sinclair

ZX Spectrum+

Handbuch für Benutzer

SPECTRUM SOFTWARE

Das gesamte Software-Sortiment, das für Spectrum Computer erhältlich ist (einschließlich aller bestehenden Titel), ist absolut kompatibel mit Ihrem neuen ZX Spectrum +.

EINFÜHRUNG IN DEN ZX-SPECTRUM+

Sinclair Research führt schon lange auf dem Gebiet der Microchip-Technologie, die den Computer für jeden ermöglicht hat. Nach der Ankunft des ersten Mikrocomputers zum niedrigen Preis, des ZX 80, haben wir erhöhte Computerleistung und größeren Wert in seinen Nachfolgern, dem ZX 81, ZX Spectrum und dem QL, kombiniert. Einfache Benutzung ist für uns ein wichtiger Punkt, sowohl im Design als auch in seiner Arbeitsweise. Der ZX Spectrum + führt Sinclair Research noch einen Schritt weiter auf diesem Weg. Hier haben Sie ein Gerät mit den besten Merkmalen des Spectrums in einer verbesserten Version, das die Benutzung dieses leistungsstarken und populären Computers noch einfacher macht. Wir hoffen, daß Sie vollen Gebrauch von den vielen Möglichkeiten machen, die Ihr neuer Computer Ihnen eröffnet.

Clive Sinclair

INHALT

VORBEREITUNGSMASSNAHMEN 3

PROGRAMMIEREN 17

ALLES ÜBER IHREN ZX SPECTRUM + 41

ALLES ÜBER SINCLAIR BASIC 49

Autor: Neil Ardley
Veröffentlich von Dorling Kindersley Ltd
In Zusammenarbeit mit
Sinclair Research Ltd.

Zur Benutzung dieses Handbuches

Dieser Führer zu Ihrem ZX Spectrum + enthält vier farbcodierte Kapitel. Um ein bestimmtes Kapitel zu finden, das Buch nur einfach an der Stelle mit der richtigen Farbe öffnen.

1 Vorbereitungsmaßnahmen

Aufbau Ihres ZX Spectrum + ■ Einstellung Ihres Fernsehgerätes ■ Fehlerbeseitigung beim Aufbau ■ Was Ihr ZX Spectrum + alles kann ■ Zur Benutzung von fertiger Software ■ Wie man ein Programm lädt ■ Fehlersuche beim Laden der Software

2 Programmieren

Die Tastatur – das Bedienungsfeld Ihres Computers ■ Wie man die Tasten bedient ■ Der Fernseh-Rechner ■ Farben und ihre Benutzung ■ Einfache Do-it-yourself Grafik ■ Der Bildschirmzeichenblock ■ Entwerfen Sie Ihre eigenen Muster und Bilder ■ Wie Sie Ihre eigenen ■ Computerzeichen erzeugen können ■ Bewegung ■ Musik und Toneffekte ■ Wie man Programme sichert ■ Fehlersuche beim Sichern von Software

3 Alles über Ihren ZX Spectrum +

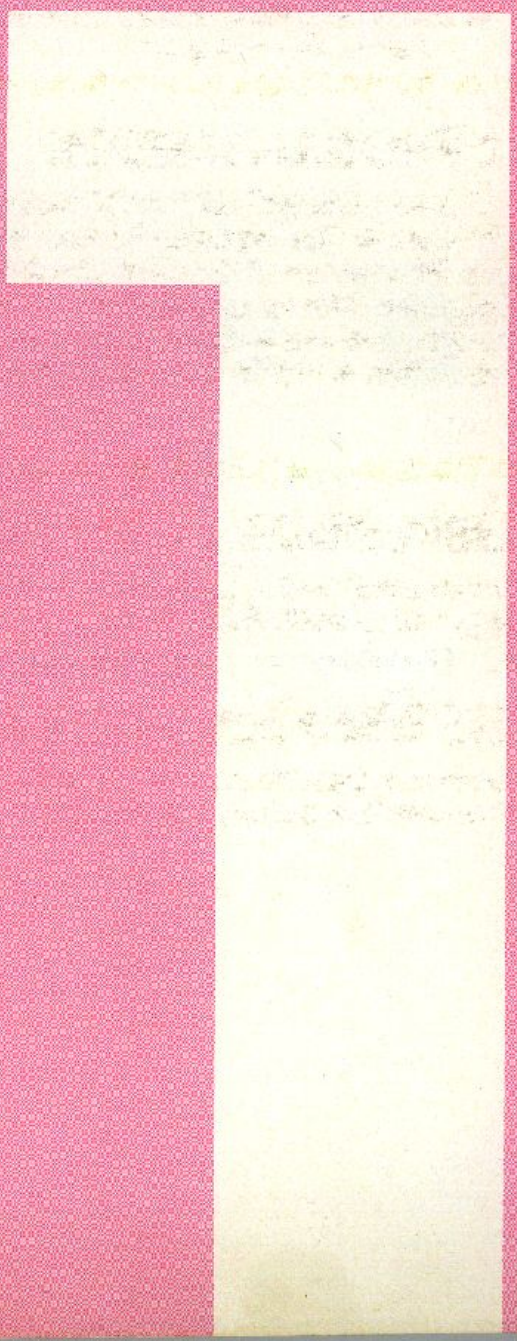
Wie sieht er innen aus? ■ Wie funktioniert Ihr Spectrum? ■ Wie man Peripheriegeräte anschließt ■ ZX Spectrum + Memory-Plan

4 Alles über Sinclair BASIC

Sinclair BASIC-Schlüsselworte zum Nachschlagen ■ Bildschirmmeldungen ■ Über BASIC hinaus ■ Computer-Jargon und seine Bedeutung

VORBEREITUNGSMAßNAHMEN

In diesem Kapitel lernen Sie, das Potential Ihres ZX Spectrum + zu erforschen. Sie erfahren, wie Sie Ihren Computer aufbauen müssen, so daß er in Aktion treten kann, wann immer Sie wollen. Und dann können Sie wählen. Sie können verschiedene Programme eintippen und der Computer durchläuft die jeweiligen Schritte und zeigt Farbgrafiken und Toneffekte – oder Sie können herausfinden, wie man fertige Software benutzt, wie, zum Beispiel, Computer-Spiele. Ganz egal, was Sie wählen- Sie werden bald sehr viel Freude an Ihrem neuen Computer haben!



AUFBAU IHRES ZX SPECTRUM +

Um Ihren Spectrum startbereit zu machen, gehen Sie zunächst durch die untenstehende Prüfliste, um sicherzustellen, daß Sie alles haben, was Sie brauchen – danach die Anweisungen auf der gegenüberliegenden Seite über Anschlüsse und Anschalten befolgen. Alle Teile müssen fest miteinander verbunden sein. Wenn Sie aus Versehen das Netzteil abtrennen oder ausschalten, während der Spectrum in Betrieb ist, verlieren Sie Ihr Programm und alle Ihre Informationen oder Ergebnisse.

Wenn Sie fertig sind mit dem Computer, ziehen Sie den Stecker aus der Steckdose.

Prüfliste: Haben Sie alles, was Sie brauchen?

Beim Auspacken finden Sie:

1 Ihren ZX Spectrum +

2 Das ZX-Netzteil – dieses liefert die vom Spectrum benötigte 9 Volt Gleichstromzufuhr.

3 Das Antennenkabel – dieses benutzen Sie für den Anschluß Ihres Spectrums an ein Fernsehgerät

4 Das Kassettenkabel – dies ist für den Anschluß Ihres Spectrums an einen Kassettenspieler

5 Eine Garantiekarte, die Sie ausfüllen und zurücksenden müssen

6 Eine Kassette zum Benutzerhandbuch

7 Dieses Handbuch

Sie selbst müssen bereit haben:

1 ein Fernsehgerät
2 einen Kassettenspieler
3 einen Netzstecker

Fragen und Antworten zum Aufbau

Muß ich ein Farbfernsehgerät benutzen?
Nein. Allderdings können Sie auf einem Schwarz-Weiß-Fernseher nicht die vom Spectrum erzeugten Farben sehen!

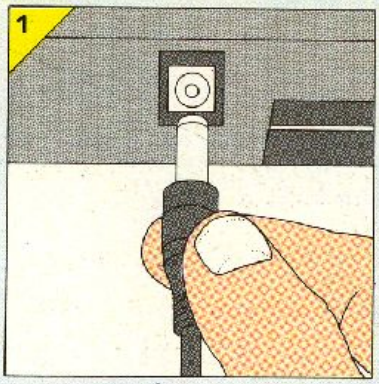
Kann ich jeden Fernseher benutzen?
Ihr Spectrum müßte ein Bild auf jedem beliebigen Fernsehgerät erzeugen. Ist dies nicht der Fall, liegt dies vielleicht daran, daß Fernsehgerät und Computer verschiedene Farbsysteme haben. Dies passiert manchmal, wenn der Fernseher sehr alt ist oder wenn Fernseher und Spectrum in verschiedenen Ländern gekauft wurden. Sollten Sie Zweifel haben, wenden Sie sich an Ihren Fernsehhändler.

Kann ich einen Monitor statt eines Fernsehers benutzen?
Ja. Ihr Händler kann Ihnen vielleicht einen Monitor besorgen, der ein besonders gutes Bild für den Spectrum hat.

Kann ich das ZX 16K RAM benutzen?
Nein. Dieses RAM-Paket ist nur für den Sinclair ZX81 Computer.

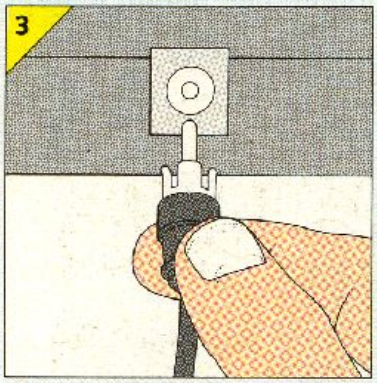
Das Einschalten Ihres ZX Spectrum +

Jetzt befolgen Sie bitte die folgenden Abbildungen, um Ihren Spectrum an das elektrische Netz und Ihr Fernsehgerät anzuschließen. Ist das System eingeschaltet, finden Sie auf der folgenden Seite alles über die Einstellung.

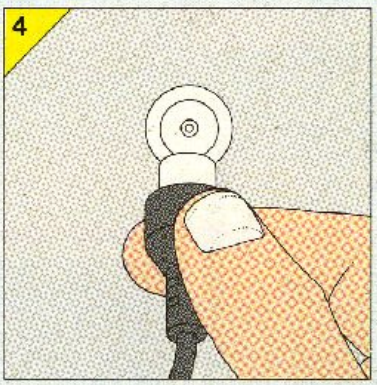


Den kleinen Stecker am Netzteildraht in die mit 9VDC bezeichnete Buchse an Ihrem Spectrum stecken.

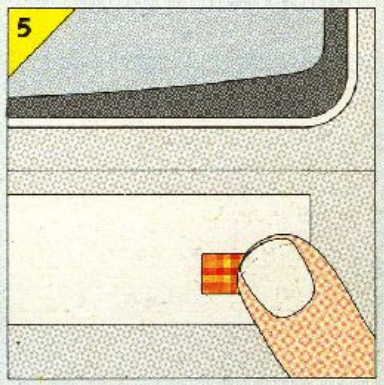
Den Hauptnetzstecker in eine Steckdose stecken. Ihr Spectrum hat keinen eigenen AN/AUS-Schalter.



Das Antennenkabel in die Buchse mit der Bezeichnung TV an Ihrem Spectrum stecken. Nur einer der Stecker an dem Antennenkabel paßt in diese Buchse.



Das Antennenkabel von Ihrem Fernseher entfernen. Den anderen Stecker am Antennenkabel Ihres Spectrums in die Antennenbuchse am Fernseher stecken.

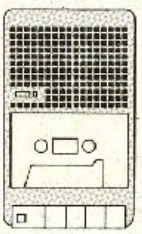


Das Fernsehgerät einschalten und die Lautstärke ganz leise drehen. Sie können den Fernseher jetzt so einstellen, daß er das Spectrum Signal empfängt.

Buchsen und Stecker am Spectrum

Strombuchse
Der vom ZX Netzteiltransformator erzeugte 9-Volt-Gleichstrom wird durch diese Buchse angeschlossen.

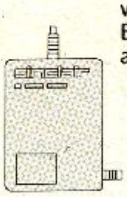
Kantenverbinder
Hier können eine breite Auswahl von Hardware, einschließlich Mikrodrives, Drucker und Modems angeschlossen werden

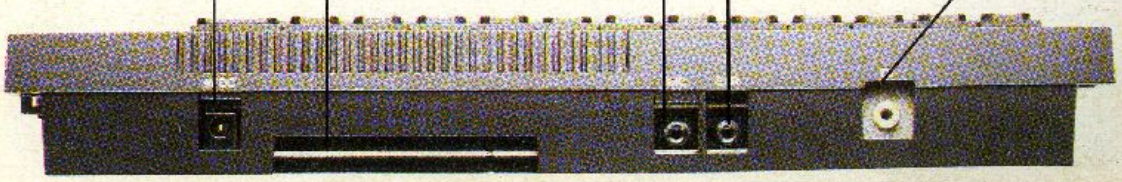


EAR-Buchse
Hier wird die Kopfhörerbuchse eines Kassettenspielers angeschlossen.

TV-Buchse
Hier wird die Antennenbuchse eines Fernsehgerätes angeschlossen.

MIC-Buchse
Hier wird die Mikrophonbuchse eines Kassettenspielers angeschlossen.

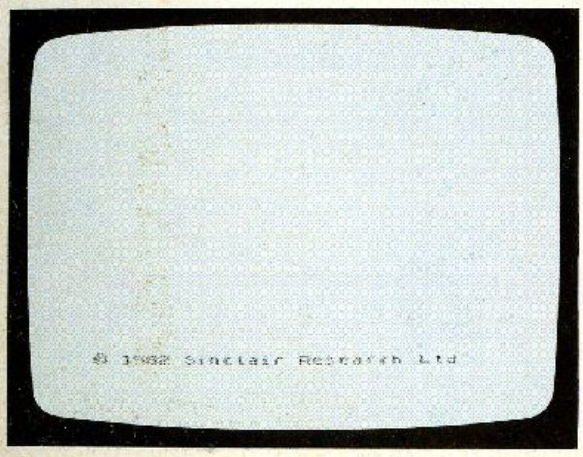


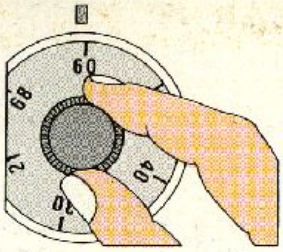


EINSTELLEN FERNSEHER

Ihr Spectrum liefert ein Farbfernseher-Videosignal bei der Frequenz von Kanal 36 auf UHF. Ihr Fernseher muß also auf diesen Kanal eingestellt werden, um das Computerbild zu zeigen.

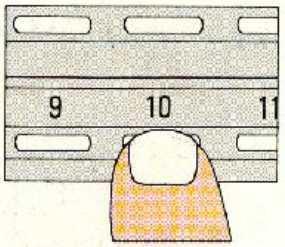
Wenn Sie Ihren Spectrum eingeschaltet haben und ihn in den Fernseher eingesteckt haben, den Einstellknopf am Fernseher so lange drehen, bis die Sinclair Copyright-Mitteilung, wie im Bildschirm unten dargestellt, erscheint. Können Sie diese Copyright-Mitteilung nicht erhalten oder sollten die Farben falsch aussehen, lesen Sie bitte die gegenüberliegende Tafel.

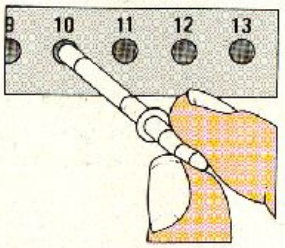




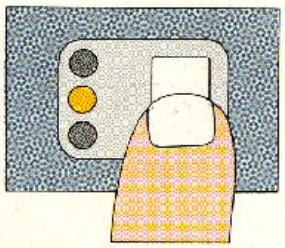
Einstellregler

Variable Einstellung
Ein variabler Regler kann jeden Kanal wählen. Einfach den Knopf drehen, bis die Copyright-Mitteilung erscheint.





Drucktasteneinstellung
Wählen Sie einen Einstellknopf aus, welcher für den Computer benutzt werden soll und stellen Sie ihn so ein, daß die Copyright-Mitteilung erscheint. Wenn möglich, einen nicht besetzten Knopf benutzen, dann brauchen Sie das Gerät nicht jedesmal neu einstellen, wenn Sie Ihren Spectrum benutzen wollen.

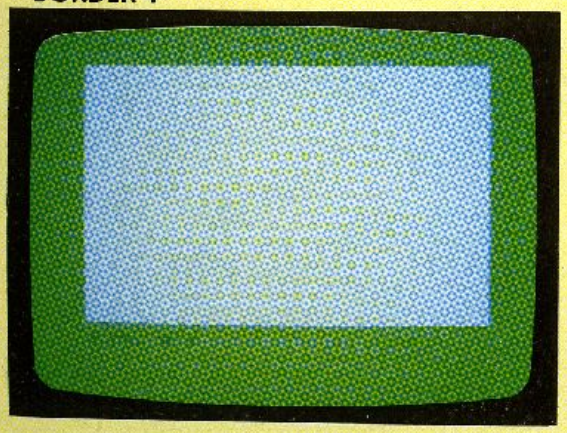


Elektronische Einstellung
Diese Einstellungmethode ist ähnlich wie die Druckknopfeinstellung, nur daß hierbei das Gerät selbst den gewünschten Kanal einstellt.

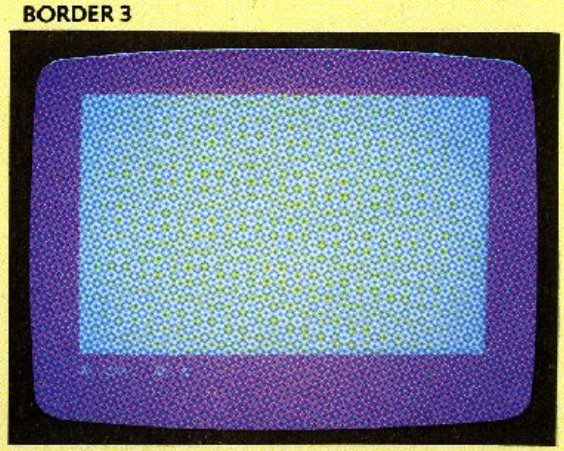
Wie man die Farben des Spectrum testet

Um die Farben des Spectrum zu testen, wird einfach die B-Taste gedrückt und eine Zahl von 1 bis 6. Die Copyright-Mitteilung verschwindet; zuerst erscheint das Wort BORDER und dann die Zahl. Jetzt die Taste mit der Aufschrift ENTER drücken. Der 'Border'-Bereich müßte jetzt die Farbe der Zahlentaste annehmen. Die untenstehenden Bildschirme zeigen, was passiert, wenn Sie BORDER4 und ENTER eintippen und dann BORDER3 und ENTER. BORDER7 macht die 'Border' (Umrandung) wieder weiß.

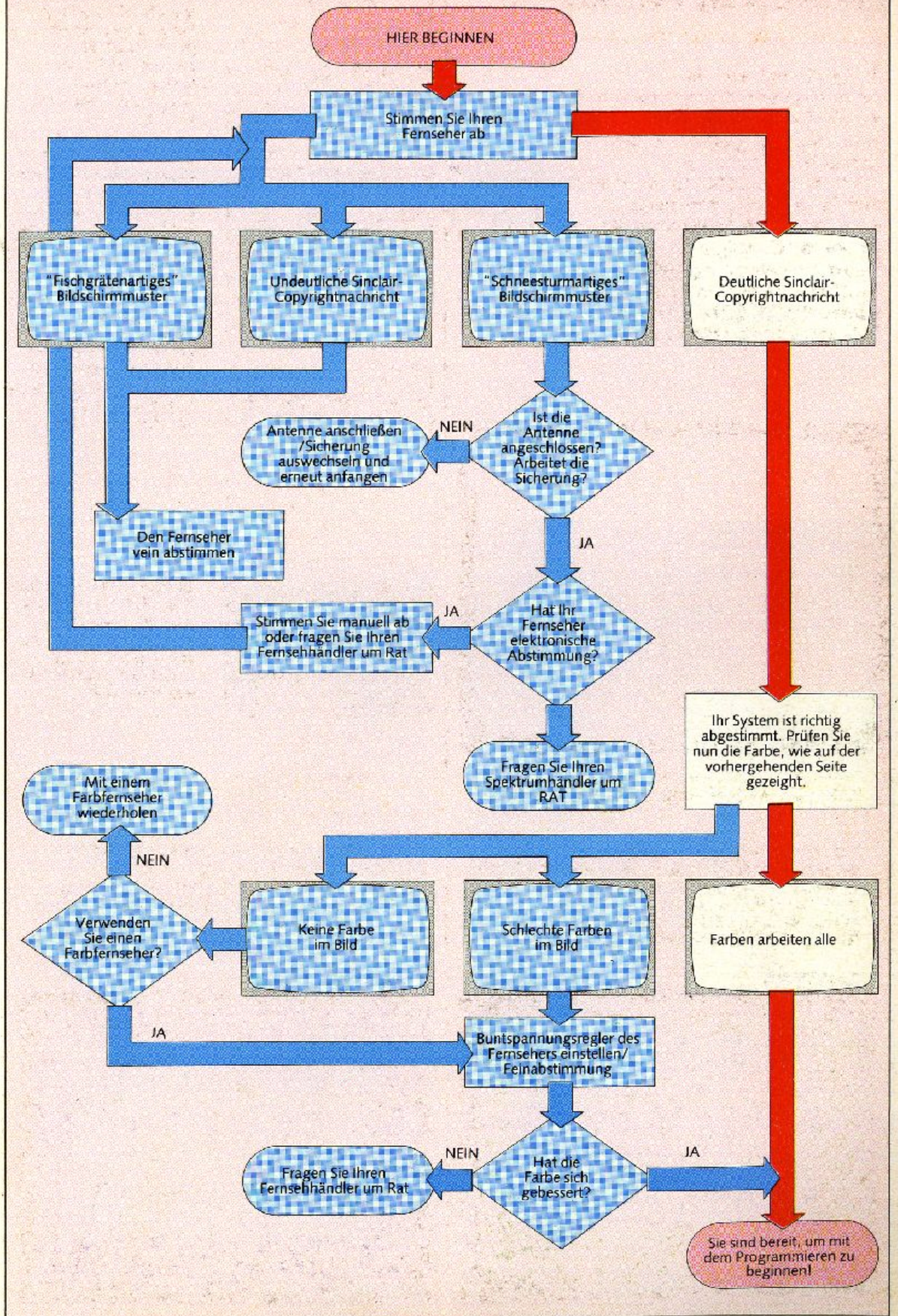
BORDER 4



BORDER 3



Aufstellungs-Fehlersuchplan



WAS IHR SPECTRUM + ALLES KANN

Zuerst experimentieren Sie!

Nachdem Ihr Spectrum jetzt eingeschaltet und Ihr Fernseher richtig eingestellt ist, drücken Sie einmal ein paar Tasten! Sie werden sehen, daß auf dem Bildschirm Wörter und Buchstaben und vielleicht auch einige Zahlen erscheinen.

Wenn Sie jedoch nicht wissen, wie man den Spectrum programmiert, ist es sehr unwahrscheinlich, daß der Computer hierauf reagiert. Aber keine Sorge, es kann nichts schiefgehen, ganz egal, welche Tasten Sie drücken.

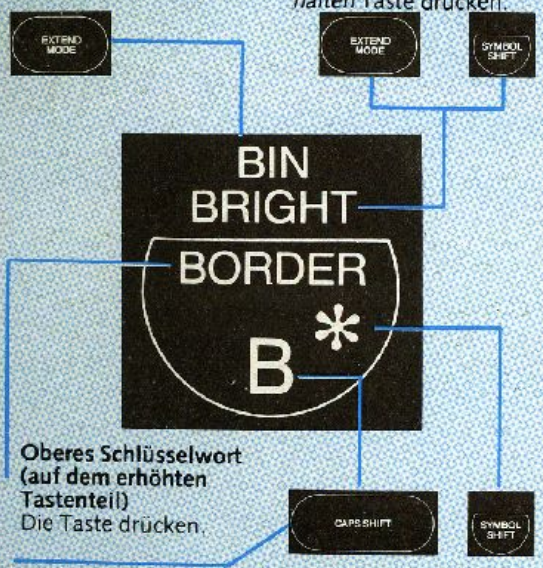
Jetzt bitte den Rückstellknopf auf der linken Seite des Computers drücken und nun können Sie den Computer arbeiten lassen.

Wie man eintastet

Um ein Wort, einen Buchstaben oder eine Zahl einzutasten, muß man sie zunächst einmal auf der Taste finden. Dann dieselbe Folge von Wahltasten benutzen, wie sie hier gezeigt ist.

Oberes Schlüsselwort
EXTEND MODE drücken und dann die Taste.

Unteres Schlüsselwort oder Zeichen
EXTEND MODE drücken, dann SYMBOL SHIFT *niedergedrückt halten* Taste drücken.



Oberes Schlüsselwort (auf dem erhöhten Tastenteil)
Die Taste drücken.

Buchstabe oder Zahl (erhöhter Teil der Taste)
Die Taste drücken. CAPS SHIFT für Großbuchstaben niedergedrückt festhalten.

Unteres Schlüsselwort oder Zeichen (erhöhter Tastenteil)
SYMBOL SHIFT niedergedrückt festhalten und die Taste drücken.

Als Nächstes programmieren Sie Ihren Spectrum!

Ihr Spectrum kann sehr viel tun, um ihn jedoch dazuzubringen, müssen Sie ihm einen Satz Anweisungen geben, welche Computerprogramm genannt werden.

Hier sind einige kurze Programme, die Ihren Spectrum dazu veranlassen, einzelne Schritte zu durchlaufen und seine Farben, Töne und Grafiken zu zeigen. Sie müssen nur einfach die Programmanweisungen *genauso* eintippen, wie sie hier stehen. Die Bildschirme zeigen Ihnen, was Sie sehen werden, wenn Sie jedoch die Tafel mit dem Titel 'Wie man ein Programm ändert' auf der gegenüberliegenden Seite lesen, können Sie selbst mit den Programmen experimentieren.

Wie man ein Programm eingibt und ablaufen läßt

Jeder Anweisungssatz wird in einer Liste aufgezeigt, welche 'Listing' genannt wird. Sie werden sehen, daß die Programm-Listings aus verschiedenen Abschnitten bestehen, die mit einer Zahl 10.20 und so weiter beginnen. Jeder

NAMEN

```
10 BORDER 1: INK RND*7
20 PAPER RND*7
30 PRINT " ZX Spectrum +";
40 GO TO 10
```

Der Name ZX Spectrum + erscheint in vielen Farben auf dem ganzen Schirm. Der Computer hält dann an und unten auf dem Bildschirm erscheint **'scroll?'** (scroll = abrollen). Damit das Display 'hochrollt', kann jede Taste außer N, SPACE, BREAK oder STOP gedrückt werden. Wenn Sie das 'Scrolling' anhalten und BREAK drücken, dann R(RUN) und dann ENTER, erscheinen die Namen in einem anderen Farbmuster.

Ein Versuch
Verändern Sie in Zeile 30 "ZX Spectrum +" in Ihren Namen um in Anführungsstrichen ("), zum Beispiel

```
30 PRINT "John    ";
```

Das Semikolon nicht vergessen (;). Ihr Name erscheint dann auf dem Bildschirm.

dieser Teile heißt 'line' (Zeile) des Programms.

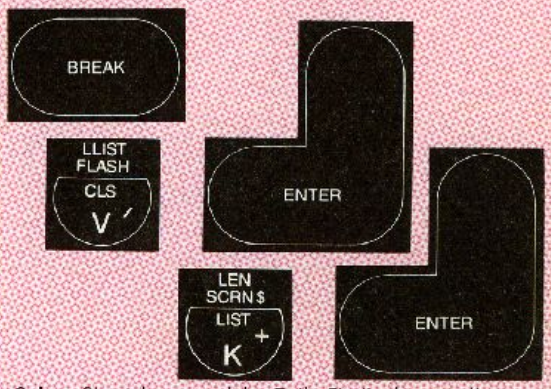
In jeder Programmzeile sehen Sie ganze Wörter oder Abkürzungen mit zwei oder mehr Buchstaben, wie zum Beispiel *PRINT, LET, RND, PI, PAPER* und *GOTO*. Diese heißen '*keywords*' (Schlüsselworte oder auch Kennworte) und Sie brauchen sie *nicht* Buchstabe um Buchstabe einzutippen, sondern nur einfach die Taste mit dem jeweiligen Schlüsselwort suchen (*PRINT* befindet sich zum Beispiel auf der P-Taste) und dann die Anweisungen in der Tafel 'Wie man eintastet' befolgen.

Wenn Sie eine Zeile eintasten, erscheint diese unten auf dem Bildschirm. Wenn Sie am Ende der Programmzeile angekommen sind, drücken Sie die Taste ENTER. Die Zeile erscheint jetzt oben auf dem Bildschirm. Nun tasten und geben Sie jede Zeile auf dieselbe Weise ein. Wenn Sie aus Versehen eine falsche Taste drücken, lesen Sie bitte den Abschnitt auf der nächsten Seite 'Wie man Fehler korrigiert'.

Wenn Sie alle Zeilen eingegeben haben, drücken Sie R. Das Schlüsselwort **RUN** erscheint. Jetzt drücken Sie ENTER und Ihr Spectrum tritt in Aktion und durchläuft das Programm.

Wie man ein Programm ändert

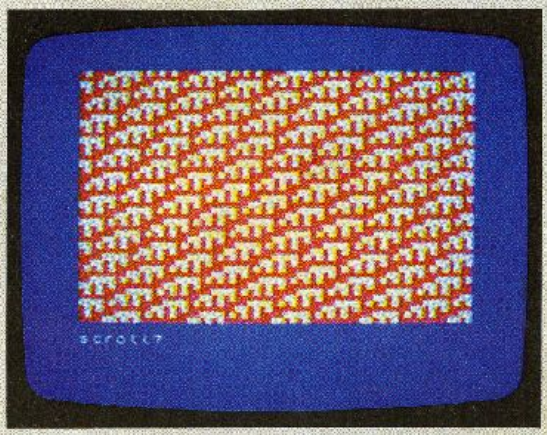
Warten, bis das Programm beendet ist oder das Programm durch Drücken von BREAK anhalten. Dann V(CLS) drücken, dann ENTER, gefolgt von K(LIST), dann ENTER. Das Programm-Listing (Liste der Zeilen) erscheint auf dem Bildschirm.



Sehen Sie sich an, welche Zeile Sie ändern wollen, tasten Sie dann die ganze neue Zeile ein, einschließlich Zeilennummer, und drücken Sie ENTER. Die neue Zeile ersetzt die alte. R(RUN) drücken und ENTER und das neue Programm fängt an.

MUSTER

```
10 LET a$=""
20 FOR x=1 TO 7
30 LET a$=a$+CHR$ (RND*14+129)
40 NEXT x
50 INK RND*7
60 BORDER RND*7
70 PRINT a$;
80 GO TO 70
```

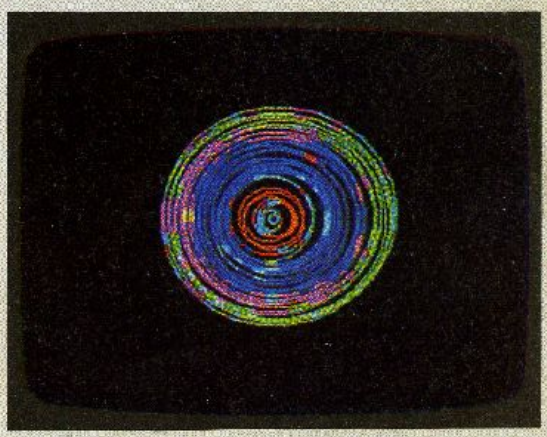


Ein geometrisches Farbmuster bildet sich auf dem Bildschirm, wenn Sie das Programm ablaufen lassen. Wenn der Bildschirm voll ist, hält das Display mit der Nachricht scroll? an. Um noch mehr von demselben Display zu sehen, kann jede Taste gedrückt werden außer N, SPACE, BREAK oder STOP, damit daß Muster sich nach oben bewegt. Um ein neues Muster in einer anderen Farbkombination zu sehen, wird N gedrückt, wenn die Nachricht scroll? erscheint. Dann BREAK drücken, gefolgt von R(RUN) und ENTER.

Ein Versuch
Ändern Sie in Zeile 20 die 7 in eine andere Zahl um, um ein anderes Muster zu erhalten. Versuchen Sie zum Beispiel 8.

BLINKENDE KREISE

```
 10 BORDER 0: PAPER 0: CLS
 20 CIRCLE INK RND*6: FLASH RND
;120+RND*8,80+RND*6,RND*80
 30 BEEP 0.1,RND*60
 40 IF RND>.9 THEN GO TO 60
 50 GO TO 20
 60 FOR y=-2 TO 4
 70 FOR x=0 TO 6
 80 BORDER x
 90 BEEP .05,x+y
100 NEXT x
110 NEXT y
120 RUN
```

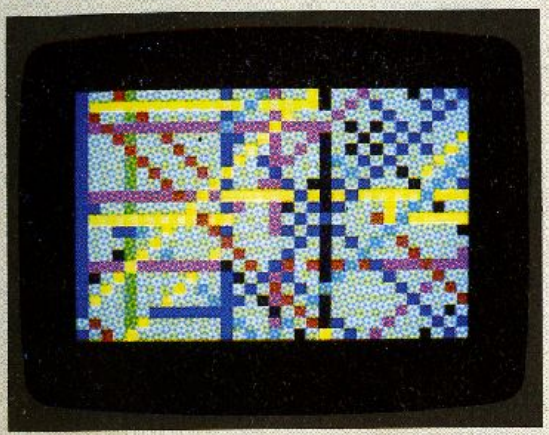


Eine Gruppe von fast konzentrischen, blinkenden Kreisen bildet sich auf dem Bildschirm in den verschiedenen Farben. Dann blinkt plötzlich die Umrandung, der Computer erzeugt ein Trillergeräusch und eine neue Gruppe von Kreisen erscheint.

Ein Versuch:
Vor der Auflistung des Programms (mit der K-Taste) tasten Sie PAPER 7 ein und drücken Sie ENTER. Dann wieder Zeile 20, ohne die zwei Schlüsselwörter FLASH RND; jetzt werden die Kreise nicht mehr blinken.

MOSAIK

```
  5 BORDER 0: CLS
 10 LET h=16: LET v=11
 20 LET x=INT (RND*3-1): LET y=
INT (RND*3-1)
 30 INK RND*7
 40 FOR z=1 TO 20
 50 PRINT AT v,h;CHR$ 143
 60 LET h=h+x
 70 LET v=v+y
 80 IF h<0 THEN LET h=31
 90 IF h>31 THEN LET h=0
100 IF v<0 THEN LET v=21
110 IF v>21 THEN LET v=0
120 NEXT z
130 GO TO 20
```



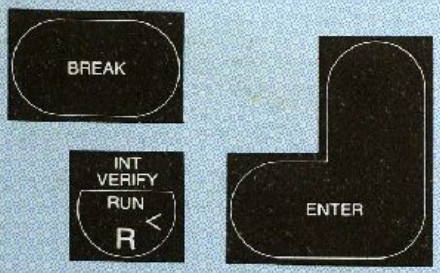
Ein farbiges Quadrat bewegt sich auf dem Bildschirm hin und her und formt so ein Farbmuster. Jedesmal, wenn Sie das Programm wieder anfangen, wird ein neues Muster erzeugt.

Ein Versuch
Verändern Sie in Zeile 50 143 um in 42 und Sie werden Sterne sehen! Versuchen Sie andere Zahlen von 33 bis 142. Lesen Sie in der Zeichensatztafel auf Seite 51 nach, was passieren wird.

Wie man ein Programm wieder neuanfängt

Einige dieser Programme, wie STARS AND STRIPES, gehen zu Ende und geben dann die Mitteilung 0 OK und die letzte Zeilennummer im Programm. Dies bedeutet, daß das ganze Programm zu Ende ist. Um wieder von vorne anzufangen, wird einfach nur R(RUN) gedrückt und dann ENTER.

Andere Programme laufen entweder kontinuierlich wie MOSAIK oder sie fangen automatisch wieder von vorne an wie SONNENAUFGANG. Um diese



Programme anzuhalten, wird BREAK gedrückt.

Halten Sie diese Taste niedergedrückt, bis das Programm anhält und die BREAK-Mitteilung erscheint. Um wieder anzufangen, nur R(RUN) und ENTER drücken.

Wie man Fehler korrigiert

Wenn Sie eine falsche Taste drücken oder die Umschalttaste oder EXTEND MODE nicht richtig anschlagen, machen Sie sich darüber keine Sorgen! Einfach die Taste DELETE anschlagen und das letzte Schlüsselwort, Zeichen, der letzte Buchstabe oder die letzte Zahl verschwindet. DELETE niedergedrückt

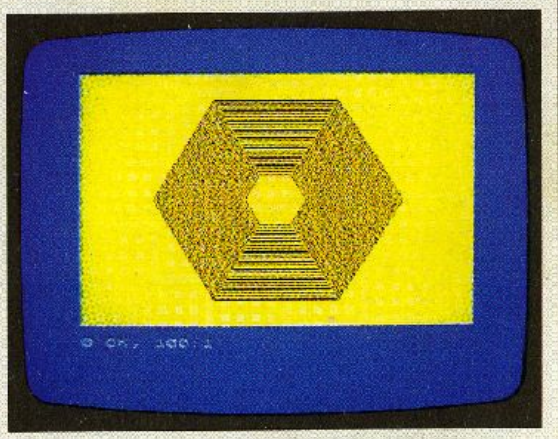
halten, um noch mehr zu löschen.

Wenn Sie einen Fehler gemacht haben, nachdem Sie ENTER gedrückt haben, kann ein blinkendes Fragezeichen in der Zeile erscheinen. Das ist die Stelle, an der Sie einen Fehler gemacht haben. Die Taste DELETE drücken, um die Zeile bis zum Fragezeichen zu löschen, dann die Zeile korrekt eingeben und ENTER drücken.

Wenn Sie eine unkorrekte Zeile eingeben, mag das Programm aufhören und zeigt eine Mitteilung, in der die Nummer der falschen Zeile unten auf dem Bildschirm erscheint. Die ganze Zeile korrekt eintasten, dann ENTER drücken, R(RUN) und ENTER. Jetzt müßte das Programm wieder funktionieren.

POLYHEDRA

```
  5 BORDER 1: PAPER 6: CLS
 10 INPUT n
 20 FOR r=20 TO 80 STEP 2
 25 LET x=128: LET y=87
 30 LET h1=x-r: LET v1=y
 35 PLOT h1,v1
 40 FOR a=0 TO 361 STEP 360/n
 50 LET h2=x-r*COS (a*PI/180)
 60 LET v2=y+r*SIN (a*PI/180)
 70 DRAW h2-h1,v2-v1
 80 LET h1=h2: LET v1=v2
 85 BEEP 0.02,r-20
 90 NEXT a
100 NEXT r
```

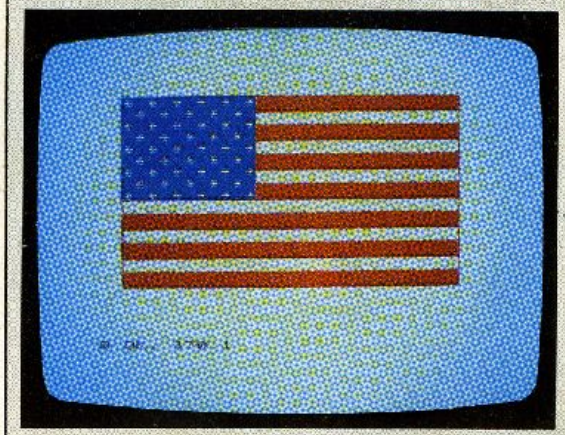


Zuerst sehen Sie einen leeren Bildschirm. Tasten Sie 6 ein und drücken Sie ENTER. Es bildet sich eine sechseitige Figur. Wenn das Programm zuende ist, fangen Sie es wieder von vorne an und tasten Sie eine andere Zahl ein, um eine Figur mit einer anderen Anzahl von Seiten zu erhalten.

Ein Versuch
Verändern Sie in Zeile 20 die 2 in eine andere Zahl. Das Muster bildet sich viel schneller, wenn die Zahl höher ist und die Polyhedra (vielseitige Figuren) sind weiter voneinander entfernt.

STARS AND STRIPES

```
 10 INK 2
 20 PAPER 7
 30 CLS
 40 FOR x=28 TO 148 STEP 20
 50 FOR z=0 TO 11
 60 PLOT 16,z+x: DRAW 216,0
 70 NEXT z
 80 NEXT x
 90 PLOT 16,28: DRAW 0,131
100 PLOT 232,28: DRAW 0,131
110 PAPER 1
120 INK 7
130 FOR x=2 TO 8 STEP 2
140 PRINT AT x,2;"* * * * * *"
150 PRINT AT x+1,2;" * * * * *
"
160 NEXT x
170 PRINT AT x,2,"* * * * * *"
```



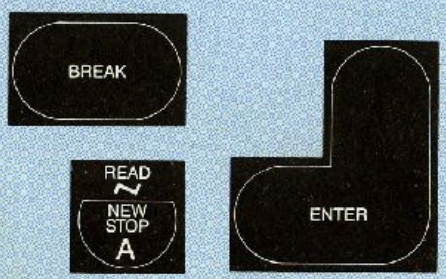
Die Flagge der Vereinigten Staaten erscheint auf dem Bildschirm.

Ein Versuch
Verändern Sie die Farbnummern der Fahne. Die Farbe der Streifen ist in Zeile 10, die Sterne bei Zeile 120 und der Hintergrund zu den Sternen bei Zeile 110.

Wie man ein neues Programm beginnt

Immer, wenn Sie mit einem Programm fertig sind und ein völlig neues Programm anfangen wollen, warten Sie, bis es aufhört oder halten Sie es an, indem Sie BREAK drücken.

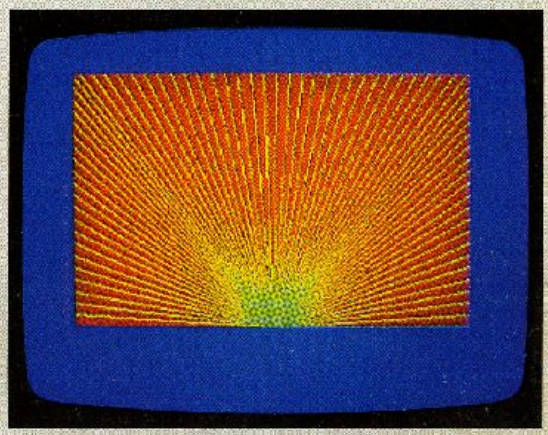
Sie haben dann zwei Möglichkeiten, das alte Programm aus dem Speicher des Computers zu löschen. Entweder Sie drücken zwei Tasten, A(NEW) und dann ENTER. Der Bildschirm wird einen Moment lang schwarz und dann wird die Copyright-Mitteilung erscheinen.



Die andere, leichtere Möglichkeit ist die, daß Sie einfach den 'reset' Knopf (Rückstellen) drücken. Dies hat dieselbe Wirkung, als wenn Sie den Spectrum durch den Netzstecker aus- und wieder einschalten.

SONNENAUFGANG

```
 10 BORDER RND*6
 20 INK RND*7
 30 PAPER RND*6
 40 CLS
 50 LET z=RND*10+2
 60 FOR x=0 TO 174 STEP z
 70 PLOT 128,0
 80 DRAW -128,x
 90 BEEP .01,x/3
100 NEXT x
110 FOR x=-127 TO 127 STEP z
120 PLOT 128,0
130 DRAW x,175
140 BEEP .01,60
150 NEXT x
160 FOR x=174 TO 0 STEP -z
170 PLOT 128,0
180 DRAW 127,x
190 BEEP .01,x/3
200 NEXT x
210 PAUSE 200
220 GO TO 10
```



Alle paar Sekunden bildet sich ein Bild wie ein glänzender Sonnenaufgang in verschiedenen Farben. Wenn der Bildschirm plötzlich leer ist, warten Sie einen Moment. Es kommt bald wieder ein neuer Sonnenaufgang!

Ein Versuch
Verändern Sie in Zeile 210 die 200 in eine andere Zahl um, um die Zeit zu verändern, die jeder Sonnenaufgang auf dem Bildschirm bleibt. 200 entspricht 4 Sekunden.

Was kommt als Nächstes?

Jetzt haben Sie die Wahl. Wenn Sie irdendwelche dieser Programme behalten möchten, um sie später wieder zu fahren, können Sie sie auf Kassettenbändern aufnehmen. Lesen Sie **'Wie man seine eigenen Programme sichert'** auf Seite 38.

Wenn Sie weiterhin mit Ihrem Spectrum experimentieren möchten, lesen Sie Kapitel 2. über **DAS PROGRAMMIEREN.** Bisher haben Sie nur Programme ausprobiert, ohne unbedingt zu verstehen, wie sie funktionieren. Kapitel 2. erklärt einige der Merkmale beim Programmieren eines Spectrums.

Wenn Sie einige Software-Bänder ausprobieren möchten, lesen Sie die nächste Seite über die Benutzung von fertiger Software.

ÜBER DIE BENUTZUNG VON FERTIGER SOFTWARE

Wenn Sie ein Programm in den Spectrum eingeben, erzeugen Sie eine Folge von elektronisch kodierten Signalen beim Anschlagen der Tasten. Die Codes gehen zum Speicher des Spectrum, welcher sie speichert, so daß der Computer sie beim Fahren des Programms verwenden kann. Die Codes bleiben im Speicher, bis Sie sie entweder entfernen (durch Eingabe von NEW oder Drücken des Rückstellknopfes, zum Beispiel) oder Ihren Spectrum abschalten.

Es ist jedoch nicht immer nötig, ein Programm einzutasten, wenn Sie Ihren Spectrum benutzen möchten. Stattdessen können Sie fertige Software kaufen, die Programme enthält, welche direkt und automatisch in den Computer gegeben werden können. Die Benutzung von fertiger Software erspart Ihnen nicht nur viel Arbeit, sondern ermöglicht es Ihnen außerdem, eine benutzungsbereite Programmbibliothek zu besitzen, deren Selbstschreiben Tage oder sogar Wochen dauern würde. Die Hersteller von Software erzeugen alle möglichen Arten von Programmen, welche von den besten Programmierern geschrieben werden, und für den Spectrum steht eine breite Auswahl zur Verfügung. Sehen Sie sich den Sinclair Spectrum Software-Katalog an, um eine Vorstellung davon zu bekommen, was für Programme Sie benutzen können und welche Ihnen Spaß machen. Sie können dann, wann immer Sie wollen, ein Programm nach Ihrer Wahl fahren.

Wie Programme in den Spectrum geladen werden

Die Code-Signale auf einem Softwareband bestehen aus hohen und tiefen Pieptönen, die mit einer Geschwindigkeit von 1500 pro Sekunde aufgenommen werden. Wenn Sie ein Softwareband auf dem Kassettenrecorder abspielen, erzeugt das Gerät die Folge von Tönen, die das Programm bilden. Schließen Sie den Kassettenspielder an den Spectrum und die Codes gehen direkt in den Spectrum-Speicher. Dies nennt man das '*Laden*' eines Programmes.

Auf diesen zwei Seiten können Sie sehen, wie Sie Ihren Kassettenspieler anschließen. Seiten 14–15 erklären seine Benutzung.

Fragen und Antworten zur Software

Was bedeutet das Wort 'Software'?
Software ist die allgemeine Bezeichnung für Programme, die zum Betreiben von Computern in sie eingegeben werden. Das Wort 'Hardware' bezeichnet die Ausrüstung selbst – die Computer und alle anderen damit verbundenen Geräte.

