

Format [LET] <Variable> = <Ausdruck>

Anwendung Den Wert einer Variablen dem Ausdruck zuzuordnen.

Anmerkung Das Wort LET muss nicht gesetzt werden (wahlfrei Die Gleichheitszeichen sind für Zuordnung ausreichend.

Beispiel

```
110 LET D=12
120 LET E=12^2
130 LET F=12^4
140 LET SUM=D+E+F
```

oder

```
110 D=12
120 E=12^2
130 F=12^4
140 SUM=D+E+F
```

1.22 LINE

Format LINE <I,J>

Anwendung Zur Anzeige von Zeilen eines gespeicherten Bildes auf dem Bildschirm

Anmerkung I ist die erste angezeigte Zeile und muss mit einer Zahl zwischen 0 und 23 limitiert werden.
J ist die letzte angezeigte Zeile und muss mit einer Zahl zwischen 0 und 23 limitiert werden.
J muss grösser sein als I.
(Siehe auch STORE)

Beispiel

```
10 STORE:PAGE:GR 1,0,1
20 CURSORX 5:CURSORY 10
30 PRINT"Wk"
40 CUSORX 5:CURSORY 11
50 PRINT"uz"
60 LINE 10,11
```

1.23 LIST

Format LIST [<Zeilennummer>]

Anwendung Zum Auflisten des augenblicklich gespeicherten Programms oder Teilen daraus auf dem Bildschirm.

Anmerkung BASIC-80 geht nach der Ausführung des LIST-Befehls automatisch wieder auf Befehlsebene.

Bei Weglassen von <Zeilennummer> wird das Programm von der niedrigsten Zeilennummer an aufgelistet. Die Auflistung wird entweder durch das Programmende oder durch Betätigen der "BREAK-Taste" beendet. Durch Angabe von <Zeilennummer> wird das Programm ab der eingegebenen Zeile aufgelistet.

Beispiel **LIST**

Listet das momentan gespeicherte Programm.

LIST 500

Listet den Teil des Programms ab der Zeile 500 bis zum Ende.

1.24 NEW

Format NEW

Anwendung Zum Löschen des gesamten Speicherinhalts (Programm und Variable).

Anmerkung NEW wird auf der Befehlsebene eingegeben, um den Speicher vor Eingabe eines neuen Programms zu löschen. BASIC-80 geht nach Ausführung des NEW-Befehls immer zurück auf die Befehlsebene.

Beispiel **NEW**

1.25 ON...GOSUB und ON...GOTO

Format

ON<Ausdruck>GOTO<Liste d. Zeilennummer>

ON<Ausdruck>GOSUB<Liste d. Zeilennummer>

Anwendung

Um abhängig von einem Wert zu mehreren festgelegten Zeilennummern zu verzweigen, um dann nach der Verarbeitung wieder an den ursprung zurückzukehren.

Anmerkung

Der Wert von <Ausdruck> bestimmt, auf welche Zeilennummer im Programm verzweigt wird. Wenn zum Beispiel der Wert 3 ist, wird das Programm auf die dritte ausgegebene Zeile wechseln. (Ist der Wert eine Dezimale, wird er auf- oder abgerundet.)

Im "ON...GOSUB"-Befehl muss jede Zeilennummer der Liste die erste Zeile eines Unterprogramms sein.

Ist der Wert des Ausdrucks gegenüber der Zahl in der Liste negativ oder grösser als 255, erscheint die Fehlermeldung "ILLEGAL FUNCTION CALL".

Beispiel

```
10 ON L GOTO 150,300,320,390
```

1.26 PAGE

Format PAGE

Anwendung Zur Stabilisierung des Bildes auf dem Bildschirm.

Anmerkung Wenn dieser Befehl eingegeben wird, kann das Bild nicht mehr auf- oder abwärts laufen. Um das Bild wieder laufen zu lassen, müssen Sie den "SCROLL-Befehl" benutzen.

Beispiel

```
10 SCROLL
20 PAGE
```

1.27 POKE

Format POKE <I,J> (I und J müssen ganze Zahlen sein)

Anwendung Zur Adressierung von Bytes im Speicher.

Anmerkung "I" ist die Adresse im Speicher, die durch "POKE belegt werden soll. "J" sind die zu speichernden Daten. "J" muss zwischen 0 und 255 liegen. "I" muss kleiner als 32768 sein.

Daten können durch "POKE" auch Speicherplätze belegen, die über die 32768 hinausgehen, wenn man eine negative Zahl für "I" setzt. Der Wert "I" errechnet sich aus 65536 minus der gewünschten Adresse. Zum Beispiel sollen Daten in Adresse 45000 gespeichert werden: I = 45000 - 65536 oder - 20536.

Der gegenteilige Befehl zu POKE ist PEEK. PEEK spricht eine Adresse an, von der Daten gelesen werden sollen.

POKE und PEEK eignen sich zur effizienten Speicherung von Daten, Speicherung von Unterprogrammen in Maschinensprache und zum Datenaustausch von und zu Maschinensprache-Unterprogrammen.

Beispiel 10 POKE 23040,255

1.28 PRINT

Format PRINT [<auszudruckender Text>]

Anwendung Datenausgabe auf dem Bildschirm

Anmerkung Wird < auszudruckender > Text nicht hinter PRINT eingesetzt, so wird eine Leerzeile gedruckt. Setzt man < auszudruckender > Text hinter PRINT, so wird dieser auf dem Bildschirm (Terminal) ausgedruckt. auszudruckender Text können numerisch und/oder Zeichenausdräucke sein. (Zeichenfolgen müssen in Anführungszeichen gesetzt werden.)

Druck - Positionen
Die Position jeden Ausdrucks richtet sich nach der Art der Interpunktion bei der Trennung in der Auflistung. BASIC-80 teilt die Zeile in 14-stellige Zonen auf. Ein Komma in der Auflistung bedeutet, dass der Text des folgenden Teils in der Position 1 der nächsten Zone beginnt. Ein Semikolon bedeutet, dass die Ausdrücke direkt hintereinander gedruckt werden. Eingaben von einer oder mehreren Leertasten hinter den Ausdrücken haben den gleichen Effekt wie ein Semikolon.

Wird eine Auflistung mit einem Komma oder einem Semikolon abgeschlossen, beginnt der nächste Ausdruck auf derselben Zeile mit entsprechendem Abstand. Wird die Liste ohne Komma und Semikolon abgeschlossen, so wird am Zeilenende das Zeichen für Spring auf neue Zeile (carriage return) gegeben.

Positiven Zahlen steht eine Leerstelle voran, negativen Zahlen steht das Minuszeichen voran. Dezimalzahlen und Potenzen, die durch 6 oder weniger Stellen hinter Komma haben, können ohne an Genauigkeit zu verlieren, in verkürzter Form dargestellt werden. Zum Beispiel, 10(-6) = .000001 und 10(-7) = 1E-7.

Anstelle des Wortes PRINT kann auch ein Fragezeichen als Print-Befehl verwendet werden.

Beispiel 1

```
10 X = 5
20 PRINT X+5, X-5, X*(-5)
30 END
RUN
 10        0        -25
OK
```

In diesem Befehl bedingen die Kommata, dass jeder Ausdrück in der nächsten Zone beginnt.

