

creatiVision

BASIC-
INTERPRETER

REFERENZ - HANDBUCH

Version 1:0 Video Technology Ltd. 1982

Copyright: SANYO VIDEO VERTRIEB GmbH & Co.
2000 Hamburg 1

Februar 1983

INHALT

Vorwort	Seite 4
Kapitel 1	Seite 5
Einleitung	
- Was ist ein Computer ?	
Kapitel 2	Seite 6
Definition	
- Programm-Konzept	
- BASIC-Interpreter	
Kapitel 3	Seite 7
Inbetriebnahme und Anschluß des Creativison-Systems	
Kapitel 4	Seite 11
Die ersten Schritte im Programmieren	
- Direkte Befehlsausführung	
- Programmierte Befehlsausführung	
- Aufbau einer Anweisungs-Zeile	
- Die LIST-Anweisung	
- Das Löschen einer Anweisungs-Zeile	
- LIST- Programmausgabe zum Drucker	
- NEW- Löschen des gesamten Programmes	
Kapitel 5	Seite 17
Zahlen und Variablen	
- Numerische Konstanten	
- Numerische Variablen	
- LET- Anweisung	
- REM- Anweisung	
- ABS- Funktion	
- SGN- Funktion	
- RND- Funktion	
Kapitel 6	Seite 22
Daten-Bearbeitung	
- Mathematische Operatoren	
- Vergleichs-Operatoren	
- Logische-Operatoren	
- BASIC-Funktionen	
Kapitel 7	Seite 30
String-Funktionen	
- Zeichenketten (Strings)	
String Konstanten	
String Variablen	
- String Verarbeitung	
LEFT\$	
RIGHT\$	
MID\$	
CHR\$	
STR\$ LEN, VAL, ASC, String-Vergleiche	

Kapitel 8	Seite 38
System-Anweisungen	
- STOP	
- END	
- CONT	
- CTRL/C	
- RESET	
Kapitel 9	Seite 41
Schleifen- und Vergleichs-Strukturen	
- IF... THEN	
- FOR...NEXT	
- GOSUB/RETURN	
- GOTO	
Kapitel 10	Seite 46
Feld-Variablen (Array)	
- Dim	
Kapitel 11	Seite 48
Kommandos für die Ein- und Ausgabe	
- INPUT	
- PRINT	
- LPRINT	
- TAB	
- READ/DATA	
- RESTORE	
Kapitel 12	Seite 54
Programm-Speicherung auf Tonband	
- CSAVE	
- CLOAD	
- CRUN	
Kapitel 13	Seite 57
Grafik-Funktionen und Tonerzeugung	
- CLS	
- COLOR	
- CHAR	
- PLOT	
- SOUND	
- JOY	
Kapitel 14	Seite 71
Speicher Zugriff	
-PEEK	
-POKE	
Kapitel 15	Seite 72
CREATIVISION System Erweiterung	
Anhang	
(A) Liste der Anweisungen	Seite 73
(B) Fehler-Meldungen	Seite 76
(C) ASCII-Code	Seite 77

V O R W O R T

Dieses Handbuch soll den Anfänger bei seinen ersten BASIC-Versuchen mit dem CreatiVision-System begleiten und ihn, 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 Verständnis kann durch Verändern der Beispiele vertieft oder korrigiert werden. So bald wie möglich sollte mit dem Erlernten versucht werden, kleine Probleme aus dem eigenen Interessenbereich zu lösen. Nur auf diese Art und Weise können Möglichkeiten und Beschränkungen der BASIC-Problemlösungen erlernt und vertieft werden.

Durch Eingabe über die Tastatur können auf keinen Fall irgendwelche Defekte oder Zerstörungen im Computersystem ausgelöst werden. Für Anweisungen, die schon erzeugte Daten im Speicher löschen könnten, sind Vorsichtsmaßnahmen in der Beschreibung angegeben. Der Leser kann in jeder Phase des Lernens frei mit dem CreatiVision Computersystem arbeiten.

Grundsätzlich sollte der Leser dem Aufbau des Handbuches folgen. Einzelne Kapitel, wie Kapitel 12 - Kassettenspeicher - können ohne weiteres vorgezogen werden, wenn, wie hier, das Abspeichern der Programme auf Tonband geübt werden soll. Dieses Handbuch ist zwar für den Benutzer des CreatiVision-Computersystems geschrieben worden, es kann aber auch als grundsätzliche 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, daß dieses Handbuch Ihnen hilft, mit den BASIC-Grundlagen vertraut zu werden.

Wir wünschen Ihnen viel Spaß mit dem
CreatiVision Computer System!

KAPITEL 1: Einleitung

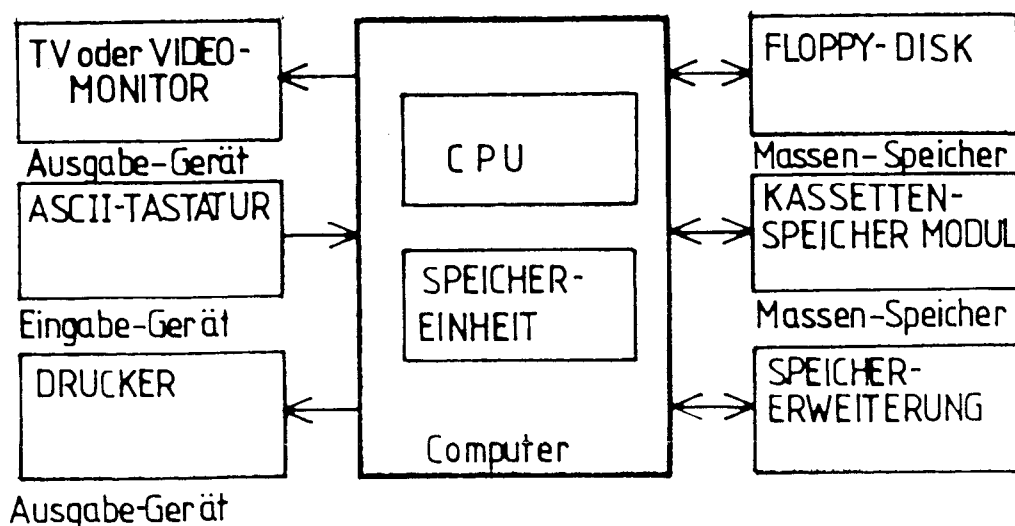
-Was ist ein Computer? -

Computer sind Geräte, die mittels einer vom Benutzer eingegebenen Anweisungsliste Tätigkeiten ausführen.