Warum wird Software auf Kassettenbändern produziert?
Kassettenbänder sind leicht zu benutzen und erfordern keine spezielle Ausrüstung. Um diese Art von Software zu laden, brauchen Sie nichts weiter als einen billigen Kassettenspieler.

Wie hören sich auf Band aufgenommene Programme an?
Spielen Sie einmal ein Band auf Ihrem Kassettenrecorder, ohne ihne an den Spectrum anzuschließen. Sie werden ein hohes Kreischgeräusch hören. Dies wird verursacht durch die Codesignale, die zum Lautsprecher des Kassettenrecorders statt zu dem des Computers gehen. Die Signale werden von der Kassette an den Spectrum mit so hoher Geschwindigkeit gesandt, daß man keine einzelnen Töne unterscheiden kann.

Gibt es noch andere Arten von Software?
Ja. Sie können Programme auf ROM-Tonabnehmern anstatt Kassettenbändern erhalten. Die Tonabnehmer lassen sich in eine Schnittstelle einstecken, welche hinten in Ihren Spectrum paßt. Ein Programm auf einem ROM-Tonabnehmer wird sofort geladen, es gibt kein Warten.

Außerdem gibt es Software auf Microdrive Disketten. Diese enthalten Programme, welche in magnetischer Form wie bei Bändern aufgenommen worden sind. Eine Diskette kann mehrere Programme enthalten und jedes beliebige Programm kann innerhalb von Sekunden statt Minuten geladen werden – anders als bei Kassettenbändern. Microdrive Tonabnehmer werden in Verbindung mit dem Microantriebsgerät verwendet (siehe Seite 46).

Welches ist der beste Kassettenspieler?
Der Spectrum ist mit einem billigen, tragbaren Kassettenspieler durchaus zufrieden, der besser an das Stromnetz angeschlossen ist als durch Batterie betrieben wird. Das Gerät sollte einen eigenen Lautstärkenregler haben, ein Tonregler ist jedoch nicht wichtig. Es gibt auch spezielle Kassettengeräte für Computer. Diese sind so ausgelegt, daß sie Programme zuverlässiger als gewöhnliche Kassettengeräte laden und speichern.

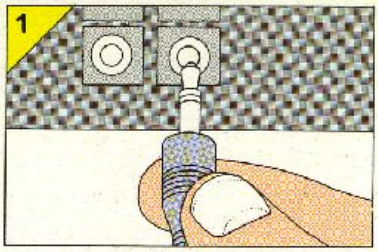
Brauchen auf Band aufgenommene Programme besondere Pflege?
Wie jede Art magnetischer Speicherung können Programme auf Kassetten durch starke magnetische Felder unterbrochen werden. Heben Sie Ihre Kassetten nicht in der Nähe von Geräten auf, die starken elektrischen Strom benötigen. Außerdem müssen Software-Kassetten möglichst staubfrei gehalten werden.

Funktioniert jede Art von Software?
Nein. Nur die für den ZX Spectrum oder ZX Spectrum + produzierte Software kann man laden.

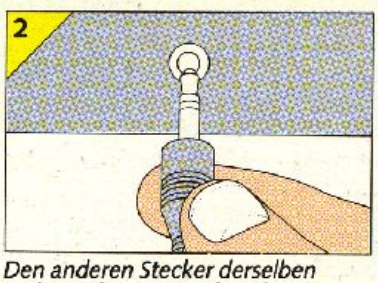
Das Anschließen Ihres Kassettenspielers

Das mit Ihrem Spectrum mitgelieferte Kassettenkabel ist für den Anschluß an Ihren Kassettenspieler gedacht. Das ist das Kabel mit den zwei kleinen Steckern an jedem Ende. Den Kassettenspieler neben den Spectrum stellen und das Kabel wie dargestellt einstecken. Kassettengerät und Spectrum können hierbei ein- oder ausgeschaltet sein, es ist jedoch besser, die Kassette aus dem Gerät herauszunehmen, bevor es ein-oder ausgeschaltet wird. Hierdurch werden die gespeicherten Programme geschützt.

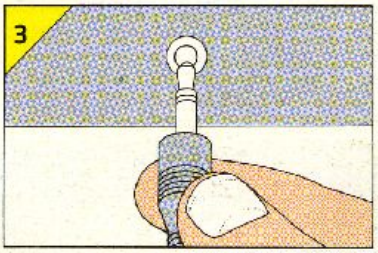
Die richtigen Anschlüsse



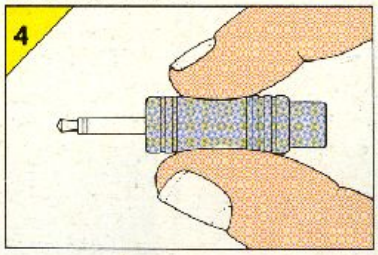
Einen der vier Stecker in die EAR-Buchse hinten am Spectrum stecken.



Den anderen Stecker derselben Farbe in die EAR-Buchse des Kassettengerätes stecken, sofern eine vorhanden ist.



Besitzt das Kassettengerät keine EAR-Buchse, den Stecker an die Kopfhörerbuchse anschließen, sofern eine vorhanden ist. Sonst an eine äußere Lautsprecherbuchse anschließen.



Wenn der Stecker am Kassettenkabel nicht in die Buchse am Kassettengerät paßt, brauchen Sie einen Adapter oder ein Spezialkabel mit den richtigen Steckern von Ihrem Elektrohändler. Die Spectrum EAR-Buchse braucht einen 3,5mm Klinkenstecker und ein Eingabesignal von ungefähr 1 Volt.

EAR- und MIC-BUCHSE
Beim Laden von Programmen können sowohl EAR- wie auch MIC-Buchse angeschlossen sein, wie hier dargestellt. Beim Sichern von Programmen (siehe Seite 38) *muß* das EAR-Kabel jedoch entfernt werden.

Software-Tips

- Das Kassettenkabel des Spectrum hat farblich gekennzeichnete Stecker, um Kreuzverbindungen zwischen Computer und Kassettengerät zu vermeiden. Wenn Sie einen Kassettenspieler mit dem Spectrum verwenden, halten Sie sich immer an das gleiche System, eine Farbe für EAR-Asnchlüsse, die andere für MIC-Anschlüsse.
- Einige Kassettenspieler werden von anderen nahestehenden elektrischen Geräten beeinflußt. Dieses kann die Signale zwischen Computer und Kassettenspieler stören, so daß das Programm nicht richtig geladen wird. Falls Ihr Kassettengerät manchmal nicht richtig arbeitet, versuchen Sie, es weiter entfernt vom Fernseher oder Computer zu benutzen.

WIE MAN EIN PROGRAMM LÄDT

Nachdem Sie jetzt Ihren Kassettenspieler an Ihren Spectrum angeschlossen haben, können Sie ein Programm laden und fahren. Sie können ein fertiges Software-Band verwenden oder Ihr eigenes Band mit Ihren eigenen Programmen. Der Vorgang ist in beiden Fällen gleich.

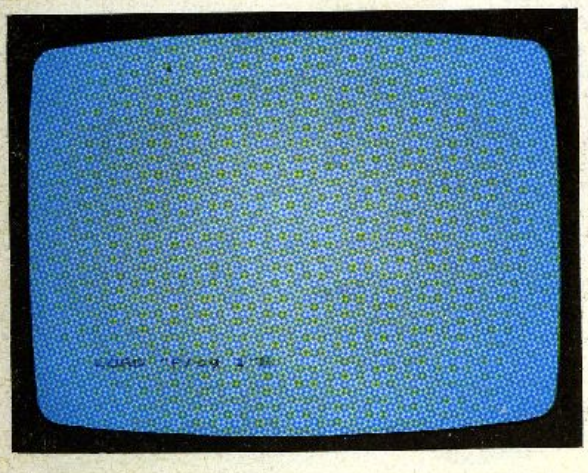
Den Kassettenspieler einschalten. Sicherstellen, daß der Spectrum auch eingeschaltet ist (Stecker in der Steckdose), dann die Kassette in den Kassettenspieler einlegen. Befindet sich bereits ein Programm im Computer, warten Sie, bis es zu Ende ist oder halten Sie es durch ein Anschlagen der BREAK-Taste an. Sie können nun NEW eingeben oder den Rückstellknopf drücken, um das Programm aus dem Speicher des Spectrum zu entfernen, aber dies ist nicht wichtig, da das Laden eines Programmes zunächst den Speicher löscht.

Folgen Sie jetzt den numerierten Anweisungen. Falls etwas schiefgeht, lesen Sie Seite 16 über die *Fehlersuche beim Laden von Software*.

1 Kassette einlegen und auf Bandbeginn einstellen.

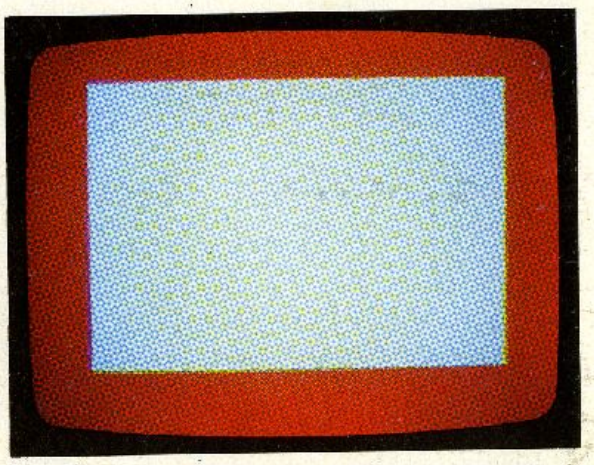
2 Lautstärken- und Tonregler am Kassettengerät auf den erforderlichen Stand einstellen. Versuchen Sie eine Lautstärke von ungefähr zwei Dritteln der Höchsteinstellung und steht eine Tonregelung zur Verfügung, stellen Sie sie auf maximale Höhe.

3 Die Taste J drücken und LOAD erscheint auf dem Bildschirm. Dann den Programmnamen in Anführungsstrichen eintasten, zum Beispiel LOAD "Prog 1".



4 ENTER drücken. Der Bildschirm ist jetzt leer.

5 Das Band beginnen. Die Umrandung des Schirms müßte jetzt rot oder blau werden oder rot und blau blinken. Dies bedeutet, daß der Spectrum ein Programm sucht.

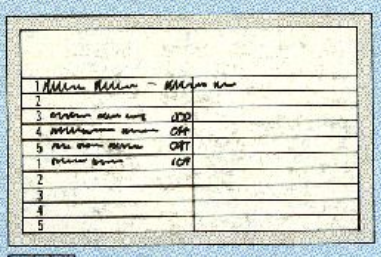


6 Nach einigen Sekunden müßten rote und blaue Streifen sich die Børder hinauf- oder herunterbewegen. Dies bedeutet, daß der Spectrum angefangen hat, ein Signal zu empfangen.

Tips zum Laden der Software

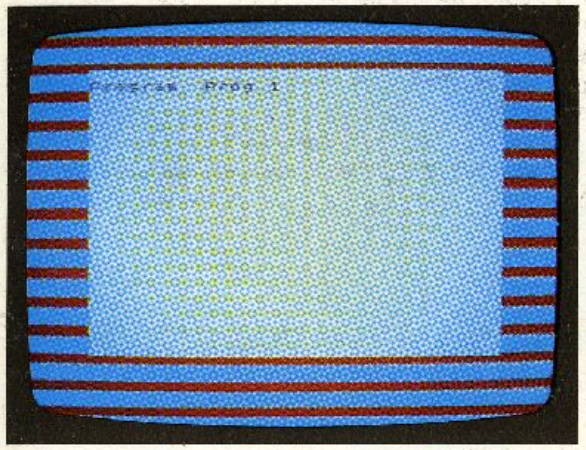
Hier sind einige Tips zur Zeitersparnis beim Laden:

1 Alle Bänder deutlich etikettieren, so daß Sie Programme leicht finden können. Wenn ein Band mehr als ein Programm enthält, schreiben Sie die Namen der Programme der Reihe nach auf das Etikett. Denken Sie daran, daß Sie den Namen genauso buchstabieren, wie der Computer ihn benutzen wird.



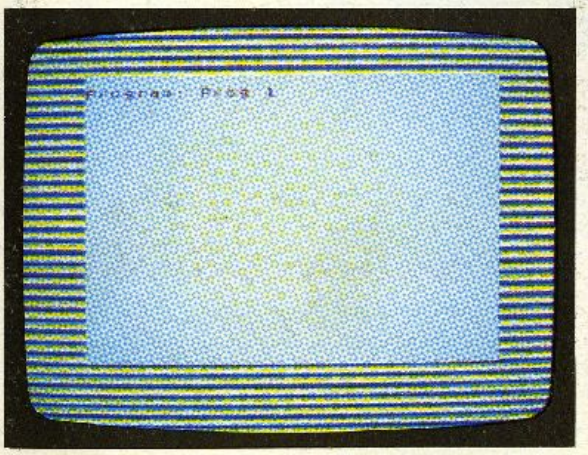
2 Wenn Ihr Kassettengerät einen Zähler hat, benutzen Sie ihn, um ein Programm schneller zu finden zu können. Den Zähler am Anfang des Bandes auf Null stellen, dann LOAD eingeben, gefolgt von einem Programmnamen (in Anführungsstrichen), welcher nicht auf dem Band ist. Das Band spielen und der Spectrum wird jedes Programm nennen, das er findet, ohne es zu laden. Die Zählernummern auf das Etikett neben die Programmnamen schreiben. Später können Sie dann Ihre Programme schnell und sicher finden.

7 Das Wort **Program**: gefolgt von dem Programmnamen, oder **Bytes**: gefolgt von einem Namen oder Buchstaben, erscheint auf dem Schirm. Dies bedeutet, daß der Computer das Programm gefunden hat.



8 Die roten und blauen Streifen treten wieder auf, während der Computer darauf wartet, das Programm zu laden.

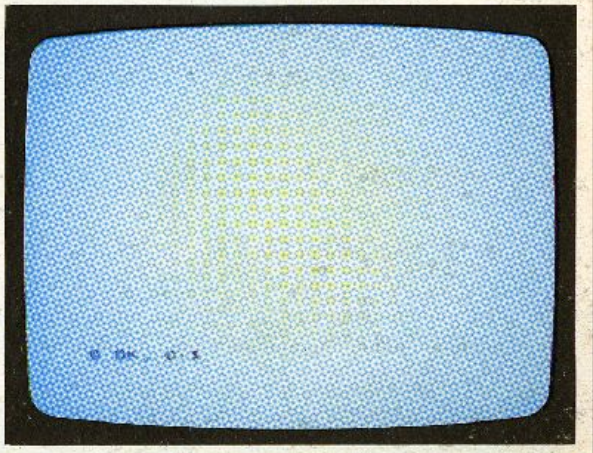
9 In der Umrandung erscheint ein Muster von gelben und blauen Linien. Dies bedeutet, daß der Spectrum das Programm lädt. Das Laden kann einige Minuten dauern, wenn das Programm sehr lang ist.



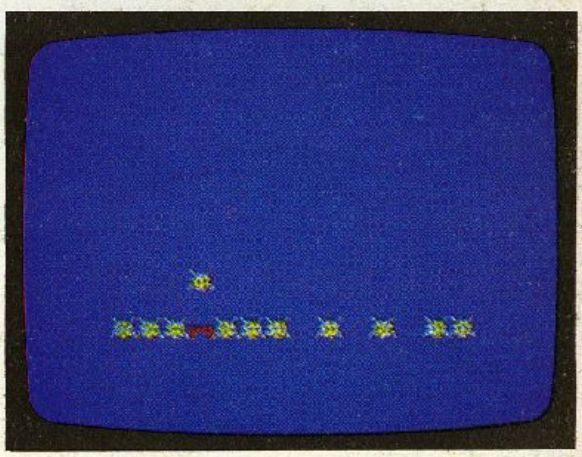
10 Vorgänge 7, 8 und 9 können ein oder mehrere Male wiederholt werden, wenn das Programm in Sektionen aufgeteilt ist.

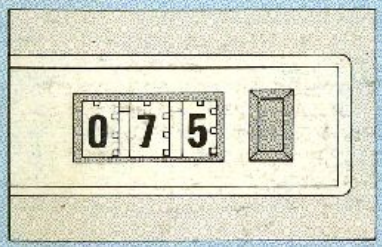
11 Das Programm kann automatisch beginnen, wenn es geladen ist. Das Band anhalten.

12 Wenn das Programm nicht automatisch anfängt, wenn es geladen ist, wird der Bildschirm leer und die Mitteilung **0 OK, 0:1** erscheint, das Band anhalten.



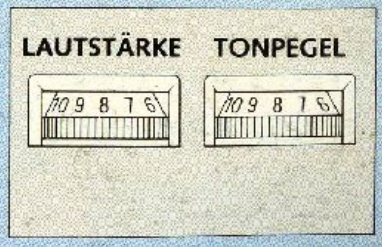
13 R(RUN) drücken und ENTER und das Programm fängt an.



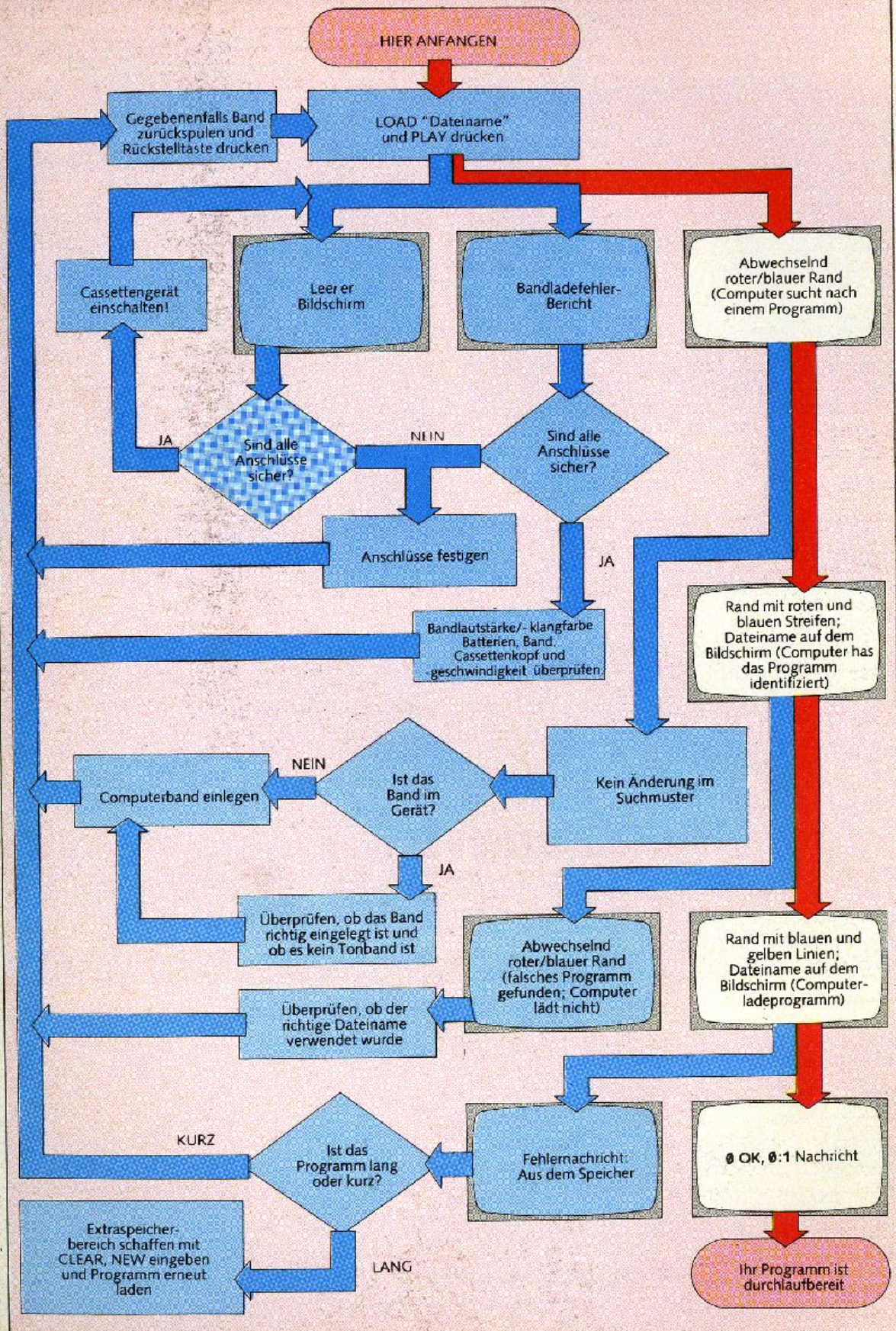


3 Wenn sich das Band beim richtigen Programm befindet oder wenn Sie den Namen des Programmes nicht kennen, geben Sie LOAD "" ein anstatt **LOAD** gefolgt von dem Namen in Anführungsstrichen. Zwischen den beiden Anführungsstrichen darf keine Leerstelle sein. Ihr Spectrum wird dann das erste Programm laden, das es auf dem Band findet. Wenn der erscheinende Programmname nicht der ist, den Sie wollen, drücken Sie BREAK, lassen Sie das Band weiterlaufen und versuchen Sie es wieder.

4 Notieren Sie sich Lautstärken – und Tonpegel, bei dem Ihr Spectrum laden kann. Stellen Sie diese Pegel vor dem Laden am Kassettengerät ein.



Softwareladungs-Fehlersuchplan



PROGRAMMIEREN

Dieses Kapitel dient als Einführung in das Programmieren des ZX Spectrum +. Es enthält alle Erklärungen zur Tastatur, so daß Sie verstehen, wie Sie Ihren Spectrum benutzen können. Die kurzen Versuchsprogramme konzentrieren sich vor allem auf die besonderen Merkmale des Spectrum, so daß Sie beim Schreiben Ihrer Programme das Potential Ihres Computers voll nutzen können.

2

DIE TASTATUR – DAS BEDIENUNGSFELD IHRES COMPUTERS

Der ZX Spectrum + hat eine eigene Sprache, die unter dem Namen BASIC bekannte Computersprache. Wenn Sie wollen, daß der Computer Ihre Anweisungen befolgt, müssen Sie mit ihm BASIC sprechen. Sie tun dies durch die Benutzung der Tastatur. Darüberhinaus können Sie durch die Tastatur den Computer steuern, während er Ihre Programme fährt.

Die Version von BASIC, die der Spectrum versteht, ist eine einfache, aber sehr effektive Form dieser Sprache. Darüberhinaus hat der Spectrum eine hervorragende Eigenschaft, die das Programmieren sehr erleichtert und das ist das Eingabesystem eines Schlüsselwortes durch eine einzige Taste.

Tasten und Schlüsselwörter

Schlüsselwörter sind besondere Wörter in BASIC, die den Computer anweisen, etwas zu tun – Wörter wie zum Beispiel PRINT oder INPUT. Bei den meisten Computern müssen Sie jeden

GRAPH
Diese Taste wird benutzt, um die Figuren oder Grafikzeichen auf Tasten 1 bis 8 zu wählen. Wenn Sie diese Taste drücken und dann eine Zahlentaste mit oder ohne die CAPS SHIFT Taste, erscheint ein Grafikzeichen auf dem Bildschirm. Sie müssen GRAPH immer noch einmal drücken, wenn der Computer wirder zum Normalbetrieb übergehen soll.

TRUE VIDEO und INV VIDEO
Diese Tasten geben Steuercodes in die Programmzeilen ein, um normale oder umgekehrte Farben zu erzeugen.

NEW
Diese Taste löscht den BASIC Speicherbereich des Computers und löscht alle darin enthaltenen Programme.

DELETE
Diese Taste wird benutzt, wenn Sie eine falsche Taste drücken und ein Schlüsselwort, einen Buchstaben, eine Zahl oder ein Zeichen entfernen wollen – siehe Seite 10.

EDIT
Diese Taste wird benutzt, um eine Zeile in einem Programm zu ändern, ohne sie völlig neu zu schreiben – siehe Seite 21.

EXTEND MODE
Diese Taste wählt das obere Schlüsselwort über der Tastenerhölung auf jeder Taste. Wenn gefolgt von SYMBOL SHIFT und einer Taste, wählt sie das Zeichen oder Schlüsselwort direkt über dem erhöhten Teil der Taste – siehe Seiten 20–21.

CAPS SHIFT
Drücken Sie diese Taste mit einer Buchstabentaste, um einen Großbuchstaben zu erhalten. Wenn Sie mehrere Großbuchstaben wollen, drücken Sie CAPS LOCK.

CAPS LOCK
Benutzen Sie diese Taste, wenn Sie immerzu Großbuchstaben wünschen. Drücken Sie sie noch einmal, wenn Sie wieder kleine Buchstaben möchten.

BEEP
Diese Taste erzeugt das Schlüsselwort, welches den Ton-Synthesizer des Spectrum steuert.

Buchstaben des Schlüsselwortes einzeln eintasten wie bei einer Schreibmaschine, und Sie müssen jedes Wort hundertprozentig korrekt buchstabieren. Beim Spectrum brauchen Sie nur eine einzige Taste zu drücken, um ein ganzes Schlüsselwort zu erhalten.

Sinclair BASIC hat über 80 Schlüsselwörter, die zugänglich sind über 36 Tasten, 26 Buchstaben-Tasten und 10-Zahlentasten. Viele Tasten liefern nicht nur ein, sondern mehrere Schlüsselwörter *und* einen Buchstaben, eine Zahl, ein Zeichen oder sogar eine Figur (Grafikzeichen), von denen alle in Programmen benutzt werden können.

Das Wählen von Schlüsselwörten und Zeichen

Auf der Spectrum-Tastatur befinden sich zwei Tasten, die Sie sehr viel benutzen werden, nämlich EXTEND MODE und SYMBOL SHIFT; dieses sind die Tasten, mit denen Sie wählen können, welche der Schlüsselwörter oder Zeichen auf den anderen Tasten Sie auf dem Bildschirm haben wollen. Nachdem Sie sich mit dem Layout der Tastatur vertraut gemacht haben, zeigen Ihnen die nächsten zwei Seiten genau, was Sie auf Ihrer Tastatur finden. Danach können Sie anfangen, Ihre eigenen Programme zu schreiben.

Farb-display-Tasten
Diese sechs Tasten erzeugen Schlüsselwörter, die steuern, wie der Spectrum Farben auf dem Bildschirm darstellt.

Zahlentasten
Diese Tasten erzeugen nicht nur Zahlen, sondern fügen auch Steuercodes in Programme ein für die aufgezeigten Farben – siehe Seite 33. Die sich direkt über den Tastenerhöhungen befindlichen Schlüsselwörter von 4 bis 0, außer Taste 8, werden nur mit ZX Microantrieben benutzt.

BREAK
Diese Taste hält ein Programm an. Sie löscht nicht das Programm aus dem Computerspeicher.

ENTER
Diese Taste wird gedrückt, damit eine Programmzeile in den Spectrum Speicher eingegeben wird. Diese Taste wird außerdem oft dazu benutzt, dem Computer während eines Programmes Informationen zuzuführen.

SYMBOL SHIFT
Diese Taste niedergedrückt halten und eine Buchstaben- oder Zahlentaste drücken, um das untere Schlüsselwort oder Zeichen auf der Tastenerhöhung zu wählen. Wenn diese Taste nach EXTEND MODE benutzt wird, wählt sie das sich unmittelbar über erhöhten Teil der Taste befindliche Symbol oder Schlüsselwort-siehe Seiten 20–21.

Leertaste
Diese erzeugt eine Leerstelle wie bei der Schreibmaschine.

Cursor-Tasten
Ein Anschlagen dieser Tasten bewegt den Cursor in dieselbe Richtung wie die Pfeile. Diese Tasten werden in Programmen oft benutzt, um die Bewegung von Figuren auf dem Bildschirm zu steuern. Sie werden außerdem beim Aufbereiten von Programmen verwendet.

ZUR BENUTZUNG DER TASTEN

Von den meisten Tasten auf Ihrem ZX Spectrum + können Sie sechs verschiedene Schlüsselwörter, Buchstaben, Zahlen oder Zeichen erhalten. Das Wählen eines Zeichens oder eines Schlüsselwortes auf der Tastatur ist jedoch nicht kompliziert, wenn Sie erst einmal mit einer besonderen Eigenschaft des Spectrum vertraut werden. Wenn Sie eine Taste drücken, hängt das Ergebnis, das Sie auf dem Bildschirm sehen, vom *Modus* ab, in dem sich der Computer in dem Moment befindet. Durch die verschiedenen Modi können Sie verschiedene Arten von Informationen eintippen, wie Schlüsselwörter, Buchstaben oder Grafikzeichen. Der Vorteil hiervon ist, daß, während Sie die Tastatur betätigen, der Spectrum Ihnen hilft, Tastatur-Modi zu wählen, so daß Sie Anweisungen und Informationen in der richtigen Reihenfolge eingeben.

Schlüsselwort-Modus

Ihren Spectrum einschalten oder zurückstellen, so daß die Copyright Mitteilung erscheint. Nun ENTER drücken. Ein blinkendes K erscheint in der linken unteren Ecke. Dieses blinkende Quadrat heißt '*Cursor*'. Er zeigt Ihnen, wo etwas auf dem Bildschirm erscheinen wird und das K gibt an, daß sich der Computer im '*Keyword-Mode*' befindet (Schlüsselwort-Modus). Drücken Sie eine beliebige Buchstabentaste und das obere Schlüsselwort auf dem erhöhten Teil der Taste erscheint auf dem Bildschirm. Versuchen Sie zum Beispiel Q und das Schlüsselwort **PLOT** erscheint. Drücken Sie die DELETE-Taste, um das Schlüsselwort zu entfernen, und versuchen Sie andere Tasten. Zahlentasten ergeben Zahlen, aber sobald Sie eine Buchstabentaste drücken, erscheint das obere Schlüsselwort auf dem erhöhten Tastenteil.

Drücken Sie DELETE noch einmal, so daß der K-Cursor wieder erscheint. Jetzt eine SYMBOL SHIFT Taste drücken, sie niedergedrückt halten und irgendeine Buchstabentaste drücken. Dieses Mal erscheint das Schlüsselwort oder Zeichen über dem Buchstaben auf dem erhöhten Tastenteil. Bei einer Zahlentaste erscheint das Zeichen rechts neben dem erhöhten Teil.

Wie man ein Schlüsselwort, Symbol oder Zeichen wählt

Hier können Sie sehen, wie man ein Schlüsselwort, Symbol oder Zeichen auf einer Buchstaben- oder Zahlentaste wählt. Wenn man eine Tastenfunktion wählt, muß man sich merken, wo diese auf der Taste ist und dann mit Hilfe der zwei Beispieltasten hier entscheiden, welche anderen Tasten man unter Umständen noch braucht, um zum richtigen Modus zu wechseln.

Buchstabentaste

Keyword (K)-Modus (Schlüsselwort-Modus)

nur eintasten	BORDER
SYMBOL SHIFT und Taste	*

Extended (E)-Modus (erweiterter Modus)

EXTEND MODE dann nur eintasten	BIN
EXTEND MODE dann SYMBOL SHIFT und Taste	BRIGHT

Letter (L)-Modus (Buchstaben-Modus)

nur eintasten	b
CAPS SHIFT und Taste	B
SYMBOL SHIFT und Taste	*

Capitals (C)-Modus (Großbuchstaben-Modus)

CAPS LOCK dann nur eintasten	B
CAPS LOCK dann SYMBOL SHIFT und Taste	*

Graphics(G)-Modus (Grafik-Modus)

GRAPH dann Tasten A bis U vom Benutzer wählbares Grafikzeichen

Zahlentaste

Keyword (K)-Modus

nur eintasten	3
SYMBOL SHIFT und Taste	#

Extended (E)-Modus

EXTEND MODE dann nur eintasten	violett Farbsteuerungscode
EXTEND MODE dann SYMBOL SHIFT und Taste	LINE

Letter (L)-Modus

nur eintasten	3
SYMBOL SHIFT und Taste	#

Capitals (C)-Modus

CAPS LOCK dann nur eintasten	3
CAPS LOCK dann SYMBOL SHIFT und Taste	#

Graphics(G)-Modus

GRAPH dann nur eintasten	▄
GRAPH dann CAPS SHIFT und Taste	▀

Letter und Capitals Modi (Buchstaben und Großbuchstaben)

Nachdem ein Schlüsselwort oder Zeichen in Keyword-Mode erzeugt worden ist, ändert der Computer den Cursor automatisch auf L. Er befindet sich jetzt im L-Modus. Jede beliebige Taste drücken und der kleine Buchstabe erscheint. Eine Zahl anschlagen und die Zahl erscheint. Wenn Sie einen Großbuchstaben wünschen, halten Sie CAPS SHIFT hiedergedruckt und drücken dann die Buchstabentaste.

Wenn alle Buchstaben Großbuchstaben sein sollen, drücken Sie zuerst CAPS LOCK. Der Cursor wird zu C. Ihr Spectrum ist jetzt im *Capitals*-Modus und jedesmal, wenn Sie eine Buchstabentaste anschlagen, erhalten Sie einen Großbuchstaben. Um wieder in den L-Modus zurückzukehren, wird noch einmal CAPS LOCK gedrückt.

Extended-Modus

Der nächste Modus heißt '*extended* mode' und wird erzeugt, indem die Taste EXTEND MODE angeschlagen wird. Der Cursor wird jetzt zu einem E. Drücken Sie irgendeine Buchstabentaste und das obere Schlüsselwort von den zwei Schlüsselwörtern *über* dem erhöhten Teil erscheint. Drücken Sie zum Beispiel B und Sie erhalten BIN. Um das untere Schlüsselwort oder Zeichen über dem erhöhten Tastenteil zu erhalten, drücken Sie eine SYMBOL SHIFT Taste zuerst und lassen Sie sie niedergedrückt, dann drücken Sie die Buchstabentaste. Auf Taste B, zum Beispiel, erhalten Sie jetzt BRIGHT. Nach Anschlagen einer Taste (oder EXTEND MODE) im erweiterten Modus, geht der Computer automatisch in den Letter- oder Capitals-Modus zurück.

Graphics-Modus

Der fünfte Modus heißt 'graphics mode' und er wird durch Anschlagen der Taste GRAPH erzeugt. Der Cursor wird zu einem G. Drücken Sie Tasten 1 bis 8 und Sie sehen, daß die Grafikzeichen auf diesen Tasten auf dem Bildschirm erscheinen. Drücken Sie jetzt CAPS SHIFT und irgendeine Zahl von 1 bis 8. Die Grafikzeichen erscheinen wieder, aber dieses Mal sind schwarz und weiß umgedreht. Um den Grafik-Modus zu verlassen, müssen Sie GRAPHICS noch einmal drücken, da der Computer diesen nicht automatsch verläßt.

Aufbereitung mit dem Spectrum

Wenn Sie Befehle geben oder Programme schreiben für Ihren Spectrum, möchten Sie natürlich Fehler in Befehlen oder Programmzeilen korrigieren oder sie verändern. Sie können dieses leicht tun.

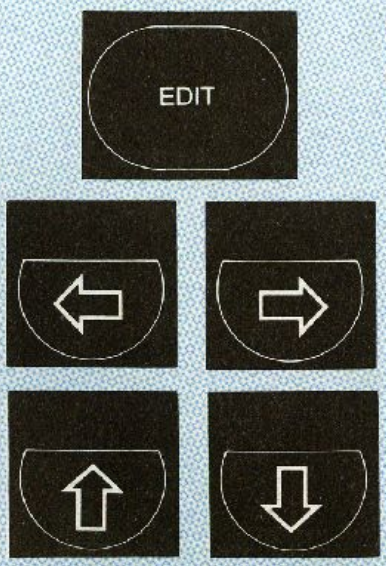
Wie man einen Fehler korrigiert

Wenn Sie versuchen, eine Zeile oder einen Befehl einzugeben, der in der BASIC-Sprache falsch ist, zeigt der Spectrum ein blinkendes ? vor dem Fehler. Um den Fehler zu korrigieren, halten Sie die Links- oder Rechts-Cursor-Steuertaste niedergedrückt, um den Cursor rechts neben den Fehler zu bringen. Dann entweder den Fehler löschen durch Anschlagen von DELETE oder etwas hinzufügen. Dann ENTER drücken.

Nehmen Sie zum Beispiel an, Sie möchten, daß der Computer 7 mit 8 multipliziert und Sie drücken nicht SYMBOL SHIFT, um das Zeichen * zu erhalten. Sie würden also eigentlich stattdessen

PRINT 7b8

eintippen. Der Spectrum kann diesen Befehl nicht ausführen, also zeigt er nach dem Anschlagen von ENTER ein blinkendes Fragezeichen vor dem b, wo der Fehler aufgetreten ist. Jetzt müssen Sie nur den Cursor nach *rechts* neben den Fehler bewegen und dann DELETE drücken, um das b zu entfernen. Dann SYMBOL SHIFT und B drücken, um * zu erhalten und ENTER anschlagen, damit der Computer jetzt die richtige Anweisung ausführt- Sie brauchen den Cursor nicht zuerst wieder an das Zeilenende bewegen.



Wie man eine Programmzeile aufbereitet

Wenn Sie ein Programm schreiben, entsteht eine Folge von numerierten Zeilen mit Anweisungen, welche 'Listing' genannt werden. Wenn Sie nach dem Schreiben eines Programmes, dieses Programm 'auflisten', indem Sie K(LIST) und ENTER drücken, müßten Sie ein > Zeichen bei einer der Programmzeilen sehen. Wenn dies nicht der Fall ist, drücken Sie die Herauf- und Herunter-Cursorsteuertaste. Wenn Sie jetzt EDIT anschlagen, wird diese Zeile unten auf dem Bildschirm kopiert und kann jetzt, wie zuvor, mit den Cursor- oder DELETE-Tasten verändert werden. ENTER drücken, um die neue Zeile in das Programm einzusetzen. Wenn Sie eine andere Zeile verändern möchten, bewegen Sie das > Zeichen mit den Herauf- oder Herunter-Cursorsteuertaste bis zu der Zeile, die Sie verändern möchten, und drücken Sie dann EDIT.

Um eine ganze Zeile aus dem Programm zu löschen, tasten Sie nur die Zeilennummer ein und drücken Sie ENTER. Wenn Sie ein Programm fahren, welches einen Fehler enthält, sehen Sie eine Fehlermitteilung. Diese wird auf Seite 74 erklärt.

DER FERNSEH-RECHNER

Der ZX Spectrum + kann sehr schnell und genau rechnen. Er braucht nur ein paar Zahlen, mit denen er arbeiten kann, und Zeichen wie + und −, die ihm sagen, was er mit den Zahlen machen soll.

Zuerst drücken Sie diese Anweisung (das + Zeichen finden Sie auf der K-Taste):

PRINT 6+2

Dies ist ein *Befehl*. Wenn Sie ENTER drücken, verschwindet der Befehl und die Antwort, die Zahl 8, wird auf dem Schirm erscheinen.

Ihr Spectrum verwendet 5 als '*Rechenoperatoren*' bezeichnete Zeichen zum Rechnen. In der Tafel unten auf dieser Seite können Sie sehen, was diese Zeichen bewirken. Sie können alle mit PRINT verwendet werden.

Der Spectrum kann auch Rechnungen und ihre Ergebnisse zusammen anzeigen. Geben Sie den folgenden Befehl ein:

PRINT "6+2=";6+2

Der Computer antwortet mit dem Display

6+2=8

Was hier geschieht ist, daß durch das PRINT *alles* zwischen den Anführungsstrichen (") auf dem Bildschirm erscheint, also 6+2=. Das Semikolon befiehlt dem Spectrum, das Ergebnis sofort nach dem Gleichheitszeichen anzugeben.

Die Rechenzeichen des Spectrum

Die folgenden Zeichen oder 'Rechenoperatoren' werden von dem Spectrum verwendet, um mathematische Berechnungen durchzuführen. Beachten, daß der Computer nicht × oder ± verwendet.

Symbol	Key	Funktion	Beispiel
+	K	zwei Zahlen addieren	8+2=10
−	J	zwei Zahlen subtrahieren	8−2=6
*	B	zwei Zahlen multiplizieren	8*2=16
/	V	zwei Zahlen dividieren	8/2=4
↑	H	die erste Zahl mit sich selber x-mal multiplizieren	8↑2=64

Ihr erstes Programm

Wenn ein Befehl ausgeführt worden ist, vergißt Ihr Spectrum ihn. Wenn Sie wollen, daß der Computer die Berechnung wiederholt, können Sie sie als Programm schreiben. Geben Sie diese Anweisung ein und drücken Sie ENTER.

10 **PRINT** 6+2

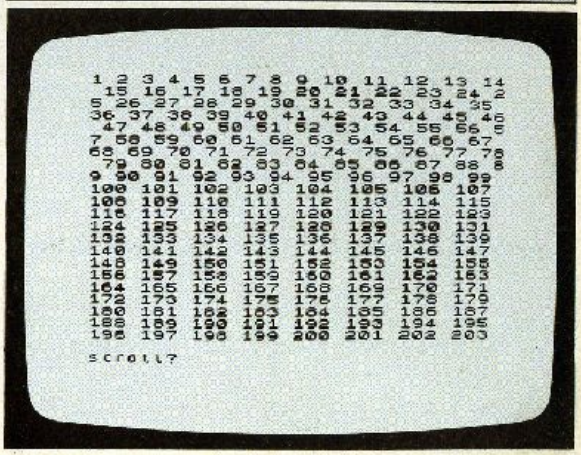
Dieses Mal wird sie nicht sofort befolgt, stattdessen zeigt der Computer die Anweisung auf dem Bildschirm. Als Nächstes R(RUN) drücken und ENTER. Jetzt erscheint das Ergebnis 8.

Die gesamte Anweisung ist jetzt ein Computerprogramm. Stellen Sie eine Nummer davor, nimmt der Spectrum die Anweisung in seinen Speicher auf, führt sie jedoch erst aus, wenn er dazu aufgefordert wird. Immer, wenn Sie ein Programm fahren durch Drücken von R(RUN) und ENTER, wird die Anweisung ausgeführt. Die Anweisung wird jetzt '*statement*' genannt anstatt 'Befehl' und sie bildet eine numerierte Zeile in einem Programm. Programm-Statements werden immer der Reihe ihrer Zeilennummern nach ausgeführt und diese steigen normalerweise in Zehnern.

Als Nächstes geben Sie das folgende Programm ein. Nicht vergessen, ENTER nach jeder Zeile zu drücken, und dann, wenn Sie fertig sind, R(RUN) und ENTER. Wenn Sie das Programm gefahren haben, müßten Sie jetzt Folgendes sehen.

ZAHLENTAFEL

```
10 LET n=1
20 PRINT n;" ";
30 LET n=n+1
40 GO TO 20
```



Alle Zahlen von 1 bis 203 sind dargestellt. Jetzt eine beliebige Taste drücken außer N, Leertaste, STOP oder BREAK. Ein völlig neuer Zahlensatz erscheint.

Dieses Programm verwendet eine '*Variable*'. In diesem Fall heißt die Variable n. Es kann jeder Buchstabe oder jedes Wort benutzt werden – n

steht einfach für 'number'.

Einer Variablen wird ein Wert gegeben, der sich ändert, während das Programm läuft. In Zeile 10, wird das Schlüsselwort LET verwendet, um den Wert auf 1 zu setzen. Zeile 20 zeigt den Wert, gefolgt von einer Leerstelle. Dann in Zeile 30 wird wieder LET verwendet, dieses Mal, um den Wert um 1 zu erhöhen, n wird also aur 2. Zeile 40 verwendet das (einzelne) Schlüsselwort GOTO, um das Programm zu Zeile 20 zurückzuschicken, welche nun 2 anzeigt. Dies wird wieder und wieder wiederholt, bis die Zahlen den Bildschirm füllen.

Wie man ein Programm dazubringt, nach einer Zahl zu fragen

Halten Sie das Programm durch Drücken von BREAK an. Tasten Sie jetzt eine neue Zeile ein

```
10 INPUT n
```

Diese Zeile ersetzt die alte Zeile 10 im Programm. Wenn Sie das Programm jetzt fahren, wartet der Computer darauf, daß Sie eine Zahl eingeben Geben Sie eine beliebige Zahl ein und drücken Sie ENTER. Jetzt beginnen die Nummern bei der von Ihnen eingegebenen Zahl, weil INPUT n den Wert von n der von Ihnen eingegeben Zahl gleichsetzt.

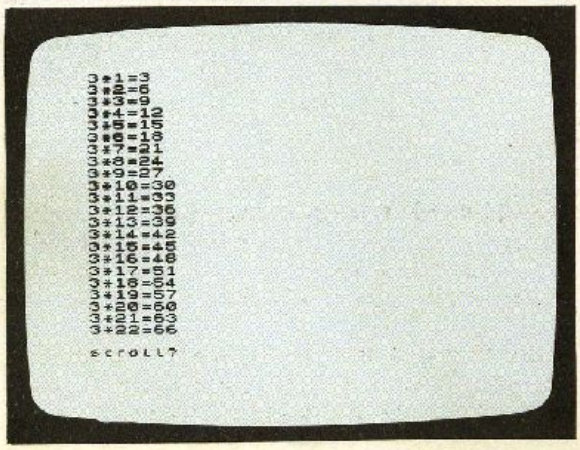
Programmierung einer Multiplikationstabelle

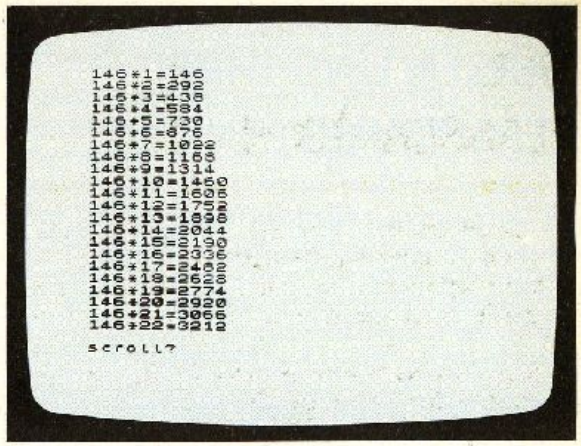
Den Reset-Knopf drücken, um das alte Programm zu entfernen, und das rächste eingeben. Dieses Programm bringt den Spectrum dazu, zu multiplizieren.

Tasten Sie irgendeine Zahl ein und eine Multiplikationstabelle für diese Zahl blinkt auf dem Bildschirm. Eine Taste außer N, BREAK oder Leertaste drücken und die Tabelle wird fortgesetzt. BREAK drücken und das Programm noch einmal fahren, um eine neue Tabelle zu erzeugen. Hier ist das Programm und was Sie auf dem Schirm sehen müßten, wenn Sie 3 eintasten und 146.

MULTIPLIKATIONSTABELLE

```
10 LET x=1
20 INPUT n
30 PRINT n;"*";x;"=";n*x
40 LET x=x+1
50 GO TO 30
```





Warum Sie Klammern benutzen müssen

Manchmal werden Sie in einer Rechnung Klammern benutzen müssen. Geben Sie diese zwei Befehle ein und vergleichen Sie die Ergebnisse:

```
PRINT 6+2/4
PRINT (6+2)/4
```

Das erste ergibt 6,5 und der zweite 2. Der Grund hierfür ist der, daß der Computer ein eingebautes Prioritätensystem hat, welches er bei Rechenaufgaben benutzt. Er führt zuerst ↑ aus, dann * oder / und zuletzt + oder −, aber er führt *immer* zuerst Kalkulationen in Klammern durch. Im ersten Befehl teilt er also erst 2 durch 4 und addiert dann das Ergebnis (0,5) zu 6. Im zweiten Befehl addiert der Computer 6 und 2 und teilt sie dann durch 4.

Interpunktion mit Ihrem Spectrum

Der Spectrum verwendet eine Reihe von Interpunktionszeichen. Sie sind sehr wichtig, weil viele sich als Anweisungen an den Computer verdoppeln und die Art und Weise beeinflussen, wie er eine Programmzeile versteht oder ein Display produziert.

