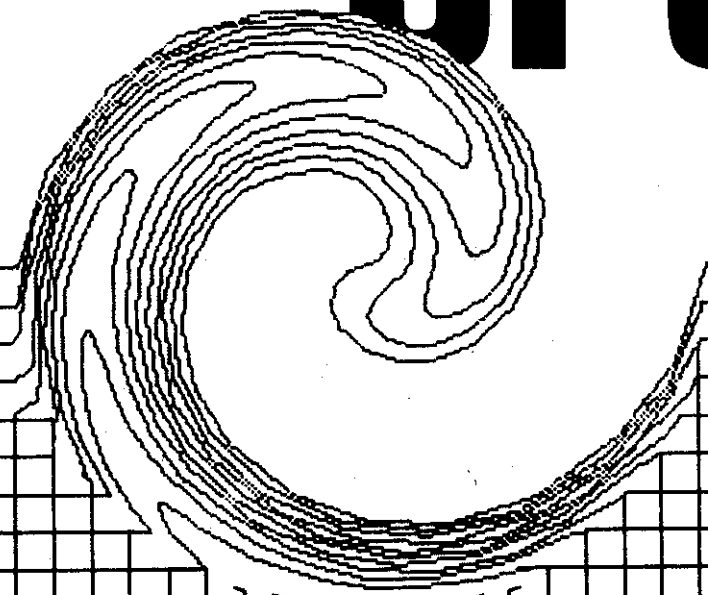
Inclusive GSX-Teil für CPC 6128/JOYCE

Steigers
Vervost

Das große GRAFIKBUCH zum CPC



ISBN 3-89011-207-2

Wichtiger Hinweis:
Die in diesem Buch wiedergegebenen Schaltungen, Verfahren und Programme werden ohne Rücksicht auf die Patentlage mitgeteilt. Sie sind ausschließlich für Amateur- und Lehrzwecke bestimmt und dürfen nicht gewerblich genutzt werden.

Alle Schaltungen, technischen Angaben und Programme in diesem Buch wurden von dem Autoren mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. DATA BECKER sieht sich deshalb gezwungen, darauf hinzuweisen, daß weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernommen werden kann. Für die Mitteilung eventueller Fehler ist der Autor jederzeit dankbar.

Es ist unglaublich,
wie viel Kraft die Seele dem Körper zu verleihen vermag.

Wilhelm von Humboldt

Vorwort

Als wir Anfang des Jahres 1986 mit dem DATA BECKER Verlag in Verhandlungen über ein Buchprojekt traten, lieferte uns der zuständige Lektor, wohl ohne es zu ahnen, das entscheidende Argument, dieses Buch und kein anderes zu schreiben - den Schneider-CPC. Wir beide hatten uns, seitdem die ersten Exemplare dieses Rechners erhältlich waren, intensiv mit ihm auseinandergesetzt. Der eine, Jürgen Steigers, indem er als einer der Autoren des CPC-Intern tief in die Bytes des Betriebssystems vordrang und der andere, Thomas A. Vervost, als absoluter Kenner schneller Grafikalgorithmen, die er für den bekannten ProfiPainter CPC entwickelte. Auf dieser Grundlage sagten wir zu, das große Grafikbuch zum CPC zu schreiben.

Nicht nur, weil unsere Freunde von uns sagen, wir seien hoffnungslose Pedanten, setzte sich schnell der Anspruch fest, ein Standardwerk in Sachen Grafik auf dem Schneider-CPC zu erschaffen. Mit diesem selbstgesetzten Ziel vor Augen entstand in Monaten intensiver Tag- und zermürbender Nachtarbeit das Buch, in dem Sie jetzt zu lesen begonnen haben.

Dieses Buch wird Ihnen nicht nur einen Einblick in die Grafikprogrammierung geben, sondern dieses Thema erschöpfend behandeln. Es liegt ein weiter Weg vor Ihnen, wenn Sie noch nie Grafiken programmiert haben, aber auch wenn Sie über Vorkenntnisse verfügen, werden Ihnen viele Dinge neu und unbekannt sein. Doch nur Mut - nach unserer Erfahrung kann jeder, der Willens ist, in die tiefen Geheimnisse der Grafikprogrammierung vorstoßen.

Damit auch Sie, während Sie dieses Buch lesen, Schritt für Schritt jene professionellen Techniken erwerben, die uns bei der Entwicklung unserer Projekte weiterhalfen, haben wir die Materie Grafik so für Sie aufbereitet, daß sie ohne unüberwindbare Schwierigkeiten nachvollzogen werden kann. Wie Sie die Grafikfähigkeiten Ihres CPC voll ausschöpfen können, werden wir Ihnen zeigen. Bestimmt.

Jürgen Steigers *Thomas A. Vervost*

Inhaltverzeichnis

Hinweis: Aus drucktechnischen Gründen wird in diesem Buch der Hochpfeil (CHR$(94)) durch das Zeichen "^" ersetzt.

1. Einleitung

Die Evolution der Computergrafik

Rasterorientierte Computergrafik ist heute etwas selbstverständliches. Sie wird beispielsweise in Kinofilmen, für Werbespots, von Firmen jeglicher Art oder auch vom Computer-Hobbyisten immer häufiger eingesetzt. Viele Menschen, die heute einen grafikfähigen Computer kaufen, können sich aus diesem Grund kaum vorstellen, daß die Computergrafik noch vor wenigen Jahren eine Domäne der Großrechner war. Um ihnen die Zusammenhänge der nahezu exponentiellen Entwicklung begreiflich zu machen, folgt an dieser Stelle ein kleiner Streifzug durch die Evolution der Computergrafik.

1966 hatten die besten verfügbaren Computer eine Speicherkapazität von etwa 16 KByte. Beim Vergleich mit einem CPC 464 fällt auf, daß dieser uns heute den vierfachen Arbeitsspeicher - nämlich 64 KByte - zur Verfügung stellt. Davon fallen alleine 16 KByte auf den für Grafiken verantwortlichen Bildschirmspeicher. Damals konnte aus Speicherplatzgründen natürlich nur sehr kleine Bereiche des Arbeitsspeichers als Bildschirmspeicher reserviert werden. Da die Größe des Bildschirmspeichers aber für die Feinheit der Computergrafiken maßgebend sind, konnte man dem zufolge nur sehr grobe Bilder zeichnen. Man entschloß sich letztendlich, auf die Bildschirmausgabe gänzlich zu verzichten und weiterhin den Drucker als Datenausgabegerät zu verwenden.

Für grafische Anwendungen ist das sogenannte *Vektorgrafiksystem* entwickelt worden. Dabei wurden direkt mit dem Kathodenstrahl Linien auf eine stark mit Phosphor beschichtete Bildröhre gezeichnet. Im Computer selbst entstanden nur sehr wenige Informationen, die sich nur auf die Steuerung des Elektronenstrahls beschränkten. Da der Bildschirm in Folge seiner starken Phosphorbeschichtung bis zu einer Stunde nachleuchtete, konnte der Rechner in dieser Zeit für andere Zwecke verwendet werden.

Diese Methode erwies sich jedoch nach kurzer Zeit als extrem uneffektiv, da beim vorzeitigen Löschen eines Bildes ein unangenehmer Blitz aufzuckte, und so keine animierte (bewegte) Grafik mit fließenden Bewegungen geschaffen werden konnte. Dieses Problem ließ die damaligen Computergrafik-Experten nicht ruhen und so wurden bald sogenannte Refresh-Bildröhren verwendet. Sie erneuern das Bild 50mal in der Sekunde und schaffen somit die Grundvoraussetzung für eine flimmerfreie Anzeige. Das Bild ist jedoch nur flackerfrei, wenn die Vektoren (Linien) eine bestimmte Länge nicht überschreiten, da sonst unweigerlich Interferenzen auftreten. Diese Synchronisationsprobleme entstehen, weil sich das Intervall zwischen zwei Halbbildern (Bildschirm-Aufbauphasen) verlängert. Aber auch dieses System hatte Nachteile. Beispielsweise war es unmöglich, mittels Vektorgrafik gefüllte Flächen auf dem Bildschirm auszugeben.

> Die Vektorgrafik übrigens hat sich bis in die heutige Zeit durchgesetzt, da sie sich hervorragend zur Darstellung dreidimensionaler Objekte eignet. Aus diesem Grund ist ihr auch in diesem Buch ein ganzes Kapitel gewidmet.

Einige Jahre später hatte sich der Datenmonitor in Folge der immer größer werdenden Speicherkapazität durchgesetzt. Damit war auch die erste rasterorientierte - wenn auch sehr schwach auflösende - Computergrafik geboren. Mit Hilfe der Bildschirmzeichen konnten Wissenschaftler erstmals einfache Grafiken wie Balkendiagramme oder das grobe Abbild einer Sinuskurve auf einem Datensichtgerät ausgeben und ohne großen Programmieraufwand in eine beliebig andere Stuktur umwandeln. Weiterhin war es problemlos möglich, Text mit Grafik zu verbinden, da letztere ja aus Textzeichen zusammengesetzt war.

Diese einfachen Grafiken befriedigten die Programmierer verständlicherweise nicht auf Dauer - Kreise oder Geraden konnten in keinster Weise auf dem Bildschirm dargestellt werden. Um diese Dinge möglich zu machen wurden Sonderzeichen wie Striche, Kreise und Dreiecke in den Zeichensatz integriert. Die damit erstellten Grafiken waren von deutlich geringerer Abstraktion und konnten bei geschickter Anwendung durchaus komplexe Dinge verbildlichen. Die PCs der ersten Generation

beinhalteten übrigens derartige Zeichensätze. Die Grafikauflösung des legendären TRS-80 (128 mal 48 Punkte) basierte beispielsweise auf 64 vordefinierten Sonderzeichen.

Aber auch diese Methode hatte Nachteile. Erstens konnte durch die mangelnde Auflösung nicht jede beliebige Form dargestellt werden, und zweitens war bei vielen Rechnern eine fließende grafische Animation kaum möglich. Um auch diese beiden Faktoren zu neutralisieren, wurde ein frei programmierbarer Zeichensatz in die Computer implementiert. Er gestattete dem Programmierer eigene, speziell auf die Anwendung zugeschnittene Zeichen zu definieren, und somit bei entsprechendem Programmieraufwand jeden einzelnen Punkt der Grafik zu manipulieren.

Heute profitieren wir - auch Sie als Benutzer des Schneider-CPC - von der rasterorientierten Computergrafik. Bei diesem Verfahren wird ein großer Teil des Arbeitsspeichers für das Speichern einer hochauflösenden Grafikseite reserviert. Jedes Bit repräsentiert dabei einen gesetzten oder gelöschten Punkt auf dem Bildschirm. Objekte jeglicher Art lassen sich nach diesem Prinzip Punkt für Punkt - wie ein Mosaik - zusammensetzen. Die so erzielte Auflösung ist dabei nur eine Frage des zur Verfügung stehenden Speicherplatzes. Soll eine farbige Grafik generiert werden, muß der Bildschirmspeicher sogar um ein Vielfaches der sonst benötigten Speicherzellen erweitert werden.

In diesem Buch werden Sie sämtliche oben vorgestellten Grafikgenerierungs-Verfahren wiederfinden, denn der Schneider-CPC hat trotz seines außergewöhnlich niedrigen Preises herausragende Grafikeigenschaften. Sie werden lernen, die verschiedenen Verfahren optimal einzusetzen und so die Kapazität Ihres Computers in Zukunft vollständig auszunutzen.

2. Punkt, Punkt, Komma, Strich...

Betrachtet man die schlichte Einschaltmeldung der CPC-Rechner, so erhält man nicht den geringsten Hinweis darauf, welche ungeahnten Grafikeigenschaften diese Rechner in ihrem Inneren bergen. So bieten sie beispielsweise die identische Grafikauflösung (640 mal 200 Punkte) eines IBM-PCs mit Color-Grafikkarte - und das serienmäßig. Aber damit nicht genug - das Auflösungvermögen ist bei den CPCs nicht auf eine bestimmte Anzahl von Punkten festgelegt sondern kann in drei verschiedenen Stufen verändert werden. Jede dieser Stufen, die auch als *Mode* bezeichnet werden, wirkt sich neben der Auflösung auch auf die Anzahl der verfügbaren Farben aus.

Da es die Eigenart aller CPCs ist, auch den Text im Grafikmodus auf dem Bildschirm darzustellen, wirken sich die drei erwähnten Modi sowohl auf die Text- als auch auf die Grafikdarstellung aus. Aus der Darstellung von Text im Grafikmodus resultiert, daß der CPC grundsätzlich Text- und Grafikelemente beliebig mischen kann.

Die Modi wirken sich im Grunde genommen auf das Auflösungsvermögen des Bildschirms und wegen der Textdarstellung in Form von Grafik auch auf die Anzahl der darstellbaren Zeichen pro Zeile aus. Da der Speicherplatz, in dem die auf dem Bildschirm dargestellten Informationen liegen beschränkt ist, ergibt sich quasi zwangsläufig, daß bei steigender Auflösung die Anzahl der darstellbaren Farben sinkt. Zwangsläufig deshalb, weil die Farbinformationen untrennbar mit den Bildinformationen verbunden und zusammen mit ihnen im Bildschirmspeicher abgelegt sind. Sie sehen schon an dieser Stelle die immense Wichtigkeit der Modi, so daß wir uns entschlossen haben, ihre Erklärung an den Anfang unserer Betrachtungen über die Grafik auf dem CPC zu stellen.

Nach dem Einschalten befindet sich der Rechner grundsätzlich in MODE 1, in dem er 40 Zeichen pro Zeile bei vier von 27 möglichen Farben darstellen kann. Um nun einen anderen Modus einzustellen, gibt man einfach das BASIC-Kommando

MODE mit einem Leerzeichen und danach die Nummer des gewünschten Modus ein. Nach dem Bestätigen der Eingabe mit der RETURN-Taste löscht der Computer den Bildschirm, aktiviert den neuen Modus und erklärt sich mit einem "Ready" in der linken oberen Ecke des Bildschirms (HOME-Position) für den Empfang des nächsten Befehls bereit.

Die nachstehende Tabelle zeigt die verschiedenen Auswirkungen aller verfügbaren MODE-Kommandos:

Mode 0:	20 Zeichen pro Zeile 160 mal 200 Punkte 16 von 27 Farben
Mode 1:	40 Zeichen pro Zeile 320 mal 200 Punkte 4 von 27 Farben
Mode 2:	80 Zeichen pro Zeile 640 mal 200 Punkte 2 von 27 Farben (monochrom)

Wann Sie welchen MODE verwenden sollen, kann nicht generell gesagt werden; wir empfehlen zum Beispiel MODE 2 mit 80-Zeichen-Auflösung für die Bearbeitung von Texten und MODE 1 für Grafiken, bei denen es darauf an kommt, daß sie verzerrungsfrei auf dem Bildschirm wiedergegeben werden. Der verbleibende MODE 0 sollte in Folge seiner niedrigen Bildschrimauflösung nur Verwendung finden, wenn möglichst viele Farben gleichzeitig dargestellt werden müssen - Anwendungsbeispiele: Analyse- und Spielprogramme.

2.1 Der erste Punkt erscheint

Die Grafik des Schneider-CPC gehört, wie Sie schon aus dem einleitenden Kapitel wissen, zur Gattung der rasterorientierten Bildschirmgrafiken. *Rasterorientiert* heißt, daß sich das Bild aus

vielen kleinen Punkten zusammensetzt, die einzeln an- und ausgeschaltet bzw. umgefärbt werden können. Diese Punkte nennt man in der Fachsprache "*Pixels*".

Die Verwaltung einer derartigen Rastergrafik ist sehr kompliziert - der Weg eines Pixels zum Bildschirm für den Laien recht unverständlich, da er massive Kenntnisse der Maschinenprogrammierung voraussetzt. Der CPC hat aber einen derart leistungsstarken BASIC-Interpreter integriert, der mit herrausragenden Grafikoptionen selbst den reinen BASIC-Programmierer die Grafikkapazität seiner Rechners zu einem großen Anteil ausschöpfen läßt. Dies ist auch ohne genaue Kenntnis der im Computer ablaufenden Vorgänge möglich.

Ein Beispiel für einen derartigen Befehl des Locomotive-BASIC ist das Kommando PLOT, mit dem ein Pixel auf dem Bildschirm gesetzt werden kann. Zur genaueren Bestimmung des PLOT-Kommandos sind vier Parameter erforderlich. Die ersten beiden Parameter legen die absolute Position fest, an der das Pixel zu setzen ist - diese Angaben sind, zur Erfüllung der Syntax des Befehls, unbedingt nötig. Die beiden folgenden Parameter sind optional. Mit ihnen kann die Farbe des zu setzenden Bildpunktes und sein Interagieren mit dem Hintergrund bestimmt werden. Die letzten Angaben sollen aber an dieser Stelle nicht von Bedeutung sein.

Syntax:

```
PLOT X-Koordinate,Y-Koordinaten[,Farbstift[,Farbmodus[1]]]
```

Betrachten wir den PLOT-Befehl einmal im Detail: Mit PLOT wird zunächst nur ein Pixel auf den Bildschirm gebracht; je nach gesetztem MODE (Auflösungsvermögen) ergibt sich aber eine unterschiedlich Ausgabe auf dem Bildschirm. Das gesetzte Pixel ist dem Augenschein nach, abhängig vom MODE, unterschiedlich groß. Untersucht man dieses Phänomen genauer, so

1 Nicht auf dem CPC 464

wird man feststellen, daß ein Pixel in MODE 0 genau so groß ist wie zwei Pixel in MODE 1 oder vier Pixel in MODE 2. Weiter wird man finden, daß sich ein und dasselbe Pixel in MODE 0 mit acht verschiedenen PLOT-Kommandos setzen läßt und analog ein Pixel in MODE 1 mit vier, in MODE 2 mit zwei. Die Erklärung ist einfacher, als es auf den ersten Blick scheinen mag.

Der CPC rechnet intern immer mit einem Auflösungsvermögen von 640 mal 400 Punkten. Die maximale arithmetische Y-Auflösung entspricht somit dem doppelten der realen Y-Auflösung. Diese Tatsache ist wichtig, um ein verzerrungsfreies Zeichnen im MODE 2 zu gewährleisten und erklärt die Adressierbarkeit eines jeden Pixels (unabhängig vom MODE) mit zwei verschiedenen Y-Koordinaten.

Da die rechnerische Auflösung in X-Richtung bei allen drei Modi mit der von MODE 2 identisch ist, muß zwangläufig ein Pixel in einem MODE mit geringerer Auflösung als der von MODE 2 größer erscheinen. Deshalb setzt sich ein Pixel, arithmetisch, in MODE 0 und MODE 1 aus mehreren Pixeln zusammen und kann damit durch mehrere unterschiedliche X-Koordinaten adressiert werden. Der CPC transferiert die eingegebenen Koordinaten automatisch in die entsprechende Auflösungsstufe und das Pixel erscheint in der ihm durch den gesetzen MODE aufgezwungenen Größe auf dem Bildschirm.

Dieses auf den ersten Blick unsinnige Verfahren hat einen entscheidenden Vorteil. Der Benutzer muß nämlich die Koordinaten bei der Auswahl eines anderen MODE nicht erst mühevoll neu berechnen. Der vollständige Zusammenhang wird noch einmal von Abbildung 1 verdeutlicht:

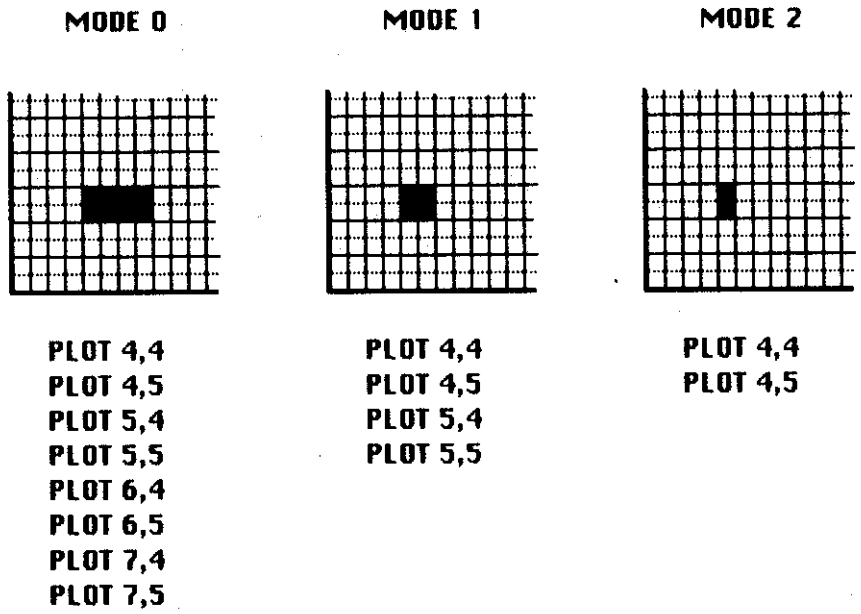


Abb. 1: Auswirkung der MODEs auf das PLOT-Kommando

Neben den bisher besprochenen Zusammenhängen hat der PLOT-Befehl noch eine weitere nicht zu unterschätzende Aufgabe. Er positioniert, neben der unmittelbar sichtbaren Ausgabe eines Pixels, durch seine X/Y Parameter einen imaginären *Grafikcursor*.

Dieser Grafikcursor wird, im Gegensatz zum herkömmlichen Textcursor, zu keinem Zeitpunkt auf dem Bildschirm dargestellt, sondern ist lediglich eine vom Computer verwaltete Positionsangabe, in der die letzten Koordinaten, an denen eine Ausgabe auf dem Grafikbildschirm erfolgt ist, aufbewahrt werden. Die mit beispielweise dem PLOT-Befehl beeinflußte Position des Grafikcursors bleibt auch nach Abarbeitung des Befehls im Rechner erhalten.

Diese Tatsache ist in vielen Fällen sinnvoll, da einige Grafikbefehle keine integrierte Koordinatendefinition zulassen, i.e. sie bestimmen ihre Koordinaten an Hand der Position des Grafikcursors. Damit derartige Befehle nicht ausschließlich von Kommandos mit Parameterdefinition abhängig sind, die Auswirkungen auf den Bildschirminhalt zeigen, befindet sich im Befehlssatz Ihres CPC eine spezielle Anweisung, die den Grafikcursor auf einen beliebigen Bildpunkt positioniert - der *MOVE-Befehl*. Seine Syntax ist abgesehen vom Schlüsselwort mit der des PLOT-Kommandos identisch.

Syntax:

```
MOVE X-Koordinate,Y-Koordinaten[,Farbstift[1][,Farbmodus[1]]]
```

Die Abbildung 2 verdeutlicht im Unterschied zur Abbildung 1 nur die Position des Grafikcursors und darf nicht insofern mißverstanden werden, daß an dieser Stelle ein Pixel gesetzt wird. Auch hier wirken sich die drei Modi in gleicher Weise wie bei PLOT aus.

Das MOVE-Kommando, und damit das Positionieren des Grafikcursors, findet insbesondere im Zusammenhang mit relativen Grafikbefehlen oder der Textdarstellung im Grafikmodus seinen Anwendungsbereich, da so der Startpunkt pixelgenau bestimmt werden kann. Relative Grafikbefehle funktionieren im Grunde genau wie ihre absoluten Abbilder, nur mit dem Unterschied, daß ihre X- und Y-Parameter einen Offset repräsentieren, der zu den Koordinaten des Grafikcursors addiert wird.

1 Nicht auf dem CPC 464

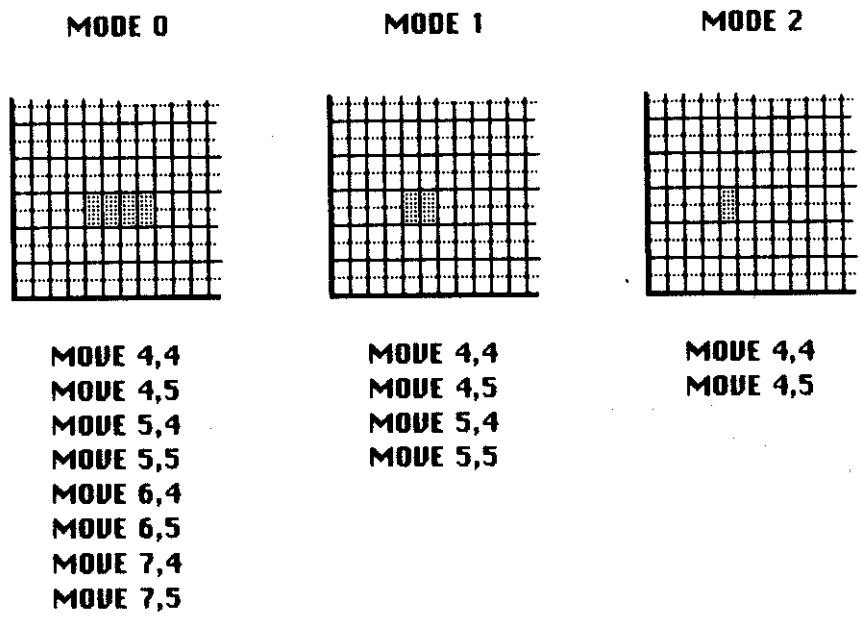


Abb. 2: Auswirkung der MODEs auf das MOVE-Kommando

2.2 Vom Punkt zur Linie

Nehmen wir doch einmal unser bis jetzt erlerntes Handwerkszeug zusammen. Wir wissen, daß im Rahmen der rasterorientierten Grafik grundsätzlich alle Bildschirmausgaben aus einzelnen Punkten zusammengesetzt werden. Ergo muß es uns möglich sein, allein mit den Kommandos PLOT und MOVE ein größeres Grafikelement zu kreieren - eine Linie.

Viele Punkte, die nebeneinander auf dem Bildschirm aufleuchten erwecken beim Betrachter den Anschein, daß es sich um einen geschlossenen Linienzug handelt. Mit einer FOR-TO-NEXT-Schleife und der PLOT-Funktion kann eine horizontale oder vertikale Linie auf dem Bildschirm erzeugt werden. Die

folgenden Programme verdeutlichen, wie aus vielen Pixels eine Linie zusammengesetzt wird.

```
10 REM ZIEHEN EINER HORIZONTALEN LINIE
20 CLS
30 FOR X=0 TO 639
40 PLOT X,200
50 NEXT
60 END
```

```
10 REM ZIEHEN EINER VERTIKALEN LINIE
20 CLS
30 FOR Y=0 TO 399
40 PLOT 320,Y
50 NEXT
60 END
```

Beim Ablauf der oben aufgeführten Programme können Sie deutlich das Verschmelzen der einzelnen Bildpunkte erkennen. Die Ausführungsgeschwindigkeit ist, da es sich um BASIC-Programme handelt, sehr gering und aus diesem Grunde wurde im Betriebssystem des Schneider-CPC eine Routine implementiert, die Linien mit vielfacher Geschwindigkeit zeichnet. Diese ist von der BASIC-Ebene aus mit dem sogenannten *DRAW-Befehl* zu erreichen.

Syntax:

```
DRAW X-Koordinate,Y-Koordinate[,Farbstift[,Farbmodus[1]]]
```

Die Startkoordinaten einer mittels DRAW-Befehl generierten Linie entsprechen der aktuellen Position des uns bereits bekannten Grafikcursors. Aus diesem Grund sind zur Spezifikation der DRAW-Funktion nur zwei Koordinaten erforderlich, die den Zielpunkt der Linie bestimmen. Auch hier gelten, wie bei den übrigen Grafik-Kommandos, die bei PLOT und MOVE angestellten Betrachtungen über die verschiedenen Grafikmodi. Exemplarisch ist der Aufbau einer Linie in Abbildung 3 für MODE 1 gezeigt.

1 Nicht auf dem CPC 464

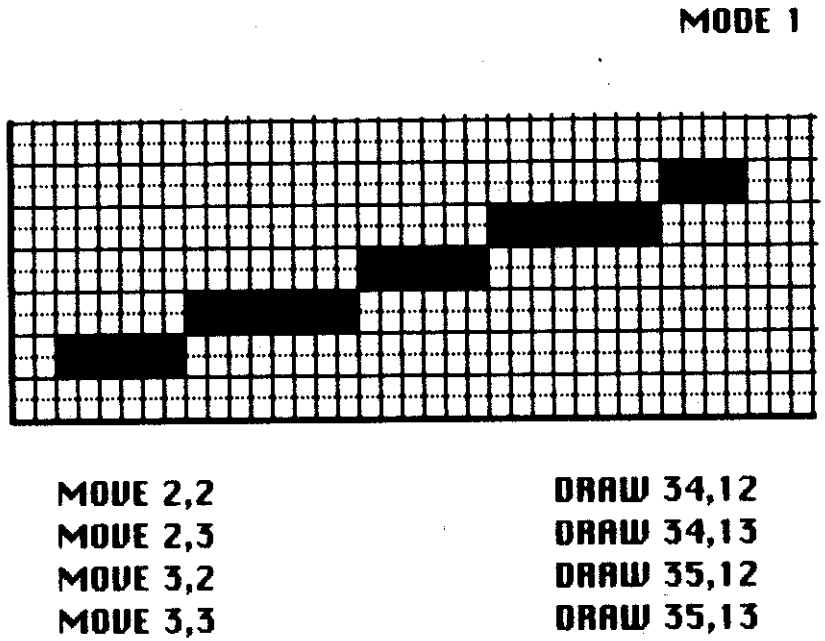


Abb. 3: Beispiel einer Linie in MODE 1

Die dargestellte Linie kann durch beliebige Kombination der in der Abbildung aufgeführten MOVE- und DRAW-Kommandos auf dem Bildschirm erzeugt werden. Die Start- und Endkoordinaten können in diesem Zusammenhang beliebig vertauscht werden, da die DRAW-Routine des Betriebssystems die Koordinaten automatisch in die korrekte Reihenfolge bringt.

2.3 Einfache geometrische Formen

Einige kommerzielle BASIC-Interpreter stellen Befehle zur Verfügung, die einfache geometrische Grundstrukturen, wie Dreiecke, Rechtecke und Kreise, auf dem Bildschirm zeichnen. Das Locomotive-BASIC beinhaltet aber keine derartigen Funktionen, was für den Benutzer bedeutet, daß er sie selbst entwerfen muß. Dabei muß er auf Kombinationen der Befehle MOVE, PLOT und DRAW zurückgreifen. Geometrische Strukturen mit spitzen Randkonturen sind relativ einfach zu erstellen, da zu Generierung derartiger Objekte keine trigonometrischen Funktionen vonnöten sind. Das folgende Programm soll eine Möglichkeit aufzeigen, ein Quadrat auf dem Bildschirm mit DRAW-Funktionen zu erstellen. In diesem Zusammenhang wird gleichzeitig noch einmal der sinvolle Einsatz des oben besprochenden Grafikcursors demonstriert, da jeder DRAW-Befehl nur zwei End-Koordinaten (X2, Y2) der zu zeichnenden Linie benötigt. Die Start-Koordinaten (X1, Y1) entsprechen der Position des Grafikcursors, die durch Angabe der End-Koordinaten jedes DRAW-Befehls automatisch aktualisiert wird.

```
10 REM ZEICHNEN EINES QUADRATS
20 MODE 1
30 CLS
40 MOVE 100,100
50 DRAW 100,200
60 DRAW 200,200
70 DRAW 200,100
80 DRAW 100,100
90 END
```

Mit der gleichen Technik, die uns zu Erstellung unseres Quadrats diente, können natürlich auch beliebige andere Strukturen wie Dreiecke oder Vielecke erstellt werden. Prinzipiell kann zur Realisation einer derartigen geometrischen Form das oben stehende Programm mit veränderter Anzahl der DRAW-Befehle dienen. Zur Generierung eines Dreiecks benötigen wir beispielsweise drei DRAW-Befehle.


```
10 REM ZEICHNEN EINES DREIECKS
20 MODE 1
30 CLS
40 MOVE 100,100
50 DRAW 150,200
60 DRAW 200,100
70 DRAW 100,100
80 END
```

Kreise und Ellipsen sind hingegen nur mit einem komplizierten Algorithmus zu berechnen, da ihre Peripherie nur mit den Produkten COS(ALPHA)*X und SIN(ALPHA)*Y berechnet werden kann, die dann mittels PLOT-Funktion auf dem Bildschirm übertragen wird. Die Generierung eines Kreises ist zudem sehr zeitaufwendig, weil für jeden zu zeichnenden Punkt ein Sinus und ein Cosinus berechnet werden muß. Das nun folgende Programm zeichnet einen Kreis mit dieser konventionellen Methode. Im Anschluß an dieses Programm werden wir eine Technik kennenlernen, die den hohen Berechnungaufwand um ein Vielfaches reduziert.

```
10 CLS
20 DEG
30 FOR A=1 TO 360
40 MOVE 320,200
50 PLOT 320+190*COS(A),200+190*SIN(A)
60 NEXT
```

Prinzipiell ist jeder Keis ein Vieleck mit unendlich vielen Eckpunkten. Diese Tatsache können wir uns insofern zunutze machen, daß wir die oben vorgestellte Berechnungsroutine durch einen Algorithmus ersetzen, der strenggenommen ein Polygon zeichnet, dessen Eckpunkte durch Linien verbunden werden können. Die Anzahl der Eckpunkte müssen dabei so gewählt werden, daß das entstehende Vieleck durch die Bildschirmauf-

lösung bedingt nicht mehr von einem Kreis unterschieden werden kann.

Um die Ausführungszeit der Kreisroutine noch weiter zu steigern, kann in der sogenannten *Viertelkreistechnik* gearbeitet werden. Die Viertelkreistechnik basiert darauf, das nur ein Viertel des zu zeichnenden Kreises berechnet werden muß und die verbleibenden drei Viertel der Peripherie durch Spiegelung konstruiert werden.

Das folgende Programm realisiert das polygone Zeichnen in Kombination mit der gerade vorgestellten Viertelkreistechnik. Da es die Berechnungszeit einer Kreisperipherie auf ein Minimum reduziert, werden wir es künftig in all den BASIC-Programmen einsetzen, die mit einer Kreisfunktion arbeiten.

```
100 CLS
110 '
120 DEG
130 '
140 INPUT "MITTELPUNKT X-KOORDINATE";XP%
160 INPUT "            Y-KOORDINATE";YP%
180 '
190 PRINT
200 '
210 INPUT "RADIUS      X-AUSDEHNUNG";XR%
230 INPUT "            Y-AUSDEHNUNG";YR%
250 '
260 CLS
270 '
280 'ERSTE KOORDINATE
290 '
300 J%=0
310 GOSUB 560
320 X1%=X2%
330 Y1%=Y2%
340 '
350 'WEITERE KOORDINATEN
360 '
```



```
370 FOR J%=6 TO 90 STEP 6
380 GOSUB 560
390 '
400 'KREIS ZEICHNEN
410 '
420 MOVE XP%+X1%,YP%+Y1%
430 DRAW XP%+X2%,YP%+Y2%
440 MOVE XP%+X1%,YP%-Y1%+1
450 DRAW XP%+X2%,YP%-Y2%+1
460 MOVE XP%-X1%+1,YP%-Y1%+1
470 DRAW XP%-X2%+1,YP%-Y2%+1
480 MOVE XP%-X1%+1,YP%+Y1%
490 DRAW XP%-X2%+1,YP%+Y2%
500 '
510 X1%=X2%:Y1%=Y2%
520 '
530 NEXT
540 '
550 END
560 '
570 'KOORDINATEN ERMITTELN
580 '
590 X2%=INT(COS(J%)*XR%+0.5)
600 Y2%=INT(SIN(J%)*YR%+0.5)
610 '
620 RETURN
```

Programmbeschreibung:

100-260 Initialisierung
In diesem Programmteil wird der Bildschirm gelöscht, das Gradmaß, das für Sinus und Cosinus notwendig ist, eingestellt und in Zeilen 140 und 150 die Koordinaten des Kreismittelpunkts mittels INPUT-Anweisung in die Variablen XP% und YP% eingelesen. Weiterhin wird in den Zeilen 210 und 230 die X- bzw. Y-Länge des Radius mit INPUT-Anweisungen in die Variablen XR% und YR% übertragen. Da mit dieser Kreisroutine auch

Ellipsen gezeichnet werden können, wird nun zur Eingabe von zwei Radien aufgefordert. Entspricht der Radius in X-Richtung dem in Y-Richtung, so wird als Ergebnis ein Kreis (Sonderform der Ellipse) auf dem Bildschirm erscheinen.

270-550 Da beim Zeichnen des Kreises nicht mit dem BASIC-Kommando PLOT gearbeitet wird, sondern das Kommando DRAW Anwendung findet, muß jeweils ein Start- und ein Endpunkt für die Linie errechnet werden. Der Startpunkt für die erste Linie (0 Grad) wird im Programmteil ERSTE KOORDINATE ermittelt, wobei die Koordinaten sich schließlich in den Variablen X1% und Y1% befinden. In der FOR-TO-NEXT-Schleife von Zeile 370 bis 530 werden dann in 6-Grad-Schritten alle weiteren Koordinaten (Von 6 bis 90 Grad) erzeugt. Zu den Koordinaten sei noch angemerkt, daß sich ausnahmslos alle im ersten Viertel des Kreises befinden. Die Koordinaten für die drei übrigen Viertel des Kreises werden durch Spiegelung am Kreismittelpunkt erzeugt. Wie das Zeichnen des Kreises exakt vonstatten geht, wird bei Betrachtung des Programmteils KREIS ZEICHNEN schnell deutlich. Ist eine Linie gezogen, so wird ihr Endpunkt zum Startpunkt für die nächste Linie erklärt (Zeile 510). Nachdem der Kreis vollständig gezeichnet ist, trifft das Programm auf END.

560-Ende Das Unterprogramm KOORDINATEN ERMITTELN errechnet mit dem Handwerkszeug der Geometrie die Koordinaten von Punkten, die sich auf der Peripherie des ersten Viertelkreises befinden. In J% befindet eine Gradangabe, zu der die entsprechenden Koordinaten (X2%, Y2%) ermittelt werden sollen. Nach der Ermittling eines Koordinatenpaares erfolgt der Rücksprung zum Hauptprogramm.

2.4 Farbe kommt ins Spiel

Bis jetzt war es uns nur möglich, monochrome d.h. zweifarbige Grafiken zu erstellen. Das Schneider-BASIC bietet jedoch weitreichende Möglichkeiten, etwas Farbe in ihre Kreationen zu bringen.

Wie wir bereits aus dem einleitenden Kapitel entnehmen konnten, besitzt unser Schneider-CPC 27 verschiedene Farben. Davon können wir - je nach verwendetem MODE - 2, 4 oder 16 gleichzeitig verwenden. Die 27 Grundfarben kann man mit einem Farbkasten vergleichen, der uns 27 unterschiedliche Farbtöpfe zur Verfügung stellt. Der Benutzer ist nun in Besitz von 2, 4 oder 16 Pinseln, denen er jeweils eine Farbe zuordnen kann. Dies geschieht bei Schneider-CPC mit Hilfe des *INK-Befehls*. Ink kommt aus dem Englischen und bedeutet frei übersetzt einfärben. Mit dem INK-Befehl kann also einem Pinsel mit einer bestimmten Farbe aus unserem Farbkasten eingefäbt werden. Programmtechnisch sieht das so aus, daß dem INK-Befehl zwei Parameter zugeordnet werden müssen. Ein Beispiel dafür ist:

```
INK 2,5
```

Der erste Parameter gibt dabei den verwendeten Pinsel an und kann je nach MODE einen Wert zwischen 0 und 15 haben. Der zweite Parameter repräsentiert den Code der gewünschten Grundfarbe. Sein Wert muß dabei immer im Bereich zwischen 0 und 26 liegen. Anders als bei einem richtigen Pinsel ist es aber mit dem INK-Kommando auch möglich, dem Pinsel zwei Farben zu zuordnen. Diese beiden Farben werden nicht etwa miteinander gemischt, sondern abwechselnd für den entsprechenden Pinsel ausgegeben. Um diesen zweiten Farbparameter in die Syntax des INK-Kommandos einzufügen, wird einfach nur eine weitere Zahl aus dem Bereich 0 bis 26 mit einem Komma angehangen.

Sind so alle Pinselfarben definiert, muß der Benutzer seinem Rechner nur noch mitteilen, welchen Pinsel er zum Zeichnen verwenden möchte. Dazu enthält das Locomotive-BASIC des CPC den Befehl *PEN*. Der PEN-Befehl benötigt für seine korrekte Syntax einen nachstehenden Parameter, der die Nummer des gewüschten Pinsels angibt. Der Wert dieses Parameters kann je nach aktiviertem MODE zwischen 0 und 1, 0 und 3 oder 0 und 15 liegen.

```
PEN 11
```

aktiviert beispielsweise Pinsel 11, d.h., daß von nun an die Bildschirmausgaben in der Farbe des Pinsels 11 erfolgen.

