creativision

BASIC

INTERPRETER

REFERENZ-HANDBUCH

Vers.1:0 Video Technology Ltd. 1982

Copyright: SANYO VIDEO Vertrieb GmbH & Co., 2000 Hamburg 1 - Februar 1983

Inhalt

VORWORT

Dieses Handbuch soll den Anfaenger bei seinen ersten BASIC-Versuchen mit dem CreatiVision-System begleiten und ihm, beginnend auf der untersten Ebene, mit den Grundlagen der BASIC-Programmsprache und der Erstellung von Programmen mit dem CreatiVision-System vertraut machen. Es beinhaltet die grundlegenden Konzepte und Anweisungen der BASIC-Programmsprache und verlangt keine Vorkenntnisse vom Leser.

Der Leser sollte neue Anweisungen und Programmkonzepte ausprobieren und die Beispiel-Programme nachvollziehen. Das Verstaendnis kann durch Veraendern der Beispiele vertieft oder korrigiert werden. So bald wie moeglich sollte mit dem Erlernten versucht werden, kleine Probleme aus dem eigenen Interessenbereich zu loesen. Nur auf diese Art und Weise koennen Moeglichkeiten und Beschraenkungen der BASIC-Problemlösungen erlernt und vertieft werden.

Durch Eingabe ueber die Tastatur koennen auf keinen Fall irgendwelche Defekte oder Zerstoerungen im Computersystem ausgeloest werden. Fuer Anweisungen, die schon erzeugte Daten im Speicher loeschen koennten, sind Vorsichtsmassnahmen in der Beschreibung angegeben. Der Leser kann in jeder Phase des Lernens frei mit dem CreatiVision Computersystem arbeiten.

Grundsaetzlich sollte der Leser dem Aufbau des Handbuches folgen. Einzelne Kapitel, wie Kapitel 12: Kassettenspeicher, koennen ohne weiteres vorgezogen werden, wenn, wie hier, das Abspeichern der Programme auf Tonband geuebt werden soll. Dieses Handbuch ist zwar fuer den Benutzer des CreatiVision-Computersystems geschrieben worden, es kann aber auch als grundsaetzliche Einweisung in das Konzept des Programmierens in BASIC benutzt werden. Der Leser sollte aber wissen, dass die in den verschiedenen Mikrocomputersystemen implementierten BASIC-Konzepte in vielen Einzelheiten unterschiedlich sind.

Wir hoffen, das dieses Handbuch Ihnen hilft, mit den BASIC-Grundlagen vertraut zu werden.

Wir wuenschen Ihnen viel Spass mit dem
CreatiVision Computer System !

SVV

Kapitel 1

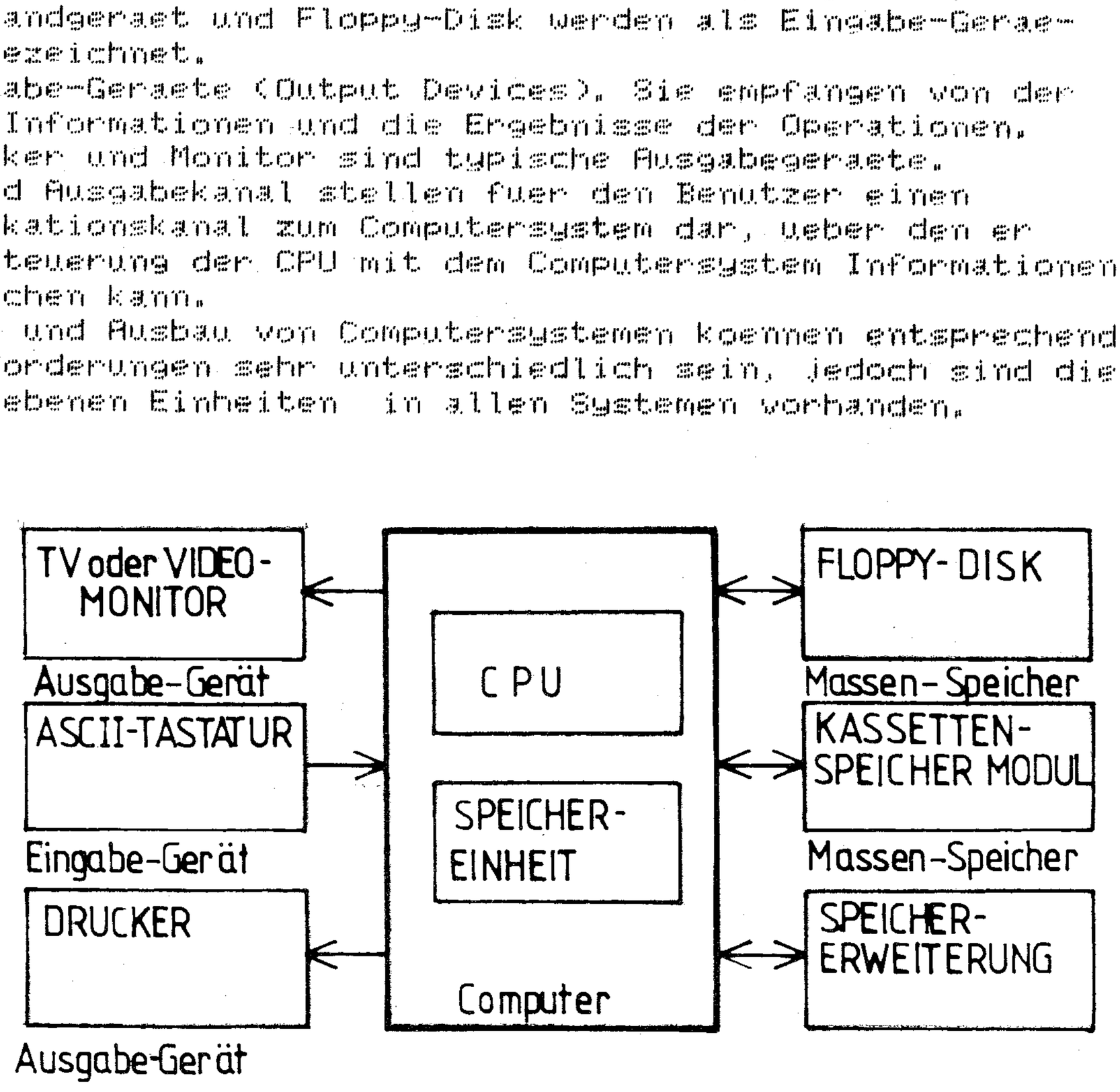
Einleitung
—Was ist ein Computer ?

Computer sind Geraete, die mittels einer vom Benutzer eingegebenen Anweisungsliste Taetigkeiten ausfuehren.

1) Die CPU (central processing unit). Sie fuehrt die in der Instruktionsliste notierten Anweisungen aus. Es koennen arithmethische, logische und speicher-bezogene Operationen ausgefuehrt werden. Man kann die CPU als das Gehirn des Computers bezeichnen.
2) Die Speicher-Einheit. In ihr werden die Instruktionsliste und alle von der CPU oder vom Benutzer gegebenen Informationen abgelegt. Die Speicher-Einheit befindet sich innerhalb des Computers. Die CPU kann direkt auf alle Instruktionen und Informationen zugreifen.
3) Der Massen-Speicher. Er befindet sich ausserhalb des Computers und nimmt ebenfalls Instruktionslisten und Informationen von Benutzer oder CPU auf. Tonbandgeraet und Floppy-Disk sind Massenspeicher. Alle Informationen und Instruktionen muessen zur Bearbeitung durch die CPU in den Computerspeicher transportiert werden.
4) Eingabe-Geraet (Input-Device). Eingabegeraete geben dem Benutzer die Moeglichkeit, Daten oder Instruktionen in den Computer zu geben. Tastatur, Joystick, Tonbandgeraet und Floppy-Disk werden als Eingabe-Geraete bezeichnet.
5) Ausgabe-Geraete (Output Devices). Sie empfangen von der CPU Informationen und die Ergebnisse der Operationen. Drucker und Monitor sind typische Ausgabegeraete.

Ein- und Ausgabekanal stellen fuer den Benutzer einen Kommunikationskanal zum Computersystem dar, ueber den er unter Steuerung der CPU mit dem Computersystem Informationen austauschen kann.

Groesse und Ausbau von Computersystemen koennen entsprechend den Anforderungen sehr unterschiedlich sein, jedoch sind die beschriebenen Einheiten in allen Systemen vorhanden.



Kapitel 2

Definitionen

—Programm-Konzept
—BASIC-Interpreter

PROGRAMM-KONZEPT

Das Entwerfen einer Instruktionsliste wird mit dem Wort programmieren bezeichnet; die Instruktionsliste selber wird als Programm bezeichnet. Der Programmierer ist derjenige, der das Programm erstellt.

Um ein Programm fuer einen Computer zu erstellen, ist in zwei Schritten vorzugehen. Erstens muss der Benutzer wissen, welche Instruktionen er anzuwenden hat und in welcher Form er sie definieren muss, und zweitens muss er in der Lage sein, in einer Form der Kommunikation die definierte Instruktionsliste dem Computer zu uebermitteln. In diesem Falle findet eine Kommunikation statt: durch das Schreiben des Programmes in einer "Programm-Sprache" und das Lesen und Ausfuehren dieses Programmes durch den Computer.

Heute sind sehr viele verschiedene Programm-Sprachen im Gebrauch. Sie sind teilweise fuer spezielle Anwendungen vorgesehen, teilweise jedoch decken sie ein breites Feld von Anwendungen ab.

BASIC ist in die letzte Kategorie einzuordnen.

DER BASIC-INTERPRETER

BASIC ist eine Abkuerzung fuer "Beginner's All-purpose Symbolic Instruction Code". BASIC hat ein einfaches, englisches Vokabular und nur wenige grammatikalische Regeln. BASIC folgt einfachen, mathematischen Grundsaetzen.

Wenn diese Programmsprache auch grundsaetzlich einfach ist, so laesst sie doch die Loesung schwierigster Aufgaben zu. Sie ermoeglicht die Loesung aller arithmetischen Operationen, logische Vergleiche, Aufbau von mehrdimensionalen Feldern, Listen, allgemeine trigonometrische Funktionen und die Manipulation von Zeichenketten.

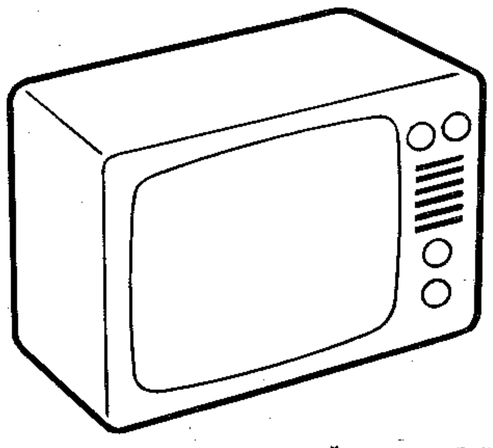
In BASIC geschriebene Programme werden von einem Uebersetzungsprogramm in die Sprache der CPU gebracht. Dieses Uebersetzungsprogramm wird BASIC-Interpreter genannt. Es befindet sich in der eingesteckten BASIC-Kassette.

Das naechste Kapitel zeigt den Aufbau und die Inbetriebnahme des CreatiVision-Systems. Die folgenden Kapitel erklaeren mit Beispielen die Funktion der verschiedenen BASIC-Anweisungen.

Kapitel 3

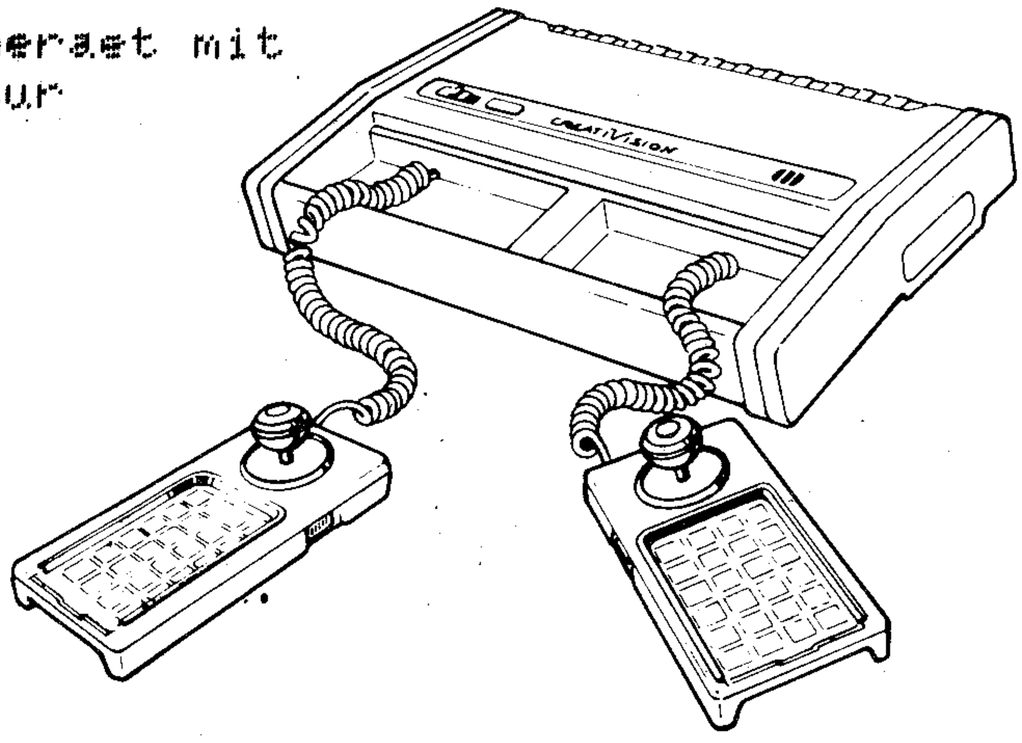
Inbetriebnahme und Anschluss des CreatiVision-Systems

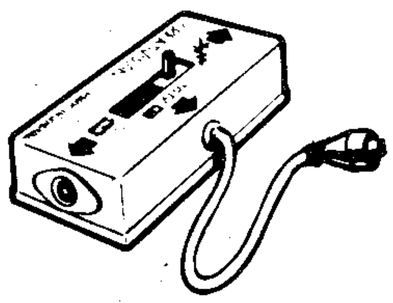
Die folgenden Teile benoetigen Sie, um mit dem CreatiVision-System zu arbeiten.