Beispiel 2

```
10 INPUT X
20 PRINT X "ZUM QUADRAT IST "X ^2 "UND"
30 PRINT X "HOCH 3 IST "X^3
40 PRINT
50 GOTO
RUN
? 9
 9 ZUM QUADRAT IST 81 UND
 9 HOCH 3 IST 729

? 21
 21 ZUM QUADRAT IST 441 UND
 21 HOCH 3 IST 9261

?
```

In diesem Beispiel sorgt das Semikolom am Ende der Zeile 20 dafür, dass beide Aussagen in der gleichen Zeile gedruckt werden, und Zeile 40 bedingt eine Leerzeile zwischen den Durchläufen.

Beispiel 3

```
10 FOR X=1 TO 4
20 J=J+5
30 K=K+10
40 ? J;K;
50 NEXT X
RUN
 5  10  10  20  15  30  20  40
OK
```

In diesem Beispiel sorgen die Semikoloms dafür, dass die PRINT-Aussagen gleich hintereinander gedruckt werden. (Vergessen Sie nicht, dass einer Zahl immer ein Leerzeichen folgt und einer positiven Zahl immer ein Leerzeichen vorangeht). In Zeile 40 wurde das Fragezeichen anstelle des Wortes PRINT eingesetzt.

1.29 READ

Format READ<Liste der Variablen>

Anwendung Lesen von Werten einer DATA-Anweisung und die Zuordnung zu den Variablen (siehe DATA)

Anmerkung READ- und DATA-Befehle gehören stets zusammen. READ-Befehlen ordnen Variablen die Werte 1:1 aus der DATA -Anweisung zu. Die Variable in der READ-Anweisung können numerisch oder ein Zeichen sein, und die zu lesenden Werte müssen mit den festgelegten Variablen übereinstimmen. Stimmen sie nicht überein, so erscheint eine Fehlermeldung "SYNTAX ERROR".

Eine einzelne READ-Befehl kann eine oder mehrere DATA-Anweisungen verarbeiten (sie werden der Reihe nach zugeordnet), oder mehrere READ-Befehle k*onnen immer der gelichen DATA-Anweisung zugewiesen werden. Wenn die Anzahl der Variablen in <"Liste der Variablen" > die Anzahl der Elemente in der DATA-Anweisung übersteigt, wird die Fehlermeldung "OUT OF DATA" angezeigt. Ist die Anzahl der spezifizierten Variablen geringer als die Anzahl er Elemente in der DATA-Anweisung, dann werden folgende READ-Anweisungen anfangen Daten zu lesen vom ersten nicht-gelesene Element. Wenn es keine folgende READ-Anweisungen gibt werden die überzähligen Daten ignoriert.

Um DATA-Anweisung von Anfang an lesen zu können, ist der RESTORE-Befehl zu benutzen (siehe RESTORE).

Beispiel

```
80 FOR I=1 TO 10
20 READ A(I)
100 NEXT I
110 DATA 3.08,5.19,3.12,3.98,4.24
120 DATA 5.08,5.55,4.00,3.16,3.37
```

Dieses Programm liest die Werte der DATA-Anweisung in das Feld (array) A. Nach Ausführung wird der Wert von A(I) 3.08 sein, usw.

Beispiel 2

```
10 PRINT "STADT","LAND","PLZ"
20 READ S$,L$,P
30 DATA "DUESSELDORF","NRW",4000
40 PRINT S$,L$,P
RUN
STADT          LAND           PLZ
DUESSELDORF    NRW            4000
OK
```

Dieses Programm liest numerische Daten und Zeichen folgen der DATA-Anweisung in Zeile 30.

1.30 REM

Format REM <Ausdruck>

Anwendung Um erklärende Anmerkungen ins Programm aufzunehmen.

Anmerkung REM-Anweisungenwerden nicht ausgeführt, aber exakt ausgegeben, wenn das Programm aufgelistet wird.

REM-Anweisungen können eingeschoben werden (von einer GOTO- oder einer GOSUB-Anweisung), die Programmausführung wird mit dem nächsten ausführbaren Befehl nach der REM-Anweisung fortgesetzt.

Beispiel

```
120 REM BERECHNUNG
130 FOR I=1 TO 20
140 SUM=SUM+V(I)
```

1.31 RESTORE

Format RESTORE [<Zeilennummer>]

Anwendung Um DATA-Anweisungen von einem bestimmten Punkt aus wieder vom Anfang lesen zu können.

Anmerkung Ist ein RESTORE-Befehl gegeben, so wird dem nächsten READ-Befehl im Programm das erste Element in der DATA-Anweisung wieder zugewiesen. Ist < Zeilennummer > angegeben, so holt sich der nächste READ-Befehl das erste Element aus der entsprechenden DATA-Anweisung.

Beispiel

```
10 READ A,B,C
20 RESTORE
30 READ D,E,F
40 DATA 57,68,79
```

1.32 RUN

Format RUN [<Zeilennummer>]

Anwendung Um das momentan gespeicherte Programm auszuführen.

Anmerkung Ist < Zeilennummer > ausgegeben, beginnt das Programmmit der Ausführung ab dieser Zeile. Andernfalls beginnt das Programm mit der niedrigsten Zeilennummer. BASIC 80 kehrt immer wieder auf die Befehlsebene zurück, wenn der RUN-Befehl ausgeführt wurde.

Beispiel RUN

1.33 SCREEN

Format SCREEN

Anwendung Um ein komplett gespeichertes Bild auf dem Bildschirm darzustellen.

Anmerkung Um ein Bild zu speichern ist ein STORE-Befehl zu benutzen.

Beispiel Siehe STORE

1.34 SCROLL

Format SCROLL

Anwendung Um das Bild auf dem Bildschirm nach einem PAGE-Befehl wieder zum Laufen zum bringen.

Anmerkung Auf diesen Befehl hin rollt das Bild von oben na unten ab, solange bis der PAGE-Befehl gegeben wird.

Beispiel Siehe PAGE

1.35 SOUND

Format SOUND <I>

Anwendung Um ein Geräusch durch eine Bestimmte Geräuschnummer zu erzeugen.

Anmerkung I ist die Geräuschnummer und muss sein:

0 = Ton 2
1 = Unfallgeräusch
2 = Ton 1
3 = Tonfolge, aufsteigend
4 = Ton 4
5 = Ton 5
6 = Tonfolge, absteigend
7 = Gewehrschussgeräusch

Beispiel 10 Sound 4

1.36 STOP

Format STOP

Anwendung Zum Beenden der Programmausführung und Rückkehr zur Befehlsebene.

Anmerkung STOP-Befehle können an beliebigen Stellen des Programms eingesetzt werden, um die Ausführung zu beenden. Ist ein STOP-Befehl gegeben, wird folgende Meldung ausgedruckt:

Break in line nnnnn

BASIC-80 kehrt immer wieder auf die Befehlsebene zurück, wenn der STOP-Befehl ausgeführt ist. Die Ausführung wird erst mit dem CONT-Befehl fortgesetzt.

Beispiel

```
10 INPUT A,B,C
20 K=A^2*5.3:L=B^3/.26
30 STOP
40 M=C*K+100:PRINT M
RUN
? 1,2,3
BREAK IN 30
OK
PRINT L
 30.7692
OK
CONT
 115.9
OK
```

1.37 STORE

Format STORE

Anwendung Alle druckbaren Zeichen werden im Bildschirmspeichergespeichert, aber können nicht sofort angezeigt werden.