Das Schneiderhandbuch bezeichnet die Pinsel übrigens nicht als Pinsel, sondern als Farbstifte. Wir haben trotz dieser Tatsache den Begriff Pinsel in den gerade erläuterten Beispielen eingeführt, um die doch relativ komplizierten Vorgänge der Farborganisation zu verdeutlichen, denn ein Pinsel läßt sich im Gegensatz zu einem Stift umfärben. Da Schneider nun mal die elektronischen Pinsel nicht "Pinsel" sondern "*Farbstifte*" nennt, möchten wir Sie bitten, zukünftig diesen von Schneider geprägten Begriff zu verwenden.

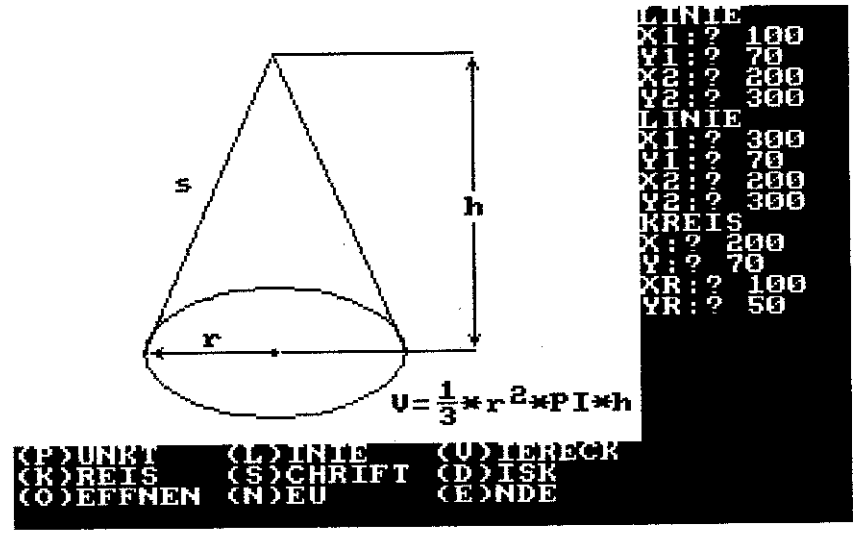
2.5 Ein kleines Zeichenprogramm

Alle bisher gelernten Verfahren, Grafiken auf dem Bildschirm zu erzeugen, haben wir zum Ende dieses Kapitel zu einem vollständigen Anwendungprogramm verschmolzen, dem "kleinen Zeichenprogramm".

Das Programm teilt den Bildschirm in drei Bereiche mit vollkommen unterschiedlichen Aufgaben. Der Bereich am linken Bildschirmrand stellt das Zeichenblatt dar, auf dem Sie Ihre Zeichnungen erstellen. Unter dem Zeichenblatt liegt ein zweiter Bereich, in dem die Optionen und ihre Kurzanwahlen aufgelistet sind. Will man zum Beispiel einen Punkt auf das Zeichenblatt

bringen, so sucht man aus der Liste der Optionen die Option "(P)unkt" heraus. Tippt dann ihre Kurzanwahl (eingeklammerter Buchstabe), im Falle eines Punktes ein "P" ein. Auf dem dritten Fenster, das sich am rechten Bildschirmrand befindet, erscheint die Meldung "PUNKT". Das dritte Fenster ist ein Dialogfenster, in dem zum einen alle angewählten Optionen aufgelistet und zum anderen die Koordinaten zum Zeichnen erfragt werden.

Die Hardcopy 1 zeigt die Bildschirmaufteilung des Zeichenprogramms; deutlich sind die unterschiedlichen Funktionen des Zeichenblattes, des Optionsfensters und des Dialogfensters zu erkennen.



Hardcopy 1: **Eine typische Anwendung des kleinen Zeichenprogramms**

Zur Darstellung eines Punktes müssen zwei Koordinaten eingegeben werden, die die Position des Punktes auf dem Zeichenblatt bestimmen. Will man dann weitere Elemente des Zeichenprogramms auf das Zeichenblatt bringen, so kann man

die im Dialogfenster mitgeschriebenen Koordinaten als Ausgangspunkte verwenden.

Neben der Möglichkeit, Punkte auf das Zeichenblatt zu bringen, bietet das Zeichenprogramm folgende Optionen:

- **(L)inie**
  Zieht eine Linie von der Position X1/Y1 zur Position X2/Y2.

- **(V)iereck**
  Zeichnet ein Viereck auf das Zeichenblatt, dessen linke untere Ecke durch die Koordinaten X1/Y1 und dessen rechte obere Ecke durch die Koordinaten X2/Y2 bestimmt wird.

- **(K)reis**
  Zeichnet einen Kreis oder eine Ellipse auf das Zeichenblatt. Der Mittelpunkt dieser geometrischen Form wird durch die Koordinaten X1 und Y1 bestimmt. Mit den Angaben XR und YR kann der Radius in X- und Y-Ausdehnung festgelegt werden. Werden für XR und YR gleich große Werte eingegeben, so entsteht ein Kreis (Sonderform der Ellipse) auf dem Zeichenblatt.

- **(S)chrift**
  Mit der Option "Schrift" können Zeichen auf dem Bildschirm ausgegeben werden. Die Eingaben X und Y bestimmen die Position der linken oberen Ecke der ersten Zeichenmatrix des auszugebenden Textes.

- **(D)isk**
  Die Option "Disk" erlaubt das Sichern einer erstellten Zeichnung auf Diskette. Die Grafik wird in Form eines Birnärfiles gespeichert und kann jederzeit auch ohne das Zeichenprogramm wieder auf den Bildschirm gebracht werden (LOAD "Name.BIN").

- **(O)effnen**
  Mit dieser Option kann eine in Form eines Binärfiles auf Diskette gespeicherte Zeichnung zurück auf das Zeichenblatt gebracht werden. Nach dem Einladen einer Zeichnung sind alle Angabe des Dialogfensters gelöscht.

- **(E)nde**
  Diese Option beendet nach der Bestätigung einer Sicherheitsabfrage das Programm. Nicht abgespeicherte Zeichnungen sind nach dem Durchlaufen dieses Programmpunktes unwiederbringlich verloren.

Die genaue Handhabung des Zeichenprogramms erlernen Sie am besten, indem Sie versuchen, Zeichnungen mit ihm zu erstellen und sich die Abläufe beim Zeichnen an Hand der Programmbeschreibung verdeutlichen.

```
100 MODE 1
110 '
120 CM$="PLVKSDONE"
130 '
140 DEG
150 '
160 ORIGIN 1,64,1,479,399,64
170 '
180 BORDER 13
190 '
200 INK 0,26
210 INK 1,0
220 INK 2,13
230 INK 3,8
240 '
250 WINDOW #0,1,30,1,21
260 WINDOW #1,1,30,22,25
270 WINDOW #2,31,40,1,25
280 '
290 PAPER #0,0
300 PEN #0,1
310 CLS #0
```

```
320 '
330 PAPER #1,2
340 PEN #1,3
350 CLS #1
360 '
370 PAPER #2,2
380 PEN #2,0
390 CLS #2
400 '
410 'OPTIONEN AUSGEBEN
420 '
430 PRINT #1,"(P)UNKT   (L)INIE   (V)IERECK (K)REIS   (S)CHRIFT (D
)ISK    (O)EFFNEN (N)EU     (E)NDE"
440 '
450 'IN UNTERPROGRAMME VERZWEIGEN
460 '
470 IUV$=INKEY$
480 IUV$=UPPER$(IUV$)
490 '
500 FOR IX=1 TO 9
510 IF IUV$<>MID$(CM$,IX,1) THEN NEXT:GOTO 440
520 ON IX GOSUB 550,650,780,940,1420,1580,1670,1770,1890
530 '
540 GOTO 440
550 '
560 'PUNKT SETZEN
570 '
580 PRINT #2,"PUNKT"
590 INPUT #2,"X:";XP
600 INPUT #2,"Y:";YP
610 '
620 PLOT XP,YP
630 '
640 RETURN
650 '
660 'LINIE ZEICHNEN
670 '
680 PRINT #2,"LINIE"
690 INPUT #2,"X1:";XP1
700 INPUT #2,"Y1:";YP1
```



```
710 INPUT #2,"X2:";XP2
720 INPUT #2,"Y2:";YP2
730 '
740 MOVE XP1,YP1
750 DRAW XP2,YP2
760 '
770 RETURN
780 '
790 'VIERECK ZEICHNEN
800 '
810 PRINT #2,"VIERECK"
820 INPUT #2,"X1:";XP1
830 INPUT #2,"Y1:";YP1
840 INPUT #2,"X2:";XP2
850 INPUT #2,"Y2:";YP2
860 '
870 MOVE XP1,YP1
880 DRAW XP1,YP2
890 DRAW XP2,YP2
900 DRAW XP2,YP1
910 DRAW XP1,YP1
920 '
930 RETURN
940 '
950 'KREIS ZEICHNEN
960 '
970 PRINT #2,"KREIS"
980 INPUT #2,"X:";XP
990 INPUT #2,"Y:";YP
1000 XP%=INT(XP)
1010 YP%=INT(YP)
1020 INPUT #2,"XR:";XR
1030 INPUT #2,"YR:";YR
1040 XR%=INT(XR)
1050 YR%=INT(YR)
1060 '
1070 'ERSTE KOORDINATE
1080 '
1090 j%=0
1100 GOSUB 1350
```

```
1110 x1%=x2%
1120 y1%=y2%
1130 '
1140 'WEITERE KOORDINATEN
1150 '
1160 FOR j%=6 TO 90 STEP 6
1170 GOSUB 1350
1180 '
1190 'KREIS ZEICHNEN
1200 '
1210 MOVE xp%+x1%,yp%+y1%
1220 DRAW xp%+x2%,yp%+y2%
1230 MOVE xp%+x1%,yp%-y1%+1
1240 DRAW xp%+x2%,yp%-y2%+1
1250 MOVE xp%-x1%+1,yp%-y1%+1
1260 DRAW xp%-x2%+1,yp%-y2%+1
1270 MOVE xp%-x1%+1,yp%+y1%
1280 DRAW xp%-x2%+1,yp%+y2%
1290 '
1300 x1%=x2%:y1%=y2%
1310 '
1320 NEXT
1330 '
1340 RETURN
1350 '
1360 'KOORDINATEN ERMITTELN
1370 '
1380 x2%=INT(COS(j%)*xr%+0.5)
1390 y2%=INT(SIN(j%)*yr%+0.5)
1400 '
1410 RETURN
1420 '
1430 'SCHRIFT AUSGEBEN
1440 '
1450 PRINT #2,"SCHRIFT"
1460 INPUT #2,"X:";XP
1470 INPUT #2,"Y:";YP
1480 INPUT #2,"TEXT:";TE$
1490 '
1500 TAG
```



```
1510 '
1520 MOVE XP,YP
1530 PRINT TE$;
1540 '
1550 TAGOFF
1560 '
1570 RETURN
1580 '
1590 'ARBEITSBLATT SICHERN
1600 '
1610 PRINT #2,"DISK"
1620 INPUT #2,"NAME:";NM$
1630 CLS #2
1640 SAVE NM$,B,&C000,&4000
1650 '
1660 RETURN
1670 '
1680 'ARBEITSBLATT LADEN
1690 '
1700 PRINT #2,"OEFFNEN"
1710 INPUT #2,"NAME:";NM$
1720 IF UPPER$(RIGHT$(NM$,4))<>".BIN" THEN NM$=NM$+".BIN"
1730 CLS #2
1740 LOAD NM$
1750 '
1760 RETURN
1770 '
1780 'ARBEITSBLATT LOESCHEN
1790 '
1800 PRINT #2,"NEU"
1810 PRINT #2,"SICHER? ";
1820 S$=INKEY$
1830 IF S$="" THEN 1820
1840 IF S$<>"J" AND S$<>"j" THEN PRINT #2,"N":RETURN
1850 CLS #0
1860 CLS #2
1870 '
1880 RETURN
1890 '
1900 'PROGRAMM BEENDEN
```

```
1910 '
1920 CALL &BB4E
1930 CALL &BBFF
1940 '
1950 END
```

Programmbeschreibung

100-390 Definitionen und Dimensionierungen. Der Stringvariablen CM$ wird eine Zeichenfolge zugewiesen, die die Anfangsbuchstaben der Optionen des Zeichenprogramms enthält. Mit Hilfe von CM$ wird im Programmteil IN UNTERPROGRAMME VERZWEIGEN entschieden, ob die betätigte Taste zum Aufruf eines Unterprogramms führt oder nicht.

Nach dem Versetzen des Computers ins Winkelgradmaß, der Bestimmung der Rahmenfarbe (BORDER) und der Zuweisung der Farben für die Farbstifte 0 bis 3 werden schließlich drei Windows definiert, die im einzelnen folgende Bedeutungen haben:

#0: Arbeitsblatt, auf dem die Zeichnung erstellt wird. Die Grenzen dieses Fensters stimmen mit der Begrenzung des Grafik-Bildschirmbereiches überein, der in Zeile 160 definiert wurde.

#1: Auf diesem Fenster werden einmal die Optionen des Programms ausgegeben (Zeile 430) und bleiben im Verlauf des Programms dort erhalten.

#2: Dialogfenster - Über dieses Fenster wird der Dialog zwischen Rechner und Benutzer geführt. Der Rechner zeigt hier beispielsweise an, in welcher Unterroutine er sich befindet oder erfragt Koordinaten. Der Benutzer tippt dann die Angaben in die Tastatur, auf daß sie im Fenster #2 protokolliert werden.

Sind alle drei Fenster definiert, so wird ihnen eine individuelle Vorder- und Hintergrundfarbe zugeordnet und ihr Inhalt gelöscht. Sie sind dann bereit für die Ausgaben des Programms.

400-430 **Optionen ausgeben**
Der Programmteil OPTIONEN AUSGEBEN schreibt in das Fenster #1, welche Möglichkeiten das Programm bietet und über welche Taste sie zu erreichen sind. Window #1 dient ausschließlich zur Ausgabe dieser Informatinen, die während des gesamten Programmlaufs dort erhalten bleiben.

440-540 **In Unterprogramme verzweigen**
Diese Programmzeilen stellen die Hauptschleife dar. Sie bildet die Schaltzentrale des Programms, von der alle Verzweigungen in die Unterprogramme vorgenommen werden. Zu diesem Zweck wird die Eingabe der Tastatur mit den einzelnen Elementen der Variablen CM$ verglichen (Zeile 510). Kann keine Übereinstimmung festgestellt werden, so wird IN UNTERPROGRAMME VERZWEIGEN erneut durchlaufen. Stimmt die Eingabe mit einem Element von CM$ überein, so wird gemäß dem aktuellen Index IX mit ON IX GOSUB in das der Eingabe entsprechende Unterprogramm verweigt. Nach der Rückkehr aus dem angesprungenen Unterprogramm beginnt wieder die Verarbeitung der Hauptschleife.

550-640 **Punkt setzen**
Durch einen Druck auf die Taste "P" hat die Hauptschleife in dieses Unterprogramm verzweigt. Auf dem Dialogfenster (#2) wird vermerkt, daß es sich um das Unterprogramm "PUNKT" SETZEN

handelt, und die Eingabe der X- und Y-Koordinate des Punktes erwartet. Die einzugebenden Werte verstehen sich relativ zum gesetzten Koordinatenursprung (ORIGIN). Mit den eingegebenen Koordinaten wird der Punkt auf dem Arbeitsblatt (#0) gezeichnet - Rücksprung zur Hauptschleife.

650-770 **Linie zeichnen**
Wurde die Taste "L" betätigt, so verzweigt die Hauptschleife in dieses Unterprogramm. Auf dem Dialogfenster (#2) wird vermerkt, daß es sich um das Unterprogramm "LINIE" ZEICHNEN handelt. Im Dialogfenster wird dann die Eingabe der Koordinaten des Start- (X1, Y1) und Endpunktes (X2, Y2) erwartet. Gemäß den eingegebenen Koordinaten (relativ zum gesetzen Ursprung) wird der Grafikcursor - auf dem Arbeitsblatt - an den Startpunkt der Linie bewegt und die Linie zum gewählten Endpunkt gezeichnet. Rücksprung zur Hauptschleife.

780-930 **Viereck zeichnen**
Durch einen Druck auf die Taste "V" hat die Hauptschleife in dieses Unterprogramm verzweigt. Nach der Ausgabe "VIERECK" wird im Dialogfenster die Eingabe von vier Werten verlangt. Die ersten beiden Werte (X1, Y1) stellen den linken unteren Eckpunkt des Rechtecks dar; das zweite Wertepaar (X2, Y2) steht für die rechte obere Ecke des Rechtecks. Auch diese Angaben verstehen sich wieder relativ zum gesetzten Koordinatenursprung. Das Viereck wird entsprechend der Eingaben auf dem Arbeitsblatt gezeichnet (Zeile 870-910).

940-1410 **Kreis zeichnen**
Wurde die Taste "K" betätigt, so verzweigt die Hauptschleife zur Zeile 940. Auf dem Dialogfenster erscheint der Hinweis, daß zur Zeit das Unterprogramm "KREIS" ZEICHNEN abgearbeitet wird. Es wird dann die X- und Y-Koordinate des

Kreismittelpunktes erfragt und unmittelbar in eine Integervariable überstellt (größtmögliche Verarbeitungsgeschwindigkeit). Da mit dieser Kreisroutine auch Ellipsen gezeichnet werden können, wird nun zur Eingabe von zwei Radien aufgefordert. Entspricht der Radius in X-Richtung dem in Y-Richtung, so wird als Ergebnis ein Kreis (Sonderform der Ellipse) auf dem Bildschirm erscheinen. Auch die beiden Radien werden in Integervariablen gespeichert (XR%, YR%).

Da beim Zeichen des Kreises nicht mit dem BASIC-Kommando PLOT gearbeitet wird, sondern das Kommando DRAW Anwendung findet, muß jeweils ein Start- und ein Endpunkt für die Linie errechnet werden. Der Startpunkt für die erste Linie (0 Grad) wird im Programmteil ERSTE KOORDINATE ermittelt, wobei die Koordinaten sich schließlich in den Variablen X1% und Y1% befinden. In der FOR-TO-NEXT-Schleife von Zeile 1160 bis 1320 werden dann in 6-Grad-Schritten alle weiteren Koordinaten (von 6 bis 90 Grad) erzeugt. Zu den Koordinaten sei noch angemerkt, daß sich ausnahmslos alle im ersten Viertel des Kreises befinden. Die Koordinaten für die drei übrigen Viertel des Kreises werden durch Spiegelung am Kreismittelpunkt erzeugt. Wie das Zeichnen des Kreises exakt vonstatten geht, wird bei Betrachtung des Programmteils KREIS ZEICHNEN schnell deutlich. Ist eine Linie gezogen, so wird ihr Endpunkt zum Startpunkt für die nächste Linie erklärt (Zeile 1300). Nachdem der Kreis vollständig gezeichnet ist, erfolgt der Rücksprung zur Hauptschleife.

Das Unterprogramm KOORDINATEN ERMITTELN wird nicht von der Hauptschleife verwaltet, sondern ausschließlich von KREIS ZEICHNEN verwendet. In diesem Unterprogramm können mit dem Handwerkszeug der Geometrie die Koordina-

ten von Punkten, die sich auf der Peripherie des ersten Viertelkreises befinden, errechnet werden. In J% befindet eine Gradangabe, zu der die entsprechenden Koordinaten (X2%, Y2%) ermittelt werden sollen.

1420-1570 **Schrift ausgeben**
Wurde die Taste "S" betätigt, so verzweigt die Hauptschleife an diese Stelle. Auf dem Dialogfenster erscheint der Hinweis, daß es sich um das Unterprogramm "SCHRIFT" AUSGEBEN handelt. Dann wird die Position des Grafikcursors erfragt, ab der die Textausgabe auf dem Arbeitsblatt erfolgen soll. Auf die Aufforderung "TEXT:" muß der betreffende Text eingetippt werden, der auf dem Arbeitsblatt erscheinen soll. In den Zeilen 1500 bis 1550 wird die Textausgabe an die Position des Grafikcursor angekoppelt (TAG), der Grafikcursor an den zuvor eingegebenen Punkt bewegt, der Text (TE$) auf dem Arbeitsblatt ausgegeben und die Textausgabe wieder an den Textcursor gebunden (TAGOFF). Rücksprung zur Hauptschleife.

1580-1660 **Arbeitsblatt sichern**
Damit eine einmal erstellte Zeichnung mit dem Ausschalten des Computers nicht unwiderruflich verloren ist, wurde die Option ARBEITSBLATT SICHERN implementiert. Sie ist durch Druck auf die Taste "D" zu erreichen. Im Dialogfenster erscheint der Hinweis "DISK", und es wird die Eingabe eines Dateinnamen erwartet, unter dem der momentane Bildschirm auf Diskette gerettet werden soll. Vor dem Abspeichern des Bildschirms wird das Dialogfenster gelöscht (CLS #2), um Komplikationen bei der Option ARBEITSBLATT LADEN von Anfang an auszuschließen. In Zeile 1640 wird dann der gesamte Bildschirmspeicher in Form eines Binärfiles auf Diskette übertragen.

1670-1760 **Arbeitsblatt laden**
Um eine mit der Option DISK auf Diskette übertragene Zeichnung wieder bearbeiten zu können, muß sie mit dem Unterprogramm ARBEITSBLATT LADEN zurück in den Bildschirmspeicher geholt werden. Der Druck auf die Taste "O" für OEFFNEN hat die Hauptschleife veranlaßt an diese Stelle des Programms zu verzweigen. Im Dialogfenster (#2) erscheint der Hinweis, daß es sich um das Unterprogramm OEFFNEN handelt. Dem eingegebenen Dateinamen NM$ wird, falls noch nicht vorhanden, die Extension ".BIN" für Binärfile angehanden. Nach den Löschen des Dialogfensters wird die ausgewählt Datei in den Bildschirmspeicher geladen und kann weiter bearbeitet werden.

1770-1880 **Arbeitsblatt loeschen**
Bei einem Druck auf die Taste "N" verzweigt die Hauptschleife in die Subroutine ARBEITSBLATT LÖSCHEN. Auf dem Dialogfenster erscheint der Hinweis "NEU" und das Programm vergewissert sich mit der Frage "SICHER?", ob die Taste "N" nicht versehentlich gedrückt wurde. Wird die Frage "SICHER?" mit etwas anderem als "J" beantwortet, so erscheint als Antwort ein "N" im Dialogfenster, und das Unterprogramm trifft auf RETURN. Wurde "SICHER?" bestätigt, werden Arbeitsblatt und Dialogfenster gelöscht.

1890-Ende **Programm beenden**
Durch Sprung auf die Betriebssystem-Vektoren &BB4E TXT INITIALISE (vollständige Initialisierung des Text-Packs) und &BBFF SCR INITIALISE (vollständige Initialisierung des Screen-Packs) wird die Bildschirmausgabe wieder auf ihren Defaultwert gebracht, d.h. umdefinierte Farben und definierte Fenster werden zurückgesetzt.

3. Der Zeichensatz - eine Fundgrube für Grafikelemente

Eines der wichtigsten Betriebsmittel bei jedem Computer ist ganz ohne Zweifel der Zeichensatz. Auf seiner Grundlage wird die gesamte Kommunikation abgewickelt - damit ist neben der Kommunikation Benutzer-Rechner auch der Austausch von Daten zwischen zwei Rechnersystemen gemeint.

Jede Form der Kommunikation unterliegt seit jeher Bestrebungen der Vereinheitlichung. Ohne Normierung wäre nicht gewährleistet, daß zwei Partner, die miteinander kommunizieren wollen, sich letztendlich auch verstehen. Voraussetzung für das Gelingen der Kommunikation ist somit, daß alle Teilnehmer den gleichen Code, also ein System festgelegter Zeichen, benutzen, um sich verständlich zu machen. Es kann damit die gleiche Sprache gemeint sein, die zwei Menschen sprechen oder die gleiche Zeichennorm, die zwei Computer verwenden.

Um Kommunikation unterschiedlicher Rechnersysteme zu ermöglichen, wurde im Bereich der elektronischen Datenverarbeitung der sogenannte ASCII[1]-Code entwickelt, der eine Norm für den Zeichensatz darstellt. Die Verwendung dieses Standards ist den Herstellern von Datenverarbeitungsalagen zwar nicht vorgeschrieben, wird aber aus Gründen der Kompatipilität zu anderen Systemen fast ohne Ausnahmen eingehalten. Für den reinen Anwender von DV-Anlagen ist der vom Rechner verwendete Code ohne Bedeutung, doch jeder, der sich mit der Programmierung von Computern beschäftigt, kommt ohne die Kenntnis des ASCII-Codes nicht aus.

Der ASCII-Code unterzieht alle Zeichen, deren Zeichennummer im durch sieben Bit darstellbaren Bereich (0-127) liegen, einer Normierung. Bestimmt durch diesen Standard sind bei allen Rechnern, die ihm folgen - auch der CPC tut das - die gleichen Zeichen unter entsprechenden Zeichennummern zu finden.

1 American Standard Code for Information Interchange

Damit ist die Grundlage für den erfolgreichen Austausch von Daten zwischen zwei Rechnern gegeben.

In den Bereich der ersten 128 Zeichen fallen neben den großen und kleinen Buchenstaben, Zahlen und Sonderzeichen auch einige Steuerzeichen, die für die Kommunikation zweier Rechnersysteme eine Rolle spielen. Wo sich die Zeichen innerhalb des ASCII-Standards befinden, können Sie Tabelle 1 entnehmen. Dort sind ausnahmslos alle ASCII-Zeichen zusammen mit ihrer Zeichennummer in binärer, dezimaler und hexadezimaler Form wiedergegeben.

Greifen wir doch einmal ein Feld aus der ASCII-Tabelle heraus. Etwas rechts von der Mitte oben befindet sich ein Feld, in dem ein "A" und die Zahl 65 steht. Die Tabelle soll die Verbindung des Buchstabens "A" und der Zeichennummer 65 innerhalb der ASCII-Norm herstellen. Für manche Anwendungen des ASCII-Codes kann es von Bedeutung sein, neben der dezimalen Zeichennummer auch die hexadezimale oder duale Entsprechung zu kennen. Um Ihnen ein aufwendiges Umrechnen der Zahlensysteme zu ersparen, kann die Zeichennummer in den beiden anderen Zahlensystemen ebenfalls der Tabelle entnommen werden.

Wieder am Beispiel des "A" - verfolgt man die Spalte, in der das mit 65 bezeichnete Kästchen steht, nach oben bis zu dem etwas fetteren Strich, so trifft man auf die Zahl 4, die die erste Stelle der hexadezimalen Zeichennummer ist. Die zweite Stelle kann gefunden werden, in dem man die Zeile, in der die 65 steht, nach links bis zu den fetteren Strich verfolgt. Es ergibt sich also für das "A" die hexadezimale Zeichennummer 41.

In entsprechender Weise kann die Zeichennummer in binärer Darstellung aus der Tabelle entnommen werden, nur müssen dann Spalten und Zeilen bis an den äußeren Rand der Tabelle verfolgt werden. Die duale Zeichennummer besteht immer aus 7 Bit.

7-bit-ASCII
(American Standard Code for Information Interchange)

Bit-Nummer 6 5 4 \ 3 2 1 0	Hexcode	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
0 0 0 0	0	NUL 00	DLE 16	SP 32	0 48	@ 64	P 80	` 96	p 112
0 0 0 1	1	SOH 01	DC1 17	! 33	1 49	A 65	Q 81	a 97	q 113
0 0 1 0	2	STX 02	DC2 18	" 34	2 50	B 66	R 82	b 98	r 114
0 0 1 1	3	ETX 03	DC3 19	# 35	3 51	C 67	S 83	c 99	s 115
0 1 0 0	4	EOT 04	DC4 20	$ 36	4 52	D 68	T 84	d 100	t 116
0 1 0 1	5	ENQ 05	NAK 21	% 37	5 53	E 69	U 85	e 101	u 117
0 1 1 0	6	ACK 06	SYN 22	& 38	6 54	F 70	V 86	f 102	v 118
0 1 1 1	7	BEL 07	ETB 23	' 39	7 55	G 71	W 87	g 103	w 119
1 0 0 0	8	BS 08	CAN 24	(40	8 56	H 72	X 88	h 104	x 120
1 0 0 1	9	HT 09	EM 25	) 41	9 57	I 73	Y 89	i 105	y 121
1 0 1 0	A	LF 10	SUB 26	* 42	: 58	J 74	Z 90	j 106	z 122
1 0 1 1	B	VT 11	ESC 27	+ 43	; 59	K 75	[91	k 107	{ 123
1 1 0 0	C	FF 12	FS 28	, 44	< 60	L 76	\ 92	l 108	| 124
1 1 0 1	D	CR 13	GS 29	- 45	= 61	M 77	] 93	m 109	} 125
1 1 1 0	E	SO 14	RS 30	. 46	> 62	N 78	^ 94	n 110	~ 126
1 1 1 1	F	SI 15	US 31	/ 47	? 63	O 79	_ 95	o 111	DEL 127

Steuerzeichen

Tabelle 1: Der 7-Bit-ASCII-Code

Wie wohl grundsätzlich jede Art der Normierung wird auch der Standard des ASCII-Codes untergraben. So stimmen auch bei Rechnern, die weitestgehend dem ASCII-Standard folgen, nicht unbedingt alle Zeichen mit der Norm überein. Da einige Zeichen für die Kommunikation vollkommen belanglos sind, werden sie gern von den Systementwicklern anders belegt. Dies geschieht auch im CPC, obwohl er mit einem 256 Zeichen großen Zeichensatz ausreichend Platz für Kreativität im Bereich der oberen 128 Zeichen bietet.

Die oberen 128 der 256 vom Betriebssystem zur Verfügung gestellten Zeichen werden von der 7-Bit-ASCII-Norm nicht erfaßt und können daher mit beliebigen Zeichen gefüllt werden.

Wie auch bei vielen anderen Rechnern mit einem 256 Zeichen großen Zeichensatz, wird wird beim CPC mit der Adressierung über das Bit mit der Wertigkeit 2^7 ein Tor in die Welt der Grafikzeichen eröffnet. Zwar gibt es Bestrebungen den 8-Bit-ASCII-Code einzuführen, der auch diesen Bereich des Zeichensatzes normiert, doch hat er sich bis zum heutigen Datum noch nicht durchsetzen können.

Welche Abweichungen vom "American Standard Code for Information Interchange" und welche zusätzlichen Zeichen die Entwickler des CPCs in seine ROMs eingebrannt haben, werden Sie im Verlauf dieses Kapitels erfahren.

3.1 Der Zeichensatz des CPC

Wir haben von Zeichen und Zeichennummern gehört, die die Grundlage für die Kommunikation bilden. Beides sind Begiffe, die Dinge beschreiben, die im Inneren des Rechner verborgen bleiben. Die Kommunikation zwischen zwei Rechnern kann durch die Übermittlung von Zeichennummern erfolgen, doch wenn der Rechner für seinen Benutzer eine Meldung auf den Bildschirm ausgibt, tut er das nicht in Form von Zeichennummern, sondern mit den zugehörigen Zeichen.

Damit der CPC die erzeugten Zeichennummern in Form von Zeichen auf dem Bildschirm ausgeben kann, hat er Routinen in seinem Betriebssystem, die zusammen den sogenannten *Charaktergenerator* ausmachen. Diese Routinen wandeln die Zeichennummern in auf dem Bildschirm darstellbare Zeichen um. Dabei greifen sie auf im ROM stehende Bitmuster (Zeichenmatrizen) zu, die für das Aussehen der Zeichen verantwortlich sind. Die Größe der Zeichen auf dem Bildschirm wird vom Charaktergenerator je nach eingestelltem MODE bestimmt, so daß eine Darstellung von 20, 40 und 80 Zeichen pro Zeile möglich ist. Des weiteren obliegt dem Charaktergenerator die Aufgabe, die Zeichen für die Bildschirmausgabe entsprechend der eingestellten Farbkombinationen einzufärben.

Jede Zeichenmatrix des CPC besteht aus acht mal acht Punkten, die in binärer Form im ROM des CPC angelegt ist. Die Zeichenmatrizen der 256 verfügbaren Zeichen belegen im ROM die Adressen von &3800 bis &3FFF. Sollten Sie versuchen wollen, die Zeichenmatrizen mit der BASIC-Funktion PEEK aus dem Speicher zu lesen, so wird Ihrem Versuch kein Erfolg beschieden sein. Denn die Hardware des CPC erlaubt keinen Zugriff auf das ROM aus der BASIC-Ebene, und so können auch die im ROM stehenden Zeichenmatrizen nicht ausgelesen werden. Wollen Sie die Zeichenmatrizen dennoch nutzen, so können Sie das nur tun, wenn Sie mit den - nur über Maschinensprache beeinflußbaren - Mechanismen, die zur Umschaltung der Speicherkonfiguration (RAM-ROM) führen, vertraut sind.

Das folgende Programm schreibt eine kleine Maschinenroutine in den Speicher des CPC, die Lesen aus dem ROM möglich macht und verdeutlicht, wie die Zeichenmatrizen gelesen werden können. Das Programm bringt alle Bytes der Zeichenmatrizen in Form von hexadezimalen Werten auf dem Bildschirm.

```
100 'RAM-ROM-UMSCHALTUNG
110 '
120 DATA &01,&82,&7F        :'LD   BC,&7F82 [1]
130 DATA &ED,&49            :'OUT  (C),C
140 DATA &1A                :'LD   A,(DE)
150 DATA &32,&7F,&AB        :'LD   (&AB7F),A
160 DATA &C9                :'RET
170 '
180 MEMORY &A000
190 '
200 FOR A=&AB70 TO &AB79
210 READ D
220 POKE A,D
230 NEXT A
240 '
```

1 Beim CPC 464 führen Remarks in DATA-Statements zu Syntax-Error - die Kommentare dürfen also nicht eingegeben werden.

```
250 'ZEICHENSATZ AUSLESEN
260 '
270 FOR A=&3800 TO &3FFF
280 CALL &AB70,A
290 PRINT HEX$(PEEK(&AB7F))
300 NEXT A
```

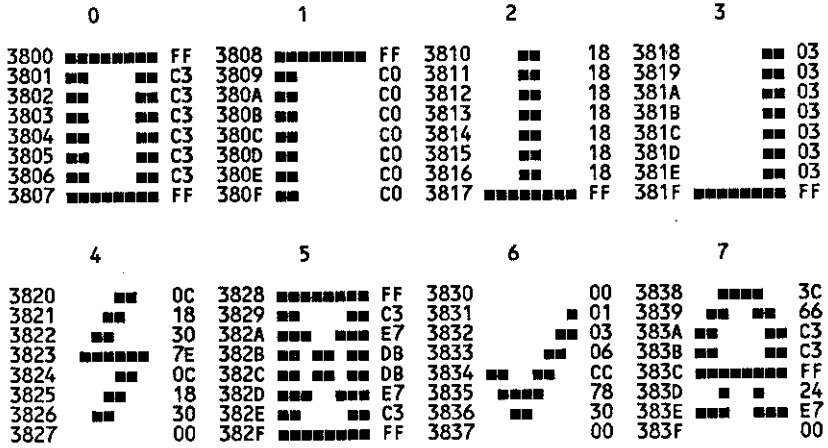
Die in den DATA-Statements verschlüsselten Maschinenbefehle werden durch die erste FOR-TO-NEXT-Schleife in den Speicher geschrieben und ergeben dort eine kleine Routine, die es ermöglicht das ROM des CPC auszulesen. In der zweiten Schleife wird stets auf diese Routine zugegriffen, wenn ein Byte aus einer Zeichenmatrix gelesen werden soll. Beim Aufruf der Maschinenroutine (CALL) wird ihr die Adresse, von der aus dem ROM gelesen werden soll, übergeben. Das Byte, das die Routine an der betreffenden Speicherstelle des ROMs vorfindet, schreibt sie an Adresse &AB7F in den RAM-Speicher. Dort kann es mit der PEEK-Funktion gelesen werden (Zeile 290). Die Aufstellung aller Zeichen des Schneider-CPC, die Sie im Anschluß an ein paar begleitende Sätze finden, basiert auf den Ergebnissen dieses Programms.

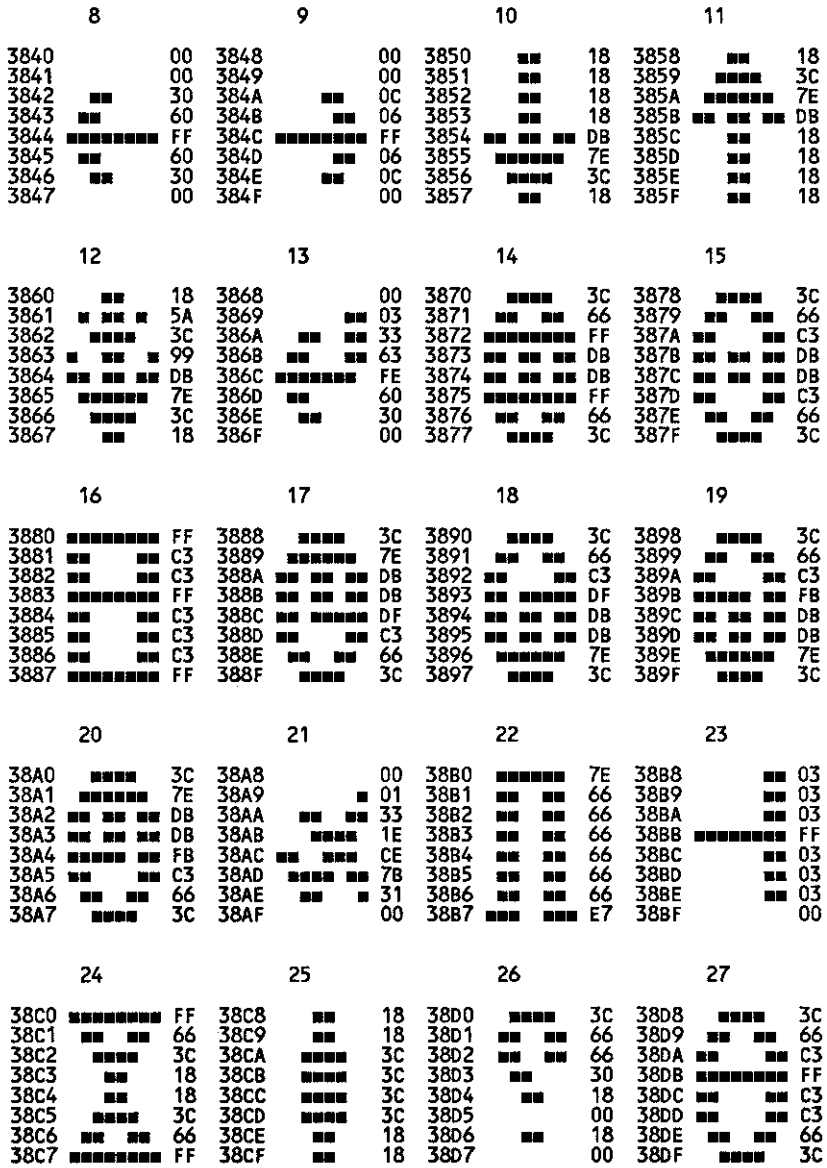
Der Zeichensatz des Schneider-CPC besteht aus 256 Zeichen, deren Zeichenmatrizen aus je acht Bytes gebildet werden. Die Zeichenmatrizen belegen also 2 KByte des ROM-Speicher im CPC. Um einen Überblick über die 256 Zeichen und ihre Gestalt zu gewinnen, haben wir die mit dem eben vorgestellten Programm gewonnenen Bytes zu Gruppen von je acht organisiert, die zusammen die Matrix eines Zeichens ausmachen. Da die vom Programm gelieferte hexadezimale Darstellung sich hervorragend zur Weiterverarbeitung eignet, nicht aber, um das Aussehen der Zeichen zu erfassen, haben wir die Bytes in ihre binären Entsprechungen umgerechnet. Gesetzte Bits, die für gesetzte Pixels bei der Darstellung eines Zeichens stehen, wurden mit einem "■" bezeichnet, nicht gesetzte mit einem Leerzeichen. Auf diese Weise entstand die acht-mal-acht-Matrix des Zeichens in einer Darstellungsform, die der Darstellung der Zeichen auf dem Bildschirm gleicht.