;	**Semikolon.** Sagt dem Computer in Verbindung mit PRINT, daß er die zwei Teile beidseitig vom Semikolon nebeneinander auf dem Bildschirm zeigen soll.
:	**Doppelpunkt.** Signalisiert das Ende eines Statements in einer Programmzeile und den Anfang der nächsten.
"	**Anführungsstriche.** Alle Zeichen in Anführungsstrichen werden nicht als Zahlen oder Variablen behandelt, sondern nur als Text. Anführungsstriche beginnen und beenden einen 'String'.
,	**Komma.** Sagt dem Computer in Verbindung mit PRINT, daß er die folgenden Teile entweder in der Mitte der Zeile oder am Anfang der nächsten zeigen soll. *Nicht* zur Kennzeichnung von Tausendern oder Millionen benutzen!
.	**Punkt.** Entweder Dezimalpunkt oder Endpunkt.

FARBEN UND IHRE BENUTZUNG

Ihr ZX Spectrum + kann acht verschiedene Farben erzeugen und jede Farbe hat eine Farbcodenummer. Sie können jede der Farben auf drei verschiedene Arten verwenden – als Border-Farbe (Umrandung), als Ink-Farbe (Schrift) und als Paper-Farbe (Papier).

ZX Spectrum + und seine Farbcodierung

Diese Tafel zeigt Ihnen die vom Spectrum benutzten Farben und ihre Codes. Sie brauchen diese Codes nicht auswendig zu wissen, da die Zahlentasten, durch die sie erzeugt werden, ebenfalls mit der jeweiligen Farbe gekennzeichnet sind.

Nummer	Farbe
0	schwarz
1	blau
2	rot
3	violett
4	grün
5	hellblau
6	gelb
7	weiß

Die verschiedenen Schattierungen, die Sie auf Ihrem Fernseher sehen, hängen vom Fernsehgerät, der Farbeinstellung, der Kontrast- und Helligkeitsregelung ab. Denken Sie daran, daß Sie einen Farbfernseher brauchen!

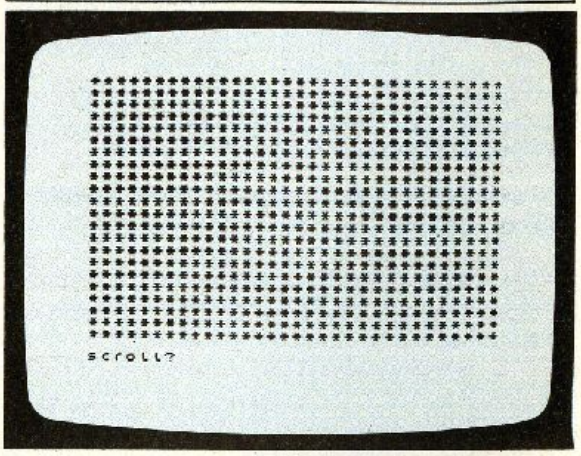
Die drei Arten der Farbenbenutzung

Sie können die Farben auf drei Arten regeln. Die *Border-Farbe* ist die Farbe der Umrandung um den zentralen Displaybereich herum. Die *Ink*-Farbe ist die Farbe, in der Zeichen (Buchstaben, Zahlen, mathematische Zeichen und Grafikzeichen) und Punkte oder Linien erscheinen. Die *Paper*-Farbe ist die Farbe des Hintergrunds, entweder des gesamte Displaybereich oder eines Quadrats um jedes Zeichen.

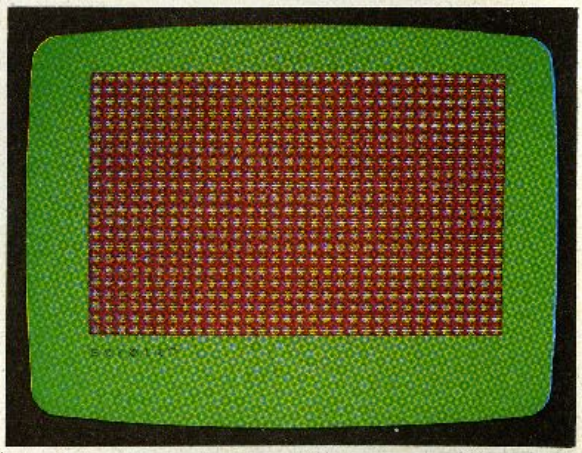
Wenn Sie den Spectrum einschalten, verwendet er die voreingestellten Farben. Die Ink-Farbe ist schwarz und die Border- und Paper-Farbe ist weiß. Sie können diese Farben sofort ändern, indem Sie direkte Befehle durch die Tastatur eingeben. Sie haben dies bereits auf Seiten 6–7 gesehen. Drücken Sie jetzt den Rückstellknopf, tasten Sie dieses einfache Programm ein und fahren Sie es.

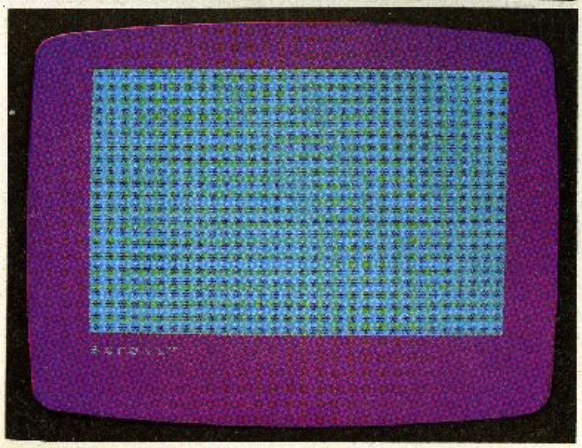
FARB-TESTER

```
10 PRINT "*";
20 GO TO 10
```



Es bildet sich ein Muster von Sternen in schwarz und weiß. Drücken Sie jetzt BREAK und geben Sie einige Farbbefehle ein. Tasten Sie die Schlüsselwörter BORDER, INK und PAPER ein, denen jeweils eine Zahl von Ø bis 7 folgt, drücken Sie ENTER nach jeder Taste und lassen Sie das Programm ablaufen. Hier sind zwei Displays, das erste mit BORDER4, PAPER2 und INK7, das zweite mit BORDER3, PAPER5 und INK1.





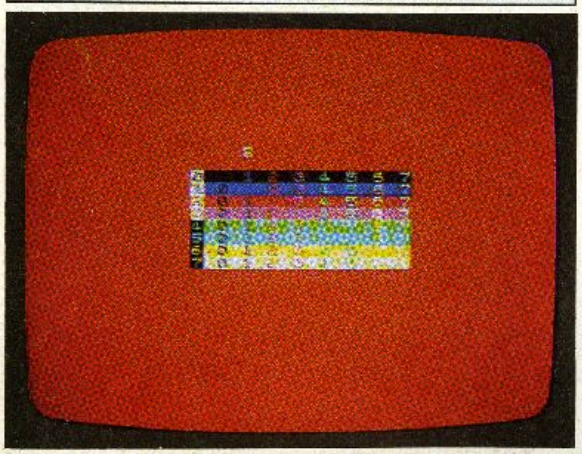
Wie man Programme mit Farben schreibt

Sie können die Schlüsselwörter BORDER, PAPER und INK in einem Programm verwenden, um Text, Tabellen, Muster und Bilder in den verschiedensten Farben erscheinen zu lassen. Wenn man BORDER in einer Programmzeile verwendet, ändert sich die Umrandungsfarbe, sobald der Spectrum an der Zeile ankommt. INK alleine in einer Zeile für sich ergibt eine neue Schriftfarbe, wenn die nächsten Zeichen oder Linien auf dem Schirm erscheinen. PAPER in einer Zeile für sich verändert die Papierfarbe, aber nur um Zeichen herum (oder Punkte oder Linien). Wenn Sie wollen, daß der ganze Hintergrund eine bestimmte Farbe hat, müssen Sie nach PAPER CLS drücken.

Sie können INK und PAPER auch nach PRINT benutzen. In diesem Fall haben nur die durch PRINT dargestellten Zeichen diese INK- und PAPER-Farben. Das nächste Programm zeigt alle Border-, Ink- und Paperfarben. Es zeigt Ihnen außerdem, wie man INK und PAPER nach PRINT verwendet.

FARBENKOMBINATIONEN

```
  10 FOR b=0 TO 7
  20 BORDER b: PAPER b: CLS
  30 PRINT AT 6,12; INK 9;b
  40 FOR p=0 TO 7
  50 PRINT AT p+8,8; INK p; PAPE
R 9;p;
  60 BEEP 0.5,b*p-20+p
  70 FOR i=0 TO 7
  80 PRINT INK i; PAPER p;" ";i;
  90 BEEP 0.01,i*5
 100 NEXT i
 110 NEXT p
 120 NEXT b
```



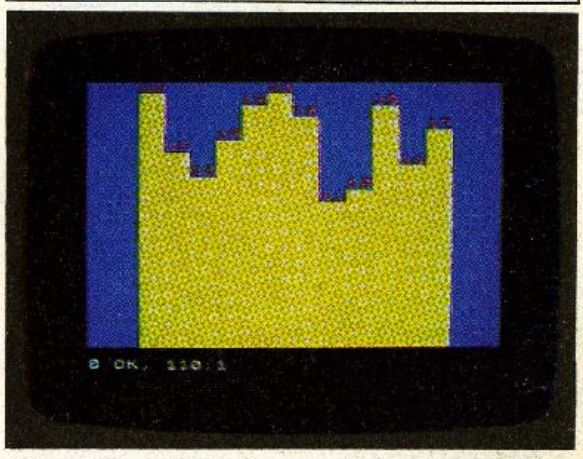
Wenn Sie dieses Programm fahren, sehen Sie alle Kombinationen von Border, Paper und Ink. Das Programm hat drei Variable, b für die Bordernummer, i für die Inknummer und p für die Papernummer. BEEP erzeugt den Ton, und die mit FOR und NEXT beginnenden Zeilen kennzeichnen den Anfang und das Ende einer Programmschleife, die alle Farbnummern von 0 bis 7 der Reihe nach ändert. Beachten Sie, daß INK und PAPER einen Wert von 9 haben können. Damit wird die Ink- bzw. Paperfarbe schwarz oder weiß, so daß sie sich gegen den Hintergrund bzw. ein Zeichen abhebt.

Das Programmieren von farbigen Balkendiagrammen

Das Nächste Programm verwendet die Farben des Spectrum, um ein Balkendiagramm zu erzeugen. Es zeigt zwölf Tagestemperaturen als gelbe Säulen mit Nummern. In Zeile 60 geben Sie zwei Leerstellen zwischen den Anführungsstrichen ein.

BALKENDIAGRAMM

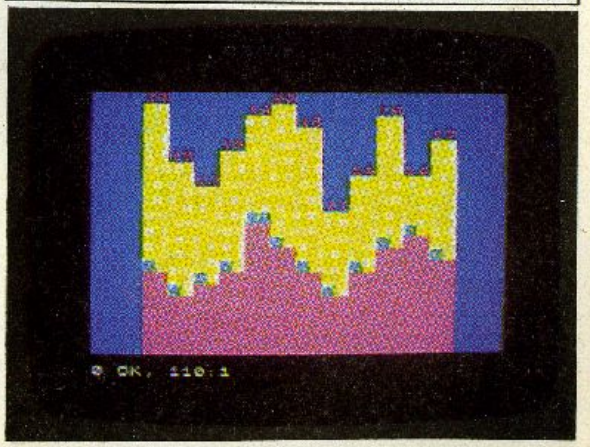
```
  10 BORDER 0: PAPER 1: CLS
  20 LET c=4
  30 FOR x=1 TO 12
  40 READ t
  50 FOR l=21 TO 21-t STEP -1
  60 PRINT PAPER 6;AT l,c;"  "
  70 NEXT l
  80 PRINT INK 2;AT 20-t,c;t
  90 LET c=c+2
 100 NEXT x
 110 DATA 20,15,13,16,19,20,18,1
1,12,19,14,17
```



Fügen Sie jetzt die folgenden Zeilen hinzu und tasten Sie die neue Zeile 110 ein wie dargestellt. Jetzt erscheint das Diagramm in zwei Farben. Mehr über READ und DATA auf Seite 33.

DOPPELBALKENDIAGRAMM

```
  85 READ t
  86 FOR l=21 TO 21-t STEP -1
  87 PRINT PAPER 3;AT l,c;"  "
  88 NEXT l
  89 PRINT INK 1; PAPER 5;AT 20-
t,c;t
 110 DATA 20,6,15,4,13,5,16,6,19
,10,20,8,18,6,11,4,14,6,19,8,14,
9,17,7
```



EINFACHE GRAFIKEN ZUM SELBERMACHEN

Ihr ZX Spectrum + kann niedrig- und hochauflösende Grafiken produzieren. Beide Arten können gleichzeitig auf dem Bildschirm erscheinen. Niedrigauflösende Grafikdisplays bestehen aus Farbblöcken. Auf diesen beiden Seiten sehen Sie, wie des Blöcke von der Tastatur aus gebildet werden, une wie sie auf dem Bildschhirm positioniert werden.

Der niedrigauflösende Bildschirm

Der niedrigauflösende Bildschirm hat 32 horizontale und 22 vertikale Positionen. Jede dieser Bildschirmpositionen hat 2 Nummern, um sie festzulegen. Die erste ist die Zeilennummer, sie gibt die Anzahl der Zeilen von oben nach unten, um die gewüschte Position zu erreichen. Die oberste Zeile ist Ø, die untere ist Zeile 21. Dann folgt die Anzahl der Spalten von lins nach rechts. Die ganz linke Spalte ist Ø, die ganz rechte ist 31. (Auf Seite 80 sehen Sie das Gitter für niedrige Auflösung.)

Das nächste Programm füllt diese Positionen mit Farben. Das Schlüsselwort RND wählt eine zufällige Ink-Farbe.

ZUFALLSQUADARATE

```
10 BORDER 1: INK RND*7
20 PRINT "■";
30 GO TO 10
```

Hier erscheinen Quadrate überall auf dem Bildschirm verteilt. Um ein Zeichen an einer bestimmten Stelle erscheinen zu lassen, müssen Sie das Schlüsselwort AT zusammen mit PRINT verwenden. AT folgt nach PRINT, dann folgen Zeilennummer, Komma, Spaltennummer und ein Semikolon. Der Befehl

PRINT AT 11, 16; "*"

beispielsweise zeigt einen Stern in Zeile 11, Spalte 16, das ist die Mitte des Bildschirms.

Wie man Regenbogenmuster zeichnet

Eine gute Art, um farbige Muster zu produzieren, ist die Verwendung von FOR NEXT Schleifen in Ihrem Grafikprogramm. FOR NEXT Schleifen sind Teile eines Programms, die sich für eine bestimmte Anzahl von Malen wiederholen. In der Zeile, in der die Schleife beginnt, können Sie dem Computer mitteilen, wie oft die Schleife

Wie man Grafikzeichen-Tasten selekteert

Ihr ZX Spectrum + besitzt Grafikzeichen-Tasten, die es sehr einfach machen, niedrigauflösende Grafiken zu programmieren. Sie sehen sie auf den Tasten 1 - 8.

Um die Grafikzeichen auf dem Bildschirm zu erstellen, drücken Sie die Graph-Taste und dann Tasten 1 bis 8, mit Leertaste zwischen jeder einzelnen. Die Grafikzeichen erscheinen dann unten auf dem Bildschirm. Der weiße Teil eines jeden Zeichens auf der Taste ist die Ink-Farbe, der schwarze Teil ist Paper-Farbe. Drücken Sie jetzt die Tasten nocheinmal und halten Sie gleichzeitig CAPS SHIFT. Die Zeichen erscheinen dieses Mal mit den umgekehrten Ink- und Paper-Farben.

Genauso geben Sie Grafikzeichen in die Programmzeilen. Um den Grafikmodus zu verlassen und zu den normalen Zahlentasten zurückzukehren, drücken Sie einfach nocheinmal GRAPH.

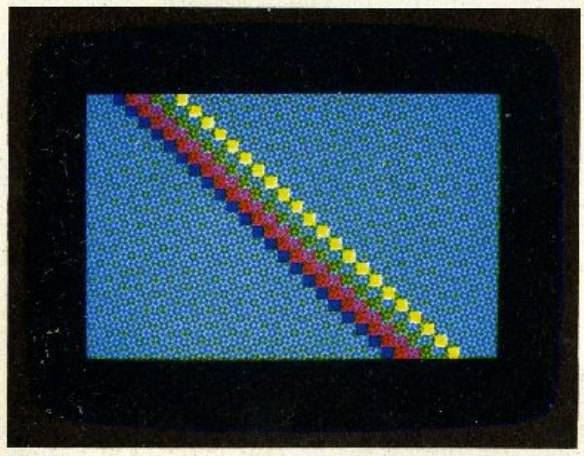
GRAPH
Diese Taste wird verwendet, um den Spectrum in den Grafikmodus umzuschalten.

Taste 8
Diese taste wird oft gebraucht mit GRAPH und CAPS SHIFT um ein solides Farbquadrat zu produzieren.

ausgeführt werden soll. Während dies geschieht, kann sie benutzt werden, um beispielsweise Zeichen auf dem Bildschirm zu plazieren.

Sie brauchen nicht nur jedesmal eine Schleife zu programmieren, sondern können eine innerhalb einer anderen anbringen, was oft sehr nützlich ist. Das nächste Programm zeigt Ihnen, wie zwei FOR NEXT Schleifen (eine in die andere verschachtelt) verwendet werden können, um mit INK und AT erstellte Farben und Positionen zu verändern. Sie sehen in dem Abschnitt am Ende dieser Seite, wie diese Schleifen programmiert werden.

```
 5 BORDER 0: PAPER 5: CLS
10 LET x=1
20 FOR l=0 TO 21
30 FOR c=1 TO 6
40 PRINT INK c;AT l,c+x;"■"
50 NEXT c
60 LET x=x+1
70 NEXT l
```



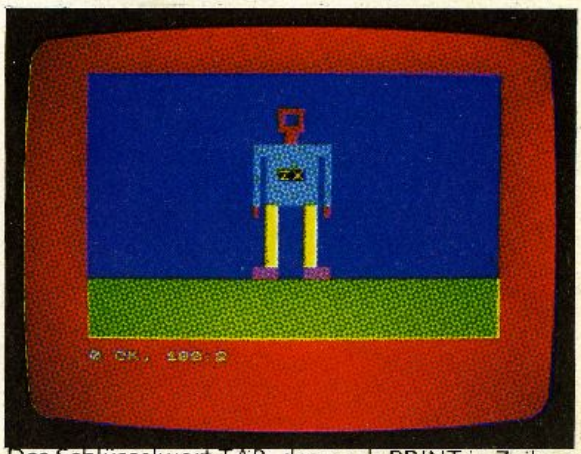
Das Programmieren von Bildern

Bei niedrigauflösenden Grafiken könne Sie Ihre Bilder 'anmalen', indem Sie die positionen und Farben der Grafikzeichen ausarbeiten. Sie können Ihr eigenes Bild plannen, indem Sie das Gitter für Niedrigauflösung auf Seite 80 benutzen. Wählen Sie nun die Grafikzeichen wie auf der gegenüberliegenden Seite gezeigt, und geben Sie die Programmzeilen eine nach der anderen ein, um das Bild aufzubauen.

Cas nächste Programm zeigt Ihnen die Art von Resultaten, die Sie erzielen können. Alle Formen darin findin Sie auf den Zahlentasten. Sie können entweder warten, bis Sie alle Zeilen eingetastet haben, ehe Sie das Programm fahren, oder Sie können es nach jeder Zeile fahren, um zu sehen, wie die einzelnen Teile des Roboters zusammengesetzt sind. (Denken Sie daran, daß Sie falsche Grafikzeichen genauso aufbereiten können wie falsche Zahlen oder Buchstaben.)

ZX ROBOTER

```
  5 BORDER 2: PAPER 1: CLS
 10 PRINT INK 2;AT 3,15;"■"
 15 PRINT INK 2;AT 4,15;"■"
 20 PRINT INK 2;AT 5,15;"■"
 30 PRINT INK 5;AT 6,13;"■■■"
 40 FOR l=7 TO 10: PRINT INK 5;
AT l,13;"■■■■■": NEXT l
 45 PRINT INK 6; PAPER 0;AT 8,1
5;"ZX"
 50 PRINT INK 2;AT 11,13;"■
■"
 60 FOR l=11 TO 15: PRINT INK 6
;AT l,14;"■ ■": NEXT l
 70 PRINT INK 3;AT 16,13;"■■";T
AB 17;"■■"
 80 FOR l=17 TO 21: FOR c=0 TO
31
 90 PRINT INK 4;AT l,c;"■"
100 NEXT c: NEXT l
```



Das Schlüsselwort TAB, das nach PRINT in Zeile 70 erscheint, wird benutzt, um das Zeichen innerhalb der Zeile, die der Computer gerade bearbeiter, hin und her zu bewegen. Nach TAB folgt eine Nummer von 0 bis 31, um die Spaltenposition festzulegen.

Wie man FOR NEXT Schleife gebraucht

Eine for NEXT Schleife beginnt immer mit einer Zeile, die die Schlüsselwörter FOR und TO enthält, sowie einer Variablen mit ihrem Anfangs- und Endwert. Beispiel:

30 **FOR** c = 1 **TO** 6

Die Variable ist hier c. Die Schleife, die hier beginnt, enthält eine oder mehere Zeilen, die den Computer eine bestimmte Tätigkeit wiederholen läßt. Diese benutzen die Variable c möglicherweise selbst. FOR NEXT Schleifen enden immer mit dem Schlüsselwort NEXT und der Variablen, zum Beispiel

50 **NEXT** c

Wenn das Programm gefahren wird, wiederholt sich die gesamte Schleife von FOR bis NEXT für eine festgelegte Anzahl von Malen. Die Variable beginnt beim ersten Wert vor TO und erhöht sich jedesmal um 1 bis sie zum Grenzwert nach TO ankommt.

In diesem Falle wiederholt sich die Schleife sechs Mal; c beginnt bei 1, wird dann zu 2, 3, 4, 5 und letztlich 6.

'NEXT' verwendet. Das bedeutet, das für jeden Zyklus der 'äußeren' Schleife die 'mittlere' Schleife all ihre Zyklen durchläuft. Die 'innere' Schleife geht am häufigsten durch ihre Zyklen, nämlich jedesmal, wenn die 'mittlere' Schleife ihren Zyklus durchläuft.

DER BILDSCHIRM-ZEICHENBLOCK

Grafiken auf dem ZX dem Spectrum + brauchen nicht nur grobe Muster und Bilder mit geringer Auflösung sein. Durch seine Hochauflösungseigenschaft können Sie mit Ihrem Spectrum detaillierte Bilder mit scharfen Umrissen und geraden oder gebogenen Linien und Kanten erzeugen.

Hochauflösungsgrafiken bestehen aus vielen Punkten, die alle hintereinander gesetzt werden, um eine Linie zu schaffen oder eine Figur mit Farbe.auszufüllen. Jeder Punkt ist ein Vierundsechzigstel der Größe des Quadrates, das in Grafiken mit niedriger Auflösung verwendet wird. Wenn Sie diesen Befehl eingeben

```
PLOT 128,87
```

sehen Sie eines in der Mitte des Bildschirms.

Die in Hochauflösungsgrafik verwendeten Punkte werden als '*Pixel*' hezeichnet, eine Abkürzung für 'picture cells' (Bildzellen). Wie ein Zeichen mit niedriger Auflösung, braucht jedes Pixel zwei Nummern, um seine Position genau zu bestimmen.

Das Hochauflösungsgitter

Das Hochauflösungsgitter besteht aus 256 Pixel waagerecht und 176 Pixel senkrecht. Anders als bei Displays mit niedriger Auflösung ist die erste Zahl jedoch hierbei die horizontale Koordinate – die Position des Pixels auf der waagerechten Linie. Diese Positionsnummern gehen von Ø an der linken Kante bis zu 255 an der rechten Kante. Die zweite Zahl ist die vertikale Koordinate, aber die Zahlen gehen von Ø unten bis 175 oben. Positio Ø,Ø ist die untere linke Ecke, nicht die obere linke Ecke wie in Displays mit niedriger Auflösung. Eine Tafel des Hochauflösungsgitters siehe Seite 80.

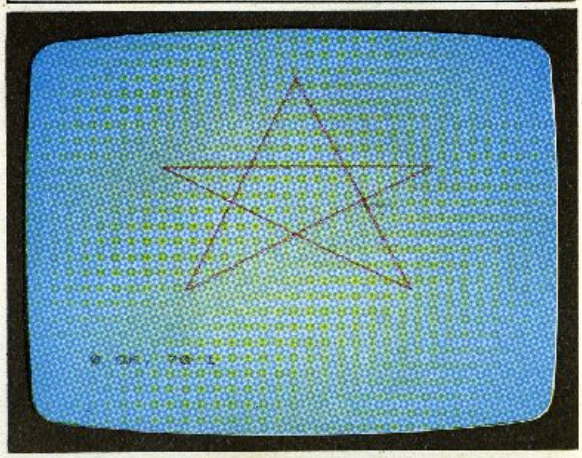
Plotten und Zeichnen

Zur Erzeugung von Hochauflösungsgrafik brauchen Sie nur drei Schlüsselwörter – PLOT, DRAW und CIRCLE. Hinter PLOT stehen die durch ein Komma getrennten horizontalen und vertikalen Koordinaten und dieses Schlüsselwort bestimmt die Position eines Pixels. Hinter DRAW stehen ebenfalls zwei Zahlen, die durch ein Komma getrennt werden, dies sind jedoch nicht die Koordinaten einer Position, sondern die Abstände in Pixel von einer Position zu der nächsten irgendwo auf dem Bildschirm, und DRAW zeichnet eine Linie zwischen den zwei Positionen.

Die erste Position ist Ø,Ø, wenn PLOT oder DRAW noch nicht im Programm benutzt worden sind. Wenn sie bereits benutzt wurden, ist diese Position die letzte PLOT-Position oder die letzte durch DRAW erreichte Position, jenachdem, welches die allerletzte war. Die DRAW-Anweisung zeichnet dann einen Strich zur neuen Position. Versuchen Sie dieses Programm.

STERN

```
10 INK 2
20 PLOT 128,174
30 DRAW 70,-140
40 DRAW -152,80
50 DRAW 164,0
60 DRAW -150,-80
70 DRAW 70,140
```

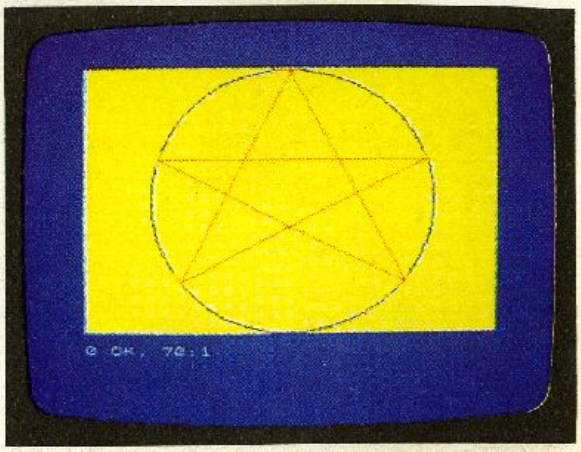


PLOT bewegt die Startposition nach oben auf dem Bildschirm. Dann zeichnen die fünf DRAW-Anweisungen die fünf roten Linien.

Fügen Sie jetzt diese Zeilen hinzu.

```
4 BORDER1:PAPER6:INK1:CLS
5 CIRCLE 128,87,87
```

Fahren Sie das Programm noch einmal und der rote Stern erscheint in einem blauen Kreis.



CIRCLE benötigt drei Werte. Die ersten beiden geben die Position der Kreismitte und der dritte ist der Radius. Sie können auch einen dritten Wert zu den DRAW-Anweisungen hinzufügen.

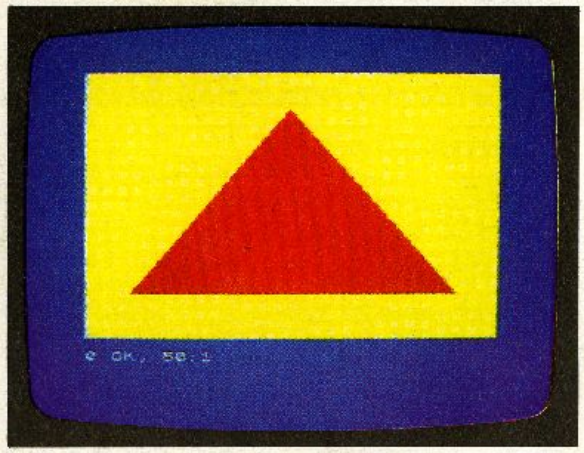
Versuchen Sie Werte zwischen 2 und −2 in dem Programm und sehen Sie, was passiert.

Wie man Figuren ausfüllt

Sie können leicht ausgefüllte Figuren in hoher Auflösung erzeugen, indem Sie viele Linien dicht aneinander zeichnen. Dies kann mit einer FOR NEXT Schleife geschehen, die die DRAW-Positionen so ändert, daß sie jedesmal um 1 größer werden.

AUSGEFÜLLTES DREIECK

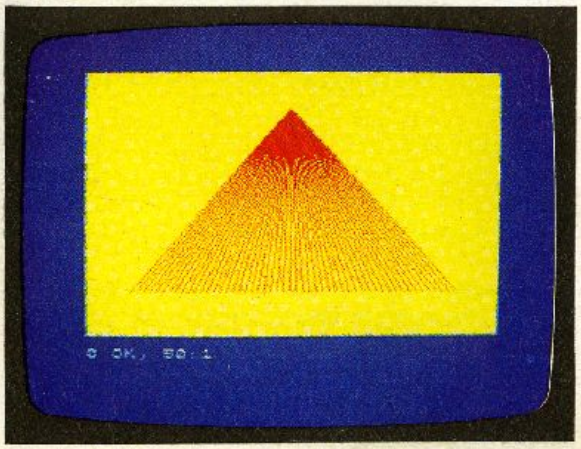
```
  10 BORDER 1: PAPER 6: INK 2: C
LS
  20 FOR x=-100 TO 100
  30 PLOT 128,150
  40 DRAW x,-120
  50 NEXT x
```



Sie können einen interessanten Effekt erzielen, wenn Sie die Linien etwas voneinander entfernt zeichnen. Sie erreichen dies, indem Sie das Schlüsselwort STEP hinzufügen und eine Nummer für die FOR-Anweisung. Geben Sie eine andere Zeile 20 ein und fahren Sie das Programm wieder

20 FOR x=−100 TO 100 STEP 4

Dieses Mal erscheint die untenstehende Fächerform. Der Grund hierfür ist, daß STEP dazuführt, daß x in Sprüngen von 4 erhöht wird anstatt um 1, wenn eine Linie gezeichnet wird.



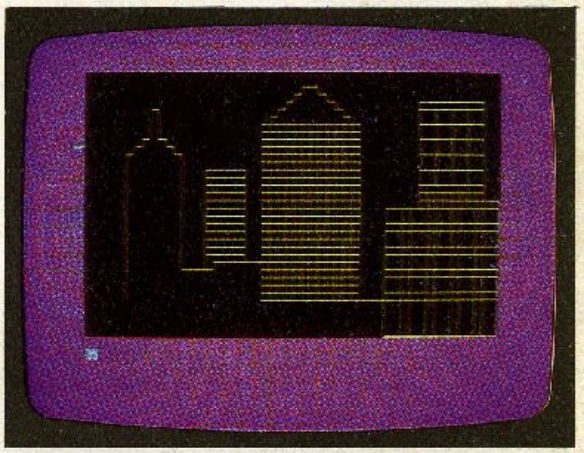
Ihr Bildschirm-Zeichenblock

Sie brauchen nicht jedesmal, wenn Sie ein Bild oder Muster wollen, extra ein Programm zu schreiben. Sie können ein Programm benutzen, mit dem SIE ein Bild direkt auf dem Schirm aufbauen können.

Es fängt damit an, daß das Schlüsselwort INPUT Sie nach einer Inkzahl fragt. Dann, wenn INPUT noch einmal gedrückt wird (dieses Mal mit einem $ Zeichen, um einen String zu kennzeichnen) zeichnet der Computer kurze Linien jedesmal, wenn Sie eine von vier bestimmten Tasten drücken – u,d,l oder r – gefolgt von ENTER. Er benutzt IF und THEN, um Entscheidungen zu treffen.

ZEICHENBLOCK UND BEISPIEL

```
  10 INPUT "INK ";i
  20 BORDER 3: PAPER 0: INK i: C
LS
  30 PLOT 25,25
  40 LET x=5
  50 INPUT k$
  60 IF k$="u" THEN DRAW 0,x
  70 IF k$="d" THEN DRAW 0,-x
  80 IF k$="r" THEN DRAW x,0
  90 IF k$="l" THEN DRAW -x,0
 100 GO TO 50
```



Entscheidungen mit IF und THEN

Zeilen 60 bis 90 in dem Sketchpad-Programm enthalten IF THEN Anweisungen. Diese ermöglichen es Ihrem Spectrum, Entscheidungen zu treffen. In diesem Fall prüft der Computer, ob der eingegebene Buchstabe u,d,l oder r ist. WENN (IF) einer von diesen gedrückt wird, DANN (THEN) wird dem Computer gesagt, daß er eine Linie zeichnen soll nach oben oder unten, links oder rechts. Er zeichnet keine Linie, wenn ein Großbuchstabe eingegeben wird.

Hinter IF steht immer etwas, das der Spectrum überprüft, um zu sehen, ob es stimmt oder ob es gerade geschieht – wie das Drücken von bestimmten Tasten. Wenn es stimmt oder gerade geschieht, wird der hinter THEN stehende Vorgang ausgeführt. Wenn nicht, geht das Programm zur nächsten Zeile über. Alles, was in einer Zeile hinter THEN steht, unterliegt der Entscheidung. In dieser Linie

110 IF b=5 THEN PRINT "." GOTO 200

Wirt der komputer nur bis Linie 200 gehen wenn b 5 ist.

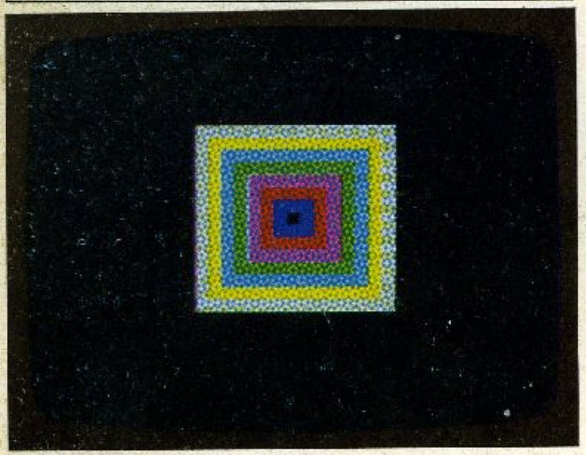
DAS ENTWERFEN VON MUSTERN UND BILDERN

Sie können mit Ihrem ZX Spectrum + alle möglichen Muster und Bilder entwerfen durch Benutzung von entweder Hochauflösungsgrafik, Niederauflösungsgrafik oder beidem. Das Beste im Fall von Grafik ist es, zuerst Ihr Design auf einer Kopie der Gitter auf Seite 80 zu zeichnen.

Um Muster und Bilder zu zeichnen, benutzen Sie oft FOR-NEXT-Schleifen, welche einen Teil des Programms x-Mal wiederholen. Jedesmal können sich Positionen und Farben des Zeichen oder Linien ändern, normalerweise auf regelmäßige Art und Weise. Hier ist ein Programm, welches diese Technik benutzt.

QUADRATE

```
10 BORDER 0: PAPER 0: CLS
20 FOR x=7 TO 0 STEP -1
30 INK x
40 FOR l=11-x TO 11+x
50 FOR c=16-x TO 16+x
60 PRINT AT l,c;"■"
70 NEXT c
80 NEXT l
90 NEXT x
```



Dieses Programm enthält drei FOR-NEXT-Schleifen. Die x-Schleife ändert die Farbe und auch die Größe der produzierten großen Quadrate, während die l-Schleife und die c-Schleife die Zeilen- und Spaltenposition des kleinen Quadrates verändern, wenn es gedruckt wird.

Zufallseffekte und Subroutinen (Unterprogramme)

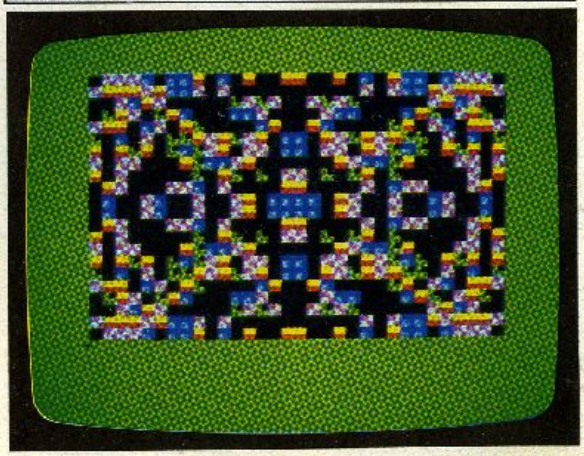
Mit Schleifen kann man nicht nur identische Muster erzeugen, jedesmal, wenn ein Grafikprogramm gefahren wird. Durch die Benutzung des Schlüsselworts RND (Abkürzung für RaNDom = Zufall) in Schleifen, können Sie Farben, Positionen und andere Displayeigenschaften jedesmal verändern. Sehen Sie sich das Mosaik-Programm auf Seite 10 an. Es funktioniert, weil die INK-Farbe RND*7 ist, d.h. jede Zahl mit einem Dezimalpunkt von 0 bis 7. INK ändert dies auf die nächste ganze Zahl um. Jedesmal, wenn also ein Quadrat aufgezeigt wird, ist seine Farbe irgendeine Farbe von INK 0 bis INK 7.

Das nächste Programm zeichnet symmetrische Muster von Grafikzeichen auf dem Bildschirm. Es verwendet RND, um diese Zeichen und ihre Positionen zu verändern. Die Variablen i und p geben die Ink- und Paperfarben und a gibt an, wieviele Muster gezeichnet werden (in diesem Fall vier). Die Variable n gibt die Anzahl von Zeichn in jedem Muster an, während x eine Zufallszahl von 129 bis 142 ist.

Die Anweisung GOSUB 1000 IN Zeile 50 schickt den Computer in ein Unterprogramm.

SYMMETRISCHE MUSTER

```
  10 BORDER 4: PAPER 0: CLS
  20 LET i=4: LET p=0
  30 FOR a=1 TO 4
  40 LET x=RND*13+129
  50 FOR n=1 TO 40: GO SUB 1000:
NEXT n
  60 LET i=i+1: LET p=p+1
  70 PAUSE 100
  80 NEXT a
  90 STOP
1000 LET l=INT (RND*11)
1010 LET c=INT (RND*16)
1020 INK i: PAPER p
1030 PRINT AT l,c;CHR$ x
1040 PRINT AT l,31-c;CHR$ x
1050 PRINT AT 21-l,c;CHR$ x
1060 PRINT AT 21-l,31-c;CHR$ x
1070 BEEP 0.01,l*c/3
1080 RETURN
```



Eine Subroutine ist eine Gruppe von Zeilen, die sich wie ein Programm innerhalb eines Programms verhalten. In diesem Programm befindet sich das Unterprogramm bei Zeile 1000. Es zeigt ein Grafikzeichen in vier Vierteln des Bildschirms, so daß alle von der Mitte gleich weit entfernt sind (Position 11,16). Diese Entfernung wird durch die Zeilen 1000 und 1010 angegeben, wobei l die Entfernung in Zeilen und c die Entfernung in Spalten angibt. INT verändert die Zufallszahl in eine ganze Zahl, so daß l eine ganze Zahl von 0 bis 10 ist und c eine ganze Zahl von 0 bis 15. Dann zeigen Zeilen 1030 bis 1060 die Grafikzeichen, deren Code x ist (siehe

Zeichensatztabelle auf Seite 51). BEEP erzeugt den Ton, der mit der Position verbunden ist, und RETURN in Zeile 1080 sendet das Programm zurück zur nächsten Anweisung nach GOSUB in Zeile 50.

Zeile 60 verändert Ink- und Paperfarbe, PAUSE 100 in Zeile 70 verzögert das Programm um zwei Sekunden, bevor die Schleife wieder von vorne beginnt. STOP in Zeile 90 wird benötigt, um das Programm davon abzuhalten, nach der vierten Schleife direkt in die Subroutine zu fahren.

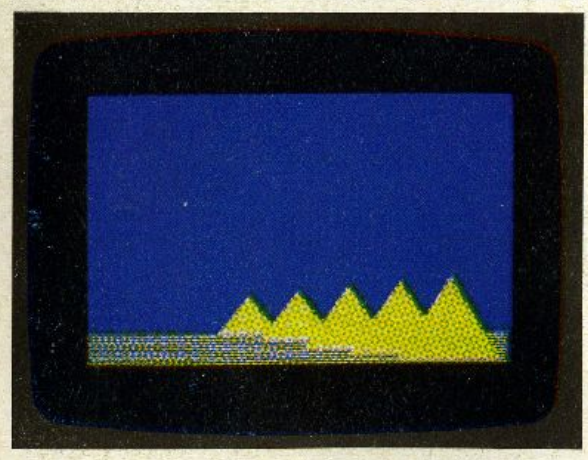
Sie können dieses Programm ändern, indem sie 4 in Zeile 30 ändern und 40 in Ziele 50. Wenn Sie die x-Spanne in Zeile 40 erweitern, erhalten Sie andere Zeichen auf dem Bildschirm.

Die Benutzung von FOR-NEXT-Schleifen in Grafiken

FOR-NEXT-Schleifen können in Hochauflösungsgrafiken sehr effektiv zur Erzeugung von Bildern verwendet werden. Tasten Sie das folgende Programm ein und fahren Sie es. Mit der ausschließlichen Verwendung von PLOT und DRAW zeichnen die zwei FOR-NEXT-Schleifen zuerst Linien auf dem Boden und dann fünf ausgefüllte Dreiecke oder Pyramiden.

PYRAMIDEN

```
 10 BORDER 0: PAPER 1: INK 6
 20 CLS
 30 FOR y=0 TO 20 STEP 2
 40 PLOT 0,y
 50 DRAW 255,0
 60 NEXT y
 70 FOR n=100 TO 220 STEP 30
 80 FOR x=-10-n/10 TO 10+n/10
 90 PLOT n,35+n/10
100 DRAW x,-n/4
110 NEXT x: NEXT n
```



Fügen Sie jetzt die Zeilen in der nächsten Spalte hinzu.
Wenn Sie das Programm wieder fahren, sehen Sie, daß ein Laserstrahl fortwährend in den Nachthimmel schießt und ein Explodieren von Sternen verursacht. Er wird von der Ecke des Bildschirms zu Position x,y gezeichnet, wobei die Variablen x und y Zufallszahlen sind.

```
120 LET x=RND*255
130 LET y=RND*104+71
140 LET l=INT (175-y)/8
150 LET c=INT (x/8)
160 PLOT 0,0: DRAW OVER 1;x,y
170 BEEP 0.01,x/4
180 PLOT 0,0: DRAW OVER 1;x,y
190 PRINT AT l,c;"*"
200 GO TO 120
```

OVER 1 in Zeilen 160 und 180 ermöglicht der ersten Zeile, den Laserstrahl zu zeichnen, und der zweiten Zeile, ihn zu entfernen, ohne daß sich das restliche Bild verändert. Bewahren Sie dieses Programm auf, da Sie es später benötigen.

FLASH, BRIGHT und INVERSE

Diese drei Schlüsselwörter haben einen großartigen Effekt auf die Farben Ihres Spectrums! Jedem Schlüsselwort folgt entweder 0 oder 1 und Sie können sie in PRINT-Befehle stellen, vorausgesetzt, Sie setzen ein Semikolon hinter die 0 oder 1. FLASH1 verursacht ein Blinken der Zeichenpositionen zwischen Ink- und Paperfarben, während BRIGHT1 die Farben heller macht. INVERSE1 ändert die Paperfarbe um in die Inkfarbe und umgekehrt. Die Benutzung von 0 nach diesen Schlüsselwörtern machen das Display wieder normal.

Führen Sie diese Veränderungen an den Programmen auf diesen zwei Seiten durch, um zu sehen, wie die Schlüsselwörter funktionieren. In dem Quadrate-Programm ändern Sie das Quadrat in Zeile 60 in einen Stern um und fügen Sie hinzu

```
15 INVERSE 1
```

Jetzt erscheint der Stern in schwarz (der Papierfarbe) gegen farbige Streifen (die sich ändernden Inkfarben). Geben Sie INVERSE0 ein, bevor Sie weitermachen.

Im Programm mit den symmetrischen Mustern fügen Sie die folgenden Zeilen hinzu, um zu sehen, wie BRIGHT und FLASH funktionieren.

```
15 BRIGHT 1
16 FLASH
```

Durch FLASH sieht es aus, als ob das Muster sich hin und her bewegt. Geben Sie FLASH 0:CLS ein, um das Blinken des Displays zu stoppen.

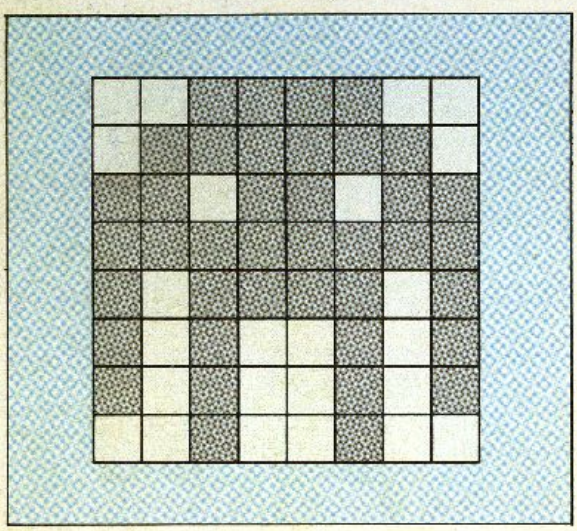
WIE MAN COMPUTERZEICHEN ERZEUGT

Ihr ZX Spectrum + ist nicht nur auf die über die Tastatur einzutippenden Grafikzeichen beschränkt. In einem speziellen Teil seines Speichers kann er noch andere Zeichen speichern, die Sie selbst entwerfen können. Diese heißen benutzerwählbare Grafikzeichen und jedes Programm kann maximal 21 davon haben.

Jedes Zeichen besteht aus bis zu 64 kleinen Punkten oder Pixeln der Inkfarbe. Diese sind in 8 Reihen mit je 8 Pixeln angeordnet und jedes Zeichen hat eine Zeichenposition auf dem Niedrigauflösungsgitter.

Das Entwerfen eines Grafikzeichens

Zuerst zeichnen Sie ein 8×8 Gitter, wie unten dargestellt. Füllen Sie dann einige der Quadrate aus, um das Zeichen zu erzeugen. Diese Quadrate stellen die Inkfarbenen Pixel dar. Dann zeichnen Sie ein – oder stellen Sie sich vor –, daß jedes volle Quadrat 1 ist und jedes leere Quadrat Ø. Hier das Design für eine Spinne.