- 1) Die CPU (central processing unit). Sie führt die in der Instruktionsliste notierten Anweisungen aus. Es können arithmetische, logische und speicher-bezogene Operationen ausgeführt 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. Tonbandgerät und Floppy-Disk sind Massenspeicher. Alle Informationen und Instruktionen müssen zur Bearbeitung durch die CPU in den Computerspeicher transportiert werden.
- 4) Eingabe-Gerät (Input-Device). Eingabegeräte geben dem Benutzer die Möglichkeit, Daten oder Instruktionen in den Computer zu geben. Tastatur, Joystick, Tonbandgerät und Floppy-Disk werden als Eingabe-Geräte bezeichnet.
- 5) Ausgabe-Geräte (Output-Devices). Sie empfangen von der CPU Informationen und die Ergebnisse der Operationen. Drucker und Monitor sind typische Ausgabegeräte.

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

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



KAPITEL 2: Definition

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 für 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 übermitteln. In diesem Falle findet eine Kommunikation statt: durch das Schreiben des Programmes in einer "Programm-Sprache" und das Lesen und Ausführen dieses Programmes durch den Computer.

Heute sind sehr viele verschiedene Programm-Sprachen in Gebrauch. Sie sind teilweise für 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 Abkürzung für "Beginner s All-Purpose Symbolic Instruction Code". BASIC hat ein einfaches, englisches Vokabular und nur wenige grammatikalische Regeln. BASIC folgt einfachen mathematischen Grundsätzen. Wenn diese Programmsprache auch grundsätzlich einfach ist, so läßt sie doch die Lösung schwierigster Aufgaben zu. Sie ermöglicht die Lösung aller arithmetischen Operationen, logische Vergleiche, Aufbau von mehrdimensionalen Feldern, Listen, allgemeine trigonometrische Funktionen und die Manipulation von Zeichenketten.

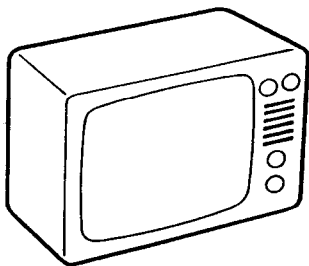
In BASIC geschriebene Programme werden von einem Übersetzungsprogramm in die Sprache der CPU gebracht. Dieses Übersetzungsprogramm wird BASIC-Interpreter genannt. Es befindet sich in der eingesteckten BASIC Kassette.

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

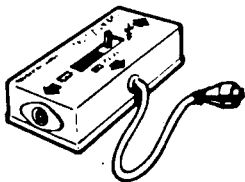
KAPITEL 3

INBETRIEBNAHME UND ANSCHLUSS DES CREATIVISION SYSTEMS

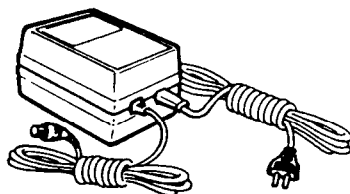
Die folgenden Teile benötigen Sie, um mit dem CreatiVision System zu arbeiten.



TV-Colorempfänger

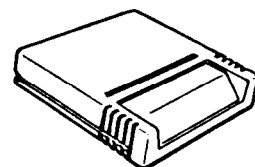
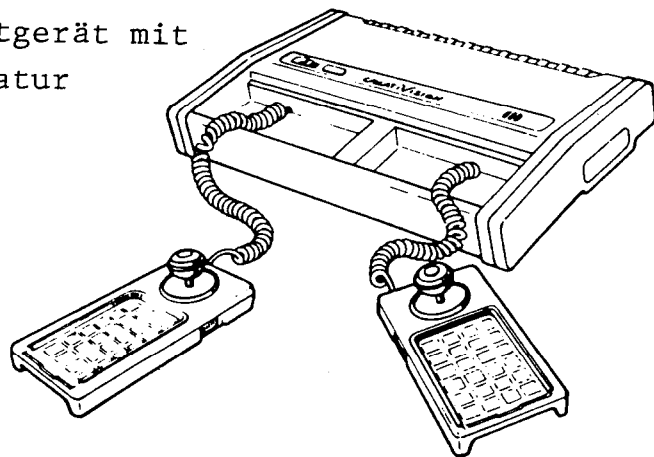


Antennenumschalter



Netzadapter

Hauptgerät mit
Tastatur

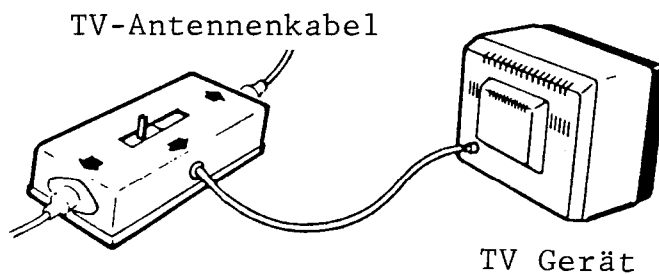


BASIC Kassette

ANTENNENUMSCHALTER

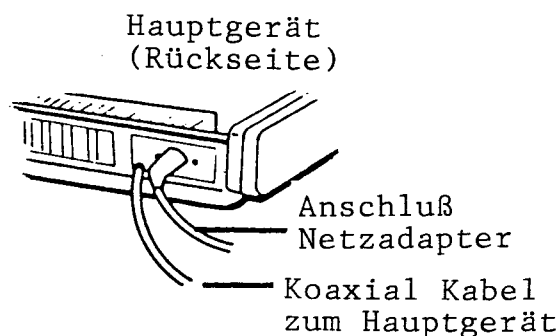
Mit dem Antennenumschalter können Sie Ihr Fernsehgerät leicht für die Arbeit mit dem System oder für normalen TV-Empfang einsetzen.