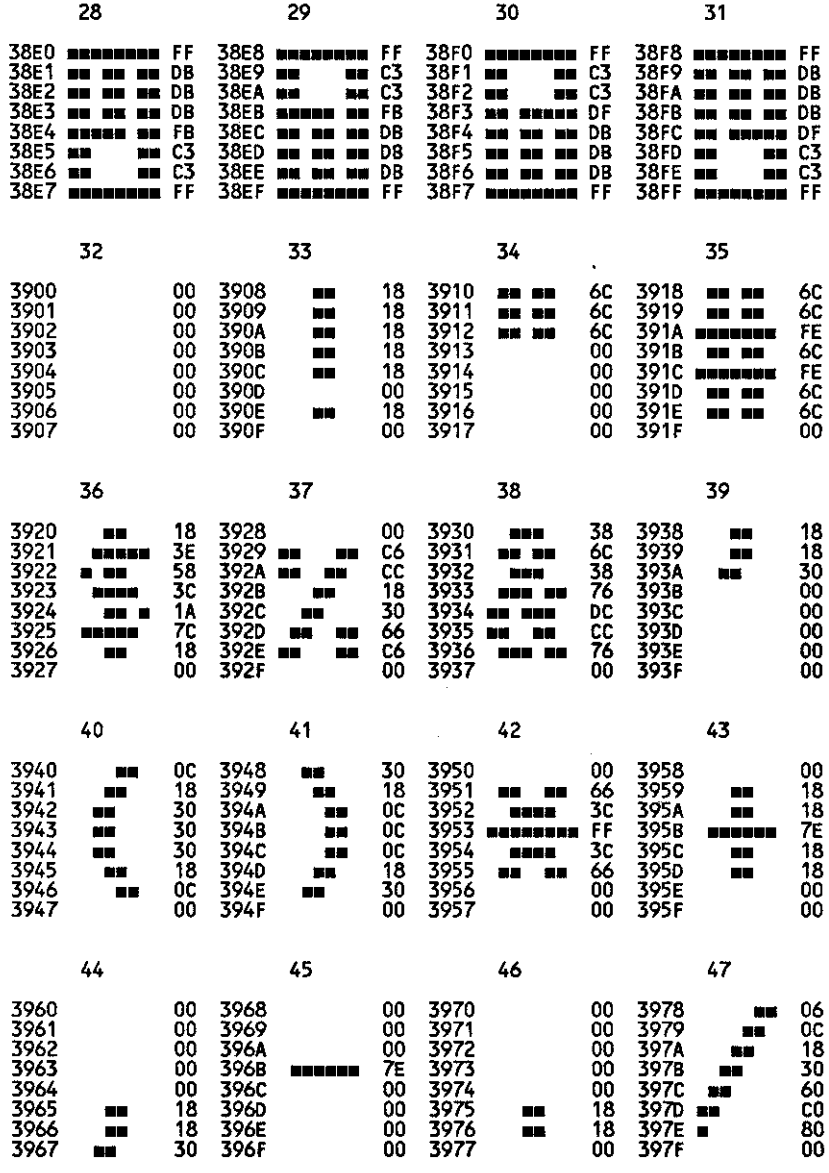
Um nun neben der Zeichenmatrix in binärer und hexadezimaler Darstellung weitere Informationen zu liefern, enthält die Aufstellung des Zeichensatzes noch die ROM-Adressen, an denen die einzelnen Bytes der Matrix zu finden sind. Die Adressen sind in hexadezimaler Form abgedruckt und können in das oben abgedruckte Programm zum Lesen der Zeichenmatrizen eingebracht werden, um eine Zeichenmatrix isoliert zu betrachten.

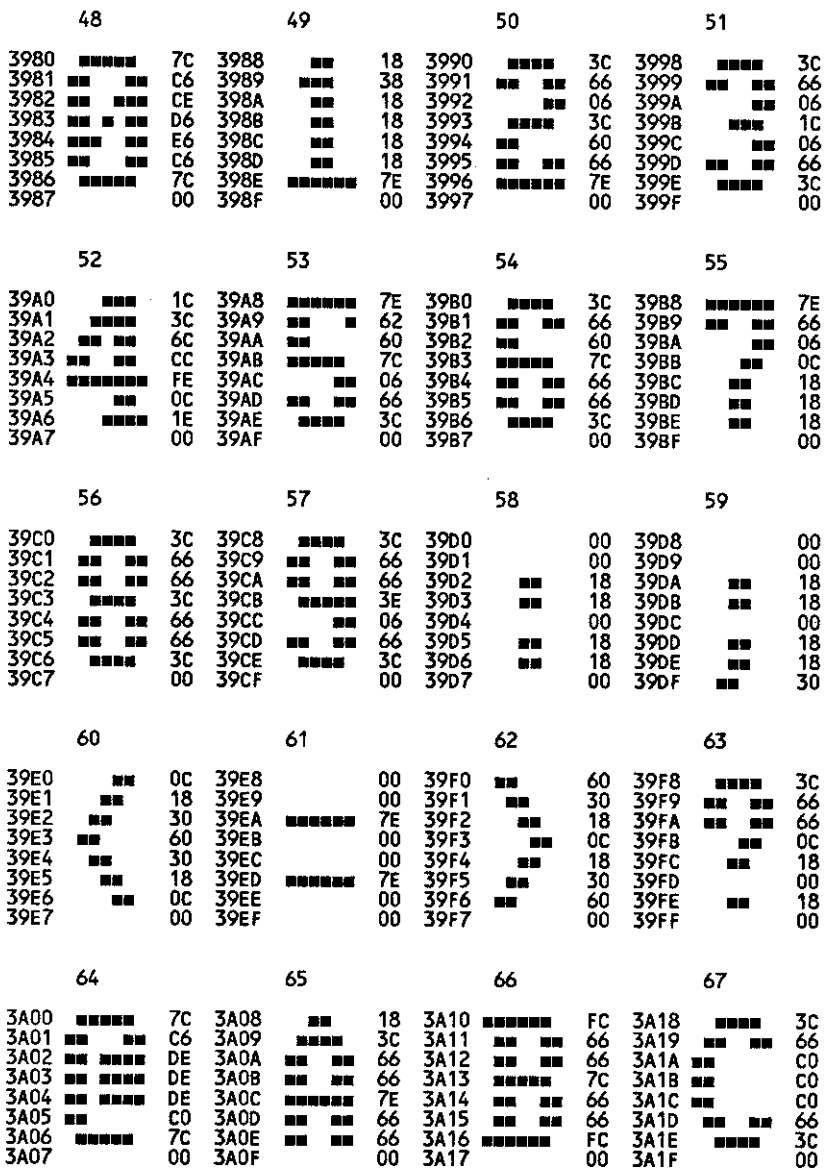
Als letzte Information, die aus der Aufstellung gezogen werden kann, befindet sich über jeder Zeichenmatrix die Nummer des dargestellten Zeichens. Mit dieser Nummer in ein PRINT CHR$-Kommando eingesetzt, kann das betreffende Zeichen auf den Bildschirm gebracht werden.

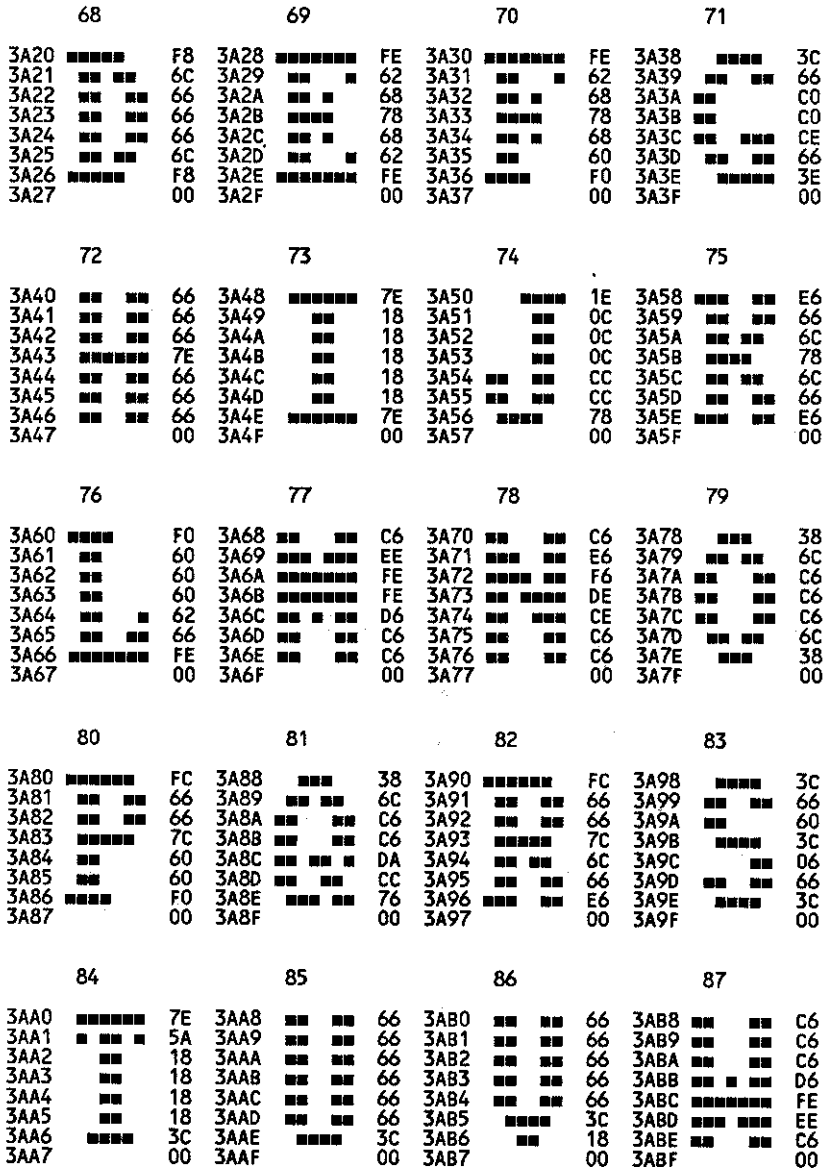
Wenn Sie bei der Betrachtung des Zeichensatzes und seiner Darstellung den Zeichen 32 bis 127 besonderes Augenmerk schenken, so werden Sie zweifellos die Grundsätze des einleitend erwähnten ASCII-Norm verwirklicht finden. Die ASCII-Steuerzeichen, die vor diesem Bereich stehen, sollten Sie noch aus Ihren Betrachtungen ausklammern, denn auf die Steuerzeichen wird in einem gesonderten Kapitel (3.2) ausführlich eingegangen werden.




8
3840 00
3841 00
3842 30
3843 60
3844 FF
3845 60
3846 30
3847 00
9
3848 00
3849 00
384A 0C
384B 06
384C FF
384D 06
384E 0C
384F 00
10
3850 18
3851 18
3852 18
3853 18
3854 DB
3855 7E
3856 3C
3857 18
11
3858 18
3859 3C
385A 7E
385B DB
385C 18
385D 18
385E 18
385F 18
12
3860 18
3861 5A
3862 3C
3863 99
3864 DB
3865 7E
3866 3C
3867 18
13
3868 00
3869 03
386A 33
386B 63
386C FE
386D 60
386E 30
386F 00
14
3870 3C
3871 66
3872 FF
3873 DB
3874 DB
3875 FF
3876 66
3877 3C
15
3878 3C
3879 66
387A C3
387B DB
387C DB
387D C3
387E 66
387F 3C
16
3880 FF
3881 C3
3882 C3
3883 FF
3884 C3
3885 C3
3886 C3
3887 FF
17
3888 3C
3889 7E
388A DB
388B DB
388C DF
388D C3
388E 66
388F 3C
18
3890 3C
3891 66
3892 C3
3893 DF
3894 DB
3895 DB
3896 7E
3897 3C
19
3898 3C
3899 66
389A C3
389B FB
389C DB
389D DB
389E 7E
389F 3C
20
38A0 3C
38A1 7E
38A2 DB
38A3 DB
38A4 FB
38A5 C3
38A6 66
38A7 3C
21
38A8 00
38A9 01
38AA 33
38AB 1E
38AC CE
38AD 7B
38AE 31
38AF 00
22
38B0 7E
38B1 66
38B2 66
38B3 66
38B4 66
38B5 66
38B6 66
38B7 E7
23
38B8 03
38B9 03
38BA 03
38BB FF
38BC 03
38BD 03
38BE 03
38BF 00
24
38C0 FF
38C1 66
38C2 3C
38C3 18
38C4 18
38C5 3C
38C6 66
38C7 FF
25
38C8 18
38C9 18
38CA 3C
38CB 3C
38CC 3C
38CD 3C
38CE 18
38CF 18
26
38D0 3C
38D1 66
38D2 66
38D3 30
38D4 18
38D5 00
38D6 18
38D7 00
27
38D8 3C
38D9 66
38DA C3
38DB FF
38DC C3
38DD C3
38DE 66
38DF 3C


28
38E0 FF
38E1 DB
38E2 DB
38E3 DB
38E4 FB
38E5 C3
38E6 C3
38E7 FF
29
38E8 FF
38E9 C3
38EA C3
38EB FB
38EC DB
38ED DB
38EE DB
38EF FF
30
38F0 FF
38F1 C3
38F2 C3
38F3 DF
38F4 DB
38F5 DB
38F6 DB
38F7 FF
31
38F8 FF
38F9 DB
38FA DB
38FB DB
38FC DF
38FD C3
38FE C3
38FF FF
32
3900 00
3901 00
3902 00
3903 00
3904 00
3905 00
3906 00
3907 00
33
3908 18
3909 18
390A 18
390B 18
390C 18
390D 00
390E 18
390F 00
34
3910 6C
3911 6C
3912 6C
3913 00
3914 00
3915 00
3916 00
3917 00
35
3918 6C
3919 6C
391A FE
391B 6C
391C FE
391D 6C
391E 6C
391F 00
36
3920 18
3921 3E
3922 58
3923 3C
3924 1A
3925 7C
3926 18
3927 00
37
3928 00
3929 C6
392A CC
392B 18
392C 30
392D 66
392E C6
392F 00
38
3930 38
3931 6C
3932 38
3933 76
3934 DC
3935 CC
3936 76
3937 00
39
3938 18
3939 18
393A 30
393B 00
393C 00
393D 00
393E 00
393F 00
40
3940 0C
3941 18
3942 30
3943 30
3944 30
3945 18
3946 0C
3947 00
41
3948 30
3949 18
394A 0C
394B 0C
394C 0C
394D 18
394E 30
394F 00
42
3950 00
3951 66
3952 3C
3953 FF
3954 3C
3955 66
3956 00
3957 00
43
3958 00
3959 18
395A 18
395B 7E
395C 18
395D 18
395E 00
395F 00
44
3960 00
3961 00
3962 00
3963 00
3964 00
3965 18
3966 18
3967 30
45
3968 00
3969 00
396A 00
396B 7E
396C 00
396D 00
396E 00
396F 00
46
3970 00
3971 00
3972 00
3973 00
3974 00
3975 18
3976 18
3977 00
47
3978 06
3979 0C
397A 18
397B 30
397C 60
397D C0
397E 80
397F 00


48
3980 7C
3981 C6
3982 CE
3983 D6
3984 E6
3985 C6
3986 7C
3987 00
49
3988 18
3989 38
398A 18
398B 18
398C 18
398D 18
398E 7E
398F 00
50
3990 3C
3991 66
3992 06
3993 3C
3994 60
3995 66
3996 7E
3997 00
51
3998 3C
3999 66
399A 06
399B 1C
399C 06
399D 66
399E 3C
399F 00
52
39A0 1C
39A1 3C
39A2 6C
39A3 CC
39A4 FE
39A5 0C
39A6 1E
39A7 00
53
39A8 7E
39A9 62
39AA 60
39AB 7C
39AC 06
39AD 66
39AE 3C
39AF 00
54
39B0 3C
39B1 66
39B2 60
39B3 7C
39B4 66
39B5 66
39B6 3C
39B7 00
55
39B8 7E
39B9 66
39BA 06
39BB 0C
39BC 18
39BD 18
39BE 18
39BF 00
56
39C0 3C
39C1 66
39C2 66
39C3 3C
39C4 66
39C5 66
39C6 3C
39C7 00
57
39C8 3C
39C9 66
39CA 66
39CB 3E
39CC 06
39CD 66
39CE 3C
39CF 00
58
39D0 00
39D1 00
39D2 18
39D3 18
39D4 00
39D5 18
39D6 18
39D7 00
59
39D8 00
39D9 00
39DA 18
39DB 18
39DC 00
39DD 18
39DE 18
39DF 30
60
39E0 0C
39E1 18
39E2 30
39E3 60
39E4 30
39E5 18
39E6 0C
39E7 00
61
39E8 00
39E9 00
39EA 7E
39EB 00
39EC 00
39ED 7E
39EE 00
39EF 00
62
39F0 60
39F1 30
39F2 18
39F3 0C
39F4 18
39F5 30
39F6 60
39F7 00
63
39F8 3C
39F9 66
39FA 66
39FB 0C
39FC 18
39FD 00
39FE 18
39FF 00
64
3A00 7C
3A01 C6
3A02 DE
3A03 DE
3A04 DE
3A05 C0
3A06 7C
3A07 00
65
3A08 18
3A09 3C
3A0A 66
3A0B 66
3A0C 7E
3A0D 66
3A0E 66
3A0F 00
66
3A10 FC
3A11 66
3A12 66
3A13 7C
3A14 66
3A15 66
3A16 FC
3A17 00
67
3A18 3C
3A19 66
3A1A C0
3A1B C0
3A1C C0
3A1D 66
3A1E 3C
3A1F 00


68
3A20 F8
3A21 6C
3A22 66
3A23 66
3A24 66
3A25 6C
3A26 F8
3A27 00
69
3A28 FE
3A29 62
3A2A 68
3A2B 78
3A2C 68
3A2D 62
3A2E FE
3A2F 00
70
3A30 FE
3A31 62
3A32 68
3A33 78
3A34 68
3A35 60
3A36 F0
3A37 00
71
3A38 3C
3A39 66
3A3A C0
3A3B C0
3A3C CE
3A3D 66
3A3E 3E
3A3F 00
72
3A40 66
3A41 66
3A42 66
3A43 7E
3A44 66
3A45 66
3A46 66
3A47 00
73
3A48 7E
3A49 18
3A4A 18
3A4B 18
3A4C 18
3A4D 18
3A4E 7E
3A4F 00
74
3A50 1E
3A51 0C
3A52 0C
3A53 0C
3A54 CC
3A55 CC
3A56 78
3A57 00
75
3A58 E6
3A59 66
3A5A 6C
3A5B 78
3A5C 6C
3A5D 66
3A5E E6
3A5F 00
76
3A60 F0
3A61 60
3A62 60
3A63 60
3A64 62
3A65 66
3A66 FE
3A67 00
77
3A68 C6
3A69 EE
3A6A FE
3A6B FE
3A6C D6
3A6D C6
3A6E C6
3A6F 00
78
3A70 C6
3A71 E6
3A72 F6
3A73 DE
3A74 CE
3A75 C6
3A76 C6
3A77 00
79
3A78 38
3A79 6C
3A7A C6
3A7B C6
3A7C C6
3A7D 6C
3A7E 38
3A7F 00
80
3A80 FC
3A81 66
3A82 66
3A83 7C
3A84 60
3A85 60
3A86 F0
3A87 00
81
3A88 38
3A89 6C
3A8A C6
3A8B C6
3A8C DA
3A8D CC
3A8E 76
3A8F 00
82
3A90 FC
3A91 66
3A92 66
3A93 7C
3A94 6C
3A95 66
3A96 E6
3A97 00
83
3A98 3C
3A99 66
3A9A 60
3A9B 3C
3A9C 06
3A9D 66
3A9E 3C
3A9F 00
84
3AA0 7E
3AA1 5A
3AA2 18
3AA3 18
3AA4 18
3AA5 18
3AA6 3C
3AA7 00
85
3AA8 66
3AA9 66
3AAA 66
3AAB 66
3AAC 66
3AAD 66
3AAE 3C
3AAF 00
86
3AB0 66
3AB1 66
3AB2 66
3AB3 66
3AB4 66
3AB5 3C
3AB6 18
3AB7 00
87
3AB8 C6
3AB9 C6
3ABA C6
3ABB D6
3ABC FE
3ABD EE
3ABE C6
3ABF 00

88

3AC0 C6
3AC1 6C
3AC2 38
3AC3 38
3AC4 6C
3AC5 C6
3AC6 C6
3AC7 00

89

3AC8 66
3AC9 66
3ACA 66
3ACB 3C
3ACC 18
3ACD 18
3ACE 3C
3ACF 00

90

3AD0 FE
3AD1 C6
3AD2 8C
3AD3 18
3AD4 32
3AD5 66
3AD6 FE
3AD7 00

91

3AD8 3C
3AD9 30
3ADA 30
3ADB 30
3ADC 30
3ADD 30
3ADE 3C
3ADF 00

92

3AE0 C0
3AE1 60
3AE2 30
3AE3 18
3AE4 0C
3AE5 06
3AE6 02
3AE7 00

93

3AE8 3C
3AE9 0C
3AEA 0C
3AEB 0C
3AEC 0C
3AED 0C
3AEE 3C
3AEF 00

94

3AF0 18
3AF1 3C
3AF2 7E
3AF3 18
3AF4 18
3AF5 18
3AF6 18
3AF7 00

95

3AF8 00
3AF9 00
3AFA 00
3AFB 00
3AFC 00
3AFD 00
3AFE 00
3AFF FF

96

3B00 30
3B01 18
3B02 0C
3B03 00
3B04 00
3B05 00
3B06 00
3B07 00

97

3B08
3B09
3B0A
3B0B
3B0C
3B0D
3B0E
3B0F

98

3B10 E0
3B11 60
3B12 7C
3B13 66
3B14 66
3B15 66
3B16 DC
3B17 00

99

3B18 00
3B19 00
3B1A 3C
3B1B 66
3B1C 60
3B1D 66
3B1E 3C
3B1F 00

100

3B20 1C
3B21 0C
3B22 7C
3B23 CC
3B24 CC
3B25 CC
3B26 76
3B27 00

101

3B28
3B29
3B2A
3B2B
3B2C
3B2D
3B2E
3B2F

102

3B30 1C
3B31 36
3B32 30
3B33 78
3B34 30
3B35 30
3B36 78
3B37 00

103

3B38 00
3B39 00
3B3A 3E
3B3B 66
3B3C 66
3B3D 3E
3B3E 06
3B3F 7C

104

3B40 E0
3B41 60
3B42 6C
3B43 76
3B44 66
3B45 66
3B46 E6
3B47 00

105

3B48
3B49
3B4A
3B4B
3B4C
3B4D
3B4E
3B4F

106

3B50 06
3B51 00
3B52 0E
3B53 06
3B54 06
3B55 66
3B56 66
3B57 3C

107

3B58 E0
3B59 60
3B5A 66
3B5B 6C
3B5C 78
3B5D 6C
3B5E E6
3B5F 00

108

3B60 38
3B61 18
3B62 18
3B63 18
3B64 18
3B65 18
3B66 3C
3B67 00

109

3B68
3B69
3B6A
3B6B FE
3B6C D6
3B6D D6
3B6E C6
3B6F

110

3B70 00
3B71 00
3B72 6C
3B73 FE
3B74 D6
3B75 D6
3B76 C6
3B77 00

111

3B78 00
3B79 00
3B7A 3C
3B7B 66
3B7C 66
3B7D 66
3B7E 3C
3B7F 00

112

3B80 00
3B81 00
3B82 DC
3B83 66
3B84 66
3B85 7C
3B86 60
3B87 F0

113

3B88
3B89
3B8A DC
3B8B CC
3B8C CC
3B8D 7C
3B8E 0C
3B8F 1E

114

3B90 00
3B91 00
3B92 76
3B93 CC
3B94 CC
3B95 7C
3B96 F0
3B97 00

115

3B98 00
3B99 00
3B9A 3C
3B9B 60
3B9C 3C
3B9D 06
3B9E 7C
3B9F 00

116

3BA0 30
3BA1 30
3BA2 7C
3BA3 30
3BA4 30
3BA5 36
3BA6 1C
3BA7 00

117

3BA8
3BA9
3BAA
3BAB
3BAC
3BAD
3BAE
3BAF

118

3BB0 00
3BB1 00
3BB2 66
3BB3 66
3BB4 66
3BB5 66
3BB6 3E
3BB7 00

119

3BB8 00
3BB9 00
3BBA C6
3BBB D6
3BBC D6
3BBD FE
3BBE 6C
3BBF 00

120

3BC0 00
3BC1 00
3BC2 C6
3BC3 6C
3BC4 38
3BC5 6C
3BC6 C6
3BC7 00

121

3BC8
3BC9
3BCA
3BCB
3BCC
3BCD
3BCE
3BCF

122

3BD0 00
3BD1 00
3BD2 66
3BD3 66
3BD4 66
3BD5 3E
3BD6 06
3BD7 7C

123

3BD8 0E
3BD9 18
3BDA 18
3BDB 70
3BDC 18
3BDD 18
3BDE 0E
3BDF 00

124

3BE0 18
3BE1 18
3BE2 18
3BE3 18
3BE4 18
3BE5 18
3BE6 18
3BE7 00

125

3BE8 70
3BE9 18
3BEA 18
3BEB 0E
3BEC 18
3BED 18
3BEE 70
3BEF 00

126

3BF0 76
3BF1 DC
3BF2 00
3BF3 00
3BF4 00
3BF5 00
3BF6 00
3BF7 00

127

3BF8 CC
3BF9 33
3BFA CC
3BFB 33
3BFC CC
3BFD 33
3BFE CC
3BFF 33

128

3C00 00
3C01 00
3C02 00
3C03 00
3C04 00
3C05 00
3C06 00
3C07 00

129

3C08 00
3C09 00
3C0A 00
3C0B 00
3C0C 00
3C0D 00
3C0E 00
3C0F 00

130

3C10 F0
3C11 F0
3C12 F0
3C13 F0
3C14 00
3C15 00
3C16 00
3C17 00

131

3C18 FF
3C19 FF
3C1A FF
3C1B FF
3C1C 00
3C1D 00
3C1E 00
3C1F 00

132

3C20 00
3C21 00
3C22 00
3C23 00
3C24 F0
3C25 F0
3C26 F0
3C27 F0

133

3C28 F0
3C29 F0
3C2A F0
3C2B F0
3C2C F0
3C2D F0
3C2E F0
3C2F F0

134

3C30 F0
3C31 F0
3C32 F0
3C33 F0
3C34 F0
3C35 F0
3C36 F0
3C37 F0

135

3C38 FF
3C39 FF
3C3A FF
3C3B FF
3C3C F0
3C3D F0
3C3E F0
3C3F F0

136

3C40 00
3C41 00
3C42 00
3C43 00
3C44 0F
3C45 0F
3C46 0F
3C47 0F

137

3C48 F0
3C49 F0
3C4A F0
3C4B F0
3C4C 0F
3C4D 0F
3C4E 0F
3C4F 0F

138

3C50 F0
3C51 F0
3C52 F0
3C53 F0
3C54 0F
3C55 0F
3C56 0F
3C57 0F

139

3C58 FF
3C59 FF
3C5A FF
3C5B FF
3C5C 0F
3C5D 0F
3C5E 0F
3C5F 0F

140

3C60 00
3C61 00
3C62 00
3C63 00
3C64 FF
3C65 FF
3C66 FF
3C67 FF

141

3C68 F0
3C69 F0
3C6A F0
3C6B F0
3C6C FF
3C6D FF
3C6E FF
3C6F FF

142

3C70 F0
3C71 F0
3C72 F0
3C73 F0
3C74 FF
3C75 FF
3C76 FF
3C77 FF

143

3C78 FF
3C79 FF
3C7A FF
3C7B FF
3C7C FF
3C7D FF
3C7E FF
3C7F FF

144

3C80 00
3C81 00
3C82 00
3C83 18
3C84 18
3C85 00
3C86 00
3C87 00

145

3C88 18
3C89 18
3C8A 18
3C8B 18
3C8C 18
3C8D 00
3C8E 00
3C8F 00

146

3C90 00
3C91 00
3C92 00
3C93 1F
3C94 1F
3C95 00
3C96 00
3C97 00

147

3C98 18
3C99 18
3C9A 18
3C9B 1F
3C9C 0F
3C9D 00
3C9E 00
3C9F 00

148

3CA0 00
3CA1 00
3CA2 00
3CA3 18
3CA4 18
3CA5 18
3CA6 18
3CA7 18

149

3CA8 18
3CA9 18
3CAA 18
3CAB 18
3CAC 18
3CAD 18
3CAE 18
3CAF 18

150

3CB0 18
3CB1 18
3CB2 18
3CB3 0F
3CB4 1F
3CB5 18
3CB6 18
3CB7 18

151

3CB8 18
3CB9 18
3CBA 18
3CBB 1F
3CBC 1F
3CBD 18
3CBE 18
3CBF 18

152

3CC0 00
3CC1 00
3CC2 00
3CC3 F8
3CC4 F8
3CC5 00
3CC6 00
3CC7 00

153

3CC8 18
3CC9 18
3CCA 18
3CCB F8
3CCC F0
3CCD 00
3CCE 00
3CCF 00

154

3CD0 18
3CD1 18
3CD2 18
3CD3 F8
3CD4 F0
3CD5 00
3CD6 00
3CD7 00

155

3CD8 18
3CD9 18
3CDA 18
3CDB FF
3CDC FF
3CDD 00
3CDE 00
3CDF 00

156

3CE0 00
3CE1 00
3CE2 00
3CE3 F0
3CE4 F8
3CE5 18
3CE6 18
3CE7 18

157

3CE8 18
3CE9 18
3CEA 18
3CEB F8
3CEC F8
3CED 18
3CEE 18
3CEF 18

158

3CF0 18
3CF1 18
3CF2 18
3CF3 F8
3CF4 F8
3CF5 18
3CF6 18
3CF7 18

159

3CF8 18
3CF9 18
3CFA 18
3CFB FF
3CFC FF
3CFD 18
3CFE 18
3CFF 18

160

3D00 10
3D01 38
3D02 6C
3D03 C6
3D04 00
3D05 00
3D06 00
3D07 00

161

3D08 0C
3D09 18
3D0A 30
3D0B 00
3D0C 00
3D0D 00
3D0E 00
3D0F 00

162

3D10 66
3D11 66
3D12 00
3D13 00
3D14 00
3D15 00
3D16 00
3D17 00

163

3D18 3C
3D19 66
3D1A 60
3D1B F8
3D1C 60
3D1D 66
3D1E FE
3D1F 00

164

3D20 38
3D21 44
3D22 BA
3D23 A2
3D24 BA
3D25 44
3D26 38
3D27 00

165

3D28 7E
3D29 F4
3D2A F4
3D2B 74
3D2C 34
3D2D 34
3D2E 34
3D2F 00

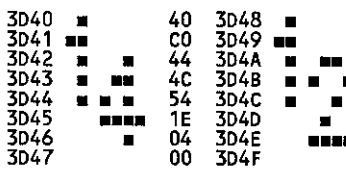
166

3D30 1E
3D31 30
3D32 38
3D33 6C
3D34 38
3D35 18
3D36 F0
3D37 00

167

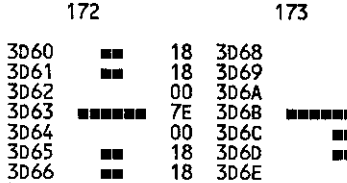
3D38 18
3D39 18
3D3A 0C
3D3B 00
3D3C 00
3D3D 00
3D3E 00
3D3F 00

168



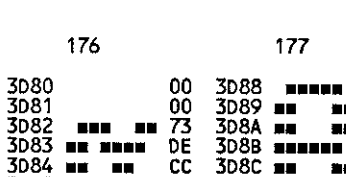
3D40		40	3D48		40	3D50		E0	3D58		00
3D41		C0	3D49		C0	3D51		10	3D59		18
3D42		44	3D4A		4C	3D52		62	3D5A		18
3D43		4C	3D4B		52	3D53		16	3D5B		7E
3D44		54	3D4C		44	3D54		EA	3D5C		18
3D45		1E	3D4D		08	3D55		0F	3D5D		18
3D46		04	3D4E		1E	3D56		02	3D5E		7E
3D47		00	3D4F		00	3D57		00	3D5F		00

172



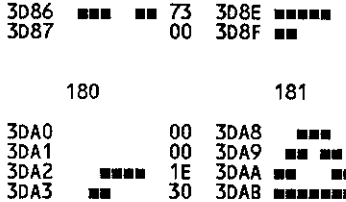
3D60		18	3D68		00	3D70		18	3D78		18
3D61		18	3D69		00	3D71		00	3D79		00
3D62		00	3D6A		00	3D72		18	3D7A		18
3D63		7E	3D6B		7E	3D73		30	3D7B		18
3D64		00	3D6C		06	3D74		66	3D7C		18
3D65		18	3D6D		06	3D75		66	3D7D		18
3D66		18	3D6E		00	3D76		3C	3D7E		18
3D67		00	3D6F		00	3D77		00	3D7F		00

176



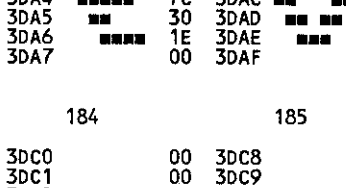
3D80		00	3D88		7C	3D90		00	3D98		3C
3D81		00	3D89		C6	3D91		66	3D99		60
3D82		73	3D8A		C6	3D92		66	3D9A		60
3D83		DE	3D8B		FC	3D93		3C	3D9B		3C
3D84		CC	3D8C		C6	3D94		66	3D9C		66
3D85		DE	3D8D		C6	3D95		66	3D9D		66
3D86		73	3D8E		F8	3D96		3C	3D9E		3C
3D87		00	3D8F		C0	3D97		00	3D9F		00

180



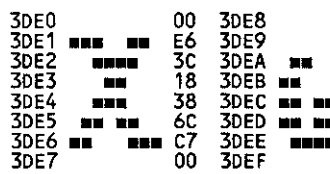
3DA0		00	3DA8		38	3DB0		00	3DB8		00
3DA1		00	3DA9		6C	3DB1		C0	3DB9		00
3DA2		1E	3DAA		C6	3DB2		60	3DBA		66
3DA3		30	3DAB		FE	3DB3		30	3DBB		66
3DA4		7C	3DAC		C6	3DB4		38	3DBC		66
3DA5		30	3DAD		6C	3DB5		6C	3DBD		7C
3DA6		1E	3DAE		38	3DB6		C6	3DBE		60
3DA7		00	3DAF		00	3DB7		00	3DBF		60

184



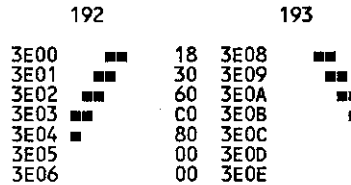
3DC0		00	3DC8		00	3DD0		03	3DD8		03
3DC1		00	3DC9		00	3DD1		06	3DD9		06
3DC2		00	3DCA		00	3DD2		0C	3DDA		0C
3DC3		FE	3DCB		7E	3DD3		3C	3DDB		66
3DC4		6C	3DCC		D8	3DD4		66	3DDC		66
3DC5		6C	3DCD		D8	3DD5		3C	3DDD		3C
3DC6		6C	3DCE		70	3DD6		60	3DDE		60
3DC7		00	3DCF		00	3DD7		C0	3DDF		C0

188



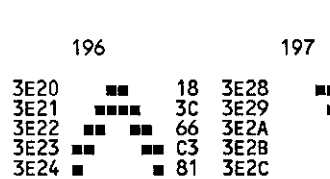
3DE0		00	3DE8		00	3DF0		FE	3DF8		00
3DE1		E6	3DE9		00	3DF1		C6	3DF9		7C
3DE2		3C	3DEA		66	3DF2		60	3DFA		C6
3DE3		18	3DEB		C3	3DF3		30	3DFB		C6
3DE4		38	3DEC		DB	3DF4		60	3DFC		C6
3DE5		6C	3DED		DB	3DF5		C6	3DFD		6C
3DE6		C7	3DEE		7E	3DF6		FE	3DFE		EE
3DE7		00	3DEF		00	3DF7		00	3DFF		00

192



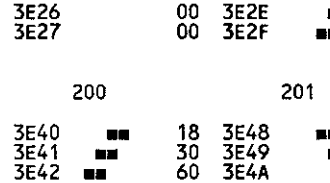
3E00		18	3E08		18	3E10		00	3E18		00
3E01		30	3E09		0C	3E11		00	3E19		00
3E02		60	3E0A		06	3E12		00	3E1A		00
3E03		C0	3E0B		03	3E13		01	3E1B		80
3E04		80	3E0C		01	3E14		03	3E1C		C0
3E05		00	3E0D		00	3E15		06	3E1D		60
3E06		00	3E0E		00	3E16		0C	3E1E		30
3E07		00	3E0F		00	3E17		18	3E1F		18

196



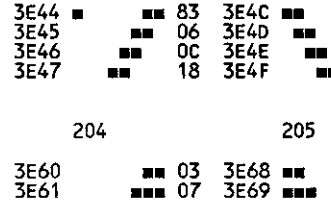
3E20		18	3E28		18	3E30		00	3E38		18
3E21		3C	3E29		0C	3E31		00	3E39		30
3E22		66	3E2A		06	3E32		00	3E3A		60
3E23		C3	3E2B		03	3E33		81	3E3B		C0
3E24		81	3E2C		03	3E34		C3	3E3C		C0
3E25		00	3E2D		06	3E35		66	3E3D		60
3E26		00	3E2E		0C	3E36		3C	3E3E		30
3E27		00	3E2F		18	3E37		18	3E3F		18

200



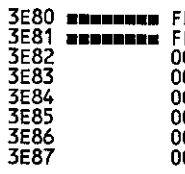
3E40		18	3E48		18	3E50		18	3E58		C3
3E41		30	3E49		0C	3E51		3C	3E59		E7
3E42		60	3E4A		06	3E52		66	3E5A		7E
3E43		C1	3E4B		83	3E53		C3	3E5B		3C
3E44		83	3E4C		C1	3E54		C3	3E5C		3C
3E45		06	3E4D		60	3E55		66	3E5D		7E
3E46		0C	3E4E		30	3E56		3C	3E5E		E7
3E47		18	3E4F		18	3E57		18	3E5F		C3

204



3E60		03	3E68		C0	3E70		CC	3E78		AA
3E61		07	3E69		E0	3E71		CC	3E79		55
3E62		0E	3E6A		70	3E72		33	3E7A		AA
3E63		1C	3E6B		38	3E73		33	3E7B		55
3E64		38	3E6C		1C	3E74		CC	3E7C		AA
3E65		70	3E6D		0E	3E75		CC	3E7D		55
3E66		E0	3E6E		07	3E76		33	3E7E		AA
3E67		C0	3E6F		03	3E77		33	3E7F		55

208



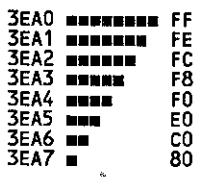
3E80	FF	3E88	03	3E90	00	3E98	C0
3E81	FF	3E89	03	3E91	00	3E99	C0
3E82	00	3E8A	03	3E92	00	3E9A	C0
3E83	00	3E8B	03	3E93	00	3E9B	C0
3E84	00	3E8C	03	3E94	00	3E9C	C0
3E85	00	3E8D	03	3E95	00	3E9D	C0
3E86	00	3E8E	03	3E96	FF	3E9E	C0
3E87	00	3E8F	03	3E97	FF	3E9F	C0

209

210

211

212



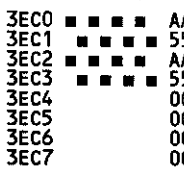
3EA0	FF	3EA8	FF	3EB0	01	3EB8	80
3EA1	FE	3EA9	7F	3EB1	03	3EB9	C0
3EA2	FC	3EAA	3F	3EB2	07	3EBA	E0
3EA3	F8	3EAB	1F	3EB3	0F	3EBB	F0
3EA4	F0	3EAC	0F	3EB4	1F	3EBC	F8
3EA5	E0	3EAD	07	3EB5	3F	3EBD	FC
3EA6	C0	3EAE	03	3EB6	7F	3EBE	FE
3EA7	80	3EAF	01	3EB7	FF	3EBF	FF

213

214

215

216



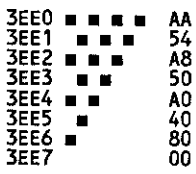
3EC0	AA	3EC8	0A	3ED0	00	3ED8	A0
3EC1	55	3EC9	05	3ED1	00	3ED9	50
3EC2	AA	3ECA	0A	3ED2	00	3EDA	A0
3EC3	55	3ECB	05	3ED3	00	3EDB	50
3EC4	00	3ECC	0A	3ED4	AA	3EDC	A0
3EC5	00	3ECD	05	3ED5	55	3EDD	50
3EC6	00	3ECE	0A	3ED6	AA	3EDE	A0
3EC7	00	3ECF	05	3ED7	55	3EDF	50