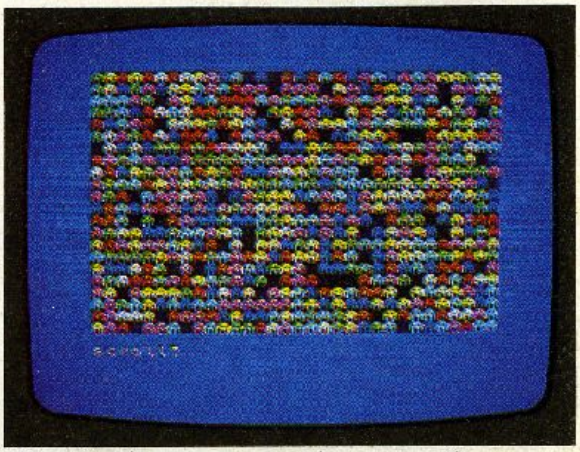
Jedes dieser benutzerwählbaren Grafikzeichen wird durch einen Buchstaben von a bis u (oder A bis U -das macht hier keinen Unterschied) gekennzeichnet. Um das Zeichen zu programmieren, geben Sie acht POKE USR Anweisungen ein, von denen jede auf BIN endet, gefolgt von einer Binärzahl, die aus den acht Ø und 1 in jeder Reihe des Gitters besteht. Nennen wir das Spinnenzeichen s.

```
10 POKE USR "s",BIN 00111100
11 POKE USR "s"+1,BIN 01111110
12 POKE USR "s"+2,BIN 11011011
13 POKE USR "s"+3,BIN 11111111
14 POKE USR "s"+4,BIN 10111101
15 POKE USR "s"+5,BIN 10100101
16 POKE USR "s"+6,BIN 10100101
17 POKE USR "s"+7,BIN 00100100
```

Fahren Sie dieses Programm nun und drücken Sie dann GRAPH und S. Anstelle eines S erscheint die Spinne! Fügen Sie die folgenden Zeilen hinzu, wobei die Spinne jetzt in Zeile 30 ist, und fahren Sie das Programm. Jetzt erscheinen Spinnen auf dem ganzen Schirm!

SPINNEN

```
20 BORDER 1: PAPER 0: CLS
30 PRINT INK RND*7; "s";
40 GO TO 30
```



Wenn Sie Ihre eigenen Zeichen entwerfen, denken Sie daran, daß Sie sie erst auf dem Bildschirm sehen können, wenn Sie ein Programm gefahren haben, welches sie definiert. Bis zu dem Zeitpunkt erscheinen sie nur in Listings als Buchstaben.

Wie man bei Quadraten Farben 'mischt'

Sie können sehr einfach Mischfarben auf Ihrem Spectrum simulieren. Um dies zu erreichen, müssen Sie ein Zeichen schaffen, welches, wenn es gedruckt wird, ein Quadrat erzeugt, das 50% Inkfarbe und 50% Paperfarbe hat. Sie müssen nur einfach zwei Pixelzeilen definieren und den Computer dann

```
10 FOR x=0 TO 6 STEP 2
20 POKE USR "a"+x,BIN 10101010
30 POKE USR "a"+x+1,BIN 01010101
40 NEXT x
```

anweisen, diese abwechselnd in einem Zeichen zu benutzen.

Wenn sie ein Programm fahren, sollten Sie ein gesprenkeltes Quadrat sehen. Wenn Sie ein Programm fahren, das Farbschlüsselwörter enthält, dann wird ein gesprenkeltes Quadrat in einer Farbe produziert, das eine Mischung aus den Paper- und Inkfarben des Programms ist.

Erleichterung von Grafiken durch READ/DATA

Es gibt eine leichtere Möglichkeit, Computergrafikzeichen zu erzeugen, und das ist die Verwendung von Dezimalzahlen mit READ und DATA. Zuerst ändern Sie die acht Binärzahlen in Dezimalzahlen um. Dies tun Sie, indem Sie PRINT BIN gefolgt von der Zahl eingeben, zum Beispiel

PRINT BIN 00111100.

Der Spectrum zeigt 60 an, das Dezimal-Äquivalent von 00111100. Im Falle der Spinne sind die acht Dezimalzahlen 60, 126, 219, 255, 189, 165, 165 und 36.

Jetzt können Sie READ und DATA benutzen. Diese zwei Schlüsselwörter geben Ihnen die Möglichkeit, viele Werte wie Zahlen in die Variablen in einem Programm einzuspeisen. READ wird von einer Variablen gefolgt-ein oder mehrere Buchstaben, wenn es sich um Zahlen handelt, oder ein einzelner Buchstabe gefolgt von einem $, wenn es sich um 'Strings' handelt.

Wenn Ihr Spectrum bei READ ankommt, sieht er sich die erste DATA-Anweisung im Programm an. Diese Anweisung enthält eine Liste von Werten, die durch Kommata getrennt sind. Der Computer nimmt den ersten Wert der Liste und gibt ihn der Variablen nach READ. Das nächste Mal, wenn der Computer bei READ ankommt, wird der Variablen der zweite Wert gegeben und so weiter, immer der Reihe nach.

Hier ist das neue Spinnen-Programm.

```
 10 FOR x=0 TO 7
 20 READ y
 30 POKE USR "s"+x,y
 40 NEXT x
 50 DATA 60,126,219,255,189,165
,165,36
```

Das Programm speichert beliebige acht Dezimalzahlen im Memory, um ein Zeichen zu erzeugen. Verändern Sie s in Zeile 30 in den Buchstaben um, den Sie wollen, und tasten Sie nach DATA in Zeile 50 die acht voneinander durch Kommata getrennten Zahlen ein. Drücken Sie GRAPH und den Buchstaben, um das Zeichen zu erhalten, nachdem Sie das Programm gefahren haben.

Zeichnung eines Schachbretts

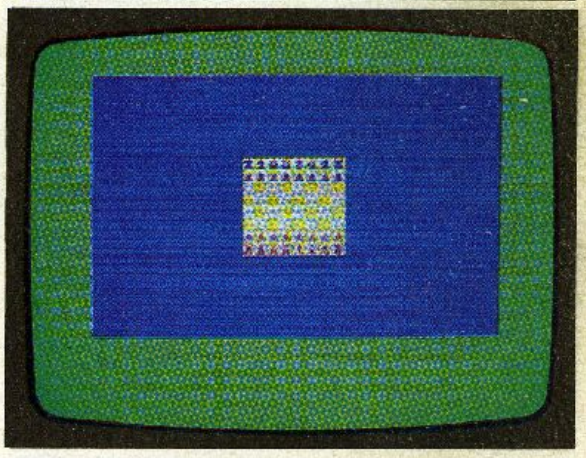
Hier ist ein Programm, das ein Schachbrett auf dem Bildschirm erzeugt und dann die Figuren spielbereit aufbaut. Sie können Farben in ein Listing bringen mit Hilfe der Farbsteuercodes – siehe untenstehende Tafel.

SCHACHBRETT

```
 10 FOR x=1 TO 8
 20 READ a$
 30 GO SUB 500
 40 NEXT x
 50 BORDER 4: PAPER 1: CLS
 60 FOR l=7 TO 14 STEP 2: FOR c
=16 TO 19 STEP 2
 70 PRINT AT l,c;" ";AT l+1,c;
"  "
 80 NEXT c: NEXT l
 90 PRINT AT 7,12;"RNBQKBNR"
100 PRINT AT 8,12;"PPPPPPPP"
110 PRINT AT 13,12;"PPPPPPPP"
120 PRINT AT 14,12;"RNBQKBNR"
130 GO TO 130
500 FOR n=0 TO 7
510 READ y
520 POKE USR a$+n,y
530 NEXT n
540 RETURN
```

Die Figuren werden durch eine Subroutine definiert, die bei Zeile 500 beginnt.

```
 550 DATA "p",0,0,16,56,56,16,12
4,0
 560 DATA "r",0,84,124,56,56,124
,124,0
 570 DATA "n",0,16,56,120,24,56,
124,0
 580 DATA "b",0,16,40,68,108,56,
124,0
 590 DATA "K",0,16,56,16,56,68,5
6,0
 600 DATA "q",0,84,40,16,108,124
,124,0
```



Die Verwendung von Farbcodes

Anstatt Schlüsselwörter wie INK und PAPER zu verwenden, können Sie Steuercodes in PRINT-Anweisungen einfügen nach den ersten Anführungsstrichen. Die darzustellenden Zeichen ändern dann ihre Farbe im Listing selbst und erscheinen auch in dieser Farbe auf dem Bildschirm.

Um die Codes zu erhalten, drücken Sie zuerst EXTEND MODE und dann eine Zahlentaste mit oder ohne CAPS SHIFT. Dieses Diagramm zeigt Ihnen, wie Sie die Farben wählen können.

Drücken	1	2	3	4	5	6	7	8	9	0
EXTEND MODE / CAPS SHIFT	blau Ink	rot Ink	violett Ink	grün Ink	hellblau Ink	gelb Ink	weiß Ink	Flash aus	Flash ein	schwarz Ink
EXTEND MODE	blau Paper	rot Paper	violett Paper	grün Paper	hellblau Paper	gelb Paper	weiß Paper	Bright aus	Bright ein	schwarz Paper

BEWEGUNG

Computergrafik sieht am besten aus, wenn sich die Zeichen oder Linien auf dem Schirm bewegen, und es ist nicht schwierig, Bewegung auf Ihrem Spectrum zu erzeugen. Sie müssen nur einfach immer wieder die Position ändern, an der sich ein Zeichen befindet oder eine Linie gezeichnet ist. Dies ist am einfachsten mit einer oder mehreren FOR-NEXT-Schleifen.

Vertikale und horizontale Bewegung

Das folgende Programm eintasten und fahren. Wenn Sie seit der Schaffung des Spinnengrafikzeichens auf Seite 32 *nicht* Ihren Spectrum zurückgestellt oder ausgeschaltet haben, brauchen Sie Zeilen 10 bis 50 nicht eingeben. Das Grafikzeichen ist immer noch im Speicher unter "s".

FALLENDE SPINNE

```
  5 BORDER 3: PAPER 5: CLS
 10 FOR x=0 TO 7
 20 READ y
 30 POKE USR "s"+x,y
 40 NEXT x
 50 DATA 60,126,219,255,189,165
,165,36
 60 FOR x=0 TO 7
 70 READ y
 80 POKE USR "t"+x,y
 90 NEXT x
100 DATA 16,16,16,16,16,16,16,1
6
110 FOR l=0 TO 20
120 PRINT AT l,3; INK 0;"|"
130 PRINT AT l+1,3; INK 2;"s"
140 NEXT l
```

Jedesmal, wenn Sie das Programm fahren, fällt die Spinne an ihrem Faden den Bildschirm herunter.

In diesem Programm entsteht in Zeilen 60 bis 100 noch ein weiteres Grafikzeichen ("t") für 'thread' (= Faden). Die Bewegung erfolgt in Zeilen 110 bis 140, die eine FOR-NEXT-Schleife bilden, in welcher l (die Zeilennummer) sich von 0 auf 20 verändert. Jedesmal, wenn die Schleife sich wiederholt, wird ein Stück Faden in einer Position gedruckt und die Spinne in der nächsten Position darunter. Beim nächsten Mal wird die Spinne durch ein weiteres Stück Faden ersetzt und erscheint wiederum darunter. Auf diese Art und Weise kann die Spinne schnell an ihrem Faden herunterklettern bis zu Zeile 21, welche das Ende des Displaybereichs darstellt.

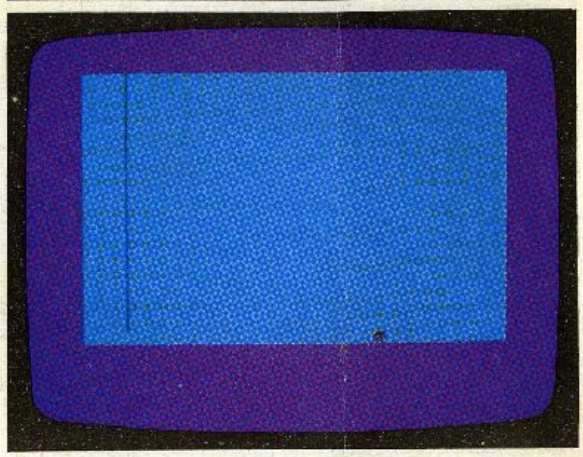
Um den Vorgang zu verlangsamen, setzen Sie die folgende Zeile ein:

```
135 PAUSE 10
```

Der Computer wartet jetzt jedesmal ein Fünftel einer Sekunde, bevor er die Spinne in der nächsten Position druckt. Fügen Sie jetzt Folgendes hinsu und fahren Sie das Programm noch einmal. Jetzt sehen Sie Bewegung, aber in einer anderen Richtung.

RENNENDE SPINNE

```
150 FOR c=3 TO 30
160 PRINT AT 21,c;" "
170 PRINT AT 21,c+1; INK 2;"s"
180 NEXT c
```



Jetzt rennt die Spinne auf eine Seite, sobald sie unten angekommen ist. Die Extrazeilen bilden eine weitere FOR-NEXT-Schleife, welche die Spalterıposition c steuert. Beachten Sie, daß zuerst eine Leerstelle gedruckt wird, und dann wird die Spinne bei der nächsten Spaltenposition dargestellt. Hierdurch verschwindet die Spinne von einer Position und erscheint in der nächsten, sich dabei nach rechts bewegend. Es ist immer besser, ein Zeichen zu löschen, bevor man es in der nächsten Position druckt. Hierdurch wird Flickern bei beweglichen Grafiken reduziert oder vermieden.

Zielübung

In vielen Computerspielen besteht die Aktion darin, daß zwei sich bewegende Figuren zusammenstoßen oder ein Objekt von einem Strahl getroffen wird.

Zusammenstöße sind nicht schwierig. Wenn zwei Zeichen bei Position l,c (für Zeile (line) und Spalte (column)) und Position v,h (vertikal und horizontal) gedruckt werden, dann müssen sie, wenn l=v und c=h, an derselben Position sein. Sie können dies als Statement schreiben, zum Beispiel

```
160 IF l=v AND c=h THEN PRINT
    "CRASH"
```

Eine andere Möglichkeit, Zusammenstöße zu finden, ist die Benutzung von Farben. Entfernen Sie das Spinnen-Programm durch Eingabe von NEW. Geben Sie dann das komplette Programm von Seite 31 ein (PYRAMIDEN) oder laden Sie das Programm, wenn Sie es auf Band gespeichert haben. Sie können es jetzt verbessern und mit Ihrer Spinne kombinieren (welche sich immer noch im Speicher befindet – sofern Sie nicht Ihren Spectrum zurückgestellt oder ausgeschaltet haben!) und erzeugen so ein neues Programm.

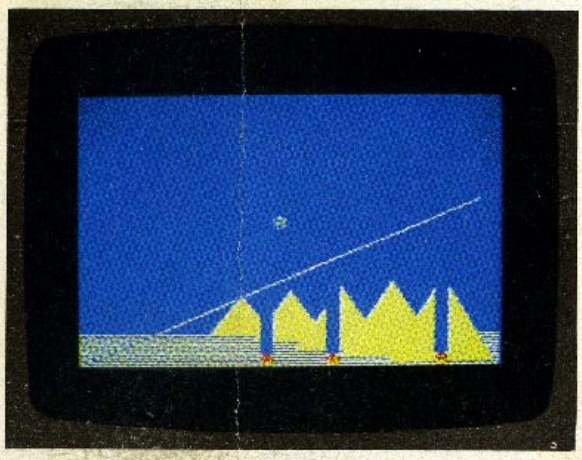
Fügen Sie zunächst die folgenden Zeilen hinzu, welche eine Explosionsgrafik mit dem Namen "e" erzeugen.

```
    5 FOR x=0 TO 7
    6 READ y
    7 POKE USR "e"+x,y
    8 NEXT x
    9 DATA 145,82,44,121,158,52,7
4,137
```

Löschen Sie jetzt Zeile 190 und addieren oder ändern Sie die folgenden Zeilen.

SPINNEN UND PYRAMIDEN

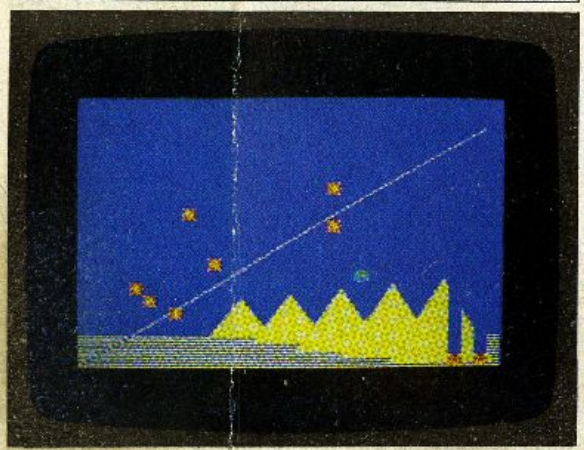
```
  114 LET h=RND*31
  115 FOR v=0 TO 20
  117 PRINT AT v,h;" ";AT v+1,h;
INK 4;"S"
  200 NEXT v
  205 PRINT AT 21,h; FLASH 1; INK
 2; PAPER 6;"S"
  210 GO TO 114
```



Jetzt erscheinen keine Sterne mehr. Stattdessen fallen Spinnen durch den Himmel und essen Pyramiden und Boden auf. Sie haben hier eine FOR-NEXT-Schleife hinzugefügt, in der v und h die Position der Spinne angeben. Die Variable h ist eine Zufallszahl, die Spinnen erscheinen also an den verschiedensten Stellen auf dem Schirm. Fügen Sie folgende Zeilen hinzu.

EXPLODIERENDE SPINNEN

```
  190 IF ATTR (v+1,h)=14 THEN GO
TO 500
  500 PRINT AT v+1,h; FLASH 1; PA
PER 2;"E"
  510 PAUSE 100
  520 GO TO 114
```



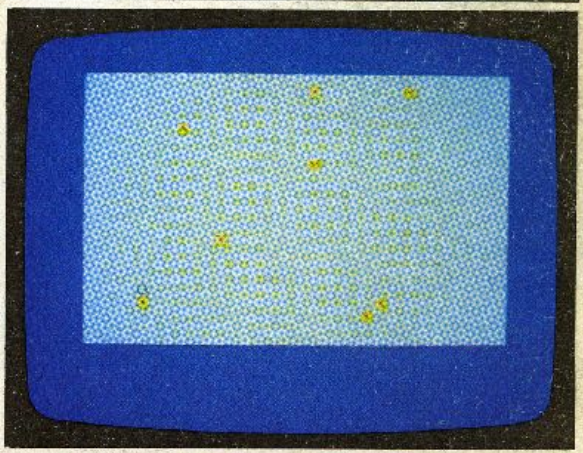
Wenn die Laserstrahlen die Spinnen treffen, werden sie kurz gelb. Wenn die durch DRAW erzeugte Linie ein Spinnenzeichen trifft, wird die Inkfarbe dieselbe wie die Farbe der Linie, welche gelb ist. In Zeile 190 entdeckt ATTR, ob die Spinne gelb wird, und sendet den Computer zur Explosions-Subroutine in Zeile 500.

Ein hüpfender Ball

Viele Grafikprogramme haben Figuren, die von den Seiten des Bildschirms abprallen. Dieses Programm zeigt Ihnen, wie man so etwas macht. Die Variablen v und h funktionieren genauso wie im Programm mit den explodierenden Spinnen, es werden jedoch +1 oder −1 zu v oder h hinzugefügt, damit der Ball rauf und runter geht und nach rechts oder links. SCREEN$ prüft, ob sich an Position v, h ein X befindet.

HÜPFENDER BALL

```
   10 BORDER 1
   20 FOR z=1 TO 10
   30 LET h=INT (RND*25): LET v=I
NT (RND*21)
   40 PRINT INK 2; PAPER 6; FLASH
 1;AT v,h;"X"
   50 NEXT z
   60 LET x=1: LET y=1
   70 PRINT AT v,h;" "
   80 LET v=v+y: LET h=h+x
   90 IF h=0 OR h=31 THEN LET x=-
x: BEEP .2,24
  100 IF v=0 OR v=21 THEN LET y=-
y: BEEP .2,12
  110 IF SCREEN$ (v,h)="X" THEN P
RINT INK 1; PAPER 5;AT v,h;"!*'*
.!": STOP
  120 PRINT AT v,h;"O"
  130 PAUSE 2
  140 GO TO 70
```



Die Verwendung von Attributen

Das Schlüsselwort ATTR entdeckt die 'Attribute' an einer bestimmten Stelle auf dem Bildschirm. Die Attribute sind die Ink- und Paperfarbe und ob die Position blinkt oder ob sie hell ist. Im Programm mit den explodierenden Spinnen, stellt ATTR sicher, daß die Spinne explodiert, wenn sie gelb wird. Dies ist dann ihre Inkfarbe (Nummer 6). Die Paperfarbe ist blau (Nummer 1). Das heißt, daß ihre Attribute insgesamt 14 ausmachen (6 für gelbe Schrift und 8 für blauen Hintergrund). Die Eintragung auf der Nachschlageseite für Programmierer zeigt Ihnen, wie diese Zahlen entstehen.

WIE MAN MUSIK MACHT UND SOUNDEFFEKTE ERZEUGT

Der ZX Spectrum + verfügt über einen Tonsynthesizer, der Ihre Programme mit einer Vielzahl von musikalischen Tönen und speziellen Soundeffekten beleben kann. Er ist leicht zu benutzen, selbst, wenn Sie nichts oder nur wenig von Musik verstehen. Der Synthesizer erzeugt ein Tonsignal, welches in den inneren Lautsprecher des Spectrum geht.

Das Programmieren von Tönen

Um Töne auf Ihren Spectrum zu erzeugen, brauchen Sie nur ein Schlüsselwort – BEEP. Ihm folgen zwei Zahlen oder Variable, die Zahlen darstellen. Die erste sagt dem Computer, wie lange der Ton dauern soll (in Sekunden) und die zweite sagt ihm, wie hoch oder tief der Ton ist. Die Höhe wird in Halbtönen gemessen. Die Höhenwerte sind 0 für das mittlere C, 1 für C# und −1 für B(Cb) und so weiter. Fahren Sie dieses Programm.

```
10 FOR P=69 TO -60 STEP -1
20 BEEP .2,P
30 PRINT AT 0,2;"   ";AT 0,0;P
40 NEXT P
```

Der Spectrum durchläuft seinen gesamten Notenbereich vom höchsten Ton (69) bis zum tiefsten (−60). Sie werden feststellen, daß die höchsten Töne kaum zu hören sind und die tiefsten wie Klickgeräusche klingen.

Das Diagramm unten auf dieser Seite zeigt die Höhenwerte einer Reihe von Noten, so daß Sie damit ein Musikstück vom Blatt in ein Spectrum-Programm verwandeln können.

Soundeffekt

Ihr Spectrum kann alle möglichen Soundeffekte erzeugen, indem normalerweise BEEP in eine Schleife gesetzt wird, die den Höherwert sehr schnell ändert. Versuchen Sie diese Programme.

Beachten Sie, daß die Dauerwerte sehr kurz sind, bis zu einem Hundertstel einer Sekunde. Drücken Sie BREAK, um die Schleifenprogramme zu beenden.

BLUBBERNDE GERÄUSCHE

```
10 LET P=INT (RND*40)-30
20 BEEP 0.05,P: BEEP 0.05,P+7:
BEEP 0.05,P+4
30 GO TO 10
```

Dieses Programm spielt eine Gruppe von drei Noten immer wieder in beliebigen Höhen. Die Höhenspanne ist breit. Sie können jedoch die Werte in Zeile 10 verändern, um die Spanne zu ändern.

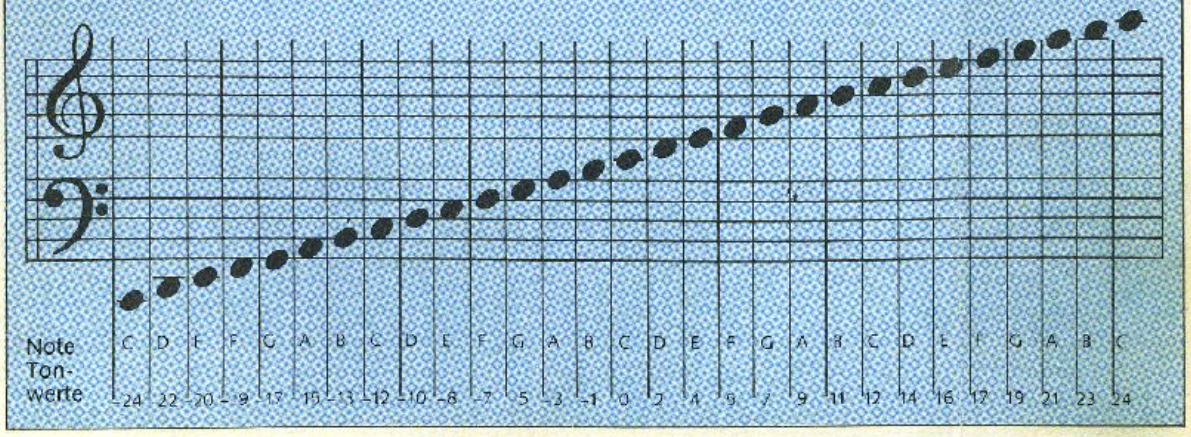
MASCHINEN

```
10 FOR x=12 TO 36
20 BEEP .01,x
30 BEEP .01,24-x
40 NEXT x
50 GO TO 10
```

Dieses Programm erzeugt zwei Töne, von denen einer nach oben geht und der andere nach unten. Das kommt daher, daß die zwei BEEP-Statements zwei Noten wieder und wieder mit nur einem Hundertstel einer Sekunde Abstand in verschiedenen Höhen ertönen lassen.

Tonwert zum Musikmachen

Hier sind die Spectrum-Werte von unten bis oben im Bass- und Diskantnotensystem. Bei Halbtonerhöhung 1 zum Höhenwert hinzufügen, bei Halbtonerniedrigung 1 abziehen.



ABHEBEN

```
10 FOR p=1 TO 48 STEP 0.2
20 BEEP .01,p: BEEP .01,p-6
30 NEXT p
```

Dieses Programm ähnelt dem Maschinenprogramm, jetzt gehen jedoch die zwei Tone zusammen hoch, sechs Halbtöne voneinander entfernt. Zusätzlich ändern sich die Höhenwert um 0.2 – einem Fünftel eines Halbtones- Das führt dazu, daß der Ton langsam höher wird. Versuchen Sie andere kleine Höhenveränderungen durch Änderung des STEP-Wertes.

TASTATUR-UMWANDLER

```
10 LET p=CODE INKEY$
15 IF p=0 THEN GO TO 10
20 BEEP .04,(p-30)/2
30 GO TO 10
```

Bei diesem Programm müssen Sie eine Taste anschlagen und jede erzeugt einen anderen Ton. Beachten Sie, daß ein Drücken von CAPS SHIFT bei gleichzeitigem Niederhalten einer anderen Taste den Ton tiefer macht. Dieses Programm funktioniert, weil CODE INKEY$ p jedesmal, wenn eine Taste gedrückt wird, einen anderen Wert gibt.

Sehen und Hören

Die von Ihrem Spectrum erzeugten Soundeffekte wirken am besten in Verbindung mit Aktion auf dem Bildschirm. Um zu zeigen, wie Sie Ton zu Programmen hinzufügen, gehen Sie zurück zum gesamten Rennende-Spinnen-Programm auf Seite 34.

Vielleicht erinnern Sie sich, daß Sie eine PAUSE-Anweisung in Zeile 135 eingefügt hatten, um den Vorgang zu verlangsamen. Anstatt den Vorgang auf diese Art zu verzögern, können Sie eine Pause programmieren, die einen Ton erzeugt. Verändern Sie Zeile 135 in

```
135 GOSUB 500
```

jetzt die folgenden Zeilen zu dem Programm.

```
200 STOP
500 FOR p=40-l TO 38-l STEP -1
510 BEEP 0.02,p
520 NEXT p
530 RETURN
```

Fahren Sie das Programm und die Spinne macht jetzt ein trillerndes Geräusch, als ob sie fällt. Die Subroutine spielt drei Noten sehr schnell, welche tiefer werden, je weiter sich die Spinne nach unten zu der nächsten Position auf dem Schirm bewegt. Versuchen Sie, noch mehr Noten hinzuzufügen, indem Sie Zeile 500 verändern, oder die Noten zu verschnellern oder zu verlangsamen, indem Sie 0.02 in Zeile 510 ändern.

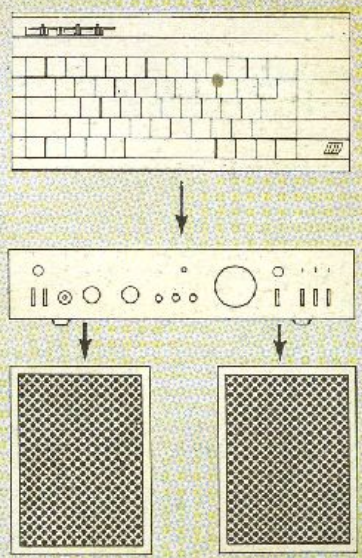
Wie Sie Ihren Spectrum verstärken können

Um den Sound Ihres Spectrum lauter zu machen, können Sie entweder die EAR- oder die MIC-Buchse mit einem Verstärker oder Lautsprecher verbinden.

Das Einfachste ist, wenn Sie das Spectrum-Kassettenkabel benutzen, um die EAR-oder MIC-Buchse mit der MIC-Buchse eines Kassettenspielers zu verbinden. Nehmen Sie die Kassette, wenn nötig, heraus, schalten Sie das Kassettengerät ein, und drücken Sie dann PLAY, REWIND (REVERSE) oder FAST FORWARD (CUE).

Stellen Sie die Lautstärke an Ihrem Kassettengerät ein und Sie müßten jetzt den Computerton aus dem Lautsprecher des Kassettenspielers hören. Sie können aber auch Kopfhörer mit dem Kassettenspieler benutzen.

Außerdem könnten Sie Ihren Spectrum an ein Hi-Fi oder Music Centre anschließen, wenn Sie einen vollen Klang haben möchten. Sie brauchen dann ein besonderes Kabel mit einem 3,5mm Klinkenstecker, der in den Spectrum paßt, und einen Stecker, der sich in die Eingangsbuchse am Hi-Fi Verstärker oder dem Music Centre einstecken läßt. Der Spectrum erzeugt ein Leitungssignal ähnlich dem bei Kassettenrekordern und Tonbandgeräten, die REPLY oder LINE IN Buchse am Verstärker müßte also funktionieren.



WIE SIE IHRE EIGENEN PROGRAMME SICHERN KÖNNEN

Es wird gar nicht lang dauern, und Sie werden Ihre eigenen Programme auf Kassette speichern wollen. Hierzu müssen Sie einen Kassettenspieler an Ihren Spectrum anschließen und 'sichern' damit das Programme im Computer. Wenn Sie das Programm das nächst Mal wieder brauchen, 'laden' Sie es von der Kassette zurück in den Computer mit Hilfe des auf Seiten 14–15 beschriebenen Ladeverfahrens. Auf diesen zwei Seiten erfahren Sie, wie man Programme sichert und wie man überprüft, daß sich richtig gesichert worden sind.

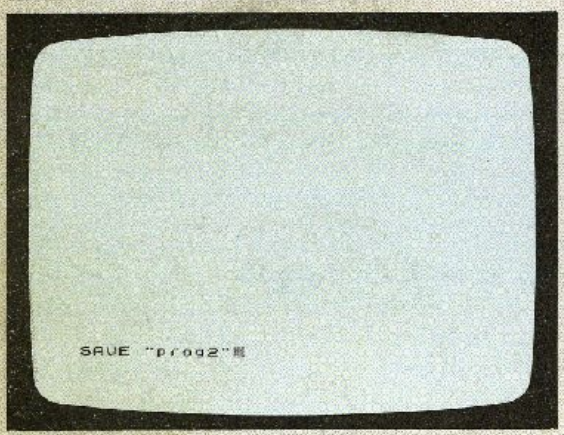
Das Aufnehmen Ihrer eigenen Programme

1 Zurst schließen Sie Ihren Spectrum an ein geeignetes Kassettengerät an mit Hilfe des Kassettenkabels – siehe Seite 14. Stellen Sie jedoch sicher, daß *nur* die MIC-Buchse des Spectrum mit dem Kassettenspieler verbunden ist.

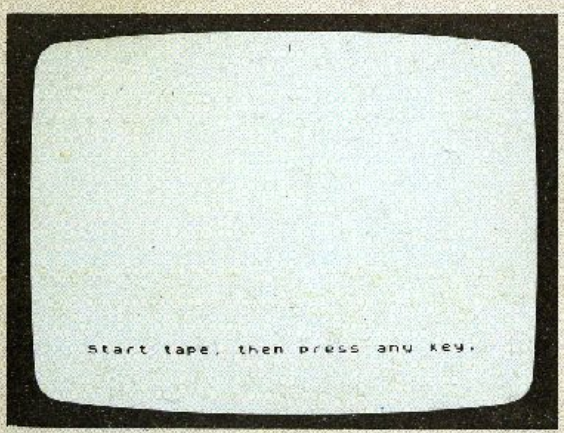
2 Wenn der Kassettenspieler einen Aufnahmepegel oder einen Lautstärkenregler hat, stellen Sie ihn auf ungefähr zwei Drittel der Maximaleinstellung. Ist dies nicht der Fall, machen Sie sich keine Gedanken, da der Aufnahmepegel automatisch eingestellt wird.

3 Tasten Sie SAVE ein, gefolgt von dem Namen des Programmes in Anführungsstrichen, zum Beispiel

SAVE "prog2"

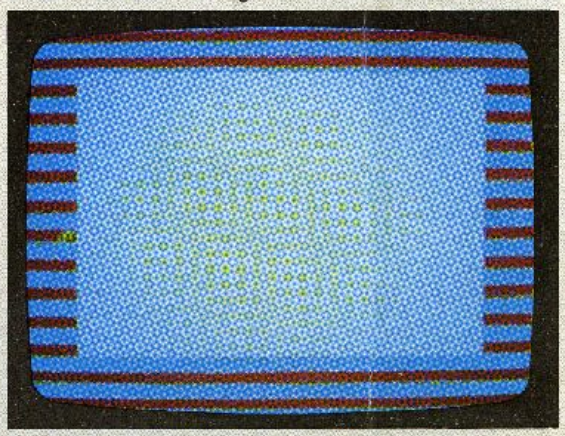


Jede Kombination von bis zu zehn Buchstaben und Zahlen kann verwendet werden. Drücken Sie jetzt ENTER. Die SAVE-Zeile verschwindet und dann sehen Sie die Kassettenarbeitsanweisung vom Spectrum.



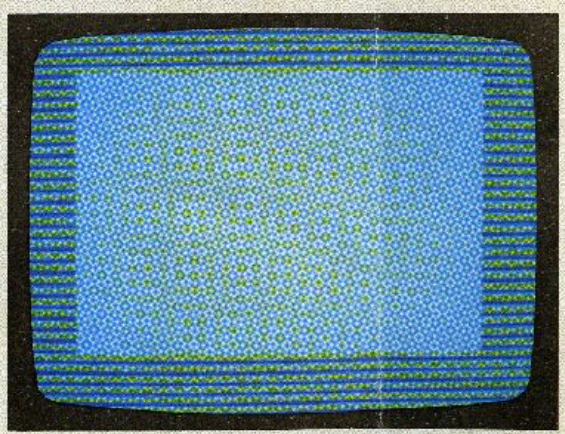
4 Stellen Sie den Kassettenrekorder auf Aufnahme, normalerweise, indem Sie RECORD und PLAY zusammen drücken. Schlagen Sie dann eine Taste auf dem Spectrum an.

5 Werden Sie jetzt, während der Spectrum das Programm sichert. Zuerst müßten Sie blaue und rote Streifen sehen, die sich langsam den Bildschirm hinaufbewegen.



Dann erhalten Sie kurz blaue und gelbe Streifen. Dies ist der Fall, wenn der Spectrum den Namen des Programmes an das Band sendet.

6 Danach kommt eine kurze Pause und dann weitere blaue und rote Streifen, gefolgt wiederum von der blauen und gelben Streifen, während der Spectrum jetzt das Programm an das Kassettengerät sendet. Ein langes Programm kann mehrere Sekunden dauern.

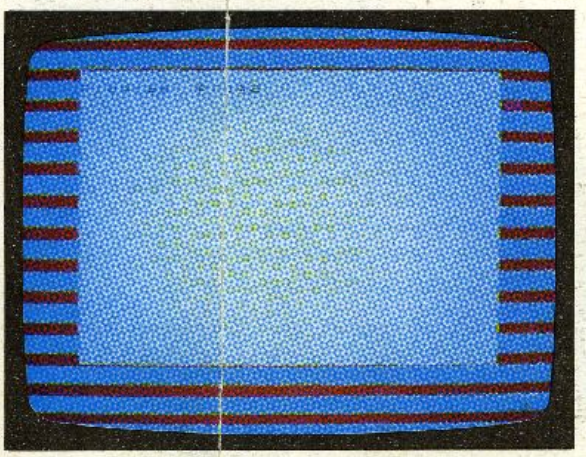


7 Wenn das Programm übertragen worden ist, erscheint die Mitteilung Ø OK,Ø:1. Halten Sie das Band an. Das Programm ist jetzt gesichert. Wenn Sie wollen, können Sie dies nun überprüfen oder 'bestätigen' (verify).

Bestätigung Ihres Programmes

Obwohl der Computer das Programm an den Kassettenrekorder gesandt hat, können Sie nicht sicher sein, daß das Programm richtig auf dem Band aufgenommen worden ist. Glücklicherweise kann Ihr Spectrum dies für Sie überprüfen.

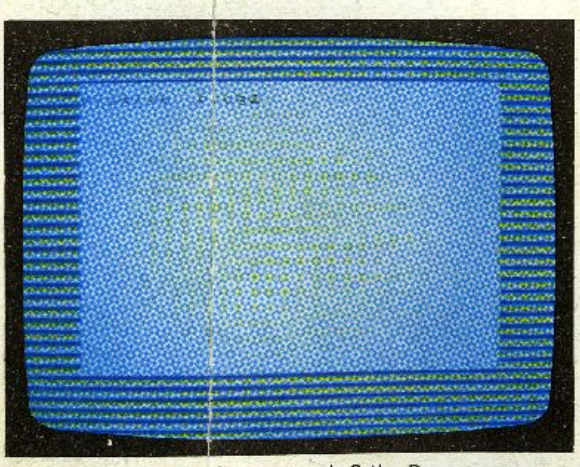
Diesen Vorgang nennt man 'verification' (Bestätigung). Stellen Sie zunächst das Band auf den Programmanfang zurück, schließen Sie dann die EAR-Buchse des Spectrums an die EAR-Buchse des Kassettengerätes an (Die MIC-Buchsen können verbunden bleiben). Tasten Sie nun VERIFY ein, gefolgt vom Programmnamen in Anführungsstrichen. Drücken Sie dann ENTER und beginnen Sie das Band. Jetzt müßte dieselbe Folge von roten und blauen Streifen und blauen und gelben Streifen erscheinen. Dann erscheint der Programmname und bleibt dort, bis der Bestätigungsprozeß beendet ist.



Wenn der zweite blaue und gelbe Teil, wie unten, endet, müßte die Mitteilung

0 OK,0:1

erscheinen. Dies bedeutet, daß Ihr Spectrum das Programm auf dem Band mit dem Programm in seinem Speicher verglichen hat und festgestellt hat, daß sie beide genau gleich sind.



Sie können nun sicher sein, daß Ihr Programm sich sicher auf dem Band befindet.

Tips zur Ersparnis von Software

1. Schreiben Sie den Namen eines Programmes auf das Kassettenetikett oder die Karte, wenn Sie es sichern. Benutzen Sie dieselben großen oder kleinen Buchstaben, die auf dem Schirm erscheinen. Wenn der Kassettenrekorder einen Zähler hat, benutzen Sie ihn zum Finden von Programmen und schreiben Sie die Zählerzahl zum Namen.
2. Vor dem Sichern stellen Sie den Programmnamen in das Programm, indem Sie eine REM-Anweisung benutzen, zum Beispiel

```
5 REM SPINNE Programm Version 3
```

Der Computer ignoriert alle REM-Anweisungen, wenn das Programm läuft, und Sie können REM verwenden, um Anmerkungen oder Hinweise an beliebige Stellen im Programm zu setzen.

Wenn Sie diese Mitteilung nicht erhalten, ist etwas schiefgegangen. Lesen Sie zuerst die Fehlersuchtabelle auf Seite 16, da der Fehler darin bestehen kann, daß das Programm zwar sicher auf dem Band ist, daß es jedoch nicht in den Computer zur Bestätigung zurücklädt. Wenn Sie den Fehler hier finden, korrigieren Sie ihn, stellen Sie das Band wieder zurück und bestätigen Sie das Programm noch einmal. Wenn der Computer dies immer noch nicht tut, lesen Sie die Fehlersuchtabelle auf der nächsten Seite. Drücken Sie nicht NEW und stellen Sie den Computer nicht zurück oder schalten ihn ab, weil Sie dann das Programm im Speicher verlieren werden, ohne eine zuverlässige Kopie auf Band zu haben.

Automatischer Programmbeginn

Sie können SAVE den Programmnamen und dann LINE1 folgen lassen, zum Beispiel

```
SAVE "SPINNE" LINE1
```

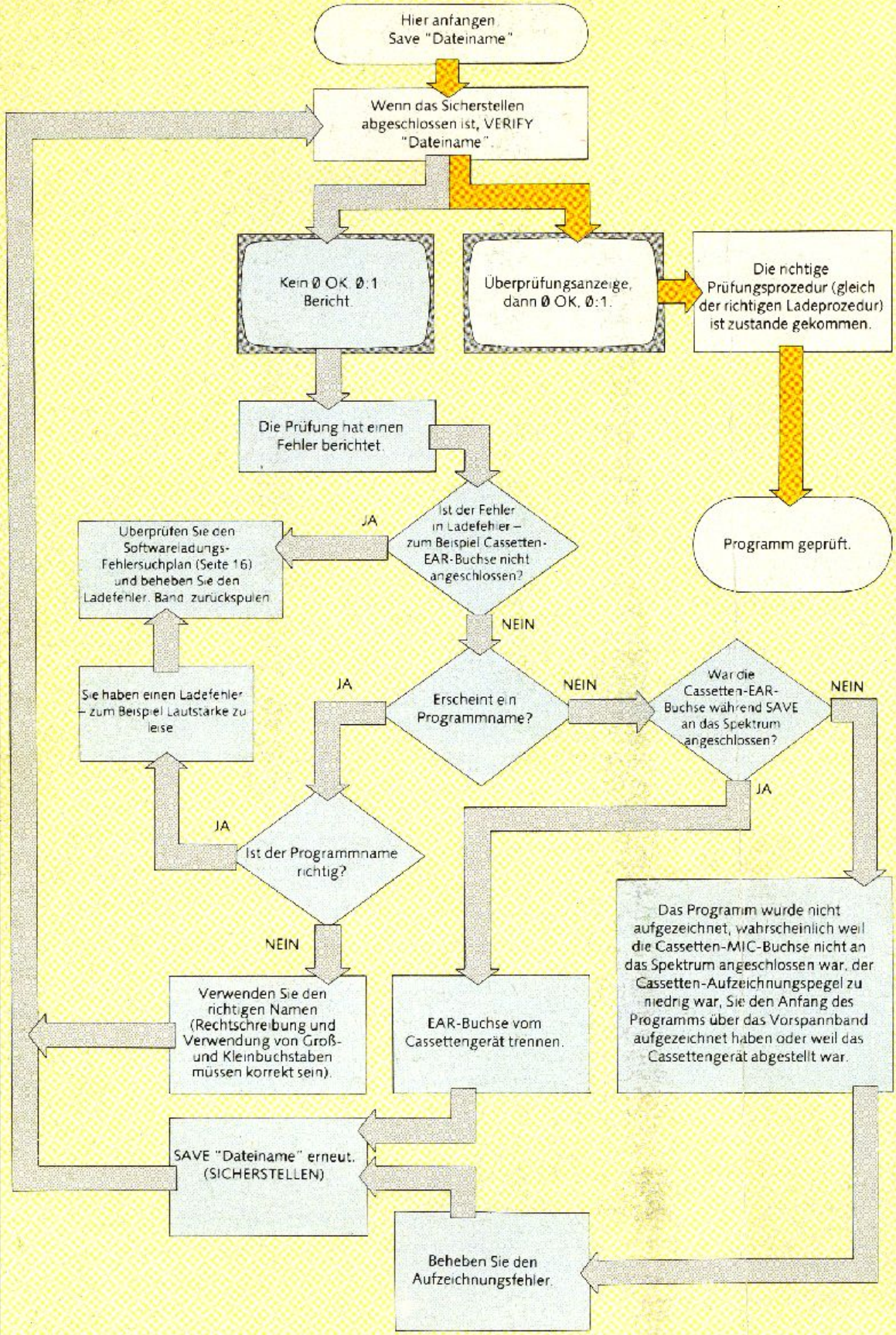
Der Sicherungsvorgang ist nicht anders als vorher, wenn Sie jedoch bestätigen, schreiben Sie LINE1 *nicht* nach VERIFY und dem Programmnamen. Mit LINE1 gesicherte Programme beginnen automatisch, wenn Sie sie laden. Man braucht RUN nicht benutzen (denken Sie jedoch daran, das Band anzuhalten, wenn das Programm anfängt).

Das Programm beginnt bei Zeile 1 und wenn es keine Zeile 1 gibt, bei der ersten Zeile im Programm. Verändert man 1 in eine andere Zahl, beginnt das Programm automatisch bei der Zeile mit derjenigen Zahl.

Sicherung von CODE, SCREEN$ und DATA

SAVE kann auch in Verbindung mit CODE oder SCREEN $ benutzt werden, um einen Teil des Spectrum Speichers zu speichern, oder mit DATA, um einen Array (Feld, Tabelle) zu speichern. Siehe SAVE CODE, SAVE SCREEN$ und SAVE DATA auf der Nachschlageseite für Programmierer.

Softwaresicherstellungs-Fehlersuchplan



ALLES ÜBER IHREN ZX SPECTRUM +

Dieses Kapitel befaßt sich mit dem Inneren Ihres Spectrums und erklärt, wie die verschiedenen Komponenten unter der Tastatur funktionieren und wie sie zusammen wirken, damit der Computer funktioniert. Außerdem wird erklärt, wie Sie 'Peripheriegeräte' benutzen können – zusätzliche Vorrichtungen, durch die Ihr Spectrum zu einem ganzen Computer-System wird. Und schließlich finden Sie in diesem Kapitel auch Erklärungen zu der technischen Seite Ihres Computers und zu der Organisation des Speichers, plus technische Daten des Spectrum.

DAS INNERE DES SPECTRUMS

Versuchen Sie *nicht*, Ihren ZX Spectrum + zu öffnen, um herauszufinden, wie er funktioniert! Sie machen damit Ihre Garantie ungültig und könnten ernste Schäden verursachen.

Innerhalb des Gehäuses befinden sich zwei Bandverbindungsglieder, die die Tastatur mit den übrigen Komponenten des Spectrum verbinden. Diese befinden sich alle auf einer einzigen gedruckten Schaltplatte. Die Platte trägt elektrische Standardteile wie Widerstände und Kondensatoren, die wichtigsten Elemente sind jedoch die schwarzen, rechteckigen Microchips, die entweder einzeln oder in Blöcken angeordnet sind.

Das Innere eines Chips

Der Arbeitsteil eines Microchips ist in Wirklichkeit viel kleiner als das Plastikgehäuse, in dem er sich befindet. Das Gehäuse ist vor allem dazu da, alle vom Chip benötigten Anschlüsse zu tragen, so daß er in Buchsen auf der Printplatte gesteckt werden kann. Der Chip selbst ist ein dünnes Siliziumplättchen mit Tausenden von elektrischen Knotenpunkten. Jeder dieser Knotenpunkte dient dazu, ankommende Signale anzuhalten, weiterzugeben oder zu speichern. Obwohl dies ein einfacher Vorgang ist, gibt es so viele von diesen Anschlußpunkten, die alle zusammen arbeiten, daß sie Signale erzeugen können, die Informationen mit erstaunlicher Geschwindigkeit und Genauigkeit speichern oder verarbeiten.

Wie die Chips miteinander verbunden sind

Ihr Spectrum ist also im Grunde eine elektrische, sehr komplexe, Schaltung. Aus elektrischen Impulsen bestehende Codesignale bewegen sich mit höchster Geschwindigkeit auf den Wegen in den Chips und zwischen Chips und anderen Komponenten hin und her, damit der Computer funktioniert.