- = Ziehen Sie das Antennenkabel von Ihrem TV-Gerät ab und schließen Sie es an den Antennenumschalter an.
- = Das Kabel des Antennenumschalters schließen Sie nun an die Antennenbuchse Ihres TV Empfängers an.
- = Zuletzt schließen Sie das Kabel vom CreatiVision Gerät an den Antennenumschalter an.

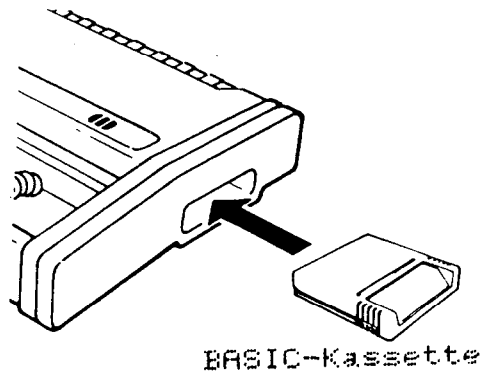


DIE SCHRITTWEISE INBETRIEBNAHME ALS COMPUTER SYSTEM

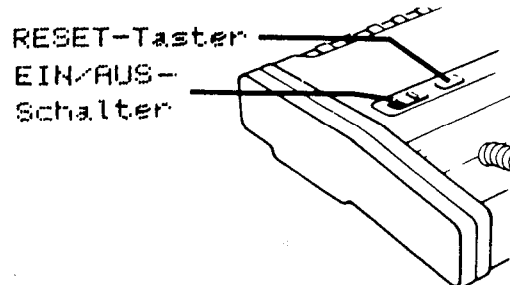
- 1) Das CreatiVision Telespielgerät muß ausgeschaltet sein.
- 2) Schließen Sie den Netzadapter an das Testspielgerät an.
- 3) Verbinden Sie dann den Netzadapter mit der Steckdose.



- 4) Stellen Sie den Schalter am Antennenumschalter auf "GAME"
- 5) Stecken Sie die BASIC Kassette in die vorgesehene Öffnung am Hauptgerät
- 6) Schalten Sie Ihr TV Gerät an und stellen Sie den vorgesehenen Kanal (VHF, BAND I, KANAL 2) ein



- 7) Stellen Sie den Schalter am Hauptgerät auf "ON". Sollten Sie Ihr System zum erstenmal in Betrieb nehmen, müssen Sie nun Ihr TV-Gerät abstimmen, ansonsten meldet sich der CreatiVision Personal Computer sofort auf dem Bildschirm.



- 8) Ihr CreatiVision System ist jetzt bereit und wartet auf Ihre Anweisungen.

VORSICHTSMASSNAHMEN

- 1) Schützen Sie alle Teile des Systems vor Feuchtigkeit.
- 2) Achten Sie darauf, daß kein Teil des Systems extremer Hitze ausgesetzt wird und den Geräten im Betrieb eine ausreichende Luftzirkulation ermöglicht wird.
- 3) Behandeln Sie die Geräte vorsichtig, lassen Sie sie nicht fallen.
- 4) Wechseln Sie die Kassetten vorsichtig und schalten Sie das Gerät zum Wechseln der Kassetten STETS aus!
- 5) Berühren Sie auf keinen Fall die in die Kassette versenkten Anschlusskontakte. Durch die Aufladung Ihres Körpers mit statischer Elektrizität können Bauteile in der Kassette zerstört werden.
- 6) Entfernen Sie die Kassette, wenn das Gerät nicht benutzt wird.

RÜCKSCHALTEN AUF TV EMPFANG

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

ZUSAMMENFASSUNG DER INBETRIEBNAHME

- 1) Setzen Sie die BASIC Kassette vorsichtig ein.
- 2) Schließen Sie den Netzadapter erst an das Gerät, 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-Gerät ein.
- 6) Schalten Sie Ihr TV-Gerät auf den richtigen Kanal.

KAPITEL 4

DIE ERSTEN SCHRITTE ZUM PROGRAMMIEREN

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

DIE DIREKTE AUSFÜHRUNG VON ANWEISUNGEN

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

```
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 Anweisungsausführung läßt sich der Computer wie ein Taschenrechner einsetzen. Zusätzliche Textausgabe und die gleichzeitige Berechnung mehrerer mathematischer Ausdrücke übertreffen einen Taschenrechner.

RUN

DIE PROGRAMMIERTE AUSFÜHRUNG VON ANWEISUNGEN

Jede Anweisung, der eine "Zeilennummer" vorangestellt ist, wird nicht nach Drücken der Taste RETURN ausgeführt, sondern wird in den BASIC Programmspeicher eingeschrieben. Eine Ausführung 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 ausgeführt wird. In dieser Reihenfolge werden alle Anweisungen eines Programmes ausgeführt. Anweisungszeilen mit niedrigeren Zeilennummern werden vor den mit höheren Zeilennummern ausgeführt.

Es ist möglich, im Programm mit den Anweisungen GOTO und GOSUB die Ausführung 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 trägt. 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 "ERGBNIS", (2*6) (RETURN)

          RUN 20 (RETURN)
          6      0.5
          ERGBNIS 12

```

AUFBAU EINER ANWEISUNGSZEILE

- 1) Jede Anweisungszeile beginnt mit einer Zeilennummer.
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 Programmausführung ab und gibt die fehlerhafte Zeile, gefolgt von der Meldung "SYNTAX-ERROR" auf den Bildschirm aus.

Beispiel: 10 PRINT 2+3 (RETURN)
 RUN (RETURN)
 10 PRINT 2+3
 SYNTAX ERROR

- 4) Jede Programmzeile wird mit Drücken der Taste (RETURN) in das Programm übernommen.

EDITIEREN

Wird bei der Eingabe ein Fehler gemacht, so kann mit der Taste (←) die fehlerhafte Eingabe gelöscht werden.

Beispiel: PRINT
 Taste ← drücken. Der Bildschirm zeigt
 PRIM
 Taste ← noch einmal drücken
 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

LÖSCHEN EINER ZEILE

Um eine Anweisungszeile aus dem Programm zu löschen, 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 gelöscht. Versuchen Sie selber, Zeile 33 zu löschen.