217

218

219

220



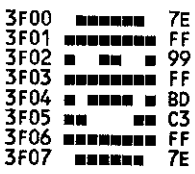
3EE0	AA	3EE8	AA	3EF0	01	3EF8	00
3EE1	54	3EE9	55	3EF1	02	3EF9	80
3EE2	A8	3EEA	2A	3EF2	05	3EFA	40
3EE3	50	3EEB	15	3EF3	0A	3EFB	A0
3EE4	A0	3EEC	0A	3EF4	15	3EFC	50
3EE5	40	3EED	05	3EF5	2A	3EFD	A8
3EE6	80	3EEE	02	3EF6	55	3EFE	54
3EE7	00	3EEF	01	3EF7	AA	3EFF	AA

221

222

223

224



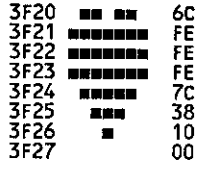
3F00	7E	3F08	7E	3F10	38	3F18	10
3F01	FF	3F09	FF	3F11	38	3F19	38
3F02	99	3F0A	99	3F12	FE	3F1A	7C
3F03	FF	3F0B	FF	3F13	FE	3F1B	FE
3F04	BD	3F0C	C3	3F14	FE	3F1C	7C
3F05	C3	3F0D	BD	3F15	10	3F1D	38
3F06	FF	3F0E	FF	3F16	38	3F1E	10
3F07	7E	3F0F	7E	3F17	00	3F1F	00

225

226

227

228



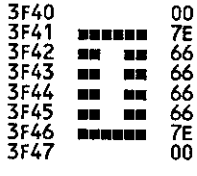
3F20	6C	3F28	10	3F30	00	3F38	00
3F21	FE	3F29	38	3F31	3C	3F39	3C
3F22	FE	3F2A	7C	3F32	66	3F3A	7E
3F23	FE	3F2B	FE	3F33	C3	3F3B	FF
3F24	7C	3F2C	FE	3F34	C3	3F3C	FF
3F25	38	3F2D	10	3F35	66	3F3D	7E
3F26	10	3F2E	38	3F36	3C	3F3E	3C
3F27	00	3F2F	00	3F37	00	3F3F	00

229

230

231

232



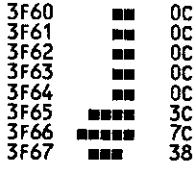
3F40	00	3F48	00	3F50	0F	3F58	3C
3F41	7E	3F49	7E	3F51	07	3F59	66
3F42	66	3F4A	7E	3F52	0D	3F5A	66
3F43	66	3F4B	7E	3F53	78	3F5B	66
3F44	66	3F4C	7E	3F54	CC	3F5C	3C
3F45	66	3F4D	7E	3F55	CC	3F5D	18
3F46	7E	3F4E	7E	3F56	CC	3F5E	7E
3F47	00	3F4F	00	3F57	78	3F5F	18

233

234

235

236



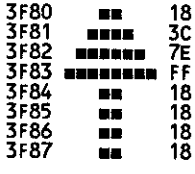
3F60	0C	3F68	18	3F70	99	3F78	10
3F61	0C	3F69	1C	3F71	5A	3F79	38
3F62	0C	3F6A	1E	3F72	24	3F7A	38
3F63	0C	3F6B	1B	3F73	C3	3F7B	38
3F64	0C	3F6C	18	3F74	C3	3F7C	38
3F65	3C	3F6D	78	3F75	24	3F7D	38
3F66	7C	3F6E	F8	3F76	5A	3F7E	7C
3F67	38	3F6F	70	3F77	99	3F7F	D6

237

238

239

240



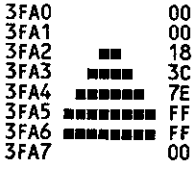
3F80	18	3F88	18	3F90	10	3F98	08
3F81	3C	3F89	18	3F91	30	3F99	0C
3F82	7E	3F8A	18	3F92	70	3F9A	0E
3F83	FF	3F8B	18	3F93	FF	3F9B	FF
3F84	18	3F8C	FF	3F94	FF	3F9C	FF
3F85	18	3F8D	7E	3F95	70	3F9D	0E
3F86	18	3F8E	3C	3F96	30	3F9E	0C
3F87	18	3F8F	18	3F97	10	3F9F	08

241

242

243

244

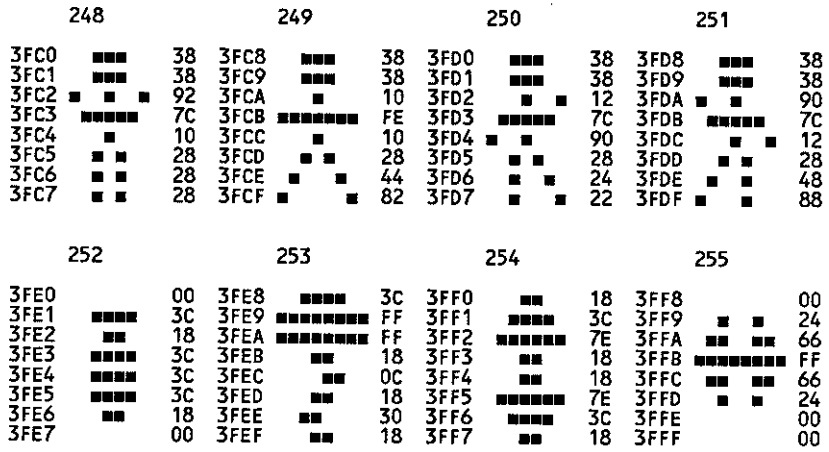


3FA0	00	3FA8	00	3FB0	80	3FB8	02
3FA1	00	3FA9	00	3FB1	E0	3FB9	0E
3FA2	18	3FAA	FF	3FB2	F8	3FBA	3E
3FA3	3C	3FAB	FF	3FB3	FE	3FBB	FE
3FA4	7E	3FAC	7E	3FB4	F8	3FBC	3E
3FA5	FF	3FAD	3C	3FB5	E0	3FBD	0E
3FA6	FF	3FAE	18	3FB6	80	3FBE	02
3FA7	00	3FAF	00	3FB7	00	3FBF	00

245

246

247



3.2 BASIC-Steuerzeichen

Es ist Ihnen sicher aufgefallen, daß einige Zeichen aus dem Zeichensatz des Schneider-CPC sich nicht ohne weiteres auf dem Bildschirm sichtbar machen lassen - so passieren zum Beispiel sehr merkwürdige Dinge, wenn man folgendes Programm ablaufen läßt:

```
10 FOR I=0 TO 255
20 PRINT CHR$(I);
30 NEXT I
```

Rein von der Logik müßte das Programm alle 256 verfügbaren Zeichen sauber nacheinander auf dem Bildschirm ausgeben. Doch weit gefehlt, es gibt einen kurzen Ton aus, das Bildschirmfenster verfärbt sich rot und nach einiger Zeit erscheint "Ready" in der linken oberen Ecke des Bildschirms. Schuld an diesem erstaunlichen Ergebnis sind die Zeichen mit den Nummer 0 bis 31, die *Steuerzeichen* genannt werden. Gibt man diese Zeichen mit PRINT CHR$ aus, so erwirken sie Vorgänge, die im Betriebssystem des Rechners verankert sind.

Greifen wir einmal exemplarisch eines der Steuerzeichen heraus und sehen, was passiert.

```
PRINT CHR$(7)
```

Nach dem Bestätigen dieser Eingabe mit der RETURN-Taste erklingt wieder der Ton, der sich schon bei unserem ersten Versuch hervorgetan hat. Nach dem Ton ist die Ton-Warteschlange entleert. Dieses Steuerzeichen gehört zu denen, die aus der ASCII-Norm entnommen sind. Es läßt sich einfach und gewinnbringend in eigenen Programmen einsetzen - es eignet sich hervorragend, um auftretende Fehler zu kommentieren.

Viele andere Steuerzeichen müssen zuerst mit weiteren Parametern versorgt werden, um sinnvolle Ergebnisse hervorzubringen; manche werden durch äquivalente BASIC-Kommandos überflüssig. Genaueres entnehmen Sie bitte der unten stehenden Aufstellung, in der alle Steuerzeichen und ihre Anwendungsmöglichkeiten aufgeführt sind.

Hinter jedem Steuerzeichen befindet sich die Bezeichnung, die das Steuerzeichen innerhalb des ASCII-Standards trägt. Diese Kennzeichnung ist bei der Verwendung der Steuerzeichen im CPC nicht immer sinnvoll, doch wurde sie dennoch in die Liste aufgenommen, um eine weitere Möglichkeit zu schaffen, Steuerzeichen beim Namen zu nennen.

CHR$(0) **NUL**

Kein Einfluß (Zeiger auf RET).

CHR$(1) **SOH**

Normalerweise werden die Zeichen von 0 bis 31 immer als Steuerzeichen ausgeführt. Möchte man aber eines der Zeichen

aus diesem Bereich auf dem Bildschirm darstellen und nicht als Steuerzeichen ausführen, so kann dies mit CHR$(1) erreicht werden.

Das folgende Programm soll die Funktionsweise des Steuer-Codes CHR$(1) verdeutlichen.

```
10 FOR ZEICHEN=0 TO 31
20 PRINT CHR$(1);CHR$(ZEICHEN);
30 NEXT ZEICHEN
40 FOR ZEICHEN=32 TO 255
50 PRINT CHR$(ZEICHEN);
60 NEXT ZEICHEN
```

Das Programm gibt alle 256 Zeichen des CPC auf dem Bildschirm aus. In der ersten FOR-TO-NEXT-Schleife, der die Darstellung der Zeichen von 0 bis 31 obliegt, kommt der CHR$(1) zur Anwendung. Er veranlaßt die grafische Darstellung der Steuerzeichen.

Bei der zweiten FOR-TO-NEXT-Schleife werden nur Zeichen ausgegeben, die ohnehin darstellbar sind. Hier ist also das Voranstellen eines CHR$(1) hinfällig.

CHR$(2) **STX**

Durch die Ausführung des CHR$(2) wird der Textcursor abgeschaltet. In der Auswirkung entspricht dieses Steuerzeichen dem BASIC-Befehl CURSOR, mit einer Null als Parameter für Benutzerschalter.

Normalerweise wird bei einem INPUT der Cursor an der Stelle des Bildschirms abgebildet, an der die Eingabe erwartet wird. Soll einmal der Cursor nicht abgebildet werden, so ist das ein

Anwendungsfall für CHR$(2). Die nachstehende Programmzeile soll den Zusammenhang verdeutlichen:

```
10 PRINT CHR$(2):INPUT "WERT";WERT
```

Die Darstellung des Cursors wurde durch den Steuercode CHR$(2) unterbunden.

CHR$(3) **ETX**

Durch die Ausführung des CHR$(3) wird der Textcursor eingeschaltet. In der Auswirkung entspricht dieses Steuerzeichen dem BASIC-Befehl CURSOR, mit einer 1 als Parameter für Benutzerschalter.

Wurde innerhalb eines BASIC-Programms der Textcursor mit CHR$(2) abgeschaltet, so kann er mit CHR$(3) wieder eingeschaltet werden.

```
10 PRINT CHR$(2):INPUT "ERSTER WERT";EWERT
20 PRINT CHR$(3):INPUT "ZWEITER WERT";ZWERT
```

In Zeile 10 des Programms wurde der Textcursor abgeschaltet, der INPUT wird also ohne Darstellung des Cursors ausgeführt (vgl. CHR$(2)). Beim zweiten INPUT in Zeile 20 soll der Textcursor wieder erscheinen - er wird also mit CHR$(3) wieder eingeschaltet.

CHR$(4) **EOT**

Dieser Steuercode entspricht in seiner Auswirkung dem BASIC-Befehl MODE X (X=0 bis 2).

```
PRINT CHR$(4);CHR$(0)   entspricht   MODE 0
PRINT CHR$(4);CHR$(1)   entspricht   MODE 1
PRINT CHR$(4);CHR$(2)   entspricht   MODE 2
```

Es sei noch angemerkt, daß als Parameter für CHR$(4) nicht der eingegebene Wert Verwendung findet, sondern sein Rest bei einer Division durch vier. Auf diese Weise ist es möglich, auch Parameter größer 2 zuzulassen.

```
PRINT CHR$(4);CHR$(1)
```

und

```
PRINT CHR$(4);CHR$(5)
```

sind somit gleichbedeutend und bringen beide den CPC in MODE 1.

CHR$(5) **ENQ**

Nach einem CHR$(5) wird das folgende Zeichen an der aktuellen Position des Grafikcursors ausgegeben. Das folgende Programm soll den Zusammenhang beleuchten:

```
10 CLS
20 FOR I=0 TO 399
30 MOVE I,I
40 PRINT CHR$(5);".";
50 NEXT
```


Das Programm läßt einen Punkt von links unten nach rechts oben über den Bildschirm gleiten. Dazu werden in einer FOR-TO-NEXT-Schleife alle ganzzahligen Werte zwischen 0 und 399 erzeugt. In Zeile 30 wird der Grafikcursor mit diesen Werten als Parameter bewegt. Zeile 40 bringt nun mit Hilfe von CHR$(5) einen Punkt an der aktuellen Position des Grafikcursors auf den Bildschirm. Bei jedem Durchlauf der Schleife verändert sich die Position des Grafikcursors und mit ihr auch die Stelle, an der der Punkt auf dem Bildschirm erscheint.

CHR$(6) **ACK**

Dieses Steuerzeichen schaltet den Textbildschirm wieder ein, nachdem er mit CHR$(21) ausgeschaltet wurde (vgl. CHR$(21))

```
10 PRINT CHR$(21)
20 FOR I=1 TO 40
30 PRINT "0";
40 NEXT
50 PRINT CHR$(6)
60 FOR I=1 TO 40
70 PRINT "1";
80 NEXT
```

Zunächst wird in Zeile 10 der Textbildschirm ausgeschaltet; er reagiert in diesem Zustand auf keine Eingaben mehr. Die Zeilen 20 bis 40 würden bei eingeschaltetem Bildschirm 40 Nullen auf dem Bildschirm ausgeben. Nach CHR$(21) unterbleibt diese Zeichendarstellung.

In Zeile 50 wird schließlich der Textbildschirm mit CHR$(6) wieder eingeschaltet, so daß die in den Zeilen 60 bis 80 ausgegebenen Zeichen auf dem Bildschirm erscheinen.

CHR$(7) **BEL**

PRINT CHR$(7) erzeugt einen Ton, der auch als ASCII-Bell bekannt ist. Nach diesem Ton ist die Ton-Warteschlange entleert.

CHR$(8) **BS**

Der Cursor wird um eine Zeichenposition nach links bewegt.

CHR$(9) **TAB**

Der Cursor wird um eine Zeichenposition nach rechts bewegt.

CHR$(10) **LF**

Der Cursor wird eine Zeile nach unten gesetzt.

CHR$(11) **VT**

Der Cursor wird eine Zeile nach oben gesetzt.

CHR$(12) **FF**

PRINT CHR$(12) entspricht dem BASIC-Kommando CLS. Es löscht das Textfenster und setzt den Cursor in die linke obere Ecke (HOME).

CHR$(13) **CR**

PRINT CHR$(13) entspricht einer Betätigung der Eingabetaste (RETURN).

CHR$(14) **SO**

Das Steuerzeichen CHR$(14) entspricht in seiner Auswirkung der BASIC-Anweisung PAPER. Der nachfolgende Wert wird, analog zu PAPER, als Parameter für die Farbe angenommen. Dabei wird nicht der Wert unmittelbar als Parameter verwendet, sondern sein Rest bei einer Division durch 16.

CHR$(15) **SI**

Das Steuerzeichen CHR$(15) entspricht in seiner Auswirkung der BASIC-Anweisung PEN. Der nachfolgende Wert wird, analog zu PEN, als Parameter für die Farbe angenommen. Dabei wird nicht der Wert unmittelbar als Parameter verwendet, sondern sein Rest bei einer Division durch 16.

CHR$(16) **DLE**

Dieses Steuerzeichen löscht das Zeichen an der Cursorposition und füllt die entstehende Lücke mit der PAPER-Farbe. Es ist zu beachten, daß CHR$(16) nur das Zeichen löscht, nicht aber eventuell folgende Zeichen um eine Position nach links rückt, um die entstandene Lücke zu schließen.

CHR$(17) **DC1**

CHR$(17) löscht in der aktuellen Zeile alle Zeichen vom linken Rand des Textfensters bis zur Cursorposition (einschließlich). Die entstehende Lücke wird analog zu CHR$(16) mit der PAPER-Farbe gefüllt.

CHR$(18) **DC2**

CHR$(18) löscht in der aktuellen Zeile alle Zeichen ab der Cursorposition (einschließlich) bis zum rechten Rand des Textfensters. Die entstehende Lücke wird analog zu CHR$(16) mit der PAPER-Farbe gefüllt.

CHR$(19) **DC3**

CHR$(19) löscht alle Zeichen von der linken oberen Ecke des Textfensters bis zur Cursorposition (einschließlich). Die entstehende Lücke wird analog zu CHR$(16) mit der PAPER-Farbe gefüllt.

CHR$(20) **DC4**

CHR$(20) löscht ab der Cursorposition (einschließlich) bis zum Ende des Textfensters (rechte untere Ecke) alle Zeichen und füllt analog zu CHR$(16) die entstehende Lücke mit der PAPER-Farbe.

CHR$(21) **NAK**

Dieses Steuerzeichen schaltet den Textbildschirm ab (vgl. CHR$(6)). Der Textbildschirm reagiert nach diesem Steuercode auf keinerlei Eingaben mehr. Mit CHR$(6) kann der Textbildschirm wieder eingeschaltet werden.

CHR$(22) **SYN**

CHR$(22) ist ein Schalter für den Transparent-Modus. Was damit gemeint ist, zeigt das folgende Programm sehr deutlich:

```
10 CLS
20 PRINT CHR$(22);CHR$(1)
```



```
30 LOCATE 10,10
40 PRINT "AAAAAAAAAA"
50 LOCATE 10,10
60 PRINT "//////////"
70 PRINT CHR$(22);CHR$(0)
80 LOCATE 10,12
90 PRINT "AAAAAAAAAA"
100 LOCATE 10,12
110 PRINT "//////////"
120 END
```

Folgt CHR$(22) der Parameter 1, so wird der Transparent-Modus eingeschaltet - wie in Zeile 20 erfolgt. Die beiden Zeichenketten, die dann im Verlauf des Programms an der gleichen Bildschirmposition ausgegeben werden, überlagern sich (Transparent-Modus ein!).

CHR$(22) gefolgt von Parameterwert 0 schaltet den Transparent-Modus aus (Zeile 70). Die gleichen Zeichenketten wie im ersten Teil des Programms überlagern sich dann nicht. Die ersten Zeichen werden bei der Ausgabe der zweiten Zeichenkette vollständig zerstört.

Auch hier wird als Parameter wieder nicht der Wert unmittelbar verwendet sondern sein Rest bei einer Division durch 2.

CHR$(23) **ETB**

CHR$(23) entscheidet über den Grafikschreibmodus. Als Parameter sind die Werte 0 bis 3 zulässig, die sich als Rest einer Division durch vier ergeben.

Parameter:

0 = Normale Einstellung
1 = XOR-Verknüpfung von Vorder- und Hintergrund

2 = AND-Verknüpfung von Vorder- und Hintergrund
3 = OR-Verknüpfung von Vorder- und Hintergrund

Abhängig von der Parametereinstellung werden die Ausgaben nach einem CHR$(23) gemäß den logischen Gesetzen XOR, AND oder OR mit dem Hintergrund verknüpft. Die Verknüpfung von Vorder- und Hintergrund erfolgt bitweise.

```
10 CLS
20 INPUT "PARAMETER";P
30 TAG
40 MOVE 200,200
50 PRINT "**********";
60 TAGOFF
70 PRINT CHR$(23);CHR$(P)
80 TAG
90 MOVE 200,200
100 PRINT "##########";
110 TAGOFF
```

In Zeile 20 wird nach der Frage "PARAMETER?" eine Zahl erwartet, die über die Verknüpfung zwischen Vorder- und Hintergrund entscheidet (vgl. Tabelle der Parameter). Die Zeilen 30 bis 60 erzeugen einen Hintergrund. Dann wird in Zeile 70 die Art der Verknüpfung (gemäß Vorgabe) festgelegt. Schließlich erfolgt eine Ausgabe auf dem Hintergrund. Vorder- und Hintergrund werden bitweise nach den Regeln der ausgewählten logischen Gesetzmäßigkeit miteinander verknüpft.

CHR$(24) CAN

Durch das Steuerzeichen CHR$(24) werden die Farbstifte von PAPER und PEN miteinander vertauscht.

CHR$(25) EM

CHR$(25) entspricht dem BASIC-Befehl SYMBOL. Wie SYMBOL benötigt CHR$(25) neun ergänzende Parameter, deren Werte zwischen 0 und 255 liegen müssen.

Mit dem ersten Wert, der auf das Steuerzeichen folgt, wird festgelegt, welche Zeichenmatrix durch eine neue vom Benutzer definierte Matrix ersetzt werden soll. Die acht dann folgenden Parameter definieren die Martix.

```
10 SYMBOL AFTER 0
20 PRINT CHR$(25);
30 PRINT CHR$(123);                                    1
40 PRINT CHR$(&X11011011);      '■■ ■■ ■■
41 PRINT CHR$(&X00111100);      '  ■■■■
42 PRINT CHR$(&X01100110);      ' ■■  ■■
43 PRINT CHR$(&X01100110);      ' ■■  ■■
44 PRINT CHR$(&X01111110);      ' ■■■■■■
45 PRINT CHR$(&X01100110);      ' ■■  ■■
46 PRINT CHR$(&X01100110);      ' ■■  ■■
47 PRINT CHR$(&X00000000);      '
```

Dieses Programm definiert mit Hilfe des Steuerzeichens CHR$(25) das Zeichen 123 ({) in ein "Ä" um. Die gewählte binäre Notation für die letzten acht Parameter verdeutlicht die Beziehung zur Zeichenmatrix. Ist im ersten dieser acht Parameter das signifikanteste Bit gesetzt, so wird dieses Bit auch in der Zeichenmatrix gesetzt. Analoges gilt für alle übrigen Bits der letzten acht Parameter.

CHR$(26) SUB

Das Steuerzeichen CHR$(26) entspricht dem BASIC-Befehl WINDOW (ohne Angabe eines Streams). Gefolgt wird dieser Steuercode von vier Parametern, die die Fenstergröße festlegen. Die ersten beiden Parameter legen den rechten und linken Rand

1 Diese Kommentare zu den DATA-Statements dienen ausschließlich der besseren Übersicht des Lesers und können nicht eingetippt werden.

des Fensters fest. Die Reihenfolge der Eingabe ist unerheblich, es wird automatisch der kleinere der beiden Werte als linke und der größere als rechte Fenstergrenze angenommen.

Analoges gilt für die letzten beiden Parameter. Der kleinere der beiden wird zur oberen und der größere zur unteren Fenstergrenze deklariert.

CHR$(27) **ESC**

Kein Einfluß (Zeiger auf RET).

CHR$(28) **FS**

Der Steuercode CHR$(28) entspricht dem BASIC-Befehl INK. Wie bei INK wird das Steuerzeichen von drei Parametern gefolgt. Der erste Parameter bzw. sein Rest bei einer Division durch 16 legt fest, welcher Farbstift durch die folgenden Farbangaben beeinflußt werden soll. Der Rest der nächsten beiden Parameter, bei einer Division durch 32, legt die beiden Farben für den gewünschten Farbstift fest.

Beispiel:

```
PRINT CHR$(28);CHR$(1);CHR$(0);CHR$(13)
```

entspricht

```
INK 1,0,13
```

CHR$(29) **GS**

Das Steuerzeichen CHR$(29) entspicht der BASIC-Anweisung BORDER. Das Steuerzeichen wird von zwei Parametern gefolgt, die die beiden Farbwerte für den Bildschirmrand repäsentieren.

Es werden für die Bestimmung der Farbwerte nicht die Parameter unmittelbar, sondern ihr Rest bei einer Division durch 32 verwendet.

CHR$(30) **RS**

Der Steuercode CHR$(30) bringt den Cursor in die linke obere Ecke des Textfensters (HOME).

CHR$(31) **US**

Wie der BASIC-Befehl LOCATE bringt das Steuerzeichen CHR$(31) den Cursor auf eine angegebene Position innerhalb des Textfensters. Die Position auf die der Cursor gebracht werden soll, wird durch die zwei auf das Steuerzeichen folgende Parameter bestimmt. Der erste Parameter gibt die Spalten- und der zweite die Zeilenposition an. Sind die Parameter größer als die Fenstergrenzen, so wird der Cursor auf die größtmögliche darstellbare Position gebracht.

3.3 Viele Zeichen ergeben ein Bild

Nachdem Sie den Zeichensatz des CPC in seiner Gesamtheit kennengelernt haben und wissen, wie man hartnäckigen Steuerzeichen zu Leibe rückt, wollen wir Ihnen eine einfache Möglichkeit zeigen, mit den vorhandenen Zeichen Grafiken zu erstellen.

Die Vorgehensweise ist denkbar einfach: Man nehme ein Bild, das als Vorlage dienen soll - es kann sowohl eine selbsterstellte Vorlage als auch ein vorhandenes Bild sein - und beginne mit einigen elementaren Überlegungen. Da ist zunächst festzulegen, welche Größe das Bild später auf dem Display haben soll und dementsprechend ist die Vorlage in einzelne quadratische Bildelemente zu zerlegen, die der Größe eines Charakters entsprechen werden. Ist die Vorlage auf diese Weise aufbereitet, so muß für jedes der Bildelemente ein Charakter gefunden werden, der

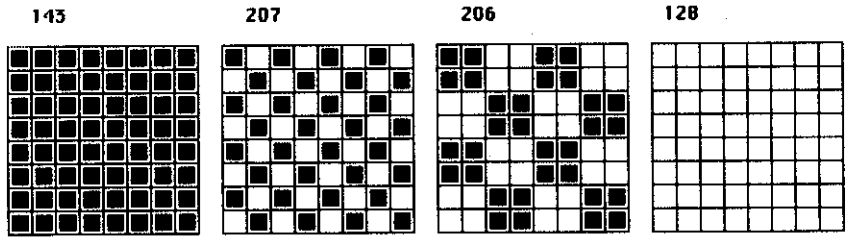
diesem möglichst genau entspricht. Die einzelnen Zeichen müssen dann nur noch mit einem geeigneten Programm auf den Bildschirm des Computers gebracht werden, und fertig ist die Grafik.

Sehen Sie sich die oben angestellten theoretischen Überlegungen doch einmal in der Praxis an! Als Vorlage für die zu erstellende Grafik soll Abbildung 4 dienen.

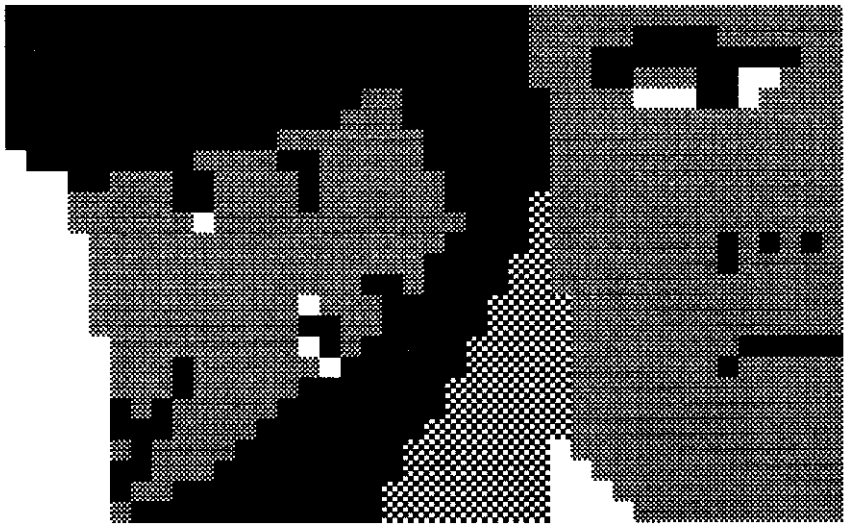
Abb. 4: **Vorlage für die Charaktergrafik**

Da es sich bei der vorliegenden Abbildung um kein Detail handelt, wie es in Verdindung mit dem Programmieren von Spielen denkbar wäre, sondern um ein in sich geschlossenes Motiv, soll es den gesamten Bildschirm ausfüllen. Bei einer Arbeit in MODE 1 bedeutet das, daß die Vorlage in 40 mal 25 Bildelemente zu unterteilen ist. Für jedes der eintausend Bildelemente muß nun ein Charakter gefunden werden, der es möglichst optimal wiedergibt. Um diese Arbeit nicht zur Sisyphusarbeit ausarten zu lassen, haben wir uns beim Digitalisieren der Grafik

auf Graustufen beschränkt, was dem Prinzip keinen Abbruch tut. Die verschiedenen Graustufen werden durch die Charakter 143 (schwarz), 207 (grau), 206 (hellgrau) und 128 (weiß) erzeugt.



Nachdem zu jedem Bildelement der Vorlage ein entsprechender Charakter bestimmt worden ist, müssen die Zeichennummern dieser Charakter zu DATA-Zeilen zusammengestellt werden. Beim Erzeugen der DATA-Statements haben wir darauf geachtet, daß die Zeichennummern, deren Zeichen später eine Zeile auf dem Bildschirm ausmachen, immer unter einer Statementnummer zusammengefaßt sind. In jeder der 25 DATA-Zeilen befinden sich also genau 40 Zeichennummern, was das Aufspüren von Tippfehlern erheblich erleichert.



Hardcopy 2: **Viele Zeichen ergeben ein Bild**

Der Weg, aus den tausend DATAs, in denen die Bildvorlage verschlüsselt ist, die Grafik auf dem Bildschirm entstehen zu lassen, ist eine Aufgabe aus der zweiten Stunde eines BASIC-Kurses. In einer FOR-TO-NEXT-Schleife, die genau 1000mal durchlaufen wird, werden die DATAs gelesen und mit PRINT CHR$ auf den Bildschirm gebracht. Von links oben nach rechts unten entsteht das Bild das auf Hardcopy 2 zu sehen ist.

```
100 MODE 1
110 INK 1,0
120 INK 0,13
130 '
140 FOR I=1 TO 1000
150 READ D
160 PRINT CHR$(D);
170 NEXT I
180 '
```



```
190 DATA 143,143,143,143,143,143,143,143,143,143,143,143,143,143,1
43,143,143,143,143,143,143,143,143,143,143,143,207,207,207,207,207,207
,207,207,207,207,207,207,207,207,207
200 DATA 143,143,143,143,143,143,143,143,143,143,143,143,143,143,1
43,143,143,143,143,143,143,143,143,143,143,143,207,207,207,207,207,143
,143,143,143,207,207,207,207,207,207
210 DATA 143,143,143,143,143,143,143,143,143,143,143,143,143,143,1
43,143,143,143,143,143,143,143,143,143,143,143,207,207,207,143,143,143
,143,143,143,143,143,143,143,207,207
220 DATA 143,143,143,143,143,143,143,143,143,143,143,143,143,143,1
43,143,143,143,143,143,143,143,143,143,143,143,207,207,207,143,143,207
,207,207,143,143,128,128,207,207,207
230 DATA 143,143,143,143,143,143,143,143,143,143,143,143,143,143,1
43,143,143,207,207,143,143,143,143,143,143,143,207,207,207,207,128
,128,128,143,143,128,207,207,207,207
240 DATA 143,143,143,143,143,143,143,143,143,143,143,143,143,143,1
43,143,207,207,207,143,143,143,143,143,143,143,207,207,207,207,207
,207,207,207,207,207,207,207,207,207
250 DATA 143,143,143,143,143,143,143,143,143,143,143,143,143,207,2
07,207,207,207,207,207,143,143,143,143,143,143,207,207,207,207,207
,207,207,207,207,207,207,207,207,207
260 DATA 128,143,143,143,143,143,143,143,143,207,207,207,207,143,1
43,207,207,207,207,207,143,143,143,143,143,143,207,207,207,207,207
,207,207,207,207,207,207,207,207,207
270 DATA 128,128,128,143,143,207,207,207,143,143,207,207,207,207,1
43,207,207,207,207,207,207,143,143,143,143,143,207,207,207,207,207
,207,207,207,207,207,207,207,207,207
280 DATA 128,128,128,207,207,207,207,207,143,143,207,207,207,207,1
43,207,207,207,207,207,207,143,143,143,143,206,207,207,207,207,207
,207,207,207,207,207,207,207,207,207
290 DATA 128,128,128,207,207,207,207,207,207,128,207,207,207,207,2
07,207,207,207,207,207,207,207,207,143,143,143,206,207,207,207,207,207
,207,207,207,207,207,207,207,207,207
300 DATA 128,128,128,128,207,207,207,207,207,207,207,207,207,207,2
07,207,207,207,207,207,207,207,143,143,143,143,206,207,207,207,207,207
,207,207,207,143,207,143,207,143,207
310 DATA 128,128,128,128,207,207,207,207,207,207,207,207,207,207,2
07,207,207,207,207,207,207,143,143,143,143,206,206,207,207,207,207,207
,207,207,207,143,207,207,207,207,207
```

```
320 DATA 128,128,128,128,207,207,207,207,207,207,207,207,207,207,2
07,207,207,143,143,207,143,143,143,143,206,206,207,207,207,207,207
,207,207,207,207,207,207,207,207,207
330 DATA 128,128,128,128,207,207,207,207,207,207,207,207,207,207,1
28,207,207,207,143,143,143,143,143,206,206,206,206,207,207,207,207
,207,207,207,207,207,207,207,207,207
340 DATA 128,128,128,128,207,207,207,207,207,207,207,207,207,207,1
43,143,207,207,143,143,143,143,143,206,206,206,206,207,207,207,207
,207,207,207,207,207,207,207,207,207
350 DATA 128,128,128,128,128,207,207,207,207,207,207,207,207,207,1
28,143,207,143,143,143,143,143,206,206,206,206,206,207,207,207,207
,207,207,207,207,143,143,143,143,143
360 DATA 128,128,128,128,128,207,207,207,143,207,207,207,207,207,2
07,128,143,143,143,143,143,143,206,206,206,206,206,207,207,207,207
,207,207,207,143,207,207,207,207,207
370 DATA 128,128,128,128,128,207,207,207,143,207,207,207,207,207,1
43,143,143,143,143,143,143,143,206,206,206,206,206,206,207,207,207
,207,207,207,207,207,207,207,207,207
380 DATA 128,128,128,128,128,143,207,143,207,207,207,207,207,143,1
43,143,143,143,143,143,143,143,206,206,206,206,206,206,207,207,207
,207,207,207,207,207,207,207,207,207
390 DATA 128,128,128,128,128,143,143,143,207,207,207,207,143,143,1
43,143,143,143,143,143,143,206,206,206,206,206,206,206,207,207,207
,207,207,207,207,207,207,207,207,207
400 DATA 128,128,128,128,128,207,143,207,207,207,207,143,143,143,1
43,143,143,143,143,143,206,206,206,206,206,206,206,128,207,207,207
,207,207,207,207,207,207,207,207,207
410 DATA 128,128,128,128,128,143,143,207,207,207,143,143,143,143,1
43,143,143,143,143,143,206,206,206,206,206,206,206,128,128,207,207
,207,207,207,207,207,207,207,207,207
420 DATA 128,128,128,128,128,143,207,207,143,143,143,143,143,143,1
43,143,143,143,143,206,206,206,206,206,206,206,206,128,128,128,207
,207,207,207,207,207,207,207,207,207
430 DATA 128,128,128,128,128,207,143,143,143,143,143,143,143,143,1
43,143,143,143,143,206,206,206,206,206,206,206,206,128,128,128,128
,207,207,207,207,207,207,207,207,207
440 '
450 GOTO 450
```

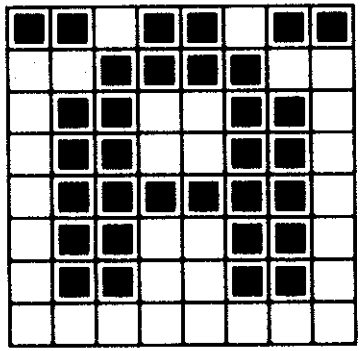

3.4 Der Zeichensatz-Editor

Bisher haben wir den Zeichensatz des CPC stets als etwas Statisches betrachtet, dessen Elemente im Inneren des Rechners festgelegt sind und an denen es nichts zu rütteln gibt. Daß es aber durchaus möglich ist, die einzelnen Charakter vollkommen umzugestalten, wollen wir Ihnen nun zusammen mit dem Handwerkszeug - wie man das macht - nahebringen.

Sie haben schon erfahren, daß sich ein einzelnes Zeichen aus acht mal acht Bildpunkten zusammensetzt, die jeder das Aussehen eines Zeichens entscheidend beeinflussen. Wir wollen Ihnen nun zeigen, wie sich ein Zeichen zu einem beliebigen anderen umdefinieren läßt. Nehmen sie zu diesem Zweck doch einmal ein Blatt kariertes Papier und einen Stift zur Hand.

Grenzen Sie mit dem Stift eine acht mal acht Felder große Fläche auf dem Papier ab. In diese Fläche hinein zeichnen Sie dann ein beliebiges Zeichen, wobei vollkommen unerheblich ist, ob es sich um ein erkennbares oder ein Phantasiezeichen handelt. Malen Sie dann alle diejenigen Kästchen aus, die für das Erscheinen des Zeichens wichtig sind. Auf diese Weise gewinnen Sie schon einen ersten Eindruck, wie dieses Zeichen später auf dem Bildschirm aussehen wird.

Der nächste Schritt ist dieses Zeichen in eine für den Computer verständliche Form zu bringen. Betrachten Sie zu diesem Zweck die einzelnen Zeilen der Matrix isoliert und schreiben neben jeden Zeile ein Reihe von Nullen und Einsen. Die Null steht dabei für ein nicht ausgefülltes und die Eins für ein ausgefülltes Kästchen. Haben Sie diese Arbeit für alle acht Zeilen der Zeichenmatrix richtig durchgeführt, so muß auf Ihrem Blatt etwas der Abbildung 5 entsprechendes zu sehen sein.



11011011
00111100
01100110
01100110
01111110
01100110
01100110
00000000

Abb. 5: Zeichendefinition per Hand

Nachdem das Zeichen nun zumindest auf dem Papier existiert, bleibt die Frage: Wie kommt das Zeichen in den Computer? Speziell für diese Aufgabe sind im Locomotive-BASIC die beiden Befehle

SYMBOL AFTER x

und

SYMBOL x,r1,r2,r3,r4,r5,r6,r7,r8

vorgesehen. Um die Funktionsweise dieser Befehle kennenzulernen, muß ein beliebiges Zeichen bestimmt werden, das durch die neue Zeichenmatrix ersetzt werden soll. In unserem Beispiel tauschen wir das Zeichen mit der Nummer 123 ({) gegen unsere Keration aus. Die Verfahrensweise ist denkbar einfach. Mit SYMBOL AFTER 123 wird die Umdefinition der Zeichen ab Nummer 123 ermöglicht; dann mit SYMBOL 123,r1,r2,r3,r4,r5, r6,r7,r8 die erstellte Matrix in den Zeichensatz integriert. Für r1 bis r8 sind, von oben nach unten abgelesen, die Kolonnen von Nullen und Einses einzusetzen, die auf dem Papier neben der Zeichenmatrix stehen. Achten Sie darauf, daß es sich um eine binäre Schreibweise handelt, die mit "&X" eingeleitet werden muß. Für unser Beispiel durchgeführt sieht das gerade beschriebene in Programmform gebracht folgendermaßen aus:


```
10 SYMBOL AFTER 123
20 SYMBOL 123,&X11011011,&X00111100,&X01100110,&X01100110,&X011111
10,&X01100110,&X01100110,&X00000000
```

Wenn dieses Programm durchgelaufen ist können Sie Ihr neu definiertes Zeichen ausprobieren, in dem Sie es mit PRINT CHR$(gewählte Zeichennummer) auf dem Bildschirm bringen!

Gewiß ist der hier beschrittene Weg einer Zeichenumdefinition nur einer von vielen, doch veranschaulicht er recht gut, wie viel Mühe es macht nur ein Zeichen zu generieren und dem Computer zugänglich zu machen. Wieviel mehr Arbeit die Gestaltung eines vollständigen Zeichensatzes macht, kann sich jeder an den Knöpfen seiner Hose abzählen.