Anmerkung Dieser Befehl erlaubt dem Anwender ein ganzes Bild im Speicher zusammenzusetzen. Um dieses Bild anzuzeigen, ist der LINE- oder SCREEN-Befehl zu benutzen. Um die druckbaren Zeichen sofort anzuzeigen, ist der DISPLAY-Befehl einzusetzen.

Beispiel

```
10 STORE
20 GR 1,0,0
30 CURSORX 5:CURSORY 5
40 PRINT "Wk"
50 CURSORX 5:CURSORY 6
60 PRINT "uz"
70 SCREEN
80 DISPLAY
90 PRINT "DAS IST SCHON"
```

1.38 TX

Format TX <I,J,K>

Anwendung Um den Text-Modus zu starten.

Anmerkung I ändert die Vordergrundfarbe und muss eine ganze Zahl sein.

0 = schwarz
1 = rot
2 = grün
3 = gelb
4 = blau
5 = purpur
6 = hellblau
7 = weiss

J ändert die Grösse der Textzeichen und muss eine ganze Zahl sein.

0 = normal
1 = doppelte Höhe
2 = doppelte Breite
3 = doppelte Höhe und doppelte Breite
4 = Negativ-Bild
5 = Negativ-Bild und doppelte Höhe
6 = Negativ-Bild und doppelte Breite
7 = Negativ-Bild, doppelte Höhe und Breite

K muss eine ganze Zahl sein.

0 = stillstehend
1 = blinkend

Die Hintergrundfarbe ist die des letzten GR - oder DELIM-Befehls.

Beispiel **10 TX 4,0,0**

Jedes Zeichen wird in normaler Grösse blau und stillstehend abgebildet.

Kapitel 2

Die wesentlichen Funktionen, die BASIC-80 vorsieht, werden in diesem Kapitel behandelt. Die Funktionen können ohne weitere Definition in jedem Programm abgerufen werden.

Argumente zu Funktionen sind immer in Klammern eingeschlossen. In diesem Kapitel sind in den Formaten der Funktionen die Argumente wie folgt abgekürzt:

X und Y	stellen alle numerischen Ausdrücke dar
I und J	bedeuten ganze Zahlen
X$ u. Y$	drücken Zeichen aus

2.1 ABS

Format ABS(X)

Wirkung Gibt dem Ausdruck (X) den absoluten Wert zurück.

Beispiel

```
PRINT ABS(7*(-5))
 35
OK
```

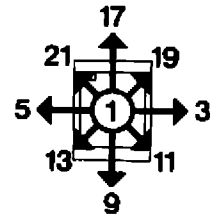
2.2 ACTION

Format ACTION(I)

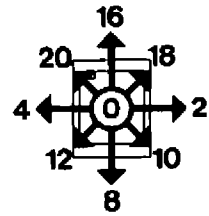
Wirkung Gibt den momentanen Richtungswert der ACTION-Taste und des Steuerhebels an.

I = 0 für linke Kontrolle
I = 1 für rechte Kontrolle

Wenn die ACTION-Taste momentan gedrückt wird, ist der zurückgegebene Wert wie folgt:



Wenn die ACTION-Taste momentan nicht gedrückt wird, ist der zurückgegebene Wert wie folgt:



Beispiel

```
10 IF ACTION(1)=1 THEN PRINT "SCHUSS"
```

2.3 ASC

Format ASC(X$)

Wirkung Gibt aus dem Zeichencode am ersten Zeichen in der Zeichenfolge X$ einen numerischen Wert. Ist X$ = 0, wird ein "FC"-Fehler angezeigt. (siehe Anhang D, Zeichencode).

Beispiel

```
10 X$="TEST"
20 PRINT ASC(X$)
RUN
 84
OK
```

2.4 ATN

Format: ATN(X)

Wirkung: Gibt den Arctan von X in Bogenmass an. Das Ergeb liegt zwischen -PI/2 und PI/2.

Beispiel

```
10 X=3
20 PRINT ATN(X)
RUN
 1.24905
OK
```

2.5 CHR$

Format CHR$(I)

Wirkung Gibt eine Zeichenfolge wieder, dessen einzelnes Element den Zeichencode (I) hat (siehe Anhang D, Zeichencode). CHR$ wird meistens dazu benutzt, um ein spezielles Zeichen zum Bildschirm zu senden. Zum Beispiel löscht CHR$ (15?)) den Bildschirm und bringt den Cursor an die Ausgangsposition zurück.

Beispiel

```
PRINT CHR$(66)
B
OK
```

2.6 COS

Format COS(X)

Wirkung Gibt den Cosinus des Wertes X in Bogenmass an.

Beispiel

```
10 X=2*COS(.4)
20 PRINT X
RUN
 1.84212
OK
```

2.7 EXP

Format EXP(X)

Wirkung Gibt den Wert für die Basis mit dem Exponenten X aus. X muss < = 87.3365 sein. Ist EXP grösser, so wird die Fehlermeldung "OV" angezeigt. (das Maschinenzeichen für Unendlich erscheint als Ergebnis, und die Programmausführung geht weiter.)

Beispiel

```
10 X=5
20 PRINT EXP(X-1)
RUN
 54.5982
OK
```

2.8 FRE

Format FRE(0)

Wirkung Gibt die Anzahl der nicht vom BASIC-80 belegten Bytes im Speichers aus.

Beispiel

```
PRINT FRE(0)
 14050
OK
```

2.9 INT

Format INT(X)

Wirkung Gibt die grösste ganze Zahl von X aus

Beispiel

```
PRINT INT(99.89)
 99
OK
PRINT INT(-12.11)
-13
OK
```

2.10 LEFT$

Format LEFT$(XS,I)

Wirkung Gibt in der Anzahl I alle Zeichen aus die links stehen. I muss eine Zahl zwischen 0 und 255 sein. Wenn I grösser als LEN (X$) ist, wird die gesamte Zeichenfolge(X$) ausgegeben. Ist I = 0 wird eine Null-Zeichenfolge (Länge Null)aus-gegeben.

Beispiel

```
10 A$="BASIC-80"
20 B$=LEFT$(A$,5)
30 PRINT B$
RUN
BASIC
OK
```

2.11 LEN

Format LEN(X$)

Wirkung Gibt die Anzahl der Zeichen in (X$) aus. Nicht zu druckbare Zeichen und Leerzeichen werden mitgezählt.

Beispiel

```
10 X$="PORTLAND, OREGON"
20 PRINT LEN(X$)
RUN
 16
OK
```

2.12 LOG

Format LOG(X)

Wirkung Gibt den natürlichen Logarithmus von X aus. X muss grösser als "0" sein.

Beispiel

```
PRINT LOG(45/7)
 1.86075
OK
```

2.13 MID$

Format MID$(X$,I[,J])

Wirkung Gibt eine Zeichenfolge der Länge J, beginnend mit dem I. Zeichen von X$ aus. I und J müssen sich zwischen 0 und 255 bewegen. Wird J weggelassen, oder es sind rechts weniger als in J angegebene Zeichen vorhanden, werden alle Zeichen rechts, von I an, aus- gedruckt. Wenn I > LEN(X$), gibt MID$ einen Null-Zeichenfolge aus.

Beispiel

```
10 A$="GUTEN"
20 B$="MORGEN ABEND MITTAG"
30 PRINT A$;MID$(B$,7,6)
RUN
GUTEN ABEND
OK
```

2.14 PEEK

Format PEEK(I)

Wirkung Liest ein Byte aus dem Speicherplatz I. I muss kleiner als 32768 sein. Um einen höheren Speicherplatz als 32768 zu lesen, muss von der gewünschten Adresse die Zahl 65536 abgezogen werden.