LLIST

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

Syntax: LLIST

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

NEW

Die Anweisung "NEW" ermöglicht das Löschen des gesamten Programmspeichers.

Beispiel: NEW (RETURN)
 LIST (RETURN)

Auf das LIST Kommando erfolgt keine Programmausgabe mehr. Der BASIC Programmspeicher ist gelöscht.

ANMERKUNG

Sie werden jetzt mit dem Gebrauch der (RETURN)-Taste ausreichend vertraut sein.

Ab dem nächsten Kapitel wird die Anwendung dieser Taste in den Beispielen nicht mehr ausgedruckt.

KAPITEL 5

ZAHLEN UND VARIABLEN

NUMERISCHE KONSTANTEN

Numerische Konstanten sind Zahlen, die während des gesamten Programmbablaufes ihres Wert behalten. Sie können positive oder negative Vorzeichen haben. Sie werden dezimal oder technisch-wissenschaftlich notiert.

Beispiel: +2, 34, 0.432, 3E18, 4E(-35), usw.

Der Bereich einer numerischen Konstante ist:

$$10 \leq n \leq 10$$

NUMERISCHE VARIABLEN

Eine Variable ist eine Information, die während des Programmlaufes ihren Wert oder Inhalt ändern kann. Eine numerische Variable wird vom Programmierer mit einem festgelegten Namen beschrieben. Alle Buchstaben von A-Z können als Variablennamen benutzt werden, so daß 26 Variablen vereinbart werden können. Der Wert der Variablen bleibt solange unverändert, wie ihr nicht mit den Anweisungen LET oder INPUT ein neuer Wert zugewiesen wird oder der Wert nicht mit der Anweisung FOR verändert 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 Programmiererpraxis, zu Beginn des Programmes alle Variablen mit ihrem Anfangswert zu beschreiben.

LET

DIE L E T ANWEISUNG

Diese Anweisung weist 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 ermöglicht die Definition von Ausdrücken und weist das Ergebnis der benannten Variable zu.

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 erhöht, das Ergebnis 10 wird dann der Variablen 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 R E M 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 A B S O L U T FUNKTION

Die ABS Funktion formt den absoluten Wert eines Ausdrucks. Als Beispiel:

ABS (-34.67) = 34,67

Syntax: ABS (Ausdruck)

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

```
RUN
23
```

SGN

DIE SGN-FUNKTION

Mit der SGN Funktion läßt 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
```

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
6
4
7
2
8

KAPITEL 6

DATEN-BEARBEITUNG

OPERATOREN

MATHEMATISCHE OPERATOREN

Es sind 4 Arten Operatoren im CreatiVision Basic:

Mathematische Operatoren

Der BASIC-Interpreter kann Addition, Subtraktion, Multiplikation, Division und Exponentiation ausführen. Zu errechnende mathematische Formeln können in der gewohnten Form im Programm notiert werden. Es stehen fünf Operatoren zur Verfügung:

<u>Operator</u>	<u>Beispiel</u>	<u>Bedeutung</u>
+	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 Ausdrücken ändert 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 Ausdrücke zuerst berechnet.
- 2) Werden keine Klammern gesetzt, werden die Operationen zur Errechnung in der folgenden Reihenfolge vom BASIC Interpreter bearbeitet:

Exponentiation

Neg. Vorzeichen auflösen

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 Ausdrücke übersichtlicher und fehlerfreier zu gestalten.

Beispiele: Algebraischer Ausdruck BASIC Ausdruck

$3X-2Y$
 $(X^2)Y$
 B^2-4AC

$3*X-2*Y$
 $X**2*Y$
 $B**2-4*A*C$

VERGLEICH - OPERATOREN

VERGLEICH - OPERATOREN

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

Operator	Getestetes Verhältnis	Ausdruck
=	Gleich	X=Y
><	Ungleich	X><Y, X<>Y
<	Kleiner als	X<Y
>	Größer als	X>Y
<=,=<	Kleiner oder gleich	X<=Y, X=<Y
>=,=>	Größer 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 werden in IF...THEN Anweisungen benutzt, um die Ergebnisse mehrerer Vergleiche logisch miteinander zu verknüpfen. A und B in der nachfolgenden Beschreibung sind Verhältnis-Ausdrücke mit den Werten "wahr" (1) und "unwahr" (0). Logische Operatoren werden nach arithmetischen und Verhältnis-Operatoren ausgeführt.

<u>Operator</u>	<u>Beispiel</u>	<u>Bedeutung</u>
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 für alle Kombinationen für A und B.

A	NOT A
wahr	unwahr
unwahr	wahr

Wahrheitstabelle der NOT-Funktion

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

Wahrheitstabelle der "AND"-Funktion

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

Wahrheitstabelle der "ODER" Funktion

Beispiel: 10 INPUT A,B,C
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

In vielen mathematischen Problemlösungen werden relativ einfache mathematische Operationen angewendet. Für viele dieser Lösungen 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 können, 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 natürlichen Logarithmus von 150 zu ermitteln. Wenn solche Werte in Ausdrücken gebraucht werden, können Funktionen wie:

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

eingesetzt werden.

Die nachfolgende Tabelle gibt die vom BASIC Interpreter geführten Funktionen an:

Funktion	Bedeutung
ABS (X)	Ergibt den Absolutwert von X
SGN (X)	Ergibt +1, -1 oder 0 entsprechend X
	Ergibt die größte 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 natürlichen 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 $X -1000 < X < 1000$.

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)

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 Anführungsstriche 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 für 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

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

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

RUN
XYZ123PQR

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. hält der BASIC Interpreter eine Reihe von Funktionen zur Bearbeitung von Zeichenketten bereit. Damit können 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 unterstützt.

LEFT \$

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

Diese Funktion ergibt nach Ausführung 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 Länge 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 Länge 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
ABCDEF

STR \$

Syntax: STR \$(numerischer Ausdruck)

Diese Funktion wandelt einen numerischen Ausdruck in einen Stringausdruck.

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

RUN
7 007

Beispiel: 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 Länge 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 Verhältnis-Testoperatoren können auch auf Stringausdrücke angewandt werden. (A\$="123A", "X">"AXE")
Verglichen wird der ASCII Wert jedes einzelnen Zeichen der Ketten. Entsprechend 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 größer als das "C" in A\$ mit dem ASCII Wert 67. Damit ist der Vergleich B\$>A\$ wahr, und die 20 wird ausgegeben.

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

```
RUN  
1  
2
```

```

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

```

```

RUN
BA

```

```

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

```

```

RUN
ORANGE

```

```

Beispiel: 10 REM PRINT ASCII CHARACTERS
          20 FOR I=1 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.