Wie ist alles so angeordnet, daß das richtige Signal zur richtigen Zeit an der richtigen Stelle ankommt? Versteckt in einem der Chips befindet sich eine Uhr. Sie funktioniert durch das Aussenden von elektrischen Impulsen – 3,5 Millionen pro Sekunde! Diese Impulse bewegen sich regelmäßig durch die Schaltung, um die Codesignale zu erzeugen, die die Arbeit jedes Teils steuern und alles im Tritt halten.

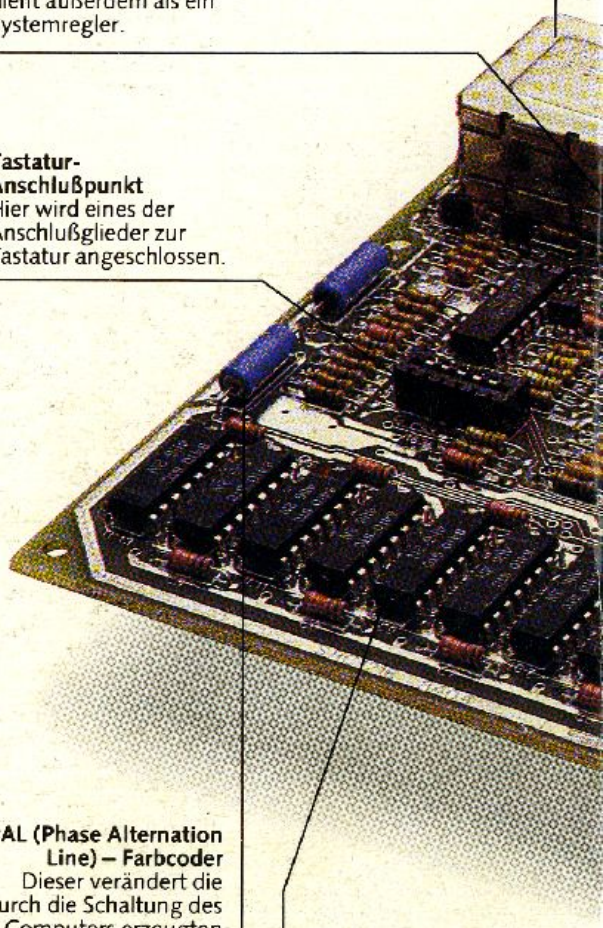
Das Innere Ihres Spectrums
In dieser Ansicht der Printplatte sind die zwei Bandverbinder zur Tastatur entfernt worden.

Wenn der Spectrum in Betrieb ist, bringt das Drücken einer Taste zwei sich unter der Tastatur befindlichen Drähte in Kontakt. Dies sendet ein Codesignal an die Zentraleinheit CPU.

Uncommitted Logic Array ULA (=freie logische Anordnung)
Dieser Chip erzeugt das Display aus den in RAM gehaltenen Informationen und dient außerdem als ein Systemregler.

TV-Ausgang
Dieser erzeugt das Signal, das zum Fernseher geht.

Tastatur-Anschlußpunkt
Hier wird eines der Anschlußglieder zur Tastatur angeschlossen.



PAL (Phase Alternation Line) – Farbcoder
Dieser verändert die durch die Schaltung des Computers erzeugten Signale um in ein Farbfernseher-Signal.

Random Access Memory (RAM)
Diese Chips enthalten das Programm, das in den Computer gegeben wird, und alle vom Programm benötigten, besonderen Informationen, wie zum Beispiel Werte von Variablen. Der Inhalt der 48K des RAM kann von der Tastatur aus geändert werden und kann vollständig gelöscht werden, indem der Computer zurückgestellt oder abgeschaltet wird.

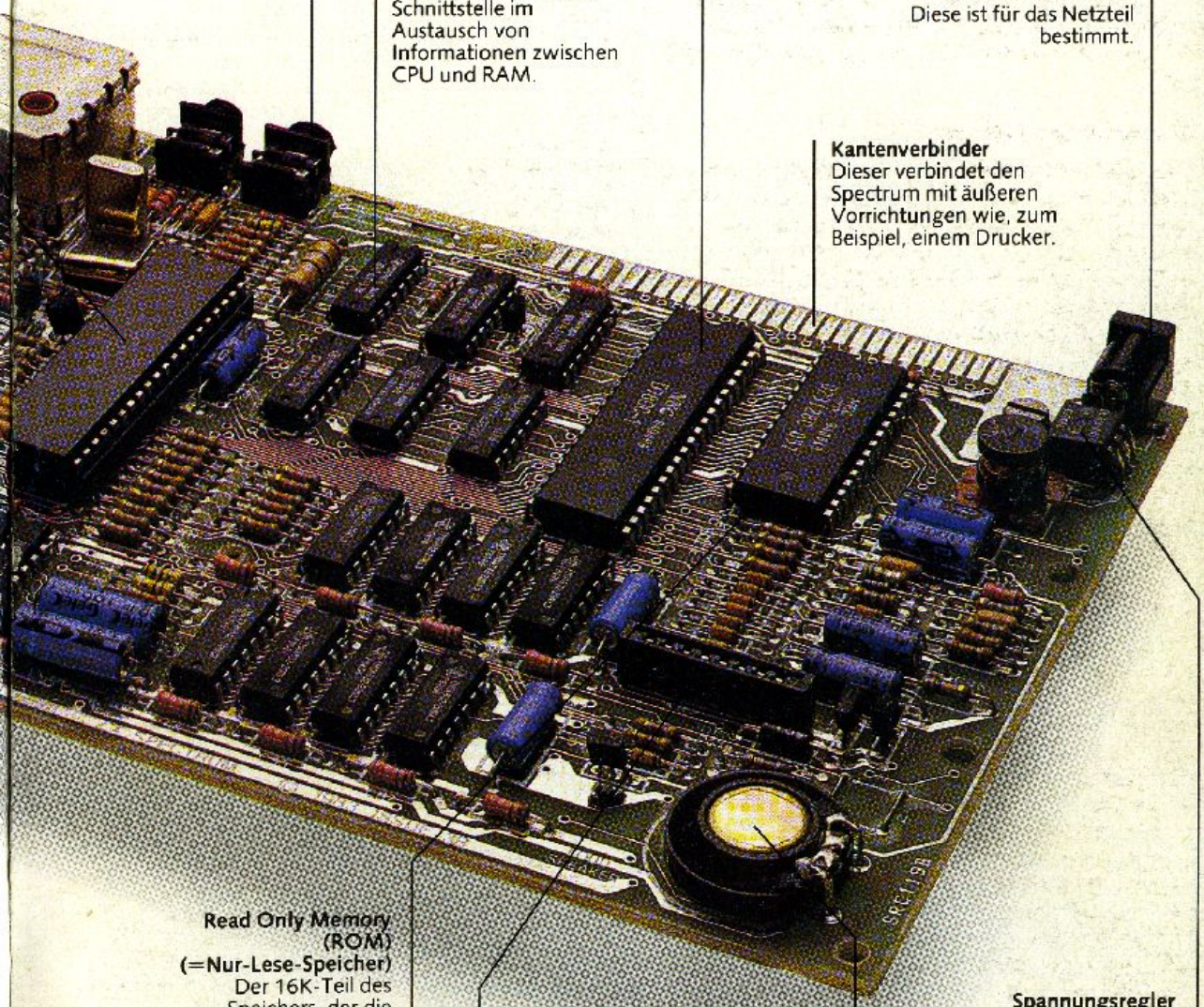
Kassettenbuchsen
Diese werden benutzt, um Informationen und Programme vom Speicher auf Band zu senden und sie vom Band aus wieder in den Speicher zurückzuspeisen.

Central Processing Unit CPU (Zentraleinheit)
Das 'Gehirn' des Computers. Der CPU ist ein Z80 Mikroprozessor. Er führt alle Rechenoperationen aus und steuert das allgemeine Funktionieren des Spectrum.

Logic Chips
Diese Chips dienen als Schnittstelle im Austausch von Informationen zwischen CPU und RAM.

9-Volt Gleichstrombuchse
Diese ist für das Netzteil bestimmt.

Kantenverbinder
Dieser verbindet den Spectrum mit äußeren Vorrichtungen wie, zum Beispiel, einem Drucker.



Read Only Memory (ROM) (=Nur-Lese-Speicher)
Der 16K-Teil des Speichers, der die permanenten Arbeitsanweisungen für den CPU enthält. Unter anderem verwandeln diese Anweisungen BASIC-Programme in eine Form, die der CPU verstehen kann. Der Inhalt dieses Speichers kann nicht durch die Tastatur geändert werden.

Tastatur-Anschlußpunkt
Hier wird einer der Bandverbinder zur Tastatur angeschlossen.

Lautsprecher
Dieser erzeugt, wenn erforderlich, die Töne.

Spannungsregler
Diese Komponente sorgt dafür, daß Änderungen in der Spannung nicht den Computer beeinträchtigen.

WIE FUNKTIONIERT IHR ZX SPECTRUM +

Der arbeitende ZX Spectrum + besteht, wie jedes andere Mikrocomputersystem, aus vier Hauptteilen. Dies sind die *'Input-Elemente'* wie die Tastatur, durch die Informationen oder ein Programm in den Computer gegeben werden; die temporären und permanenten *'Speicher'* (memories), die Informationen, Programme und Arbeitsanweisungen speichern; die Zentraleinheit (Central Processing Unit, CPU), welches die Programmanweisungen über die Informationen ausführt, und die *Output-Elemente*, die das Ergebnis bringen.

Eingabe und Fahren eines Programmes

Was geschieht im Spectrum, wenn Sie ein einfaches Programm eingeben und fahren? Hier ein einzeiliges Beispiel:

```
10 PRINT 6+2
```

Zuerst betätigen Sie die Tastatur. Unter den Tasten befindet sich ein Gitter sich überkreuzender Drähte. Jedesmal, wenn Sie eine Taste drücken, entsteht ein Kontakt zwischen zwei Drähten und diese senden ein Codesignal an die Zentraleinheit CPU. Die Zentraleinheit wiederum sendet den Code an RAM, wo er gespeichert wird.

Wenn Sie das Programm fahren, nimmt die Zentraleinheit die gespeicherten Codes aus RAM, immer der Reihe nach, wie es im Programm steht. Zuerst empfängt sie den Code für PRINT, welcher ihr mitteilt, daß sie einen bestimmten Arbeitscode aus ROM nehmen muß. Dieser Arbeitscode geht an die Zentraleinheit und die Zentraleinheit bereitet jetzt die Vorgänge vor, die einen Wert auf den Bildschirm bringen. Als nächstes erhält die Zentraleinheit den Wert 6 aus RAM. Dies ist ebenfalls ein Code und die Zentraleinheit speichert ihn in einem kleinen inneren Speicher, welcher 'Register' genannt wird. Danach kommt der Code für die Addition und wieder erhält die Zentraleinheit den nötigen Arbeitscode von ROM. Als letztes nimmt die Zentraleinheit den Code für 2 aus RAM. Sie addiert diesen Code zu dem Wert im Register und erhält das Ergebnis (8). Die Zentraleinheit verwandelt dann das Ergebnis in einen anderen Code und die Zahl 8 erscheint auf dem Bildschirm.

Speichern eines Programmes

Wenn Sie den Spectrum anweisen, das Programm auf Band zu sichern, nimmt die Zentraleinheit wieder die Codes aus RAM. Anstatt sie jedoch zu befolgen, sendet sie die Codes an einen Umwandler, welcher sie in Tonsignale verwandelt. Diese Signale werden dann an das Kassettengerät gesandt und auf Band aufgenommen.

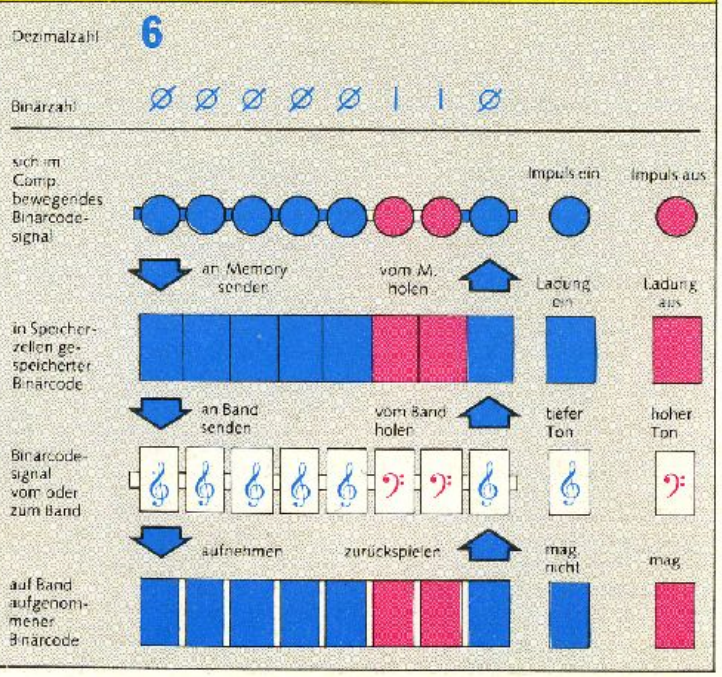
Wenn Sie das Programm später laden, werden die Tonsignale vom Band wieder in die Computercodes zurückverwandelt durch den Umwandler. Die Zentraleinheit sendet sie zurück an RAM, wo sie gespeichert bleiben, bis sie gebraucht werden.

Binärcodes

All die Codes für Ihren Spectrum sind in 'Binärform'. Sie heißen Binärcodes, weil sie alle aus nur zwei Arten von Signalen bestehen. Sie können als Binärzahlen dargestellt werden, das heißt, Zahlen, die nur zwei Ziffern enthalten – **0** und **1**. Die Binärzahl für 6 ist, zum Beispiel, **00000110**.

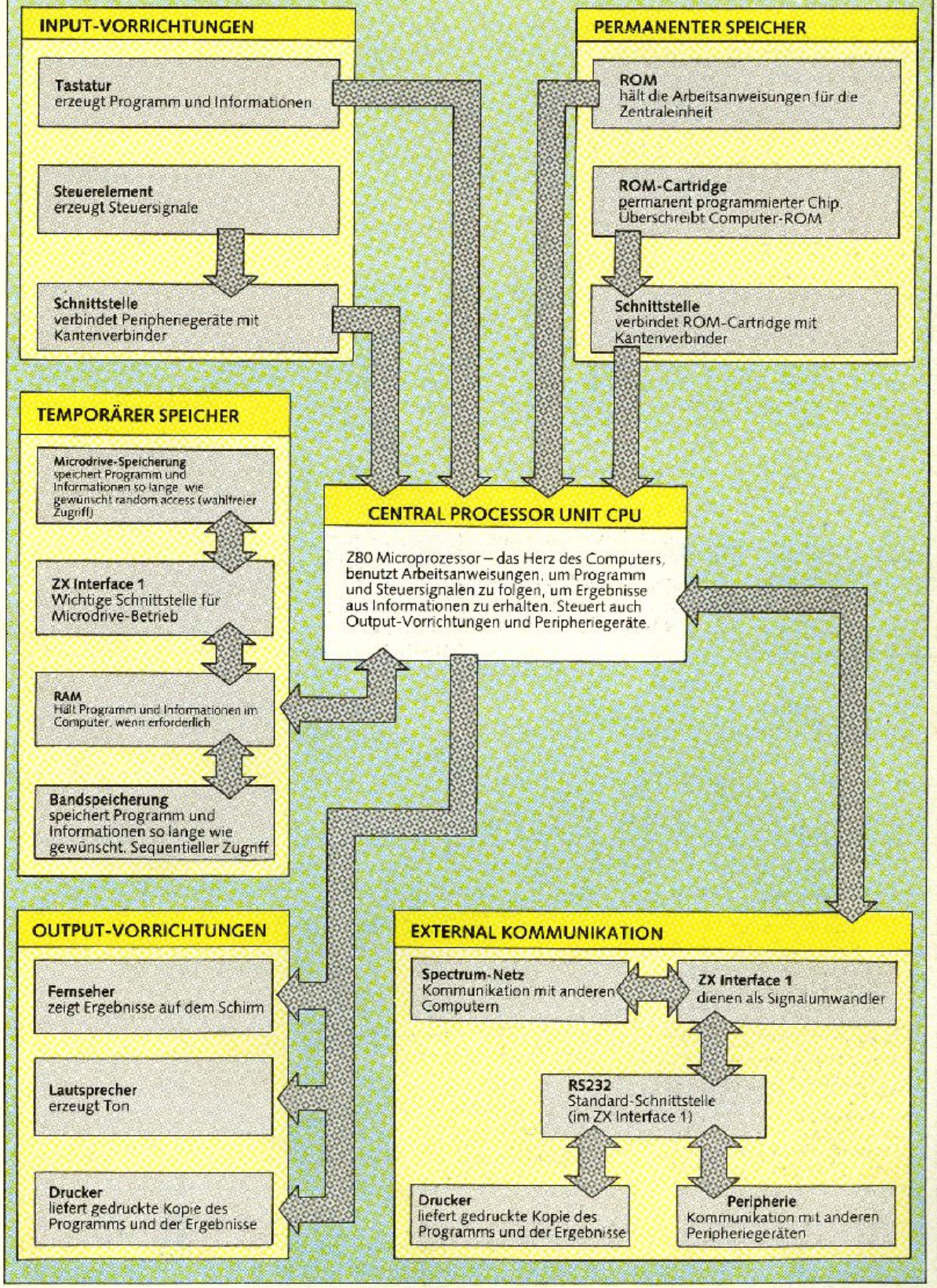
Innerhalb Ihres Spectrum bestehen die Codes aus schnellen, elektrischen Impulsfolgen. Wenn ein Impuls an einem Punkt ankommt, stellt dieser kurze Elektrizitätsmoment eine 1 in Binärform dar. Wenn ein Impuls nicht innerhalb einer gegebenen Zeit ankommt, stellt dies eine 0 dar. In Computercodes ist 6 daher *aus-aus-aus-aus-aus-ein-ein-aus*.

Sie können in diesem Diagramm sehen, wie verschiedene Arten von Binärcodes vom Computer benutzt werden, um Informationen von einer Stelle an eine andere zu bringen.



Input-zu-Output – die Wege im Spectrum

Dieses Diagramm zeigt, wie Informationen von Eingabevorrichtungen wie der Tastatur durch das Verarbeitungssystem des Spectrum laufen und dann zu Output-Vorrichtungen wie einen Fernsehbildschirm. Einköpfige Pfeile geben die Wege an, die nur in einer Richtung wirken. Pfeile mit zwei Köpfen zeigen Wege, die in beide Richtungen verwendet werden.



WIE MAN PERIPHERIEGERÄTE ANSCHLIESST

Sie können mit Hilfe von Sinclair und anderen passenden Peripheriegeräten Ihren Spectrum in ein komplettes und effektives Computersystem verwandeln. Im Zentrum dieses Systems steht die Schnittstelle ZX Interface 1, durch die Sie Microdrives zur schnellen und einfachen Handhabung von Programmen und Daten anschließen können und durch die sich außerdem eine Reihe von anderen Peripheriegeräten, einschließlich anderer Spectrum, anschließen lassen. Mit dieser Schnittstelle können Sie Ihren Spectrum an einen Drucker anschließen und den Computer an ein Modem anhängen. Es stehen noch andere Schnittstellen zur Verfügung, die die einsteckbaren ROM-Cartridges zu sofortigen Programmladung mit dem Computer verbinden. Mit ihnen können Sie auch Steuerelemente befestigen, damit Spiele mehr Spaß machen.

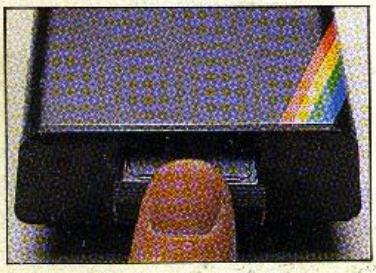
Mit dem Spectrum kompatible Drucker

Einige Drucker lassen sich direkt in den Kantenverbinder des Spectrum einstecken. Wenn Sie bereits einen Sinclair ZX Drucker besitzen, können Sie ihn ohne Schnittstelle an den Computer anschließen. Um jedoch Drucker zu benutzen, die einen RS232-Ausgang brauchen, müssen Sie die D-Buchse an der ZX Schnittstelle 1 benutzen.

ZX Interface 1

Die ZX Interface 1-Vorrichtung ist hinten am Spectrum unten angeschlossen. Es verbindet Ihren Spectrum mit bis zu 8 Microdrives, bis zu 63 anderen Spectrum-Comptern und durch seine standardmäßige RS232-Schnittstelle mit einer riesigen Auswahl von standardmäßigen Peripheriegeräten.

Microdrives und Microdrive Cartridge ersetzen Kassettenspieler und Bänder beim Speichern von Programmen und Daten. Durch das Einschieben von Microdrive Cartridges können Sie innerhalb von Sekunden Programme sichern, bestätigen und laden. Jede Diskette kann bis zu 85K Daten



Das Laden eines Microdrives
Microdrive Disketten werden in den Schlitz vor dem Antriebelement eingesteckt.

Das Einstecken eines ROM-Cartridge
Das Cartridge wird einfach in die Schnittstellenbuchse eingesteckt. Wenn der Computer eingeschaltet wird, wird das Programm automatisch geladen, wobei es ROM im Computer umgeht.

Anschluß der Peripheriegeräte
Das ZX Interface 1 ist immer in den Kantenverbinder eingesteckt, so daß es sich unter und hinter dem Computer befindet. Die Abbildung hier zeigt das System vor dem Anschluß des Computers.

Microdrive-Elemente
An einen Spectrum können bis zu acht dieser Speicherelemente angeschlossen werden.

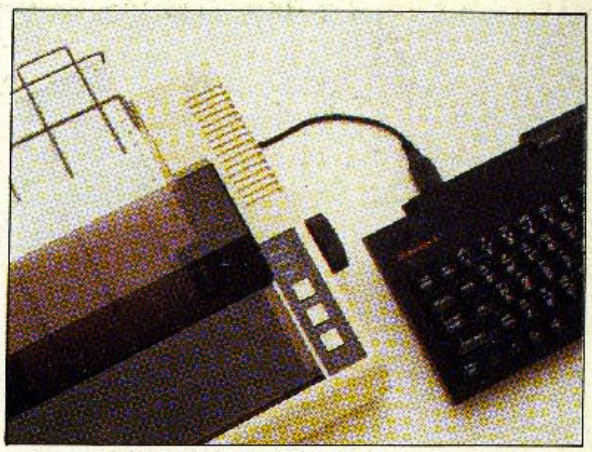
Bandkabel
Dieses verbindet den Microdrive über das ZX Interface 1 mit dem Computer.

Bermerkung
Der ZX Spectrum + hat zwei eingebaute Füsse um die Taste zu heben. Diese werden nicht benötigt wenn ein ZX Interface montiert ist.

speichern und wenn Sie alle acht Microdrives benutzen, hat Ihr Spectrum eine Speicherkapazität von 680K! Jedes Programm wird automatisch innerhalb einer üblichen Zugriffszeit von 3,5 Sekunden gefunden!

Das mit der Schnittstelle mitgelieferte Netzkabel kann Ihren Computer mit einem anderen Computer verbinden – entweder einem ZX Spectrum oder einem weiteren ZX Spectrum +. Dieses Netz kann auf maximal 63 Spectrum erweitert werden!

Der ZX Interface 1 Aufbau enthält auch ein RS232-Interface mit einer 9-Wege D-Buchse. Ein standardmäßiges Interfacekabel steht ebenfalls zur Verfügung.



Standardmäßige Drucker werden durch das ZX Interface 1 angeschlossen.

ROM-Cartridge und Steuerelement

Schnittstellen wie das ZX Interface 2 ermöglichen es Ihnen, Rom-Cartridge und Steuerelement anzuschließen. ROM-Cartridges laden sofort beim Einschalten und Sie erhalten Programme, deren Laden von Band sehr lange dauern würde.

ROM-Steuerelement Interface

Kantenverbinder
Durch dieses Anschlußteil werden Peripheriegeräte mit dem Computer verbunden

Warnung
Peripheriegeräte sollten immer verbunden werden bevor Strom einschaltung.

DER ZX-SPECTRUM UND SPEICHERTABELLE

Sehen sie sich das Foto auf Seite 42-43 an, werden sie sehen, dass dort ein ROM-Chip und 16 kleinere RAM-Chips zu sehen sind. Diese Chips geben dem Spectrum seinen Speicher. Dieser Speicher besteht aus 65936 Speichereinheiten, von denen jede ein Byte enthält. (also eine Zahl von Ø-255) Jede Einheit ist durch eine Zahl, der sogenannten Adresse gekennzeichnet.

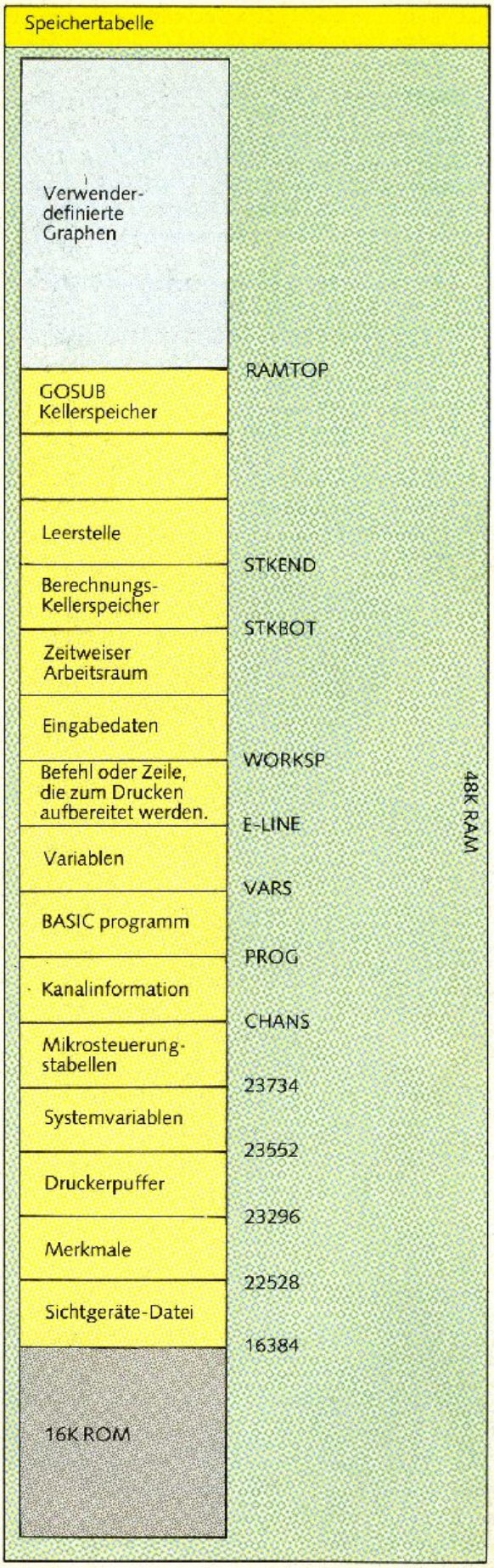
ROM steht für *Read Only Memory*, das heißt also, dieser Teil des Speichers enthält die Befehlsanweisungen für die Zentraleinheit. Es ist eine 16k ROM, was bedeutet, dasß sie 16×1024 (16384) bytes oder Adressen enthält. Diese bytes können nur von diesem Speicher gelesen werden, was bedeutet, daß sie nicht verändert werden können. (Wäre dies der Fall, so würde der Komputer aufhören, zu arbeiten.) Sie können jedes byte von jedem Speicher erhalten, indem PEEK verwendet wird.

RAM steht für *Random Access Memory*. Diese Einheit enthält das Programm und die Informationen, die dem Komputer eingegeben wurden. Der Spektrum besitzt einen 48k RAM, das heißt also, er besitzt 48×1024 (49152) bytes oder Adressen. Random Access bedeutet, daß ein byte in jeder Adresse verändert werden kann, was durch die Verwendung von POKE erreicht wird.

Diese Speichenadressen haben eine Bandbreite von Ø bis 65535, wobei das erste Viertel ROM und der Rest RAM ist.

Systemvariabeln

Die Sparte gegenüber zeigt; wie der Speicher des Spektrum aufgebaut ist. Dort sehen Sie, wo die verschiedenen Abteilungen, die den Komputer steuern, angeordnet sind. Einige von ihnen mögen zwar ihre Position leicht verändern, ihr Grenzbereich ist aber durch die Systemvariablen festgelegt. Die Systemvariablen des Spektrum sind keine Variablen, wie sie beim BASIC verwendet werden; es handelt sich hierbei um Namen für einige nützliche Werte, die in besonderen Adressen oder an besonderen Stellen in dem Speicher angebracht sind. Man hat sich diesen Namen ausgedacht, damit man sich die Bedeutung eines speziellen Wertes merken kann, der dort gespeichert ist. So ist zum Beispiel die Systemvariable RAMTOP die oberste Adresse in RAM. Dieser Speicherbereich enthält ein Programm (BASIC) und die Werte ihrer Variablen. Die Adresse von RAMTOP ist 23730.



ALLES ÜBER SINCLAIR BASIC

In diesem Kapitel wird der Sinclair BASIC genau beschrieben-Sie finden eineZusammenfassung über die Art und Weise, wie jedes Schlüsselwort verwendet wird, sowie weitere Einzelheiten über die Arbeitsweise des Sinclair BASIC. Sowohl die einfachste als auch die schwierigste Programmiertechnik des BASIC wird hier genau erklärt. Die folgenden Ausführungen sind allerdings nicht dazu gedacht, von Anfang bis zum Ende durchgelesen zu werden; das Kapitel ist als Nachschlagewerk gedacht, durch das es ihnen ermöglicht wird, den größtmöglichen Nutzen aus dem System zu ziehen.

LEITFADEN FÜR DEN PROGRAMMIERER- DER SINCLAIR BASIC-SCHLÜSSELWORTE

Dieser Leitfaden enthält eine umfassende Beschreibung aller BASIC- Schlüsselworte, die im ZX Spektrum + enthalten sind. Jede Eingabe enthält:

- Schlüsselwort-Speicherstelle
- Schlüsselwort-Klasse
- Schlusselwort-Zweck
- Schlusselwort-einsatz
- Programmformat

Die Erklärungen für Speicherstelle, Zweck und Verwendung sprechen für sich. Klasse und Format sind komplexere Begriffe, und deshalb ist es ratsam, zunächst die Informationen auf dieser Seite lesen, um den größtmöglichen Nutzen aus diesem Leitfaden zu ziehen.

Schlüsselwort-Gruppierungen:

Die Schlüsselworte fallen in eine oder mehrere von vier Klassen:

Kommando
ist ein Schlüsselwort, das eine Aktion anregt. Es kann zur Bildung eines direkten Kommandos verwendet werden. Es wird durch Eingabe ausgeführt. Beispiel: RUN LOAD

Anweisung
ist ein Schlüsselwort, das eine Aktion anregt. Es kann in einer direkten Programmabfolge verwendet werden. Es wird dur dann durchgeführt, wenn das Programm läuft. Beispiel: DRAW, INPUT.

Funktionen
ist ein Schlüsselwort, das einen Wert bestimmter Art schafft. Es ist Teil eines Kommandos oder einer Anweisung. Beispiel: RND, INT.

Boolscher Operator
ist ein Schlüsselwort, das logische Anweisungen und Kommanodos gibt. Es kann den wahren Gehalt bestimmter Bedingungen bestimmen oder verändern. Der Spektrum besitzt drei logische Operatoren: AND, OR und NOT.

Zahlen und Variablen

Zahlen
Werden bis zu einer Genauigkeit von 9 oder 10 Ziffern gespeichert. Umfang der verarbeitenten Zahlen liegt etwa bei 10^{38} 4 * 10^{-39}

Akzeptierte Variabeln
Numerisch: Jede Länge, beginnend mit einem Buchstaben. Abstände werden ignoriert und alle Buchstaben in Kleinschreibweise umgewandelt. Es besteht also kein Unterschied zwischen Groß-und Kleinschreibweise!
Folge: Jeder Einzelbuchstabe, auf den ein 5 folgt. Es wird nicht unterschieden zwischen Groß-und Kleinschreibweise.
Array: Für Feldvariablen und Laufgrenzen, siehe unter DIM.

Schlüsselwort – Format

Das Format eines Schlüsselworts drückt die Sytax eines jeden Schlüsselwortes aus; das heißt die korrekte Kombination des Schlüsselwortes und anderer Faktoren wie zum Beispiel Werte und Variabeln. Die folgenden Abkürzungen werden zur Angabe des Formats gegeben:

Abkürzung	Erklärung	Beispiel
num-const	Numerische Konstante (eine Zahl)	24.5
num-var Summe	Numerische Variable (eine Variable, die eine Nummer enthalten kann)	
num-expr	Numerischer Terminus (jede Kombination von numerischen Konstanten, Variablen und Schlüsselworte, die eine Zahl bilden)	Summe*24.5 RND*7
int-num-const int-num-var int-num-exp	Numerische Konstante, Variabel oder Ausdruck, dessen Wert auf den nächstliegenden Integral ert aufgerundet wird.	
string-const	Eine Folgekonstante oder Folge (jede Komination von Charaktern in Anführungszeichen)	"ZX Spectrum +"
string-var	Folgevariable (eine Folge, die eine Variable enthält).	a$
string-expr	Folgeausdruck (jede funktionierende Kombination von Folgekonstanten, Variablen und Schlüsselworte, die eine Folge bilden).	a$+"ZX Spectrum +" a$(6TO8)
Buchstaben	Jeder Groß- und Kleinbuchstabe	Y x
Buchstaben$	Jeder Groß-oder Kleinbuchstabe	B$ a$
cond	Bedingung oder Unterbedingungeinnerhalb einer Bedingung	x = 10 AND t <= 10
statement	Jede BASIC-Anweisung, die gilt, wenn kombiniert halb einer Bedingung	IF t>10 THEN STOP PRINT INK 2;x
[]	Kommando, das wahlweise gegeben werden kann	

Zeichen in Sinclair-BASIC Sprache

Zeichen	Befehl	Funktion/Verwendung
$	4	Folgevariable
'	7	Kennzeichnet Beginn neuer Zeile
(	8	Klammer auf
)	9	Klammer zu
<=	Q	Kleiner als oder gleich
<>	W	ungleich
>=	E	Größer als oder gleich
<	R	ist kleiner als
>	T	Größer als
↑	H	in die Potenz erheben
−	J	Subtraktion, negativ
+	K	Addition/positiv Stringkette
=	L	gleich
.	Z	trennt Anweisungen in einem Programm
/	V	Trennung, Division
*	B	Multiplikation
.	Eigenes Kennwort	Dezimalpunkt
;	Eigenes Kennwort	Anzeige in nächster Spalte Trennt Anzeigen innerhalb von Programmanzeigen.
"	Eigenes Kennwort	öffnet und schließt Anzeige
,	Eigenes Kennwort	Anzeige in Spalte Ø oder 16 Trent Wert nach Schlüsselworten.

	0	1	2	3	4	5	6	7	8	9
0							**PRINT** Komma	**EDIT**	Cursor links	Cursor rechts
10	Cursor runter	Cursor rauf	**DELETE**	**ENTER**	Zahl		**INK** Kontrolle	**PAPER** Kontrolle	**FLASH** Kontrolle	**BRIGHT** Kontrolle
20	**INVERSE** Kontrolle	**OVER** Kontrolle	**AT** Kontrolle	**TAB** Kontrolle						
30			leer	!	"	#	$	%	&	'
40	(	)	*	+	,	−	.	/	Ø	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[	\	]	↑	_	£	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{	|	}	~	©	□	▗
130	▖	▄	▟	▐	▚	▜	▛	▞	▌	▙
140	▃	▙	▜	■	GRAPHICS A	GRAPHICS B	GRAPHICS C	GRAPHICS D	GRAPHICS E	GRAPHICS F
150	GRAPHICS G	GRAPHICS H	GRAPHICS I	GRAPHICS J	GRAPHICS K	GRAPHICS L	GRAPHICS M	GRAPHICS N	GRAPHICS O	GRAPHICS P
160	GRAPHICS Q	GRAPHICS R	GRAPHICS S	GRAPHICS T	GRAPHICS U	**RND**	**INKEY$**	**PI**	**FN**	**POINT**
170	**SCREEN$**	**ATTR**	**AT**	**TAB**	**VAL$**	**CODE**	**VAL**	**LEN**	**SIN**	**COS**
180	**TAN**	**ASN**	**ACS**	**ATN**	**LN**	**EXP**	**INT**	**SQR**	**SGN**	**ABS**
190	**PEEK**	**IN**	**USR**	**STR$**	**CHR$**	**NOT**	**BIN**	**OR**	**AND**	<=
200	>=	<>	**LINE**	**THEN**	**TO**	**STEP**	**DEF FN**	**CAT**	**FORMAT**	**MOVE**
210	**ERASE**	**OPEN #**	**CLOSE #**	**MERGE**	**VERIFY**	**BEEP**	**CIRCLE**	**INK**	**PAPER**	**FLASH**
220	**BRIGHT**	**INVERSE**	**OVER**	**OUT**	**LPRINT**	**LLIST**	**STOP**	**READ**	**DATA**	**RESTORE**
230	**NEW**	**BORDER**	**CONTINUE**	**DIM**	**REM**	**FOR**	**GO TO**	**GO SUB**	**INPUT**	**LOAD**
240	**LIST**	**LET**	**PAUSE**	**NEXT**	**POKE**	**PRINT**	**PLOT**	**RUN**	**SAVE**	**RANDOMIZE**
250	**IF**	**CLS**	**DRAW**	**CLEAR**	**RETURN**	**COPY**				

ABS ABSoluter Wert

Taste
EXTEND MODE
G

Funktion

ABS gibt die absolute Größe einer numerischen Zahl, das heißt, der Wert ohne positives oder negatives Vorzeichen.

Benutzung von ABS
ABS folgt eine numerische Zahl. Ein Ausdruck muß in Klammern eingeschlossen werden, z.B.

```
50 LET x=ABS (y-z)
```

ABS gibt den absoluten Wert des numerischen Wertes

Beispiel
Der Befehl

```
PRINT ABS-43.2
```

zeigt an 34.2

Format
ABS num-const
ABS num-var
ABS (num-expr)

ACS ArcCoSinus

Taste
EXTEND MODE
SYMBOL SHIFT W

Funktion

ACS Berechnet den Wert eines Winkels von seinem Cosinus.

Benutzung von ACS
ACS folgt ein numerischer Wert. Ein Ausdruck muß in Klammern eingeschlossen werden, z.B.

```
60 LET x=ACS(y*z)
```

Der Wert nach ACS (y*z oben) ist der Cosinus des gewünschten Winkels und kann von -1 bis 1 gehen. ACS gibt dann den Wert des Winkels in Radianten. Um Radianten in Grad umzuwandeln, wird der von ACS gegebene Wert mit 180/PI multipliziert.

Beispiel
Der Befehl

```
PRINT 180:PI *ACS0.5
```

zeigt 60. den Winkel in Grad, der einen Cosinus von 0.5 hat.

Format
ACS num-const
ACS num-var
ACS (num-expr)

AND

Taste
SYMBOL SHIFT Y

Boolescher Operator/Funktion

AND dient als Boolescher Operator, um die Richtigkeit einer Kombination von Bedingungen zu testen. Nur wenn die Bedingungen richtig sind, ist die Gesamtkombination richtig. AND dient außerdem als Funktion zur Durchführung numerischer Vorgänge an zwei numerischen oder String-Werten.

Benutzung von AND
Als Boolescher Operator verbindet AND zwei Bedingungen in einem Statement, wo die Wahrheit des Ganzen zu überprüfen ist, z.B.

```
90 IF x=y+z AND Zeit < 10
      THEN PRINT "Korrekt"
```

Nur, wenn beide Bedingungen (x=y+z und Zeit < 10) wahr sind, wird der Computer 'Korrekt' anzeigen. Wenn eine oder beide Bedingungen falsch sind, ist die Gesamtkombination falsch und in diesem Beispiel geht das Programm dann zur nächsten Zeile weiter.

AND als Funktion
Als Funktion kann AND in Verbindung mit zwei numerischen Werten funktionieren, z.B.

```
50 LET x = y AND z
```

AND gibt den ersten Wert (y), wenn der zweite (z) nicht gleich 0 ist, und gibt 0, wenn der zweite Wert (z) 0 ist.

AND kann auch bei einem Wert-String funktionieren, wenn AND davorsteht. Ein numerischer Wert muß immer AND folgen, z.B.

```
50 LETa$ = b$ AND z
```

AND gibt den ersten Wert (b$), wenn der zweite (z) nicht Null ist, und einen Null-String (" "), wenn der zweite Wert (z) Null ist. Beachten Sie, daß der Spectrum einer wahren Bedingung den Wert 1 zuordnet und den Wert 0 einer falschen Bedingung und jeden Nicht-Nullwert als wahr erkennt und jeden Null-Wert als falsch. Es wertet Kombinationen von numerischen Werten nicht in Übereinstimmung mit standardmäßigen Wahrheitstabellen aus.

Beispiel

```
60 LET correct=(x=t+z) AND time
   < 10
70 itILET score=score + 10 * (1
AND correct)
80 LET a$ =("Out Of Time Or Not
   * AND NOT correct) +
   "Correct"
```

Wenn die beiden Bedingungen in Zeile 60 wahr wind, erhält die numerische Variable den Wert 1. "Score" wird um 10 erholt und a$ wird "Correct". Wenn eine der beiden Bedingungen falsch ist, hat "correct" den Wert 0; "score" bleibt unverandert und a$ wird "Out Of Time Or Not Correct".

Format
cond**AND**cond
num-expr**AND**-expr
string-expr **AND** string-expr
(cond = condition = Bedingung; expr = expression = Ausdruck)

ASN ARCSiNe

Taste
EXTEND MODE
SYMBOL SHIFT Q

Funktion

ASN berechnet den Wert eines Winkels von seinem Sinus.

Benutzung von ASN
ASN folgt ein numerischer Wert. Ein Ausdruck *muß in Klammern eingeschlossen werden*, z.B.

```
60 LET x=ASN(y*z)
```

Der Wert nach ASN (y*z oben) ist der Sinus des gewünschten Winkels und kann von -1 bis 1 gehen. ASN gibt dann den Wert des Winkels in *Radianten*. Um Radianten in Grad zu verwandeln, wird der durch ASN gegebene Wert mit 180/PI multipliziert.

Beispiel
Der Befehl

```
PRINT 180:PI *ASN 0.5
```

zeigt 30, den Winkel in Grad, der einen Sinus von 0,5 hat.

Format
ASNnum-const
ASNnum-var
ASN(num-expr)

AT

Taste
SYMBOL SHIFT I

Siehe INPUT;LPRINT;PRINT

ATN ArcTaNgent

Taste
EXTEND MODE
SYMBOL SHIFT E

Funktion

ATN (Arcustanges) berechnet den Wert eines Winkels von seiner Tangente.

Benutzung von ATN
ATN folgt ein numerischer Wert. Ein Ausdruck muß in Klammern eingeschlossen werden, z.B.

```
60 LET x=ATN(y*z)
```

Der Wert nach ATN (y*z oben) ist die Tangente des gewünschten Winkels. ATN gibt den Wert des Winkels in *Radianten*. Um Radianten in Grad zu verwandeln, muß der von ATN gegebene Wert mit 180/PI multipliziert werden.

Beispiel
Der Befehl

```
PRINT 180/PI *ATN 1
```

zeigt 45, den Winkel in Grad, der eine Tangente von 1 hat.

Format
ATNnum-const
ATNnum-var
ATN(num-expr)

ATTR ATTRibute

Taste
EXTEND MODE
SYMBOL SHIFT L

Funktion

ATTR gibt die Attribute einer bestimmten Zeichenposition auf dem Bildschirm. Dies sind Ink- und Paperfarbe, Helligkeit und Blinken

oder Nicht-Blinken des Zeichens und der Position.

Benutzung von ATTR
ATTR folgen zwei numerischer Werte *getrennt durch ein Komma und in Klammern* eingeschlossen z.B.

```
150 IF ATTR(v,h)=115 THEN
        GOSUB 2000
```

Der erste Wert nach ATTR (oben v) kann von 0 bis 23 gehen und ist die Zeilennummer einer Position auf dem Schirm. Der zweite Wert (oben h) kann von 0 bis 31 gehen und ist die Spaltennummer der Position. ATTR ergibt dann eine Zahl von 0 bis 255. Diese Zahl ist die Summe der Attribute an der bestimmten Position und baut sich folgendermaßen auf:

Inkfarbe	Farbcode (0 bis 7)
Paperfarbe	8-mal Farbcode
Bright	64
Flashing	128

Beispiel
Wenn ein Zeichen bei Position 11,16 in Inkfarbe 3 (violett) und Paperfarbe 6 (gelb) dargestellt wird und hell ist, aber nicht blinkt, dann zeigt der Befehl

```
PRINT ATTR (11,16)
```

115 (3+8×6+64+0)

ATTR in Binärform
ATTR gibt ein Byte, in dem Bit 7 (das bedeutsamste) 1 für Blinken und 0 für normal ist, Bit 6 ist 1 für hell und 0 für normal, Bits 5 bis 3 sind die Paperfarbe (binär) und Bits 2 bis 0 die Inkfarbe.

Format
ATTR(num-expr, num-expr)

BEEP

Taste
EXTEND MODE
SYMBOL SHIFT Z

Statement/Befehl

Erzeugt im Lausprecher einen einzelnen Ton von gegebener Dauer und Höhe.

Benutzung von BEEP
BEEP kann verwendet werden, um ein Statement in einem Programm zu bilden oder einen direkten Befehl. Ihm folgen zwei numerische Werte, *die durch ein Komma getrennt werden*, z.B.

```
80 BEEP x,y
```

Der erste Wert (x) kann von 0 bis 10 gehen und definiert die Dauer der Note in Sekunden. Der zweite Wert (y) kann von −60 bis 69 gehen und definiert die Höhe der Note in Halbtönen unter dem mittleren C, wenn negativ, oder oberhalb vom mittleren C, wenn positiv.

Beispiel
Der Befehl

```
BEEP 0.5,1
```

führt dazu, daß die Note C # über dem mittleren C eine halbe Sekunde lang ertönt.

Format
BEEP num-expr, num-expr

BIN BINary number

Taste
EXTEND MODE
B

BIN verwandelt eine Binärzahl in eine Dezimalzahl.

Benutzung von BIN
BIN folgt ein Binärzahl, die aus bis zu sechsehn Einsen und Nullen bestehen kann, z.B.

```
50 POKE USR "a", BIN 1010
     1010
```

Bin gibt den Dezimalwert der Binärzahl an. Es wird normalerweise in Verbindung mit POKE und USR, wie oben, benutzt, um benutzergewählte Zeichen zu erzeugen, wobei 1 ein Pixel Inkfarbe und 0 ein Pixel Paperfarbe bedeutet.

Beispiel
Der Befehl

```
PRINT BIN 1111110
```

zeigt 254, den Dezimalwert der Binärzahl.

Format
BIN [1] [0]

BORDER

Taste
B

Statement/Befehl

BORDER bestimmt die Farbe der Umrandung um de Bildschrim-Displaybereich.

Benutzung von BORDER
BORDER kann als direkter Befehl oder als Statement in einem Programm benutzt werden. Ihm folgt ein numerischer Wert, z.B.

```
30 BORDER RND * 7
```

Der Wert nach BORDER wird auf die nächste ganze Zahl auf/abgerundet und bestimmt die Farbe der Umrandung folgendermaßen:

0 schwarz
1 blau
2 rot
3 violett
4 grün
5 hellblau
6 gelb
7 weiß

Beachten, daß BORDER auch die Paperfarbe des unteren Bildschirmteils ergibt. Im Gegensatz zu INK und PAPER kann ein BORDER-Statement nicht in eine PRINT-Anweisung eingefügt werden.