Beispiel A=PEEK(23040)

2.15 POS

Format POS(I)

Wirkung Gibt die momentane Cursor-Position an. Die äusserste linke Position ist 0. I ist ein "Blindargument".

Beispiel

```
IF POS(X)>38 THEN PRINT CHR$(141)
```

2.16 RIGHT$

Format RIGHT$(X$,I)

Wirkung Gibt die unter I angegebene Anzahl von Zeichenn des Zeichenfolge X$, von rechts gesehen, an. I muss zwischen 0 und 255 liegen. Ist I gröseer als LEN (X$), wird die gesamte Zeichenfolge (X$) ausgegeben. Wenn I = 0, wird der Null-Zeichenfolge (Länge Null) ausgegeben.

Beispiel

```
10 A$="SCHNEIDER BASIC-80"
20 PRINT RIGHT$(A$,8)
RUN
BASIC-80
OK
```

2.17 RND

Format RND(X)

Wirkung Gibt eine Zufallszahl zwischen 0 und 1 aus. Wenn X = 0 RND, wird die letzte generierte Zahl von RND wiederholt. Wenn X ≤ 0 RND, erzeugt RND die nächste Zahl in der Folge.

Beispiel

```
10 FOR I=1 TO 5
20 PRINT INT(RND(+1)*100);
30 NEXT I
RUN
 49  67  98  73  78
OK
```

2.18 SGN

Format SGN(X)

Wirkung Wenn X > 0 gibt SGN (X) 1 aus
Wenn X = 0 gibt SGN (X) 0 aus
Wenn X < 0 gibt SGN (X)-1 aus

Beispiel

```
On SGN(X)+2 GOTO 100,200,300
```

Es wird auf Zeile 100 gesprungen, wenn X negativ ist; auf Zeile 200, wenn X = 0 ist; auf Zeile 300, wenn X positiv ist.

2.19 SIN

Format SIN(X)

Wirkung Gibt den Sinuswert von (X) in Bogenmass an.
COS(X) = SIN(X + 3.14159/2)

Beispiel

```
PRINT SIN(1.5)
 .997495
OK
```

2.20 SPC

Format SPC (I)

Wirkung Druckt I Leerzeichen auf den Bildschirm. SPC kann nur mit dem PRINT-Befehl benutzt werden. I muss eine Zahl zwischen 0 und 39 sein.

Beispiel

```
PRINT "UEBER";SPC(18);"DORT"
UEBER                  DORT
OK
```

2.21 SQR

Format SQR(X)

Wirkung Gibt die Quadratwurzel von X aus. X muss > = 0 sein.

Beispiel

```
10 X=10
20 PRINT X;SQR(X)
RUN
 10  3.16228
OK
```

2.22 STR$

Format STR$(X)

Wirkung Gibt eine Zeichenfolge mit dem Wert von X an.

Beispiel

```
10 N=1325
20 PRINT LEN(STR$(N))
RUN
 5
OK
```

2.23 STICKX

Format STICKX(J)

Wirkung Zeigt den horizontalen Wert des Steuerhebels an.
J = 0 für den linken Steuerhebel
J = 1 für den rechten Steuerhebel
Der Wert beträgt 255, wenn der Steuerhebel nach links, und 1, wenn der Steuerhebel nach rechts gedrückt wurde. Der ausgegebene Wert ist 0, wenn der Hebel nicht betätigt wurde.

Beispiel

```
10 LR=STICKX(0)
```

2.24 STICKY

Format STICKY(J)

Wirkung Zeigt den vertikalen Weg der Steuerhebel an.
J = 0 für den linken Steuerhebel
J = 1 für den rechten Steuerhebel
Der Wert beträgt 255, wenn der Steuerhebel nach oben gedrückt wurde und 1, wenn der Steuerhebel nach unten gedrückt wurde. Der angegebene Wert ist 0, wenn der Handhebel nicht betätigt wurde.

Beispiel

```
10 LV=STICKY(0)
```

2.25 TAB

Format TAB(I)

Wirkung Zur Erzeugung von I Leerstellen zur auf dem Bildschirm. Wenn die augenblickliche Druckposition schon über die Position I hinaus ist, hat TAB keine Wirkung. TAB(1) ist die äusserste linke Position. I muss eine Zahl zwischen 1 und 39 sein. TAB kann nur mit dem PRINT-Befehl benutzt werden.

Beispiel

```
PRINT "NAME";TAB(20);"ANZAHL"
NAME                 ANZAHL
OK
```

2.26 TAN

Format TAN(X)

Wirkung Gibt den Tangens X in Bogenmass an. Ist TAN zu gross, wi eine Fehlermeldung "OV" angezeigt.

Beispiel

```
10 X=15
20 Y=2*TAN(X)/2
RUN
-.855994
OK
```

2.27 USR

Format USR(X)

Wirkung Ruft das in Maschinensprache geschriebene Unterprogramm des Anwenders mit dem Argument (X) auf. (Siehe Anhang A)

Beispiel

```
10 B=T*SIN(Y)
20 C=USR(B)
```

2.28 VAL

Format VAL(X$)

Wirkung Gibt den numerischen Wert des Zeichenfolge X$ au
Ist das erste Zeichen von X$ nicht "+"," -", oder
eine Ziffer, dan ist VAL(X$) = 0.

Beispiel

```
10 INPUT A$
20 IF VAL(A$) <10 THEN STOP
30 END
RUN
? 5
Break in 20
OK
```

2.29 KEY

Format KEY(0)

Wirkung Gibt den Zeichencode der auf der Tastetur gedrückten Taste aus, oder gibt 0 an, wenn keine Taste gedrückt wurde. Es werden keine Zeichen auf dem Bildschirm angezeigt, und alle Symbole an das Programm direkt weitergegeben; ausser der BREAK-Taste, die das Programm beendet.

Beispiel

```
10 FOR I=1 TO 1000
20 IF KEY(0)=97 GOTO 60
30 IF KEY(0)=65 GOTO 80
40 NEXT I
50 END
60 PRINT "TASTE a IST GEDRUCKT"
70 END
80 PRINT "TASTE A IST GEDRUCKT"
90 END
```

Anhang A

In BASIC-80 Programmen können Unterprogramme in Assemblersprache verwandt werden. Die USR-Funktion erlaubt den Aufruf von Assembler-Unterprogrammen auf die gleiche Art wie den Aufruf von BASIC-Funktionen. Um diese BASIC-80 Eigenschaft zu benutzen, muss der Anwender Kenntnisse über die z80-Assemblersprache besitzen. Es gibt schon mehrere Bücher, die diese Kenntnisse vermitteln.

Speicherplatzanweisung

Vor dem Einlesen eines Assemblerprogramms muss entsprechender Speicherplatz reserviert werden. BASIC-80 braucht den gesamten Speicherraum von Beginn an aufwärts. So können nur die obersten Speicherplätze für BASIC-80 verfügbar zu machen, ist die CLEAR-Taste zu benutzen. Das Assembler-Unterprogramm kann mit Hilfe der POKE-Anweisung eingelesen werden.