Da Computer den Ruf genießen Arbeitsprozesse zu beschleunigen, liegt es geradezu auf der Hand, die zeitaufwendige Arbeit der Zeichenumdefinition dem CPC zu übertragen. Damit der CPC diese Aufgabe erledigen kann, muß ein Programm geschrieben werden, mit dem es möglich ist den Zeichensatz zu editieren. Welche Features muß ein solcher Zeichensatz-Editor haben?

1. Der Editor muß über ein Editierfeld verfügen, das mit dem acht mal acht Felder großen Bereich auf dem Rechenkästchenpapier vergleichbar ist. Es muß möglich sein eine Zeichenmatrix in dieses Feld hineingeladen und modifiziert zu können.

2. Das Programm muß einen Editiercursor haben, der mit dem Cursortasten innerhalb der Grenzen des Editierfeldes bewegt werden kann.

3. Zur Modifikation des Zeichens im Editierfeld muß es möglich sein an der gegenwärtigen Position des Editiercursors einen Bildpunkt zu setzen oder zu lö-

schen. Dieser Vorgang ist mit dem Ausmalen oder -radieren eines Kästchens auf dem Papier vergleichbar.

4. Es muß mit dem Editor möglich sein fertiggestellte Zeichen in den aktuellen Charaktersatz des CPC aufzunehmen und von diesem Zeitpunkt an dem Benutzer zur Verfügung zu stellen.

5. Damit einmal veränderte Zeichen auch für spätere Anwendungen konserviert werden können, muß es möglich sein die modivizierten Zeichen auf Diskette wegzuschreiben. Dabei sollen die Zeichen in Form eines lauffähigen Programmfiles gespeichert sein, das bei Bedarf zu anderen Programmteilen hinzugeladen werden kann, aber auch eigenständig lauffähig ist.

Auf der Grundlage dieser Forderungen haben wir für Sie einen Zeichnsatz-Editor geschrieben, dessen Handhabung wir zuerst erklären wollen, bevor das Programm abgedruckt wird.

Nach dem Starten des Programms mit RUN wird auf dem Bildschirm eine acht mal acht Charakter große Fläche umrandet, die sich später als Editierfeld erweisen wird. Rechts neben diesem Feld werden die Optionen des Programms zusammen mit ihren Anwahlmöglichkeiten ausgegeben. In der linken oberen Ecke (Defaultposition) des Editierfeldes erscheint der Editiercursor. Er hat die Gestalt eines marmoriert-ausgefüllten Charakters und läßt sich mit den Cusortasten innerhalb des Editierfeldes frei bewegen. Stößt er bei der Bewegung an die Umrandung des Feldes, so erklingt ein Ton, der anzeigt, daß die gewünschte Position unzulässig ist.

Akustische Fehleranzeigen erklingen immer dann, wenn dem Programm etwas abverlangt wird, was unsinnig oder unmöglich ist. So bewirkt ein Druck auf alle Tasten außer "N" und "Q", zu diesem Zeitpunkt, eine Fehlermeldung. Ein Druck auf die Taste "Q" beendet das Programm. Betätigt man hingegen die Taste "N" wie NEUES ZEICHEN, so fordert das Programm zur Eingabe einer Zeichennummer ein. Wurde ein Zahl eingegeben, mit

RETURN bestätigt und vom Programm aktzeptiert, d.h. es gab keine Fehlermeldung auf Grund von Bereichsüberschreitung, so gibt der Editor das der Eingabe entsprechende Zeichen in der linken oberen Ecke des Bildschirms aus. Von dieser Stelle aus wird es dann in vergrößerter Form im Editierfeld sichtbar gemacht und an der HOME-Position wieder gelöscht. Ist die Bedienung des Editors bis zu diesem Punkt fortgeschritten, so können alle angezeigten Optinen angewählt werden. Betrachten Sie die übrigen Optionen im Detail:

S: Durch einen Druck auf die Taste "S" wird der Bildpunkt an der momentanen Position des Editiercursors gesetzt.

L: Das Löschen des Bildpunktes an Cursorposition bewirkt hingegen das Betätigen des "L"-Taste.

I: Soll ein Zeichen, so wie es im Editierfeld sichtbar ist, in den Zeichensatz aufgenommen werden, so kann das mit dieser Option erreicht werden. Ist ein Zeichen implementiert worden, so darf man sich nicht wundern, wenn es plötzlich in mitten einer Ausgabe des Editor erscheint.

Q: Die Option PROGRAMM BEENDEN ist nicht so simpel, wie es auf den ersten Blick scheinen mag, denn sie beendet das Programm nur, wenn

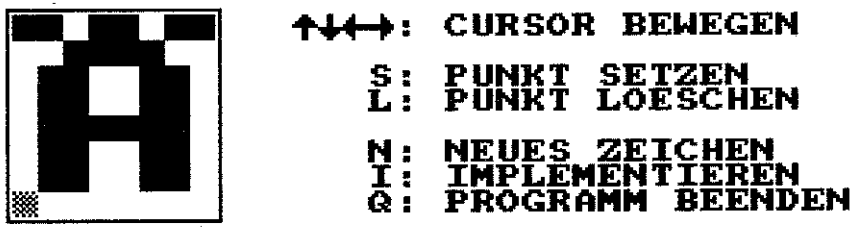
1. kein Zeichen zum Editieren ausgewählt wurde (Option N)

2. wenn keines der editierten Zeichen in den Zeichensatz aufgenommen wurde (Option I).

Wurde ein Zeichen in den Zeichensatz übernommen, so erscheint beim Druck auf die Taste "Q" die Frage: "WOLLEN SIE DIE EDITIERTEN ZEICHEN ALS PROGRAMMFILE AUF DISKETTE SICHER (J/N)". Wird diese Frage verneint, so bricht das Programm ab und stellt die umdefinierten Zeichen nur

im Speicher zur Verfügung - dort bleiben die Zeichen erhalten, bis der Speicher gelöscht oder überschrieben wird. Wird die Frage aber mit "J" beantwortet, so fordert das Programm zur Eingabe eines Dateinamen auf und schreibt die editierten Zeichen auf Diskette weg. Danach trifft das Programm auf END.

Als erste Anwendung für den Zeichensatz-Editor können Sie ja unser Beispiel aufgreifen, mit dem wir Ihnen die Definition eines Zeichen per Hand veranschaulicht haben. Nach dem Starten des Programms betätigen Sie die Taste "N" und geben als Zeichennummer, wie in unserem Beispiel, die Zahl 123 ein. Das Programm erstellt nun das Abbild einer geschweiften Klammer im Editierfeld; die geschweifte Klammer verbirgt sich normalerweise hinter diesem Zeichencode. Mit Cursortasten und den Optionen PUNKT SETZEN und PUNKT LOESCHEN erstellen Sie das hinlänglich bekannte "Ä". Nach abgeschlossener Definition sieht der Bildschirm aus wie auf Hardcopy 3 zu sehen.



Hardcopy 3: Definition eines "Ä"

Es folgt der Druck auf die "I"-Taste und damit die Aufnahme des "Ä" statt der "geschweiften Klammer auf" in den Zeichensatz des CPC. Wenn Sie das mit der Option PROGRAMM BEENDEN auf Diskette geschriebene Programmfile einlesen und mit LIST sichtbar machen, muß es folgendermaßen aussehen:

```
1 'DIES IST EIN AUTOMATISCH ERZEUGTES PROGRAMMFILE
2 SYMBOL AFTER 0
10 SYMBOL  123, 219, 60, 102, 102, 126, 102, 102, 0
```

Dieses Programm können Sie jederzeit einladen und laufenlassen, es wird stets ein "Ä" auf Ihren Rechner implementieren. Sie sehen, mit dem Zeichensatz-Editor ist es wesentlich einfacher, ein Zeichen zu entwerfen und zu modifizieren. Als besonders effektiv erweist sich die Arbeit mit dem Zeichensatz-Editor, wenn man viele Zeichen verändern will, was zahlreiche Anwendungen im Bereich der Grafikprogrammierung erfordern.

```
100 '************************************************************
110 '***                                                      ***
120 '***         ZEICHENSATZ-EDITOR  JST 21.8.1986            ***
130 '***                                                      ***
140 '************************************************************
150 '
160 MODE 1
170 SYMBOL AFTER 0
180 WINDOW#1,1,40,22,25
190 '
200 DIM ZEI(255,7)
210 DIM EZ(255)
220 '
230 CX=5
240 CY=9
250 '
260 'EDITIERFELD UMRANDEN
```

```
270 '
280 MOVE 60,141
290 DRAWR 134,0
300 DRAWR 0,134
310 DRAWR -134,0
320 DRAWR 0,-134
330 '
340 'MENUE AUSGEBEN
350 '
360 LOCATE 16,9:PRINT CHR$(240);CHR$(241);CHR$(242);CHR$(243);": C
URSOR BEWEGEN"
370 LOCATE 16,11:PRINT "   S: PUNKT SETZEN"
380 LOCATE 16,12:PRINT "   L: PUNKT LOESCHEN"
390 LOCATE 16,14:PRINT "   N: NEUES ZEICHEN"
400 LOCATE 16,15:PRINT "   I: IMPLEMENTIEREN"
410 LOCATE 16,16:PRINT "   Q: PROGRAMM BEENDEN"
420 '
430 GOSUB 650
440 '
450 'HAUPTSCHLEIFE
460 '
470 B=1
480 '
490 E$=UPPER$(INKEY$)
500 IF E$="" THEN 490
510 IF E$=CHR$(240) OR E$=CHR$(241) OR E$=CHR$(242) OR E$=CHR$(243
) THEN B=0:GOSUB 600
520 IF E$="S" OR E$="L" THEN B=0:GOSUB 850
530 IF E$="N" THEN B=0:GOSUB 1080
540 IF E$="I" THEN B=0:GOSUB 970
550 IF E$="Q" THEN B=0:GOSUB 1400
560 '
570 IF B=1 THEN PRINT CHR$(7);
580 '
590 GOTO 440
600 '
610 'CURSOR BEWEGEN MIT BUFFER
620 '
630 LOCATE CX,CY
640 PRINT BUFFER$;
```



```
650 '
660 'CURSOR BEWEGEN OHNE BUFFER
670 '
680 IF E$=CHR$(240) THEN CY=CY-1
690 IF E$=CHR$(241) THEN CY=CY+1
700 IF E$=CHR$(242) THEN CX=CX-1
710 IF E$=CHR$(243) THEN CX=CX+1
720 '
730 IF CX<5  THEN PRINT CHR$(7);:CX=5
740 IF CX>12 THEN PRINT CHR$(7);:CX=12
750 IF CY<9  THEN PRINT CHR$(7);:CY=9
760 IF CY>16 THEN PRINT CHR$(7);:CY=16
770 '
780 LOCATE CX,CY
790 BUFFER$=COPYCHR$(#0) [1]
800 '
810 LOCATE CX,CY
820 PRINT CHR$(207);
830 '
840 RETURN
850 '
860 'PUNKT SETZEN - PUNKT LOESCHEN
870 '
880 IF FLAG=0 THEN PRINT CHR$(7);:RETURN
890 '
900 ZEILE=7-(CY-9)
910 SUM=2^(8-(CX-4))
920 '
930 IF E$="S" AND BUFFER$=" " THEN BUFFER$=CHR$(143):ZEI(ZEICHEN,Z
EILE)=ZEI(ZEICHEN,ZEILE)+SUM
940 IF E$="L" AND BUFFER$=CHR$(143) THEN BUFFER$=" ":ZEI(ZEICHEN,Z
EILE)=ZEI(ZEICHEN,ZEILE)-SUM
```

---

1 Programmänderung für CPC 464:

```
10 '********
20 DATA &CD,&60,&BB,&32,&7D,&01,&C9,&00
30 FOR A=374 TO 381
40 READ D
50 POKE A,D
60 NEXT A
790 CALL 374:BUFFER$=CHR$(PEEK(381))
```

```
950 '
960 RETURN
970 '
980 'IMPLEMENTIEREN
990 '
1000 IF FLAG=0 THEN PRINT CHR$(7);:RETURN
1010 '
1020 SYMBOL ZEICHEN,ZEI(ZEICHEN,7),ZEI(ZEICHEN,6),ZEI(ZEICHEN,5),Z
EI(ZEICHEN,4),ZEI(ZEICHEN,3),ZEI(ZEICHEN,2),ZEI(ZEICHEN,1),ZEI(ZEI
CHEN,0)
1030 '
1040 ZZ=ZZ+1
1050 EZ(ZZ)=ZEICHEN
1060 '
1070 RETURN
1080 '
1090 'NEUES ZEICHEN
1100 '
1110 INPUT#1,"BITTE WAEHLEN SIE EIN ZEICHEN";ZEICHEN
1120 CLS#1
1130 '
1140 IF ZEICHEN<0 OR ZEICHEN>255 THEN PRINT CHR$(7);:GOTO 1110
1150 '
1160 IF ZEICHEN<32 THEN LOCATE 1,1:PRINT CHR$(1);CHR$(ZEICHEN);:GO
TO 1190
1170 LOCATE 1,1:PRINT CHR$(ZEICHEN);
1180 '
1190 FOR ZEILE=0 TO 7
1200 '
1210 ZEI(ZEICHEN,ZEILE)=128*TEST(1,385+ZEILE*2)+64*TEST(3,385+ZEIL
E*2)+32*TEST(5,385+ZEILE*2)+16*TEST(7,385+ZEILE*2)+8*TEST(9,385+ZE
ILE*2)+4*TEST(11,385+ZEILE*2)+2*TEST(13,385+ZEILE*2)+1*TEST(15,385
+ZEILE*2)
1220 '
1230 LOCATE 5,(8-ZEILE)+8
1240 '
1250 FOR I=7 TO 0 STEP -1
1260 IF (ZEI(ZEICHEN,ZEILE) AND 2^I)=2^I THEN PRINT CHR$(143); ELS
E PRINT " ";
1270 NEXT I
```



```
1280 PRINT
1290 '
1300 NEXT ZEILE
1310 '
1320 LOCATE 1,1
1330 PRINT " ";
1340 '
1350 GOSUB  650
1360 '
1370 FLAG=1
1380 '
1390 RETURN
1400 '
1410 'PROGRAMM BEENDEN
1420 '
1430 IF ZZ=0 THEN CLS:END
1440 '
1450 PRINT#1,"WOLLEN SIE DIE EDITIERTEN ZEICHEN ALS PROGRAMMFILE A
UF DISKETTE SICHER (J/N)"
1460 E$=INKEY$
1470 IF E$="" THEN 1460
1480 '
1490 IF UPPER$(E$)<>"J" THEN CLS:END
1500 '
1510 CLS#1
1520 INPUT#1,"GEBEN SIE EINEN DATEINAMEN EIN:";DATEI$
1530 IF LEN(DATEI$)>8 THEN 1510
1540 OPENOUT DATEI$
1550 '
1560 PRINT#9,"1 'DIES IST EIN AUTOMATISCH ERZEUGTES PROGRAMMFILE"
1570 PRINT#9,"2 SYMBOL AFTER 0"
1580 '
1590 FOR I=1 TO ZZ
1600 PRINT#9,STR$(I*10)+" SYMBOL "+STR$(EZ(I))+","+STR$(ZEI(EZ(I),
7))+","+STR$(ZEI(EZ(I),6))+","+STR$(ZEI(EZ(I),5))+","+STR$(ZEI(EZ(
I),4))+","+STR$(ZEI(EZ(I),3))+","+STR$(ZEI(EZ(I),2))+","+STR$(ZEI(
EZ(I),1))+","+STR$(ZEI(EZ(I),0))
1610 NEXT I
1620 '
1630 CLOSEOUT
```

```
1640 '
1650 CLS
1660 END
```

Programmbeschreibung:

100-240 Definitionen, Dimensionierungen und Vorbelegungen. Der Bildschirm wird in MODE 1 gebracht, die Umdefinition aller Zeichen ermöglicht (SYMBOL AFTER 0) und ein Dialogfenster (#1) definiert, das im Verlauf des Programms als Ein-/Ausgabeschnittstelle zwischen Rechner und Benutzer dienen soll.

Die indizierte Variable ZEI wird als zweidimensionales Feld mit 256 mal 8 Elementen dimensioniert. Diese Variable soll die Zeichenmatrizen aufnehmen, die während des Programmlaufs editiert werden. Erster Index ist die Nummer des betreffenden Zeichens, zweiter Index ist die Nummer der Zeile innerhalb einer Zeichenmatrix, wenn man sich diese als aus 7 bis 0 Zeilen zusammengesetzt vorstellt. Die auf diese Weise angesprochene Zahl liegt immer im Bereich 0-255. Sie spiegelt, in eine Dualzahl umgerechnet, die gesetzten und nicht gesetzten Pixel innerhalb der angewählten Zeile des gewünschten Zeichens wieder.

Die indizierte Variable EZ wird als Array mit 256 Elementen dimensioniert und soll die Nummer der editierten Zeichen aufnehmen. Die Dimensionierung auf 256 Elemente wurde gewählt, damit jedes der 256 Zeichen des CPC umdefiniert werden kann. Die Variable ZZ enthält im Verlauf des Programms den Index für das letzte belegte Feld in EZ(x).

Die Variablen CX und CY, die die aktuelle Position des Editiercursors enthalten, werden auf ihren Defaultwert gebracht (linke obere Ecke des Editierfeldes).

250-320 **Editierfeld unranden**
Eine Linie wird um die acht mal acht Charakter große Fläche gezeichnet, die später zum Editieren der Zeichen dienen wird.

330-410 **Menue ausgeben**
Neben dem Editierfeld wird die Liste der Programmoptionen und ihrer Anwahlmöglichkeit ausgegeben.

430 Der Editiercursor wird an die Defaultposition im Editierfeld gebracht (CURSOR BEWEGEN OHNE BUFFER).

440-590 **Hauptschleife**
Die Hauptschleife bildet die Schaltzentrale im Zeichensatzeditor; von ihr werden alle Verzweigungen in die Unterprogramme vorgenommen.

Zu Beginn der Hauptschleife wird die Variable B auf den Wert 1 gesetzt. Steht dieser Wert am Ende der Schleife unverändert in B, so wurde eine unzulässige Eingabe gemacht und, es erfolgt die Ausgabe eines Tones zur Fehleranzeige. Eine zulässige Eingabe setzt den Wert von B auf Null und verzweigt in das gewünschte Unterprogramm (GOSUB). Nach dem Rücksprung aus dem Unterprogramm wird die Hauptschleife erneut durchlaufen.

600-640 **Cursor bewegen mit Buffer**
Der Druck auf eine der vier Cursortasten veranlaßt die Hauptschleife, in dieses Unterprogramm zu verzweigen. An der aktuellen Position des Cursors wird der Inhalt der Variablen BUFFER$ ausgege-

ben. Die weitere Verarbeitung erfolgt über CURSOR BEWEGEN OHNE BUFFER.

650-840 **Cursor bewegen ohne Buffer**
Zunächst wird die Bewegungsrichtung des Cursor aus der gedrückten Cursortaste (E$) ermittelt und die Position des Cursors, die in den Variablen CX und CY steht, entsprechend neu bestimmt. Ergibt sich durch die Bewegung eine Cursorposition, die außerhalb des Editierfensters liegt, so erfolgt eine akustische Fehlermeldung. Die Position des Cursors wird automatisch auf den letzten zulässigen Wert zurückgesetzt. Das Zeichen an der ermittelten Position des Editiercursors wird aus dem Bildschirmspeicher gelesen und in der Variablen BUFFER$ gerettet (vgl. CURSOR BEWEGEN MIT BUFFER) - nur so ist eine Bewegung des Editiercursors möglich, ohne den Inhalt des Editierfeldes zu zerstören. Jetzt endlich erscheint der Cursor (CHR$(207)) an der ihm zugedachten Stelle im Editierfeld. Rücksprung!

850-960 **Punkt setzen - Punkt loeschen**
Ein Druck auf die Tasten "S" oder "L" hat die Hauptschleife dazu veranlaßt, die Verarbeitung an dieser Stelle das Programms weiterzuführen. Enthält die Variable FLAG den Wert 0, so wurde noch kein Zeichen ausgewählt, das editiert werden soll. Folglich ist es nicht sinnvoll, einen Punkt setzen oder löschen zu können - akustische Fehlermeldung und Rücksprung zur Hauptschleife.

Aus der aktuellen Zeilen-Position des Cursors (CY) wird ein Index errechnet (ZEILE), der den Zugriff auf die indizierte Variable ZEI(x,y), entsprechend der Bildschirmausgabe gestattet. Aus der Spalten-Position des Cursors (CX) wird, wiederum entsprechend der Darstellung auf dem Editierfeld, die Wertigkeit (SUM) des korrespondierenden Pixels der Zeichenmatrix an Cursorposition ermittelt.

Der nächste Schritt ist einfach: Soll der Punkt an Cursor-Position gesetzt werden (E$="S") und war er zuvor noch nicht gesetzt (BUFFER$=" "), dann wird die Variable BUFFER$ so manipuliert, daß sie bei der nächsten Bewegung des Cursors einen gesetzten Bildpunkt (CHR$(143)) hinterläßt. Des weiteren wird der Inhalt der Variablen ZEI, indiziert durch die Nummer des aktuellen Zeichens und den Index, der sich aus der aktuellen Zeilen-Position des Editiercursors ergibt (ZEILE), um die Wertigkeit des entsprechenden Pixels (SUM) erhöht. Analoges gilt für das Löschen eines Punktes.

970-1070 **Implementieren**
Das Betätigen der Taste "I" führte zum Aufruf dieses Unterprogramms, das unmittelbar, mit akustischer Fehlermeldung, wieder verlassen wird, wenn noch kein Zeichen zum Editieren ausgewählt wurde (FLAG=0).

Wurde ein Zeichen zum Editieren bestimmt, so enthält die Variable ZEICHEN die Nummer des aktuellen Zeichens. Mit Hilfe von ZEICHEN ist ein Zugriff auf ZEI(x,y) möglich und die aktuelle Zeichenmatrix kann unter ihrer Zeichennummer in den Charaktersatz des CPC aufgenommen werden (Zeile 1020). Ist dies erfolgt, so wird der Inhalt der Variablen ZZ inkrementiert, und EZ(ZZ) wird die Nummer des aktuellen Zeichens zugewiesen.

1080-1390 **Neues Zeichen**
Ein Druck auf die Taste "N" hat die Hauptschleife in das Unterprogramm NEUES ZEICHEN verzweigt. Hier wird nun über das Dialogfenster (#1) die Eingabe einer Zeichennummer erwartet. Ist die Eingabe nicht im zulässigen Bereich, so wird erneut zur Eingabe einer Zeichennummer aufgefordert. Ist der Inhalt der Variablen ZEICHEN im Bereich 0-255, also im zugelassenen Rahmen, so

müssen zunächst die nicht sofort darstellbaren Steuerzeichen (0-31) ausgesondert werden (Zeile 1160). Der Textcursor wird auf die Position 1,1 gebracht, die grafische Ausgabe von Steuerzeichen ermöglicht (vgl. CHR$(1)) und das Zeichen an dieser Stelle ausgegeben. Die Ausgabe herkömmlicher Zeichen (Zeichennummer größer 31) wird übersprungen. Hat die Differenzierung der Zeichen ergeben, daß es sich um kein Steuerzeichen handelt, so wird es in Zeile 1170 ausgegeben.

In der FOR-TO-NEXT-Schleife (Zeile 1190-1300) wird das aktuelle Zeichen an der Charakterposition 1,1 pixelweise aus dem Bildschirmspeicher gelesen und die Variable ZEI entsprechend ihrer Indizierung gefüllt. Die Zeilen 1250 bis 1270 haben ausschließlich die Aufgabe, die Vergrößerung des aktuellen Zeichens im Editierfeld zu erzeugen. Nach dem Durchlauf der Schleife wird das Zeichen in der linken oberen Ecke wieder gelöscht; es befindet sich nun sowohl in der Variablen ZEI(x,y) als auch in vergrößerter Form auf dem Bildschirm.

Vor dem Rücksprung in die Hauptschleife wird noch der Cursor in das Editierfeld gesetzt, ohne BUFFER$ auszugeben, und die Variable FLAG auf den Wert 1 gesetzt, was anzeigt, daß ein aktuelles Zeichen vorhanden ist.

1400-Ende **Programm beenden**
Wurde innerhalb der Hauptschleife die Taste "Q" betätigt, so wird das Unterprogramm PROGRAMM BEENDEN angesprungen (Vorsicht, aus diesem Programmteil gibt es keinen Rückweg in die Hauptschleife!).

Die erste Möglichkeit, daß das Programm ein Ende findet, ist, daß die Variable ZZ den Wert 0 hat, i.e. es wurde kein Zeichen neu in den Zeichensatz aufgenommen.

Die zweite Möglichkeit für ein Programmende ist eine andere Antwort als "J" auf die Frage, ob die editierten Zeichen als Programmfile auf Diskette gesichert werden sollen. Wurde mit einem Druck auf die Taste "J" geantwortet, so fragt der Rechner im Dialogfenster an, welchen Namen er dem zu erzeugenden Programmfile geben soll. Ist der Name ordnungsgemäß eingegeben, so wird mit ihm eine Ausgabedatei auf Diskette eröffnet.

In diese Datei wird zunächst der Kommentar geschrieben, daß es sich um ein automatisch generiertes Programmfile handelt, dann das Kommando SYMBOL AFTER 0, das eine Umdefinition des gesamten Zeichensatzes zuläßt. Ist dies geschehen, so werden alle umdefinierten Zeichen als vollständige BASIC-Programmzeilen, mit aufsteigender Statementnummer, SYMBOL-Kommando, Zeichennummer und zugehöriger Zeichenmatrix in die Ausgabe-Datei geschrieben. Das Zusammenspiel der Variablen ZZ und der indizierten Variablen EZ gewährleistet, daß ausschließlich die Zeichen auf Diskette gesichert werden, die auch tatsächlich umdefiniert wurden. Alle anderen bleiben unberücksichtigt.

Dritte und zugleich letzte Möglichkeit für ein Ende des Programms ist, daß die editierten Zeichen auf Diskette in Form eines Programmfiles gesichert wurden.

3.5 Aus "ae" wird "ä" - Umlaute für den CPC

Sicher wird es Ihnen ebenso wie uns stets ein Greul sein, bei Ihrem CPC auf Umlaute verzichten zu müssen. Statt der beiden Pünktchen über "a", "o" und "u" wird diesen Lauten dann ein "e" angehangen, um zumindest auf diese Weise den Eindruck eines Umlauts zu vermitteln. Uns befriedigte diese Notlösung auf

Dauer nicht! So erstellten wir, angelehnt an die vorhandenen Zeichen - Umlaute sollen sich schließlich harmonischen in den übrigen Zeichensatz einfügen - die Zeichenmatrizen für die Buchstaben Ä, Ö, Ü, ä, ö, ü und ß.

Da Umlaute im Deutschen keine Seltenheit sind, ist es erstrebenswert, diese genau wie andere Zeichen unmittelbar über die Tastatur in den Rechner tasten zu können, ohne umständlich mit CHR$-Codes zu jonglieren. Es müssen also drei Tasten auf dem Keyboard des CPC-Rechners gefunden werden, unter denen, sowohl im Basis- als auch im Shift-Modus, Zeichen versteckt sind, die für den normalen Betrieb entbehrlich sind. Ebenso muß ein Platz auf der Tastatur für das "ß" aufgetan werden. Hier reicht jedoch eine vakate Stelle, da das "ß" als Versalie nicht exisiert. Wir entschlossen uns, mit einem Auge auf den DIN-Standard schielend, zu folgenden Umdefinitionen:

Zeichennummer	Zeichen (alt)	Zeichen (neu)
64	@	ü
91	[	ä
93	]	ö
123	{	Ä
124	|	Ü
125	}	Ö
163	£	ß

Sicher stellt ein auf diese Weise umdefiniertes Keyboard ein Kuriosum dar, denn es handelt sich um eine amerikanische Tastatur (QWERTY) mit deutschen Umlauten. Auf ein Vertauschen der Buchstaben "Y" und "Z" haben wir aber bewußt verzichtet, um unnötige Verwirrung zu vermeiden. Wer dennoch eine richtige deutsche Tastatur haben möchte, kann sie jederzeit mit dem BASIC-Kommando KEY DEF erzeugen.

Zwei weitere Besonderheiten sind noch zu beachten: Durch einmaligen Druck auf die CAPS-LOCK-Taste werden alle mit Buchstaben belegten Tasten in den Shift-Modus versetzt, bis dies

durch erneutes Betätigen von CAPS-LOCK widerrufen wird. Diese Regelung gilt nicht für die Tasten, die von uns für die Umlaute ausgewählt wurden, da sie ja normalerweise Sonderzeichen statt Buchstaben tragen. Soll also ein großer Umlaut auf dem Bildschirm erscheinen, so muß immer eine der Shift-Tasten bemüht werden.

Die Erwähnung der Shift-Tasten bringt uns unmittelbar zur zweiten Besonderheit der geänderten Tastatur, dem "ß". Da dieser Buchstabe ein Kleinbuchstabe ist, sollte er wie andere Kleinbuchstaben im Basis-Modus der Tastatur erreichbar sein. Doch damit standen wir vor einem Dilemma - es gibt im Basis-Modus keine Taste mehr, die ein entbehrliches Zeichen beherbergt, es sein denn, das "ß" würde an eine vollkommen unübliche Stelle der Tastatur verbannt. Wir entschieden uns für einen Kompromiß, der Umdefinition des "£" zum "ß". Damit war der rechte Platz für diesen Laut gefunden, allerdings kann er nur über Shift erreicht werden.

```
1000 '************************************************************
1001 '***                                                      ***
1002 '***        UMLAUTE FUER DEN CPC  JST 2.6.1986            ***
1003 '***                                                      ***
1004 '************************************************************
1005 '
1006 SYMBOL AFTER 64
1007 '
1008 FOR I=0 TO 6
1009 '
1010 READ CH
1011 '
1012 FOR J=0 TO 7
1013 READ X
1014 TX(J)=X
1015 NEXT J
1016 '
1017 SYMBOL CH,TX(0),TX(1),TX(2),TX(3),TX(4),TX(5),TX(6),TX(7)
1018 '
1019 NEXT I
```

```
1020 '
1021 NEW
1022 '
1023 DATA 123
1024 '
1025 DATA &X11011011        '■■ ■■ ■■
1026 DATA &X00111100        '  ■■■■
1027 DATA &X01100110        ' ■■  ■■
1028 DATA &X01100110        ' ■■  ■■
1029 DATA &X01111110        ' ■■■■■■
1030 DATA &X01100110        ' ■■  ■■
1031 DATA &X01100110        ' ■■  ■■
1032 DATA &X00000000        '
1033 '
1034 DATA 125
1035 '
1036 DATA &X01100110        ' ■■  ■■
1037 DATA &X00111100        '  ■■■■
1038 DATA &X01100110        ' ■■  ■■
1039 DATA &X01100110        ' ■■  ■■
1040 DATA &X01100110        ' ■■  ■■
1041 DATA &X01100110        ' ■■  ■■
1042 DATA &X00111100        '  ■■■■
1043 DATA &X00000000        '
1044 '
1045 DATA 124
1046 '
1047 DATA &X01100110        ' ■■  ■■
1048 DATA &X00000000        '
1049 DATA &X01100110        ' ■■  ■■
1050 DATA &X01100110        ' ■■  ■■
1051 DATA &X01100110        ' ■■  ■■
1052 DATA &X01100110        ' ■■  ■■
1053 DATA &X00111100        '  ■■■■
1054 DATA &X00000000        '
1055 '
1056 DATA 91
1057 '
1058 DATA &X11000110        '■■   ■■
1059 DATA &X00000000        '
1060 DATA &X01111000        ' ■■■■
1061 DATA &X00001100        '    ■■
1062 DATA &X01111100        ' ■■■■■
1063 DATA &X11001100        '■■  ■■
1064 DATA &X01110110        ' ■■■ ■■
1065 DATA &X00000000        '
1066 '
```

1

1 Diese Kommentare zu den DATA-Statements dienen ausschließlich der besseren Übersicht des Lesers und können nicht eingetippt werden.


```
1067 DATA 93
1068 '
1069 DATA &X01100110        ' ■■  ■■
1070 DATA &X00000000        '
1071 DATA &X00111100        '  ■■■■
1072 DATA &X01100110        ' ■■  ■■
1073 DATA &X01100110        ' ■■  ■■
1074 DATA &X01100110        ' ■■  ■■
1075 DATA &X00111100        '  ■■■■
1076 DATA &X00000000        '
1077 '
1078 DATA 64
1079 '
1080 DATA &X00000000        '
1081 DATA &X01100110        ' ■■  ■■
1082 DATA &X00000000        '
1083 DATA &X01100110        ' ■■  ■■
1084 DATA &X01100110        ' ■■  ■■
1085 DATA &X01100110        ' ■■  ■■
1086 DATA &X00111110        '  ■■■■■
1087 DATA &X00000000        '
1088 '
1089 DATA 163
1090 '
1091 DATA &X01111100        ' ■■■■■
1092 DATA &X11000110        '■■   ■■
1093 DATA &X11000110        '■■   ■■
1094 DATA &X11111100        '■■■■■■
1095 DATA &X11000110        '■■   ■■
1096 DATA &X11000110        '■■   ■■
1097 DATA &X11111000        '■■■■■
1098 DATA &X11000000        '■■
```

Programmbeschreibung:

Auf den Programmkopf folgend wird in Zeile 1006 mit dem BASIC-Kommando SYMBOL AFTER 64 die Umdefinition der Zeichen ab Zeichen Nummer 64 ermöglicht. Von Zeile 1008 bis Zeile 1019 reicht eine FOR-TO-NEXT-Schleife, in der die Zeichen "ÄÖÜäöüß" in den Zeichensatz aufgenommen werden. Genauer betrachtet, wird in der Schleife zuerst ein DATA-Statement ausgelesen, das die Zeichennummer enthält. In einer zweiten FOR-TO-NEXT-Schleife werden dann die acht Zahlenwerte, in denen die Zeichenmatrix enthalten ist, wiederum aus DATA-Statements ausgelesen und den indizierten Variablen TX(0) bis TX(7) zugewiesen. Unter der aktuellen Zeichennummer (CH) wird schließlich die Matrix in den Zeichensatz aufge-

nommen (Zeile 1017). In gleicher Weise wird mit allen sieben Zeichen verfahren. Ist die Umdefinition abgeschlossen, löscht sich das Programm im Speicher (Zeile 1021)!

Von Zeile 1022 bis zum Ende des Programms stehen DATA-Statements, von denen jeweils neun eine Gruppe bilden. Im ersten DATA einer solchen Gruppe steht die Nummer des Zeichens, das umdefiniert werden soll. Die darauf folgenden acht DATAs enthalten die Zeichenmatrix. Bei allen sieben Neunergruppen wurden die Zeichenmatrizen zur besseren Transparenz in binärer Darstellung abgedruckt. Die Kommentare zu den DATA-Statements dienen ausschließlich der besseren Übersicht des Lesers und können nicht eingetippt werden.

3.6 Ein alternativer Zeichensatz für den CPC

Die CPCs zeichnen sich unter anderem dadurch aus, daß sie 80 Zeichen auf ihrem Bildschirm darstellen können. Diese Tatsache verleiht dem Homecomputer CPC einen Hauch von Professionalität, der aber durch die Tatsache geschmälert wird, daß sich die Zeichen des integrierten Charaktergenerators in MODE 2 nicht ermüdungsfrei lesen lassen. Nutzt man seinen CPC für professionelle Anwendungen, wie beispielsweise Textverarbeitung oder Programmieren, so sehnt man sich schnell nach einem Zeichensatz, wie er bei guten Textterminals anzutreffen ist.

Solche Zeichensätze bestechen geradezu durch ihre Einfachheit - so sind senkrechte wie waagerechte Linienzüge in der Regel aus einzelnen Pixeln zusammengesetzt. Einfache Linienelemente können zwar, besonders bei Farbmonitoren, zu Problemen bei der Wiedergabe führen, doch hat sich in der Praxis erwiesen, daß es sehr stark davon abhängt welche, Farbkombination gerade eingestellt ist.

Beim Schreiben aller Programme, die etwas größeren Zeitaufwand erforderten, haben wir grundsätzlich mit dem unten abgedruckten Zeichensatz gearbeitet, wobei sich die Farbzusammenstellung


```
BORDER 0
INK 0,0
INK 1,11
```

als besonders gut im 80-Zeichenmodus lesbar erwiesen hat.