Beispiel
Der Befehl

```
BORDER 2
```

erzeugt eine rote Umrandung um den Displaybereich.

Format
BORDER int-num-expr

BRIGHT

Taste
EXTEND MODE
SYMBOL SHIFT B

Statement/Befehl

BRIGHT führt dazu, daß Zeichen heller als gewöhnlich dargestellt werden.

Benutzung von BRIGHT
BRIGHT kann als direkter Befehl benutzt werden, wird jedoch normalerweise als Statement in einem Programm verwendet. Ihm folgt ein numerischer Wert, z.B.

```
80 BRIGHT 1
```

Der Wert nach BRIGHT wird auf die nächste ganze Zahl gerundet, falls erforderlich, und kann dann entweder 0, 1 oder 8 sein. Ein Wert von 1 führt dazu, daß alle danach durch PRINT oder INPUT-Statements aufgezeigten Zeichen in hellerer Schrift und Papierfarbe erscheinen und der Wert 8 führt dazu, daß helle Zeichenpositionen hell bleiben und normale Positionen normal bleiben, wenn neue Zeichen dort gedruckt werden. BRIGHT gefolgt von 0 macht sowohl BRIGHT1 wie BRIGHT8 ungültig, so daß alle danach erzeugten Zeichen normal sind.

BRIGHT kann auch in ein Display-Statement mit PRINT, INPUT, PLOT, DRAW und CIRCLE eingesetzt werden. BRIGHT folgt dem Schlüsselwort, steht jedoch vor den Daten oder Display-Parametern; ihm folgen dieselben Werte *und ein Semikolon*, z.B.

```
50 PRINT BRIGHT 1;
     "WARNUNG"
```

Die Wirkung von BRIGHT ist dann lokalisiert und gilt nur für die vom Display-Statement aufgezeigten Zeichen, geplotteten Punkte oder gezeichneten Linien. Beachten, daß BRIGHT 1 die Paperfarbe der *ganzen* Zeichenposition von 8×8 Pixeln erhellt, wenn ein Pixel in der Position in einer Inkfarbe geplottet wird.

Format
BRIGHT int-num-expr[;]

CAT CATalogue

Microdrive-Befehl zur Handhabung von Dateien Siehe Handbuch über Microdrive und Interface 1

CHR$ CHaRacter string

Taste
EXTEND MODE
U

Funktion

Die auf der Tastatur zur Verfügung stehenden Zeichen und Schlüsselwörter plus benutzergewählte Grafikzeichen ergeben den Zeichensatz des Spectrum. Durch die Verwendung von CHR$ und einer Codezahl kann man jedes Zeichen als String erhalten. Der Zeichensatz enthält auch mehrere Steuercodes, die das Display der Zeichen beeinflussen. Diese Codes können wirksam gemacht und Zeichen angezeigt werden, durch die Verwendung von PRINT vor CHR$. Kompletter Zeichensatz und Codezahlen sind auf Seite 51 zu finden.

Benutzung von CHR$
CHR$ folgt ein numerischer Wert, z.B.

```
80 PRINT$ x
```

Ein Ausdruck muß *in Klammern eingeschlossen werden*. Der Wert nach CHR$ (oben x) wird auf die nächste ganze Zahl aberundet. Wenn sie im Bereich 32 bis 255 liegt, erzeugt CHR$ ein Tastaturzeichen, ein benutzergewählte Grafikzeichen oder ein Schlüsselwort als String. Der Spectrum verwendet den ASCII code für Werte von 32 bis 95 und 97 bis 126. Wenn x einen Wert von 65 hat, zeigt das obige Statement A.

CHR$ Steuercodes
Werte von 1 bis 31 ergeben entweder Steuercodes oder werden nicht verwendet. CHR$6 (PRINT Komma), 8 (eine Stelle zurück) und 13 (neue Zeile oder ENTER) beeinflussen Displays auf dem Schirm, wenn sie in einem PRINT-Statement auftauchen. CHR$ kann ein Codewert folgen *und ein Semikolon*, z.B.

```
60 PRINT "A"; CHR$6;"B"
```

Dieses Statement zeigt
A B

Eine andere Verwendung von CHR$ Steuercodes ist die Bildung eines zusammengesetzten Strings, in dem sie vorkommen. Das Statement

```
60 PRINT"A" + CHR$6 + "B"
```

hat genau diegleiche Wirkung wie das obige Beispiel.

Codes 16 bis 23 beeinflussen Farbe und Position und beide können in einem zusammengesetzten String benutzt werden, zusammen mit CHR$ gefolgt von einem Farbcodewert von 0 bis 7 für CHR$16 (INK-Steuerung) und CHRS17 (PAPER-Steuerung) oder durch 0 oder 1 für CHR$18 bis CHR$21 (FLASH, BRIGHT, INVERSE und OVER Steuerungen). Der Befehl

```
PRINT CHR$16 + CHR$3 +
CHR$17 + CHR$6 + CHR$18 +
CHR$1 + "ZX SPECTRUM +"
```

zeigt ZX Spectrum + rot und gelb blinkend. Als Alternative kann, wie oben jedes Pluszeichen durch ein Semikolon ersetzt werden.

CHR$22 (AT-Steuerung) folgen zwei CHR$-Werte, um Zeilen- und Spaltennummern anzugeben. Der Befehl

```
PRINT CHR$22 + CHR$11 +
CHR$16 + CHR$42
```

zeigt einen Stern in der Mitte des Bildschirms.

CHR$23 (TAB-Steuerung) folgen ebenfalls zwei Werte auf dieselbe Art und Weise. Der zweite Wert ist normalerweise 0 und der erste Wert gibt die TAB-Position. Der Befehl

```
PRINT CHR$23 + CHR$16 +
CHR$0 + CHR$42
```

zeigt einen Stern in der Mitte des Schirms.

Beachten, daß nur diese Steuerungen zur Verfügung stehen. Die Verwendung von PRINT CHR$ mit einem Schlüsselwort-Wert von mehr als 164 zeigt einfach nur das Schlüsselwort und bringt es nicht zur Anwendung.

Format
CHR$int-num-const [;] [+]
CHR$int-num-var [;] [+]
CHR$(int-num-expr) [;] [+]

CIRCLE

Taste
EXTEND MODE
SYMBOL SHIFT H

Statement/Funktion

CIRCLE zeichnet einen Kreis auf den Bildschirm.

Benutzung von CIRCLE
CIRCLE folgen drei numerische Werte, *die durch Kommata getrennt werden*, zum Beispiel.

```
80 CIRCLEx,y,z
```

Jeder dieser drei Werte wird, wenn nötig, auf die nächste ganze Zahl abgerundet. CIRCLE zeichnet dann einen Kreis auf dem Hochauflösungsgrafikgitter in der jeweiligen Inkfarbe. Die ersten zwei Werte (x,y) bestimmen die horizontalen und vertikalen Koordinaten der Mitte, der dritte Wert (z) die Länge des Radius. Die Abmessungen müssen so gewählt werden, daß der Kreis nicht über die Displayfläche hinausgeht.

CIRCLE wird von Farb-Statements oder Befehlen bestimmt und kann eingeschobene Farb-Statements genauso enthalten wie PLOT und DRAW.

Beispiel
Der Befehl

```
CIRCLE 128,88,87
```

zeichnet einen Kreis, der fast über den ganzen Displaybereich geht.

Format
CIRCLE [statement;] int-num-expr, int-num-expr, int-num-expr
(int=Abkürzung von 'integer' = ganze Zahl)

CLEAR

Taste
X

Statement/Befehl

CLEAR löscht die gegenwärtigen Werte aller Variablen und macht den Speicherraum frei, den diese Werte einnahmen, und Raum bis zu RAMTOP, der obersten Adresse des BASIC-Systembereichs. CLEAR kann auch benutzt werden, um RAMTOP zurückzusetzen.

Benutzung von CLEAR
CLEAR kann als direkter Befehl verwendet werden oder ein Statement in einem Programm bilden. Es braucht keinerlei Parameter, zum Beispiel

```
50 CLEAR
```

CLEAR löscht dann die allen Variablen gegenwärtig zugeordneten Werte, einschließlich Arrays. Außerdem führt es CLS und RESTORE aus, um den Bildschirm freizumachen und den Datenzeiger auf den ersten Datenteil zurückzustellen. Darüberhinaus wird die PLOT-Position auf die untere linke Ecke der Displayfläche zurückgesetzt und der GOSUB-Stapel wird gelöscht.

Beachten, daß CLEAR nicht gebraucht wird vor Neudimensionierung von Arrays, da DIM einen bestehenden Array desselben Namens löscht. Beachten Sie außerdem, daß auch RUN die Durchführung von CLEAR bewirkt.

CLEAR und RAMTOP
CLEAR kann auch ein numerischer Wert folgen, zum Beispiel

```
CLEAR 65267
```

CLEAR führt dann CLEAR wie oben aus und setzt außerdem RAMTOP, die höchste Adresse des BASIC-Systems, auf den gegebenen Wert. RAMTOP wird im ZX Spectrum + auf 65367 gesetzt und liegt unterhalb der für benutzergewählte Grafik vorgesehenen Fläche. NEW löscht den Speicher bis RAMTOP, ein Heruntersetzen von RAMTOP durch CLEAR (im obigen Beispiel um 100 Bytes) ergibt also mehr Speicherraum, der von NEW unbeeinflußt bleibt. Eine Erhöhung von RAMTOP ergibt mehr Platz für BASIC auf Kosten der benutzergewählten Grafik. Beachten Sie, daß der GOSUB-Stapel dann bei RAMTOP liegt, vorausgesetzt, RAMTOP bleibt über dem Rechnerstapel. Um RAMTOP zu finden, geben Sie

```
PRINT PEEK 23730 + 256*PEEK
23731
```

Format
CLEAR [num-expr]

CLOSE

Microdrive-Befehl zur Handhabung von Dateien Siehe Handbuch Microdrive and Interface 1

CLS Clear Screen

Taste
V

Statement/Befehl

CLS löscht allen Text und jegliche graphische Darstellungen vom Displaybereich und läßt ihn einfach leer in der jeweiligen Paperfarbe (Hintergrund).

Benutzung von CLS
CLS kann als direkter Befehl benutzt werden oder ein Statement in einem Programm bilden. Es braucht keinerlei Parameter, z.B.

```
250 IFa$="NEIN" THEN CLS
```

Die Displayfläche (jedoch nicht die Umrandung) wird dann gelöscht und nimmt die durch die vorherige PAPER-Anweisung gewählte Farbe an oder die standardmäßige Papierfarbe weiß.

Beachten, daß CLS nach PAPER verwendet wird und vor PRINT oder jeglichem anderen Display-Statement, um einen farbigen Hintergrund über die ganze Displayfläche zu erhalten.

Format
CLS

CODE

Taste
EXTEND MODE
I

Funktion

CODE gibt die Codenummer eines Zeichens im Spectrum Zeichensatz (siehe Seite 51)

Benutzung von CODE
CODE folgt ein String-Wert, zum Beispiel

```
90 IF CODE a$ < 65 OR CODE a$ > 90 THEN GOTO 80
```

Ein Stringausdruck muß *in Klammern eingeschlossen werden.* CODE liefert die Codenummer des *ersten* Zeichens im String. Wenn dies ein Null-String ist (""), liefert CODE 0.

SAVE CODE und LOAD CODE
CODE wird anders benutzt in Verbindung mit SAVE und LOAD. Siehe SAVE CODE und Load CODE.

Format
CODEstring-const
CODEstring-var
CODE(string-expr)

CONTINUE

Taste
C

Befehl

Wenn ein Programm anhält, kann CONTINUE benutzt werden, um das Programm an dem Punkt, an dem es aufgehört hatte, wieder anfangen zu lassen. Wenn ein Fehler gemacht wurde, der das Programm anhielt, muß dieser erst korrigiert werden, bevor CONTINUE es möglich macht, das Programm weiterzuführen.

Benutzung von CONTINUE
CONTINUE wird als direkter Befehl benutzt, wenn ein Programm angehalten hat. Es benötigt keinerlei Parameter. Nach CONTINUE macht das Programm normalerweise an dem Statement weiter, bei dem es aufgehört hatte. Wenn die Ursache ein Fehler war, kann ein Befehl eingegeben werden, um den Fehler zu korrigieren und danach kann durch CONTINUE das Programm wiederaufgenommen werden. Wenn das Programm bei einem STOP-Statement angehalten hatte mit Mitteilung 9 oder wenn es anhielt, weil die BREAK-Taste gedrückt wurde und Mitteilung L lieferte, führt CONTINUE das Programm ab dem nächsten Statement weiter. Ein Korrigierbefehl kann, wenn erforderlich, zuerst eingegeben werden.

Wenn CONTINUE benutzt wird, um einen direkten Befehl wiederaufzunehmen, geht es in eine Schleife, wenn der Befehl am ersten Statement im Befehl angehalten hatte. Das Display verschwindet, die Steuerung wird jedoch wiedergewonnen durch Drücken von BREAK. CONTINUE liefert Report (Mitteilung) 0, wenn der Befehl am zweiten Statement angehalten hatte und Report N am dritten oder jedem darauffolgenden Statement.

COPY

Taste
Z

Befehl

COPY führt bei Druckern vom Typ Sinclair dazu, daß eine Kopie des Bildschirmdisplays erzeugt wird.

Benutzung von Copy
COPY wird als direkter Befehl benutzt, wenn ein Programm beendet oder angehalten wurde. Es benötigt keinerlei Parameter. Nach COPY – und vorausgestzt, der Drucker ist angeschlossen – wird eine Kopie der ersten 22 Zeilen des Bildschirmdisplays gedruckt. Beachten Sie, daß alle Ink-Farben (Vordergrund) in schwarz gedruckt werden; Paper-Farben (Hintergrund) werden nicht gedruckt. Der Drucker kann durch ein Drücken von BREAK angehalten werden.

Wenn auf dem Schirm ein Programm-Listing erscheint, kann es durch COPY gedruckt werden, vorausgesetzt, es war durch einen LIST-Befehl erzeugt worden. Beachten Sie, daß auf dem Schirm ein Listing erscheint beim Drücken von ENTER, nachdem ein Programm beendet oder angehalten wurde, dieses 'automatische' Listing kann jedoch nicht durch COPY gedruckt werden.

Format
COPY

COS COSine

Taste
EXTEND MODE
W

Funktion

COS liefert den Cosinus eines Winkels.

Benutzung von COS
COS folgt ein numerischer Wert, zum Beispiel

```
140 LET x=COSy
```

Ein Ausdruck muß *in Klammern eingeschlossen werden.* Der Wert nach COS ist der Winkel *in Radianten.* COS liefert dann den Cosinus des Winkels. Grade können in Radianten verwandelt werden druch Multiplikation mit PI/180.

Beachten Sie, daß COS einen negativen Wert für Winkel von 90 bis 270 liefert und einen positiven Wert für Winkel von 0 bis 90 und 270 bis 360.

Beispiel
Der Befehl

```
PRINT COS (60*PI/180)
```

zeigt 0,5, den Cosinus von 60 Grad.

Format
COSnum-const
COSnum-var
COS(num-expr)

DATA

Taste
EXTEND MODE
D

Statement

DATA liefert eine Liste von Daten innerhalb eines Programms. Diese Datenelemente können Werte von darzustellenden Variablen oder Strings sein. Jedes Element wird durch ein READ-Statement einer Variablen zugeordnet.

Die Zuordnung erfolgt in der Reihenfolge, in der Datenelemente im Programm erschienen, RESTORE kann jedoch verwendet werden, um die Zuordnung beim ersten Punkt in einem gegebenen DATA-Statement beginnen zu lassen.

Benutzung von DATA
DATA kann nur als Statement in einem Programm benutzt werden. Ihm folgt normalerweise eine Liste von numerischen oder String-Konstanten, *die durch Kommata getrennt werden,* zum Beispiel

```
50 DATA 31, "JAN",28,
    "FEB"
```

Jede Konstante wird dann einer Variablen durch das READ-Statement zugeordnet, welches DATA liest. Das DATA-Statement kann an beliebiger Stelle im Programm stehen. Die Anzahl, Art (numerisch oder String) und Reihenfolge der Konstanten muß der Anzahl von Malen entsprechen, die das READ-Statement ausgeführt wird und der Art und Reihenfolge der Variablen im READ-Statement. Die Datenliste kann in mehrere aufeinanderfolgende DATA-Statements aufgeteilt werden, wenn es zuviele Teile gibt, um sie alle in einem Statement unterzubringen.

Beispiel
Das folgende Programm

```
10 FORn= 1 TO 2
20 READ x,a$
30 PRINT a$,x;"Tage"
40 NEXT n
50 DATA 31, "JAN", 28,
    "FEB"
```

zeigt an

```
JAN          31 Tage
FEB          28 Tage
```

Benutzung von DATA mit Variablen
Die Datenteile in einem DATA-Statement können aus numerischen oder String-Variablen bestehen oder aus Ausdrücken, vorausgesetzt, den Variablen sind zuvor Werte zugeordnet worden. Im obigen Beispiel kann das DATA-Statement umgewandelt werden in

```
50 DATA d, m$, d-3, "FEB"
```

Wenn d zuvor ein Wert von 31 zugeordnet wurde und m$ ein Wert von "JAN", wird dasselbe Display erzielt.

LOAD DATA, SAVE DATA und **VERIFY DATA**
DATA kann auch in Verbindung mit LOAD, SAVE und VERIFY verwendet werden, um Arrays auf Band zu speichern. Siehe LOAD DATA, SAVE DATA und VERIFY.

Format
DATAnum-expr [,num-expr] [,string-expr]
DATAstring-expr [,num-expr]],string-expr]

DEF FN DEFine FuNction

Taste
EXTEND MODE
SYMBOL SHIFT 1

Statement

Durch DEF FN kann der Benutzer eine Funktion definieren, die nicht als Schlüsselwort zur Verfügung steht. Eine Reihe von Parametern können der Funktion in einem FN-Statement verliehen werden, welches die Funktion abruft und entweder einen numerischen oder einen String Wert als Ergebnis liefert.

Benutzung von DEF FN
DEF FN kan nur als Statement in einem Programm benutzt werden. Wenn eine numerische Funktion definiert werden soll, folgt DEF FN ein einzelner, beliebiger Buchstabe und dann eine oder mehrere numerische Variable, *die alle durch Kommata getrennt und in Klammern eingeschlossen sind*, zum Beispiel DEF FNr(x,y). Diesem folgt ein Gleichheitszeichen und dann ein numerischer Ausdruck mit den Variablen, zum Beispiel

```
1000 DEF FNr(x,y)=
     SQR(x↑2+y↑2)
```

Der nach DEF FN stehende Buchstabe (oben r) ist ein Name, der die Funktion identifiziert. Die Variablen können ebenfalls nur einzelne Buchstaben sein. Beachten Sie, daß der Spectrum in beiden Fällen keinen Unterschied zwischen Klein- und Großbuchstaben macht.

Der Ausdruck nach dem Gleichheitszeichen verwendet die Variablen (oben x und y), um die Funktion zu definieren.

Ein DEF-FN Statement kann an beliebiger Stelle im Programm stehen. Um die Funktion, die es definiert, abzurufen, wird ein FN-Statement verwendet. Diesem folgt dann der Funktionsnamenbuchstabe und eine Liste von numerischen Werten, *die alle durch Kommata getrennt und in Klammern eingeschlossen werden*, zum Beispiel

```
50 PRINT FN r(3,4)
```

Die Werte in Klammern werden der Funktion unterworfen in derselben Reihenfolge, wie die Variablen im DEF FN Statement. In diesem Beispiel wird so x ein Wert von 3 zugeordnet und y ein Wert von 4. FN wertet den Ausdruck aus und liefert den Wert.

DEF FN können auch ein Buchstabe und zwei Klammerzeichen folgen, zum Beispiel

```
1000 DEF FNr( )=INT(x+0.5)
```

Der der Variablen zugeordnete Wert (oben x) wird der Funktion unterworfen wenn sie durch FN abgerufen wird. In diesem Fall liefert FNR( ) den gegenwärtig x zugeordneten Wert, auf/abgerundet auf die nächste ganze Zahl.

DEF FN und Strings
DEF FN und FN können genauso verwendet werden, um eine String-Funktion zu definieren und abzurufen. In diesem Fall ist der Funktionsname ein einzelner Buchstabe gefolgt von $ und eine oder mehrere der Variablen in dem Statement ist ein Buchstabe gefolgt von $. Ein entsprechender String-Ausdruck bildet die Definition, zum Beispiel

```
1000 DEF FN a$(b$,x,y)=
     b$(x TO y)
```

Der String-Ausdruck nach dem Gleichheitszeichen in diesem Beispiel ist ein String-Slicer und x und y sind die ersten und letzten Zeichen eines Teils von b$. FN muß der Funktionsname folgen und, in Klammern, ein String-Wert zusammen mit anderen Parametern, die der Funktion unterworfen werden.

In diesem Fall zeigt der Befehl

```
PRINT FN a$
("FUNDAMENTAL",1,3)
```

zeigt FUN und der Befehl

```
PRINT FN a$
("FUNDAMENTAL",5,8
```

zeigt AMEN.

Format
DEF FN letter ([letter], [,letter]) = num expr
DEF FN letter$ ([letter$] [letter] [,letter] [,letter$] = string-expr
FN letter $ ([num-expr] [,num-expr] [,num-expr] [,string-expr]

DIM DIMension

Taste
D

Statement

DIM wird verwendet, um ein Array einer gegebenen Anzahl von numerischen oder String-Variablen zu dimensionieren (setzen). Ein Array enthält eine Gruppe von Variablen desselben Namens, die durch ihren *Index* (Wert, der jede Variable oder jedes Element in einem Array identifiziert) unterschieden werden.

Benutzung von DIM mit numerischen Arrays
DIM wird zur Bildung eines Statements in einem Programm verwendet. Ihm folgt ein einzelner Buchstabe, der das Array benennt und einem oder mehreren numerischen Werten, *die durch Kommata getrennt und in Klammern eingeschlossen werden*, zum Beispiel

```
20 DIM x (10)
80 DIM z (20,5)
```

Im ersten Fall wird ein eindimensionales, numerisches Array geschaffen, welches zehn Elemente mit Indexen von 1 bis 10 enthält. Das Array hat den Namen x und die Variablen mit Index sind x(1) bis x (10) einschließlich. Jegliches bestehendes Array desselben Namens wird gelöscht und den Variablen wird je ein Wert von 0 zugeordnet. Bitte beachten, daß bei der Dimensionierung eines Array der Spectrum nicht zwischen Namen mit Groß- oder Kleinbuchstaben unterscheidet – Variable x(2) ist dasselbe wie Variable X(2). Einfache numerische Variable mit demselben Buchstaben wie ein Arrayname können jedoch gleichzeitig bestehen bleiben, und unabhängig, wenn erforderlich, benutzt werden.

Die Anzahl der Werte in Klammern ist gleich der Anzahl von Dimensionen, die in einem numerischen Array geschaffen werden. Das zweite Beispiel setzt ein zweidimensionales Array von 100 Elementen mit 20 Elementen in der ersten Dimension und 5 in der zweiten. Diese Elemente sind numeriert z(1,1) bis zu z (20,5).

Es können Arrays mit beliebiger Anzahl von Dimensionen geschaffen werden.

Die Elemente eines numerischen Arrays können anschließend durch den Arraynamen identifiziert werden, gefolgt von einem Wert *in Klammern*, zum Beispiel

```
70 PRINT x(a)
160 PRINT z(7,b)
```

DIM und String-Arrays
DIM wird genauso verwendet wie bei numerischen Arrays, außer daß ein einzelner Buchstabe, gefolgt von $, für den Arraynamen benutzt wird. Außerdem muß zu den Dimensionswerten in Klammern ein Extrawert hinzugefügt werden, um die Länge jedes Strings zu bestimmen, zum Beispiel

```
30 DIM a$ (20,5)
90 DIM b$ (20,5,510)
```

Das erste Statement schafft ein Array mit 20 Elementen, von denen jedes einen String von 5 Zeichen hat. Die Variablen mit Index werden a$(1) bis a$(20) einschließlich genannt und ihnen wird zunächst ein Null-String (leer) (" ") zugeordnet. Jedes bestehende Array desselben Namens wird gelöscht und, im Gegensatz zu numerischen Arrays, kann eine einfache String-Variable desselben Namens nicht gleichzeitig bestehen.

Das sweite Beispiel erzeugt ein zweidimensionales String-Array mit 100 Elementen, 20 Elementen in der ersten Dimension und 5 in der zweiten. Alle Elemente haben eine Länge von 10 Zeichen.

Wenn anschließend String-Werte einem String-Array zugeordnet werden, werden sie mit Leeräumen am Ende des String aufgefüllt oder, wenn nötig, auf die definierte Länge abgehackt.

Die Elemente eines String-Arrays werden durch den Arraynamen identifiziert, dem in Klammern einer oder mehrere numerische Werte folgen, die die Indexnummer oder -nummern angeben. Zum Beispiel, Element a$(2) könnte "SMITH" sein und Element b$(12,4) "DERBYSHIRE". Ein zusätzlicher Wert kann jedoch noch hinzugefügt werden, um ein bestimmtes Zeichen in dem String zu definieren. In diesen Beispielen wäre a$(2,2) "M" (das zweite Zeichen in "SMITH") und b$(12,4,5) wäre "Y".

String-Arrays ohne Dimension
Es ist möglich, einen String-Array ohne Dimensionen zu schaffen, indem nur ein Wert in Klammern benutzt wird, zum Beispiel

```
10 DIM c$ (15)
```

Dieses Array hat nur ein Element, welches c$ ist, und seine Länge ist festgesetzt auf den definierten Wert (15 Zeichen)

Format
DIMletter (num-expr [,num-expr])
DIMletter$ (num-expr [,num-expr])

DRAW

Taste
W

Statement/Befehl

DRAW wird zum Zeichnen von geraden Linien und Kurven auf dem Bildschirm verwendet.

Benutzung von DRAW
DRAW wird normalerweise zur Bildung eines Statements in einem Programm benutzt. Wird eine gerade Linie gewünscht, folgen ihm zwei numerische Werte, die durch Komma getrennt werden, zum Beispiel

```
40 DRAW x,y
```

Es wird dann eine gerade Linie auf das Hochauflösungsgrafikgitter gezeichnet von der Position, die durch die vorhergehende PLOT-Anweisung definiert wurde oder die Position, die durch die vorhergehende DRAW-Anweisung erreicht wurde, jenachdem, welches der letzte Vorgang war. Beide auf DRAW folgenden Werte werden auf die nächste ganze Zahl abgerundet, sofern dies nötig ist. Der erste Wert (oben x) definiert den horizontalen Abstand von dieser Position, während der zweite Wert (oben y) den vertikalen Abstand bestimmt. Diese Werte sind negativ, wenn die Linie nach links gehen soll oder nach unten, und die erreichte Position muß sich innerhalb der Displayfläche befinden.

Ist kein vorhergehendes PLOT- oder DRAW-Statement vorhanden, beginnt DRAW bei Position 0,0 (der unteren linken Ecke des Bildschirms).

DRAW wird durch Farb-Statements oder- Befehle beeinflußt und kann eingesetzte Statements mit derselben Wirkung enthalten wie PLOT und CIRCLE.

DRAW für Kurven
DRAW kann ein dritter Wert folgen, um eine Kurve zu erzeugen, die Teil eines Kreises ist, zum Beispiel

```
40 DRAW x,y,z
```

Der dritte Wert (oben z) definiert den Winkel (in Radianten), den die Linie durchläuft, während sie gezeichnet wird. Die Linie geht nach links, wenn dieser Wert positiv ist, und nach rechts, wenn er negativ ist. Werte von PI oder -PI erzeugen einen Kreis.

Beispiel
Das folgende Programm zeichnet ein Dreieck:

```
10 PLOT 127,150
20 DRAW 70,-100
30 DRAW -140,0
40 DRAW 70,100
```

Ein Hinzufügen von, 1 oder, –1 zu dem DRAW-Statement führt dazu, daß die Kurven nach innen bzw. nach außen gehen

Format
DRAW [statement;] int-num-expr, int-num-expr [int-num-expr]

ERASE

Microdrive-Befehl zur Handhabung von Dateien. Siehe Handbuch Microdrive and Interface 1

EXP EXPONENT

Taste
EXTEND MODE
X

Funktion

EXP ist eine mathematische Funktion, die den Exponenten e zu einer gegebenen Potenz erhebt.

Benutzung von EXP
EXP folgt ein numerischer Wert, zum Beispiel

```
60 LET y=EXP x
```

Ein Ausdruck muß *in Klammern eingeschlossen* sein. EXP liefert dann den Exponenten e zur Potenz des Arguments erhoben (oben x).

Beispiel
Der Befehl

```
PRINT EXP 1
```

zeigt 2.7182818, den Wert von e.

Format
EXPnum-const
EXPnum-var
EXP(num-expr)

FLASH

Taste
EXTEND MODE
SYMBOL SHIFT V

Statement/Befehl

FLASH führt dazu, daß Zeichenpositionen blinken, wobei sich Ink- und Paperfarbe mit konstanter Geschwindigkeit abwechseln.

Benutzung von FLASH
FLASH kann als direkter Befehl verwendet werden, wird jedoch normalerweise zur Bildung eines Statements in einem Programm benutzt. Ihm folgt ein numerischer Wert, zum Beispiel

```
50 FLASH 1
```

Der auf FLASH folgende Wert wird, wenn nötig, auf die nächst ganze Zahl abgerundet und kann dann 0, 1 oder 8 sein. Ein Wert von 1 führt dazu, daß alle danach durch PRINT oder INPUT aufgezeigten Zeichen blinken. Der Wert 8 sorgt dafür, daß blinkende Zeichenpositionen weiterblinken und normale

Zeichenpositionen normal bleiben, wenn neue Zeichen an der Stelle gedruckt werden. Folgt FLASH eine Ø, werden sowohl FLASH 1 wie FLASH 8 gelöscht, so daß alle danach aufgezeigten Zeichen normal sind.

FLASH kann auch in Displaystatements eingebettet (eingesetzt) werden, die durch PRINT, INPUT, PLOT, DRAW und CIRCLE gebildet werden. FLASH steht hinter dem Schlüsselwort, jedoch vor den Daten oder Displayparametern; ihm folgen dieselben Werte *und ein Semikolon*. zum Beispiel

120 PRINT FLASH 1; **INK** 2; **PAPER** 6; "WARNUNG"

Die Wirkung von FLASH ist dann lokalisiert und bezieht sich dann nur auf die durch das Displaystatement aufgezeigten Zeichen, geplotteten Punkte oder gezeichnete Linie. Bitte beachten, daß FLASH 1 dazu führt, daß die ganze 8x8 Pixelposition blinkt, wenn ein Pixel in Inkfarbe geplottet ist.

Format
FLASH int-num-expr[;]

FN FuNction

Taste
EXTEND MODE
SYMBOL SHIFT 2

Funktion

FN ruft eine benutzergewählte Funktion ab. Es wird immer in Verbindung mit DEF FN verwendet, welches die abzurufende Funktion definiert.

Benutzung von FN
Soll eine numerische Funktion abgerufen werden, folgt FN ein Buchstabe und dann zwei Klammern. Wenn Parameter der Funktion unterworfen werden sollen, werden diese *durch Kommata getrennt und in Klammern eingeschlossen*, zum Biespiel

170 LET x=**FN** r(3,4)

Die Parameter (oben 3 und 4) werden dann der als r bezeichneten Funktion unterzogen. FN liefert dann das Ergebnis. Wenn keine Parameter dan sind, müssen die Klammern trotzdem erscheinen, zum Beispiel

70 PRINT FN r( )

In diesem Fall benutzt die Funktion die gegenwärtig ihren Variablen zugeordneten Werte.

FN ruft genauso eine String-Funktion ab, außer daß nach dem Buchstaben, der die Funktion benennt, ein **$** stehen muß. Weitere Einzelheiten siehe DEF FN.

Format
FN letter ([num-expr] [,num-expr])
FN letter **$** ([string-expr] [num-expr] [,num-expr] [,string-expr])

FOR

Taste
F

Statement/Befehl

FOR wird immer mit den Schlüsselwörtern TO und NEXT verwendet, um eine FOR NEXT Schleife zu bilden. Diese Konstruktion ermöglicht es, daß ein Teil des Programmes eine gegebene Anzahl von Malen wiederholt wird.

Benutzung von FOR
FOR bildet immer ein Statement mit TO. Ihm folgt ein Buchstabe, ein Gleichheitszeichen und dann zwei numerische Werte, die durch TO getrennt werden, zum Beispiel

60 FOR a=1 **TO** 9

Der Buchstabe (oben a) bildet eine Steuervariable. Die zu wiederholenden Statements folgen und eine oder mehrere von ihnen machen normalerweise Gebrauch von der Steuervariablen. Die Schleife endet dann mit einem NEXT-Statement, in dem nach NEXT die Steuervariable steht, zum Beispiel

90 NEXT a

Bei der Ausführung löscht FOR alle Variablen mit demselben Namen wie die Steuervariable und gibt ihr einen Ursprungswert, welcher dem Wert vor TO (oben 1) gleich ist. Die Anweisungen werden dann mit der so bewerteten Steuervariablen ausgeführt. Wenn NEXT erreicht wird, wird der Wert der Steuervariablen um 1 erhöht. Wenn dieser Wert gleich dem Wert nach TO ist oder weniger als dieser Wert (Limit-Wert oben 9), geht das Programm zur FOR-Anweisung zurück und die FOR NEXT Schleife wird wiederholt. Wenn die Steuervariable einen größeren Wert hat als der Limit- Wert, hört die Schleife auf und das Programm wird mit dem nächsten Statement nach NEXT weitergeführt.

Im obigen Beispiel wird die Schleife 9 Mal wiederholt, wobei sich die Steuervariable a von 1 bis 9 erhöht. Beim Verlassen der Schleife hat a einen Wert von 10.

Beachten, daß der Spectrum bei der Benennung der Steuervariablen keinen Unterschied zwischen Klein- und Großbuchstaben macht.

Die Benutzung von STEP in einer FOR NEXT Schleife
STEP ist ein Schlüsselwort, welches in eine FOR-Anweisung eingebaut werden kann, wenn sich die Steuervariable um einen Wert erhöhen soll, der nicht 1 ist, oder wenn er abnehmen soll. STEP folgt auf den Limit-Wert und hinter STEP folgt ein numerischer Wert, zum Beispiel

60 FOR a=1 **TO** 9 **STEP** 2

Die Steuervariable wird um den Step-Wert (oben 2) erhöht, bis sie größer als der Limitwert ist. Die Steuervariable a hat die aufeinanderfolgenden Werte 1,3,5,6 und 9 und verläßt die Schleife mit einem Wert von 11.

Ein negativer Step-Wert führt zu einem Abnehmen des Wertes der Steuervariablen. In diesem Fall muß der Ursprungswert größer sein als der Limitwert, und die Schleife endet, wenn der Wert der Steuervariablen kleiner als der Limitwert ist, zum Beispiel

60 FOR a=9 **TO** 1 **STEP** −1

Der Wert von a nimmt von 9 auf 1 ab und verläßt die Schleife mit einem Wert von Ø.

Nesting-Schleifen (verschachtelte Schleifen)
Eine oder mehrere FOR NEXT Schleifen können ineinander gelegt werden, ein Vorgang, der als 'nesting' bezeichnet wird. Die Reihenfolge der Steuervariablen im NEXT-Statement muß entgegengesetzt der Reihenfolge der Steuervariablen im FOR-Statement sein. FOR NEXT Schleifen können beliebig verschachtelt werden, das heißt, es können soviele Schleifen wie gewünscht ineinander gelegt werden.

Format
FOR letter=num-expr **TO** num-expr [**STEP** num-expr]
NEXT letter

FORMAT

Microdrive-Befehl für Handhabung von Dateien. Siehe Handbuch Microdrive und Interface 1

GOSUB

Taste
H

Statement/Befehl

GOSUB führt dazu, daß das Programm zu einer Subroutine (Unterprogramm) geht, welche ein getrennter Teil des Programmes ist. Dies ist vor allem nützlich, wenn eine Subroutine mehrere Male im Programm gebraucht wird

Benutzung von GOSUB
GOSUB kann als Anweisung oder direkter Befehl verwendet werden und ihm folgt ein numerischer Wert, zum Beispiel

GOSUB 1000

Bei der Ausführung wird der Wert nach GOSUB (oben 1000) auf die nächste ganze Zahl aufgerundet und das Programm geht zu der Zeilennummer mit diesem Wert. Die Benutzung einer Variablen oder eines Ausdrucks ermöglicht es dem Programm, bei einer berechneten Zeilennummer zu der Subroutine überzugehen. Beachten, daß, wenn die Zeilennummer nicht existiert, das Programm trotzdem abzweigt und mit dem ersten Statement weitermacht, das ihm begegnet.

Eine Subroutine endet mit RETURN und das Programm geht dann wieder zum Statement zurück, daß nach dem GOSUB-Statement folgt. Subroutinen können verschachtelt werden, so daß eine von einer anderen aus erreicht werden kann, in welchem Fall RETURN das Programm zurücksendet zu dem Statement nach dem letzten ausgeführten GOSUB-Statement.

Der GOSUB-Stapel
Immer, wenn GOSUB ausgeführt wird, wird seine Zeilennummer auf dem GOSUB-Stapel im Speicher abgelegt. Wenn zwei oder mehr GOSUB vor der RETURN-Anweisung ausgeführt werden, werden ihre Zeilennummern so gestapelt, daß die letzte Nummer oben auf dem Stapel ist. RETURN nimmt immer die oberste Zeilennummer vom Stapel und geht zu dieser Zeile, um das Program fortzusetzen.

Beachten, daß Fehler 4 (out of memory) auftreten kann, wenn nicht genügend RETURN-Anweisungen vorhanden sind.

Format
GOSUB int-num-expr

GOTO

Taste
G

Statement/Befehl

Durch GOTO geht ein Programm zu einer bestimmten Zeile.

Benutzung von GOTO
GOTO kann als direkter Befehl benutzt werden, um ein Programm ab einer gegebenen Zeilennummer zu fahren, *ohne* erst den Schirm freizumachen. Es kann außerdem verwendet werden, um eine Answeisung in einem Programm zu bilden. GOTO folgt ein numerischer Wert, zum Beispiel

```
60 GOTO 350
```

Bei der Ausführung wird der nach GOTO stehende Wert auf die nächste ganze Zahl abgerundet und das Programm geht zu der Zeilennummer mit diesem Wert. Die Verwendung einer Variablen oder eines Ausdrucks ermöglicht es dem Programm, zu einer berechneten Zeilennummer zu springen. Wenn diese Zeile nicht besteht, springt das Programm trotzdem und macht bei dem ersten Statement weiter, das ihm begegnet.

Format
GOTO int-num-expr

IF

Taste
U

Statement/Befehl

IF wird immer mit dem Schlüsselwort THEN benutzt, um eine Entscheidung herbeizuführen, die die darauffolgende Ausführung beeinflußt. Um dies zu tun, prüft der Computer etwas, um herauszufinden, ob es wahr ist oder nicht. Wenn es wahr ist, folgt eine Vorgehensweise, wenn es nicht wahr ist, folgt eine andere.

Benutzung von IF und THEN
IF bildet normalerweise eine Anweisung mit THEN. Ihm folgt zunächst ein numerischer Wert oder eine Bedingung und zweitens das Wort THEN und ein oder mehrere gültige BASIC-Statements, zum Beispiel

```
80 IF x THEN GOGO 250
240 IF a$ = "NEIN" THEN PRINT
"ENDE": STOP
```

Eine Konstante, Variable oder ein Ausdruck werden als wahr erachtet, wenn sie einen Nicht-Nullwert haben. In diesem Fall werden das Statement nach THEN und eventuelle weitere Anweisungen *in derselben Zeile* ausgeführt. Das Programm geht dann zur nächsten Zeile über. Wenn der Wert 0 ist, wird die Konstante oder die Variable oder der Ausdruck als falsch betrachtet. Die folgenden Anweisungen werden dann nicht ausgeführt und das Programm springt zur nächsten Zeile. In unserem Beispiel geht das Programm nicht zu Zeile 250, wenn x Null ist.

Wenn eine Bedingung (a$"NEIN") nach IF wahr ist, werden die Anweisungen nach THEN ausgeführt. Wenn die Bedingung falsch ist, geht das Programm zur nächsten Zeile über.

In diesem Beispiel wird, wenn a$ den Wert "NEIN" hat, "ENDE" angezeigt und das Programm hört auf. Wenn a$ einen anderen Wert hat, macht das Programm ab der nächsten Zeile weiter.

Der Spectrum gibt einer wahren Bedingung den Wert 1 und einer falschen Bedingung den Wert 0. Er erkennt Nicht-Nullwerte als wahr an und Null-Werte als falsch. Einer Variablen kann der Wert einer Bedingung zugeordnet werden durch eine Anweisung wie

```
70 LET x=a$="NEIN"
```

Beachten Sie, daß im Gegensatz zu einigen anderen BASIC-Versionen, THEN nicht vor GOTO ausgelassen werden kann.

Format
IFnum-expr THEN statement [:statement]
IFcond THEN statement [:statement]

IN

Taste
EXTEND MODE
SYMBOL SHIFT I

Funktion

IN überprüft den Status der Tastatur und anderer Input- und Output-Vorrichtungen. Es liest ein Byte von einer gegebenen Port-Adresse, welches den Status der Vorrichtung angibt, die an den Port angeschlossen ist.

Benutzung von IN
In folgt ein numerischer Wert, z.B.

```
150 LET x=IN y
```

Der Wert hinter IN kann von 0 bis 65535 gehen und bestimmt die Port-Adresse, die gelesen werden soll. IN ergibt dann das von diesem Port (=Baustein) gelesene Byte.

Tastatur-Adressen
Die Tastatur hat acht Adressen, von denen jede einen von fünf verschiedenen Bytes enthalten kann, jenachdem, welche Taste gedrückt wird. Die Adressen sind 65278, 65022, 64510, 63486, 61438, 57342, 49150 und 32766. Byte-Werte an diesen Adressen können sein 175, 183, 187, 189 oder 190.

Format
IN num-const
IN num-var
IN (num-expr)

Zeichensatzcode 191

INK

Taste
EXTEND MODE
SYMBOL SHIFT X

Statement/Befehl

INK bestimmt die Vordergrundsfarbe, in der die Zeichen gezeigt, Punkte geplottet und Linien und Kurven gezeichnet werden.

Benutzung von INK
INK kann als direkter Befehl verwendet werden, ist jedoch normalerweise ein Statement in einem Programm. Ihm folgt ein numerischer Wert, zum Beispiel

```
70 INK x
```

Der Wert hinter INK wird auf den nächsten ganzen Wert gerundet und kann dann von 0 bis 9 gehen. Die folgenden Vordergrundsfarben erscheinen:

0 schwarz
1 blau
2 rot
3 violett
4 grün
5 hellblau (Cyan)
6 gelb
7 weiß
8 transparent
9 kontrastierendes Schwarz oder Weiß

INK 8 bestimmt, daß die bestehende Farbe unverändert bleibt bei jeder Position auf dem Bildschirm, an der INK 8 benutzt wird. INK 9 führt dazu, daß die INK-Farbe entweder schwarz oder weiß ist, so daß sie sich gegen die Paper-Farbe (Hintergrund) abhebt.

Globale oder örtlich begrenzte Ink-Farben
Wenn INK alleine ein Statement bildet, wie in obigem Beispiel, ist die Farbe global und alle darauffolgenden Displays erfolgen in dieser Vordergrundfarbe. INK kann auch in Display-Statements eingebettet (eingesetzt) werden, die durch PRINT, INPUT, PLOT, DRAW und CIRCLE gebildet werden. INK steht hinter dem Schlüsselwort, aber vor den Daten oder Displayparametern, ihm folgen dieselben Werte *und ein Semikolon*, zum Beispiel

```
60 CIRCLE INK 4; 128, 88, 87
```

Die Wirkung von INK ist dann örtlich begrenzt und bezieht sich nur auf die dargestellten Zeichen, geplotteten Punkte oder gezeichneten Linien, die durch das Display-Statement erzeugt wurden, wobei dieses Beispiel einen grünen Kreis zeichnet.

Danach geht die Ink-Farbe zu der globalen Farbe oder der standardmäßigen Farbe schwarz zurück.

Format
INK int-num-expr [;]

INKEY$ INput KEY string

Taste
EXTEND MODE
N

Funktion

INKEY$ wird benutzt, um das Anschlagen der Tasten auf der Tastatur festzustellen.

Benutzung von INKEY$
INKEY$ braucht kein Argument und wird allgemein verwendet, um ein Zeichen einer String-Variablen zuzuordnen oder um auf ein bestimmtes Zeichen hin zu prüfen, zum Beispiel

```
70 LET a$=INKEY$1
130 IF INKEY$="N" THEN STOP
```

Bei der Ausführung liefert INKEY$ das Zeichen, das durch die in dem Moment angeschlagene Taste gegeben wird. Wenn keine Taste gedrückt wird, liefert INKEY$ einen (leeren) Null-String (""). Bitte beachten, daß INKEY$ zwischen Groß- und Kleinbuchstaben unterscheidet und anderen umgeschalteten und zurückgeschalteten Zeichen. (Benutzen Sie IN, um eine Taste ohne kennzeichnende Zeichen zu finden).

Im Gegensatz zu INPUT, wartet INKEY$ nicht, sondern geht sofort zum nächsten Statement. Es wird daher normalerweise in eine Schleife gesetzt, welche wiederholt, bis die erforderliche Taste gedrückt wird.

Beispiel
Diese Zeile hält alle Operationen an, bis die Taste Y gedruckt wird (ohne CAPS SHIFT oder CAPS LOCK).

```
60 IF INKEY$ <> "y" THEN GOTO 60
```

Format
INKEY$

INPUT

Taste
N

Statement/Befehl

INPUT macht es möglich, daß Daten während des Fahrens eines Programms eingegeben werden können.

Benutzung von INPUT
INPUT bildet normalerweise eine Anweisung in einem Programm und wird sehr ähnlich wie PRINT verwendet. In seiner einfachsten Form folgt ihm eine numerische oder eine String-Variable, zum Beispiel

```
60 INPUT x
90 INPUT a$
```

Der Computer wartet dann, bis entweder eine Zahl oder ein String eingegeben worden sind. Der Wert wird am Anfang der untersten Zeile dargestellt, während er eingetastet wird. Beim Drücken von ENTER wird der Wert der genannten Variablen zugeordnet und das Programm wird fortgesetzt.

Eine INPUT-Anweisung kann mehr als eine Variable enthalten und wird Zeichen darstellen, um ein 'Prompt' zu bilden. Dies geschieht genauso wie bei PRINT, mit der Benutzung von Anführungsstrichen zum Einschließen der Prompt-Zeichen und Semikolons oder Kommata, wie es erforderlich ist, um einzelne Teile voneinander zu trennen. Display-Statements wie INK, FLASH und PAPER können eingebettet werden, zum Beispiel

```
80 INPUT INK2; "Was ist dein Name?";n$, ("Wie alt bis du," +n$ +"?"); Alter
```

Hier die wichtigsten Unterschiede im Vergleich zu PRINT. INPUT wartet, wenn es an eine Variable kommt, es müssen also alle Variablen und Ausdrücke (so wie der oben mit n$), die in 'prompts' enthalten sein sollen *in Klammern eingeschlossen* werden. Display fängt am Anfang der unteren Zeile an und rollt dann nach oben, wenn mehr als eine Zeile gebraucht werden. AT kann in einem INPUT-Statement verwendet werden genauso wie mit PRINT. AT 0,0 zeigt an am Beginn der Zeile über der untersten und das Display rollt hoch, wenn mehr als zwei Zeilen gebraucht werden.

Wie man INPUT anhält
Folgt INPUT eine numerische Variable und STOP wird eingegeben, hält das Programm an. Mit einer String-Variablen kann das erste Anführungszeichen, das auftritt, gelöscht werden und dann STOP eingegeben werden, um das Programm anzuhalten.