KAPITEL 8

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 Ausführung der STOP Anweisung wird auf dem Bildschirm ein STOP AT Zeilennummer ausgegeben.

Ein CONTINUE Kommando nach Ausführung der STOP Anweisung führt 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 Ausführung 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 Programmausführung. Sie ist die letzte Anweisung im Programm.

Syntax: Zeilennummer END

Die Zeilennummer der END-Anweisung ist normalerweise die letzte und höchste 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 Ausführung 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 Programmablauf und gibt die Zeilennummer der STOP Zeile aus. Der Programmierer kann dann Daten untersuchen und im Kommandobetrieb ändern und mit dem Kommando CONT den Programmablauf fortsetzen.

CTRL/C

Es wurde besprochen, wie mit STOP ein Programm unterbrochen werden kann. Ein Programmabbruch ist auch jederzeit durch gleichzeitiges Drücken der Tasten CTL und C möglich. Wir bei STOP, können jetzt aus der Kommandoebene Untersuchungen vorgenommen werden und dann kann der Programmablauf 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:

CTRL

C

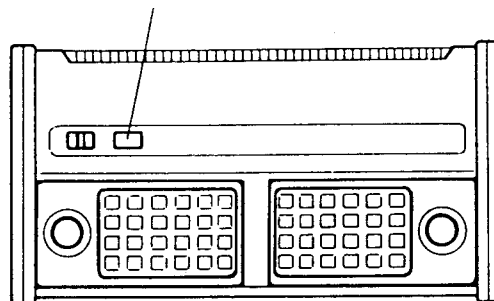
Beide Tasten sind gleichzeitig zu drücken!

RESET

Der RESET Taster befindet sich neben dem Netzschalter des Hauptgerätes. Mit diesem Taster wird der Computer aus dem Programmlauf in eine Grundposition zurückgesetzt. Alle durch Grafikanweisungen geänderten Zeichen erhalten wieder ihre ursprüngliche Form.

Befindet sich ein Programmlauf durch Fehler im Programm in einem Zustand, aus dem auch mit CTRL/C nicht mehr in die Kommandoebene zurückgekehrt werden kann, so ist der RESET Taster zu betätigen. Das existierende BASIC Programm wird nicht beeinflusst.

RESET-Taster



KAPITEL 9

SCHLEIFEN UND VERGLEICHS-STRUKTUREN

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

IF...THEN

Der Programmablauf wird entsprechend dem Ergebnis eines logischen Vergleiches gesteuert.

Die IF...THEN Anweisung führt 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 Stärken von Computersystemen ist ihre Fähigkeit, 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 benötigt, 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 fünfmal, bei jedem Schleifendurchlauf um eins erhöht. Sollen nur die Quadrate aller ungeraden Zahlen von 1 bis 10 errechnet werden, so muß N bei jedem Lauf um 2 erhöht 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 verändert wird. Der Ausdruck nach "=" ist der Startwert der Schleifenvariable, der Ausdruck nach TO ist der Endwert. Wird STEP nicht definiert, so wird die Schleifenvariable automatisch mit +1 erhöht.

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

Syntax: NEXT Variable

Der Variablenname in der NEXT Anweisung muß der gleiche wie in der FOR Anweisung sein.

RETURN

GOTO

Die GOTO Anweisung übergibt den Programmmlauf direkt und ohne Bedienung an die nach GOTO definierte Anweisungszeile. Normalerweise ist die mit GOTO angesprochene Anweisung nicht die auf die GOTO Anweisung folgende.

Syntax: GOTO Zeilennummer oder numerischer Ausdruck

Die Zeilennummer, an welcher die Programmausführung fortgesetzt wird, kann größer oder kleiner ALS die Zeilennummer der GOTO Anweisung sein. Eine Verzweigung vorwärts oder rückwärts im Programmmlauf wird dadurch möglich.

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 ausgeführt, 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 (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)

DIM

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

Die DIM Anweisung im Programm muß gegeben werden, bevor zum erstenmal eine Variable dieses Feld angesprochen wird. DIM A(6) dimensioniert ein eindimensionales Feld mit sechs Elementen, während DIM B(6,6) ein zweidimensionales Feld für die indizierte Variable "B" mit $6 \times 6 = 36$ 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 $3 \times 5 = 15$ Speicherplätze für 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 benötigt 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 FÜR DIE EIN- / AUSGABE

INPUT

Mit der Anweisung "INPUT" werden Daten von der Tastatur übernommen. 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 mögliche Zahl!
?4
3      4      7      Computerausgabe!
```


PRINT

Die PRINT Anweisung gibt die Werte von Variablen und Ausdrücken auf dem 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 Ausdrücke durch ein "," getrennt, so erfolgt die Ausgabe auf den Bildschirm an drei vortabulierten Positionen auf dem Bildschirm. Das Erstellen von Tabellen wird damit unterstützt. (siehe voriges Beispiel).

Werden PRINT Anweisungen mit einem ";" gegeben, so wird damit das Fortschalten auf die nächste Zeile des Bildschirms unterdrückt. (Es wird kein Wagenrücklauf 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 LPRINT 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 müssen betriebsbereit sein, bevor Programme mit LPRINT Anweisungen bearbeitet werden.

TAB

Die TAB Funktion wird in PRINT Anweisungen verwendet und tabuliert die nächste Ausgabe auf dem Bildschirm. Die vorgesehene Ausgabestelle wird durch die in Klammern angegebene Spaltennummer des Bildschirms festgelegt.

Syntax: TAB (Variable)

TAB(N) stellt den Cursor auf die N-te Spalte des Bildschirms. 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 übergeben 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 hält Informationen für die READ Anweisung bereit. Die READ Anweisung übernimmt diese Daten in der Reihenfolge der Zeilennummern. Die DATA Zeilen müssen 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
```

Hinweis : READ und DATA bearbeiten nur numerische Konstanten..