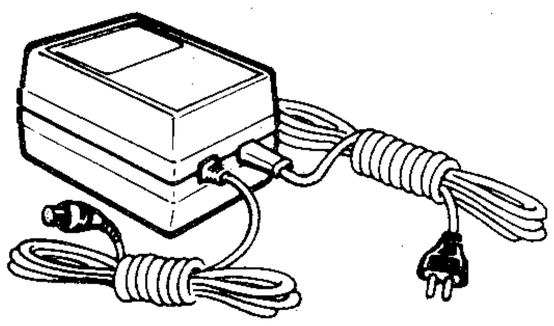
TV-Colorempfaenger

Hauptgeraet mit Tastatur

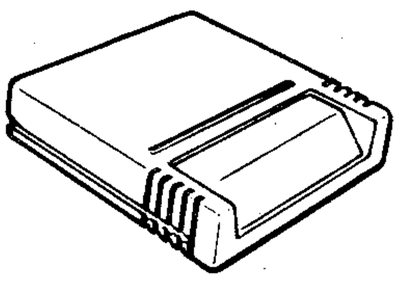




Antennenumschalter



Netzadapter

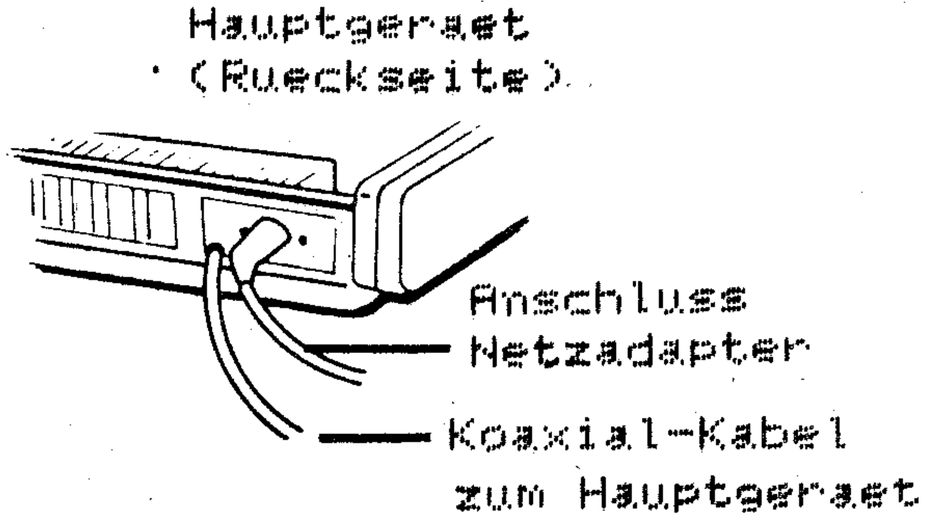


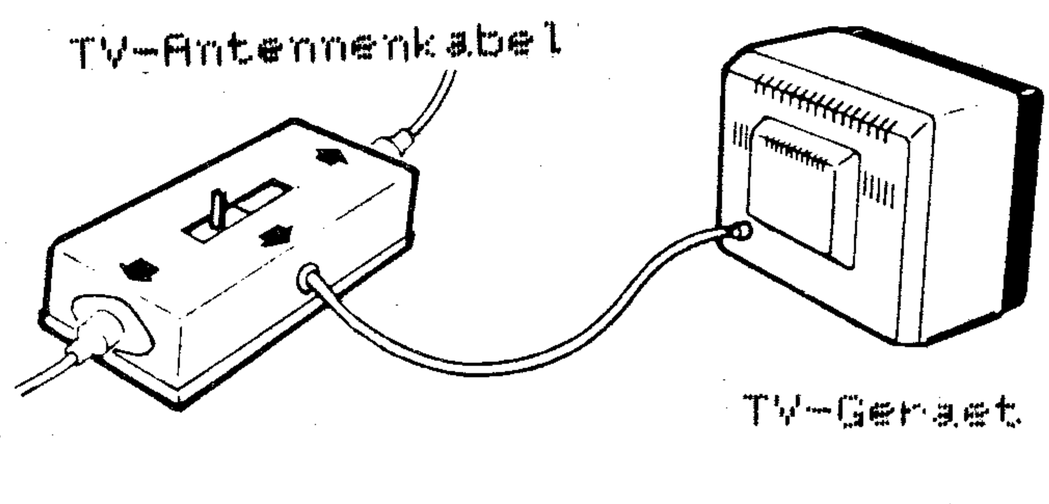
BASIC-Kassette

ANTENNENUMSCHALTER

Mit dem Antennenumschalter koennen Sie Ihr Fernsehgeraet leicht fuer die Arbeit mit dem Suetem oder fuer normalen TV-Empfang einsetzen.

* Ziehen Sie das Antennenkabel von Ihrem TV-Geraet ab und schliessen Sie es an den Antennenumschalter an.
* Das Kabel des Antennenumschalters schliessen Sie nun an die Antennenbuchse Ihres TV-Empfaengers an.
* Zuletzt schliessen Sie das Kabel vom CreatiVision-Geraet an den Antennenumschalter an.





DIE SCHRITTWEISE INBETRIEBNAHME ALS COMPUTER-SYSTEM

1) Das CreatiVision-Telespielgeraet muss ausgeschaltet sein.
2) Schliessen Sie den Netzadapter an das Telespielgeraet an.
3) Verbinden Sie dann den Netzadapter mit der Steckdose.
4) Stellen Sie den Schalter am Antennenumschalter auf "GAME"
5) Stecken Sie die BASIC-Kassette in die vorgesehene Oeffnung am Hauptgeraet.
6) Schalten Sie Ihr TV-Geraet an und stellen Sie den vorgesehenen Kanal (VHF, BAND I, KANAL 2) ein.
7) Stellen Sie den Schalter am Hauptgeraet auf "ON". Sollten Sie Ihr System zum erstenmal in Betrieb nehmen, muessen Sie nun Ihr TV-Geraet abstimmen, ansonsten meldet sich der CreatiVision Personal Computer sofort auf dem Bildschirm.
   Anmerkung: Bei TV-Geraeten mit automatischer Feinabstimmung sollte Ihre Einstellung bei abgeschalteter Automatik erfolgen.
8) Ihr CreatiVision-System ist jetzt bereit und wartet auf Ihre Anweisungen.

VORSICHTSMASSNAHMEN.

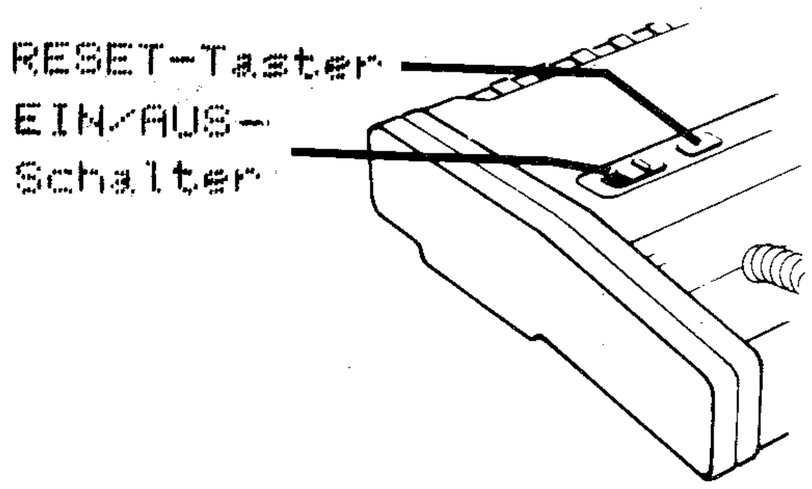
1) Schuetzen Sie alle Teile des Systems vor Feuchtigkeit.
2) Achten Sie darauf, dasskein Teil des Systems extremer Hitze ausgesetzt wird und den Geraeten im Betrieb eine ausreichende Luftzirkulation ermoeglicht wird.
3) Behandeln Sie die Geraete vorsichtig, lassen Sie sie nicht fallen.
4) Wechseln Sie die Kassetten vorsichtig und schalten Sie das Geraet zum Wechseln der Kassetten STETS aus!
5) Beruehren Sie auf keinen Fall die in die Kassette versenkten Anschlusskontakte. Durch die Aufladung Ihres Koerpers mit statischer Elektrizitaet koennen Bauteile in der Kassette zerstoert werden.
6) Entfernen Sie die Kassette, wenn das Geraet nicht benutzt wird.

RUECKSCHALTEN AUF TV-EMPFANG

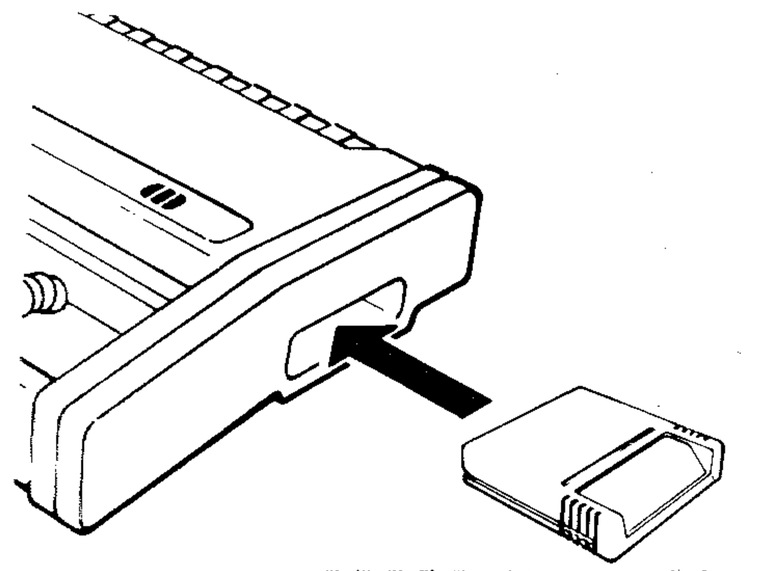
Schalten Sie den Computer ab und stellen Sie den Antennenumschalter auf "TV". Das Fernsehgeraet laesst sich nun wieder normal benutzen.

ZUSAMMENFASSUNG DER INBETRIEBNAHME

1) Setzen Sie die BASIC-Kassette vorsichtig ein.
2) Schliessen Sie den Netzadapter erst an das Geraet, dann an die Steckdose an.
3) Schalten Sie den Antennenumschalter auf "GAME".
4) Achten Sie auf den richtigen Anschluss der Antennenkabel.
5) Schalten Sie den Computer und das TV-Geraet ein.
6) Schalten Sie Ihr TV-Geraet auf den richtigen Kanal.



SVV



Kapitel 4

Die ersten Schritte zum Programmieren

-Direkte Befehlsausfuehrung
-Programmierte Befehlsausfuehrung
-Aufbau einer Anweisungszeile
-Editieren einer Anweisungszeile
-Die LIST-Anweisung
-Das Loeschen einer Anweisungszeile
-LLIST - Programmausgabe zum Drucker
-NEW - Loeschen des gesamten Programmes

Die ersten Schritte zum Programmieren

Der Computer fuehrt "Statements" aus (die vom Programmierer gegebenen Anweisungen). Diese Ausfuehrung kann direkt von der Tastatur oder aus einem Programm veranlasst werden.

Die direkte Ausfuehrung von Anweisungen.

Jede Anweisung, die nicht durch eine vorgestellte "Zeilennummer" in ein Programm eingestellt wird, wird sofort nach dem Druecken der "RETURN"-Taste ausgefuehrt.

```
Beispiel:    PRINT 2+4   (RETURN)
             6
             PRINT 2+4,3/6   (RETURN)
             6             0.5
             PRINT "ERGEBNIS",(2+4)/(3/6)   (RETURN)
             ERGEBNIS      12
```

Durch die direkte Anweisungsausfuehrung laesst sich der Computer wie ein Taschenrechner einsetzen. Zusaetzliche Textausgabe und die gleichzeitige Berechnung mehrerer mathematischer Ausdruecke uebertreffen einen Taschenrechner.

RUN

Die programmierte Ausfuehrung von Anweisungen.