Benutzung von INPUT und LINE
INPUT LINE kann nur mit String-Variablen benutzt werden. Normalerweise verursacht INPUT mit einer String-Variablen das Display von zwei Anführungsstrichen. Während der String eingetastet wird, erscheint er zwischen den Anführungsstrichen. Um diese Anführungsstriche zu entfernen, wird INPUT LINE, gefolgt von einer String-Variablen, benutzt. Ist ein 'prompt' erforderlich, steht es zwischen INPUT und LINE, zum Beispiel

```
70 INPUT "Was ist dein Name?";LINE n$
```

Format
INPUT [prompt] [;] [,] ['] num-var
INPUT [prompt] [;] [,] ['] string-var
INPUT [prompt] [;] [,] ['] LINE string-var
[prompt]=[string-const] [(string-expr)] [AT int-num-expr, int-num-expr] [statement] [;] [,] ['] [,]

INT INTeger

Taste
EXTEND MODE
R

Funktion

INT verwandelt Nicht-Integer (Zahlen, die keine ganzen Zahlen sind) in Integer oder ganze Zahlen.

Benutzung von INT
INT folgt ein numerischer Wert, zum Beispiel

```
70 LETx=INTy
```

Ein Ausdruck muß *in Klammern eingeschlossen* werden. INT liefert dann den auf einen Integer *abgerundeten* Wert.

Beispiel
Der Befehl

```
PRINT INT 45.67,INT-7.66
```

zeigt an

45	-8

Format
INT num-const
INT num-var
INT (num-expr)

INVERSE

Taste
EXTEND MODE
SYMBOL SHIFT M

Statement/Befehl

INVERSE führt dazu, daß Farben an Zeichenpositionen so umgekehrt werden, daß die Ink-Farbe die Paper-Farbe wird und umgekehrt.

Benutzung von INVERSE
INVERSE wird normalerweise zue Bildung einer Anweisung in einem Programm verwendet. Ihm folgt ein numerischer Wert, zum Beispiel

```
70 INVERSE 1
```

Der Wert nach INVERSE wird auf den nächsten Integer aufgerundet und kann dann entweder 0 oder 1 sein. INVERSE 1 führt dazu, daß alle

danach durch PRINT oder INPUT erzeugten Displays in umgekehrten Farben erscheinen. INVERSE Ø setzt Ink- und Paperfarben wieder in den Normalzustand zurück.

Bitte beachten, daß INVERSE in Display-Statements eingebettet werden kann auf die gleiche Art und Weise wie INK. Wird INVERSE 1 jedoch mit CIRCLE, PLOT, oder DRAW verwendet, wird eine Linie oder ein Punkt in der Paperfarbe geplottet, so daß er verschwindet.

Format
INVERSE int-num-expr

LEN LENgth of string

Taste
EXTEND MODE
K

Funktion

LEN gibt die Länge eines String.

Benutzung von LEN
LEN folgt ein numerischer Wert, zum Beispiel

```
5Ø LET x=LENa$
```

Ein Ausdruck muß *in Klammern eingeschlossen werden*. LEN liefert die Anzahl Zeichen in dem String.

Beispiel
Die folgende Zeile

```
12Ø INPUT a$ : IF LEN a$ > 9
   THEN GOTO 12Ø
```

akzeptiert nur Strings mit bis zu neun Zeichen.

Format
LEN string-const
LEN string-var
LEN (string-expr)

LET

Taste
L

Statement/Befehl

LET wird verwendet, um einer Variablen einen Wert zuzuweisen. In Sinclair BASIC kann LET in einem Zuordnungs-Statement nicht ausgelassen werden.

Benutzung von LET
LET bildet normalerweise ein Statement in einem Programm, kann jedoch als direkter Befehl benutzt werden. Ihm folgt eine numerische oder eine String-Variable, ein Gleichheitszeichen und dann ein Wert. Der Wert kann numerisch oder ein String sein, abhängig von der Variablen vor LET, zum Beispiel

```
6Ø LET x=x+1
8Ø LETa$="Korrekt"
```

Der Wert wird dann der Variablen zugewiesen.

Beachten, daß einfache Variablen undefiniert bleiben, bis ihnen durch LET, READ oder INPUT Werte zugewiesen werden.
Array-Variablen werden jedoch auf Ø initialisiert oder einen Null-String (siehe DIM).

Format
LET num-var=num-expr
LET string-var=string-expr

LINE

Taste
EXTEND MODE
SYMBOL SHIFT 3

see **SAVE**

LIST

Taste
K

Befehl/Statement

LIST erzeugt ein Listing des sich gegenwärtig im Speicher befindlichen Programms.

Verwendung von LIST
LIST wird normalerweise als direkter Befehl verwendet, kann jeDoch eine Anweisung in einem Programm bilden. Um ein komplettes Programm aufzulisten, wird es alleine benutzt. Nach dem direkten Befehl

```
LIST
```

erscheint die erste Seite des Listing und darauffolgende Seiten rollen beim Berühren jeglicher Taste – außer N, Leerstelle oder STOP – den Bildschirm hoch.

LIST kann auch eine Zeilennummer folgen, in Form eines numerischen Wertes, zum Beispiel

```
LIST 1ØØ
```

Der Wert nach LIST wird dann auf den nächsten Integer aufgerundet, sofern dies nötig ist, und das Listing beginnt bei dieser Zeile. Gibt es keine Zeile mit der Nummer, beginnt das Listing bei der nächsten Zeile.

Format
LIST [int-num-expr]

LLIST Line printer LIST

Taste
EXTEND MODE
V

Befehl/Statement

LLIST führt dazu, daß Drucker vom Typ Sinclair ein ausgedrucktes Listing des sich gegenwärtig im Speicher befindlichen Programms erzeugen.

Benutzung von LLIST
LLIST wird genauso verwendet wie LIST (weitere Einzelheiten unter LIST). Beachten Sie, daß sich das Bildschirmdisplay nicht verändert, wenn das Listing gedruckt wird.

Format
LLIST [int-num-expr]

LN Logarithm (Natural)

Taste
EXTEND MODE
Z

Funktion

LN liefert der natürlichen Logarithmus (auf Basis e) eines Wertes. Es dient als die Umkehrung von EXP.

Benutzung von LN
LN folgt ein numerischer Wert, zum Beispiel

```
6Ø LETx=LN y
```

Ein Ausdruck muß *in Klammern eingeschlossen werden*. Der Wert nach LN muß großer sein als Ø. LN liefert dann den natürlichen Logarithmus dieses Wertes.

Format
LN num-cons
LN num-var
LN (num-expr)

LOAD

Taste
J

Befehl/Statement

LOAD wird normalerweise als direkter Befehl benutzt, kann jedoch eine Anweisung in einem Programm bilden, um ein neues Programm zu laden. LOAD folgt ein Dateiname, welcher ein String-Wert bis zu zehn Zeichen lang ist, zum Beispiel

```
LOAD "filename"
```

Bei der Ausführung wird das sich gegenwärtig im Speicher befindliche Programm und alle Werte seiner Variablen gelöscht. Der Spectrum sucht dann das genannte Programm und lädt es, wenn er es gefunden hat. Beachten, daß der Computer zwischen Groß- und Kleinbuchstaben unterscheidet in Programmnamen.

Steht nach LOAD ein Null-String, wie in diesem Befehl

```
LOAD " "
```

lädt der Spectrum das erste komplette Programm, das er findet.

Beachten, daß LOAD anders benutzt wird, wenn ein Microdrive angeschlossen ist. Einzelheiten siehe Handbuch Microdrive and Interface 1.

Format
LOAD string-expr

LOAD CODE

Tasten
J
EXTEND MODE
I

Befehl/Statement

LOAD CODE wird benutzt, um einen Teil des Speichers mit Informationen zu laden, die auf Band gespeichert sind. Die Information besteht aus einer Menge von Bytes und diese werden an eine Menge von Adressen im Speicher gesandt. LOAD CODE kann verwendet werden, um ein Display zu laden oder um Informationen für benutzergewählte Grafik zu laden.

Benutzung von LOAD CODE
LOAD CODE kann als direkter Befehl benutzt werden oder eine Anweisung in einem Programm bilden. Ihm folgt ein 'filename' (Dateiname), welcher ein String-Wert ist, und dann CODE, zum Beispiel

```
LOAD "Daten" CODE
```

Der Dateiname nach LOAD ist der Name der zu ladenden Informationen und unterliegt denselben Beschränkungen wie Programmnamen (siehe LOAD). LOAD CODE sucht dann die genannte Information und stellt, wenn es sie gefunden hat, Bytes dar, gefolgt von dem Namen. Der Spectrum lädt dann die Bytes in den Speicher an die Adressen, von denen sie gesichert worden waren. Jegliche bestehende Informationen werden überschrieben.

CODE können auch ein oder mehrere numerische Werte folgen, die durch Kommata getrennt werden, zum Beispiel

```
LOAD "Bild" CODE
16384, 6912
```

Die hinter Code stehenden Werte werden auf den nächsten Integer gerundet und definieren dann die Startadresse (oben 16384), bei der die genannte Information geladen werden soll und die Anzahl von Bytes, die an Stellen, welche bei dieser Adresse beginnen, gesandt werden sollen. (hier 6912). Wenn die Zahl nicht stimmt, erfolgt die 'tape loading error'-Mitteilung. Wenn hinter CODE nur ein Wert steht, definiert dieser die Startadresse, ab der *alle* Bytes eingesetzt werden müssen.

Das obige Beispiel kann auch durch die Schlüsselwörter LOAD SCREEN$ ausgeführt werden.

Einzelheiten über die Speicherung von Bytes siehe SAVE CODE.

Format
LOAD string-expr CODE
[int-num-expr] [,int-num-expr]

LOAD DATA

Tasten
J
EXTEND MODE
D

Statement/Befehl

LOAD DATA wird verwendet, um Arrays von Band zu laden. Die Arrays werden durch SAVE DATA aufgenommen.

Benutzung von LOAD DATA
LOAD DATA kann benutzt werden, um eine Anweisung in einem Programm zu bilden oder als direkter Befehl. LOAD folgt zunächst ein 'filename', welcher ein String-Wert ist. Danach folgen DATA und ein Buchstabe oder ein Buchstabe und ein $ und am Ende *leere* Klammern, zum Beispiel

```
270 LOAD "Zahlen"
DATA n()
300 LOAD "Namen" DATA n$()
```

Der 'filename' hinter LOAD ist der Name, der dem Array auf Band gegeben wird und er unterliegt denselben Beschränkungen wie mit LOAD verwendete Programmnamen. Der Buchstabe oder Buchstabe $ nach DATA ist der Name, den das Array im Programm erhalten soll, wenn es geladen ist und benutzt wird.

Bei der Ausführung sucht der Spectrum das genannte Array auf dem Band. Wenn er es gefunden hat, erscheint die Nachricht Number array: oder Character array:, gefolgt von dem Namen, auf dem Schirm und das Array wird geladen. Jedes sich im Speicher befindliche Array mit demselben Namen (n oder n$ oben) wird gelöscht und ein neues Array mit diesem Buchstabennamen und den auf Band gespeicherten Werten entsteht. Beachten, daß bei Zeichenarrays jede sich im Speicher befindliche String-Variable desselben Namens ebenfalls gelöscht wird.

Format
LOAD string-expr DATA
letter [$] ( )

LOAD SCREEN$

Tasten
J
EXTEND MODE
K

Statement/Befehl

LOAD SCREEN$ ermöglicht es, daß ein Bildschirmdisplay direkt von Band geladen werden kann. Es sendet Informationen vom Band an den Teil des Speichers, welches das Bildschirmdisplay steuert, um das Bild zu erzeugen.

Benutzung von LOAD SCREEN$
LOAD SCREEN$ kann zur Bildung eines Statement in einem Programm verwendet werden oder als direkter Befehl. LOAD folgt ein Dateiname, welcher ein String-Wert ist, und dann SCREEN$, zum Beispiel

```
LOAD "Bild" SCREEN$
```

Der Dateiname nach LOAD ist der Name, der der Bildschirminformation auf Band gegeben worden ist, und er unterliegt denselben Beschränkungen wie mit LOAD benutzte Programmnamen. Der Spectrum sucht dann die genannte Information und lädt sie, wenn er sie gefunden hat, zuerst in die Displaydatei und dann den Attributenteil des Speichers. Das Bild baut sich langsam in den gegenwärtigen Ink- und Paperfarben auf und dann werden die Attribute (wirkliche Farben und so weiter) hinzugefügt.

Einzelheiten über die Speicherung von Bildschirminformationen siehe SAVE SCREEN$

Format
LOAD string-expr SCREEN$

LPRINT Line printer PRINT

Taste
EXTEND MODE
C

Statement/Befehl

LPRINT sorgt dafür, daß Drucker vom Typ Sinclair einen Datenposten genauso drucken, wie PRINT den Posten auf dem Bildschirm erscheinen läßt.

Benutzung von LPRINT
LPRINT kann eine Anweisung in einem Programm bilden oder ein direkter Befehl sein. Ihm folgen Datenposten, die durch Semikolon, Komma oder Apostroph voneinander getrennt werden können, zum Beispiel

```
60 LPRINT "Nummer";x'
   "Name";n$, "Alter";a
```

Wenn sie an den Drucker ausgegeben werden, werden diese Datenposten in demselben Format gedruckt, wie sie durch PRINT auf dem Bildschirm erscheinen würden. Eine LPRINT-Anweisung kann auch TAB-Statements enthalten, bestimmte CHR$-Steuerungen, INVERSE und OVER-Statements und Steuercodes, alle mit demselben Effekt wie bei PRINT. Ein AT-Statement kann ebenfalls eingeschlossen werden, die Zeilennummer wird jedoch ignoriert und der Datenposten bei der gegebenen Spaltenposition in derselben Zeile gedruckt.

Format
LPRINT [**TAB** int-num-expr;)]
[**AT** int-num-expr, int-num-expr;]
[**CHR$** (int-num-expr);]
[statement;] [num-expr]
[string-expr] [;] [,] ['']

MERGE

Taste
EXTEND MODE
SYMBOL SHIFT T

Statement/Befehl

MERGE ermöglicht es, zwei Programme zu mischen.

Benutzung von MERGE
MERGE kann zur Bildung einer Anweisung in einem Programm benutzt werden oder als direkter Befehl. Ihm folgt ein Dateiname in Form eines String-Wertes, zum Beispiel

```
500 MERGE "Prog2"
```

Der Dateiname nach MERGE ist der Name des Programmes, welches mit dem sich gegenwärtig im Programm befindlichen Programm gemischt werden soll. Dieser Name unterliegt denselben Beschränkungen wie mit LOAD verwendete Programmnamen. MERGE lädt das neue Programm, ohne das bestehende Programm zuerst zu

löschen. Das neue Programm überschreibt jedoch alle Zeilen im bestehenden Programm, deren Zeilennummern dieselben sind wie die im neuen Programm, und Variablen desselben Namens werden ebenfalls gelöscht.

Format
MERGE string-expr

MOVE

Microdrive-Befehl zur Handhabung von Dateien Siehe Handbuch Microdrive and Interface 1

NEW

Taste
A

Befehl/Statement

NEW löscht den BASIC-Speicherbereich (den Bereich bis zu RAMTOP) und entfernt jedes sich gegenwärtig in der Speicherstelle befindliche Programm.

Benutzung von NEW
NEW wird normalerweise als direkter Befehl verwendet, kann jedoch auch eine Anweisung in einem Programm bilden. Es wird für sich allein benutzt. Bei der Ausführung werden Programm und Variable gelöscht. Der Speicher wird bis zu RAMTOP freigemacht, so daß benutzergewählte Grafik, die oberhalb von RAMTOP gespeichert wird, unbeeinträchtigt bleibt.

Format
NEW

NEXT

Taste
N

Statement/Befehl

NEXT wird immer in Verbindung mit FOR benutzt, um eine FOR NEXT Schleife zu bilden.

Benutzung von NEXT
NEXT wird normalerweise zur Bildung einer Anweisung in einem Programm benutzt, um eine FOR NEXT Schleife zu vervollständigen. Ihm folgt ein Buchstabe, welcher die Steuervariable in der Schleife ist, zum Beispiel

```
90 NEXT a
```

In der Sinclair-BASIC-Sprache muß die Steuervariable eingeschlossen werden.

Weitere Einzelheiten über FOR siehe FOR NEXT Schleife.

Format
NEXT buchstabe

NOT

Taste
SYMBOL SHIFT S

Boolescher Operator/Funktion

NOT wird verwendet, um die Wahrheit einer Bedingung umzukehren, so daß eine falsche Bedingung wahr wird und eine wahre Bedingung falsch.

Benutzung von NOT
NOT folgt eine Bedingung oder ein numerischer Wert, z.b.

```
90 IF NOT x=y+z THEN
   PRINT "Falsch"
```

```
90 LET korrekt=x=y+z:IF NOT
korrekt THEN PRINT "falsch"
```

When NOT eine Bedingung folgt (x=y+z oben), ordnet der Spectrum zuerst einen Wert von 1 der Bedingung zu, wenn sie wahr ist, und Ø, wenn sie falsch ist. NOT dient dann als Funktion und kehrt die erzeugten Wert um, so daß die umgekehrte Bedingung geprüft werden kann. Bitte beachten, daß eine Bedingung in Klammern stehen muß, wenn sie AND oder OR enthält.

Folgt NOT ein numerischer Wert, ergibt es Ø, wenn der Wert nicht Null ist, und 1, wenn der Wert Ø ist. In den obigen Beispielen druckt der Spectrum also "falsch", wenn x <> y+z ist oder wenn '*korrekt*' einen Wert von Ø hat.

Format
LOAD cond
NOT num-expr

OPEN

Microdrive-Befehl zur Handhabung von Dateien Siehe Handbuch Microdrive and Interface 1

OR

Taste
SYMBOL SHIFT U

Boolescher Operator/Funktion

OR dient als Boolescher Operator,um die Wahrheit einer Kombination von Bedingungen zu testen. Wenn eine oder mehrere der Bedingungen wahr sind, dann ist die Gesamtkombination wahr. OR dient außerdem als Funktion zur Durchführung von binären Operationen an zwei numerischen Werten.

Benutzung von OR
Als Boolescher Operator verbindet OR zwei Bedingungen in einem Statement, wenn die Wahrheit des Ganzen zu prüfen ist, z.B.

```
70 IF INKEY$="N" OR
   INKEY$="n" THEN STOP
```

Ist eine oder sind beide der Bedingungen wahr, ist die Gesamtkombination wahr. In der Zeile oben wird eine der Bedingungen (INKEY$="N" und INKEY$="n") wahr, sobald die Taste N gedrückt wird, unabhängig davon, ob CAPS SHIFT oder CAPS LOCK benutzt werden oder nicht. Die ganze Kombination ist dann wahr und das Programm hält an.

OR als Funktion
Der ZX Spectrum + ordnet einer wahren Bedingung einen numerischen Wert von 1 zu und einer falschen Bedingung einen Wert von Ø. Er erkennt jeden Nicht-Nullwert als wahr an und Ø als falsch. OR kann daher ein numerischer Wert vor- oder nachgestellt werden, z.B.

```
40 LET x=y OR z
```

Der variablen x wird dann ein Wert 1 zugeordnet, wenn z nicht Null oder eine wahre Bedingung ist, oder ein Wert von y,wenn z Ø oder eine falsche Bedingung ist.

Beachten Sie, daß der Spectrum Kombinationen von numerischen Werten nicht in Übereinstimmung mit standardmäßigen Wahrheitstabellen auswertet.

Format
cond OR cond
num-expr OR num-expr

OUT

Taste
EXTEND MODE
SYMBOL SHIFT O

Statement/Befehl

OUT sendet ein Byte an eine gegebene Input-Output-Portadresse, um eine Ausgabevorrichtung zu betreiben.

Benutzung von OUT
OUT kann zur Bildung einer Anweisung in einem Programm verwendet werden oder als direkter Befehl. Ihm folgen zwei numerische Werte, die durch Komma getrennt werden, zum Beispiel

```
40 OUT 254,3
```

Beide Werte werden auf die nächste ganze Zahl gerundet. Der erste Wert (oben 254) kann dann von Ø bis 65535 gehen und ist die Portadresse. Der zweite Wert (3) kann von Ø bis 255 gehen und ist das an diese Adresse zu sendende Byte.

Bits 0 bis 2 des an Portadresse 254 ausgegebenen Bytes setzen die Borderfarbe; das obige Beispiel macht daher die Umrandung violett. Bit 3 an dieser Adresse betreibt die MIC-Buchse und Bit 4 den Lautsprecher. Portadresse 251 treibt den Drucker an und Port 254, 247 und 239 werden mit anderen Peripheriegeräten verwendet.

Format
Out int-num-expr, int-num-expr

OVER

Taste
EXTEND MODE
SYMBOL SHIFT N

Statement/Befehl

OVER wird benutzt, um ein Zeichen über ein anderes zu schreiben. Es kann auch verwendet werden, um in einer Paperfarbe statt in einer Inkfarbe Punkte zu plotten und Linien oder Kurven zu zeichnen.

Benutzung von OVER
OVER wird normalerweise zur Bildung einer Anweisung in einem Programm benutzt. Ihm folgt ein numerischer Wert, zum Beispiel

```
80 OVER 1
```

Der Wert nach OVER wird auf die nächste ganze Zahl gerundet und kann dann entweder 0 oder 1 sein. OVER 0, welches der standardmäßige (voreingestellte) Zustand ist, führt dazu, daß ein Zeichen ein vorheriges Zeichen an derselben Zeichenposition auslöscht und durch ein anderes ersetzt. OVER 1 führt dazu, daß zwei an derselben Zeichenposition erscheinenden Zeichen kombiniert werden.

OVER kann in ein PRINT-oder ein INPUT-Statement eingebettet werden auf dieselbe Art und Weise wie INK, so daß es nur die durch das Statement dargestellten Zeichen beeinflußt. Diese Anweisung unterstreicht zum Beispiel ein Wort

```
60 PRINT AT 11,15;"JA";
OVER 1;AT 11,15;"__"
```

Beachten Sie jedoch, daß Zeichen so kombiniert werden, daß die Paperfarbe geliefert wird, wo sich die Inkfarben überschneiden.

OVER in Hochauflösung
OVER kann mit PLOT, DRAW und CIRCLE verwendet werden. Ohne OVER können Linien und Kurven sich überlappen, aber sie müssen dieselbe Inkfarbe haben, weil sich sonst die Inkfarbe an der ganzen Zeichenposition ändert, wo sie sich überschneiden. Wird OVER 1 benutzt, liefern Linien und Kurven da die Paperfarbe, wo sie sich überlappen oder auf Zeichen treffen. Ein Plotten von Punkten oder Zeichnen von Linien und Kurven wiederum an genau derselben Stelle mit OVER 1 führt dazu daß sie verschwinden.

Format
OVER int-num-expr

PAPER

Taste
EXTEND MODE
SYMBOL SHIFT C

Statement/Befehl

PAPER wird verwendet, um die Paper- oder Hintergrundsfarbe für das Bildschirmdisplay zu wählen. Dies kann entweder die Farbe des Hintergrunds über die ganze Displayfläche sein oder die Farbe hinter einzelnen Zeichen, Punkten oder Linien, die in Einzelzeichenpositionen erscheinen.

Benutzung von PAPER
PAPER kann zur Bildung einer Anweisung in einem Programm verwendet werden oder als direkter Befehl. Ihm folgt ein numerischer Wert, zum Beispiel

```
80 PAPER x
```

Der Wert hinter PAPER wird auf die nächste ganze Zahl gerundet und kann dann von 0 bis 9 gehen. Die dann gelieferten Paperfarben sind dieselben wie die mit INK. Paperfarben können ebenfalls global sein oder lokalisiert werden, indem sie in Display-Statements eingebettet (eingesetzt) werden, genauso, wie dies bei den Inkfarben geschieht. Weitere Einzelheiten siehe INK.

Immer, wenn Zeichen nach einer PAPER-Anweisung gedruckt werden, ob sie nun global oder örtlich begrenzt sind, ändert sich der Hintergrund der ganzen Zeichenposition in die gewählte Farbe um. Dies ist auch der Fall, wenn Punkte geplottet oder Linien und Kurven gezeichnet werden mit einem eingebetteten PAPER-Statement, aber nicht nach einem globalen Befehl oder Statement.

Um einen farbigen Hintergrund auf der gesamten Displayfläche zu erzeugen, muß man CLS nach der PAPER-Anweisung benutzen. Das ganze Display erhält dann diese Farbe, die die allgemeine Hintergrundsfarbe bleibt.

Format
PAPER int-num-expr [;]

PAUSE

Taste
M

Statement/Befehl

PAUSE kann verwendet werden, um ein Programm für eine bestimmte oder unbestimmte Zeit anzuhalten.

Benutzung von PAUSE
PAUSE wird normalerweise zur Bildung einer Anweisung in einem Programm verwendet. Ihm folgt ein numerischer Wert, zum Beispiel

```
130 PAUSE 100
```

Der Wert nach PAUSE wird auf die nächste ganze Zahl gerundet und kann dann von 0 bis 65535 gehen. Er definiert die Verzögerung, welche entritt, als diese Anzahl von Einzelbildern des Fernsehens, so daß ein Wert von 50 eine Pause von 1 Sekunde in Ländern erzeugt, in denen die Einzelbilderfrequenz 50Hz ist.

Beachten Sie jedoch, daß jede Pause verkürzt werden kann durch das Drücken einer Taste und daß PAUSE 0 eine unbegrenzte Pause ergibt, die andauert, bis eine Taste angeschlagen wird.

Format
PAUSE in-num-expr

PEEK

Taste
EXTEND MODE
O

Funktion

PEEK liefert den Wert des an einer bestimmten Adresse im Speicher gespeicherten Bytes.

Benutzung von PEEK
PEEK folgt ein numerischer Wert, zum Beispiel

```
80 LET x=PEEK(256*y)
```

Bitte beachten, daß ein Ausdruck *in Klammern eingeschlossen werden muß*. Der Wert hinter PEEK wird auf den nächsten Integer (ganze Zahl) gerundet, wenn nötig, und kann dann von 0 bis 65535 gehen, um eine Adresse im Speicher zu ergeben. PEEK liefert dann den Wert des Bytes (eine Zahl von 0 bis 255) an der spezifizierten Adresse.

Beispiel
Die Anzahl von Einzelbildern des Fernsehdisplays, die auftraten, seit der Spectrum zuletzt angeschaltet wurde, wird bei Adressen 23672 bis 23674 gespeichert. Da die Einzelbilder mit einer regelmäßigen Geschwindigkeit erzeugt werden, bietet ein PEEKen dieser Speicherstellen eine Methode der Zeitmessung. Die folgende Zeile zeigt die Zeit in Sekunden, seit der Spectrum zuletzt eingeschaltet wurde (minus Zeit für Tonerzeugung und Betreiben von Peripheriegeräten wie Kassettenspieler und Drucker).

```
10 PRINT (PEEK 23672+256*
   PEEK 23673+65536*PEEK
   23674)/50
```

Anmerkung: *Ist die Netzstromfrequenz 60 Hz und nicht 50 Hz verändern Sie 50 in 60 um.*

Format
PEEK int-num-const
PEEK int-num-var
PEEK (int-num-expr)

PI

Taste
EXTEND MODE
M

Funktion

PI gibt den Wert von pi (π) zur Benutzung in Berechnungen.

Benutzung von PI
PI braucht keinerlei Werte oder Variable, wenn es in einer Anweisung oder einem Befehl verwendet wird, zum Beispiel

```
DRAW 255,0,-PI
```

PI liefert einen Wert von 3.1415927, so daß der obige Befehl einen großen Halbkreis auf den Bildschirm zeichnet.

Format
PI

PLOT

Taste
O

Statement/Befehl

PLOT wird in hochauflösender Grafik benutzt, um ein Pixel oder einen Punkt Farbe an einer bestimmten Position auf dem

Bildschirm zu plotten.

Benutzung von PLOT
PLOT wird zur Bildung einer Anweisung in einem Programm verwendet oder als Befehl. Ihm folgen normalerweise zwei numerische Werte, die durch Komma getrennt werden, zum Beispiel

```
50 PLOT 128,87
```

Beide Werte hinter PLOT werden auf die nächsten ganzen Zahlen gerundet, sofern dies nötig ist. Der erste Wert kann dann von Ø bis 255 gehen und definiert die horizontale Koordinate einer Position auf dem Bildschirm. Der zweite Wert kann von Ø bis 175 gehen und definiert eine vertikale Koordinate. Ein Pixel wird dann normalerweise in der gegenwärtigen Inkfarbe an der angegebenen Position geplottet.

Beachten Sie die folgenden Effekte von Farbanweisungen oder -befehlen auf PLOT. Nach OVER1 wird ein bestehender Punkt an derselben Stelle in die Paperfarbe umgewandelt. Nach INVERSE 1 wird der Punkt in der gegenwärtigen Paperfarbe geplottet. Nach BRIGHT 1 oder FLASH 1 wird die ganze Zeichenposition auf dem Schirm mit geringer Auflösung, auf dem das Pixel geplottet ist, entweder hell sein oder blinken.

Diese vier Schlüsselwörter und INK können auch in einem PLOT-Statement eingebettet (eingesetzt) sein genauso wie bei PRINT, zum Beispiel

```
160 PLOT INK 2;x,y
```

Ihre Wirkung ist dieselbe, ist jedoch lokalisiert und begrenzt auf das durch die Anweisung geplottete Pixel. Wenn PAPER in ein PLOT-Statement eingebettet wird, ändert sich die Paperfarbe der ganzen Zeichenposition um den Pixel in die gegebene Farbe.

Format
PLOT [statement;]
int-num-expr,int-num-expr

POINT

Taste
EXTEND MODE
SYMBOL SHIFT 8

Funktion

POINT wird verwendet, um herauszufinden, ob die Farbe an einer bestimmten Position des hochauflösenden Bildschirms eine Ink- oder eine Paperfarbe ist.

Benutzung von POINT
POINT folgen zwei numerische Werte, *die durch Komma getrennt und in Klammern eingeschlossen werden*, zum Beispiel

```
240  IF POINT (x,y)=1 THEN
     GOSUB 600
```

Die zwei Werte hinter POINT werden auf Integer gerundet, sofern dies erforderlich ist. Der erste Wert kann dann von Ø bis 255 gehen und definiert die horizontale Koordinate eines Pixels auf dem Schirm. Der zweite Wert kann von Ø bis 175 gehen und definiert eine vertikale Koordinate. POINT liefert dann 1, wenn das Pixel an der definierten Position eine Ink-Farbe ist, oder Ø, wenn es eine Paper-Farbe ist.

Format
POINT (int-num-expr, int-num-expr)

POKE

Taste
O

Statement/Befehl

POKE wird verwendet, um den Wert des Bytes an einer bestimmten Adresse im Speicher zu ändern.

Benutzung von POKE
POKE wird zur Bildung einer Anweisung in einem Programm verwendet oder als Befehl. Ihm folgen zwei numerische Werte, die durch Komma getrennt werden, zum Beispiel

```
POKE 23609,255
```

Die zwei Werte hinter POKE werden, wenn nötig, auf die nächsten Integer gerundet. Der erste Wert kann dann von 16384 bis 65535 gehen und ist eine Adresse in RAM. Der zweite Wert kann von Ø bis 255 gehen und ist das Byte, das an die definierte Adresse geschrieben werden soll.

Format
POKE int-num-expr, int-num-expr

PRINT

Taste
P

Statement/Befehl

PRINT stellt Daten auf dem Bildschirm dar. Die Daten können jedes einzelne Zeichen oder Zeichenfolgen sein. Eine PRINT-Anweisung kann andere Schlüsselwörter enthalten, um die Position und Farbe von Daten zu definieren.

Benutzung von PRINT
PRINT kann alleine für sich benutzt werden oder ihm können Daten folgen. Diese Daten können die Form von beliebigen numerischen oder String-Ausdrucken oder einer Mischung von beiden haben.

Wird PRINT mit Daten verwendet wird, müssen zwei oder mehr Posten durch Semikolon, Komma oder Apostroph voneinander getrennt werden.

Gewisse andere Schlüsselwörter können in beliebiger Reihenfolge zwischen PRINT und den Daten eingesetzt werden, vorausgesetzt, daß jede durch das Schlüsselwort gebildete Anweisung durch ein Semikolon endet. Diese Schlüsselwörter sind CHR$, TAB, AT, INK, PAPER, FLASH, BRIGHT, INVERSE und OVER.

PRINT mit Strings

PRINT für sich alleine oder gefolgt von einem Null-String (" ") zeige eine leere Zeile und bewegt der Cursor zum Anfang der nächsten Zeile.

PRINT gefolgt von einer String-Konstanten (beliebigen Zeichen in doppelten Anführungsstrichen) stellt die Zeichen so dar, wie sie zwischen den Anführungsstrichen erscheinen. Der Befehl

```
PRINT "3/542/76/21"
```

zeigt zum Beispiel

```
3/542/76/21
```

PRINT gefolgt von String-Variabler oder-Ausdruck zeigt den String bzw. die Strings, die sie darstellen.

PRINT mit Zahlen

PRINT gefolgt von einem numerischen Ausdruck zeigt den Wert des Ausdrucks. Zahlen werden in Dezimalform mit bis zu acht bedeutsamen Ziffern und ohne Nachlaufende Nullen nach dem Dezimalpunkt aufgezeigt.

Sehr große und sehr kleine Zahlen werden in kürzerer, wissenschaftlicher Schreibweise Angezeight als zwei Zahlen, die durch den Buchstaben E getrennt werden Dies bedeutet eine Zahl, in der der erste Teil (der Mantissenteil) in die Potenz des zweiten Teils (dem Exponenten) erhoben wird. Der Befehl

```
PRINT 3/542/76/21
```

zeigt zum Beispiel an

```
3.4680798E-6
```

Formatierung von PRINT mit Interpunktionszeichen
PRINT gefolgt von Datenposten, die durch ein Semikolon getrennt werden, zeigt die Posten nebeneinander ohne Leerstelle. Der Befehl

```
PRINT 1;2;3
```

zeigt an

```
123
```

PRINT gefolgt von Datenposten, die durch Komma getrennt sind, zeigt jeden Posten am Anfang oder in der Mitte einer Zeile, abhängig von der Position des ersten Postens. Der Befehl

```
PRINT 1,2,3
```

zeigt an

```
1              2
3
```

PRINT gefolgt von Datenposten, die durch Apostroph getrennt sind, zeigt den Posten nach dem Apostroph am Anfang der nächsten Zeile. Der Befehl

```
PRINT 1'2'3
```

zeigt an

```
1
2
3
```

Wenn eine PRINT-Anweisung oder ein Befehl mit Semikolon, Komma oder Apostroph enden, wird der durch das nächste PRINT-Statement angezeigte Posten genauso beeinflußt.

PRINT und andere Schlüsselwörter
PRINT kann außerdem TAB folgen, ein numerischer Wert, ein Semikolon und dann ein Datenposten, zum Beispiel

```
60 PRINT TAB x; a$
```

Der Wert hinter TAB (oben x) wird, wenn nötig, auf die nächste ganze Zahl aufgerundet und wird dann durch 32 geteilt und der Rest wird angegeben und stellt einen Wert zwischen 0 und 31 dar. Der Datenposten wird dann an dieser Spaltenposition in derselben oder der nächsten Zeile angezeigt.

PRINT kann auch AT folgen und dann zwei *durch Komma getrennte* numerische Werte, ein Semikolon und ein Datenposten, zum Beispiel

```
50 PRINT AT l,c;"Daten"
```

Der erste Wert (oben 1) kann von 0 bis 21 gehen und definiert die Nummer der Zeile oder Reihe, in der die Daten angezeigt werden. Der zweite Wert (c) kann von 0 bis 31 gehen und definiert die Nummer der Spalte, in der das erste Zeichen oder die erste Ziffer des Datenpostens angezeigt werden. Nicht-Integer werden akzeptiert und auf die nächste ganze Zahl gerundet. Der Befehl **PRINT AT 11,16;"*"** zeigt einen Stern in der Mitte des Bildschirms an.

PRINT können auch ein oder mehrere CHR$-Funktionen folgen. Einzelheiten siehe CHR$.

PRINT und Farb-Schlüsselwörter
Das von PRINT erzeugte Display wird durch Farbanweisungen oder -befehle mit INK, PAPER, FLASH, BRIGHT, INVERSE und OVER, welche gegenwärtig aktiv sind, beeinflußt. PRINT können auch eines oder mehrere dieser sechs Statements folgen, wobei jedem ein Semikolon folgen muß, bevor der Datenposten erscheint, zum Beispiel

```
50 PRINT AT 11,16; INK 2;
   FLASH 1;"*"
```

Der Datenposten wird dann mit den durch die Schlüsselwörter spezifizierten Attributen angezeigt. Diese Attribute sind örtlich begrenzt und beziehen sich nur auf den angezeigten Posten. Nach Ausführung des PRINT-Statements gehen sie zurück auf ihre standardmäßigen oder zuvor erklärten, globalen Werte. PRINT befolgt auch örtliche Farbsteuercodes, die mit den Daten eingesetzt worden sind (siehe Seite 33).

Format
PAPER [TABint-num-expr;]
[**AT** int-num-expr,
int-num-expr;]
[**CHR$** (int-num-expr);]
[statement;] [num-expr]
[string-expr] [;] [,] ['']

RANDOMIZE

Taste
T

Statement/Befehl

RANDOMIZE, welches auf der Tastatur als RAND erscheint, wird in Verbindung mit RND benutzt, um Folgen von Zahlen zu generieren, die entweder zufällig oder voraussagbar sind.

Benutzung von RANDOMIZE
RANDOMIZE wird entweder zur Bildung eines Statements in einem Programm verwendet oder als Befehl. Ihm folgt wahlweise ein numerischer Wert, zum Beispiel

```
RANDOMIZE 1
10 RANDOMIZE
```

Der Wert hinter RANDOMIZE wird, wenn nötig, auf den nächsten Integer gerundet und kann dann von 0 bis 65535 gehen. Ein Wert über 0 setzt die Systemvariable names SEED auf diesen Wert, nach welcher RND immer dieselbe Folge von Zahlen generiert (Inforation über Systemvariable siehe Seite 48). Die eigentliche Folge hängt vom Wert von RANDOMIZE ab.

Folgen RANDOMIZE 0 oder kein Wert, wird SEED der Wert einer anderen Systemvariablen namens FRAMES gegeben, welche die Einzelbilder zählt, die seit dem Einschalten des Spectrum auf dem Fernseher erschienen sind. Da SEED sich 50 oder 60 Mal in der Sekunde ändert, ist die Folge der Zahlen, die durch RND hinter RANDOMIZE oder RANDOMIZE 0 erzeugt worden sind, höchst zufällig.

Format
RANDOMIZE [int-num-expr]

READ

Taste
EXTEND MODE
A

Statement/Befehl

READ wird in Verbindung mit DATA benutzt, um Variablen Werte zuzuordnen, wobei die Werte in einem DATA-Statement verwendet werden.

Benutzung von READ
READ wird normalerweise zur Bildung einer Anweisung in einem Programm verwendet. Ihm folgen eine oder mehrere numerische Variable oder String-Variable, die durch Kommata getrennt werden, zum Beispiel

```
20 READ a$,x
```

Wenn READ zuerst ausgeführt wird, nimmt es dieselbe Anzahl von Werten, wie es Variable gibt ab Beginn der ersten DATA-Liste und ordnet die Werte der Reihe nach den Variablen zu. Wenn READ jetzt wieder ausgeführt wird, wird der nächste Satz von DATA-Werten den in der READ-Anweisung genannten Variablen zugeordnet, und so weiter.

Weitere Einzelheiten siehe DATA.

Format
READ num-var [,num-var]
[,string-var]
READ string-var [,num-var]
[,string-var]

REM REMark

Taste
E

Statement

REM wird verwendet, um Anmerkungen oder Gedächtnisstützen in ein Programm einzusetzen. Dies können Titel und Autor des Programms sein und Erklärungen von Programmzeilen wie zum Beispiel der Zweck einer Variablen. Die Bemerkungen spielen beim Fahren des Programms keine Rolle und sind nur im Listing zu sehen.

Benutzung von REM
REM bildet entweder eine eigene Zeile in einem Programm oder das letzte Statement in einer Zeile. Ihm folgt jede mögliche, auf der Tastatur einzutippende Bemerkung, zum Beispiel

```
80 INPUT n$: REM n$ ist der Name
```

Wenn der Computer auf REM stößt, ignoriert er alles, was in der Zeile hinter REM folgt.

Format
REM beliebige Zeichen

RESTORE

Taste
EXTEND MODE
S

Statement/Befehl

RESTORE wird in Verbindung mit READ und DATA benutzt, damit READ Werte aus einem bestimmten DATA-Statement nimmt anstatt aus dem ersten oder nächsten DATA-Statement im Programm.

Benutzung von RESTORE
RESTORE bildet normalerweise eine Anweisung in einem Programm. Ihm folgt wahlweise ein numerischer Wert, zum Beispiel

```
160 RESTORE 800
```

Der Wert hinter RESTORE wird, wenn nötig, auf den nächsten ganzen Wert gerundet und müßte dann die Nummer einer Programmzeile mit einem DATA-Statement sein. Nach RESTORE wird die nächste READ-Anweisung die in diesem DATA-Statement enthaltenen Werte zuweisen. Wenn die numerierte Zeile nicht besteht oder kein DATA-Statement enthält, geht

READ zum nächsten DATA-Statement nach dieser Zeile.
Folgen RESTORE eine Ø oder kein Wert, geht das nächste READ-Statement zum ersten DATA-Statement im Programm.

Format
RESTORE [int-num-expr]

RETURN

Taste
Y

Statement/Befehl

RETURN wird verwendet, um eine Subroutine (Unterprogramm) zu beenden und den Computer zum Hauptprogramm oder einer vorhergegangenen Subroutine zurückzubringen.

Benutzung von RETURN
RETURN wird normalerweise zur Bildung einer Anweisung in einem Programm verwendet. Es wird alleine für sich am Ende eines Unterprogrammes benutzt, zum Beispiel

```
1080 RETURN
```

Bei der Ausführung geht das Programm zu dem Statement nach der letzten ausgeführten GOSUB-Anweisung.
Weitere Einzelheiten siehe GOSUB.

Format
RETURN

RND

Taste
EXTEND MODE
T

Funktion

RND wird verwendet, um eine Zufallszahl zu erzeugen.

Benutzung von RND
RND wird alleine in einer Anweisung oder einem Befehl benutzt, zum Beispiel

```
60 LET x=RND
```

RND liefert dann eine zufällige Zahl, die kleiner als 1 ist und größer als oder gleich Ø.
Wenn der Spectrum angeschaltet oder zurückgestellt wird oder wenn NEW benutzt wird, werden Zahlen anschließend von RND in derselben Sequenz geliefert. Diese Sequenz wird generiert durch die Potenzen von 75 (75, 75*75, 75*75*75, und so weiter), das Dividieren jeder Potenz durch 65537 und der ausschließlichen Benutzung des Rests, dann wird 1 von diesem Rest abgezogen und dieses Ergebnis geteilt durch 65536.
Wenn eine noch zufälligere Sequenz oder eine andere feste Sequenz gewünscht wird, wird RANDOMIZE vor RND benutzt.

Zufällige ganze Zahlen
Jede ganze Zahl 1 bis x wird durch INT(RND*x)+1 geliefert. Um einen zufälligen Integer von Ø bis x zu erzeugen, wird INT(RND*x+Ø.5) verwendet.

Format
RND

RUN

Taste
R

Befehl/Statement

RUN führt dazu, daß ein Programm, normalerweise ab der ersten Zeile, gefahren wird.

Benutzung von RUN
RUN kann als direkter Befehl verwendet werden oder es kann eine Anweisung in einem Programm bilden. Ihm folgt wahlweise ein numerischer Wert, zum Beispiel

```
RUN 50
```

Folgt auf RUN kein Wert, fährt das Programm ab der ersten Zeile. Ist ein Wert enthalten, wird er, wenn nötig, auf die nächste ganze Zahl gerundet und das Programm läuft dann ab der Zeile. Existiert diese Zeile nicht, läuft das Programm ab der nächsten Zeile im Programm.
Beachten Sie, daß RUN vor dem Anfahren des Programms auch Clear ausführt, so daß variable Werte gelöscht werden. Um das zu vermeiden, benutzen Sie GOTO gefolgt von einer Zeilennummer.
Wenn ein Programm durch LINE gesichert wurde, läuft es automatisch ab, wenn es geladen ist, und RUN wird nicht gebraucht.

Format
RUN [int-num-expr]

SAVE

Taste
S

Befehl/Statement

SAVE sendet ein Programm an den Kassettenspieler, um es auf Band zu speichern.

Benutzung von SAVE
SAVE wird normalerweise als direkter Befehl verwendet, kann jedoch eine Anweisung in einem Programm bilden. Ihm folgt ein Dateiname, welcher ein String-Wert ist, zum Beispiel

```
SAVE "Dateiname"
```

Der Dateiname kann bis zu zehn Zeichen enthalten. Bei der Ausführung erscheint das Display

```
Start tape, then press any key
```

Beim Drücken einer Taste wird das Programm an den Kassettenspieler gesandt und am Ende erscheint die Mitteilung Ø OK,Ø,1
Beachten Sie, daß SAVE anders benutzt wird, wenn ein Microdrive angeschlossen ist. Einzelheiten in Microdrive- und Interface 1-Handbuch.

Automatisches Fahren
Wenn ein gespeichertes Programm automatisch nach dem Laden fahren soll, muß SAVE in Verbindung mit LINE benutzt werden. Dem Programmnamen folgt LINE und ein numerischer Wert, zum Beispiel

```
SAVE "Dateiname" LINE 1
```

Der Wert hinter LINE wird, wenn nötig, auf die nächste ganze Zahl gerundet und müßte dann 1 sein oder die Nummer einer Programmzeile. Das Programm wird dann genauso wie mit SAVE an das Band gesandt. Beim Laden läuft das Programm automatisch ab der Zeile mit der festgelegten Nummer oder, wenn eine derartige Zeile nicht existiert, ab der nächsten Zeile im Programm.

Format
SAVE string-expr [LINE int-num-expr]

SAVE CODE

Taste
EXTEND MODE
I

Befehl/Statement

SAVE CODE sendet einen Teil der Information im Speicher an den Kassettenspieler, um ihn auf Band zu speichern. Die Information kann dann mit LOAD CODE wieder zurück in den Speicher gestellt werden.

Benutzung von SAVE CODE
SAVE CODE kann als direkter Befehl benutzt werden oder eine Anweisung in einem Programm bilden. SAVE folgt ein Dateiname, welcher ein String-Wert ist, und dann CODE, gefolgt von zwei durch Komma getrennten numerischen Werten, zum Beispiel

```
SAVE "Bild" CODE
16384, 6912
```

Der Dateiname nach kann aus bis zu zehn Zeichen bestehen. Die zwei Werte nach CODE werden, wenn nötig auf die nächsten Integer gerundet. Der erste gibt dann die Startadresse (oben 16384) der Information im Speicher, und der zweite Wert (6912) gibt die Anzahl der zu speichernden Bytes. Die Information wird dann genauso wie ein Programm mit SAVE an das Band gesandt.

Format
SAVE string-expr **CODE**
int- num-expr, int-num-expr

SAVE DATA

Taste
S
EXTEND MODE
D

Statement/Befehl

SAVE DATA speichert ein Array auf Band. Das Array kann dann mit LOAD DATA geladen werden.

Benutzung von SAVE DATA
SAVE DATA kann zur Bildung einer

Anweisung in einem Programm verwendet werden oder als direkter Befehl. SAVE folgt ein Dateiname, dann DATA, ein Buchstabe oder ein Buchstabe mit $ und als letztes zwei *leere* Klammern, zum Beispiel

```
450 SAVE "Nummern" DATA
    n( )
750 SAVE "Namen" DATA n$( )
```

Der Dateiname des Array kann bis zu zehn Zeichen enthalten. Der Buchstabe oder der Buchstabe mit $ hinter DATA ist der Name des Arrays in dem Programm, des gespeichert werden soll auf Band. Das Array wird dann genauso an das Band gesandt wie ein Programm mit SAVE.