Wenn Sie auch gerne mit unserem alternativen Zeichensatz arbeiten möchten, so haben Sie die Wahl, das Programm in der abgedruckten Form einzutippen, oder mit dem Hilfsmittel Zeichensatz-Editor die extra für diesen Zweck in Kommentarform neben den Matrizen stehenden Zeichen nachzuempfinden.

```
1000 '************************************************************
1001 '***                                                      ***
1002 '***     EIN ALTERNATIVER ZEICHENSATZ FUER DEN CPC        ***
1003 '***                  JST/TAV 2.6.1986                    ***
1004 '***                                                      ***
1005 '************************************************************
1006 '
1007 SYMBOL AFTER 33
1008 '
1009 FOR I=33 TO 125
1010 '
1011 FOR J=0 TO 7
1012 READ X
1013 TX(J)=X
1014 NEXT J
1015 '
1016 SYMBOL I,TX(0),TX(1),TX(2),TX(3),TX(4),TX(5),TX(6),TX(7)
1017 '
1018 NEXT I
1019 '
1020 NEW
1021 '
```

```
1022 '!
1023 '                                  1
1024 DATA &X00001000        '     ■
1025 DATA &X00001000        '     ■
1026 DATA &X00001000        '     ■
1027 DATA &X00001000        '     ■
1028 DATA &X00001000        '     ■
1029 DATA &X00000000        '
1030 DATA &X00001000        '     ■
1031 DATA &X00000000        '
1032 '
1033 '"
1034 '
1035 DATA &X00100100        '   ■  ■
1036 DATA &X00100100        '   ■  ■
1037 DATA &X00100100        '   ■  ■
1038 DATA &X00000000        '
1039 DATA &X00000000        '
1040 DATA &X00000000        '
1041 DATA &X00000000        '
1042 DATA &X00000000        '
1043 '
1044 '#
1045 '
1046 DATA &X00100100        '   ■  ■
1047 DATA &X00100100        '   ■  ■
1048 DATA &X01111110        '  ■■■■■■
1049 DATA &X00100100        '   ■  ■
1050 DATA &X01111110        '  ■■■■■■
1051 DATA &X00100100        '   ■  ■
1052 DATA &X00100100        '   ■  ■
1053 DATA &X00000000        '
1054 '
1055 '$
1056 '
1057 DATA &X00001000        '     ■
1058 DATA &X00011110        '    ■■■■
1059 DATA &X00100000        '   ■
1060 DATA &X00011100        '    ■■■
1061 DATA &X00000010        '      ■
1062 DATA &X00111100        '   ■■■■
1063 DATA &X00001000        '     ■
1064 DATA &X00000000        '
1065 '
1066 '%
1067 '
1068 DATA &X00000000        '
1069 DATA &X01100010        '  ■■   ■
1070 DATA &X01100100        '  ■■  ■
1071 DATA &X00001000        '     ■
1072 DATA &X00010000        '    ■
1073 DATA &X00100110        '   ■  ■■
1074 DATA &X01000110        '  ■   ■■
1075 DATA &X00000000        '
1076 '
```

1 Diese Kommentare zu den DATA-Statements dienen ausschließlich der besseren Übersicht des Lesers und können nicht eingetippt werden.


```
1077 '&
1078 '
1079 DATA &X00110000        '   ■■
1080 DATA &X01001000        '  ■  ■
1081 DATA &X01001000        '  ■  ■
1082 DATA &X00110000        '   ■■
1083 DATA &X01001010        '  ■  ■ ■
1084 DATA &X01000100        '  ■   ■
1085 DATA &X00111010        '   ■■■ ■
1086 DATA &X00000000        '
1087 '
1088 ''
1089 '
1090 DATA &X00001000        '     ■
1091 DATA &X00010000        '    ■
1092 DATA &X00100000        '   ■
1093 DATA &X00000000        '
1094 DATA &X00000000        '
1095 DATA &X00000000        '
1096 DATA &X00000000        '
1097 DATA &X00000000        '
1098 '
1099 '(
1100 '
1101 DATA &X00000100        '       ■
1102 DATA &X00001000        '     ■
1103 DATA &X00010000        '    ■
1104 DATA &X00010000        '    ■
1105 DATA &X00010000        '    ■
1106 DATA &X00001000        '     ■
1107 DATA &X00000100        '       ■
1108 DATA &X00000000        '
1109 '
1110 ')
1111 '
1112 DATA &X00100000        '   ■
1113 DATA &X00010000        '    ■
1114 DATA &X00001000        '     ■
1115 DATA &X00001000        '     ■
1116 DATA &X00001000        '     ■
1117 DATA &X00010000        '    ■
1118 DATA &X00100000        '   ■
1119 DATA &X00000000        '
1120 '
1121 '*
1122 '
1123 DATA &X00001000        '     ■
1124 DATA &X00101010        '   ■ ■ ■
1125 DATA &X00011100        '    ■■■
1126 DATA &X00111110        '   ■■■■■
1127 DATA &X00011100        '    ■■■
1128 DATA &X00101010        '   ■ ■ ■
1129 DATA &X00001000        '     ■
1130 DATA &X00000000        '
1131 '
```

```
1132 '+
1133 '
1134 DATA &X00000000        '
1135 DATA &X00001000        '
1136 DATA &X00001000        '
1137 DATA &X00111110        '
1138 DATA &X00001000        '
1139 DATA &X00001000        '
1140 DATA &X00000000        '
1141 DATA &X00000000        '
1142 '
1143 ',
1144 '
1145 DATA &X00000000        '
1146 DATA &X00000000        '
1147 DATA &X00000000        '
1148 DATA &X00000000        '
1149 DATA &X00000000        '
1150 DATA &X00001000        '
1151 DATA &X00001000        '
1152 DATA &X00010000        '
1153 '
1154 '-
1155 '
1156 DATA &X00000000        '
1157 DATA &X00000000        '
1158 DATA &X00000000        '
1159 DATA &X01111110        '
1160 DATA &X00000000        '
1161 DATA &X00000000        '
1162 DATA &X00000000        '
1163 DATA &X00000000        '
1164 '
1165 '.
1166 '
1167 DATA &X00000000        '
1168 DATA &X00000000        '
1169 DATA &X00000000        '
1170 DATA &X00000000        '
1171 DATA &X00000000        '
1172 DATA &X00000000        '
1173 DATA &X00001000        '
1174 DATA &X00000000        '
1175 '
1176 '/
1177 '
1178 DATA &X00000000        '
1179 DATA &X00000010        '
1180 DATA &X00000100        '
1181 DATA &X00001000        '
1182 DATA &X00010000        '
1183 DATA &X00100000        '
1184 DATA &X01000000        '
1185 DATA &X00000000        '
1186 '
```



```
1187 '0
1188 '
1189 DATA &X00111000        '
1190 DATA &X01000100        '
1191 DATA &X01001100        '
1192 DATA &X01010100        '
1193 DATA &X01100100        '
1194 DATA &X01000100        '
1195 DATA &X00111000        '
1196 DATA &X00000000        '
1197 '
1198 '1
1199 '
1200 DATA &X00010000        '
1201 DATA &X00110000        '
1202 DATA &X01010000        '
1203 DATA &X00010000        '
1204 DATA &X00010000        '
1205 DATA &X00010000        '
1206 DATA &X01111100        '
1207 DATA &X00000000        '
1208 '
1209 '2
1210 '
1211 DATA &X00111100        '
1212 DATA &X01000010        '
1213 DATA &X00000010        '
1214 DATA &X00001100        '
1215 DATA &X00110000        '
1216 DATA &X01000000        '
1217 DATA &X01111110        '
1218 DATA &X00000000        '
1219 '
1220 '3
1221 '
1222 DATA &X00111100        '
1223 DATA &X01000010        '
1224 DATA &X00000010        '
1225 DATA &X00011100        '
1226 DATA &X00000010        '
1227 DATA &X01000010        '
1228 DATA &X00111100        '
1229 DATA &X00000000        '
1230 '
1231 '4
1232 '
1233 DATA &X00000100        '
1234 DATA &X00001100        '
1235 DATA &X00010100        '
1236 DATA &X00100100        '
1237 DATA &X01111110        '
1238 DATA &X00000100        '
1239 DATA &X00000100        '
1240 DATA &X00000000        '
1241 '
```

```
1242 '5
1243 '
1244 DATA &X01111110        '  ******
1245 DATA &X01000000        '  *
1246 DATA &X01111000        '  ****
1247 DATA &X00000100        '      *
1248 DATA &X00000010        '       *
1249 DATA &X01000100        '  *    *
1250 DATA &X00111000        '   ***
1251 DATA &X00000000        '
1252 '
1253 '6
1254 '
1255 DATA &X00011100        '    ***
1256 DATA &X00100000        '   *
1257 DATA &X01000000        '  *
1258 DATA &X01111100        '  *****
1259 DATA &X01000010        '  *    *
1260 DATA &X01000010        '  *    *
1261 DATA &X00111100        '   ****
1262 DATA &X00000000        '
1263 '
1264 '7
1265 '
1266 DATA &X01111110        '  ******
1267 DATA &X01000010        '  *    *
1268 DATA &X00000100        '      *
1269 DATA &X00001000        '     *
1270 DATA &X00001000        '     *
1271 DATA &X00001000        '     *
1272 DATA &X00001000        '     *
1273 DATA &X00000000        '
1274 '
1275 '8
1276 '
1277 DATA &X00111100        '   ****
1278 DATA &X01000010        '  *    *
1279 DATA &X01000010        '  *    *
1280 DATA &X00111100        '   ****
1281 DATA &X01000010        '  *    *
1282 DATA &X01000010        '  *    *
1283 DATA &X00111100        '   ****
1284 DATA &X00000000        '
1285 '
1286 '9
1287 '
1288 DATA &X00111100        '   ****
1289 DATA &X01000010        '  *    *
1290 DATA &X01000010        '  *    *
1291 DATA &X00111110        '   *****
1292 DATA &X00000010        '       *
1293 DATA &X00000100        '      *
1294 DATA &X00111000        '   ***
1295 DATA &X00000000        '
1296 '
```



```
1297 ':
1298 '
1299 DATA &X00000000        '
1300 DATA &X00000000        '
1301 DATA &X00001000        '     *
1302 DATA &X00000000        '
1303 DATA &X00000000        '
1304 DATA &X00001000        '     *
1305 DATA &X00000000        '
1306 DATA &X00000000        '
1307 '
1308 ';
1309 '
1310 DATA &X00000000        '
1311 DATA &X00000000        '
1312 DATA &X00001000        '     *
1313 DATA &X00000000        '
1314 DATA &X00000000        '
1315 DATA &X00001000        '     *
1316 DATA &X00001000        '     *
1317 DATA &X00010000        '    *
1318 '
1319 '<
1320 '
1321 DATA &X00000100        '      *
1322 DATA &X00001000        '     *
1323 DATA &X00010000        '    *
1324 DATA &X00100000        '   *
1325 DATA &X00010000        '    *
1326 DATA &X00001000        '     *
1327 DATA &X00000100        '      *
1328 DATA &X00000000        '
1329 '
1330 '=
1331 '
1332 DATA &X00000000        '
1333 DATA &X00000000        '
1334 DATA &X01111110        '  ******
1335 DATA &X00000000        '
1336 DATA &X01111110        '  ******
1337 DATA &X00000000        '
1338 DATA &X00000000        '
1339 DATA &X00000000        '
1340 '
1341 '>
1342 '
1343 DATA &X00100000        '   *
1344 DATA &X00010000        '    *
1345 DATA &X00001000        '     *
1346 DATA &X00000100        '      *
1347 DATA &X00001000        '     *
1348 DATA &X00010000        '    *
1349 DATA &X00100000        '   *
1350 DATA &X00000000        '
1351 '
```

```
1352 '?
1353 '
1354 DATA &X00111100        '   ####
1355 DATA &X01000010        '  #    #
1356 DATA &X00000010        '       #
1357 DATA &X00001100        '     ##
1358 DATA &X00010000        '    #
1359 DATA &X00000000        '
1360 DATA &X00010000        '    #
1361 DATA &X00000000        '
1362 '
1363 '@
1364 '
1365 DATA &X00011100        '   ###
1366 DATA &X00100010        '  #   #
1367 DATA &X01001010        ' #  # #
1368 DATA &X01010110        ' # # ##
1369 DATA &X01001100        ' #  ##
1370 DATA &X00100000        '   #
1371 DATA &X00011100        '   ###
1372 DATA &X00000000        '
1373 '
1374 'A
1375 '
1376 DATA &X00011000        '   ##
1377 DATA &X00100100        '  #  #
1378 DATA &X01000010        ' #    #
1379 DATA &X01111110        ' ######
1380 DATA &X01000010        ' #    #
1381 DATA &X01000010        ' #    #
1382 DATA &X01000010        ' #    #
1383 DATA &X00000000        '
1384 '
1385 'B
1386 '
1387 DATA &X01111100        ' #####
1388 DATA &X00100010        '  #   #
1389 DATA &X00100010        '  #   #
1390 DATA &X00111100        '  ####
1391 DATA &X00100010        '  #   #
1392 DATA &X00100010        '  #   #
1393 DATA &X01111100        ' #####
1394 DATA &X00000000        '
1395 '
1396 'C
1397 '
1398 DATA &X00011100        '   ###
1399 DATA &X00100010        '  #   #
1400 DATA &X01000000        ' #
1401 DATA &X01000000        ' #
1402 DATA &X01000000        ' #
1403 DATA &X00100010        '  #   #
1404 DATA &X00011100        '   ###
1405 DATA &X00000000        '
1406 '
```



```
1407 'D
1408 '
1409 DATA &X01111000        ' ####
1410 DATA &X00100100        '  #  #
1411 DATA &X00100010        '  #   #
1412 DATA &X00100010        '  #   #
1413 DATA &X00100010        '  #   #
1414 DATA &X00100100        '  #  #
1415 DATA &X01111000        ' ####
1416 DATA &X00000000        '
1417 '
1418 'E
1419 '
1420 DATA &X01111110        ' ######
1421 DATA &X01000000        ' #
1422 DATA &X01000000        ' #
1423 DATA &X01111100        ' #####
1424 DATA &X01000000        ' #
1425 DATA &X01000000        ' #
1426 DATA &X01111110        ' ######
1427 DATA &X00000000        '
1428 '
1429 'F
1430 '
1431 DATA &X01111110        ' ######
1432 DATA &X01000000        ' #
1433 DATA &X01000000        ' #
1434 DATA &X01111100        ' #####
1435 DATA &X01000000        ' #
1436 DATA &X01000000        ' #
1437 DATA &X01000000        ' #
1438 DATA &X00000000        '
1439 '
1440 'G
1441 '
1442 DATA &X00011100        '   ###
1443 DATA &X00100010        '  #   #
1444 DATA &X01000000        ' #
1445 DATA &X01001110        ' #  ###
1446 DATA &X01000010        ' #    #
1447 DATA &X00100010        '  #   #
1448 DATA &X00011100        '   ###
1449 DATA &X00000000        '
1450 '
1451 'H
1452 '
1453 DATA &X01000010        ' #    #
1454 DATA &X01000010        ' #    #
1455 DATA &X01000010        ' #    #
1456 DATA &X01111110        ' ######
1457 DATA &X01000010        ' #    #
1458 DATA &X01000010        ' #    #
1459 DATA &X01000010        ' #    #
1460 DATA &X00000000        '
1461 '
```

```
1462 'I
1463 '
1464 DATA &X00011100        '   ###
1465 DATA &X00001000        '    #
1466 DATA &X00001000        '    #
1467 DATA &X00001000        '    #
1468 DATA &X00001000        '    #
1469 DATA &X00001000        '    #
1470 DATA &X00011100        '   ###
1471 DATA &X00000000        '
1472 '
1473 'J
1474 '
1475 DATA &X00001110        '    ###
1476 DATA &X00000100        '     #
1477 DATA &X00000100        '     #
1478 DATA &X00000100        '     #
1479 DATA &X00000100        '     #
1480 DATA &X01000100        ' #   #
1481 DATA &X00111000        '  ###
1482 DATA &X00000000        '
1483 '
1484 'K
1485 '
1486 DATA &X01000010        ' #    #
1487 DATA &X01000100        ' #   #
1488 DATA &X01001000        ' #  #
1489 DATA &X01110000        ' ###
1490 DATA &X01001000        ' #  #
1491 DATA &X01000100        ' #   #
1492 DATA &X01000010        ' #    #
1493 DATA &X00000000        '
1494 '
1495 'L
1496 '
1497 DATA &X01000000        ' #
1498 DATA &X01000000        ' #
1499 DATA &X01000000        ' #
1500 DATA &X01000000        ' #
1501 DATA &X01000000        ' #
1502 DATA &X01000000        ' #
1503 DATA &X01111110        ' ######
1504 DATA &X00000000        '
1505 '
1506 'M
1507 '
1508 DATA &X01000010        ' #    #
1509 DATA &X01100110        ' ##  ##
1510 DATA &X01011010        ' # ## #
1511 DATA &X01011010        ' # ## #
1512 DATA &X01000010        ' #    #
1513 DATA &X01000010        ' #    #
1514 DATA &X01000010        ' #    #
1515 DATA &X00000000        '
1516 '
```



```
1517 'N
1518 '
1519 DATA &X01000010        ' #    #
1520 DATA &X01100010        ' ##   #
1521 DATA &X01010010        ' # #  #
1522 DATA &X01001010        ' #  # #
1523 DATA &X01000110        ' #   ##
1524 DATA &X01000010        ' #    #
1525 DATA &X01000010        ' #    #
1526 DATA &X00000000        '
1527 '
1528 'O
1529 '
1530 DATA &X00111100        '  ####
1531 DATA &X01000010        ' #    #
1532 DATA &X01000010        ' #    #
1533 DATA &X01000010        ' #    #
1534 DATA &X01000010        ' #    #
1535 DATA &X01000010        ' #    #
1536 DATA &X00111100        '  ####
1537 DATA &X00000000        '
1538 '
1539 'P
1540 '
1541 DATA &X01111100        ' #####
1542 DATA &X01000010        ' #    #
1543 DATA &X01000010        ' #    #
1544 DATA &X01111100        ' #####
1545 DATA &X01000000        ' #
1546 DATA &X01000000        ' #
1547 DATA &X01000000        ' #
1548 DATA &X00000000        '
1549 '
1550 'Q
1551 '
1552 DATA &X00111100        '  ####
1553 DATA &X01000010        ' #    #
1554 DATA &X01000010        ' #    #
1555 DATA &X01000010        ' #  # #
1556 DATA &X01001010        ' #  # #
1557 DATA &X01000100        ' #   #
1558 DATA &X00111010        '  ### #
1559 DATA &X00000000        '
1560 '
1561 'R
1562 '
1563 DATA &X01111100        ' #####
1564 DATA &X01000010        ' #    #
1565 DATA &X01000010        ' #    #
1566 DATA &X01111100        ' #####
1567 DATA &X01001000        ' #  #
1568 DATA &X01000100        ' #   #
1569 DATA &X01000010        ' #    #
1570 DATA &X00000000        '
1571 '
```

```
1572 'S
1573 '
1574 DATA &X00111100        '  ####
1575 DATA &X01000010        ' #    #
1576 DATA &X01000000        ' #
1577 DATA &X00111100        '  ####
1578 DATA &X00000010        '       #
1579 DATA &X01000010        ' #     #
1580 DATA &X00111100        '  ####
1581 DATA &X00000000        '
1582 '
1583 'T
1584 '
1585 DATA &X00111110        '  #####
1586 DATA &X00001000        '    #
1587 DATA &X00001000        '    #
1588 DATA &X00001000        '    #
1589 DATA &X00001000        '    #
1590 DATA &X00001000        '    #
1591 DATA &X00001000        '    #
1592 DATA &X00000000        '
1593 '
1594 'U
1595 '
1596 DATA &X01000010        ' #    #
1597 DATA &X01000010        ' #    #
1598 DATA &X01000010        ' #    #
1599 DATA &X01000010        ' #    #
1600 DATA &X01000010        ' #    #
1601 DATA &X01000010        ' #    #
1602 DATA &X00111100        '  ####
1603 DATA &X00000000        '
1604 '
1605 'V
1606 '
1607 DATA &X01000010        ' #    #
1608 DATA &X01000010        ' #    #
1609 DATA &X01000010        ' #    #
1610 DATA &X00100100        '  #  #
1611 DATA &X00100100        '  #  #
1612 DATA &X00011000        '   ##
1613 DATA &X00011000        '   ##
1614 DATA &X00000000        '
1615 '
1616 'W
1617 '
1618 DATA &X01000010        ' #    #
1619 DATA &X01000010        ' #    #
1620 DATA &X01000010        ' #    #
1621 DATA &X01011010        ' # ## #
1622 DATA &X01011010        ' # ## #
1623 DATA &X01100110        ' ##  ##
1624 DATA &X01000010        ' #    #
1625 DATA &X00000000        '
1626 '
```



```
1627 'X
1628 '
1629 DATA &X01000010        ' #    #
1630 DATA &X01000010        ' #    #
1631 DATA &X00100100        '  #  #
1632 DATA &X00011000        '   ##
1633 DATA &X00100100        '  #  #
1634 DATA &X01000010        ' #    #
1635 DATA &X01000010        ' #    #
1636 DATA &X00000000        '
1637 '
1638 'Y
1639 '
1640 DATA &X00100010        '  #   #
1641 DATA &X00100010        '  #   #
1642 DATA &X00100010        '  #   #
1643 DATA &X00011100        '   ###
1644 DATA &X00001000        '    #
1645 DATA &X00001000        '    #
1646 DATA &X00001000        '    #
1647 DATA &X00000000        '
1648 '
1649 'Z
1650 '
1651 DATA &X01111110        ' ######
1652 DATA &X00000010        '      #
1653 DATA &X00000100        '     #
1654 DATA &X00011000        '   ##
1655 DATA &X00100000        '  #
1656 DATA &X01000000        ' #
1657 DATA &X01111110        ' ######
1658 DATA &X00000000        '
1659 '
1660 '[
1661 '
1662 DATA &X00011100        '   ###
1663 DATA &X00010000        '   #
1664 DATA &X00010000        '   #
1665 DATA &X00010000        '   #
1666 DATA &X00010000        '   #
1667 DATA &X00010000        '   #
1668 DATA &X00011100        '   ###
1669 DATA &X00000000        '
1670 '
1671 '\
1672 '
1673 DATA &X00000000        '
1674 DATA &X01000000        ' #
1675 DATA &X00100000        '  #
1676 DATA &X00010000        '   #
1677 DATA &X00001000        '    #
1678 DATA &X00000100        '     #
1679 DATA &X00000010        '      #
1680 DATA &X00000000        '
1681 '
```

```
1682 ']
1683 '
1684 DATA &X00111000        '   ###
1685 DATA &X00001000        '      #
1686 DATA &X00001000        '      #
1687 DATA &X00001000        '      #
1688 DATA &X00001000        '      #
1689 DATA &X00001000        '      #
1690 DATA &X00111000        '   ###
1691 DATA &X00000000        '
1692 '
1693 '^
1694 '
1695 DATA &X00001000        '      #
1696 DATA &X00011100        '    ###
1697 DATA &X00101010        '   # # #
1698 DATA &X00001000        '      #
1699 DATA &X00001000        '      #
1700 DATA &X00001000        '      #
1701 DATA &X00001000        '      #
1702 DATA &X00000000        '
1703 '
1704 '_
1705 '
1706 DATA &X00000000        '
1707 DATA &X00000000        '
1708 DATA &X00000000        '
1709 DATA &X00000000        '
1710 DATA &X00000000        '
1711 DATA &X00000000        '
1712 DATA &X00000000        '
1713 DATA &X11111111        '########
1714 '
1715 ''
1716 '
1717 DATA &X00010000        '    #
1718 DATA &X00001000        '      #
1719 DATA &X00000100        '       #
1720 DATA &X00000000        '
1721 DATA &X00000000        '
1722 DATA &X00000000        '
1723 DATA &X00000000        '
1724 DATA &X00000000        '
1725 '
1726 'a
1727 '
1728 DATA &X00000000        '
1729 DATA &X00000000        '
1730 DATA &X00111000        '   ###
1731 DATA &X00000100        '       #
1732 DATA &X00111100        '   ####
1733 DATA &X01000100        ' #   #
1734 DATA &X00111010        '   ### #
1735 DATA &X00000000        '
1736 '
```



```
1737 'b
1738 '
1739 DATA &X01000000        ' #
1740 DATA &X01000000        ' #
1741 DATA &X01011100        ' # ###
1742 DATA &X01100010        ' ##   #
1743 DATA &X01000010        ' #    #
1744 DATA &X01100010        ' ##   #
1745 DATA &X01011100        ' # ###
1746 DATA &X00000000        '
1747 '
1748 'c
1749 '
1750 DATA &X00000000        '
1751 DATA &X00000000        '
1752 DATA &X00111100        '   ####
1753 DATA &X01000010        ' #    #
1754 DATA &X01000000        ' #
1755 DATA &X01000010        ' #    #
1756 DATA &X00111100        '   ####
1757 DATA &X00000000        '
1758 '
1759 'd
1760 '
1761 DATA &X00000010        '       #
1762 DATA &X00000010        '       #
1763 DATA &X00111010        '   ### #
1764 DATA &X01000110        ' #   ##
1765 DATA &X01000010        ' #    #
1766 DATA &X01000110        ' #   ##
1767 DATA &X00111010        '   ### #
1768 DATA &X00000000        '
1769 '
1770 'e
1771 '
1772 DATA &X00000000        '
1773 DATA &X00000000        '
1774 DATA &X00111100        '   ####
1775 DATA &X01000010        ' #    #
1776 DATA &X01111110        ' ######
1777 DATA &X01000000        ' #
1778 DATA &X00111100        '   ####
1779 DATA &X00000000        '
1780 '
1781 'f
1782 '
1783 DATA &X00001100        '     ##
1784 DATA &X00010010        '    #  #
1785 DATA &X00010000        '    #
1786 DATA &X01111100        ' #####
1787 DATA &X00010000        '    #
1788 DATA &X00010000        '    #
1789 DATA &X00010000        '    #
1790 DATA &X00000000        '
1791 '
```

```
1792 'g
1793 '
1794 DATA &X00000000        '
1795 DATA &X00000000        '
1796 DATA &X00111010        '  ■■■ ■
1797 DATA &X01000110        ' ■   ■■
1798 DATA &X01000110        ' ■   ■■
1799 DATA &X00111010        '  ■■■ ■
1800 DATA &X00000010        '      ■
1801 DATA &X00111100        '  ■■■■
1802 '
1803 'h
1804 '
1805 DATA &X01000000        ' ■
1806 DATA &X01000000        ' ■
1807 DATA &X01011100        ' ■ ■■■
1808 DATA &X01100010        ' ■■   ■
1809 DATA &X01000010        ' ■    ■
1810 DATA &X01000010        ' ■    ■
1811 DATA &X01000010        ' ■    ■
1812 DATA &X00000000        '
1813 '
1814 'i
1815 '
1816 DATA &X00001000        '    ■
1817 DATA &X00000000        '
1818 DATA &X00011000        '   ■■
1819 DATA &X00001000        '    ■
1820 DATA &X00001000        '    ■
1821 DATA &X00001000        '    ■
1822 DATA &X00011100        '   ■■■
1823 DATA &X00000000        '
1824 '
1825 'j
1826 '
1827 DATA &X00000100        '     ■
1828 DATA &X00000000        '
1829 DATA &X00001100        '    ■■
1830 DATA &X00000100        '     ■
1831 DATA &X00000100        '     ■
1832 DATA &X00000100        '     ■
1833 DATA &X01000100        ' ■   ■
1834 DATA &X00111000        '  ■■■
1835 '
1836 'k
1837 '
1838 DATA &X01000000        ' ■
1839 DATA &X01000000        ' ■
1840 DATA &X01000100        ' ■   ■
1841 DATA &X01001000        ' ■  ■
1842 DATA &X01010000        ' ■ ■
1843 DATA &X01101000        ' ■■ ■
1844 DATA &X01000100        ' ■   ■
1845 DATA &X00000000        '
1846 '
```



```
1847 'l
1848 '
1849 DATA &X00011000        '   ■■
1850 DATA &X00001000        '    ■
1851 DATA &X00001000        '    ■
1852 DATA &X00001000        '    ■
1853 DATA &X00001000        '    ■
1854 DATA &X00001000        '    ■
1855 DATA &X00011100        '   ■■■
1856 DATA &X00000000        '
1857 '
1858 'm
1859 '
1860 DATA &X00000000        '
1861 DATA &X00000000        '
1862 DATA &X01110110        ' ■■■ ■■
1863 DATA &X01001001        ' ■  ■  ■
1864 DATA &X01001001        ' ■  ■  ■
1865 DATA &X01001001        ' ■  ■  ■
1866 DATA &X01001001        ' ■  ■  ■
1867 DATA &X00000000        '
1868 '
1869 'n
1870 '
1871 DATA &X00000000        '
1872 DATA &X00000000        '
1873 DATA &X01011100        ' ■ ■■■
1874 DATA &X01100010        ' ■■   ■
1875 DATA &X01000010        ' ■    ■
1876 DATA &X01000010        ' ■    ■
1877 DATA &X01000010        ' ■    ■
1878 DATA &X00000000        '
1879 '
1880 'o
1881 '
1882 DATA &X00000000        '
1883 DATA &X00000000        '
1884 DATA &X00111100        '  ■■■■
1885 DATA &X01000010        ' ■    ■
1886 DATA &X01000010        ' ■    ■
1887 DATA &X01000010        ' ■    ■
1888 DATA &X00111100        '  ■■■■
1889 DATA &X00000000        '
1890 '
1891 'p
1892 '
1893 DATA &X00000000        '
1894 DATA &X00000000        '
1895 DATA &X01011100        ' ■ ■■■
1896 DATA &X01100010        ' ■■   ■
1897 DATA &X01100010        ' ■■   ■
1898 DATA &X01011100        ' ■ ■■■
1899 DATA &X01000000        ' ■
1900 DATA &X01000000        ' ■
1901 '
```

```
1902 'q
1903 '
1904 DATA &X00000000        '
1905 DATA &X00000000        '
1906 DATA &X00111010        '  ■■■ ■
1907 DATA &X01000110        ' ■   ■■
1908 DATA &X01000110        ' ■   ■■
1909 DATA &X00111010        '  ■■■ ■
1910 DATA &X00000010        '      ■
1911 DATA &X00000010        '      ■
1912 '
1913 'r
1914 '
1915 DATA &X00000000        '
1916 DATA &X00000000        '
1917 DATA &X01011100        ' ■ ■■■
1918 DATA &X01100010        ' ■■   ■
1919 DATA &X01000000        ' ■
1920 DATA &X01000000        ' ■
1921 DATA &X01000000        ' ■
1922 DATA &X00000000        '
1923 '
1924 's
1925 '
1926 DATA &X00000000        '
1927 DATA &X00000000        '
1928 DATA &X00111110        '  ■■■■■
1929 DATA &X01000000        ' ■
1930 DATA &X00111100        '  ■■■■
1931 DATA &X00000010        '      ■
1932 DATA &X01111100        ' ■■■■■
1933 DATA &X00000000        '
1934 '
1935 't
1936 '
1937 DATA &X00010000        '    ■
1938 DATA &X00010000        '    ■
1939 DATA &X01111100        ' ■■■■■
1940 DATA &X00010000        '    ■
1941 DATA &X00010000        '    ■
1942 DATA &X00010010        '    ■  ■
1943 DATA &X00001100        '     ■■
1944 DATA &X00000000        '
1945 '
1946 'u
1947 '
1948 DATA &X00000000        '
1949 DATA &X00000000        '
1950 DATA &X01000010        ' ■    ■
1951 DATA &X01000010        ' ■    ■
1952 DATA &X01000010        ' ■    ■
1953 DATA &X01000110        ' ■   ■■
1954 DATA &X00111010        '  ■■■ ■
1955 DATA &X00000000        '
1956 '
```



```
1957 'v
1958 '
1959 DATA &X00000000        '
1960 DATA &X00000000        '
1961 DATA &X01000010        ' ■    ■
1962 DATA &X01000010        ' ■    ■
1963 DATA &X01000010        ' ■    ■
1964 DATA &X00100100        '  ■  ■
1965 DATA &X00011000        '   ■■
1966 DATA &X00000000        '
1967 '
1968 'w
1969 '
1970 DATA &X00000000        '
1971 DATA &X00000000        '
1972 DATA &X01000001        ' ■     ■
1973 DATA &X01001001        ' ■  ■  ■
1974 DATA &X01001001        ' ■  ■  ■
1975 DATA &X01001001        ' ■  ■  ■
1976 DATA &X00110110        '  ■■ ■■
1977 DATA &X00000000        '
1978 '
1979 'x
1980 '
1981 DATA &X00000000        '
1982 DATA &X00000000        '
1983 DATA &X01000010        ' ■    ■
1984 DATA &X00100100        '  ■  ■
1985 DATA &X00011000        '   ■■
1986 DATA &X00100100        '  ■  ■
1987 DATA &X01000010        ' ■    ■
1988 DATA &X00000000        '
1989 '
1990 'y
1991 '
1992 DATA &X00000000        '
1993 DATA &X00000000        '
1994 DATA &X01000010        ' ■    ■
1995 DATA &X01000010        ' ■    ■
1996 DATA &X01000110        ' ■   ■■
1997 DATA &X00111010        '  ■■■ ■
1998 DATA &X00000010        '      ■
1999 DATA &X00111100        '  ■■■■
2000 '
2001 'z
2002 '
2003 DATA &X00000000        '
2004 DATA &X00000000        '
2005 DATA &X01111110        ' ■■■■■■
2006 DATA &X00000100        '     ■
2007 DATA &X00011000        '   ■■
2008 DATA &X00100000        '  ■
2009 DATA &X01111110        ' ■■■■■■
2010 DATA &X00000000        '
2011 '
```

```
2012 '{
2013 '
2014 DATA &X00001100        '    ■■
2015 DATA &X00010000        '   ■
2016 DATA &X00001000        '    ■
2017 DATA &X00110000        '  ■■
2018 DATA &X00001000        '    ■
2019 DATA &X00010000        '   ■
2020 DATA &X00001100        '    ■■
2021 DATA &X00000000        '
2022 '
2023 '|
2024 '
2025 DATA &X00001000        '    ■
2026 DATA &X00001000        '    ■
2027 DATA &X00001000        '    ■
2028 DATA &X00001000        '    ■
2029 DATA &X00001000        '    ■
2030 DATA &X00001000        '    ■
2031 DATA &X00001000        '    ■
2032 DATA &X00000000        '
2033 '
2034 '}
2035 '
2036 DATA &X00110000        '  ■■
2037 DATA &X00001000        '    ■
2038 DATA &X00010000        '   ■
2039 DATA &X00001100        '    ■■
2040 DATA &X00010000        '   ■
2041 DATA &X00001000        '    ■
2042 DATA &X00110000        '  ■■
2043 DATA &X00000000        '
```

Programmbeschreibung:

Auf den Programmkopf folgend wird in Zeile 1007 mit dem BASIC-Kommando SYMBOL AFTER 33 die Umdefinition der Zeichen ab Nummer 33 eingeleitet. Von Zeile 1009 bis Zeile 1018 wird in einer FOR-TO-NEXT-Schleife der Zeichensatz modifiziert. Die äußere Schleife erzeugt die Nummern der Zeichen die umdefinierte werden sollen. In einer zweiten FOR-TO-NEXT-Schleife werden dann acht Zahlenwerte, in denen die Zeichenmatrix enthalten ist, aus DATA-Statements ausgelesen und den indizierten Variablen TX(0) bis TX(7) zugewiesen. Unter der aktuellen Zeichennummer (I) wird schließlich die Matrix in den Zeichensatz aufgenommen (Zeile 1016). In gleicher Weise wird mit allen Zeichen verfahren. Ist die Umdefinition abgeschlossen, stellt das Programm den neuen Zeichensatz zur Verfügung und löscht sich selbst im Speicher (Zeile 1020)!

Von Zeile 1021 bis zum Ende des Programms stehen DATA-Statements, von denen jeweils acht eine Gruppe bilden, die die Matrix eines Zeichens enthält. Um etwas Struktur in den Wust von DATAs zu bringen, sind die Statements einer Gruppe durch Kommentarzeilen getrennt, die jeweils das Zeichen in Klarschrift enthalten, das in der nachfolgenden Matrix verschlüsselt steht. Aus Gründen besserer Anschaulichkeit wurde eine binäre Darstellungsform für die Zeichenmatrizen gewählt. Die Kommentare zu den DATA-Statements dienen ausschließlich der besseren Übersicht des Lesers und können nicht eingetippt werden.

3.7 DESTROYED - Ein Arcade-Spiel in BASIC

Mit Videoautomaten wie "Space Invaders", "Defender" oder "Puc-Man" wurde Mitte der siebziger Jahre ein Meilenstein in der Unterhaltungsindustrie gelegt. Erstmals war es gelungen, den Computer - wenn auch im eingeschränkten Maße - kommerziell nutzbar zu machen. Als weltweit die Spielleidenschaft der Bevölkerung drastisch zunahm, eskalierte die ganze Sache zu einem regelrechten "Videoboom". Er war auch maßgeblich an der rasanten Entwicklung der Homecomputer beteiligt.

Heute kann man für nahezu jedes Computersystem Videospiele erwerben. Auch und gerade beim Schneider-CPC ist die Auswahl an Videospielen beachtlich. Leider ist dieses Spielvergnügen nicht gerade billig und so verzichten viele Computerbesitzer auf Einsatz ihres Gerätes im Freizeitbereich. Dem kann abgeholfen werden!

Obwohl der Schneider-CPC nicht unbedingt für den Videospielbetrieb konzipiert worden ist, bietet er dank seiner enormen Grafikfähigkeiten die Möglichkeit, sowohl in BASIC als auch in Maschinensprache eigene Videospiele zu entwickeln. Erstaunlicherweise kann auch ein in BASIC realisiertes Spiel - dank des mächtigen BASIC-Befehlsschatzes aller CPCs - einen durchaus schnellen und rasanten Spielverlauf beinhalten und somit aktionsreich gestaltet werden. Und gerade diese Tatsache ermöglicht auch dem Anfänger mit relativ wenig Aufwand ein Videospiel

mit hohem Qualitätsstandard zu entwickeln. Wie und mit welcher Technik man eine Spielidee in ein Programm umsetzt, soll an dieser Stelle erläutert werden. In diesem Zusammenhang wird die vorgestellte "Charakter-Definition" eine bedeutende Rolle spielen. Alle in unserem Spiel vorkommenden Objekte werden nämlich mit Hilfe von umdefinierten Charaktern erzeugt. Große Objekte bestehen in diesem Zusammenhang - ähnlich wie ein Mosaik - aus mehreren Bestandteilen, wobei wiederum umdefinierte Charakter die Objektteile repräsentieren.

3.7.1 Die Spielidee

Jedem Videospiel geht eine Spielidee vorraus. Es ist sinnlos, nach bester Hackermanier einfach "drauflos zu programmieren". Derartige Versuche enden in den meisten Fällen in einem absoluten Chaos und sind geradezu zum Scheitern verurteilt. Was jeder Spieleprogrammierer benötigt, ist eine Grundidee und ein Spielziel - einen Leitfaden, den er Stück für Stück in sein Programm umsetzen kann. Derartige Ideen sind leicht zu finden. Beispielsweise kann eine Film, die Story eines actiongeladenen Buches oder ein professioneller Spielautomat als Vorlage für Ihr Videospiel dienen. Selbstverständlich ist auch Ihrer eigenen Phantasie keine Grenzen gesetzt.

Das Videospiel, das wir Ihnen vorstellen möchten, trägt den Namen "DESTROYED" und gehört zur Kategorie der Weltraumspiele. Die Hintergrundgeschichte ist schnell erklärt.

> "Irgendwo im Universum befindet sich eine riesige Raumstation. Feindliche Piraten wollen sie zerstören, um die absolute Macht über das heimische Sonnensystem zu erlangen. Es gilt nun, die Piraten mit vier zu Verfügung stehenden gigantischen Bordkanonen abzuwehren. Da die Akkus des Waffensystems der Raumstation nicht für den simultanen Einsatz der Laserkanonen ausgelegt sind, kann nur immer abwechselnd mit einem Geschütz geschossen werden. Die gesamte Aktion wird zusätzlich durch den begrenzten Energievorrat erschwert.
>
> Die Piraten greifen in vier Angriffswellen von allen Seiten an. Zuerst versuchen sie, mit ihren Schiffen die Station nach bester Kamikaze-Manier zu rammen. Danach setzen sie Photonentorpedos ein. In den beiden letzten Angriffswellen kombinieren die Piraten ihre Strategien und greifen mit allem an, was sie haben. Das Ganze endet in einer gigantischen Materialschlacht, bei der der Commander unserer Raumstation - also der Spieler - einen kühlen Kopf behalten muß und nicht durch sinnlose Ballerei Energie verschwenden darf. Jeder Schuß muß sitzen! Ist auch diese Schlacht erfolgreich geschlagen, erwartet den siegreichen Helden ein gefahrloses Training (Bonus-Runde), die sich positiv auf sein Punktekonto auswirken kann. Danach beginnt der ganze Zauber mit erhöhtem Schwierigkeitsgrad von neuem..."

Um die feindlichen Piratenschiffe abzuwehren, muß der Spieler die gewünschte Laserkanone mit der in deren Richtung zeigenden Pfeiltaste aktivieren. Geschossen wird mittels der Leertaste!

3.7.2 Aufbau der Szenarien

Wie aus der Spielidee ersichtlich ist, besteht eine Spielrunde aus fünf Szenarien - vier Angiffswellen und einer Bonus-Runde. Jede dieser Spielszenen muß nun grafisch gestaltet werden. Damit wir einen Überblick über die für den Spielablauf benötigten Objekte bekommen, muß die Handlung jedes Szenarios definiert werden. Die einzelnen Spielszenen unterscheiden sich wie folgt:

Szene 1 (Destruction Wave):

Feindliche Raumschiffe greifen an und müssen zerstört werden.

Szene 2 (Missile Wave):

Feindliche Geschosse müssen zerstört werden.

Szene 3 (Survival Wave):

Feindliche Raumschiffe und Geschosse müssen zerstört werden. Falls dabei nicht die eigene Station eliminiert wird, erhält der Spieler einen Bonus.

Szene 4 (Final Wave):

Feindliche Raumschiffe und Geschosse müssen zerstört werden. Für die verbleibende Energie erhält der Spieler am Ende der Angrifswelle einen Bonus.

Szene 5 (Bonus Round):

Es müssen so viele feindliche Geschosse wie möglich eleminiert werden. Die Raumstation ist in dieser Runde unzerstörbar.

Nach diesem Überblick ist es uns möglich, eine Liste aller benötigen Objekte aufzustellen:

1. eine Raumstation
2. vier Piratenschiffe (ein Schiff aus vier Perspektiven)
3. eine Raumtorpedo (Geschoß)

Weiterhin soll die Anzahl der verbleibenden Stationen mit je einer Flagge gekennzeichnet werden und durch die ASCII-Zeichen 48 bis 122 eine futuristische Gestalt bekommen. Die Umdefinition dieser Fülle von Charaktern realisiert der Benutzer am besten mit dem im Kapitel 3.4 vorgestellten Zeichen-Editor. Er ist so konzipiert, daß er den umdefinierten Zeichensatz in Form von SYMBOL-Zeilen in einem automatisch generierten BASIC-Programm auf Diskette ablegt. Diese Daten können dann

im nächsten Arbeitsschritt mittels MERGE-Befehl problemlos mit dem Hauptprogramm verbunden werden.

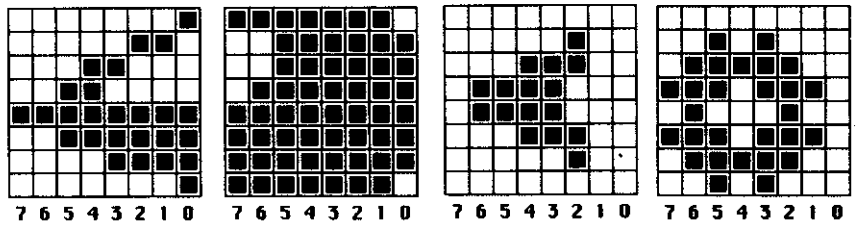


Abb 6: **Feindliches Raumschiff und Mine**

Nach der Definition aller im Spiel vorkommenden Objekte gilt es nun, ein oder mehrere zum Spiel passende Hintergrundgrafiken zu gestalten. Diese Grafiken können kompliziert oder einfach gestaltet werden. Wir empfehlen für das Erstellen aufwendiger Hintergrundbilder einen Grafik-Editor zu verwenden, da er die Zeit, die zum Erstellen eines Bildes notwendig ist, um ein Vielfaches reduziert. Derartige Grafik-Editoren (Malprogramme) sind im Handel erhältlich und oft schon für weniger als DM 100.- zu erwerben. Alternativ können Sie natürlich auch den Grafik-Editor selbst entwickeln. Als Grundlage dazu kann das in Kapitel 2.5 beschriebende Zeichenprogramm dienen. Sind alle benötigten Hintergundgrafiken erstellt, können sie im Spielprogramm mit dem LOAD-Befehl jederzeit in den Bildschirmspeicher geladen werden.