USR-Funktionsablauf

Das Assembler-Unterprogramm muss in die Speicherplätze 347709 und 34780 eingelesen (POKE) werden. Das Byte mit der niedrigsten Stelle musszuerst gespeichert werden; gefolgt von dem BYTE mit der höheren Stellenzahl. Die USR-Funktion ruft das Programm aus den Adressen 34779 und 34780 ab.

Beispiel

Assembler-Sprache

```
LD A,"X"
LD (H800A),A
RET
```

Maschinencode

```
62 88
50 10 128
201
```

Um diese Assembler-Routine in der Speicheradresse 45058 zu speichern, können die folgenden BASIC-Anweisungen benutzt werden:

```
10 CLEAR 0,-20480
20 DATA 62,88,50,10,128,201
30 FOR I=0 TO 5:READ T
40 POKE (-20478+I),T
50 NEXT I
```

Um diese Programme aufzurufen, können folgende BASIC-Befehle benutzt werden:

```
60 POKE -30757,2
70 POKE -30756,176
80 T=USR(0):SCREEN
```

Das Format eines USR-Funktionsaufrufs ist: USR (Argument). Das Argument ist hierbei ein numerischer Ausdruck. Um das Argument zu erhalten, muss das Assembler-Unterprogramm Speicherplatz-Adresse 5431 mit der Routine DEINT aufrufen. DEINT plaziert das Argument in das Registerpaar D.E.

Um das Ergebnis aus einem Assembler-Unterprogramm wieder in BASIC auszugeben, muss das Ergebnis in Register A geladen werden und zu der Routine GNABF, Speicheradresse 6649, springen.

Anhang B

Code	Nummer	Meldung

BS 9 Subscript out of range

Es wird mit dem Subscript (Index) ein Feld (array) element angespro-chen, das ausserhalb der definierten Feldgrösse liegt, oder der Index spricht ein Feldelement an, das innerhalb der definierten Feldgrösse nicht existent ist.

CN 17 Can't continue

Es wurde ein Versuch gemacht, ein Programm fortzuführen, das:
1 durch einen Fehler unterbrochen ist
2 während einer Unterbrechung modifiziert wurde.
3 nicht existiert

DD 10 Redimensioned array

Zwei DIM-Aussagen wurden für das gleiche Feld (array) gegeben, oder eine DIM-Anweisung wurde für ein Feld gegeben, nachdem die unterlassende Dimensionierung in dieses Feld eingesetzt wurde.

FC 5 Illegal function call

Ein Parameter, der ausserhalb der definierten Grösse liegt, ist einer mathematischen oder Textfunktion zugewiesen worden. Ein "FC" kann ebenso auftreten als Ergebnis von:
1 einem negativen oder übertrieben grossen Index
2 einem negativen oder 0-Argument mit LOG
3 einem negativen Argument von SQR
4 einer negativen Mantisse mit nicht ganzzahligem Exponenten
5 einem Aufruf einer USR-Funktion, deren Startadresse noch nicht eingegeben wurde.
6 einem falschen Argument von MID$, LEFT$, PEEK,POKE,TAB,SPC, oder ON ... GOTO

ID	12	**Illegal direct** Eine im Direkt-Modus nicht zulässige Anweisung ist im Direkt-Modus als Befehl eingegeben worden.
LS	15	**String too long** Eine Zeichenfolge ist zu lang.
MO	19	**Missing operand** Ein Ausdruck enthält einen Operator ohne Operand.
NF	1	**Next without for** Eine NEXT-Anweisung kann ohne eine vorher gegebene FOR-Anweisung nicht ausgeführt werden.
OD	4	**Out of data** Eine READ-Anweisung wird ausgeführt, obwohl keine ungelesenen Daten der DATA-Anweisung mehr im Programm vorhanden sind.
OM	7	**Out of memory** Ein Programm ist zu gross, hat zu viele FOR-Schleifen oder GOSUB-Befehle, zu viele Variablen oder zu komplizierte Ausdrücke.
OS	14	**Out of the string space** Die Zeichenvariablen übersteigen die vorgesehene Speicherkapazität. Mit CLEAR kann man den Zeichen mehr Leerstellen zuweisen oder die Grösse und die Anzahl der Zeichenfolgen verringern.
OV	6	**Overflow** Das Ergebnis einer Rechnung ist zu gross für das BASIC-80 Zahlen Format. Tritt ein zu kleines Ergebnis auf, wird das Ergebnis und das Programm ohne Fehleranzeige weiterverarbeitet.
SN	2	**Syntax-error** Eine Programmzeile enthält unkorrekte Zeichen (zum Beispiel ungleiche Klammern, falsch eingegebene Befehle oder Aussagen, falsche Interpunktion etc.)
ST	16	**String formula too komplex** Eine Zeichenfolge ist zu komplex. Der Ausdruck muss in kleinere Ausdrücke zerlegt werden.

TM	13	**Type Mismatch** Eine Zeichenvariablename ist einer numerischen Variablen zugeordnet oder umgekehrt; eine Funktion, die ein numerisches Argument erwartet, ist ein Zeichenargument gegeben worden oder umgekehrt.
RG	3	**Return without GOSUB** Eine RETURN-Anweisung wird gefunden, ohne dass vorher eine GOSUB-Anweisung gemacht wurde.
RF	18	**Undefined user function** Eine Gebraucher-Funktion wird aufgerufen, ohne dass die Funktion definiert worden ist (DEF-Anweisung).
UL	8	**Undefined line** In einer GOTO, GOSUB, IF ... THEN Anweisung wird in eine nicht existierende Zeile gesprungen.
/0	11	**Division by zero** In einem Ausdruck wird durch Null geteilt oder bei einer Potenzierung wird Null zu einer negativen Potenz erhoben. Als Resultat der Teilung wird die grösste in den Speicher passende Zahl mit dem Zeichen des Zählers gegeben, oder als Resultat der Potenzierung wird die grösste positive Zahl, die in den Speicher passt, gegeben und die Durchführung wird fortgesetzt.

Anhang C

Eine Anzahl abgeleiteter Funktionen, die nicht charakteristisch für BASIC-80 sind, können in BASIC-80 wie folgt definiert werden:

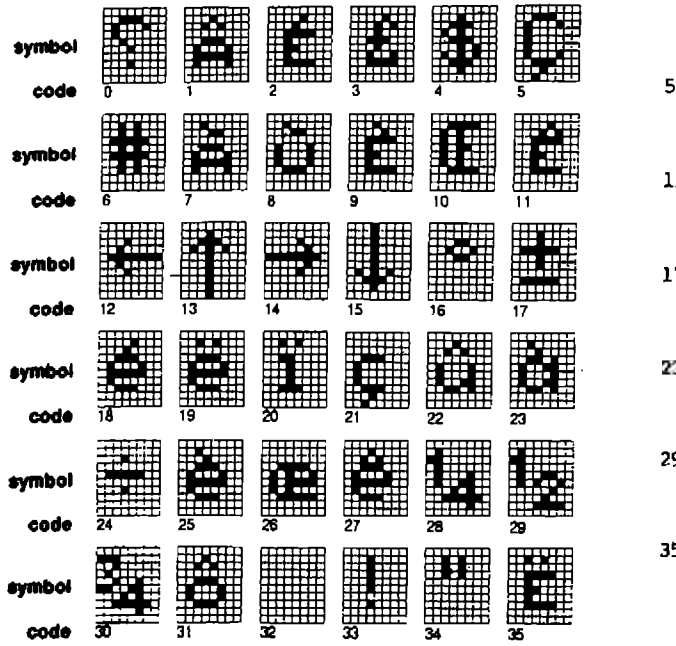
Abegegleitete Funktionen	BASIC-80 Definition
SEKANTE	= 1/COS(X)
KOSEKANTE	= 1/SIN(X)
KOTANGENS	= 1/TAN(X)
ARKUSSINUS	= ATN(X/SQR(-X*X+1))
ARKUSCOSINUS	= ATN(X/SQR(-X*X+1))+1.5708
ARKUSSEKANTE	= ATN(X/SQR(X*X-1))+SGN(X)-1)*1.5708
ARKUSKOSEKANTE	= ATN(X/SQR(X*X-1))+(SGN(X)-1)*1.5708
ARKUSKOTANGENS	= ATN(X)+1.5708
HYPERBELSINUS	= (EXP(X)-EXP(-X))/2
HYPERBELCOSINUS	= (EXP(X)+EXP(-X))/2
HYPERBELTANGENS	= EXP(-X)/EXP(X)+EXP(-X))*2+1
HYPERBELSEKANTE	= 2/(EXP(X)+EXP(-X))