Format
SAVE string-expr **DATA** letter [$] ( )

SAVE SCREEN$

Taste
S
EXTEND MODE
SYMBOL SHIFT K

Befehl/Statement

SAVE SCREEN$ speichert das Bildschirmdisplay auf Band. Es kann zu einem späteren Zeitpunkt mit Hilfe von LOAD SCREEN$ in den Computer zurückgeladen werden.

Benutzung von SAVE SCREEN$
SAVE SCREEN$ wird als direkter Befehl verwendet oder zur Bildung einer Anweisung in einem Programm. SAVE folgt ein Dateiname, welcher ein String-Wert ist, und dann SCREEN$, zum Beispiel

```
SAVE "Bild" SCREEN$
```

Der Dateiname kann bis zu zehn Zeichen haben. Das Display wird dann genauso an das Band gesandt wie ein Programm mit SAVE.

Format
SAVE string-expr **SCREEN$**

SCREEN$

Taste
EXTEND MODE
SYMBOL SHIFT K

Funktion

SCREEN$ findet heraus, welches Zeichen an einer bestimmten Position auf dem Bildschirm erscheint.

Benutzung von SCREEN$
SCREEN$ folgen zwei *durch Komma getrennte und in Klammern eingeschlossene* numerische Werte, zum Beispiel

```
160 IF SCREEN$ (l,c)="X"
    THEN PRINT
"ZUSAMMENSTOSS"
```

Die Werte hinter SCREEN$ werden, wenn nötig, auf die nächsten Integer gerundet. Der erste Wert (oben 1) kann dann von 0 bis 21 gehen und gibt die Zeilennummer einer Position auf dem Bildschirm. Der zweite Wert (oben c) kann von 0 bis 31 gehen und bestimmt die Spaltennummer der Position. SCREEN$ liefert dann das an dieser Position angezeigte Zeichen als eine String-Konstante (das Zeichen in Anführungsstrichen, wie zum Beispiel oben "X"). Wenn kein Zeichen vorhanden ist an dieser Stelle, liefert SCREEN$ einen Null-String ("").

Beachten Sie, daß SCREEN$ auch mit SAVE und LOAD verwendet werden kann, um das Bildschirmdisplay auf Kassette zu sichern oder von Kassette zu laden. Siehe SAVE SCREEN$ und LOAD SCREEN$ für weitere Einzelheiten.

Format
SCREEN$ (int-num-expr, int-num-expr)

SGN SiGN

Taste
EXTEND MODE
F

Funktion

SGN gibt an, ob eine Zahl positiv, negativ oder Null ist.

Benutzung von SGN
SGN folgt ein numerischer Wert, zum Beispiel

```
50 LET x=SGN y
```

Ein Ausdruck muß *in Klammern eingeschlossen werden.* SGN liefert dann 1, wenn der Wert des Arguments (oben y) positiv ist, –1, wenn er negativ ist, und 0, wenn er Null ist.

Format
SGN num-const
SGN num-var
SGN (num-expr)

SIN SINe

Taste
EXTEND MODE
Q

Funktion

SIN ergibt den Sinus eines Winkels.

Benutzung von SIN
SIN folgt ein numerischer Wert, zum Beispiel

```
80 LET x=SIN y
```

Ein Ausdruck muß *in Klammern eingeschlossen werden.* Der Wert hinter SIN ist der Winkel in Radianten, und SIN liefert den Sinus des Winkels. Grade können in Radianten umgewandelt werden durch Multiplikation mit PI/180.

Bitte beachten Sie, daß SIN einen positiven Wert für Winkel zwischen 0 und 180 Grad liefert und einen negativen Wert für Winkel zwischen 180 und 360 Grad.

Format
SIN num-const
SIN num-var
SIN (num-expr)

SQR SQuare Root

Taste
EXTEND MODE
H

Funktion

SQR liefert die Quadratwurzel eines Winkels

Benutzung von SQR
SQR folgt ein numerischer Wert, zum Beispiel

```
70 LET x=SQR y
```

Ein Ausdruck muß *in Klammern eingeschlossen werden.* Der Wert hinter SQR (oben y) muß größer sein als Null und SQR liefert seine Quadratwurzel

Format
SQR num-const
SQR num-var
SQR (num-expr)

STEP

Taste
SYMBOL SHIFT D

Siehe **FOR**

STOP

Taste
SYMBOL SHIFT A

Befehl/Statement

STOP stopt ein Programm an einem bestimmten Punkt. Es kann nötig sein STOP zu gebrauchen um das Haupteil des Programms zu stoppen damit die Subroutinen beschrenkt sind zu einem anderen Teil.

Benutzung von STOP
STOP wird normaler weise benutzt um ein statement in einem Programm zu formen. Es wird allein benutzt, zum Beispiel

```
650 STOP
```

Beim ablauf stopt das Programm und das Rapport

9 STOP statement

erscheint mit Linten und Statement Nummer wo das Programm haltet.

Es ist dann möglich Werte oder Variabelen zu zeigen oder zu ändern.

Einführung von CONTINUE lässt das Programm weiter ablaufen, ab dem nächsten Statement, mit neue Werte.

Format
STOP

STR$

Taste
EXTEND MODE
Y

Funktion

STR$ ändert einem Nummer in einem String.

Benutzung von STR$
STR$ folgt ein nummerischer Wert, zum Beispiel

```
90 LET a$ = STR$ x
```

Ein Ausdruck muss in klammern eingeschlossen werden. STR$ liefert den Wert seines Arguments (x oben) als ein String-konstante. Wenn x ist 65, der Statement gibt a$ den Wert "65".

Format
STR$ num-const
STR$ num-var
STR$ (num-expr)

TAB

Taste
EXTEND MODE
P

Siehe **LPRINT, PRINT**

TAN TANgent

Taste
EXTEND MODE
E

Funktion

TAN ergibt den Tangens eines Winkels.

Benutzung von TAN
TAN folgt ein nummerischer Wert, zum Beispiel

```
130 LET x = TAN y
```

Ein Ausdruck muss in klammern eingeschlossen werden. Der Wert folgend auf TAN ist der Winkel in Radianten, und TAN liefert den Tangens des Winkels. Grade können in Radianten umgewandelt werden durch Multiplikation mit PI/180.

Bitte beachten Sie dass TAN einen positiven Wert für Winkel zwischen 0 und 90 GRAD und zwischen 180 und 270 und 360 GRAD

Format
TAN num-const
TAN num-var
TAN (num-expr)

THEN

Taste
SYMBOL SHIFT G

Siehe **IF**

TO

Taste
SYMBOL SHIFT F

Funktion

TO wird in der Sinclair-BASIC-Sprache auf zwei verschiedene Arten benutzt. Es wird in Verbindung mit FOR benutzt,um eine FOR NEXT Schleife aufzustellen (Einzelheiten siehe unter FOR) und es wird auch beim String-Slicing, dem Aufteilen von Strings in kleinere Substrings durch 'Zerschneiden', verwendet.

Benutzung von TO beim String-Slicing
TO wird benutzt, um das erste und letzte Zeichen eines Substrings innerhalb eines Hauptstrings zu definieren. TO geht ein String-Wert voran, Klammer auf und wahlweise ein numerischer Wert. Ihm folgen ein weiterer wahlweiser numerischer Wert und dann Klammer zu, zum Beispiel

```
80 PRINT a$ (4 TO 7)
```

Ein String-Ausdruck muß in Klammern eingeschlossen werden. Der String-Wert (oben a$) ist der zu 'zerschneidende' String. Die zwei numerischen Werte (4 und 7) definieren die Positionen des ersten und letzten Zeichens des Substrings innerhalb des Strings. TO liefert dann den Substring (Zeichen 4 bis 7 von a$).

Der erste numerische Wert hat einen standardmäßigen Wert von 1 und der zweite einen standardmäßigen Wert, der der Position des letzten Zeichens im String gleich ist.

Format
string-const ([num-expr] **TO** [num-expr])
string-var ([num-expr] **TO** [num-expr])
(string-expr) ([num-expr] **TO** [num-expr])

USR

Taste
EXTEND MODE
L

Funktion

USR wird benutzt, um eine Maschinencode-Subroutine zu rufen, die im Speicher an eine bestimmte Adresse gestellt worden ist. Es wird außerdem benutzt, um die Daten für benutzergewählte Grafik an die reservierten Stellen oben im Speicher zu stellen.

USR und Maschinencode
Um den Maschinencode zu benutzen, folgt USR ein numerischer Wert, zum Beispiel

```
 80 PRINT USR 65000
100 RANDOMIZE USR 65000
```

Ein Ausdruck muß *in Klammern eingeschlossen werden.* Der Wert hinter USR wird auf den nächsten Integer gerundet und bildet dann die Startadresse im Speicher, an die ein Maschinencode-Unterprogramm gestellt worden ist. Jedes Statement mit USR ruft dann die Subroutine an dieser Adresse und USR liefert den Wert des Inhalts des bc Registerpaares.

USR und benutzergewählte Grafik
Um benutzergewählte Grafik zu erzeugen, wird USR in Verbindung mit POKE verwendet. Ihm folgt eine String-Konstante oder Variable, um eine Adresse für das POKE-Statement zu liefern, zum Beispiel

```
50 POKE USR "a", 255
```

Der String-Wert hinter USR kann ein einzelner Buchstabe von A bis U oder a bis u sein, es wird kein Unterschied zwischen Klein- und Großbuchstaben gemacht.

USR liefert dann die Startadresse von einem der 21 Teile des Speichers, die für benutzergewählte Grafik reserviert werden. Jeder Teil enthält acht Adressen, zu denen acht Bytes gePOKEt werden können, um ein Grafikzeichen zu schaffen.

Format
USR int-num-const
USR int-num-var
USR (int-num-expr)
USR string-const
USR string-var

VAL VALue

Taste
EXTEND MODE
J

Funktion

VAL verwandelt einen String mit einem numerischen Wert in eine Zahl um.

Benutzung von VAL
VAL folgt eine String-Konstante oder -Variable, zum Beispiel

```
70 LET x=VAL a$
```

Der Wert der String-Konstanten oder Variablen verliert seine Anführungsstriche und muß dann ein numerischer Wert sein. VAL wertet diesen aus und liefert ihn als numerische Konstante.

Beispiele
Wenn a$ den Wert "435" hat, ordnet die obige Anweisung x einen Wert von 435 zu. VAL kann jedoch auch Ausdrücke auswerten, zum Beispiel

```
10 INPUT a$,x
20 PRINT VAL a$
```

Der String-Wert, welcher a$ zugewiesen wird, sollte ein Ausdruck mit x sein, zum Beispiel "x*x". Ein numerischer Wert wird x dann zugewiesen, zum Beispiel 5. VAL nimmt dem String-Wert die doppelten Anführungsstriche, um x*x zu erhalten, und wertet ihn mit dem x zugewiesenen Wert aus – dann wird das Ergebnis 25 angezeigt.

Format
VAL string-const
VAL string-var

VAL$ VALue (string)

Taste
EXTEND MODE
SYMBOL SHIFT J

Funktion

VAL$ wertet einen String als einen Stringausdruck aus.

Benutzung von VAL$
VAL$ folgt eine String-Variable, zum Beispiel

```
130 PRINT VAL$ a$
```

Der Wert der String-Variablen

verliert seine Anführungsstriche und muß dann ein Stringausdruck sein. VAL$ wertet den Ausdruck aus und liefert den Wert als eine String-Konstante.

Beispiel
Versuchen Sie dieses Program.

```
10 INPUT a$,x$
20 PRINT VAL$ a$
```

Der a$ zugewiesene String-Wert sollte ein Ausdruck mit x$ sein, zum Beispiel "x$ +x$. x$ wird dann ein String-Wert zugewiesen, zum Beispiel "TUN". VAL$ nimmt dem Wert von a$ die Anführungsstriche, um x$ +x$ zu erhalten, und wertet es mit dem x$ zugewiesenen Wert aus, das angezeigte Ergebnis ist TUNTUN.

Format
VAL$ string-var

VERIFY

Taste
EXTEND MODE
SYMBOL SHIFT R

Befehl/Statement

VERIFY überprüft, ob ein Programm nach SAVE richtig auf einem Band gespeichert worden ist.

Benutzung von VERIFY
SAVE wird normalerweise als direkter Befehl verwendet genauso wie LOAD und ihm folgt der Programmname, zum Beispiel

VERIFY "Dateiname"

Wenn das Band angefangen wird, wird der Name jedes gefundenen Programmes angezeigt und jedes Programm auf dem Band mit demselben Namen wird mit dem Programm im Speicher verglichen. Wenn sich die zwei als identisch herausstellen, wird die Mitteilung

Ø OK, Ø:1

VERIFY CODE und VERIFY DATA
VERIFY CODE wird genauso wie LOAD CODE verwendet, um sicherzustellen, daß ein Teil des Speichers auf Band gespeichert worden ist. VERIFY DATA funktioniert genauso wie LOAD DATA, um zu überprüfen, ob ein Array auf Band gespeichert worden ist. Weitere Einzelheiten siehe LOAD CODE und LOAD DATA.

Format
VERIFY string-expr
VERIFY string-expr CODE [int-num-expr] [,int-num-expr]
VERIFY string-expr DATA letter [$] ( )

VERIFY CODE

Taste
EXTEND MODE
SYMBOL SHIFT R
EXTEND MODE
I

siehe **VERIFY**

VERIFY DATA

Taste
EXTEND MODE
SYMBOLD SHIFT R
EXTEND MODE
D

siehe **VERIFY**

ZX-SPECTRUM + BILDSCHIRMLISTE

Beendet der Spectrum+ die Ausführung von BASIC, erscheint eine Liste am Fuße des Bildschirms. Dies bedeutet, daß ein Kommando oder Programm vollständig aúsgeführt wurde, oder daß ein Fehler vorliegt. Diese Liste besteht aus einer Kodenummer oder einem Kodebuchstaben, gefolgt von einer kurzen Nachricht. Darauf erscheinen die Zahlen der Zeile und Anweisung, wo der Komputer aufhörte, zu operieren. Als Reihe Ø erscheint ein Kommando (sie nimmt nur eine Reihe in Anspruch); Anweisung 1 steht dabei am Anfang der Zeile, gefolgt von Anweisung 2, die nach der ersten Spalte oder nach THEN kommt, u.s.w. CONTINUE heißt normalerweise, daß das Programm bei der auf der Liste spezifizierten Anweisung wieder weitergeht.

Ø OK
heißt: erfolgreich abgeschlossen; oder aber Versuch, auf eine Zeile zu springen, oder auf eine Zahl, die größer ist als irgendeine im Programm angegebene. CONTINUE macht diese Liste zunichte und nimmt die Funktion bei der auf der Liste spezifiz erten Anweisung wieder auf.

1 NEXT without FOR
NEXT wurde ohne das entsprechende FOR angetroffen; eine Variabel existiert mit dem selben Namen wie die Kontrollvariabel.

2 Variable not found
hier wurde eine einfache Variabel verwendet, ohne das ihr ein Wert zugeordnet wurde oder daß vom Band ein Wert geladen wurde. Oder aber es es wurde eine Kontrollvariable mit NEXT verwendet, ohne dass es zunächst in eine FOR-Anweisung gesetzt wurde. Oder es wurde eine indizierte Variabel vor der Dimensionierung des Feldes mit DIM verwendet-oder vor der Ladung eines Feldes von einem Band.

3 Subscript wrong
Die Indizierung liegt außerhalb des Bereiches des Feldes.

4 Out of memory
es liegt nicht genug Speicherraum vor, um die Anweisung oder das Kommando zu vollenden.

5 Out of screen
Durch den INPUT wurden mehr als 23 Reihen auf dem unteren Teil des des Bildschirms produziert, oder aber eine Zeilennummer von 22 oder es wurde mehr mit dem Kommando PRINT AT verwendet.

6 Number too big
Hier hat der Komputer versucht, eine Zahl auszudrucken, die größer ist als etw 10^{38}.

7 RETURN without GOSUB
die Anzahl der RETURN-Anweisungen ist um eine größer als die Anzahl der GOSUB -Anweisungen.

8 End of file
Datenbearbeitungspunkt mit Mikroantrieb.

9 STOP statement
STOP wird zum Anhalten des Programms verwendet. CONTINUE wird verwendet, um anzuzeigen, daß es bei dem nächsten Anweisung weitergeht.

A Invalid argument
eine Funktion wurde als falsch oder mit einem Wert bezeichnet.

B Integer out of range
ein Wert wurde auf die nächste ganze Zahl aufgerundet und ist außerhalb des akzeptablen Bereichs.

C Nonsense in BASIC
Für den Inhalt des Programms gibt die Anweisung keinen Sinn.

D BREAK – CONT repeats
hier wurde BREAK gedrückt und CONTINUE wird die Anweisung wiederholen, bei der der Komputer angehalten hat.

E Out of DATA
READ hat versucht, über das Ende der letzten DATA Anweisung im Programm hinauszulesen.

F Invalid file name
SAVE wurde mit einem Namen verwendet, der mehr als zehn Buchstaben hat.

G No room for line
Für die Eingabe einer neuen Programmzeile ist nicht mehr genügend Raum vorhanden.

H STOP in INPUT
STOP wurden als Folge von INPUT eingegeben oder begann die Daten die eingegeben wurden. CONTINUE wiederholt die Anweisung des INPUT.

I FOR without NEXT
Eine FOR NEXT Programmschleife wurde nicht durchgeführt, weil die Grenz-oder STEP- Werte falsch waren. (Z.B.: FOR x=5 TO Ø ohne STEP) und außerdem wurde das entsprechende NEXT nicht gefunden.

J Invalid I/O device
Datenbearbeitungspunkt mit Mikroantrieb.

K Invalid colour
Der Wert, der für INK, PAPER, FLASH, BRIGHT, INVERSE oder OVER vorgesehen ist, oder aber die entsprechenden Kontrollzeichen sind.

L BREAK into program
BREAK-Taste war gedrückt. Die Liste spezifiziert die letzte Anweisung, die gelöscht werden soll und CONTINUE nimmt bei der nächsten Anweisung wieder auf.

M RAMTOP no good
Der Wert, der für RAMTOP spezifiziert ist, ist entweder zu groß oder zu klein.

N Statement lost
Es wurde ein Sprung zu einer Anweisung versucht, die nicht mehr existiert.

O Invalid stream
Datenbearbeitungspunkt mit Mikroantrieb.

P FN without DEF
Eine FN-Anweisung wurde ohne die entsprechende DEF FN Anweisung verwendet.

Q Parameter error
Eine FN- Anweisung enthält die falsche Anzahl von Werten, die an die Funktion übergeben werden sollen. Oder aber eine der Werte ist die falsche Art (d.h. zum Beispiel eine Folge statt einer Zahl oder umgekehrt.)

R Tape loading error
Der Lade- Misch-oder Verifizierungsvorgang ist fehlgeschlagen.

JENSEITS VON BASIC

BASIC ist eine Allzweck-Computersprache, die sich für die meisten Anwendungsbereiche gut bewährt; es ist aber nicht die einzige Computersprache, die man für den Spectrum verwenden kann. Software, die auch andere Computersprachen zur Vergügung stellt, wie zum Beispiel FORTH, micro-PROLOG, und LOGO, ist ebenfalls erhältlich. Diese Sprachen arbeiten in einer Völlig anderen Weise als BASIC und eröffnen ganz neue Möglichkeiten für Euren Komputer.

Da BASIC eine Komputer-Allzwecksprache ist, kann sie manchmal etwas mühsam bei einigen Anwendungen sein. Sie ist auch recht langsam im Vergleich zu anderen Sprachen, die Euch eine größere Flexibilität zusammen mit einer einfachen Programmierungsweise und einer schnelleren Laufgeschwindingkeit gewährleistet. Mit FORTH zum Beispiel, kann man eigene Worte definieren und sie beim Programmieren verwenden, zusammen mit den Befehlen, die der Komputer versteht, umd die er fast 10 Mal so schnell ausführt wie die entsprechenden Kommandos in BASIC. Mit micro- PROLOG wird der komputer einfache englische Sätzt verstehen. Er speichert sie in dem Speicher als Grundlage für die Kommunikation mit demjenigen, der den Komputer bedient.

Die Komputersprache LOGO wurde für erzeiherische Zwecke entwickelt. Sie bietet ganz einfache Kommandos, die auf äußerst vielfältige Art und Weise verwendet werden können. Wenn Ihr aber ein ganz schnelles Programm für Euren ZX Spectrum+ schreiben wollt, müßt Ihr wissen, wie man in Maschinensprache programmieren kann.

Maschinenkode

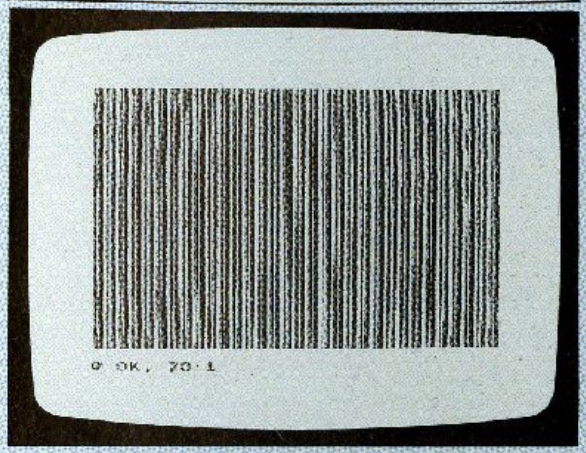
BASIC wird verwendet, damit Ihr in der Lage sein könnt, dem Komputer solche Instruktionen zu geben, die ihr leicht verstehen könnt. Die Zentraleinheit des Spektrum- das Z 80 A Chip- an sich unters versteht die Komputersprache nicht. Ein Teil des Speichers enthält aber ein ein permanentes Programm, den BASIC Interpretierer, oder des BASIC-Interpretierprogramm, das die eingegebenen Befehle in BASIC in eine- für den Komputer verständliche Reihe von Kodesignalen unwandelt. Es sind also diese Signale, die den Z80 A dazu bringen, die Befehle auszuführen, die dem Komputer eingegeben wurden.

Das Interpretierprogramm braucht einige zeit, bis es die Instruktionen in BASIC in den Z80 A-Kode oder den Maschinenkode, wie er genannt wird, umgewandelt hat. Dies kann man, wenn man will, umgehen, indem man den Maschinenkode direkt zur Zentraleinheit sendet. Dann wird das Programm rasch durchgeführt. Den Preis, den man dafür zahlen muß, ist die Zeit, die aufzuwenden ist, um das Maschinencode-Programm zu schreiben. Im Gegensatz zu BASIC dauert, es eine ganze Zeit um sie zu lernen.

Eine Programmieranleitung würde den Rahmen dieser Einführung sprengen. Es gibt aber zahlreiche Bücher, die den Spectrum-Maschinenkode bis, zum fortgeschrittenen Niveau lehren.

Sie doch mal dieses kurze Programm in ihren Komputer ein:

```
10 FOR x=0 TO 15
20 READ n: POKE 65000+x,n
30 NEXT x
40 DATA 33,255,63,1,1,24,22
50 DATA 55
60 DATA 35,11,120,177,200,114,
24,248
70 RANDOMIZE USR 65000
```



Versucht, 55 in Reihe 50 zu irgendeinem Wert von 1 bis 255 zu verändern und achtet darauf, wie sich die Steifen verändern. Gebt Ihr zuerste ein. INK Kommando ein, kann man auch farbige Streifen erhalten. Dies ist aber nicht der Sinn des Programms. Achtet mal darauf, wie schnell die Anzeige erscheint, indem der Maschinenkode verwendet wird. Sie erscheint praktisch zur gleichen Zeit! BASIC dagegen benötigt mehr als 2 Sekunden, bis der Bildschirm voll ist.

Dies funktioniert so, weil die DATA Angaben 16 Kode enthalten, die unter Adresse 65000 im Speicher zwischen Reihe 10 und 30 gespeichert sind. Reihe 70 schickt die Kodes an die zentraleinheit und sofort wird das Ergebnis auf dem Bildschirm erscheinen.

Bitte beachten Sie, daß der 8. Kode die Streifenweite kontrolliert. Viele der Komputerspiele, die für den Spektrum erhältlich sind, sind in Maschinenkode geschrieben, um die hohe Geschwindigkeit zu gewährleisten. Um beim Schreiben des Maschinenkodes behilflich zu sein, gibt es Programme mit dem Namen Assembler. Sie bieten Kommandos, die eingegeben werden, nicht Zahlen; also das, was der Maschinenkode eigentlich benötight.

Die Instruktionen sind keine Englischen Worte, wie es bei BASIC der Fall ist, sondern Abkürzungen, mnemotechnischer Kode, die die Instruktionen für die Operationen enthalten, die der Komputer unternehmen soll.

Bevor also die Assembliersprecheangewendet werden kann, muß erst ein Grundverständois vohanden sein, wie der Komputer funktioniertund zwar Schritt für Schritt.

COMPUTER-JARGON UND SEINE BEDEUTUNG

Viele der Computer-Wörter werden auch im täglichen Leben verwandt, jedoch mit anderen Bedeutungen. Hier sind einige der in diesem Buch auftretenden Wörter erklärt, zusammen mit speziellen Computerausdrücken. Die in Schrägschrift gedruckten Wörter werden an anderer Stelle selber auch erklärt. Wenn Sie ein Wort oder einen Ausdruck in dem Buch finden, den Sie nicht verstehen, schlagen Sie ihn, wenn er hier nicht aufgeführt ist, hinten im Index nach.

Adresse Eine einzelne Einheit des *memory*. Im ZX Spectrum + gibt es 65536 Adressen.

Argument Ein *Wirt*, welcher durch eine *Funktion* benutzt wird, um ein Ergebnis zu erhalten.

Array Eine Gruppe zusammengehöriger *Daten*, die zusammen in einem Teil des Speichers *(memory)* gehalten werden.

Attribute Codes, die die Farbe von *Zeichen* angeben.

Aufbereitung Einzelheiten in einem *Programm* ändern (= edit).

Auflösung Der in einer Computer-*grafik* mögliche Grad an Detail.

Ausdruck (expression) Eine Kombination von *Konstanten, Variablen* und *Schlüsselwörtern*.

BASIC Die vom ZX Spectrum + und den meisten anderen Heim-Microcomputern verwendete Computersprache.

Befehl Ein einzelne Anweisung, die vom Computer ausgeführt wird, oder ein *direkter Befehl*.

Binärcode Die von Computern benutzte Codeart. Er besteht aus Folgen von Ein- und Aus-Zuständen, zum Beispiel ein-aus elektrische Impulse.

Bit Ein Ein- oder Aus-Zustand in *Binärcode*. Kurzform für 'binary digit'.

Byte Eine Gruppe von acht Bits, die eine Zahl mit einem Wert von Ø von 255 darstellt. Jede *Adresse* in *memory* enthält ein Byte

Cursor Die Position auf dem Schirm, an der als nächstes etwas erscheint. Sie kann durch ein Blinkzeichen gekennzeichnet werden, die den *Modus* des Computers angibt.

Daten *Informationen*, die der Computer entweder vom *Programm* enthält oder die in den Computer eingespeist werden, um Ergebnisse zu erhalten.

Direkter Befehl Eine Satz einer oder mehrere Anweisungen, welcher sofort ausgeführt wird, wenn er im Computer ist.

Eingeben (=enter) Dem Computer eine vollständige Anweisung oder Information geben

Funktion Ein Vorgang, in dem der Computer einen oder mehrere *Werte* (oder *Argumente*) nimmt und sie benutzt, um ein Ergebnis zu erhalten, welches auch wieder ein Wert ist.

Grafik Das Erzeugen von Bildern wie Abbildungen, Tabellen oder Diagrammen durch den Computer.

Hardware Der Computer selbst und damit verbundene Vorrichtungen oder *Maschinen*, wie zum Beispiel *Peripheriegeräte*.

Information Wörter, Zahlen und Zeichen in beliebiger Kombination, die der Computer verarbeiten soll.

Input In den Computer gespeiste *Programme* und *Daten*.

Interface Ein Element, welches den Computer und/oder *Peripheriegeräte* verbindet und sicherstellt, daß sie miteinander kommunizieren können.

K Zur Messung der *Memory*-Kapazität eines Computers. 1K entspricht 1 Kilobyte oder 1024 Bytes. Die Speicherkapazität in K entspricht der Gesamtanzahl an *Adressen* im Speicher, vom denen jede ein Byte speichern kann. Der ZX Spectrum + hat ein 48K RAM und ein 16K ROM, das ergibt zusammen 64K.

Konstante Eine Anzahl oder Gruppe von einem oder mehreren Buchstaben oder anderen *Zeichen*.

Listing Die der Reihe nach aufgestellten *Zeilen* eines *Programmes*.

Laden Das Einspeisen eines *Programmes* oder von *Daten* aus einer Speichervorrichtung wie Disketten oder Kassette in den Computer.

Logik Der Prozeß, durch den der Computer entscheidet, ob Ergebnisse richtig oder falsch sind und Zustände wahr oder unwahr.

Maschinencode Die Sprache, die der ZX Spectrum + versteht. Programme in *BASIC* werden in den Maschinencode übersetzt durch den Computer, bevor er sie fährt.

Memory Der Teil des Computers, der *Programm* und *Daten* hält, bis sie gebraucht werden, und auch permanente Arbeitsanweisungen.

Mitteilung Eine vom Computer aufgezeigte Nachricht, die seine Vorgänge angibt (report).

Modus Im Spectrum einer von fünf Zuständen, die bestimmen, welches *Schlüsselwort* und *Zeichen* durch jede Taste auf der Tastatur erzeugt wird. Während des Programmierens wird der Modus durch einen blinkenden Buchstaben im *Cursor* angegeben.

Nesting Die Anordnung von *Schleifen* in einem *programm*, so daß eine oder mehrere *Schleifen* ineinander ausgeführt werden können.

Numerischer Wert Eine Variable, die eine Zahl hat. Numerische Variable bestehen aus einem oder mehreren Buchstaben.

Operator Eine Anweisung, die Rechenoperationen oder *Logik* durchführt.

Output Die Ergebnisse aus dem Computer.

Peripherie Jede Vorrichtung, die an den Computer angeschlossen wird.

Pixel Der kleinste Farbpunkt, der auf dem Schirm erscheinen kann. Kurzform für 'picture cell'.

Print (drucken) Entweder die Darstellung von Ergebnissen oder *Grafik* auf dem Bildschirm oder das Drucken auf einem Drucker.

Program Eine Folge von Anweisungen, die vom Computer auszuführen sind.

RAM (Random Access Memory) Der Teil des Speichers, dem ein *Programm* und *Daten* und andere sich ändernde *Werte* gegeben werden können. Wird auch 'volatile memory' genannt. RAM-Inhalt wird gelöscht, wenn Strom abgeschaltet wird. Der ZX Spectrum + hat einen 48K RAM.

Register Eine vom Hauptspeicher unabhängige kleine Speichereinheit. Register im *CPU* werden verwendet, um den Computing-Vorgang auszuführen.

ROM (Read Only Memory) Der Teil des Speichers, der permanente *Programme* und Anweisungen für den Computer anthält. Ihr Spectrum hat einen 16K ROM.

Schleife Ein Teil eines *Programmes*, der ein oder mehrere Male wiederholt wird.

Schlüsselwort (keyword) Eine Computeranweisung in *BASIC*. Braucht zum Funktionieren *Werte*.

Scroll ('rollen') Die gleichmäßige Bewegung von *Zeichen* auf einem Bildschirm, wodurch ein Display, das größer als der Bildschirm ist, betrachtet werden kann.

Sichern (save) Ein *Programm* oder *Daten* in einer Speichervorrichtung wie einer Diskette oder Kassette speichern.

Statement Entweder ein *Schlüsselwort*, das zur Bildung einer Anweisung in einer Programm*zeile* verwendet wird, oder die Anweisung selbst.

String Eine Gruppe von einem oder mehreren *Zeichen*, die in Anführungsstriche eingeschlossen sind, um sie von Zahlen und numerischen Variablen zu unterscheiden.

String-Variable Eine *Variable* mit einem *String*. String-Variablen bestehen immer aus einem einzelnen Buchstaben und dem Zeichen **$**.

Syntax Die korrekte Folge von *Schlüsselwörtern*, *Konstanten* und Ausdrücken, die eine gültige Anweisung bilden sollen.

Value Ein beliebige Anzahl oder Gruppe von *Konstanten*, *Variablen* oder *Ausdrücken*, die vom Computer verlangt oder erzeugt wird.

Variable Eine oder mehrere Einheiten des *Memory*, die eine bestimmte *Konstante* haben zur Verwendung durch den Komputer. Jede hat einen Namen oder Buchstaben, um leicht identifiziert werden zu können. Der ZX Spectrum + unterscheidet zwischen *numerischen Variablen* und *String-Variablen*.

Zeichen Buchstabe, Zahl (Ø bis 0), Symbol oder Grafikeinheit, die dargestellt oder gedruckt werden kann.

Zeichensatz Der komplette Satz Zeichen und bestimmte vom Computer benutzte Steuercodes.

Zeile (Line) Eine Anweisung oder Gruppe von Anweisungen in einem *Programm*. Sie hat eine Nummer, so daß sie in einer Folge von weiteren Zeilen an der richtigen Stelle ausgeführt wird.

Zentraleinheit (Central Processing Unit – CPU) Der zentrale Teil des Computers, der die eigentliche Datenverarbeitung durchführt und die anderen Geräte steuert. Der ZX Spectrum + verwendet einen Z80 Mikroprozessor.

INDEX

AMP-Verbindungen 5;43;47
Anführungszeichen 23, 51
Anschlüsse 5
Antennenanschlüsse und -leitunge 4-5
Arithmetische Operatoren 22;22
ATTR 35
Aufbereitung 18, 21
Aufbewahren von Programmen 13, 38-40
Auflisten 8, 21
Aufrollen (scrolling) 8

Balkendiagramme 22;26
Bandkabel 46
BASIC 18;49;73
BEEP 36;18
Befehle 33, 50
Benutzergewählte Grafiken 80;32-3
Berechnungen 22-3; 22-23
Betriebsbereite Software 12-13; 13
Bewegung 34
Bilder, Entwurf 31-1
  niedrigauflösend 26-7
Bildmuster-Programm 9
Bildschirmberichte 74
BIN 33
Binärkode 44
Blinkende Kreise - Programm 9
Border-Farbee 24-5; 6
BREAK 19
BRIGHT 31
Buchstabenmodus 21;20

CAPS LOCK 21; 18
CAPS SHIFT 8;21;18
Cartridges, ROM 12, 47; 47
Chips 42-3
CIRCLE 28
Cursortaste 19

DATA 33
DELETE 10
Doppelpunkt 23,51
DRAW 28-9
DRUCKER 45, 47; 46, 47

EAR-Anschluß 37;5,13
EDIT (Aufbereitung) 18,21
  Programmzeilen 21
Eingeben von Programmen 8-9
Einstellung von Fernsehgeraten 18
ENTER 9, 10, 11, 19
EXTEND MODE 8, 21; 18
Extend Modus 21; 20

Farbe 24-5; 24-5
  Displaytasten 19
  Kode 24
  Kombinationen 25
  Kontrollkode 33
  Mischen 32
  Testen 6;24
Fehler – Korrigieren 10,21
  Bildschirmberichte 74
Fernsehen, Verbindungen 5
  Verwendbarkeit 4
Einstellung 6;6

FLASH 31
Formen, Einfüllen von 29,29
FOR NEXT 26-7, 29, 30, 31, 34
FORTH 75
Funktionen 50

Geräusche 36-7
  Niederauflösung 26,80
GOTO 23
Grafik, Bewegung und 34-5
  Farbe 24-5
  Formen ausfüllen 29;29
  Hochauflösung 26, 28-9
  Muster 30-1
  Niederauflösung 26-1
  Zeichen erzeugen 32-5
  Zufallseffekte (Random) 30
Grafikmodus 21, 30
GRAPH 21;18;26
Großbuchstaben-Modus 21;20

Hardware, Definition 12
Hochauflösend Grafiken 26, 28-9
Hüpfender Ball – Programm 35

IF THEN 29
INK-Farbe 24-5
INPUT 24-5
Input/Output-Wege 45
Interface 45, 46-7
Interpunktionszeichen 23, 51
INVERSE 31
INV VIDEO 18

Joysticks 45, 47

Kassetten, Microdrive 12, 46, 46
Kassetten 12, 44, 45
  Beschriftung 14
  Lagerung 12
  Pflege 12
  Ton von 12

Kassettengerät, wie Verstärker 37,37
  Aufbewahrung von Programmen 38-40
  Auswahl 12
  Laden von Programmen 14-16
  Lautstärkenregler 14, 15, 16
  Tonregler 14, 15, 16
  Verbindungen 5, 13, 13
  Zähler 14
Keywords 9, 18-19, 50, 52-73, 20-1
Komma 23, 51

Laden 13, 14-15; 14-16
Lagerung 44,45
Lautsprecher 43
Lautstärkenregler, Kassettengerat 12, 14, 15
LET 23
Leertaste 19
LIST 21
Listing (Auflisten) 8, 21
LOAD 14-16
Logikchips 43
LOGO 75

Maschinenkode 75
Memory 12; 42, 43, 44-8
MIC-Anschluß 37; 5, 13
Microdrive 46; 5, 46
  Kassetten 12, 45
  Laden 46
Micro-PROLOG 75
Modem 46
Modi 20-21
Mosaik-Programm 10
Multiplikationstabellenprogram 23
Musik 36-7

Namensprogramm 8
Neue Programme 11
9 V Gleichstrom-Anschluß 5, 43
NEW 11,12;18

Niedrigauflösende Grafik 26-7

Paper-Farbe 24-5
Peripheriegeräte 45, 46-7

Pixel 28
PLOT 28
Polyhedra-Programm 10
POKE 48
PRINT 22
Programme, ändern 9
  Aufbewahren 13, 38-40
  Beginn eines neuen Programms 11
  Bestätigung (Verify) 39
  Eingabe 8-9, 44
  Fahren 8-9, 44
  Fehlerkorrektur 10
  Laden 12, 13, 14-15; 14-15
  Neubeginn von 10
Programmieren 17-40
Programmzeilen, Löschen von 21
  Aufbereiten von 21
Punkt 23, 51
Pyramidenprogramm 31

Quadrate-Programm 30

Radiostörungen 4
RAM (Random Access Memory) 42, 48, 42, 45
RAM-Paket 4
RAMTOP 48
READ 33
Rechenoperatoren 22; 22
Regenbogenprogramm 26-7
REM 39
RND 26,30
ROM (Read only Memory) 12, 47; 46-7
ROM-Cartridges 12, 47, 46-7
RS 232-Schnittstelle 47, 45
Rückstellknopf 11, 12; 5

SAVE 38-9
Schachbrett-Programm 33
Schleifen 26-27, 30
Schlüsselwörter 9, 18-19, 50, 52-73, 20-1
  Auswahl der 19, 20
Schlüsselwortmodus 20;20
Schnittstellen 45, 46-7
Scrolling 8
Semikolon 23, 51
Sinclair BASIC 49-73
Software 12
  Arten von 12
Geeignete 12
Laden von 14-16; 14-16
  Startbereite 12-13; 13
Sonnenaufgang-Programm 11
Spannungsregler 43
Speicher 12; 42, 43, 44-8
Speichertabelle 48
Stars and Stripes-programm 11
Statement 22, 50
STEP 29
Sterneprogramm 28
Steuerknüppel 45, 47
Strings 22
Stomzufuhr 4, 5; 5, 43
Subroutine 30-31
Symbole, Auswahl der 20
SYMBOL SHIFT 8, 21; 19
Symmetrisches Muster-Programm 30
Systemvariablen 48
Tasten 18-19; 18-19
  Bedienung 20-1; 20-1
  Bedienung 20-1; 20-1

Testatur 18-19
  Grafikzeichen 26
  Modi 20-1
Tonhöhe, Musik 36
Tonkontrolle, Kassettengerät 12, 14, 15
TRUE VIDEO 18
TV-Ausgang 42

Uncommitted Logic Array (ULA) 42
Unterprogramme 30-1

Variablen 22-3, 50
Verbindungen 5
  Kassettengerät 13
  Strom 5
Verstärkerton 37
Vorzeichen, Berechnungs- 22,50
  Auswahl 19

Zahlen 50
Zahlentaste 19
Zeichen, Bildung von 32-3
  Auswahl von 20
Zeichenblock-Programm 29
Zeichenset 52
Zeilen 8
Zentraleinheit CPU 43, 44, 48, 75; 43, 45
Zusammenstöße 34-35

Z80 Miroprozessor 43, 75; 45
ZX Schnittstelle 1 45, 46-7
ZX Robeterprogramm 27
ZX 16K RAM 4

First published 1984 by Dorling Kindersley Ltd, 9 Henrietta Street, London WC2E 8PS in association with Sinclair Research Ltd, 25 Willis Road, Cambridge

Redakteur David Burnie
Künstlerische Redakteur Peter Luff
Entwurf Debra Lee
Photographie Trevor Melton
Bildschirmaufnahmen Vincent Oliver
Leitender Herausgeber Alan Buckingham

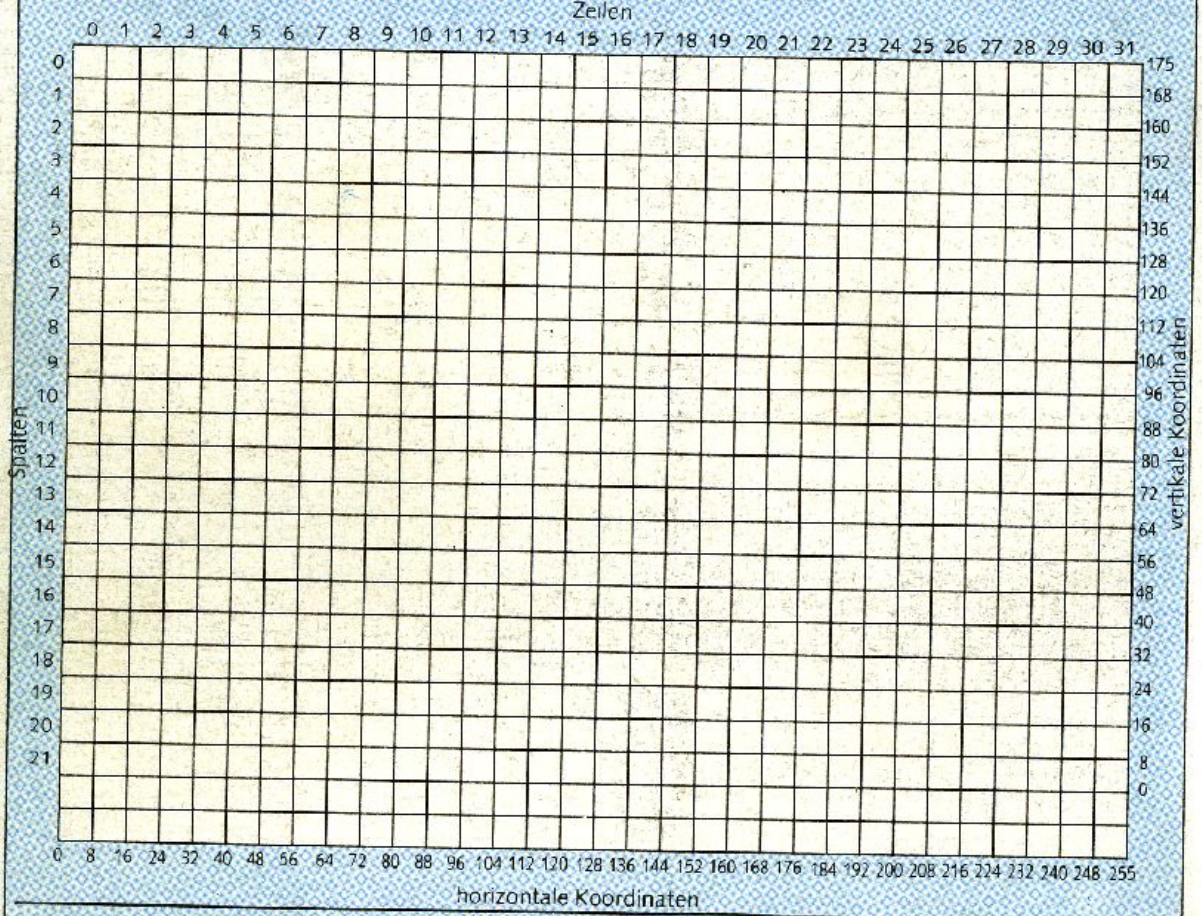
Satzherstellung: Express Typesetters Ltd, Aldershot, England.
Reproduction durch A. Mondadori, Verona.
Gedruckt und gebunden von A. Mondadori, Verona in Italien.

Bildschirmanzeigeraster

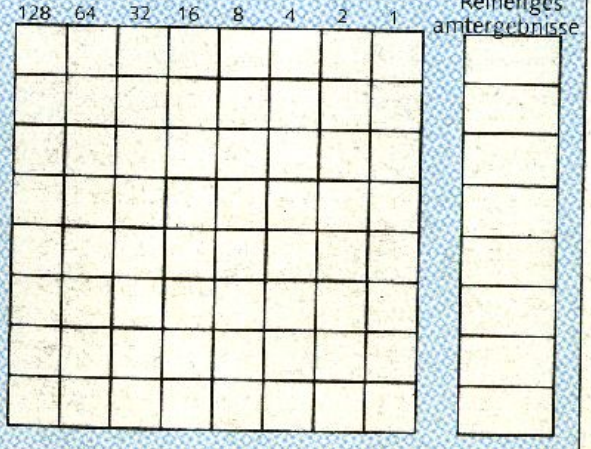
Das Koordinatennetz unten zeigt Koordinaten sowohl für Kurven mit niedriger Auflösung als auch für Kurven mit hoher Auflösung. Obwohl die einzelnen Kurvenschlüsselwörter entweder nur mit Koordinaten mit niedriger Auflösung oder mit Koordinaten mit hoher Auflösung arbeiten, können Sie beide Schlüsselwörter verwenden, wenn Sie eine Anzeige produzieren.

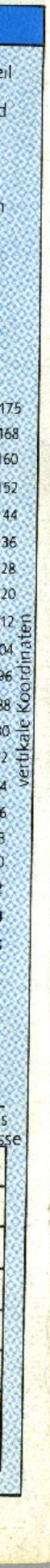
Das Niedrigauflösungs-Koordinatennetz ist in zwei Teile eingeteilt, den Hauptanzeigebereich und dann zwei Zeilen unten auf dem Bildschirm. PRINT AT wird Zeichen im oberen Teil und INPUT AT im unteren Teil produzieren. Die Koordinaten für den Niedrigauflösungsschirm werden auf der oberen und linken Seite des Koordinatennetzes gezeigt.

Das Hochauflösungs-Koordinatennetz nimmt nur den Hauptanzeigebreich ein. PLOT, DRAW und CIRCLE werden verwendet, um Hochauflösungskurven zu erstellen. Die Koordinaten für Hochauflösung werden auf der rechten und unteren Seite des Koordinatennetzes angezeigt.



Benutzer-bestimmtes Kurvennetz
Addieren Sie die gegebenen Zahlen, um den Dezimalwert für jede Reihe des Koordinatennetzes zu erhalten; verwenden Sie dann POKE USR, um das Zeichen zu programmieren.




vertikale Koordinaten

SPECTRUM SOFTWARE

Das gesamte Software-Sortiment, das für Spectrum Computer erhältlich ist (einschließlich aller bestehenden Titel), ist absolut kompatibel mit Ihrem neuen ZX Spectrum +.

DORLING KINDERSLEY LTD.
in Zusammenarbeit mit
SINCLAIR RESEARCH LTD.

ZX Spectrum+ Handbuch für Benutzer