Jede Anweisung, der eine "Zeilennummer" vorangestellt ist, wird nicht nach Druecken der Taste RETURN ausgefuehrt, sondern wird in den BASIC-Programmspeicher eingeschrieben. Eine Ausfuehrung dieser Anweisungen wird mit dem Schreiben des Kommandos RUN, gefolgt von (RETURN), gestartet.

```
Beispiel:    10 PRINT 2+4   (RETURN)
             RUN   (RETURN)
             6

             10 PRINT 2+4,3/6   (RETURN)
             RUN   (RETURN)
             6             0.5
```

```
10 PRINT 7+9  <RETURN>
20 PRINT 2+4,3/6  <RETURN>
30 PRINT "RESULT",(2*6)  <RETURN>

RUN  <RETURN>
16
6                0.5
RESULT           12
```

Das letzte Beispiel zeigt, dass die Zeile mit der Nummer 10 zuerst, dann die Zeile 20 und zuletzt die Zeile 30 ausgefuehrt wird. In dieser Reihenfolge werden alle Anweisungen eines Programmes ausgefuehrt. Anweisungszeilen mit niedrigeren Zeilennummern werden vor den mit hoeheren Zeilennummern ausgefuehrt.

Es ist moeglich, im Programm mit den Anweisungen GOTO und GOSUB die Ausfuehrung von Anweisungen mit beliebigen Zeilennummern zu veranlassen (siehe Kapitel 9).

Mit dem Kommando RUN beginnt der Computer die Abarbeitung des Programmes mit der Anweisungszeile, welche die niedrigste Zeilennummer traegt. Soll der Start an anderer Stelle erfolgen, so ist das Kommando

RUN (Zeilennummer) zu geben <RETURN>.

Beispiel:

```
10 PRINT 7+9  <RETURN>
20 PRINT 2+4,3/6  <RETURN>
30 PRINT "ERGEBNIS",(2*6)  <RETURN>

RUN 20  <RETURN>
6              0.5
ERGEBNIS   12
```


Aufbau einer Anweisungszeile.

1) Jede Anweisungszeile beginnt mit einer Zeilenummer.
   Beispiel: 10, 20,30, usw.
2) Eine Zeilennummer ist eine vorzeichenlose, ganze Zahl (Integer) von 0 bis 9999
3) Auf die Zeilennummer folgt die korrekte BASIC-Anweisung. Im anderen Falle bricht der Computer die Programmausfuehrung ab und gibt die fehlerhafte Zeile gefolgt von der Meldung "SYNTAX-ERROR" auf den Bildschirm aus.

Beispiel:

```
10 PRIMT 2+3  <RETURN>
RUN  <RETURN>
10 PRIMT 2+3
SYNTAX ERROR
```

4) Jede Programmzeile wird mit Druecken der Taste <RETURN> in das Programm uebernommen.

EDITIEREN

Wird bei der Eingabe ein Fehler gemacht, so kann mit der Taste (<-) die fehlerhafte Eingabe geloescht werden.

Beispiel:

```
PRIMT
Taste <- druecken„ Der Bildschirm zeigt
PRIM
Taste <- noch einmal druecken
PRI
und nun kann das Wort korrekt geschrieben werden.
```

LIST

Um das im Speicher befindliche Programm auf den Bildschirm zu bringen, wird das "LIST"-Kommando benutzt. List bildet die Anweisungszeilen in der Reihenfolge der Zeilennummern ab.

```
Beispiel:       Eingabe:
                47 PRINT "GOODBYE"  <RETURN>
                25 PRINT 2+3  <RETURN>
                33 PRINT 4+3  <RETURN>
                12 PRINT "HELLO"  <RETURN>
                LIST  <RETURN>
                12 PRINT "HELLO"
                25 PRINT 2+3
                33 PRINT 4+3
                47 PRINT "GOODBYE"
```

Wird auf das LIST-Kommando folgend eine Zeilennummer eingegeben, so wird nur die so spezifizierte Programmzeile auf den Bildschirm gebracht.

```
Beispiel:       LIST 47  <RETURN>
                47 PRINT "GOODBYE"
```

Werden dem LIST-Kommando zwei, durch Kommas getrennte Zeilennummern nachgestellt, so wird der Programmtext von der ersten Nummer bis zur zweiten Nummer auf den Bildschirm gebracht.

```
Beispiel:       LIST 25,33  <RETURN>
                25 PRINT 2+3
                33 PRINT 4+3
```


LOESCHEN EINER ZEILE

Um eine Anweisungszeile aus dem Programm zu loeschen, ist die Zeilennummer gefolgt von <RETURN> einzugeben.

```
Beispiel:       12  <RETURN>
                LIST  <RETURN>
                25 PRINT 2+3
                33 PRINT 4+3
                47 PRINT "GOODBYE"
```

Die Anweisungszeile mit der Nummer 12 ist nun geloescht. Versuchen Sie selber, Zeile 33 zu loeschen.

LLIST

LLIST arbeitet wie LIST, die Ausgabe des Programmes erfolgt jedoch nicht zum Bildschirm, sondern zum angeschlossenen Drucker.

Syntax: LLIST

Wird LLIST angewendet, muessen das I/O-Interface und der Drucker betriebsbereit sein. Einzelheiten sind dem Handbuch des I/O-Interface zu entnehmen.

NEW

Die Anweisung "NEW" ermoeglicht das Loeschen des gesamten Programmspeichers.

Beispiel:
```
NEW  <RETURN>
LIST  <RETURN>
>
```

Auf das LIST-Kommando erfolgt keine Programmausgabe mehr, der BASIC-Programmspeicher ist geloescht.

ANMERKUNG

Sie werden jetzt mit dem Gebrauch der <RETURN>-Taste ausreichend vertraut sein.
Ab dem naechsten Kapitel wird die Anwendung dieser Taste in den Beispielen nicht mehr ausgedruckt.

Kapitel 5

Zahlen und Variablen

-Numerische Konstanten
-Numerische Variablen
-LET - Anweisung
-REM - Anweisung
-ABS - Funktion
-SGN - Funktion
-RND - Funktion

NUMERISCHE KONSTANTEN

Numerische Konstanten sind Zahlen, die waehrend des gesamten Programmablaufes ihren Wert beibehalten. Sie koennen positive oder negative Vorzeichen haben. Sie werden dezimal oder technisch-wissenschaftlich notiert.

Beispiel: +2, 34, 0.432, 3E10, 4E(-35), usw.

Der Bereich einer numerischen Konstante ist:
10 <= n <= 10

NUMERISCHE VARIABLEN

Eine Variable ist eine Information, die waehrend des Programmlaufes ihren Wert oder Inhalt aendern kann. Eine numerische Variable wird vom Programmierer mit einem festgelegten Namen beschrieben. Alle Buchstaben von A-Z koennen als Variablennamen benutzt werden, so das 26 Variablen vereinbart werden koennen. Der Wert der Variablen bleibt solange unveraendert, wie ihr nicht mit den Anweisungen LET oder INPUT ein neuer Wert zugewiesen wird oder der Wert nicht mit der Anweisung FOR veraendert wird.

Das Starten des Programmes mit RUN setzt alle numerischen Variablen gleich Null. Eine Variable muss im Programm mit Namen und Wert beschrieben werden, wenn ihr Anfangswert ungleich Null sein soll.

Es ist gute Programmierpraxis, zu Beginn des Programmes alle Variablen mit ihrem Anfangswert zu beschreiben.

LET

Die LET - Anweisung

Diese Anweisung weisst einer Variablen links vom "="-Zeichen den Wert eines mathematischen Ausdruckes rechts vom "="-Zeichen zu.

Jede LET-Anweisung hat die Form

LET (Variable) = Ausdruck

Die Anweisung ist keinesfalls eine algebraische Gleichung, aber sie ermoeglicht die Definition von Ausdruecken und weist das Ergebnis der benannten Variable zu.

SGN

Die SGN-Funktion

Mit der SGN-Funktion laesst sich ermitteln, ob der Wert einer Zahl positiv, negativ oder gleich Null ist.
Die SGN-Funktion bewertet eine positive Zahl mit +1, eine negative mit -1 ,und wenn die Zahl gleich Null ist, mit 0.
Beispiele: SGN(3.34)=1; SGN(-42)=-1; SGN(23-23)=0.
Syntax: SGN (Ausdruck)

```
Beispiel:    10 A=-12
             20 PRINT SGN(A); SGN(A-A)

             RUN
             -1   0
```


RND

Die RND (0) - Anweisung

Die Funktion RND(0) ermittelt Zufallszahlen im Bereich zwischen 0 und 1.
Syntax: RND (0)

```
Beispiel:   10 FOR  I=1 TO 5
            20 PRINT RND (0)
            30 NEXT I
            40 END

            RUN
            0.53675
            0.1463
            0.80221
            0.34245
            0.36985
```

Die Funktion RND (N)

Diese Funktion ermittelt Zufallszahlen im Bereich von 0 bis N. N ist eine positive Zahl, das Ergebnis eine ganze, positive Zahl (Integer) zwischen 0 und N.
Syntax: RND (N)

```
Beispiel: 10 FOR I=1 TO 5
          20 PRINT RND (10)
          30 NEXT I
          40 END

          RUN
          5
          4
          7
          2
          8
```

```
Beispiel:     10 LET A=5
              20 LET B=20
              30 LET C=60
              40 LET A=A+5
              50 PRINT A+B+C
              60 END

              RUN
              90
```

In Zeile 40 wird der alte Wert von A in Zeile 10 um 5 erhoeht, das Ergebnis 10 wird dann der Variable A zugewiesen.

Der BASIC-Interpreter erlaubt das Auslassen des Wortes LET.

```
Beispiel:     10 A=5
              20 B=20
              30 C=60
              40 A=A+5
              50 PRINT A+B+C
              60 END

              RUN
              90
```


REM

Die REM - Anweisung

Diese Anweisung erlaubt es dem Programmierer, Hinweise und Informationen zur Dokumentation des Programmes im Programmlisting zu notieren. Der BASIC-Interpreter ignoriert diese Zeilen.
Syntax: REM <Text>

```
Beispiel:     10 REM DIES IST EINE ANMERKUNG
              20 REM BASIC PROGRAMMIERUNG
              30 A=3
              40 PRINT A
              50 END

              RUN
              3
```


ABS

Die ABSOLUT-Funktion

Die ABS-Funktion formt den absoluten Wert eines Ausdruckes.
Als Beispiel: ABS (-34,67) = 34,67
Syntax: ABS (Ausdruck)

```
Beispiel:     10 PRINT ABS (3+4-6*5)

              RUN
              23
```

Kapitel 6

Daten-Bearbeitung

- Arithmetisch Bearbeitung
- Vergleichende Bearbeitung
- Logische Vergleiche
- BASIC-Funktionen

MATHEMATISCHE OPERATOREN

OPERATOREN

Es sind 4 Arten Operatoren im CreatiVision-Basic:

Mathematische Operatoren

Der BASIC-Interpreter kann Addition, Subtraktion, Multiplikation, Division und Exponentation ausfuehren. Zu errechnende mathemathische Formeln koennen in der gewohnten Form im Programm notiert werden. Es stehen fuenf Operatoren zur Verfuegung:

Operator	Beispiel	Bedeutung
+	A + B	Addiere B zu A
-	A - B	Subtrahiere B von A
*	A * B	Multipliziere A mit B
/	A / B	Dividiere A durch B
**	A ** B	Rechne A zur Basis B

Der Gebrauch von Klammern in Ausdruecken aendert die Wertigkeit der Operatoren. Operationen innerhalb von Klammern werden zuerst bearbeitet. Innerhalb der Klammern gilt wieder die Wertigkeit der Operatoren.

Arithmetische Formeln werden entsprechend den folgenden Regeln bearbeitet, wobei die unter 1) notierten Regeln vorrangig sind.

1) Jeder Ausdruck in Klammern wird berechnet, bevor er im weiteren Verlauf der Formel gebraucht wird. Wenn Klammerebenen ineinander verschachtelt sind (Beispiel: A+(B*(D**2))), werden die inneren Ausdruecke zuerst berechnet.
2) Werden keine Klammern gesetzt, werden die Operationen zur Errechnung in der folgenden Reihenfolge vom BASIC-Interpreter bearbeitet:
   Exponentation
   Neg. Vorzeichen aufloesen
   Multiplikation und Division
   Addition und Subtraktion

Der Interpreter fasst -A**B als -(A**B) auf. Das bedeutet, dass -2**2 zu -4 werden statt wie erwartet, +4. Entsprechend dieser Regel wird der Term A**-B als A**(-B) errechnet.

3) Folgen in Terms ohne Klammern Operatoren mit gleicher Wertigkeit, so wird von links nach rechts, wie geschrieben, gerechnet. Als Beispiel:

A/B/C wird als (A/B)/C errechnet.
A*B/C wird als (A*B)/C errechnet.

Der Term A+B*C**D wird in folgender Reihenfolge errechnet:

1) C**D wird errechnet.
2) Das Resultat aus 1) wird mit B multipliziert.
3) Das Resultat aus 2) wird zu A addiert.

Klammern sollten nur dort gesetzt werden, wo sie notwendig sind, um die Ausdruecke uebersichtlicher und fehlerfreier zu gestalten.