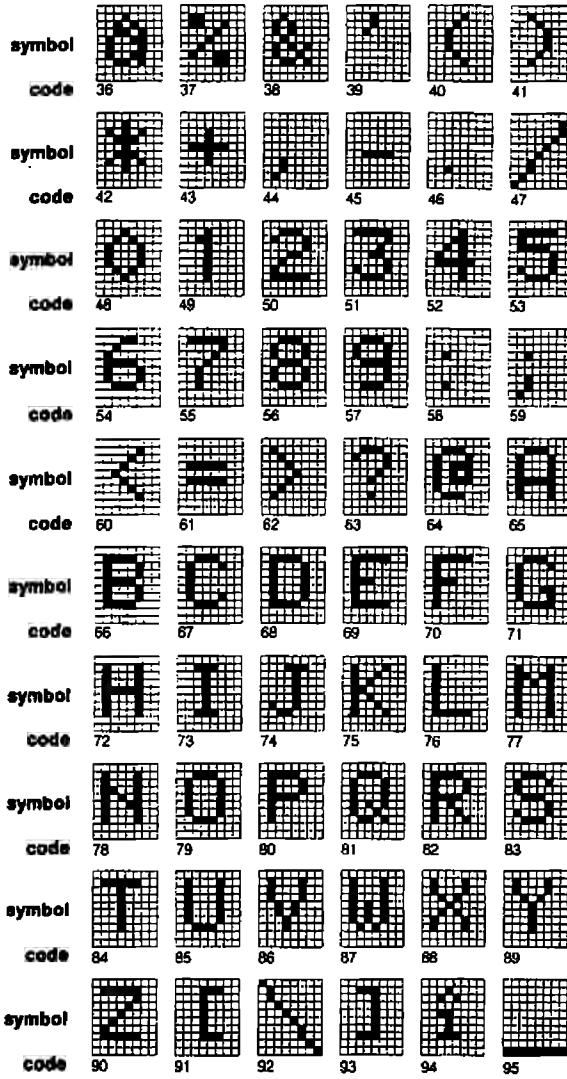
```
HYPERBELROSEKANTE = 2/(EXP(X)-EXP(-X))
HYPERBELKOTANGENS = EXP(-X)/(EXP(X)-EXP(-X))*2+1
AREASINUS         = LOG(X+SQR(X*X+1))
AREACOSINUS       = LOG(X+SQR(X*X-1))
AREATANGENS       = LOG((1+X)/(1-X))/2
AREASEKANTE       = LOG((SQR(-X*X+1)+1)/X
AREAKOSEKANTE     = LOG((SGN(X)*SQR(X*X+1)+1)/X
AREAKOTANGENS     = LOG((X+1)/(X-1))/2
```

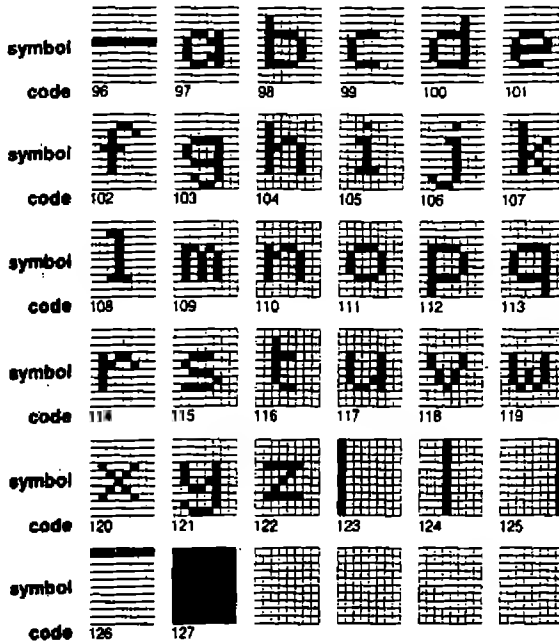
Anhang D

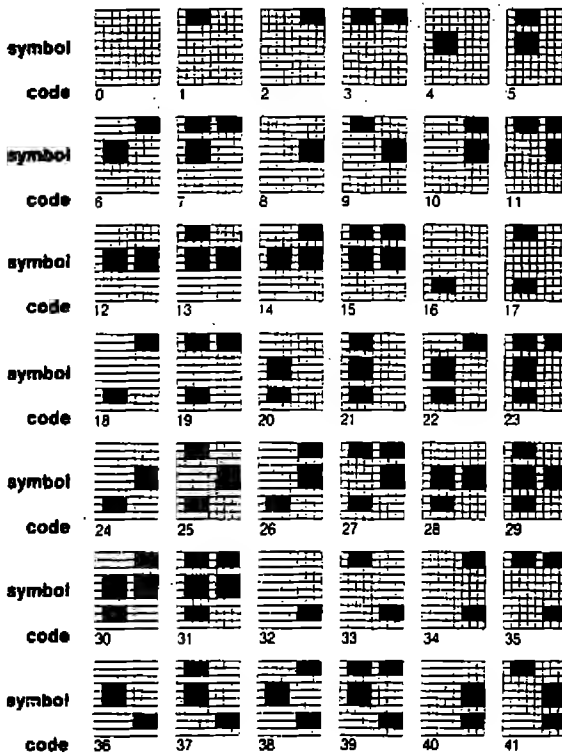
Diese Anhang ist im drei Teilen getrennt worden: Text Mode, Grafische Mode, Text und Grafische Mode.