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 nächste 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

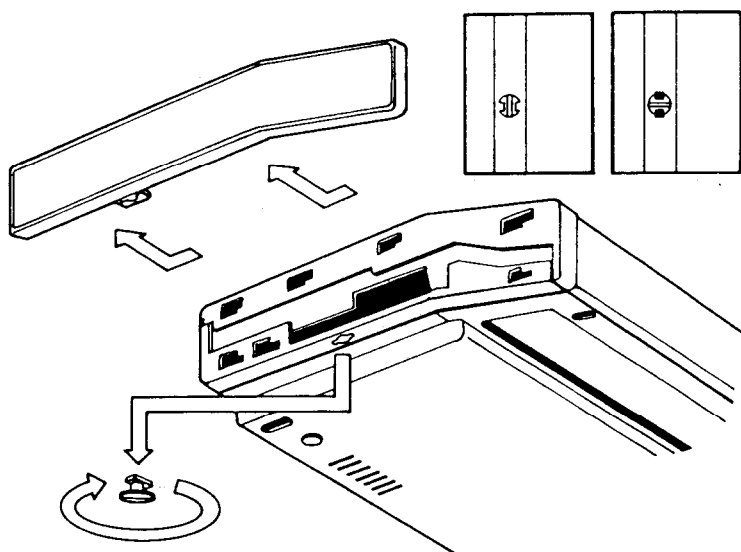
KASSETTENSPEICHER MODUL

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 Kassettenpeicher-Modul wird an den Computer über ein Kabel wie in Bild 12.1 angeschlossen.

Dieses Seitenteil wird entfernt

Arretiert Geöffnet



CSAVE

Syntax: CSAVE

Ablauf:

- 1) Verbinden Sie das Kassettenspeicher-Modul über das Kabel mit dem Computersystem.
- 2) Verwenden Sie eine qualitativ gute Kassette.
- 3) Drücken Sie die Tasten "PLAY" und "RECORD" am Recorder gleichzeitig.
- 4) Notieren Sie die Anzeige des Bandzählwerkes am Kassettenspeicher-Modul. Beim späteren Laden des Programmes kann das Band damit in die richtige Position gebracht werden.
- 5) Geben Sie das CSAVE Kommando über die Tastatur des Computers. Das Band läuft 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 drücken.

CLOAD

Syntax: CLOAD

Ablauf

- 1) Verbinden Sie das Kassettenspeicher Modul über das Kabel mit dem Computersystem.
- 2) Setzen Sie die Kassette ein und spulen Sie sie auf die korrekte Position.
- 3) Drücken Sie die Taste PLAY am Kassettenrekorder.
- 4) Geben Sie das CLOAD Kommando über 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 Prompt (">") auf den Bildschirm ausgegeben und der Kassettenrekorder sollte durch die STOP Taste abgeschaltet werden.

Wird ein Programm aus der

 CreatiVision - Programm - Bibliothek

geladen, so sind Erklärungen und Musik im Tonkanal des TV-Gerätes hörbar.

CRUN

Syntax: CRUN

Das Kommando CRUN entspricht den Kommandos CLOAD+RUN. Der Computer lädt das Programm und startet es dann automatisch. CRUN kann auch die letzte Anweisung in einem Programm sein. Bleibt die PLAY Taste am Kassettenspeicher Modul gedrückt, wird das nächste 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)

KAPITEL 13

GRAFIK-FUNKTIONEN UND TONERZEUGUNG

GRAFIK KOMMANDOS

Die Grafik Kommandos arbeiten auf den in 32 Spalten und 24 Zeilen organisierten Bildschirm. Da viele TV-Geräte den Bildschirm links und rechts überschreiben, 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:

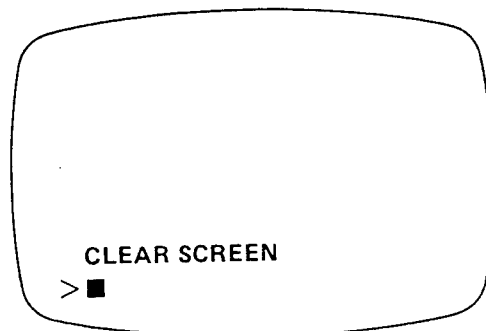
- 1) CLS - Bildschirm löschen
- 2) COLOR - Farbe definieren
- 3) CHAR - Character definieren
- 4) PLOT - Character auf definierte Bildschirmstelle ausgeben

CLS

Mit dem CLS Kommando (clear screen) wird der Bildschirm gelöscht. Wird im Programm eine CLS Anweisung ausgeführt, so werden alle Bildschirmspeicher mit dem ASCII Code für SPACE=32 gefüllt.

Beispiel: 10 CLS
 20 PRINT "CLEAR SCREEN"

RUN



COLOR

Die COLOR Anweisung gibt alle Möglichkeiten, 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 das Zeichen umgebende Farbe (Hintergrundfarbe). 16 Farben sind im CreatiVision BASIC definierbar.

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

Die Farbumschaltung mit der COLOR Anweisung beeinflusst jeweils eine Gruppe von 8 Zeichen. Im ersten Argument der COLOR Anweisung ist die gewünschte 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, daß leere Bildschirmspeicher mit dem Code für SPACE=32 gefüllt sind. Wenn Sie den Charactersatz 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 Hellgrün zu Hellblau.
Programmstop mit CTL/C !

CHAR

Mit der CHAR Anweisung können 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 für 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 8x8 Gitter der neue Grafikcharacter gebildet wird.