Beispiele:	Algebraischer Ausdruck	BASIC-Ausdruck
	3X-2Y	3*X-2*Y
	(X Y	X**2*Y
	B -4AC	B**2-4*A*C

VERGLEICHS-OPERATOREN

Vergleichs-Operatoren

Mit diesen Operatoren werden zwei Werte miteinander verglichen. In Abhaengigkeit vom Ergebnis dieser Vergleiche kann der Programmablauf veraendert werden. Das Ergebnis der Vergleichsoperation ist 1, wenn der Vergleich wahr ist, und 0, wenn der Vergleich unwahr ist.

Operator	Getestetes Verhaeltnis	Ausdruck
=	Gleich	X=Y
><	Ungleich	X><Y, X<>Y
<	Kleiner als	X<Y
>	Groesser als	X>Y
<=,=<	Kleiner oder gleich als	X<=Y, X=<Y
>=,=>	Groesser oder gleich als	X>=Y, X=>Y

Beispiel:

```
10 INPUT A,B
20 IF A<B THEN PRINT "A<B"
30 IF A=B THEN PRINT "A=B"
40 IF A>B THEN PRINT "A>B"
50 END
```

```
RUN
?10
?5
A>B
```

LOGISCHE OPERATOREN

Logische Operatoren

Logische Operatoren werden in IF...THEN Anweisungen benutzt, um die Ergebnisse mehrerer Vergleiche logisch miteinander zu verknuepfen. A und B in der nachfolgenden Beschreibung sind Verhaeltnis-Ausdruecke mit den Werten "wahr" (1) und "unwahr" (0). Logische Operatoren werden nach arithmetischen und Verhaeltnis-Operatoren ausgefuehrt.

Operator	Beispiel	Bedeutung
NOT	NOT A	Logische Invertierung von A. A ist wahr, NOT A ist unwahr.
AND	A AND B	Das logische Produkt von A und B. A AND B hat den Wert "wahr", wenn beide "wahr" sind. Der Ausdruck ist "unwahr", wenn A oder B "unwahr" sind.
OR	A OR B	Die logische Summe von A und B. Der Ausdruck ist "wahr", wenn A oder B "wahr" sind, und "unwahr", wenn A und B "unwahr" sind.

Die folgenden Tabellen werden Wahrheitstabellen genannt. Sie zeigen die Ergebnisse der besprochenen logischen Operationen fuer alle Kombinationen fuer A und B.

A	I	NOT A
wahr	I	unwahr
unwahr	I	wahr

Wahrheitstabelle der "NOT"-Funktion

A	B	I	A AND B
wahr	wahr	I	wahr
wahr	unwahr	I	unwahr
unwahr	wahr	I	unwahr
unwahr	unwahr	I	unwahr

Wahrheitstabelle der "AND"-Funktion

A	B	I	A OR B
wahr	wahr	I	wahr
wahr	unwahr	I	wahr
unwahr	wahr	I	wahr
unwahr	unwahr	I	unwahr

Wahrheitstabelle der "ODER" Funktion

Beispiel:

```
10 INPUT A,B,C                                    SEITE 46
20 IF A>B AND B>C THEN PRINT "A>B>C"
30 IF NOT(A>B) OR NOT(B>C) THEN PRINT "A>B>C
   IST UNWAHR"
40 END

RUN
?10
?5
?7
A>B>C IST UNWAHR
```

SEITE 47

FUNKTIONEN

In vielen mathematischen Problemloesungen werden relativ einfache mathematische Operationen angewendet. Fuer viele dieser Loesungen wurden die Werte aus Tabellen entnommen (z.B.: Logarithmus e, sinus, cosinus, Wurzel, usw.)
Da solche Operationen von Computern sehr genau und mit entsprechender Geschwindigkeit erledigt werden koennen, sind sie in den BASIC-Interpreter aufgenommen worden. Der Benutzer braucht nicht mehr in Tabellen nachschlagen, um den Wert des Sinus von 25 Grad oder den natuerlichen Logarithmus von 150 zu ermitteln. Wenn solche Werte in Ausdruecken gebraucht werden, koennen Funktionen wie:

```
SIN (25*3.14/180)
LOG (150)
```

eingesetzt werden.

Die nachfolgende Tabelle gibt die vom BASIC-Interpreter gefuehrten Funktionen an:

Funktion	Bedeutung
ABS (X)	Ergibt den Absolutwert von X
SGN (X)	Ergibt +1, -1 oder 0 entsprechend x
	Ergibt die groesste Integerzahl, welche kleiner oder gleich X ist. (INT(-0.5)=-1)
COS (X)	Ergibt den Cosinus von X (in Radian)
SIN (X)	Ergibt den Sinus von X (in Radian)
TAN (X)	Ergibt den Tangens von X (in Radian)
SQR (X)	Ergibt die Wurzel von X.
EXP (X)	Ergibt den Wert von e hoch X. e=2.71828
LOG (X)	Ergibt den natuerlichen Logarithmus von X.
RND (X)	Erzeugt eine Zufallszahl zwischen 0 und 1.
RND (N)	Erzeugt eine Zufallsinteger 0 bis N-1.

In SIN(X),COS(X),TAN(X) ist X -1000 < X < 1000.

SEITE 48

Beispiel:

```
10 REM DRUCK EINER SINUS- COSINUS TABELLE
20 PRINT "SIN(X)","COS(X)"
30 FOR I=0 TO 2 STEP 0.5
40 PRINT SIN(X),COS(X)
50 NEXT I

RUN

SIN(X)          COS(X)
0               1
0.47942         0.87758
0.84147         0.5403
0.9975          0.07073
0.9093          -0.41614
```

Kapitel 7

STRING FUNKTIONEN

- Zeichenketten (Strings)
- String Konstanten
- String Variablen
- String Verarbeitung:
- LEFT$ RIGHT$
- MID$ CHR$
- STR$ LEN
- VAL ASC
- Vergleich von Strings

ZEICHENKETTEN (Strings)

In den vorhergehenden Kapiteln wurde die Verarbeitung von numerischen Informationen besprochen. Zeichen in numerisch codierter Form (s. Anhang D, ASCII-Code Tabelle), verarbeitet der BASIC-Interpreter ebenfalls. Sie werden in Form von Zeichenketten (Strings) eingegeben.Ein String ist also eine Kette von ASCII-codierten Zeichendarstellungen.
Beispiel: "ABC"
"BASIC"

STRING KONSTANTEN

Genauso wie Zahlen als Konstanten ,kann der BASIC-Interpreter String-Konstanten verarbeiten. Strings werden im Programmtext durch Anfuehrungsstriche am Anfang und Ende gekennzeichnet.

Beispiel: 10 LET A$ = "XYZ123"

Der Inhalt der Variable A$ ist der Character String (Zeichenkette) XYZ123 .

STRING VARIABLEN

Einem String kann ein Variablenname zugewiesen werden. Jedes Zeichen des Alphabetes , gefolgt von dem Zeichen fuer Dollar ($), kann ein Variablenname sein. Es kann die numerische Variable A neben der Stringvariablen A$ existieren. Die Stringvariable kann maximal 32 Character (Zeichen) aufnehmen.

STRING VERARBEITUNG

Fuer die Manipulation von Zeichenketten steht nur ein Operator im CreatiVision-BASIC zur Verfuegung: "+" erlaubt das Aneinanderketten von Strings und Character.

Beispiel 1:
```
10 LET A$="XYZ123"
20 P$=A$+"PQR"
30 PRINT P$
40 END

RUN
XYZ123PQR
```

SVV

```
Beispiel 2:    10 A$=""
               20 REM NULLSTRING, CLEAR A$
               30 FOR I=1 TO 5
               40 A$=A$+"#"
               50 PRINT A$
               60 NEXT I

               RUN
               #
               ##
               ###
               ####
               #####
               ######
```


STRING FUNKTIONEN

Neben den mathematischen Funktionen, wie LOG, SIN, usw, haelt der BASIC-Interpreter eine Reihe von Funktionen zur Bearbeitung von Zeichenketten bereit. Damit koennen numerische Strings mathematisch bearbeitet werden, zwei Strings ineinandergekettet werden, Teilstrings ausgewertet werden, die Anzahl der Character in einem String begrenzt werden, numerische Zahlen zu Strings gewandelt werden und vieles mehr. Die Suche nach Teilstrings wird ebenfalls unterstuetzt.

LEFT$

```
Syntax: LEFT$ (Stringausdruck,numerischer Ausdruck)
z.B     LEFT$ (A$,N)
```

Diese Funktion ergibt nach Ausfuehrung einen Teilstring von A$ mit der Anzahl der Character N. Der Teilstring beginnt mit dem ersten, linken Zeichen.

```
Beispiel:    10 A$="ABC"
             20 C$=LEFT$(A$+"XYZ",3+1)
             30 PRINT C$

             RUN
             ABCX
```

RIGHT$

```
Syntax: RIGHT$ (Stringausdruck, numerischer Ausdruck)
z.B.    RIGHT$ (A$,N)
```

Diese Funktion erzeugt einen Substring von A$, beginnend vom N-ten Character in A$ bis zum letzten Character in A$.

```
Beispiel:   10 A$=RIGHT$("BASIC",3)
            20 PRINT A$

            RUN
            SIC
```

MID$

Syntax: MID$ (Stringausdruck,numer.Ausdruck,numer.Ausdruck)
z.B. MID$ (A$,M,N)

Diese Funktion erzeugt einen Teilstring aus A$, beginnend mit dem M-ten Character und einer Laenge von N.

```
Beispiel:   10 A$="ABCDEFGH"
            20 C$=MID$(A$,3,4)
            30 PRINT C$

            RUN
            CDEF
```

CHR$

Syntax: CHR$ (numerischer Ausdruck)
z.B. CHR$ (N)

Diese Funktion erzeugt einen String mit der Laenge eines Zeichens aus dem ASCII-Wert (siehe Tabelle im Anhang D).

```
Beispiel:   10 FOR I=65 TO 70
            20 PRINT CHR$(I);
            30 NEXT I

            RUN
            A          B          C
            D          E          F
```


STR$

Syntax: STR$ (numerischer Ausdruck)

Diese Funktion wandelt einen numerischen Ausdruck in einen Stringausdruck.

```
Beisp.: 10 A$=STR$(3*2+1)
        20 C$="00"+A$
        30 PRINT A$,C$

        RUN
        7 007
```

```
Beisp.: 10 A$=STR$(0.125+0.5)
        20 C$=A$+"K"
        30 PRINT C$

        RUN
        0.625K
```

LEN

Syntax: LEN (Stringausdruck) z.B. LEN (A$)

Diese Funktion ermittelt die Laenge eines Strings einschliesslich der Leerzeichen (Spaces).

```
Beispiel 1: 10 A$="ABCDEF"
            20 X=LEN(A$)
            30 PRINT X

            RUN
            6
```

```
Beispiel 2: 10 A$="XY"
            20 C$="123"
            30 X=LEN(A$+C$)+6
            40 PRINT X

            RUN
            11
```

VAL

Syntax: VAL (Stringausdruck)
z.B. VAL (A$)

Diese Funktion erzeugt aus einer numerischen Zeichenkette den numerischen Wert.

```
Beispiel:   10 A$="123+1"
            20 X=VAL(A$+"-100")
            30 PRINT X

            RUN
            24
```

ASC

Syntax: ASC (Stringausdruck)
z.B. ASC (A$)

Diese Funktion erzeugt den ASCII-Dezimalwert des ersten Zeichens im String. Beispielsweise ist der ASCII-Dezimalwert von "X" gleich 88. Wenn B$="XAB" ist, dann ist ASC(B$) gleich 88.

```
Beispiel:   10 X=ASC("AXY")
            20 PRINT X

            RUN
            65
```


STRING-VERGLEICHE

Die Verhaeltnis-Testoperatoren koennen auch auf Stringausdruecke angewandt werden. (A$="123A", "X">"AXE")
Verglichen wird der ASCII-Wert jedes einzelnen Zeichen der Ketten. Entsprechen Anhang D sind die Dezimal-Werte jedes ASCII-Zeichens bekannt: "A" ist 65, "B" ist 66, "C" ist 67 und "D" ist 68, usw.

```
Beispiel:   10 A$="AA"
            20 B$="BA"
            30 IF A$<B$ THEN PRINT 20

            RUN
            20
```

* Weil der ASCII-Wert von "A" kleiner ist als der von "B", ist A$ kleiner als B$. ("A"=65, "B"=66)

```
Beispiel:   10 A$="ABC"
            20 B$="ABD"
            30 IF B$>A$ THEN PRINT 20

            RUN
            20
```

* Die ersten beiden Character in A$ und B$ sind gleich. Das "D" in B$ wird als ASCII 68 gewertet und ist damit groesser als das "C" in A$ mit dem ASCII-Wert 67. Damit ist der Vergleich B$>A$ wahr, und die 20 wird ausgegeben.

Beispiele:

```
10 A$=CHR$(30*2+6)
20 IF A$>="B" THEN PRINT 1
30 IF A$>="9" THEN PRINT 2

RUN
1
2

10 W$="BASIC"
20 X$="BA"
30 IF LEFT$(W$,3)>X$ THEN 60
40 STOP
60 PRINT X$

RUN
BA

10 W$="APPLE"
20 X$="ORANGE"
30 IF W$<>X$ THEN 60
40 STOP
60 PRINT X$

RUN
ORANGE

10 REM PRINT ASCII CHARACTERS
20 FOR I=! TO 128
30 PRINT CHR$(I)
40 NEXT I

RUN
"
"
"
"
A
B
C
D
"
"
```

* 128 ASCII-Character werden erzeugt. Nur druckbare Zeichen erscheinen auf dem Bildschirm.