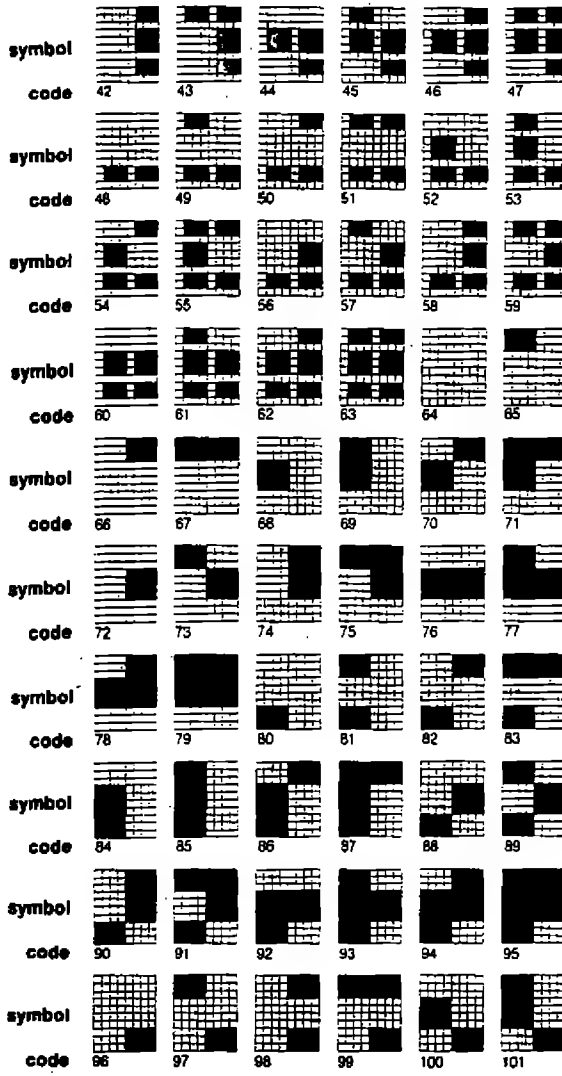
Im Text Mode

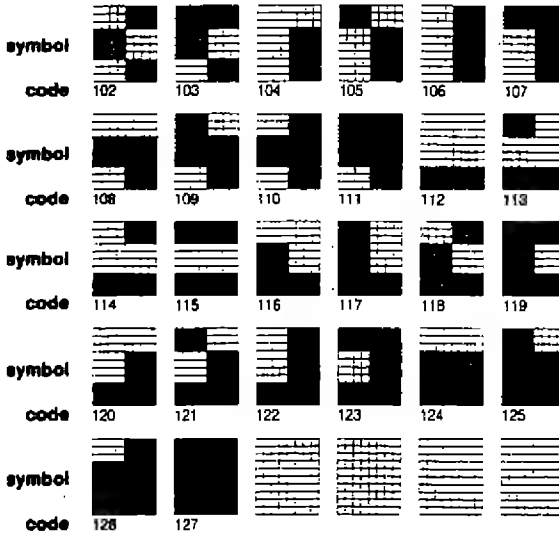



symbol
code
36
37
38
39
40
41
symbol
code
42
43
44
45
46
47
symbol
code
48
49
50
51
52
53
symbol
code
54
55
56
57
58
59
symbol
code
60
61
62
63
64
65
symbol
code
66
67
68
69
70
71
symbol
code
72
73
74
75
76
77
symbol
code
78
79
80
81
82
83
symbol
code
84
85
86
87
88
89
symbol
code
90
91
92
93
94
95


symbol
code
96
97
98
99
100
101
symbol
code
102
103
104
105
106
107
symbol
code
108
109
110
111
112
113
symbol
code
114
115
116
117
118
119
symbol
code
120
121
122
123
124
125
symbol
code
126
127


symbol
code
0 1 2 3 4 5
symbol
code
6 7 8 9 10 11
symbol
code
12 13 14 15 16 17
symbol
code
18 19 20 21 22 23
symbol
code
24 25 26 27 28 29
symbol
code
30 31 32 33 34 35
symbol
code
36 37 38 39 40 41


symbol
code
42
43
44
45
46
47
symbol
code
48
49
50
51
52
53
symbol
code
54
55
56
57
58
59
symbol
code
60
61
62
63
64
65
symbol
code
66
67
68
69
70
71
symbol
code
72
73
74
75
76
77
symbol
code
78
79
80
81
82
83
symbol
code
84
85
86
87
88
89
symbol
code
90
91
92
93
94
95
symbol
code
96
97
98
99
100
101


symbol
code
102
103
104
105
106
107
symbol
code
108
109
110
111
112
113
symbol
code
114
115
116
117
118
119
symbol
code
120
121
122
123
124
125
symbol
code
126
127

ZEICHENCODES

Im Text- und Graphik-Modus

Code	Zeich.	Code	Zeich.
128	NULL	144	INSCAR
129	NULL	145	SNDPIC
130	RUBOUT	146	SNDLIN
131	ENTER	147	NULL
132	DEOL	148	NULL
133	NULL	149	NULL
134	NULL	150	CLSYS
135	NULL	151	NULL
136	CURUPP	152	NULL
137	CURLEFT	153	NULL
138	CURDWN	154	INSLIN
139	CURRGT	155	SCRLU
140	HOME	156	SCRLDN
141	RETURN	157	CLEARA
142	NULL	158	CLEARC
143	NULL	159	VIDINI

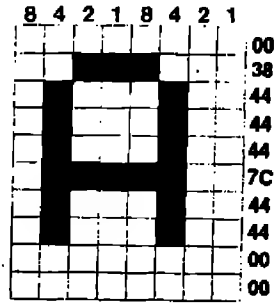
NULL = Keine Wirkung
RUBOUT = Löschen des Zeichens vor dem Cursor
ENTER = Carriage Return
DEOL = Löschen ab Cursor bis Zeilenende
CURUPP = Cursor 1 Zeile aufwärts
CURLEFT = Cursor 1 Stelle nach links
CURDWN = Cursor 1 Zeile abwärts
CURRGT = Cursor 1 Stelle nach rechts
HOME = Cursor auf X = 1 Y = 2
RETURN = Carriage Return und Zeilenvorschub
INSCAR = Ein Zeichen auf Cursorposition einfügen

SNDPIC = Gespeichertes Bild auf dem Bildschirm anzeigen
SNDLIN = Eine Zeile vom Bildspeicher auf dem Bildschirm anzeigen
INSLIN = Eine Zeile einfügen
SCRLUP = Bildschirm 1 Zeile nach oben laufen lassen
SCRLDN = Bildschrim 1 Zeile nach unten laufen lassen
CLEARC = Bildschirm von Cursorposition an löschen
VIDINI = Bildschirm auf Anfangswert setzen
CLSYS = System-Zeile löschen
CLEARA = Bildschirm löschen

Anhang E

Auch der Anwender kann Zeichen definieren. Das Maximum sind 96 verschiedene Zeichen im Text-Modus und 96 im Graphik-Modus.

Wenn man eigene Zeichen definieren will, muss man zuerst wissen, wie die Symbole auf den Bildschirm gezeichnet werden. Jedes Symbol besteht aus Punkten, die in einer Matrix von 8 mal 10 Punkten angeordnet sind. Zum Beispiel sieht der Grossbuchstabe "A" so aus:

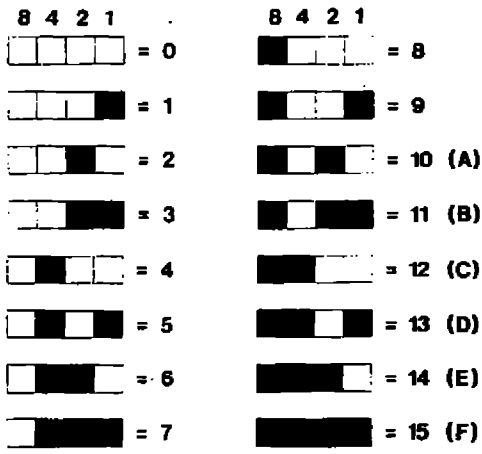


Die dunklen Quadrate werden auf dem Bilschirm in der gewählten Vordergrundfarbe dargestellt, während die hellen Quadrate in der gewählten Hintergrundfarbe gezeichnet werden. Jedes Symbol wird in einer 8 mal 10 Matrix gezeichnet. Da der Zeichen, in BASIC, in 23 Zeilen - jede mit 40 Zeichen - aufgeteilt ist, besteht der Bildschirm aus (40 x 8) 320 mal (23x10) 230 Punkte oder 73.600 Punkten insgesamt.