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

LINKE RECHTE
BLOECKE BLOECKE

ZEILE1 00000000
ZEILE2 00000000
ZEILE3 00000000
ZEILE4 00000000
ZEILE5 00000000
ZEILE6 00000000
ZEILE7 00000000
ZEILE8 00000000

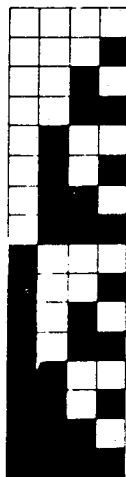
Jede Zeile: 0000 0000
Linker Block ↑ Rechter Block

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 nächsten 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.

BLÖCKE



0 AUS
1 EIN

0 AUS	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

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 für den Zeichencode -32.

Beachten Sie, daß die CHAR-Anweisung nur einen Character definiert. Die Ausgabe auf den Bildschirm wird mit der Anweisung PLOT vorgenommen. Wird eine CHAR Anweisung im Programm durchgeführt, ändern 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 gefüllt, weil sein Code 32 ist. 32 ist aber auch der Code für SPACE, mit dem der Bildschirm nach Ausführung der Anweisung CLS gefüllt war.

Beachten Sie, daß die Character 32 bis 95 durch den Computer definiert sind. Eine Neudefinition ist möglich, wie das Beispiel für den Code 32 zeigt.

Alle definierten Farben und Character werden mit Betätigung des RESET Tasters in den Zustand nach dem Einschalten zurückversetzt.

PLOT

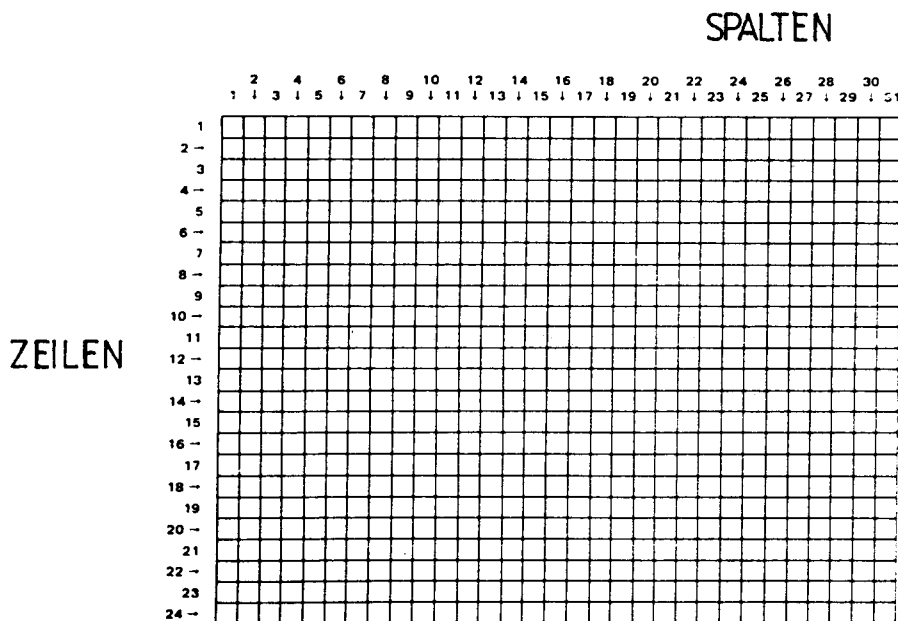
Die PLOT Anweisung ermöglicht es, ein Zeichen an jede beliebige Stelle des Bildschirmes auszugeben.

Syntax: PLOT Spaltennummer, Zeilennummer, Charactercode

z.B. PLOT 15,10,65

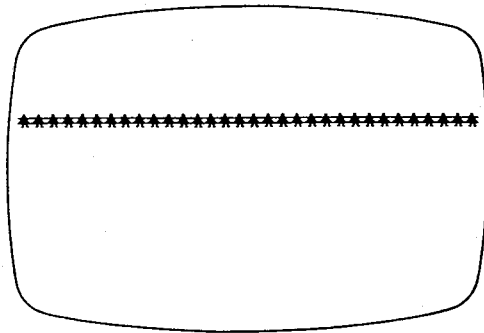
Ein "A" wird in die 15.Stelle der Zeile 10 ausgegeben.

Ist die Spaltennummer =1, wird die Ausgabe ganz links auf dem Bildschirm erfolgen. Ist die Zeilennummer =1, wird die Ausgabe am obersten Rand des Bildschirmes erfolgen. Da die meisten TV Empfänger den Bildschirm links und rechts überschreiben, sollten für Grafiken nur die Spalten 3 bis 30 benutzt werden.



Beispiel: 10 CLS
20 COLOR 5,4,6
30 CHAR 32,18 18 FF 3C 7E FF 14 36
40 FOR X=1 TO 29
50 PLOT X,10,32
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,X⁴+128
70 NEXT Y
80 NEXT X
90 GOTO 90

RUN

Ein Farbbalken wird erzeugt.

SOUND

Die SOUND Anweisung veranlasst den Computer, Töne unterschiedlicher Höhe in drei Kanälen gleichzeitig auszugeben.

Syntax: SOUND Frequenz;Dauer;Frequenz;Dauer;Frequenz;Dauer
z.B. SOUND 4;5;6;7;7;7 (Klang einer Glocke)

Die Codes für die Frequenz und die Dauer eines Tones sind numerische Ausdrücke. Ist das Ergebnis eines Ausdruckes eine nicht integer Zahl, so wird die auf den nächst höheren Wert abgerundet.