SVV

Kapitel 8

SYSTEM-ANWEISUNGEN

- STOP END
- CONT CTL/C
- RESET

SYSTEM-ANWEISUNGEN

Um den Programmablauf zu steuern und die Fehlersuche zu erleichtern, sind diverse Kommandos und Anweisungen vorgesehen. Es sind die Anweisungen STOP, END und die Kommandos CTRL/C und CONT. Der Programmablauf kann mit dem Taster RESET unterbrochen werden.

STOP

Die STOP-Anweisung beendet den Programmablauf und schaltet den Computer aus der RUN-Betriebsart in die Kommandoebene. In Verbindung mit Vergleich-gesteuerten Sprunganweisungen kann ein Programm an aktuelle Stelle beendet werden.

Syntax: Zeilennummer STOP

Nach Ausfuehrung der STOP-Anweisung wird auf dem Bildschirm ein STOP AT Zeilennummer ausgegeben.

Ein CONTINUE-Kommando nach Ausfuehrung der STOP-Anweisung fuehrt zu einer Wiederaufnahme des Programmlaufes bei der Anweisung, die dem STOP folgt. Dadurch ist die STOP-Anweisung besonders gut zur Fehlersuche bei der Entwicklung des Programmes geeignet. Nach Ausfuehrung von STOP lassen sich auf der Kommandoebene die Inhalte der Variablen untersuchen. (PRINT A <RETURN>)

Beispiel:
```
10 PRINT "TEST STOP"
20 PRINT 123
30 STOP
40 END

RUN
TEST STOP
123
STOP AT 30
```


END

Die END-Anweisung beendet die Programmausfuehrung. Sie ist die letzte Anweisung im Programm.

Syntax: Zeilennummer END

Die Zeilennummer der END-Anweisung ist normalerweise die letzte und hoechste im Programm. Die Anweisung ist aber nicht unbedingt notwendig, das Programm wird mit der letzten Anweisung automatisch beendet. Jedoch sollte im Interesse der Lesbarkeit ein Programm mit END abgeschlossen werden.

Nach Ausfuehrung von END kann ein Programm nicht mit CONT, wie bei der STOP-Anweisung, gestartet werden.

CONT

Der Programmierer kann STOP-Anweisungen frei im Programm benutzen. Jede STOP-Anweisung unterbricht den Programmlauf und gibt die Zeilennummer der STOP-Zeile aus. Der Programmierer kann dann Daten untersuchen und im Kommando-Betrieb aendern, und mit dem Kommando CONT den Programmablauf fortsetzen.

CTL/C

Es wurde besprochen, wie mit STOP ein Programm unterbrochen werden kann. Ein Programmabbruch ist auch jederzeit durch gleichzeitiges Druecken der Tasten CTL und C moeglich.
Wie bei STOP, koennen jetzt aus der Kommandoebene Untersuchungen vorgenommen werden, und dann kann der Programmlauf mit dem Kommando CONT fortgesetzt werden.

Wird ein Programm durch die CTRL/C-Kombination unterbrochen, so wird auf dem Bildschirm die folgende Meldung gegeben:

```
STOP AT
Zeilennummer, Anweisung
```

CTRL/C -Kombination:

```
.-------------.   .-------.
I CTRL I    I C I
'-------------'   '-------'
```

* Beide Tasten sind gleichzeitig zu druecken !

RESET

Der RESET-Taster befindet sich neben dem Netzschalter des Hauptgeraetes. Mit diesem Taster wird der Computer aus dem Programmlauf in eine Grundposition zurueckgesetzt. Alle durch Grafikanweisungen geaenderten Zeichen erhalten wieder ihre urspruengliche Form.
Befindet sich ein Programmlauf durch Fehler im Programm in einem Zustand, aus dem auch mit CTRL/C nicht mehr in die Kommandoebene zurueckgekehrt werden kann, so ist der RESET-Taster zu betaetigen. Das existierende BASIC-Programm wird nicht beeinflusst.

Fig. 8.1

Kapitel 9

SCHLEIFEN & VERGLEICHS-STRUKTUREN

- IF...THEN
- FOR...TO....STEP
- NEXT
- GOSUB/RETURN
- GO TO

In diesem Kapitel werden Anweisungen besprochen, welche den Programmlauf in Abhaengigkeit von den Ergebnissen arithmetischer Operationen oder logischer Vergleiche steuern oder direkte Sprueunge zu beliebigen Stellen des Programmes durchfuehren.

IF...THEN

Der Programmablauf wird entsprechend dem Ergebnis eines logischen Vergleiches gesteuert.

Die IF...THEN-Anweisung fuehrt einen "bedingten" Sprung zu einer anderen Anweisung aus.

Syntax: IF logischer Ausdruck THEN Basic-Anweisung oder Zeilennummer.

Beispiel:
```
10 S=0
20 N=0
30 N=N+1
40 S=S+N
50 IF N<>10 THEN 30
60 PRINT S
70 END

RUN
55
```


FOR..TO..STEP
NEXT

Eine der Staerken von Computersystemen ist ihre Faehigkeit, Programmsequenzen beliebig zu wiederholen. Eine Schleife bearbeitet eine Reihe von BASIC-Anweisungen so oft wie definiert. Wird z.B. das Quadrat der Zahlen von 1 bis 10 benoetigt, so braucht diese Anweisung N**2 nicht zehnmal im Programm geschrieben werden, sondern die Berechnung wird in einer Schleife programmiert.

Beispiel:
```
10 FOR N=1 TO 5
20 PRINT N,N**2
30 NEXT N

RUN
1          1
2          4
3          9
4          16
5          25
```

In diesem Beispiel wird N fuenfmal, bei jedem Schleifendurchlauf, um eins erhoeht. Sollen nur die Quadrate aller ungeraden Zahlen von 1 bis 10 errechnet werden, so muss N bei jedem Lauf um 2 erhoeht werden:

```
Beispiel:   10 FOR N=1 TO 10 STEP 2
            20 PRINT N,N**2
            30 NEXT N

            RUN
            1          1
            3          9
            5          25
            7          49
            9          81
```

Syntax: FOR Variable = Ausdruck TO Ausdruck STEP Ausdruck

STEP definiert den Wert, um den die Schleifenvariable bei jedem Durchlaufen der Schleife veraendert wird. Der Ausdruck nach "=" ist der Startwert der Schleifenvariablen, der Ausdruck nach TO ist der Endwert. Wird STEP nicht definiert, so wird die Schleifenvariable automatisch mit +1 erhoeht.

Alle mit FOR beginnenden Schleifen muessen mit der Anweisung NEXT abgeschlossen werden.

Syntax: NEXT Variable

Der Variablenname in der NEXT-Anweisung muss der gleiche wie in der FOR-Anweisung sein.

GOSUB/RETURN

Wenn identische oder fast identische Anweisungssequenzen in einem Programm haeufiger an verschiedenen Stellen benoetigt werden, so ist es unpraktisch, sie mehrmals zu schreiben. Zeit und Speicherplatz werden gespart, wenn diese Teile des Programmes als "Subroutinen" geschrieben werden. Von jeder Stelle im Programm kann diese Sequenz dann als Subroutine mit der Anweisung "GOSUB" ausgefuehrt werden. Jede Subroutine wird mit der "RETURN"-Anweisung abgeschlossen. Nach RETURN wird der Programmlauf automatisch mit der Anweisung fortgesetzt, die auf die GOSUB-Anweisung folgt.

```
Syntax: GOSUB erste Zeilennummer der Subroutine
            .
            .
        RETURN
```


GOTO

Die GOTO-Anweisung uebergibt den Programmlauf direkt und ohne Bedingung an die nach GOTO definierte Anweisungszeile. Normalerweise ist die mit GOTO angesprochene Anweisung nicht die auf die GOTO-Anweisung folgende.

Syntax: GOTO Zeilennummer

SVV

Die Zeilennummer, an welcher die Programmausfuehrung fortgesetzt wird, kann groesser oder kleiner als die Zeilennummer der GOTO-Anweisung sein. Eine Verzweigung vorwaerts oder rueckwaerts im Programmlauf wird dadurch moeglich.

Beispiel:
```
10 N=1
20 S=0
30 S=S+N
40 N=N+1
50 PRINT N,S
60 GOTO 30

RUN
1       1
2       2
3       6
4       10
.       .
.       .
```

* Dieses Programm wird ausgefuehrt, bis es mit CTRL/C abgebrochen wird.

Beispiel:
```
10 FOR I=1 TO 5
20 GOSUB 60
30 PRINT  I,S
40 NEXT I
50 END

60 S=0
70 FOR J=1 TO I
80 S=S+J
90 NEXT J
100 RETURN

RUN
1       1
2       3
3       6
4       10
5       15
```

Beispiel:
```
10 A=30
15 PRINT A
20 GOTO A
25 PRINT A*A
30 END

RUN
30
```

* Zeile 25 wird nicht bearbeitet !

Kapitel 10

FELD-VARIABLEN (ARRAY)

- DIM

Feld-Variablen (Arrays)

Ein Feld ist eine Gruppe von Variablen oder Elementen, alle mit dem gleichen Namen, die sich nur durch eine in Klammern geschriebene Zahl nach dem Variablennamen unterscheiden (dem Index).
Die Variable A(I) ist das I-te Element im Feld A. I muss eine Integerzahl sein.

Syntax: Variable (Integer-Ausdruck)

Die Anweisung "DIM" (DIMensions) stellt den Speicherplatz fuer ein Feld bereit. Das Feld kann eindimensional oder zweidimensional sein.

Die DIM-Anweisung im Programm muss gegeben werden, bevor zum erstenmal eine Variable dieses Feldes angesprochen wird. DIM A(6) dimensioniert ein eindimensionales Feld mit sechs Elementen, waehrend DIM B(6,6) ein zweidimensionales Feld fuer die indizierte Variable "B" mit 6 * 6 = 36 Elementen dimensioniert.

Soll beispielsweise ein zweidimensionales Feld "A" mit den Dimensionen "3" und "5" gebildet werden, so ist die DIM-Anweisung:

```
DIM A (3,5)
```

Es werden dann 3 * 5 = 15 Speicherplaetze fuer die indizierte Variable A bereitgestellt:

```
A(1,1)  A(1,2)  A(1,3) ..... A(1,5)
A(2,1)  A(2,2)  A(2,3) ..... A(2,5)
A(3,1)  A(3,2)  A(3,3) ..... A(3,5)
```

Ein zweidimensionales Feld benoetigt also zwei Indizes, um jedes Element anzusprechen.(Wie Spalten- und Zeilennummern einer Matrix.)

Beispiel:

```
10 REM DIE GASRECHNUNG EINES HALBEN JAHRES
20 DIM A(6)
30 REM FRAGE NACH RECHNUNGSBETRAG
40 FOR I=1 TO 6
50 PRINT "RECHNUNGSBETRAG=";
60 INPUT A(I)
70 S=S+A(I)
80 NEXT I
90 PRINT "GESAMTBETRAG=";S
100 END
```

Beispiel:

```
10 DIM A(3,3)
20 FOR I=1 TO 3
30 FOR J=1 TO 3
40 A(I,J)=I+J
50 PRINT A(I,J);
60 NEXT J
70 PRINT
80 NEXT I
90 END

RUN
2   3   4
3   4   5
4   5   6
```

Kapitel 11

Kommandos fuer die Ein-/Ausgabe

- INPUT
- PRINT
- TAB
- READ/DATA
- RESTORE

INPUT

Mit der Anweisung "INPUT" werden Daten von der Tastatur uebernommen. Ein "?" auf dem Bildschirm fordert zur Eingabe auf. Jede Eingabe wird mit <RETURN> abgeschlossen.

Syntax: INPUT Variable,Variable, ...

Beispiel:
```
10 INPUT A,B
20 PRINT A,B,A+B
30 END

RUN
?3              Jede moegliche Zahl!
?4
3       4       7          Computerausgabe!
```


PRINT

Die PRINT-Anweisung gibt die Werte von Variablen und Ausdruecken auf den Bildschirm aus.

Syntax: PRINT Variable,Variable, ...
oder Ausdruck,Ausdruck, ...

Beispiel:
```
10 PRINT "BASIC PROGRAMM"
20 LET A=3
30 PRINT  A,A+A,A*A
40 END

RUN
BASIC PROGRAMM
3       6       9
```

Werden in PRINT-Anweisungen Variablen und Ausdruecke durch ein "," getrennt, so erfolgt die Ausgabe auf den Bildschirm an drei vortabulierten Positionen auf dem Bildschirm. Das Erstellen von Tabellen wird damit unterstuetzt (siehe voriges Beispiel).

Werden PRINT-Anweisungen mit einem ";" gegeben / so wird damit das Fortschalten auf die naechste Zeile des Bildschirmes unterdrueckt. (Es wird kein Wagenruecklauf mit Zeilenvorlauf "CARRIAGE RETURN mit LINE FEED " erzeugt.)