Einfach gestaltete Bilder können direkt vom Spielprogramm aus generiert werden. Unser Weltraumspiel plottet beispielsweise mit der Zeile

```
1500 FOR i=1 TO 100:PLOT(RND(1)*639),(RND(1)*399),3:NEXT
```

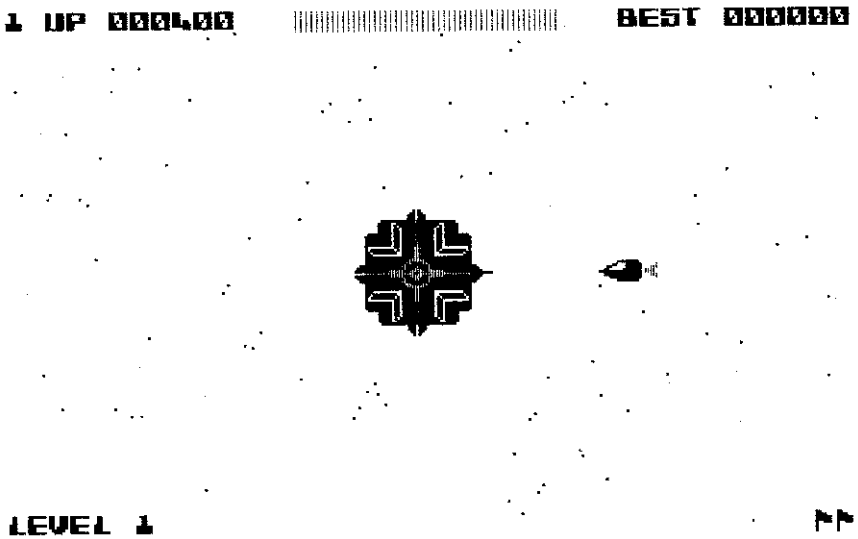
hundert weiße Punkte, die Sterne symbolisieren sollen, mit Hilfe der Zufallsfunktion RND auf dem Bildschirm. Danach wird der Bildschirm mit

SCORE	(Punktzahl)
BEST	(beste Punktzahl) und
WAVE	(Angriffswelle)

beschriftet. Weiterhin wird mit der Zeile

```
1770 FOR EN=221 TO 421 STEP 4:MOVE EN,400:DRAW EN,384,2:NEXT
```

ein Energieanzeiger, der aus mehreren hintereinander angeordneten Strichen besteht, in den oberen Teil des Bildschirms gebracht.

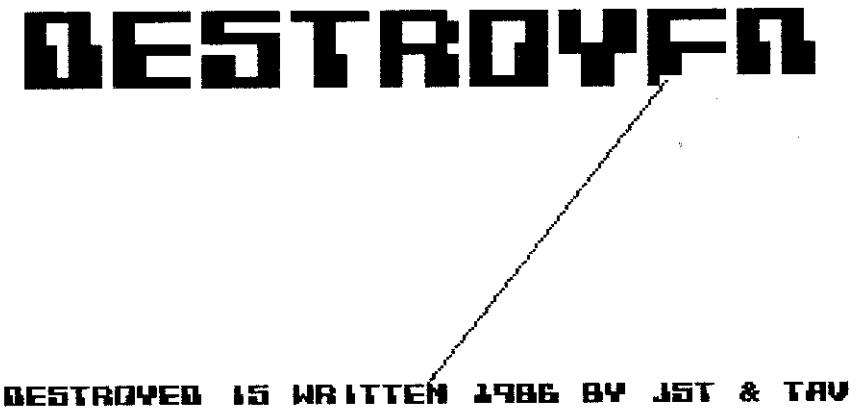


Hardcopy 4: Spielsituation

3.7.3 Animation

Die fertig definierten und im Spielprogramm implementierten Zeichen müssen jetzt per Programm auf dem Bildschirm animiert werden. Die von uns verwendete Routine löst dieses Problem mit einer einfachen Methode: Alle Objekte werden in einer Hauptschleife, deren Laufvariable den zu ändernden Positionswert enthält, bewegt. Sie werden dabei grundsätzlich mit einem nachstehenden Leerzeichen (Space) auf dem Bildschirm gebracht. Wird nun das Objekt im nächsten Schleifendurchlauf ein Zeichen weiter bewegt, wird automatisch das Objekt oder letzte Zeichen des Objekts durch das der Zeichenkette folgende Leerzeichen überschrieben.

Die Laserstrahlen, mit denen der Spieler auf die feindlichen Objekte schießen kann, sind nichts weiteres als vier DRAW-Funktionen, von denen jeweils eine aufgerufen werden kann. Da die Linie mit einem PEN, dem zwei INK-Parameter zugeordnet sind, gezeichnet wird, entsteht der Eindruck eines fluoreszierenden Strahls.



Hardcopy 5: Titelbild

Hardcopy 6: Score-Table

Damit Ihnen das Abtippen des sehr umfangreichen Listings etwas leichter fällt, dokumentieren die oben aufgeführten Hardcopies einige Szenen des Spielverlaufs. Hardcopy 5 zeigt wie zu Anfang des Spiels der Titel mit einem "Turbo-Laser" in eine "Titan-Platte" eingebrannt wird. Ist der Schriftzug auf diese Weise vollendet, wird er an das obere Ende des Bildschirms gescrollt, und der auf Hardcopy 6 dargestellte Score-Table aufgebaut.

Sowohl während des Titelaufbaus (Hardcopy 5) als auch beim Erstellen des Score-Tables (Hardcopy 6) kann jederzeit durch das Drücken einer beliebigen Taste mit dem Spiel begonnen werden.


```
100 MODE 1
110 BORDER 0:SPEED KEY 255,255
120 DEFINT A-Z:DIM KN(3,5),HI!(9),INI$(9)
130 FOR I=0 TO 9:HI!(I)=1000:INI$(I)="XXX":NEXT
140 RANDOMIZE TIME
150 WV$(0)="DESTRUCTION WAVE":WV$(1)="MISSILE WAVE":WV$(2)="SURVIVAL WAV
E":WV$(3)="FINAL WAVE":WV$(4)="BONUS ROUND"
160 '
170 'DATAS IN ARRAYS
180 '
190 FOR I=0 TO 3:FOR J=0 TO 5:READ X:KN(I,J)=X:NEXT:NEXT
200 '
210 SYMBOL AFTER 48
220 SYMBOL 48,254,206,206,214,230,230,254,0
230 SYMBOL 49,56,24,24,24,126,126,126,0
240 SYMBOL 50,254,6,6,254,192,192,254,0
250 SYMBOL 51,124,12,12,126,14,14,254,0
260 SYMBOL 52,224,224,224,236,238,254,14,0
270 SYMBOL 53,254,192,192,254,14,14,254,0
280 SYMBOL 54,248,216,192,254,206,206,254,0
290 SYMBOL 55,254,6,6,62,56,56,56,0
300 SYMBOL 56,252,204,204,254,206,206,254,0
310 SYMBOL 57,254,198,198,254,14,14,14,0
320 SYMBOL 65,126,102,102,254,230,230,230,0
330 SYMBOL 66,252,204,204,254,206,206,254,0
340 SYMBOL 67,254,206,206,192,198,198,254,0
350 SYMBOL 68,252,204,204,238,238,238,254,0
360 SYMBOL 69,254,192,192,254,224,224,254,0
370 SYMBOL 70,254,192,192,254,224,224,224,0
380 SYMBOL 71,254,198,192,222,206,206,254,0
390 SYMBOL 72,198,198,198,254,206,206,206,0
400 SYMBOL 73,24,24,24,28,28,28,28,0
410 SYMBOL 74,12,12,12,14,14,110,126,0
420 SYMBOL 75,204,216,216,254,206,206,206,0
430 SYMBOL 76,48,48,48,112,112,112,126,0
440 SYMBOL 77,206,254,254,238,206,206,206,0
450 SYMBOL 78,206,238,254,222,206,206,206,0
460 SYMBOL 79,254,198,198,206,206,206,254,0
470 SYMBOL 80,254,198,198,254,224,224,224,0
480 SYMBOL 81,252,204,204,204,206,222,254,0
```

```
490 SYMBOL 82,252,198,198,252,206,206,206,0
500 SYMBOL 83,254,192,254,6,230,230,254,0
510 SYMBOL 84,254,48,48,48,56,56,56,0
520 SYMBOL 85,206,206,206,206,206,206,254,0
530 SYMBOL 86,206,206,206,206,222,124,56,0
540 SYMBOL 88,230,230,230,124,206,206,206,0
550 SYMBOL 89,206,206,206,254,56,56,56,0
560 SYMBOL 90,254,204,216,48,110,206,254,0
570 '
580 SYMBOL 240,0,0,0,0,40,108,238,238
590 SYMBOL 241,238,238,108,40,0,0,0,0
600 SYMBOL 242,0,3,7,15,0,15,7,3
610 SYMBOL 243,0,192,224,240,0,240,224,192
620 SYMBOL 244,16,16,16,16,56,124,254,254
630 SYMBOL 245,254,254,124,56,16,16,16,16
640 SYMBOL 246,0,3,7,15,255,15,7,3
650 SYMBOL 247,0,192,224,240,255,240,224,192
660 '
670 SYMBOL 207,1,6,24,48,255,63,15,1
680 SYMBOL 208,254,63,63,127,255,255,255,254
690 SYMBOL 209,0,4,28,120,120,28,4,0
700 '
710 SYMBOL 204,143,78,78,46,60,28,8,8
720 SYMBOL 205,126,255,255,255,255,255,159,143
730 SYMBOL 206,0,0,102,60,60,24,24,0
740 '
750 SYMBOL 201,8,8,28,60,46,78,78,143
760 SYMBOL 202,143,159,255,255,255,255,255,126
770 SYMBOL 203,0,24,24,60,60,102,0,0
780 '
790 SYMBOL 210,128,96,24,12,255,252,240,128
800 SYMBOL 211,127,252,252,254,255,255,255,127
810 SYMBOL 212,0,32,56,30,30,56,32,0
820 '
830 SYMBOL 213,0,40,86,84,168,54,212,42
840 SYMBOL 214,0,40,124,238,68,238,124,40
850 '
860 SYMBOL 215,3,7,7,7,7,127,255,255
870 SYMBOL 216,255,207,215,219,219,219,219,219
880 SYMBOL 217,239,239,239,239,239,239,239,239
```



```
890 SYMBOL 218,255,243,235,219,219,219,219,219
900 SYMBOL 219,192,224,224,224,224,254,255,255
910 SYMBOL 220,255,255,128,191,223,224,255,255
920 SYMBOL 221,219,219,27,235,243,3,255,255
930 SYMBOL 222,239,239,199,199,199,199,199,1
940 SYMBOL 223,219,219,216,215,207,192,255,255
950 SYMBOL 224,255,255,1,253,251,7,255,255
960 SYMBOL 225,255,255,255,255,0,255,255,255
970 SYMBOL 226,254,252,252,129,0,129,252,252
980 SYMBOL 227,40,238,198,131,16,131,198,238
990 SYMBOL 228,255,127,127,3,0,3,127,127
1000 SYMBOL 229,255,255,255,255,0,255,255,255
1010 SYMBOL 230,255,255,255,224,223,191,128,255
1020 SYMBOL 231,254,255,255,3,243,235,27,219
1030 SYMBOL 232,40,1,199,199,199,199,199,239
1040 SYMBOL 233,255,255,255,192,207,215,216,219
1050 SYMBOL 200,96,120,126,126,78,64,64,64
1060 SYMBOL 234,255,255,255,7,251,253,1,255
1070 SYMBOL 235,255,255,127,7,7,7,7,3
1080 SYMBOL 236,219,219,219,219,219,215,207,255
1090 SYMBOL 237,239,239,239,239,239,239,239,239
1100 SYMBOL 238,219,219,219,219,219,235,243,255
1110 SYMBOL 239,255,255,254,224,224,224,224,192
1120 '
1130 'INTRO
1140 '
1150 LBX=320: LBY=16
1160 SPEED INK 4,4:INK 0,0:INK 1,24:INK 2,11,6:INK 3,13:PEN 3:CLS
1170 LOCATE 1,25: PRINT" DESTROYED IS WRITTEN 1986 BY JST & TAV"
1180 TAG
1190 FOR Y=7 TO 1 STEP -1
1200 FOR X=0 TO 72
1210 IF TEST(X*2+16,Y*2)=3 THEN LPX=X*8+32: LPY=Y*8+230:MOVE LBX,LBY:DRA
W LPX+4,LPY-4,2:MOVE LBX,LBY:DRAW LPX+4,LPY-4,0:PLOT LPX,LPY,1:PRINT CHR
$(129);
1220 IF INKEY$<>"" THEN 1430
1230 NEXT X
1240 NEXT Y
1250 INK 1,6,11:FOR I=1 TO 3000:NEXT
1260 TAGOFF
```

```
1270 LOCATE 1,25:PRINT SPACE$(40)
1280 FOR I=1 TO 5:PRINT:NEXT
1290 INK 3,24:PEN 3
1300 LOCATE 4,10:PRINT"YOUR MISSION IS TO DESTROY ALIENS,":LOCATE 23,12:
PRINT"GOOD LUCK!"
1310 LOCATE 7,17:PRINT"MINE:"
1320 LOCATE 7,19:PRINT"ALIEN SHIP:"
1330 LOCATE 7,21:PRINT"EXTRA BASE:"
1340 INK 2,13:PEN 2:FOR I=40 TO 20 STEP-1:LOCATE I,17:PRINT CHR$(214);"
";:FOR J=1 TO 300:NEXT:NEXT:PEN 3:PRINT"    50 PTS"
1350 FOR I=38 TO 19 STEP-1:LOCATE I,19:PEN 3:PRINT CHR$(207);CHR$(208);:
PEN 1:PRINT CHR$(209);" ";:FOR J=1 TO 300:NEXT:NEXT:PEN 3:PRINT"  100 PT
S"
1360 INK 2,13:PEN 2:FOR I=40 TO 20 STEP-1:LOCATE I,21:PRINT CHR$(200);"
";:FOR J=1 TO 300:NEXT:NEXT:PEN 3:PRINT" 20000 PTS"
1370 PEN 2:LOCATE 6,25:PRINT"PRESS ANY KEY TO INSERT COIN";
1380 FOR I=1 TO 15000:IF INKEY$="" THEN NEXT ELSE 1430
1390 CLS:INK 2,6:LOCATE 13,1:PRINT"ALL TIME HEROS:":INK 3,1
1400 INK 1,18,24:SPEED INK 15,15:FOR I=0 TO 9:LOCATE 11,5+I*2:PRINT USIN
G"##";I+1;:PRINT".   ";:PEN 1:PRINT USING"######";HI!(I);:PEN 3:PRINT SP
C(3);INI$(I):PEN 2:NEXT
1410 FOR I=1 TO 15000:IF INKEY$="" THEN NEXT ELSE 1430
1420 CLS:GOTO 1160
1430 '
1440 TAGOFF:CLS:SPEED INK 4,4:INK 0,0:INK 1,24:INK 2,11,6:INK 3,13
1450 '
1460 LEVEL=0:OFS=-5:WAVE=-1:PP=1:BASE=3::FB=1:AS=0
1470 '
1480 'STERNENHIMMEL
1490 '
1500 FOR i=1 TO 100:PLOT(RND(1)*639),(RND(1)*399),3:NEXT
1510 '
1520 'ENERGIEZENTRUM
1530 '
1540 PEN 2
1550 LOCATE 20,11:PRINT CHR$(143)
1560 LOCATE 20,12:PRINT CHR$(143)
1570 LOCATE 18,13:PRINT STRING$(5,143)
1580 LOCATE 20,14:PRINT CHR$(143)
1590 LOCATE 20,15:PRINT CHR$(143)
```



```
1600 '
1610 'STATION AUFBAUEN
1620 '
1630 PEN 1
1640 PRINT CHR$(22);CHR$(1)
1650 S0=0:FOR i=1 TO 5:LOCATE 18,i+10:FOR j=215+S0 TO 219+S0:PRINT CHR$(
j);:NEXT:S0=S0+5:NEXT
1660 PRINT CHR$(22);CHR$(0)
1670 LOCATE 17,13:PRINT CHR$(242);
1680 LOCATE 23,13:PRINT CHR$(243);
1690 LOCATE 20,10:PRINT CHR$(244);
1700 LOCATE 20,16:PRINT CHR$(241);
1710 '
1720 PEN 3:LOCATE 1,1:PRINT"1 UP 000000";SPC(18);"BEST";:LOCATE 35,1:SCO
RE!=BEST!:GOSUB 2570:SCORE!=0:LOCATE 1,25:PRINT"LEVEL 1":LOCATE 39,25:PR
INT STRING$(2,200);:PEN 1
1730 '
1740 'HAUPTSCHLEIFE
1750 '
1760 LEVEL=LEVEL+1:PEN 3:LOCATE 6,25:PRINT LEVEL
1770 FOR EN=221 TO 421 STEP 4:MOVE EN,400:DRAW EN,384,2:NEXT
1780 OFS=OFS+5:IF OFS>20 THEN OFS=20
1790 '
1800 SHPC=30+OFS:WAVE=WAVE+1:IF WAVE>4 THEN WAVE=-1:GOTO 1760
1810  IF WAVE=4 THEN I=EN:FOR I=EN TO 220 STEP-1:MOVE I,400:DRAW I,384,0
:NEXT:EN=(EN-220)*10:TEXT$="FUEL BONUS ="+STR$(EN):GOSUB 2540:SCORE!=SCO
RE!+EN:GOSUB 2550:INK 1,0 ELSE INK 1,24
1820 IF WAVE=3 THEN IF SUR=1 THEN TEXT$="SURVIVAL POINTS ="+STR$(SHPC*50
):GOSUB 2540:SCORE!=SCORE!+SHPC*50:GOSUB 2550 ELSE TEXT$="SORRY, NO BONU
S":GOSUB 2540
1830 IF WAVE=2 THEN INK 0,1:BORDER 1 ELSE INK 0,0:BORDER 0:SUR=1
1840 TEXT$=WV$(WAVE):GOSUB 2540
1850 '
1860 GOSUB 2280
1870 IF OX<17 OR OX>23 OR OY<10 OR OY>16 THEN 1930
1880 IF WAVE=4 THEN AS=KNX+1:PP=0:GOTO 2100
1890 BASE=BASE-1:SUR=SUR-1:IF BASE<>0 THEN GOSUB 2580
1900 INK 0,27,0:FOR i=1 TO 1000:NEXT:IF WAVE=2 THEN INK 0,1 ELSE INK 0,0
1910 GOSUB 2550:IF BASE=0 THEN 2590 ELSE TEXT$="GET READY!":GOSUB 2540:K
NX=AS-1:IF KNX>3 THEN KNX=KNX-4
```

```
1920 GOTO 2100
1930 Y$=INKEY$
1940 IF Y$=CHR$(32) THEN 2030
1950 IF y$<CHR$(240) OR Y$>CHR$(243) THEN 1860
1960 '
1970 'KANONE SETZEN
1980 '
1990 LOCATE KN(KNX,0),KN(KNX,1):PRINT CHR$(KNX+240);
2000 KNX=ASC(Y$)-240
2010 LOCATE KN(KNX,0),KN(KNX,1):PRINT CHR$(KNX+244);
2020 GOTO 1860
2030 '
2040 'SCHUSS
2050 '
2060 SOUND 1,4000,40,11,5,3,7
2070 IF WAVE=4 THEN 2090
2080 EN=EN-1:MOVE EN,400:DRAW EN,384,0:IF EN-220<0 THEN 2590
2090 MOVE KN(KNX,2),KN(KNX,3):DRAW KN(KNX,4),KN(KNX,5),2:FOR I=1 TO 20:N
EXT:DRAW KN(KNX,2),KN(KNX,3),0
2100 IF KNX+1=AS THEN LOCATE OX,OY:PEN 2:PRINT CHR$(213);:GOSUB 2190:LOC
ATE OX,OY:PRINT" ";:IF PP THEN SCORE!=SCORE!+50:GOTO 2170 ELSE 2180
2110 IF KNX+5<>AS THEN 1860 ELSE LOCATE OX,OY:PEN 2
2120 IF AS=5 THEN PRINT CHR$(213);CHR$(11);CHR$(8);CHR$(213);CHR$(11);CH
R$(8);:GOSUB 2190:PRINT " ";CHR$(10);CHR$(8);" ";CHR$(10);CHR$(8);" ";
2130 IF AS=6 THEN PRINT CHR$(213);CHR$(10);CHR$(8);CHR$(213);CHR$(10);CH
R$(8);:GOSUB 2190:PRINT " ";CHR$(11);CHR$(8);" ";CHR$(11);CHR$(8);" ";
2140 IF AS=7 THEN PRINT CHR$(8);CHR$(213);CHR$(213);:GOSUB 2190:PRINT ST
RING$(3,8);STRING$(3,32);
2150 IF AS=8 THEN PRINT CHR$(213);CHR$(213);:GOSUB 2190:PRINT STRING$(2,
8);STRING$(3,32);
2160 SCORE!=SCORE!+100 ELSE 2180
2170 GOSUB 2550
2180 PEN 1:AS=0:PP=1:SHPC=SHPC-1:IF SHPC=0 THEN 1800 ELSE 1860
2190 FOR I=1 TO 100:NEXT:RETURN
2200 '
2210 'DATAS MIT POSITIONSANGABEN
2220 '
2230 DATA 20,10,310,255,310,399
2240 DATA 20,16,310,143,310,0
2250 DATA 17,13,256,198,0,198
```



```
2260 DATA 23,13,368,198,639,198
2270 '
2280 IF AS<>0 THEN 2410
2290 AS=INT(RND(1)*4)+1
2300 IF WAVE=1 OR WAVE=4 THEN 2330
2310 IF WAVE=0 THEN AS=AS+4:GOTO 2330
2320 AS=INT(RND(1)*8)+1
2330 IF AS=3 THEN AX=1:AY=13
2340 IF AS=4 THEN AX=40:AY=13
2350 IF AS=1 THEN AX=20:AY=2
2360 IF AS=2 THEN AX=20:AY=25
2370 IF AS=7 THEN AX=1:AY=13
2380 IF AS=8 THEN AX=38:AY=13
2390 IF AS=5 THEN AX=20:AY=3
2400 IF AS=6 THEN AX=20:AY=22
2410 LOCATE AX,AY
2420 IF AS=3 THEN:PEN 3:PRINT" ";CHR$(214);:AX=AX+1:OX=AX:OY=AY
2430 IF AS=4 THEN:PEN 3:OX=AX:OY=AY:PRINT CHR$(214);" ";:AX=AX-1
2440 IF AS=1 THEN PEN 3:PRINT" ";:AY=AY+1:LOCATE AX,AY:PRINT CHR$(214);:
OX=AX:OY=AY
2450 IF AS=2 THEN PEN 3:PRINT" ";:AY=AY-1:LOCATE AX,AY:PRINT CHR$(214);:
OX=AX:OY=AY
2460 IF AS=7 THEN PEN 2:PRINT" ";CHR$(212);:PEN 1:PRINT CHR$(211);CHR$(2
10);:AX=AX+1:OX=AX+2:OY=AY
2470 IF AS=8 THEN OX=AX:OY=AY:PRINT CHR$(207);CHR$(208);:PEN 2:PRINT CHR
$(209);" ";:AX=AX-1
2480 IF AS=5 THEN PEN 2:PRINT" ";:LOCATE AX,AY+1:PRINT CHR$(206);:PEN 1:
LOCATE AX,AY+2:PRINT CHR$(205);:LOCATE AX,AY+3:PRINT CHR$(204);:AY=AY+1:
OX=AX:OY=AY+2
2490 IF AS=6 THEN PRINT CHR$(201);:LOCATE AX,AY+1:PRINT CHR$(202);:LOCAT
E AX,AY+2:PEN 2:PRINT CHR$(203);:PEN 1:LOCATE AX,AY+3:PRINT" ";:OX=AX:OY
=AY:AY=AY-1
2500 '
2510 IF WAVE<>4 THEN FOR I=1 TO 200-OFS*10:NEXT
2520 PEN 1
2530 RETURN
2540 INK 2,6:PEN 2:I=LEN(TEXT$):LOCATE 21-I/2,7:PRINT TEXT$;:FOR J=1 TO
4000:NEXT:LOCATE 21-I/2,7:PRINT SPC(I);:INK 2,11,6:PEN 1:RETURN
2550 IF FB AND SCORE!>20000 THEN BASE=BASE+1:FB=0:GOSUB 2580
2560 LOCATE 6,1
```

```
2570 PEN 3:SCR$=STR$(SCORE!):LNS=LEN(SCR$)-1:SCR$=RIGHT$(SCR$,LNS):PRINT
 STRING$(6-LNS,48);SCR$;:RETURN
2580 PEN 3:TEXT$=" "+STRING$(BASE-1,200):LOCATE 41-LEN(TEXT$),25:PRINT T
EXT$;:PEN 1:RETURN
2590 IF SCORE!>BEST! THEN BEST!=SCORE!:LOCATE 35,1:GOSUB 2570
2600 TEXT$="GAME OVER":GOSUB 2540
2610 IF SCORE!> HI!(9) THEN CLS:LOCATE  1,10:INK 2,6:INK 3,18:INK 0,0:BO
RDER 0:PEN 2:PRINT"CONGRATIULATIONS!":PRINT:PEN 3:PRINT"YOU ARE ONE OF T
HE TEN BEST PLAYERS":PRINT"ENTER YOUR INITIALS:" ELSE GOTO 1150
2620 TEXT$="":I=0:LOCATE 18,16
2630 Y$=INKEY$:Y$=UPPER$(Y$):IF Y$<"A" OR Y$>"Z" THEN 2630 ELSE I=I+1:PE
N I:PRINT Y$;:TEXT$=TEXT$+Y$
2640 IF I<>3 THEN 2630 ELSE FOR I=0 TO 9:IF SCORE!>HI!(I) THEN FOR J=9 T
O I+1 STEP-1:HI!(J)=HI!(J-1):INI$(J)=INI$(J-1):NEXT:HI!(I)=SCORE!:INI$(I
)=TEXT$ ELSE NEXT
2650 GOTO 1150
```


4. Abstrakte Kunst in BASIC

Hatten Sie auch schon einmal das Vergnügen, Gäste zu haben, die sich hervorragend in der Rolle gefielen, Ihren Schneider-CPC schlecht zu machen? Oder können Sie nicht vollkommen ausschließen, einmal solche Gäste zu haben? Sie sollten jedenfalls wissen, daß diese Leute immer die Gretchenfrage nach der Grafikfähigkeit stellen und so den Gastgeber richtig fordern. Mit drei PLOT- und zwei DRAW-Kommandos läßt sich nichts erzeugen, was den Kritikern die Argumente raubt - da müssen schon handfestere Grafiken erscheinen. Doch wie es so ist, in einer solchen Situation fällt auch dem sonst mit allen Wassern gewaschenen Programmierer nichts ein, was so recht beeindrucken kann und es setzt sich bei den ohnehin nicht objektiven Betrachtern schnell der Eindruck fest: Der CPC kann keine Grafik!

Um einen derart jeder Basis entbehrenden Vorwurf in Zukunft mit einem Handstreich beiseite schieben zu können oder ganz einfach das letzte aus der Grafikmaschine CPC zu holen, was BASIC zu leisten vermag, haben wir dieses Kapitel eingefügt. Es soll eine ganze Reihe von Grafikprogrammen vorstellen, die geeignet sind, dem Betrachter ein erstauntes "Ahh" zu entlocken.

Alle Programme sind ohne Erklärungen abgedruckt und können nach eigenem Ermessen beliebig erweitert oder modifiziert werden. Das Ergebnis, das sie in der abgedruckten Form auf dem Bildschirm liefern, ist jeweils in einer Hardcopy festgehalten.

Einige der Programme benötigen zum Gewinnen der Grafik, die dann keiner Bewegung mehr unterliegt, oft beachtliche Rechenzeit, so daß es sinnvoll ist, die statische Grafik auf Diskette zu speichern. Um dies zu erreichen, bedarf es nur der Anweisung

```
SAVE "BILDINHALT",B,&C000,&4000
```

und die 16 KByte Bildschirmspeicher werden als Binärfile auf Diskette gerettet. Diese Anweisung sollte an der Stelle in den Programmfluß eingefügt werden, an der die Grafik fertig erstellt ist und durch das Programm keine Veränderung mehr erfährt. Auf diese Weise läßt sich auch eine Sammlung von Grafiken auf Diskette anlegen und für die nächste Betrachtung, ob mit oder ohne unliebsamen Besuch, konservieren.

Eine auf diese Weise gesicherte Grafik läßt sich jederzeit mit der Anweisung

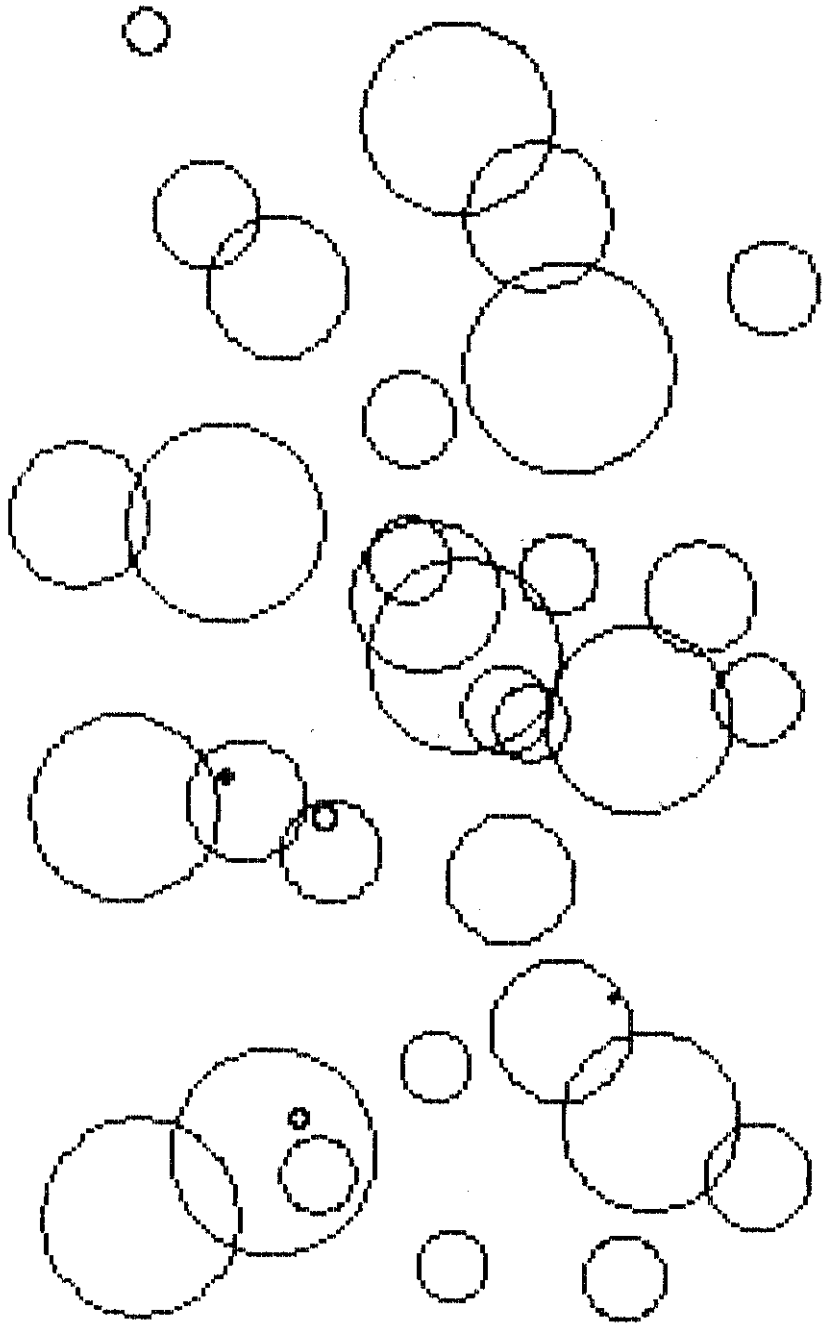
```
LOAD "BILDINHALT.BIN"
```

wieder auf den Bildschirm holen. Durch Zusammenfügen mehrer LOAD-Kommandos kann auch leicht eine Grafikdemo geschrieben werden, die im fortwährenden Wechsel die Grafikfähigkeiten des CPC vorführt. Als Besitzer eines 6128 können Sie diesen Effekt durch die einfache Möglichkeit mit mehreren Bildschirmseiten arbeiten zu können (BANKMAN), noch verstärken.

4.1 "Störe meine Kreise nicht!"

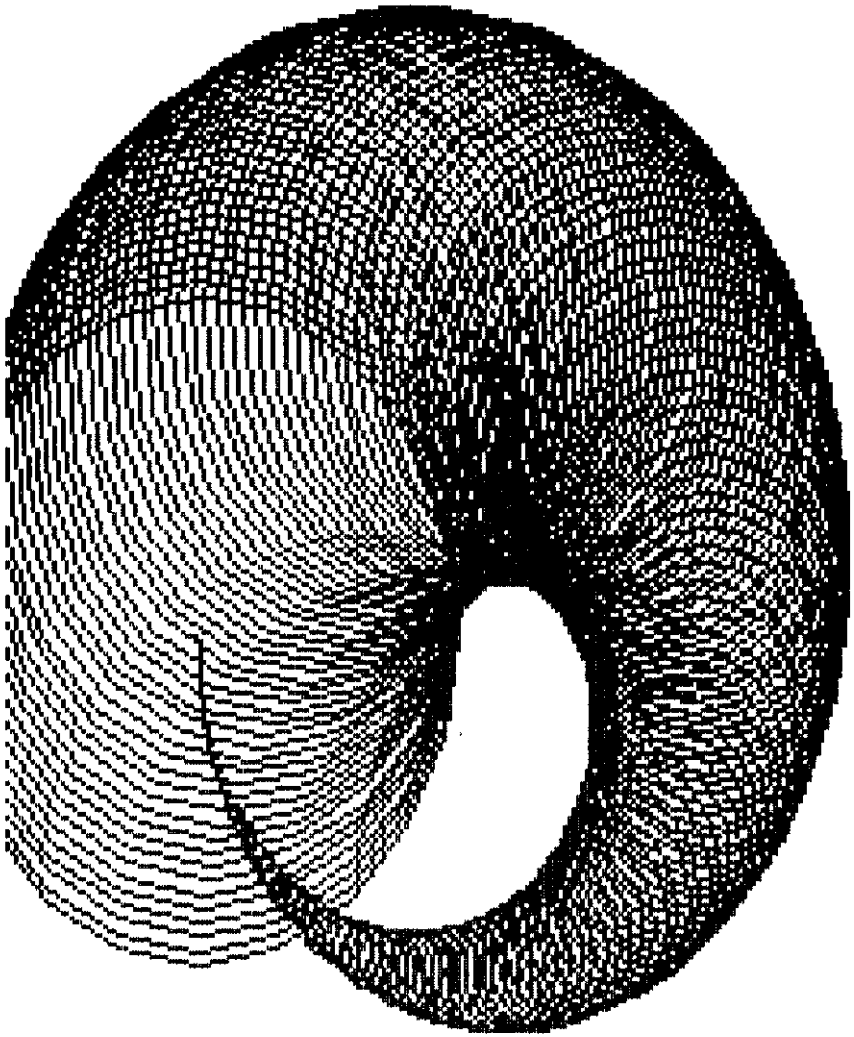
```
100 '************************************************************
110 '***                                                      ***
120 '***          "STOERE MEINE KREISE NICHT!"                ***
130 '***            ABSTRAKTE KUNST IN BASIC                  ***
140 '***                                                      ***
150 '***                JST/TAV 2.8.1986                      ***
160 '***                                                      ***
170 '************************************************************
180 '
190 MODE 1
200 BORDER 0
210 INK 0,0
220 INK 1,26
230 '
240 DEG
250 '
260 'ZUFALLSWERTE FUER MITTELPUNKT UND RADIUS BESTIMMEN
270 '
280 XP%=INT(RND(1)*640)
290 YP%=INT(RND(1)*400)
300 D% =INT(RND(1)*50)
310 '
320 'KANN DER AKTUELLE KREIS VOLLSTAENDIG DARGESTELLT WERDEN?
330 '
340 IF XP%+D%>639 THEN 280
350 IF YP%+D%>399 THEN 280
360 IF XP%-D%<  0 THEN 280
370 IF YP%-D%<  0 THEN 280
380 '
390 'ERSTE KOORDINATEN
400 '
410 J%=0
420 GOSUB 690
430 '
440 X1%=X2%
450 Y1%=Y2%
460 '
```

```
470 'WEITERE KOORDINATEN
480 '
490 FOR J%=0 TO 15
500 GOSUB 690
510 '
520 'KREIS ZEICHEN
530 '
540 MOVE XP%+X1%,YP%+Y1%
550 DRAW XP%+X2%,YP%+Y2%
560 MOVE XP%+X1%,YP%-Y1%+1
570 DRAW XP%+X2%,YP%-Y2%+1
580 MOVE XP%-X1%+1,YP%-Y1%+1
590 DRAW XP%-X2%+1,YP%-Y2%+1
600 MOVE XP%-X1%+1,YP%+Y1%
610 DRAW XP%-X2%+1,YP%+Y2%
620 '
630 X1%=X2%
640 Y1%=Y2%
650 '
660 NEXT
670 '
680 GOTO 280
690 '
700 'KOORDINATEN BERECHNEN
710 '
720 X2%=INT(COS(J%*6)*D%+0.5)
730 Y2%=INT(SIN(J%*6)*D%+0.5)
740 '
750 RETURN
```

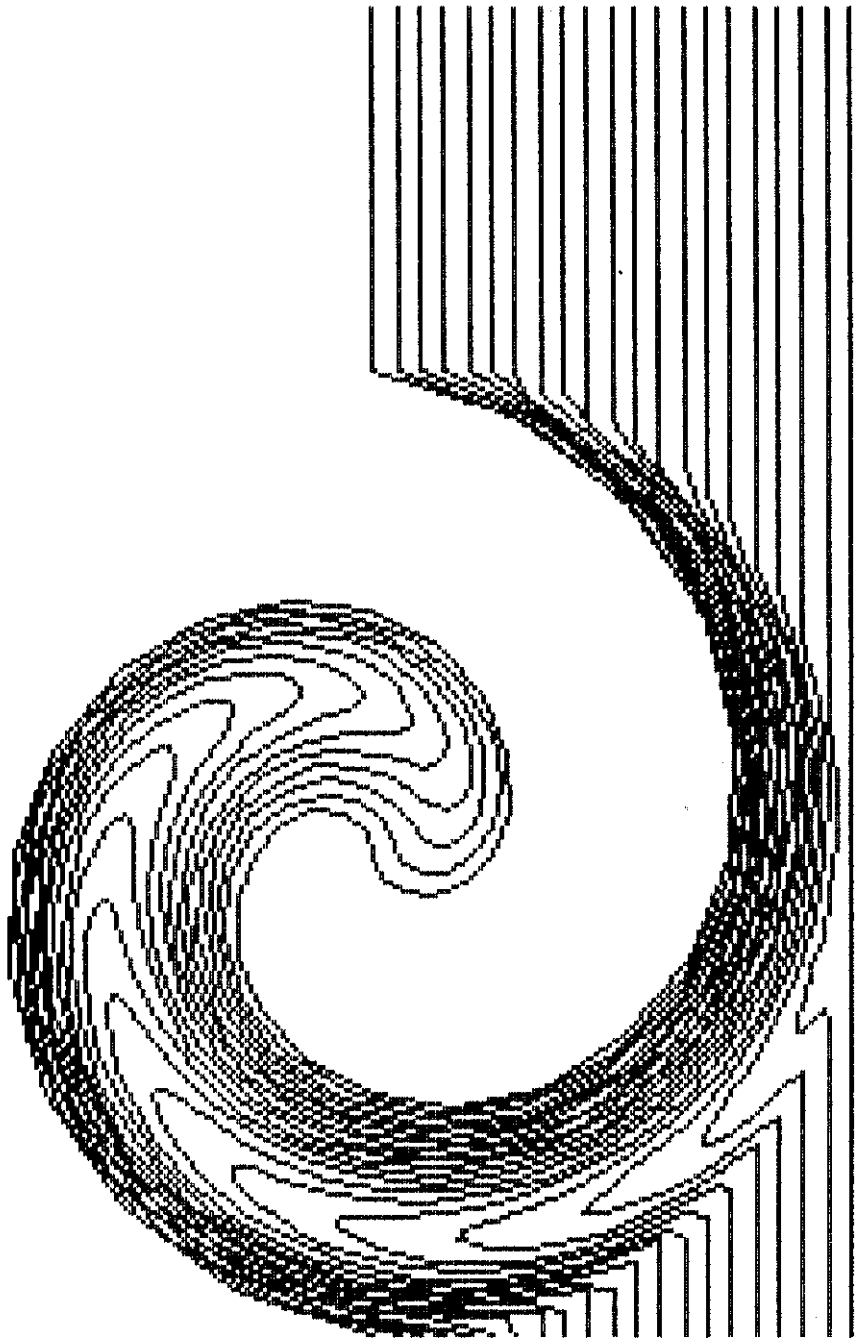
4.2 Horn

```
100 '************************************************************
110 '***                                                      ***
120 '***  HORN - ABSTRAKTE KUNST IN BASIC   JST 21.6.1986  ***
130 '***                                                      ***
140 '************************************************************
150 '
160 MODE 1
170 '
180 BORDER 0
190 INK 0,0
200 INK 1,26
210 '
220 FOR I%=0 TO 3000
230 '
240 A=I%*PI/3000
250 B=120*(0.5+COS(A)/2)
260 C=A*2
270 D=C*150
280 E=B*COS(D)
290 F=B*SIN(D)
300 G=240+120*SIN(C)+F
310 X%=INT(1.3*G)
320 Y%=INT(240+120*COS(C)+E)
330 Y%=Y%-50
340 '
350 IF I%=0 THEN MOVE X%,Y%
360 IF I%>0 THEN DRAW X%,Y%
370 '
380 NEXT I%
390 '
400 GOTO 400
```