CODE TON

1 Rest	17 D [#] , E ^b
2 C	18 E
3 C [#] , D ^b	19 F
4 D	20 F [#] , G ^b
5 D [#] , E ^b	21 G
6 E	22 G [#] , A ^b
7 F	23 A (above middle C)
8 F [#] , G ^b	24 A [#] , B ^b
9 G	25 B
10 G [#] , A ^b	26 C (high C)
11 A (below middle C)	27 C [#] , D ^b
12 A [#] , B ^b	28 D
13 B	29 D [#] , E ^b
14 C (middle C)	30 E
15 C [#] , D ^b	31 F
16 D	32 Rest

CODE TAKT

0	$\frac{1}{4}$	
1	$\frac{1}{2}$	
2	$\frac{3}{4}$	
3	1	
4	$1\frac{1}{2}$	
5	2	
6	3	
7	4	

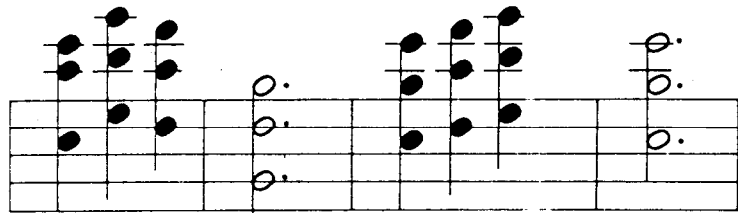
Das 1. Datenpaar in einer SOUND Anweisung gilt dem 1. Tonkanal, das 2. Paar dem 2. Kanal und das 3. Paar dem 3. Tonkanal.

```

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;3,18;3
          90 SOUND 26;7,21;7,14;7
          100 GOTO 20

```

RUN



Der Computer spielt eine Melodie nach BILD 13.8,

JOY

Die JOY Funktion (Joy-stick = Steuerknüppel) erlaubt dem Programmierer, Informationen über die Position der Steuerknüppel an das Programm zu geben. Damit wird der Entwurf von Spielen durch den Benutzer unterstützt.

Die Tastatur des CreatiVision Computers ist in zwei Teile geteilt.- Jeder Teil hat einen Steuerknüppel und zwei "Feuer"-Taster. Informationen über den Stand dieser Bedienungselemente werden mit der JOY Funktion ausgewertet.

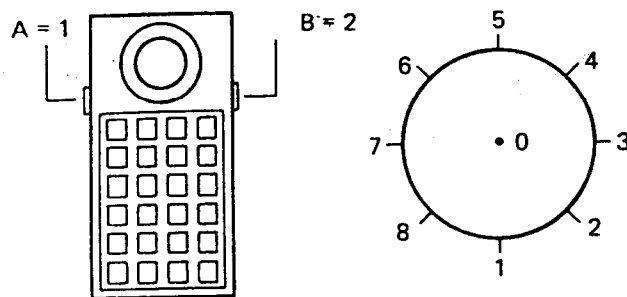


Bild 13.8 Die den Joy-stick-Positionen und den "FEUER"-Tasten zugeordneten Werte.

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 prüft den linken Joy-stick. Entsprechend Bild 13.8 ist er nach rechts gedrückt.

Beispiel: 05 CLS
10 LEFT 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

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

Während die PEEK Funktion eine Speicherzelle liest, 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 geprüft werden. Versuchen Sie es mit anderen Werten!

KAPITEL 15

creatiVision - SYSTEM - ERWEITERUNGEN

Es sind diverse Erweiterungsmöglichkeiten für Ihr CreatiVision System vorgesehen. Sollten Sie mehr Informationen als die in diesem Kapitel gegebenen benötigen, wenden Sie sich bitte an Ihren Händler oder an die in diesem Handbuch angegebene Adresse.

1) Das Kassettenspeicher Modul

Mit den Anweisungen CSAVE und CLOAD können Programme auf Band geschrieben werden und vom Band gelesen werden. Die Recorderfunktionen werden vom Programm gesteuert. Die Datenübertragung wird mit 600 Baud vorgenommen.

2) Das I/O Interface für parallele und serielle Datenübertragung.

Viele Möglichkeiten für die Kommunikation mit der Außenwelt stellt das Interfacemodul zur Verfügung. Ein Parallelport (Centronics Bus) ist für 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 gummibeleagten Tasten dieser Standard Tastatur erleichtern das Arbeiten mit Ihrem Computer System.

BASIC QUICK REFERENCE

- Zahlen werden mit einer Genauigkeit von 6 Stellen verarbeitet
- Der numerische Bereich: $10 \text{ E}-38 \leq N \leq 10 \text{ E } 38$
- Alle Anweisungen können vom Programm oder als Kommandos ausgeführt 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 - natürlicher Logarithmus

4) String Funktionen

LEN - Länge 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 Verhältnis- und logische Ausdrücke setzen
OR Wahr=1 und unwahr=0
NOT

6) Grafik- und Tonfunktionen

CLS - Bildschirm löschen
PLOT - Character auf Bildschirm plotten
COLOR - Farbe setzen
SOUND - Ton versch. Frequenz und Dauer erzeugen
CHAR - Character definieren
JOY - Joy stick prüfen
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

FEHLER-MELDUNGEN

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

CODE FEHLER

00	Fehler während 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) überbelegt
11	Nicht zulässiger Operand
12	IF falsch definiert
13	Falsche Klammer Anordnung
14	Zuviele Klammerebenen
15	String ist nicht definiert
16	Stringearbeitung fehlerhaft
17	Verbotene Division durch Null
18	Mehr READ Anweisungen, als Daten definiert sind
19	Speicherbereich für Daten überbelegt
20	DIM Anweisung nicht ordnungsgemäß
21	Zeichenkette zu lang

ASCII CODE		ZEICHEN (Character)
32		Leerzeichen
33	!	Ausrufe-Zeichen
34	"	Anführungs Zeichen
35	#	Nummern Zeichen
36	\$	Dollar
37	%	Prozent
38	&	Und Zeichen
39	'	Apostroph
40	(Klammer öffnen
41)	Klammer schließen
42	*	Asterisk
43	+	Plus
44	,	Komma
45	-	Minus
46	.	Punkt
47	/	Division
48	0	
49	1	
50	2	
51	3	
52	4	
53	5	
54	6	
55	7	
56	8	
57	9	
58	:	Doppelpunkt
59	;	Semicolon
60	<	Kleiner als
61	=	Gleich
62	>	Größer als
63	?	Fragezeichen
64	@	AT-Zeichen
65	A	
66	B	
67	C	
68	D	
69	E	
70	F	
71	G	
72	H	
73	I	
74	J	
75	K	
76	L	
77	M	
78	N	
79	O	
80	P	
81	Q	
82	R	
83	S	
84	T	
85	U	
86	V	91 [
87	W	92 \
88	X	93]
89	Y	94 ↑
90	Z	95 ■