Syntax: PRINT Variable;Ausdruck;....;....

```
Beispiel:   10 FOR A=1 TO 10
            20 PRINT A;
            30 NEXT A
            40 END

            RUN
            1  2  3  4  5  6  7  8  9  10
```


LPRINT

Wie PRINT eine Ausgabe auf den Bildschirm veranlasst, steuert LLPRINT die Ausgabe an den Drucker.

```
Syntax: LPRINT Variable,Ausdruck,...

Beispiel:   10 LPRINT "DIESES PROGRAMM DRUCKT"
            20 LPRINT "DRUCKT ALLE ZEICHEN"
            30 FOR N=32 TO 96
            40 LPRINT CHR$(N);
            50 NEXT N
            60 END

            RUN
            (Alle druckbaren Zeichen werden ausgedruckt.)
```

* Drucker und I/O-Interface muessen betriebsbereit sein, bevor Programme mit LPRINT-Anweisungen bearbeitet werden.

TAB

Die TAB-Funktion wird in PRINT-Anweisungen verwendet und tabuliert die naechste Ausgabe auf dem Bildschirm. Die vorgesehende Ausgabestelle wird durch die in Klammern angegebene Spaltennummer des Bildschirmes festgelegt.

```
Syntax: TAB (Variable)
```

```
TAB(N)  stellt den Cursor auf die N-te Spalte des Bild-
schirmes. N kann jede Integerzahl von 1 bis 64 sein.
Beispiel: PRINT TAB(5);"A";TAB(10);"A"
          A                 A
```

READ

Die READ und DATA-Anweisungen uebergeben in Zusammenarbeit eine Liste von Informationen an das Programm. Die READ-Anweisung initialisiert eine Variable mit einem Wert aus der DATA-Liste.

```
Syntax: READ Variable,Variable,..

Beispiel:    10 DATA 1,3,5,7
             20 READ X,Y
             30 READ A
             40 READ B
             50 PRINT X+Y,A+B
             60 END

             RUN
             4     12
```

DATA

DATA haelt Informationen fuer die READ-Anweisung bereit. Die READ-Anweisung uebernimmt diese Daten in der Reihenfolge der Zeilennummern. Die DATA-Zeilen muessen am Anfang des Programmes liegen, mindestens aber vor der READ-Anweisung.

Syntax: DATA Konstante,Konstante,...

Beispiel:
```
10 DATA 8,1,6
20 DATA 3,5,7,4,9,2
30 FOR I=1 TO 3
40 READ A,B,C
50 PRINT A,B,C
60 NEXT I
70 END

RUN
8     1     6
3     5     7
4     9     2
```

Beispiel:
```
10 REM ERRECHNE MAXIMUM
20 REM UND ABWEICHUNG
30 DATA 0.125,3,0.6,7
40 DATA 23.9.3,25.2,8
50 M=-99
60 S=0
70 FOR I=1 TO 8
80 READ N
90 S=S+N
100 IF N>M THEN M=N
110 NEXT I
120 A=S/8
130 PRINT M,A

RUN
25.2     9.5281
```


RESTORE

Sollen im Ablauf des Programmes die Daten einer DATA-Liste noch einmal benutzt werden, so kann mit der RESTORE-Anweisung der Zeiger auf das naechste zu lesende Element der Liste wieder auf den Beginn der Liste gestellt werden.

Syntax: RESTORE

Beispiel:
```
10 DATA 1,3,5,7,9
20 READ A,B,C
30 RESTORE
40 READ X
45 PRINT A,C
50 PRINT X
60 END

RUN
1       5
1
```

Kapitel 12

Programmspeicherung auf Tonband

- CSAVE
- CLOAD
- CRUN

KASSETTENSPEICHER - MODUL

Das CreatiVision-Computersystem kann durch ein Kassetten-speicher-Modul erweitert werden. CreatiVision-BASIC erlaubt das Speichern und Lesen von Programmen auf Tonband. Ist ein Programm auf Band gespeichert, so kann es jederzeit wieder in den Computer geladen und benutzt werden.

Das Kassettenspeicher-Modul wird an den Computer ueber ein Kabel wie in Bild 12.1 angeschlossen.

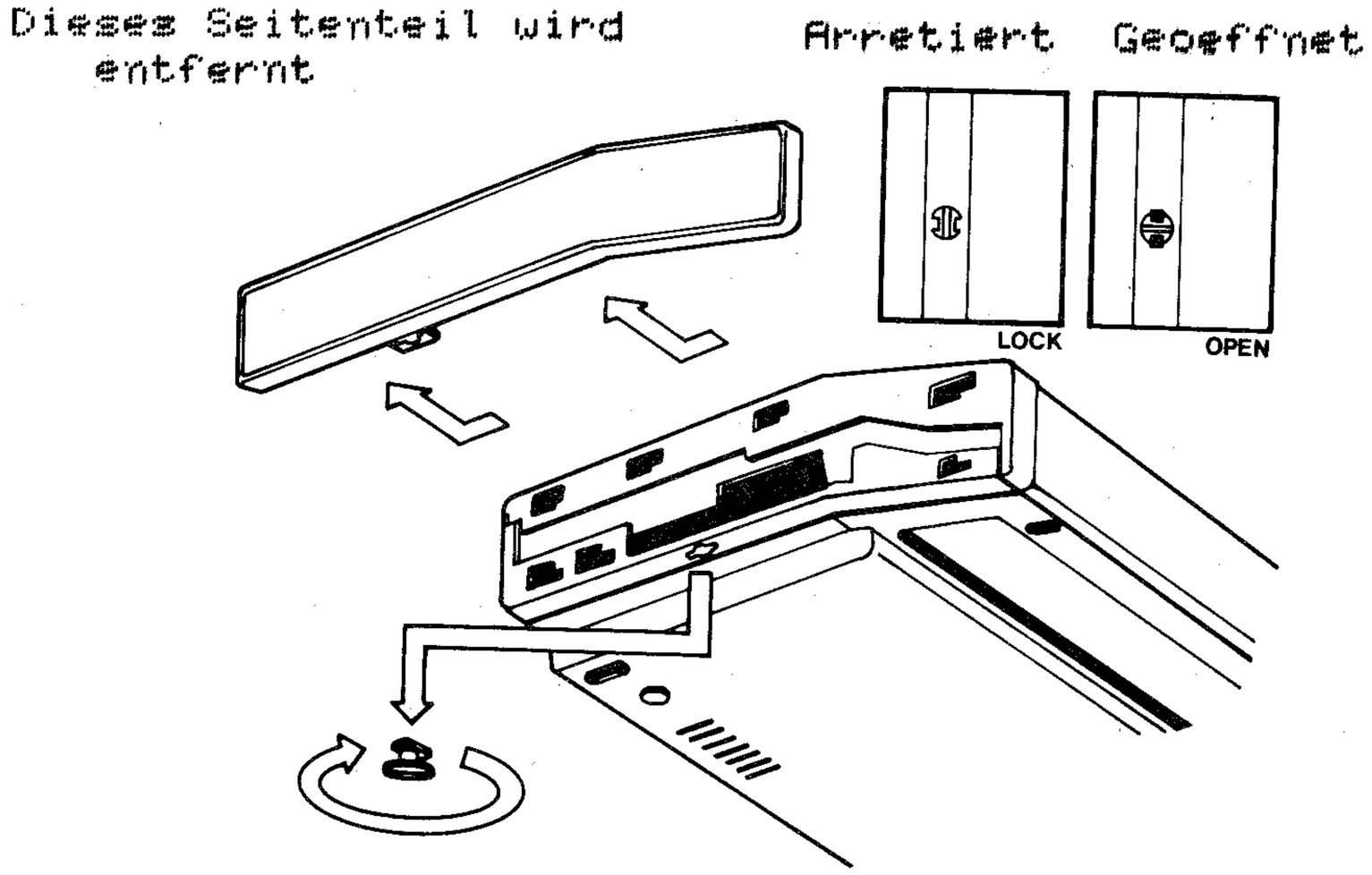


Fig. 12.1

CSAVE

Syntax: CSAVE

Ablauf:

1) Verbinden Sie das Kassettenspeicher-Modul ueber das Kabel mit dem Computersystem.
2) Verwenden Sie eine qualitativ gute Kassette.
3) Druecken Sie die Tasten "PLAY" und "RECORD" am Recorder gleichzeitig.
4) Notieren Sie die Anzeige des Bandzaehlwerkes am Kassettenspeicher-Modul. Beim spaeteren Laden des Programmes kann das Band damit in die richtige Position gebracht werden.
5) Geben Sie das CSAVE-Kommando ueber die Tastatur des Computers. Das Band laeuft automatisch an, und das Programm wird auf dem Bildschirm gelistet.
6) Nach Beendigung der Aufzeichnung erscheint die Aufforderung zur weiteren Eingabe, das Zeichen ">" auf dem Bildschirm. Die STOP-Taste am Recorder ist dann zu druecken.

CLOAD

Syntax: CLOAD

Ablauf

1) Verbinden Sie das Kassettenspeicher-Modul ueber das Kabel mit dem Computersystem.
2) Setzen Sie die Kassette ein und spulen Sie sie auf die korrekte Position.
3) Druecken Sie die Taste "PLAY" am Kassettenrecorder
4) Geben Sie das CLOAD-Kommando ueber die Tastatur ein. Der Kassettenrecorder startet dann automatisch. Wird ein Programm gefunden, so wird es geladen und dabei auf dem Bildschirm gelistet.
5) Nach Beendigung des Ladevorganges wird der Promt (">") auf den Bildschirm ausgegeben, und der Kassettenrecorder sollte durch die STOP-Taste abgeschaltet werden.

* Wird ein Programm aus der
CreatiVision - Programm - Bibliothek
geladen, so sind Erklaerungen und Musik im Tonkanal des TV-Gerates hoerbar.

CRUN

Syntax: CRUN

Das Kommando CRUN entspricht den Kommandos CLOAD+RUN. Der Computer laedt das Programm und startet es dann automatisch. CRUN kann auch die letzte Anweisung in einem Programm sein. Bleibt die PLAY-Taste am Kassettenspeicher-Modul gedrueckt, wird das naechste Programm unter Programmkontrolle geladen.
Der Bedienungsablauf ist sonst der gleiche wie bei CLOAD.

* Anmerkung: Soll bei den Speicher- und Ladeoperationen das Listen des Programmes unterbunden werden, so sind die Kommandos in der folgenden Form zu geben:
CSAVE (N)
CLOAD (N)
CRUN (N)

SVV

Kapitel 13

Grafik-Funktionen und Tonerzeugung

- CLS
- COLOR
- CHAR
- PLOT
- SOUND
- JOY

Grafik-Kommandos

Die Grafik-Kommandos arbeiten auf den in 32 Spalten und 24 Zeilen organisierten Bildschirm. Da viele TV-Geraete den Bildschirm links und rechts ueberschreiben, werden im CreatiVision-BASIC nur 28 Print-Positionen benutzt. Um eine sichere Grafikausgabe zu erreichen, sollte im Grafik-Betrieb auch auf die Ausgabe in den beiden rechten und in den beiden linken Spalten verzichtet werden.

Es sind 4 Grafik-Kommandos vorhanden:

1) CLS - Bildschirm loeschen
2) COLOR - Farbe definieren
3) CHAR - Character definieren
4) PLOT - Character auf definierte Bildschirmstelle ausgeben

CLS

Mit dem CLS-Kommando (clear screen) wird der Bildschirm geloescht. Wird im Programm eine CLS-Anweisung ausgefuehrt, so werden alle Bildschirmspeicher mit dem ASCII-Code fuer SPACE = 32 gefuellt.

Beispiel:

```
10 CLS
20 PRINT "CLEAR SCREEN"

RUN
```

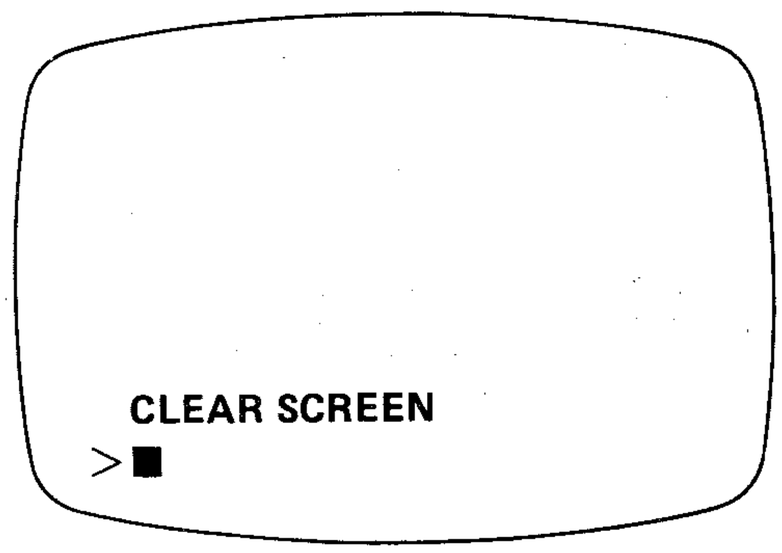


Fig. 13.1

COLOR

Die COLOR-Anweisung gibt alle Moeglichkeiten, die Bildschirmfarbe zu beeinflussen.

Syntax: COLOR Charactersatz-Nummer, Vordergrund-Farbcode, Hintergrund-Farbcode

Beispiel: COLOR 2,10,14

Jedes Zeichen auf dem Bildschirm hat zwei Farben: die Farbe des Zeichens selber (Vordergrundfarbe) und die es umgebene Farbe (Hintergrundfarbe). 16 Farben sind im Creati-Vision-BASIC definierbar.