Jedes Quadrat der Matrix hat einen Wert, der wie folg
ausgedrückt werden kann:

8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1
8	4	2	1	8	4	2	1

So ist jede Zeile der Matrix in 2x4 Quadrate aufgeteilt. Die 4 Quadrate haben die Werte 8,4,2 und 1.
Vier Quadrate zusammen können einen Wert zwischen 0 und 15 haben;

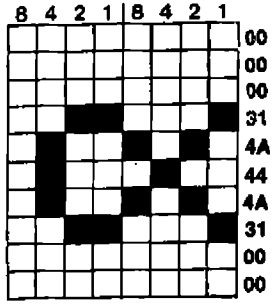


Die Werte zwischen 10 und 15 werden durch die Buchstaben A bis F dargestellt. Der Minimalwert in einer Zeichenmatrixzeile ist 00 und der Maximalwert ist FF.

Der Buchstabe "A", wie am Anfang dieses Anhangs gezeigt, kann mit der Notationstechnik folgendermassen definiert werden:

00 38 44 44 44 7C 44 44 00 00

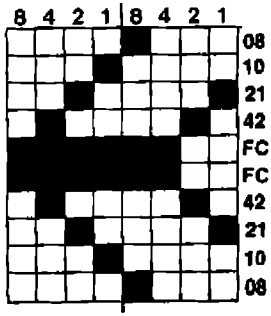
Der Anwender kann seine eigenen Zeichen in der gleichen Art definieren. Wenn gewünscht werden würde, den griechischen Buchstaben Alpha zu definieren, wird er folgendermassen in der Zeichenmatrix erscheinen:



Die Notation für dieses Symbol ist:

00 00 00 31 4A 44 4A 31 00 00

Spezielle Graphikzeichen können auch auf diese Art definiert werden. "Speziell" in diesem Sinne bedeutet dass diese Zeichen nicht in der Liste des Standard-Zeichnencodes im Graphik-Modus.



Die Codierung für dieses Zeichen ist:

08 10 21 42 FC FC 42 21 10 08

Jedes selbstdefinierte Zeichen muss vom Anwender mit einem Zeichencode versehen werden. Mit Hilfe dieses Zeichencodes können diese Zeichen in einem BASIC-Programm vom Anwender verwandt werden. Im Text-Modus sowie im Graphik-Modus können maximal je 96 Zeichen definiert werden. Die Zeichencodes 032 bis 127 können benutzt werden. Die Definition eigener Textzeichen kann mit der Anweisung SETET und selbstdefinierte Graphikzeichen mit SETEG durchgeführt werden. Durch Starten des speziellen Textmodus mit BASIC-Anweisung ET (siehe folgende Seiten) können die eigenen Textzeichen auf dem Bildschirm dargestellt werden.

Im Gegensatz zu den Zeichen, die in Anhang D aufgeführt sind, verbleiben selbstdefinierte Zeichen nicht im Speicher, wenn die RESET-Taste gedrückt oder der G7400 ausgeschaltet wird.

E.1 SETET

Format SETET I,"JJJJJJJJJJJJJJJJJJ"

Anwendung Um ein spezielles Zeichen zu definieren.

Anmerkung I ist der Zeichencode und muss zwischen 032 und 127 liegen.
J ist der Zeichen-Matrix-Code; JJ ist eine Symbol-Matrix-Zeile.

J muss in dieser Anweisung 20 mal benutzt werden.
J muss zwischen 0 und F liegen.

Beispiel 10 SETET 035,"000000314A444A310000"

ist die Definition des graphischen Buchstaben "alpha". Dieser Buchstabe hat den Zeichencode 35.

E.2 SETEG

Format SETEG I, "JJJJJJJJJJJJJJJJJJ"

Anwendung Um ein spezielles Graphik-Zeichen zu definieren.

Anmerkung I ist der Zeichen-Codeund muss zwischen 032 und 127 liegen. J ist der Zeichen-Matrix-Code. JJ ist eine Symbol-Matrix-Zeile.
J muss in dieser Anweisung 20 mal benutzt werden.
J muss zwischen 0 und F liegen.

Beispiel

```
10 SETEG 071,"08102142FCFC42211008"
```

Die Zeile ist die Definition des schon gezeigten Graphik-Zeichens. Dieses Zeichen hat den Zeichencode 71.

E.3 ET

Format ET I,J,K

Anwendung Um den speziellen Text-Modus auf die Anfangswert zu setzen.

Anmerkung I ändert die Vordergrundfarbe und muss eine ganze Zahl sein:

0 = schwarz
1 = rot
2 = grün
3 = gelb
4 = blau
5 = purpur
6 = hellblau
7 = weiss

J ändert den Status der vorher definierten Zeichen und muss eine ganze Zahl sein:

0 = normal
1 = doppelte Höhe
2 = doppelte Breite
3 = doppelte Höhe und Breite
4 = Negativ-Bild
5 = Negativ-Bild und doppelte Höhe
6 = Negativ-Bild und doppelte Breite
7 = Negativ-Bild, doppelte Höhe und Breite

K muss eine ganze Zahl sein.

0 = stehendes Bild
1 = blinkendes Bild

Die Hintergrundfarbe der vordefinierten Textzeichen ist die aus der letzten GR,DELIM oder EG-Anweisung.

Beispiel

```
10 ET 4,0,0
20 PRINT CHR$(35)
```

E.4 BG

Format BG I,J,K

Anwendung Um den speziellen Graphik-Modus auf die Anfangsweste zu setzen.

Anmerkung I ändert die Vordergrundfarbe und muss eine ganze Zahl sein.

0 = schwarz
1 = rot
2 = grün
3 = gelb
4 = blau
5 = purpur
6 = hellblau
7 = weiss

J ändert die Hintergrundfarbe von der Cursorposition an bis zum Ende der Zeile und muss ein ganzer Ausdruck sein.

0 = schwarz
1 = rot
2 = grün
3 = gelb
4 = blau
5 = purpur
6 = hellblau
7 = weiss

K muss ein ganzer Ausdruck sein

0 = stehendes BILD
1 = blinkendes BILD

Beispiel

```
10 BG 1,4,0
20 PRINT CHR$(71)
```

Anhang F

Die folgenden Wörter dürfen nicht als Variablennamen benutzt werden:

ABS	GR	READ
ACTION	GOSUB	REM
AND	GOTO	RESTORE
ASC		RETURN
ATN	IF	RIGHT$
	INIT	RND
BRIGHT	INPUT	RUN
	INT	
CHR$		SCREEN
CLEAR	KEY	SCROLL
CLOAD		SETEG
CONT	LEFT$	SETET
COS	LEN	SGN
CURSORX	LET	SIN
CURSORY	LINE	SOUND
CSAVE	LIST	SPL
	LOG	SQR
DATA		STEP
DEF	MID$	STICKX
DELIM		STICKY
DIM	NEW	STOP
DISPLAY	NEXT	STORE
		STR$
EG	ON	
END	OR	TAB
ET		TAN
EXP	PAGE	TX
	PEEK	
FOR	POKE	USR
FOUND	POS	
FRE	PRINT	VAL