Color-Code	Farbe
1	Transparent
2	Schwarz
3	Gruen
4	Hellgruen
5	Dunkelblau
6	Hellblau
7	Dunkelrot
8	Cyan
9	Rot
10	Hellrot
11	Dunkelgelb
12	Hellgelb
13	Dunkelgruen
14	Magenta (Purpur)
15	Grau
16	Weiss

Die Farbumschaltung mit der COLOR-Anweisung beeinfluest jeweils eine Gruppe von 8 Zeichen. Im ersten Argument der COLOR-Anweisung ist die gewuenschte Gruppe anzugeben. Die Liste der ASCII-Zeichen-Codes finden Sie im Anhang D. Die Zeichen-Gruppennummer ist der folgenden Tabelle zu entnehmen:

Character-Gruppe	ASCII-Code	Character-Gruppe	ASCII-Code
1	0-7	17	128-135
2	8-15	18	136-143
3	16-23	19	144-151
4	24-31	20	152-159
5	32-39	21	160-167
6	40-47	22	168-175
7	48-55	23	176-183
8	56-63	24	184-191
9	64-71	25	192-199
10	72-79	26	200-207
11	80-87	27	208-215
12	88-95	28	216-223
13	96-103	29	224-231
14	104-111	30	232-239
15	112-119	31	240-247
16	120-127	32	248-255

Beachten Sie, dassleere Bildschirmspeicher mit dem Code fuer SPACE =32 gefuellt sind. Wenn Sie den Characterssatz 5 in einer COLOR-Anweisung benutzen, werden alle SPACE-Character des Bildschirms auf die definierte Hintergrundfarbe geschaltet. ASCII 32 ist in Satz 5 enthalten.
Das folgende Beispiel demonstriert dies:

```
Beispiel: 10 REM FARB DEMONSTRATION
          20 CLS
          30 COLOR 5,6,6
          40 GOTO 40

          RUN
* Die Bildschirmfarbe wechselt von Hellgruen zu Hellblau.
* Programmstop mit CTL/C !
```

CHAR

Mit der CHAR-Anweisung koennen neue Grafik-Character erzeugt werden. Im Programm kann so ein ganzer Satz neuer Zeichen gebildet werden.

Syntax: CHAR Charactercode, Musterdefinition

Beispiel: CHAR 32,18 18 FF 3C 7E FF 14 36

Mit Charactercode wird der Code des Zeichens definiert, dass durch das neu erzeugte ersetzt werden soll. Der Charactercode kann ein numerischer Ausdruck zwischen 0 und 255 sein.
Der Code fuer die Musterdefinition besteht aus 16 Zeichen, mit denen der neue Character beschrieben wird. Diese 16 Zeichen stellen die 64 Punkte dar, aus denen in einem 8 X 8 - Gitter der neue Grafikcharacter gebildet wird.

Jede Zeile ist in zwei Blocks zu je vier Punkte aufgeteilt.

```
          LINKE   RECHTE
        BLOECKE   BLOECKE

ZEILE1  OOOOOOOO
ZEILE2  OOOOOOOO
ZEILE3  OOOOOOOO
ZEILE4  OOOOOOOO
ZEILE5  OOOOOOOO
ZEILE6  OOOOOOOO
ZEILE7  OOOOOOOO
ZEILE8  OOOOOOOO
```

Bild 13.2

Jede Zeile ist in zwei Bloecke mit je vier Punkten geteilt:

```
Jede Zeile:   OOOO OOOO
     Linker Block ↑ Rechter Block
```

Bild 13.3

Jeder Character des Musterdefinitions-Strings beschreibt die Punktanordnung der Zeile eines Blocks. In diesem String werden die Reihen von links nach rechts und von oben nach unten beschrieben. Die ersten zwei Character dieses Strings beschreiben also Zeile 1 des 8 X 8 - Schemas, die naechsten zwei beschreiben Zeile 2, usw.

Character werden durch "einschalten" oder "ausschalten" von Punkten in der 8 X 8 -Matrix gebildet. In der folgenden Tabelle wird ein "ausgeschalteter" Punkt durch "0" definiert, ein "eingeschalteter" Punkt durch "1".
Wird ein Musterdefinition-String mit weniger als 16 Character in das Programm aufgenommen, so werden die fehlenden Character als "0" angenommen. Werden mehr als 16 Character definiert, werden die restlichen ignoriert.

BLOCK	0=Aus 1=Ein	CODE
□□□□	0000	0
□□□■	0001	1
□□■□	0010	2
□□■■	0011	3
□■□□	0100	4
□■□■	0101	5
□■■□	0110	6
□■■■	0111	7
■□□□	1000	8
■□□■	1001	9
■□■□	1010	A
■□■■	1011	B
■■□□	1100	C
■■□■	1101	D
■■■□	1110	E
■■■■	1111	F

Bild 13.4

Beispiel: Das in Bild 13.5 abgebildete Punktmuster wird so beschrieben:

CHAR 32,18 18 FF 3C 7E FF 14 36

ZEILE	LINKER BLOCK	RECHTER BLOCK	BLOCK-CODE
1	□□□■■□□□		18
2	□□□■■□□□		18
3	■■■■■■■■		FF
4	□□■■■■□□		3C
5	□■■■■■■□		7E
6	■■■■■■■■		FF
7	□□□■□■□□		14
8	□□■■□■■□		36

Zeichenmuster fuer den Zeichencode -32.

Bild 13.5

Beachten Sie, das die CHAR-Anweisung nur einen Character definiert. Die Ausgabe auf den Bildschirm wird mit der Anweisung PLOT vorgenommen. Wird eine CHAR-Anweisung im Programm durchgefuehrt, aendern sich alle auf den Bildschirm schon ausgegebenen Character mit dem selben Code.

Beispiel:
```
10 CLS
20 COLOR 5,4,6
30 CHAR 32,18 18 FF 3C 7E FF 14 36
30 GOTO 30

RUN
```

Der Bildschirm ist jetzt mit dem Character entsprechend Bild 13.5 gefuellt, weil sein Code 32 ist. 32 ist aber auch der Code fuer SPACE, mit dem der Bildschirm nach Ausfuehrung der Anweisung CLS gefuellt war.
Beachten Sie, dassdie Character 32 bis 95 durch den Computer definiert sind. Eine Neudefination ist moeglich, wie das Beispiel fuer den Code 32 zeigt.

SVV

Alle definitierten Farben und Character werden mit Betaetigung des RESET-Tasters in den Zustand nach dem Einschalten rueckgesetzt.

PLOT

Die PLOT-Anweisung ermoeglicht es, ein Zeichen an jede beliebige Stelle des Bildschirmes auszugeben.

Syntax: PLOT Spaltennummer,Zeilennummer,Charactercode

z.B. PLOT 15,10,65

Ein "A" wird in die 15.Stelle der Zeile 10 ausgegeben.

Ist die Spaltennummer =1, wird die Ausgabe ganz links auf den Bildschirm erfolgen. Ist die Zeilennummer =1, wird die Ausgabe des Zeichen am obersten Rand des Bildschirmes erfolgen.
Da die meisten TV-Empfaenger den Bildschirm links und rechts ueberschreiben, sollten fuer Grafiken nur die Spalten 3 bis 30 benutzt werden.

Beispiel:

```
10 CLS
20 COLOR 5,4,6
30 CHAR 33,18 18 FF 3C 7E FF 14 36
40 FOR X=1 TO 29
50 PLOT X,10,33
60 NEXT X
70 GOTO 70

RUN
```

****

Mit dem definierten Muster wird eine Zeile auf den Bildschirm ausgegeben.

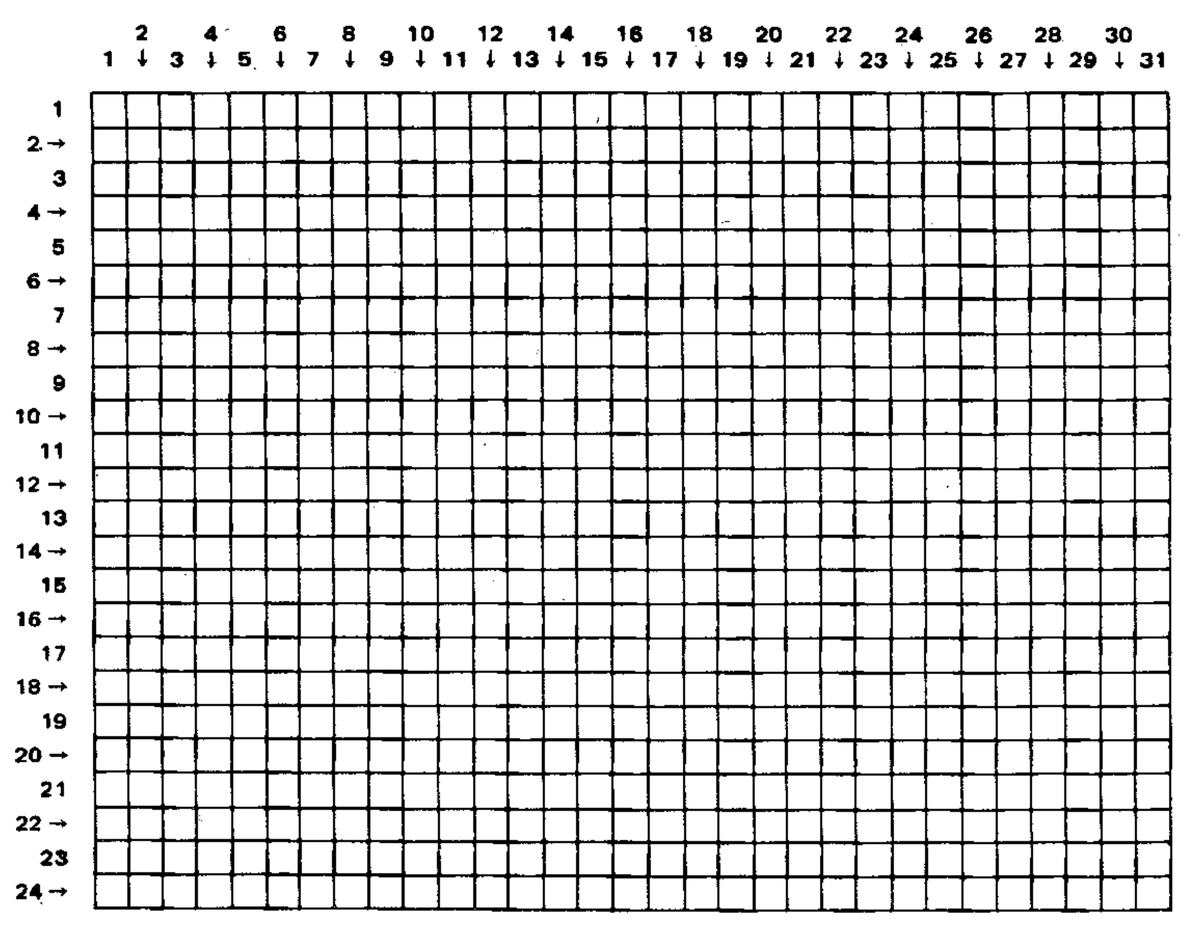
Beispiel:

```
05 CLS
10 FOR A=1 TO 16
20 COLOR A+16,A,A
30 NEXT A
40 FOR X=1 TO 32
50 FOR Y=1 TO 20
60 PLOT X,Y,X*4+128
70 NEXT Y
80 NEXT X
90 GOTO 90

RUN
```

Ein Farbbalken wird erzeugt.

Bild 13.7
Bildschirm-Organisation



SOUND

Die SOUND-Anweisung veranlasst den Computer, Toene unterschiedlicher Hoehe in drei Kanaelen gleichzeitig auszugeben.

Syntax: SOUND Frequenz;Dauer,Frequenz;Dauer,Frequenz;Dauer

z.B. SOUND 4;5,6;7,7;7 (Klang einer Glocke)

Die Codes fuer die Frequenz und die Dauer eines Tones sind numerische Ausdruecke. Ist das Ergebnis eines Ausdruckes eine nicht integer Zahl, so wird sie auf den naechst hoheren Wert aufgerundet.

Code	Ton
1	Rest
2	C
3	$C^{\#}$, D^{b}
4	D
5	$D^{\#}$, E^{b}
6	E
7	F
8	$F^{\#}$, G^{b}
9	G
10	$G^{\#}$, A^{b}
11	A (below middle C)
12	$A^{\#}$, B^{b}
13	B
14	C (middle C)
15	$C^{\#}$, D^{b}
16	D

Code	Ton
17	$D^{\#}$, E^{b}
18	E
19	F
20	$F^{\#}$, G^{b}
21	G
22	$G^{\#}$, A^{b}
23	A (above middle C)
24	$A^{\#}$, B^{b}
25	B
26	C (high C)
27	$C^{\#}$, D^{b}
28	D
29	$D^{\#}$, E^{b}
30	E
31	F
32	Rest

Code	Takt	
Ø	¼	
1	½	
2	¾	
3	1	
4	1½	
5	2	
6	3	
7	4	

Das 1. Datenpaar in einer SOUND-Anweisung gilt dem 1.Tonkanal, das 2.Paar dem 2.Kanal, und das 3.Paar dem 3.Tonkanal.

Beispiel:

```
10 REM LIED
20 SOUND 26;3,21;3,14;3
30 SOUND 30;3,25;3,16;3
40 SOUND 28;3,23;3,16;3
50 SOUND 21;5,16;5,9;5
60 SOUND 26;3,21;3,14;3
70 SOUND 28;3,23;3,16;3
80 SOUND 30;3,25;3,16;3
90 SOUND 26;7,21;7,14;7
100 GOTO 20

RUN
```

(der Computer spielt eine Melodie nach Bild 13.8)

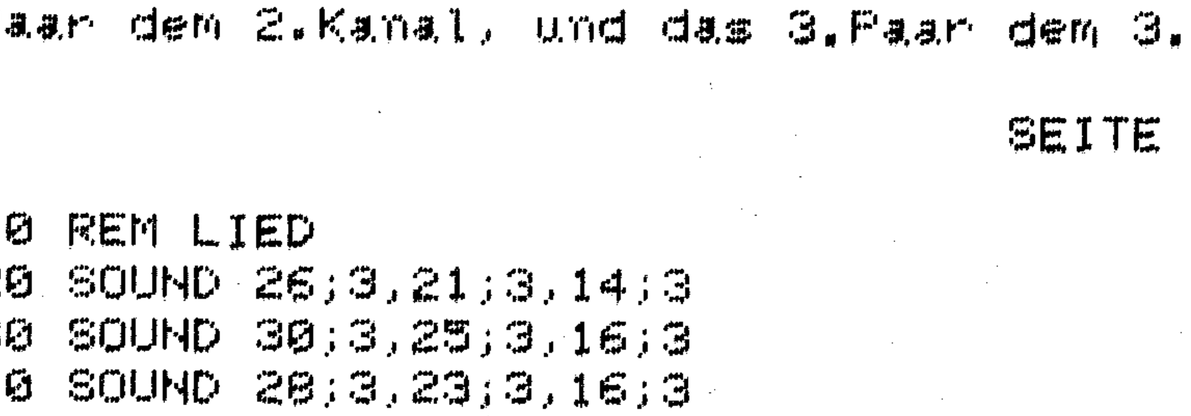


Fig. 13.8

JOY

Die JOY-Funktion (Joy-stick = Steuerknueppel) erlaubt dem Programmierer, Informationen ueber die Position der Steuerknueppel an das Programm zu uebergeben. Damit wird der Entwurf von Spielen durch den Benutzer unterstuetzt. Die Tastatur des CreatiVision-Computers ist in zwei Teile geteilt. Jeder Teil hat einen Steuerknueppel und zwei "Feuer"-Taster. Informationen ueber den Stand dieser Bedienungselemente werden mit der JOY-Funktion ausgewertet.

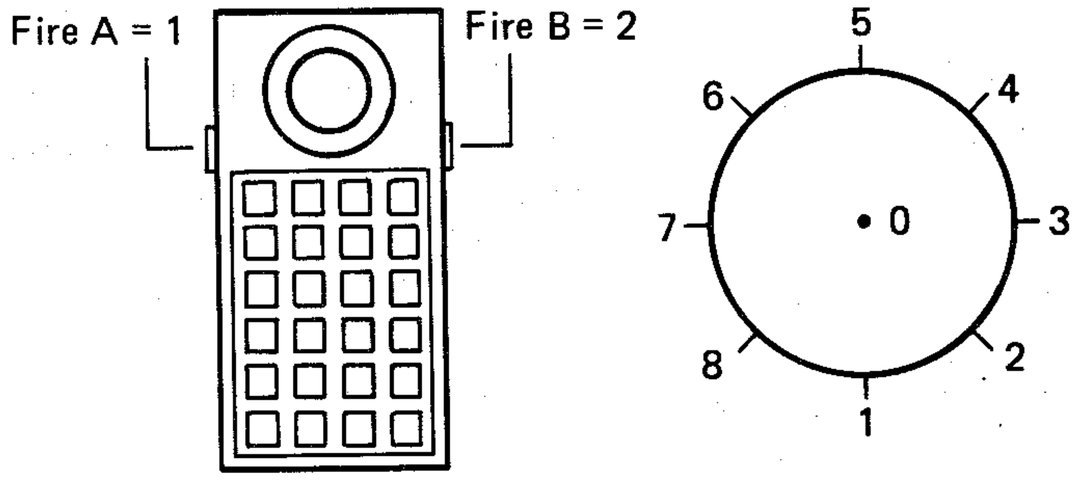


Bild 13.8 Die den Joy-stick-Positionen und den "FEUER"-Taste zugeordneten Werte.

Syntax: JOY (N)

N=1 Joy-stick links
N=2 Joy-stick rechts
N=3 Fire button links
N=4 Fire button rechts

Das Ergebnis der Funktion ist die Position der Joy-sticks oder das Fire-Kommando der Fire buttons.

Beispiel:
```
10 PRINT JOY (1)
RUN
3
```

Dieses Beispiel prueft den linken Joy-stick. Entsprechend Bild 13.8 ist er nach rechts gedrueckt.

Beispiel:
```
05 CLS
10 LET A=JOY(1)
20 IF A=3 THEN GOTO 100
30 IF A=7 THEN GOTO 200
40 GOTO 10
100 PLOT 4,10,32
110 PLOT 28,10,65
120 GOTO 10
200 PLOT 28,10,32
210 PLOT 4,10,65
220 GOTO 10

RUN
```

Entsprechend der Position des Joy-sticks wird der Character "A" am linken oder rechten Rand des Bildschirmes geschrieben.

Kapitel 14

SPEICHER-ZUGRIFF

- PEEK
- POKE

PEEK

Die PEEK-Funktion liest einen Systemspeicher aus, dessen Adresse als Argument notiert ist. Das Ergebnis ist eine Integer im Bereich 0 bis 255.

Syntax: PEEK (Adresse)

```
Beispiel:   10 PRINT "ADRESSE","INHALT"
            20 FOR K=0 TO 20
            30 PRINT K, PEEK(K)
            40 NEXT K
            50 END
```

Das Beispiel schreibt den Inhalt der ersten 20 Systemspeicherzellen auf den Bildschirm.

POKE

Waehrend die PEEK-Funktion eine Speicherzelle liesßt, kann mit der POKE-Anweisung eine Speicherzelle beschrieben werden.

Syntax: POKE Adresse,Integer

Ein Wert von 0 bis 255 kann unter der angegebenen Adresse abgespeichert werden.

```
Beispiel:     POKE 63300,48
```

Damit wird dez. 48 in die Speicherzelle 63300 geschrieben.
Mit
```
              PRINT PEEK(63300)
              48
```
kann Die Funktion der POKE-Anweisung gepruefft werden.
Versuchen Sie es mit anderen Werten !

Kapitel 15

CREATIVISION - SYSTEM

Erweiterung

Es sind diverse Erweiterungsmoeglichkeiten fuer Ihr CreatiVision-System vorgesehen. Sollten Sie mehr Informationen als die in diesem Kapitel gegebenen benoetigen, wenden Sie sich bitte an Ihren Haendler oder an die in diesem Handbuch angegebene Adresse.

1) Das Kassettenspeicher-Modul
   Mit den Anweisungen CSAVE und CLOAD koennen Programme auf Band geschrieben und vom Band gelesen werden. Die Recorderfunktionen werden vom Programm gesteuert. Die Datenuebertragung wird mit 600 Baud vorgenommen.

2) Das I/O-Interface fuer parallele und serielle Datenuebertragung
   Viele Moeglichkeiten fuer die Kommunikation mit der Aussenwelt stellt das Interfacemodul zur Verfuegung. Ein Parallelport (Centronics Bus) ist fuer Drucker wie: EPSON MX80, SEIKOSHA GP-80, CENTRONICS 779, usw. vorgesehen. Die beiden anderen Ports sind zum Anschluss eines Floppy-Disk-Laufwerkes und eines Telephon-Modems vorgesehen.

3) Das Speicher-Erweiterungs-Modul
   16 und 32 KByte-Speichererweiterungen erweitern Ihren Systemspeicher auf max. 64 KByte. Es kann mehr als ein Modul angeschlossen werden.

4) Die Standard-ASCII-Tastatur
   Die gummibelegten Tasten dieser Standard-Tastatur erleichtern das Arbeiten mit Ihrem Computer-System.

ANHANG A

Liste der BASIC-Anweisungen

BASIC QUICK REFERENZ

-Zahlen werden mit einer Genauigkeit von 6 Stellen verar-
 beitet.
-Der numerische Bereich: 10 E-38 =< N =< 10 E 38
-Alle Anweisungen koennen vom Programm oder als Kommandos
 ausgefuehrt werden.

FUNKTIONEN:

1) Arithmetische Operatoren
   +, -, *, /, **

2) Vergleichs-Operatoren
   >, <, =, <=, >=,<>

3) Arithmetische Funktionen

SQR	- Wurzel
INT	- Integerteil einer Zahl
RND	- Zufallszahl
ABS	- Absolutwert
SGN	- Signaturfunktion
COS	- Cosinus
SIN	- Sinus
EXP	- e
TAN	- Tangens
LOG	- Natuerlicher Logarithmus

4) String Funktionen

LEN	- Laenge des Strings
STR$	- String des numerischen Arguments
VAL	- Numerischer Wert des Strings
ASC	- ASCII-Code
CHR$	- Character
LEFT$	- Linke Character
RIGHT$	- Rechte Character
MID$	- Mittlere Character
+	- Verkettungsoperator

5) Logische Operatoren

AND	Verhaeltnis- und logische Ausdruecke setzen
OR	wahr=1 und unwahr=0
NOT	

6) Grafik- und Tonfunktionen

CLS	- Bildschirm loeschen
PLOT	- Character auf Bildschirm plotten
COLOR	- Farbe setzen
SOUND	- Ton versch. Frequenz und Dauer erzeugen
CHAR	- Character definieren
JOY	- Joy-stick pruefen
MODE	- Umschaltung alphanumerisch / Grafik

7) Programm-Anweisungen

DIM - Dimensionieren
STOP
END
GOTO
GOSUB
RETURN
FOR...TO...STEP
NEXT
REM
IF...THEN
INPUT
PRINT
TAB
LET
DATA
READ
RESTORE

8) Kommandos

LIST
RUN
NEW
CONT
CLOAD - Lese Programm vom Band
CSAVE - Schreibe Programm auf Band
CRUN - Lese Programm vom Band und starte es
CTRL/C = Stop Programm

9) Andere Anweisungen

PEEK - Lese System-Speicherzelle
POKE - Schreibe in System-Speicherzelle
LPRINT - Ausgabe zum Drucker
LLIST - Programmlisting zum Drucker

ANHANG B

Fehler-Meldungen

Tritt waehrend des Programmlaufes ein Fehler auf, so wird er vom CreatiVision-BASIC mit einem Fehlercode angezeigt:

CODE	FEHLER
00	Fehler waehrend der CLOAD-Operation
01	NEXT ohne vorhergehende FOR-Anweisung
02	RETURN ohne GOSUB-Anweisung
03	Fehlende Zeilennummer
04	Fehlender Operand
05	SYNTAX-ERROR, falsche Schreibweise
06	Zu hoher Zahlenwert
07	Falsche Verschachtelung von FOR...NEXT-Schleifen
08	Falsche Verschachtelung von GOSUB...RETURN
09	System-Fehler
10	Zahlenspeicher (STACK) Ueberbelegt
11	Nicht zulaessiger Operand
12	IF falsch definiert
13	Falsche Klammer-Anordnung
14	Zuviele Klammerebenen
15	String ist nicht definiert
16	Stringverarbeitung fehlerhaft
17	Verbotene Division durch Null
18	Mehr READ-Anweisungen, als Daten definiert sind
19	Speicherbereich fuer Daten ueberbelegt
20	DIM-Anweisung nicht ordnungsgemaess
21	Zeichenkette zu lang

ANHANG D

ASCII-CODE

ASCII CODE	ZEICHEN (Character)		ASCII CODE	ZEICHEN (Character)	
32		Leerzeichen	64	@	AT-Zeichen
33	!	Ausrufe-Zeichen	65	A	
34	"	Anfuehrungs-Zeichen	66	B	
35	#	Nummern-Zeichen	67	C	
36	$	Dollar	68	D	
37	%	Prozent	69	E	
38	&	Und-Zeichen	70	F	
39	'	Apostroph	71	G	
40	(	Klammer oeffnen	72	H	
41	)	Klammer schliessen	73	I	
42	*	Asterisk	74	J	
43	+	Plus	75	K	
44	,	Komma	76	L	
45	-	Minus	77	M	
46	.	Punkt	78	N	
47	/	Division	79	O	
48	0		80	P	
49	1		81	Q	
50	2		82	R	
51	3		83	S	
52	4		84	T	
53	5		85	U	
54	6		86	V	
55	7		87	W	
56	8		88	X	
57	9		89	Y	
58	:	Doppelpunkt	90	Z	
59	;	Semicolon			
60	<	Kleiner als			
61	=	Gleich			
62	>	Groesser als			
63	?	Fragezeichen			

Copyright: SANYO VIDEO Vertrieb GmbH & Co., 2000 Hamburg 1 - Februar 1983