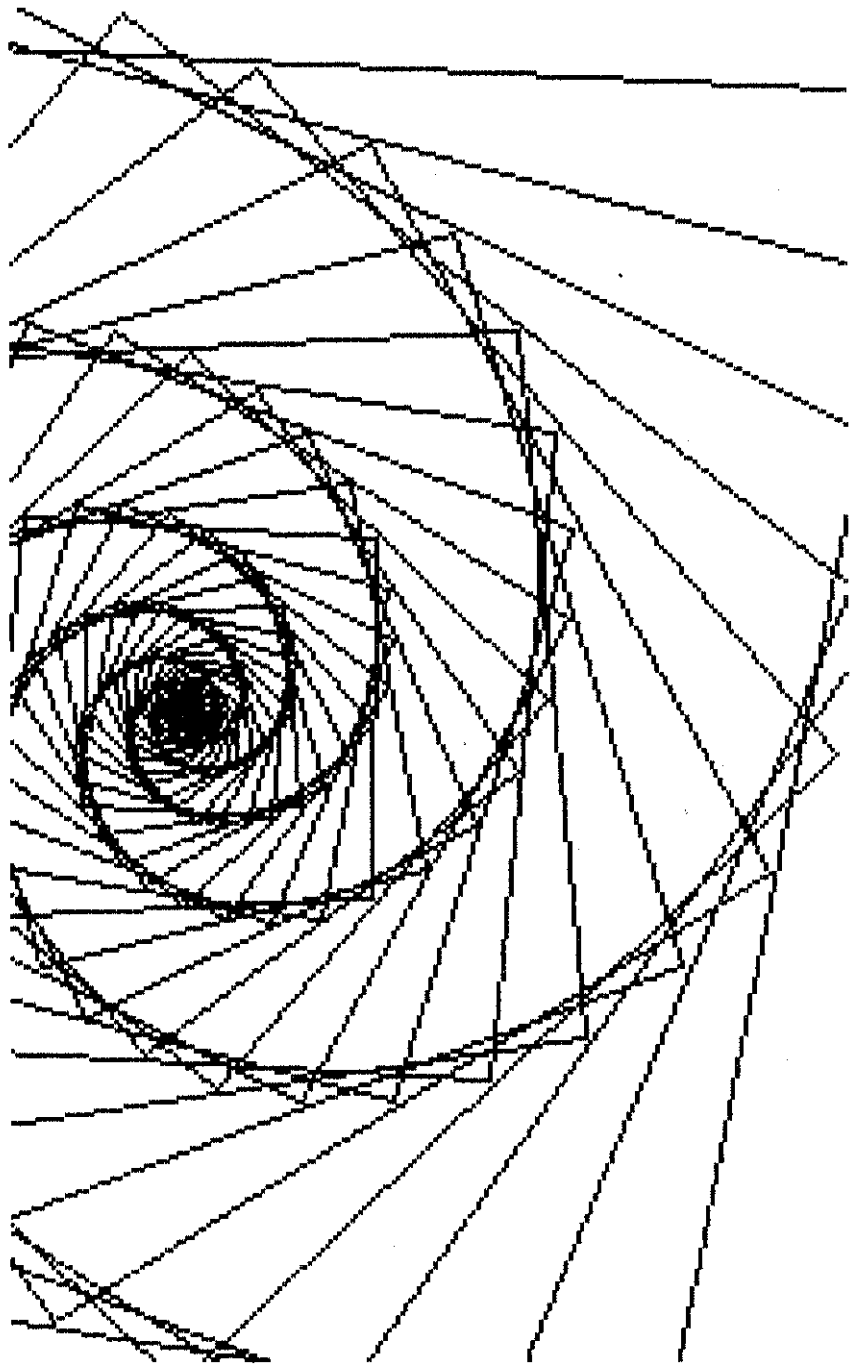
4.3 Wave

```
100 '************************************************************
110 '***                                                      ***
120 '***  WAVE - ABSTRAKTE KUNST IN BASIC   JST 21.6.1986  ***
130 '***                                                      ***
140 '************************************************************
150 '
160 MODE 1
170 '
180 BORDER 0
190 INK 0,0
200 INK 1,26
210 '
220 FOR I%=0 TO 20
230 FOR J%=0 TO 55
240 '
250 X=I%/20-1
260 Y=J%/20-1
270 D=SQR(X*X+Y*Y)
280 IF X<>0 THEN A=ATN(Y/X)
290 IF X=0 THEN A=PI/2*SGN(Y)
300 IF X<0 THEN A=A+PI
310 IF D<1 THEN A=A+PI*2*(1-D)
320 B=D*SIN(A)
330 C=D*COS(A)
340 E=1+0.95*B
350 F=1+0.95*C
360 X%=INT(449/2*E)
370 X%=X%+15
380 Y%=INT(449/2*F)
390 IF J%=0 THEN MOVE X%,Y%
400 IF J%>0 THEN DRAW X%,Y%
410 '
420 NEXT J%
430 NEXT I%
440 '
450 GOTO 450
```

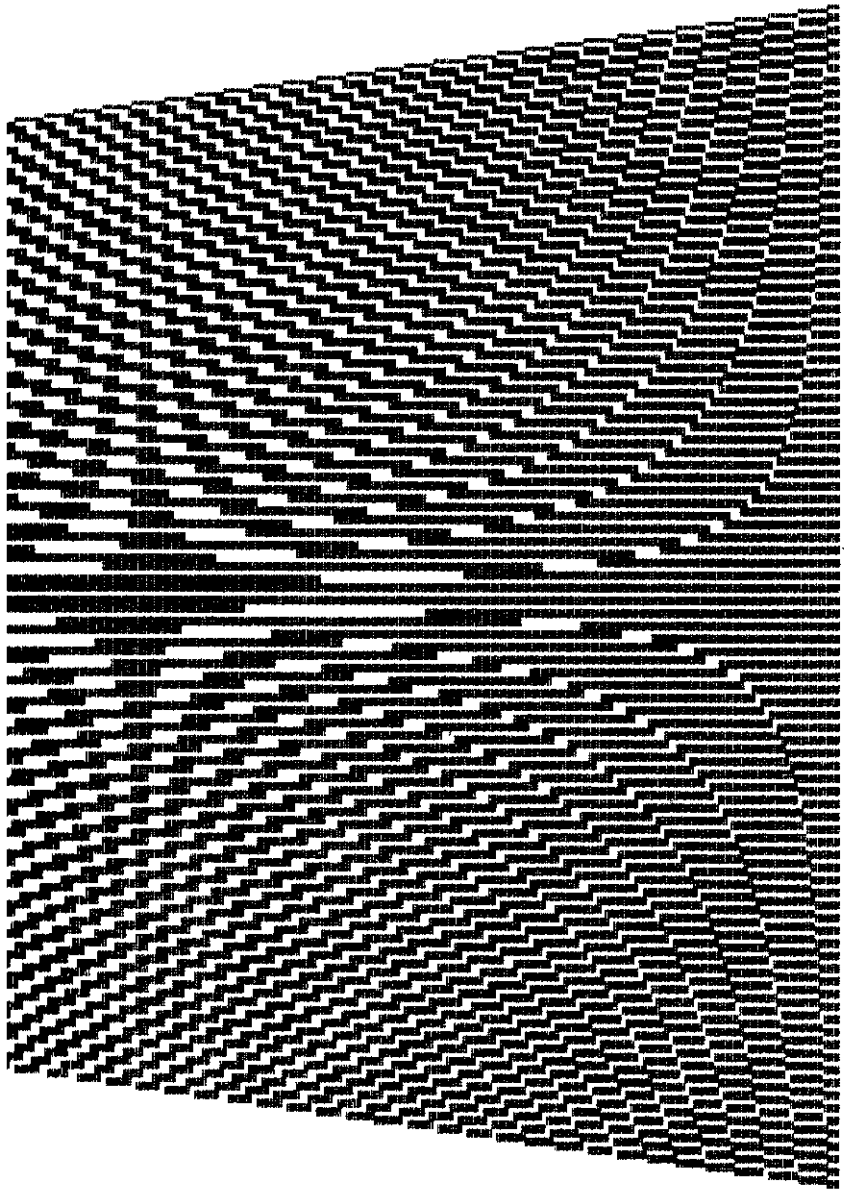
4.3 Tiefe

```
100 '************************************************************
110 '***                                                      ***
120 '***  TIEFE - ABSTRAKTE KUNST IN BASIC  JST 21.6.1986  ***
130 '***                                                      ***
140 '************************************************************
150 '
160 MODE 1
170 '
180 INK 0,0
190 INK 1,26
200 BORDER 0
210 '
220 A=0
230 B=600
240 '
250 FOR i=0 TO 300
260 '
270 X=X+B*COS(A)
280 Y=Y+B*SIN(A)
290 X%=INT(X)
300 Y%=INT(Y)
310 '
320 IF I=0 THEN MOVE X%,Y%
330 IF I>0 THEN DRAW X%,Y%
340 '
350 A=A+1.52
360 B=B*0.98
370 '
380 NEXT I
390 '
400 GOTO 400
```

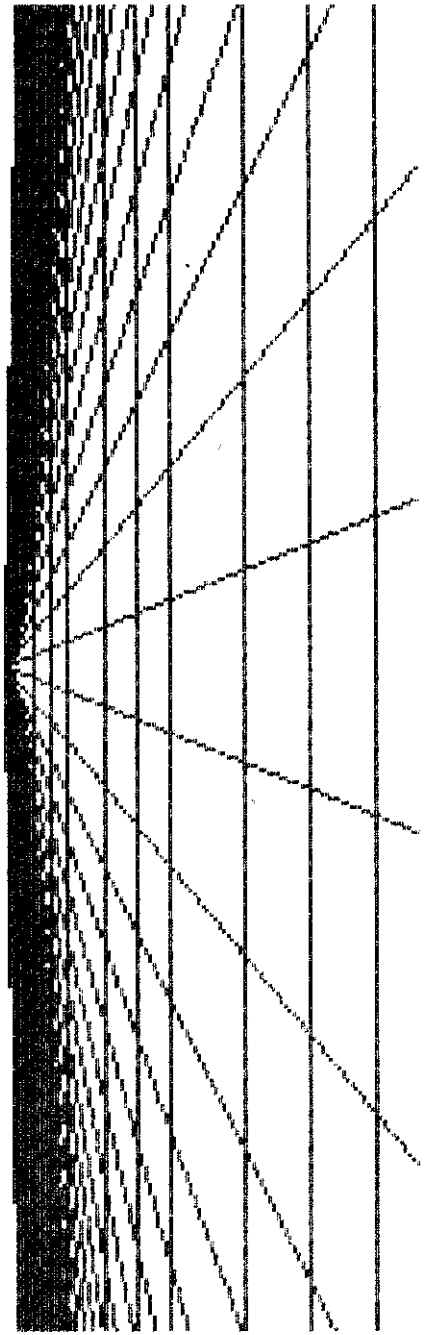
4.4 Interferenzmuster

```
100 '************************************************************
110 '***                                                      ***
120 '***   INTERFERENZMUSTER - ABSTRAKTE KUNST IN BASIC       ***
130 '***                    JST 14.6.1986                     ***
140 '***                                                      ***
150 '************************************************************
160 '
170 MODE 1
180 '
190 N=0
200 X1=40
210 Y1=0
220 X2=600
230 Y2=4000
240 '
250 X1=X1+3
260 X2=X2-3
270 '
280 IF X1>600 THEN X1=41:N=N+1
290 IF X2<40 THEN X2=599
300 '
310 PLOT X1,Y1,1
320 PRINT CHR$(23);CHR$(1);
330 DRAW X2,Y2,1
340 '
350 IF N=1 THEN N=2
360 IF X1=41 THEN INK 1,N
370 '
380 GOTO 250
```

4.5 Gridrunner

```
100 '************************************************************
110 '***                                                      ***
120 '***                     GRIDRUNNER                       ***
130 '***                   TAV 20.10.1986                     ***
140 '***                                                      ***
150 '************************************************************
160 '
170 '
180 MODE 0
190 BORDER 0
200 INK 0,0
210 INK 1,9
220 FOR I=2 TO 15
230 INK I,0
240 NEXT
250 FOR I=2 TO 15
260 J=196-I*2
270 FOR K=1 TO 3
280 MOVE 0,J,I
290 DRAW 639,J,I
300 J=J-28*K-(I*2-4)*K
310 NEXT
320 NEXT
330 FOR I=-10000 TO 10000 STEP 160
340 MOVE 320,200
350 DRAW I,0,1
360 NEXT
370 FOR I=2 TO 15
380 INK I,9
390 CALL &BD19
400 INK I,0
410 NEXT
420 GOTO 370
```

5. Ein Bild sagt mehr als tausend Worte - Geschäftsgrafiken

Versuchen Sie sich einmal in die Lage eines Konzernchefs zu versetzen, der seinen Filialleitern klar machen muß, daß das Unternehmen nicht mehr genug Umsatz erwirtschaftet und deshalb größere Anstrengungen gemacht werden müssen, die Umsatzzahlen wieder in die Höhe zu schrauben. Zum Zwecke dieser Mitteilung hat er alle Filialleiter zu einer Konferenz zusammengerufen. Nun sitzen die Herren alle in Erwartung großer Ereignisse beisammen, und der Konzernchef beginnt seinen Vortrag über die Umsatzlage des Unternehmens. Nach einer eindringlichen Vorrede kommt er zu den Fakten: "Im Jahre 1975 erreichten wir ein Umsatzvolumen von 13,53 Milliarden Mark, 1976 waren es 3,4 Milliarden mehr und 1977 wurde schließlich ein Umsatz von nahezu 19 Milliarden Mark..."

Die Rede setzt sich noch über einige Zeit fort, doch spätestens an dieser Stelle hat auch der wohlwollende Zuhörer gänzlich die Übersicht verloren und schweift mit seinen Gedanken zu fernen Zielen.

Sicher kann es nicht im Sinne eines guten Geschäftsmannes sein, daß es unmöglich ist, seinen Ausführungen über Umsatzzahlen zu folgen. Doch welche Alternative hat unser Konzernchef? Im Grunde bieten sich ihm mehrere Möglichkeiten, und eine davon liegt geradezu auf der Hand, oder genauer gesagt, vor ihm auf dem Tisch. Da er natürlich auch nicht mehr die Umsätze im Kopf hat, die Jahre zurückliegen, hat er diese Zahlen in Form einer Tabelle von der Buchhaltungsabteilung zusammenstellen lassen.

Diese Tabelle vervielfältigt und an die Zuhörer verteilt könnte dem Konzernchef viele Worte ersparen und würde die Übersichtlichkeit der zahlreichen Daten erhöhen. Jedem Zuhörer wäre es möglich, sich unmittelbar über die Ertragslage eines bestimmten Jahres zu orientieren, und der Vortrag könnte sich auf Trends und Tendenzen beschränken.

Auch diese tabellarische Form, Zahlen dazustellen, hat einige Nachteile. Zahlen sind etwas Abstraktes, und aus diesem Grunde kann man sie sich nur schwer vorstellen. Müssen dann auch noch viele Zahlen in Relation gebracht werden, so wird auch hier die Übersichtlichkeit sehr stark gemindert.

Umsatzzahlen von 1975 bis 1985

1975:	13,53 Mrd. DM
1976:	16,93 Mrd. DM
1977:	18,97 Mrd. DM
1978:	23,21 Mrd. DM
1979:	27,47 Mrd. DM
1980:	28,90 Mrd. DM
1981:	31,23 Mrd. DM
1982:	32,31 Mrd. DM
1983:	32,98 Mrd. DM
1984:	30,26 Mrd. DM
1985:	27,45 Mrd. DM

Sicher, aus der oben stehenden Tabelle ist ersichtlich, daß die aufstrebende Umsatzlage des Unternehmens in den letzten beiden Jahren rückläufig geworden ist, doch noch um einiges deutlicher wird dieser Tatbestand, wenn man die Tabelle in grafischer Form darstellt. Dazu kann beispielsweise für jedes der Jahre 1975 bis 1985 ein Balken gezeichnet werden, wobei die Länge des Balkens durch die Größe der Zahl bestimmt wird, die er repräsentiert. Abbildung 7 zeigt, wie die grafische Umsetzung der Erträge in einem Balkendiagramm aussehen kann.

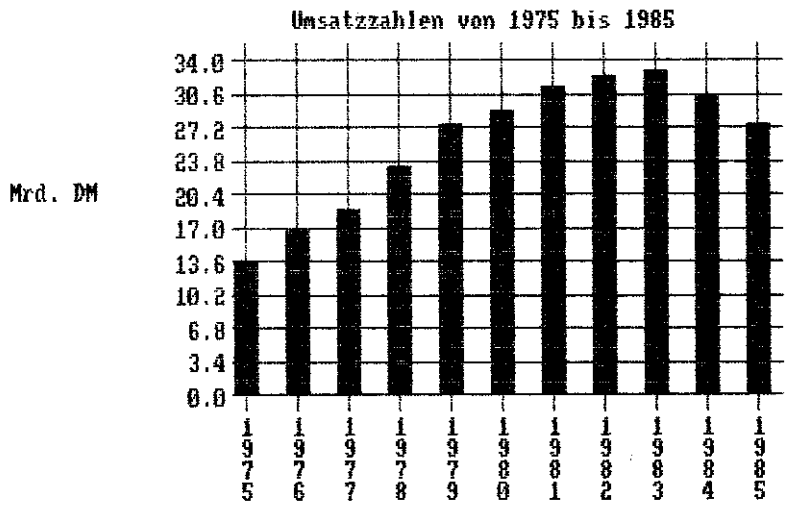


Abb. 7: Umsetzung in ein Balkendiagramm

Durch den Vergleich der Balken kann die Umsatzentwicklung unmittelbar abgelesen werden, und unser Konzernchef hat eine schlagkräftige Argumentationshilfe, mit der er auch den letzten Zweifler davon überzeugen kann, daß es nicht rosig um die Erträge des Unternehmens bestellt ist.

Es gibt noch einige andere Verfahren, Zahlenkolonnen in aussagekräftige Grafiken zu verwandeln, aber am Beispiel der Balkengrafik ist doch recht deutlich zu sehen, wieviel transparenter eine Grafik im Vergleich zu nackten Zahlen ist - dies gilt nicht unbedingt nur in der Geschäftswelt.

5.1 Das Punktediagramm

Als erste Darstellungform der Geschäftsgrafik wollen wir Ihnen das sogenannte Punktediagramm vorstellen, bei dem die Stellen, an denen sich die darzustellenden Werte befinden, durch Punkte bezeichnet werden. Dieses Diagramm haben wird aus dem

Grunde an den Anfang unserer Betrachtungen gestellt, weil bei ihm eine elementare Umsetzung des Zahlenmaterials in Grafik erfolgt, ohne zusätzliche Aufbereitung wie bei den später vorgestellten Diagrammformen.

Der Aufbau eines Geschäftsdiagramms ist immer gleich - es besitzt zwei Achsen, die zueinander senkrecht stehen. Auf einer der Achsen befindet sich eine Skala, die anzeigt, woher das Zahlenmaterial stammt, auf der anderen eine Einteilung in Größenordungen. Die Einteilung der ersten Achse kann in Jahre, Monate, Filialen oder dergleichen erfolgen, die der zweiten Achse in Stückzahlen, Geldbeträge oder ähnlich. Das Zahlenmaterial, das dargestellt werden soll, bestimmt dabei die Form der Einteilung. Um nicht allzu theoretisch zu bleiben, wollen wir das Beispiel eines Privatmannes zur Hilfe nehmen, der sich einen Überblick über die Kilometerleistung seines Kraftfahrzeuges im Jahre 1985 machen will. Er hat zu diesem Zweck die zurückgelegten Kilometer für jeden einzelnen Monat in einer Tabelle zusammengestellt.

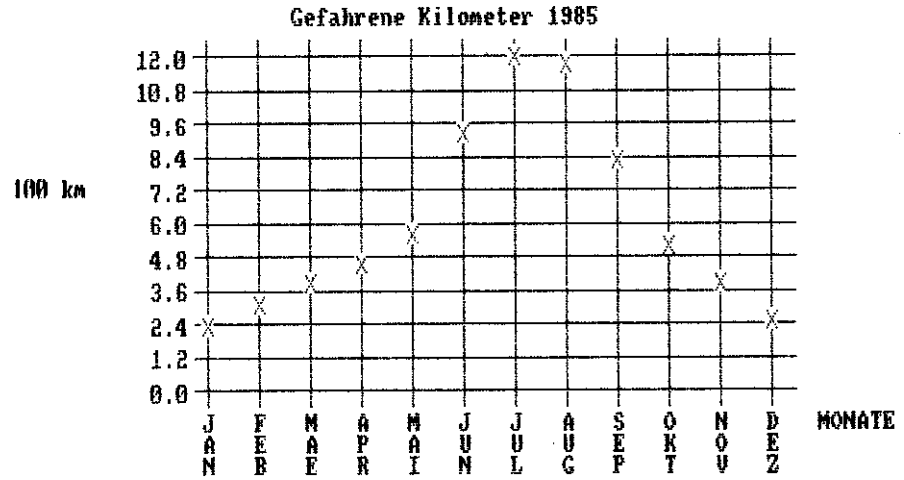
Gefahrene Kilometer 1985

Januar:	230 km
Februar:	310 km
März:	390 km
April:	450 km
Mai:	560 km
Juni:	920 km
Juli:	1200 km
August:	1170 km
September:	830 km
Oktober:	520 km
November:	390 km
Dezember:	250 km

Um nun das zugrundeliegende Zahlenmaterial in ein Punktediagramm umzusetzen, müssen zunächst einmal die beiden angesprochenen Achsen eingeteilt werden. Die Einteilung der ersten

Achse erfolgt sinnvollerweise in die zwölf Monate des Jahres 1985. Die zweite Achse ist in Kilometer einzuteilen, wobei als größter Einteilungspunkt die maximal pro Monat zurückgelegten Kilometer veranschlagt werden sollten - im Falle unseres Beispiels sind das 1200 km. Da sich alle darzustellenden Entfernungen in der Größenordung von einigen hundert Kilometern befinden, kann die Einteilung in 100km-Einheiten vorgenommen werden.

Ist das Diagramm soweit vorbereitet, ist das Eintragen der Punkte leicht gemacht. Aus der Tabelle kennen wir die Werte für die einzelnen Monate, so daß wir für jeden Monat die Stelle der Kilometerskala markieren können, an der sich der betreffende Wert befindet. Diese Arbeit für alle Monate von Januar bis Dezember durchgeführt - und das Punktediagramm ist fertig. Damit es auch später noch von Nutzen sein kann, ist es sinnvoll, das Diagramm zu beschriften. Zur Beschriftung gehört sowohl eine Überschrift, die Mitteilung darüber macht, um was für ein Diagramm es sich handelt, als auch eine Beschriftung der beiden Achsen.



Hardcopy 7: Punktediagramm

Das gerade abgedruckte Punktediagramm wurde mit einem Programm erstellt, das Ihnen jegliche Mühen beim Erstellen eines Diagramms abnimmt. Das einzige, was Sie brauchen ist, eine Tabelle mit Werten, die Sie in das Programm eingeben können. Achsen zeichnen, beschriften und die Umsetzung der Werte übernimmt das Programm.

Nach dem Start des Programms wird der Benutzer danach befragt, welche Überschrift das Diagramm erhalten soll - in unserem Beispiel lautete die Überschrift: "Gefahrene Kilometer 1985" -, und welche Bezeichnung der Werteskala vorgesehen ist. Sind diese Angaben gemacht, erfragt das Programm vom Benutzer die Zahlenwerte für die einzelnen darzustellenden Monate (die eingegebenen Werte dürfen nicht kleiner Null sein).

Bevor das Programm mit dem Zeichnen des Diagramms beginnt, wird aus dem maximalen Wert eines Monats automatisch die Einteilung der Werteskala errechnet. Bei dieser automatischen Einteilung kann es schon einmal vorkommen, daß sich recht "krumme" Werte ergeben, das läßt sich aber um den Preis der Automatisierung nicht verhindern.

Nachdem das Programm die beiden Achsen und deren Einteilung, die sich in Form von Hilfslinien über das ganze Diagramm fortsetzen und eine bessere Lesbarkeit des Diagramms erwirken, gezeichnet hat, beginnt es mit dem Eintragen der Werte in das Diagramm. Dazu werden ganz einfach die Stellen, an denen sich die Werte der Monate befinden, durch ein Kreuz markiert. In der Mitte des Kreuzes befindet sich dann der betreffende Wert.

```
100 MODE 2
110 BORDER 0
120 INK 0,0
130 INK 1,11
131 SYMBOL 255,0,65,34,20,8,20,34,65
140 DIM MY(12)
150 MA=0
```



```
160 '
170 'PROGRAMMKOPF AUSGEBEN
180 '
190 LOCATE 1,3
200 PRINT TAB(33);"PUNKTEDIAGRAMM"
210 PRINT TAB(33);"JST  28.7.1986"
220 MOVE 16,383
230 DRAW 16,320
240 DRAW 623,320
250 DRAW 623,383
260 DRAW 16,383
270 '
280 'BESCHRIFTUNGEN EINLESEN
290 '
300 LOCATE 36,8
310 PRINT"|----------------------------------|"
320 LOCATE 1,8
330 INPUT"UEBERSCHRIFT (MAX. 40 BUCHSTABEN)";UE$
340 LOCATE 36,10
350 PRINT"|--------|"
360 LOCATE 1,10
370 INPUT"WERTESKALA   (MAX. 10 BUCHSTABEN)";WE$
380 '
390 'EINLESEN DER EINZELNEN WERTE UND MAXIMALWERT ERMITTELN
400 '
410 PRINT
420 PRINT"GEBEN SIE JETZT DIE WERTE FUER DIE EINZELNEN MONATE EIN!
"
430 PRINT
440 '
450 FOR I=1 TO 12
460 READ MONAT$
470 PRINT USING "\       \";MONAT$;
480 INPUT MY(I)
490 IF MA<MY(I) THEN MA=MY(I)
500 NEXT I
510 '
520 CLS
530 '
```

```
540 'POSITION FUER BESCHRIFTUNG ERMITTELN
550 '
560 T1=LEN(UE$)
570 T2=LEN(WE$)
580 PU=40-(T1/2)
590 PW=6-(T2/2)
600 '
610 'BESCHRIFTUNG AUSGEBEN
620 '
630 LOCATE 69,20
640 PRINT "MONATE"
650 LOCATE PU,1
660 PRINT UE$
670 LOCATE PW,9
680 PRINT WE$
690 '
700 'KOORDINATENSYSTEM ZEICHNEN
710 '
720 MOVE 163,375
730 DRAW 163,112
740 DRAW 530,112
750 '
760 'SKALIERUNG DER X-ACHSE
770 '
780 FOR X=163 TO 515 STEP 32
790 MOVE X,100
800 DRAW X,375
810 NEXT X
820 '
830 'BESCHRIFTUNG DER X-ACHSE
840 '
850 FOR MY=1 TO 3
860 MY1=MY+19
870 RESTORE
880 FOR M=1 TO 12
890 MX=M*4+17
900 READ MONAT$
910 MONAT$=RIGHT$(LEFT$(MONAT$,MY),1)
920 LOCATE MX,MY1
930 PRINT MONAT$
```



```
940 NEXT M
950 NEXT MY
960 '
970 'SKALIERUNG DER Y-ACHSE
980 '
990 FOR Y=362 TO 112 STEP -25
1000 MOVE 155,Y
1010 DRAW 530,Y
1020 NEXT Y
1030 '
1040 'BESCHRIFTUNG DER Y-ACHSE
1050 '
1060 IF INT(MA)<>MA THEN MA=INT(MA)+1
1070 IF INT(MA/2)<>MA/2 THEN MA=MA+1
1080 '
1090 TAG
1100 '
1110 FOR I=0 TO 10
1120 PY=25*I+117
1130 MOVE 95,PY
1140 PRINT USING "#####.#";MA/10*I;
1150 NEXT I
1160 '
1170 'PUNKTE SETZEN
1180 '
1190 F=131
1200 FOR I=1 TO 12
1210 MY=((MY(I)/MA)*250)+112
1220 F=F+32
1240 MY=MY+8
1280 MOVE F-4,MY
1290 PRINT CHR$(255);
1300 NEXT I
1310 '
1320 GOTO 1320
1330 '
1340 'DATAS MIT MONATSNAMEN
1350 '
1360 DATA JANUAR,FEBRUAR,MAERZ,APRIL,MAI,JUNI
1370 DATA JULI,AUGUST,SEPTEMBER,OKTOBER,NOVEMBER,DEZEMBER
```

Programmbeschreibung:

100-150 Definitionen, Dimensionierungen und Vorbelegungen. MODE 2 und eine kontrastreiche Farbkombination werden eingestellt. Unter der Zeichennummer 255 wird ein neues Zeichen definiert, das das Aussehen eines Kreuzes hat, wie man es von Wahlzetteln kennt. Dieses Zeichen eignet sich besonders, um einen bestimmten Punkt zu markieren. Im Programm "Punktediagramm" soll es die Stellen kennzeichnen, an denen die Werte für die einzelnen Monate auf der Skala liegen.

Die indizierte Variable MY wird auf 13 Elemente dimensioniert. Unter den Indizes 1 bis 12 sollen die Werte für die zwölf Monate gespeichert werden. Die Variabe MA, in der im Verlauf des Programms der Maximalwert für die Monate entstehen soll, wird mit 0 vorbelegt.

160-260 **Programmkopf ausgeben**
Der Name und die Eignerkennung des Programms werden auf dem Bildschirm ausgegeben und umrandet.

270-370 **Beschriftungen einlesen**
Überschrift und Beschriftung der Werteskala des zu erzeugenden Diagramms werden mit der INPUT-Funktion eingelesen. Der für die Eingaben zur Verfügung stehende Platz wird auf dem Bildschirm gekennzeichnet. Bei Eingaben, die den gekennzeichneten Raum überschreiten, findet keine Korrektur statt.

380-520 **Einlesen der einzelnen Werte und Maximalwert ermitteln**
Nach der Meldung, daß nun die Werte für die darzustellenden Monate eingegeben werden sollen,

beginnt das Programm mit der Abarbeitung einer FOR-TO-NEXT-Schleife. Innerhalb dieser Schleife, die genau 12mal durchlaufen wird, werden Eingaben für die einzelnen Monate vom Benutzer erfragt. Hierzu wird bei jedem Schleifendurchlauf der Inhalt von MONAT$ mit einem der Monatsnamen gefüllt, die im DATA-Block am Ende des Programms stehen.

Nach der Ausgabe des Monatsnamen mit einem für alle Monate einheitlichen Format wird der Wert erwartet, der für die grafische Darstellung des betreffenden Monats zugrunde gelegt werden soll. In Zeile 490 wird der aktuelle Inhalt von MY(I) mit dem von MA verglichen und, sollte MA den kleineren Wert enthalten, zugewiesen. Nach dem Durchlauf der FOR-TO-NEXT-Schleife steht in MA der Maximalwert aller zwölf Monate.

Nachdem alle Angaben gesammelt sind, wird der Bildschirm für das Erstellen des Diagramms gelöscht.

530-590 **Position fuer Beschriftung ermitteln**
Die Länge der Überschrift und der Beschriftung der Werteskala werden mit der Funktion LEN ermittelt und den Variablen T1 und T2 zugewiesen. Der durch 2 dividierte Inhalt dieser Variablen wird jeweils von der X-Position subtrahiert, die später genau in der Mitte der Beschriftung liegen wird. Auf diese Weise kann auf die X-Position des Textcursors geschlossen werden, an der mit der Ausgabe der Texte begonnen werden muß, um sie an der gewünschten Position zentriert erscheinen zu lassen. Die X-Koordinaten für den Start der Textausgaben befinden sich in den Variablen PU (Überschrift) und PW (Werteskala).

600-680 **Beschriftung ausgeben**
Die Beschriftung für das Diagramm wird auf dem Bildschirm ausgegeben, wobei die errechneten Positionen für die Textausgabe Verwendung finden.

690-740 **Koordinatensystem zeichnen**
Mit einem MOVE- und zwei DRAW-Kommandos wird das Koordinatensystem gezeichnet, in dem das Diagramm ruhen wird.

750-810 **Skalierung der X-Achse**
Nach dem Durchlauf dieses Programmteils ist das Koordinatensystem von der X-Achse aus durch 12 senkrechte Linien aufgeteilt. Diese an den zwölf Monaten orientierte Unterteilung wird zur besseren Übersichtlichkeit des Diagramms durchgeführt.

820-950 **Beschriftung der X-Achse**
Innerhalb dieses Programmteils werden die ersten drei Buchstaben der Monatsnamen senkrecht untereinander an die für sie bestimmten Positionen der X-Achse geschrieben. Diese Aufgabe wird von zwei ineinander verschachtelten FOR-TO-NEXT-Schleifen gelöst.

Die äußere Schleife hat genau drei Durchläufe, von denen jeder einen der drei Buchstaben der Monatsnamen behandelt. Aus der Indexvariablen dieser Schleife (MY) wird in Zeile 860 die Y-Position für die Ausgabe der Buchstaben des aktuellen Schleifendurchlaufs ermittelt. Vor dem Eintritt in die innere FOR-TO-NEXT-Schleife wird der DATA-Zeiger zurückgesetzt und damit ein Auslesen der Monatsnamen bei jeden Schleifendurchlauf ermöglicht.

Jeder der 12 Durchläufe der inneren Schleife handelt einen Monat ab. Zunächst wird in der Schleife aus der Indexvariablen M die X-Koordinate für die Ausgabe des Buchstaben ermittelt und MX zugewiesen. Damit steht die Position für die Textausgabe in den Variablen MX und MY1. Welcher Buchstabe auszugeben ist, wird in den Zeilen 900 und 910 bestimmt.

Aus dem Monatsnamen, der dem DATA-Block entnommen wird, isoliert die Programmzeile 910 den entsprechenden Buchstaben. Dazu werden entsprechend der Indexvariablen MY der erste, die ersten beiden oder die ersten drei Buchstaben des Monatsnamen abgetrennt (LEFT$(MONAT$,MY)) und der jeweils letzte für die Ausgabe bestimmt. Es erfolgt die Positionierung des Textcursors auf die ermittelte Stelle und die Ausgabe des isolierten Buchstaben, der sich in der Variablen MONAT$ befindet.

960-1020 **Skalierung der Y-Achse**
Nach dem Durchlauf dieses Programmteils ist das Koordinatensystem von der Y-Achse aus durch 11 waagerechte Linien aufgeteilt. Die unterste Linie (X-Achse) steht für den Nullpunkt der Werteskala. Alle weiteren Unterteilungen dienen zur besseren Zuordnungsmöglichkeit der Angaben der Werteskala zu den Punkten des Diagramms und sind deshalb, wie auch die senkrechten Unterteilungen, über das gesamte Diagramm durchgeführt.

1030-1150 **Beschriftung der Y-Achse**
Der erste Schritt dieses Programmteils ist die Aufbereitung des Maximalwertes (MA), der sich bei der Eingabe der Werte für die einzelnen Monate ergeben hat. Zeile 1060 trennt eventuell vorhandene Nachkommastellen ab und erhöht den Maximalwert um eins, falls es zu einem Abtrennen kommt. Dies geschieht, damit sich keine Werte ergeben können, die größer sind als die größte Einteilung der Skala. Sollte der Maximalwert eine ungerade Zahl sein, so wird sie in Zeile 1070 inkre-

mentiert, um eine allzu komplexe Einteilung der Werteskala auszuschließen.

Nach dem Koppeln der Textausgabe mit dem Grafikcursor (TAG) wird in einer FOR-TO-NEXT-Schleife die Beschriftung der Y-Achse vorgenommen. Bei jedem der 11 Schleifendurchläufe wird zuerst die Position ermittelt, an der die Ausgabe der Beschriftung für den aktuellen Durchlauf erfolgen soll, und der Variablen PY zugewiesen. Die Beschriftung wird von unten nach oben vorgenommen, wobei unter Berücksichtigung der Formatangabe das Zehntel des Maximalwertes multipliziert mit dem Schleifenindex ausgegeben wird.

1160-1320 **Punkte setzen**
In einer FOR-TO-NEXT-Schleife werden die Kreuze, die das eigentliche Diagramm ausmachen, gezeichnet. In Zeile 1210 wird zunächst aus dem Wert für den Monat, der sich durch die Indizierung ergibt, unter Einbeziehung des Maximalwertes und zweier Konstanten, die Y-Position für den Mittelpunkt des Kreuzes ermittelt (MY).

Die X-Position für das Kreuz ergibt sich durch Hinzuaddieren von 32 - bei jedem Schleifendurchlauf - zu der Variablen F, die vor der Schleife auf den Defaultwert 131 gesetzt wurde. Die beiden rechnerischen Koordinaten müssen, da kein Punkt, sondern das umdefinierte Zeichen 255 ausgegeben werden soll, so umgewandelt werden, daß sich die Koordinaten der linken oberen Ecke der Zeichenmatrix ergeben.

Für die Y-Position geschieht das in Zeile 1240. Die X-Position wird beim Versetzen des Grafikcursors (Zeile 1280) angepaßt, denn sie darf nicht zurückgeschrieben werden. An der so erreichten Position des Grafikcursors wird das unter Zeichennummer 255 definierte Kreuz auf den Bildschirm gesetzt.

Am Schnittpunkt der beiden Linien des Kreuzes liegt der Wert für den jeweiligen Monat.

In Zeile 1320 befindet sich eine Endlosschleife, die verhindern soll, daß das Diagramm durch die "Ready"-Meldung zerstört wird.

1330-Ende **DATAs mit Monatsnamen**
Hier stehen, zu zwei DATA-Statements zusammengefaßt, die Namen der zwölf Monate, die von den Programmteilen EINLESEN DER EINZELNEN WERTE UND MAXIMALWERT ERMITTELN und BESCHRIFTUNG DER X-ACHSE benötigt werden.

5.2 CPC-Chart - Der Diagrammgenerator

Zahlen in einem Punktediagramm darzustellen ist eine objektive Darstellungsweise, da sie wirklich nur die reinen Fakten vermittelt. Es gibt aber andere Darstellungsformen, die neben der Vorstellung von Tatsachen auch noch eine Wertung des Zahlenmaterials implizieren und so in gewissem Umfang auf den Betrachter Einfluß nehmen. Da ist beispielsweise das Kurvendiagramm, das die einzelnen Werte innerhalb des Diagramms durch einen Linienzug miteinander verbindet. Durch die so entstehende Kurve werden Tendenzen, die das Zahlenmaterial aufweist, besonders stark unterstrichen.

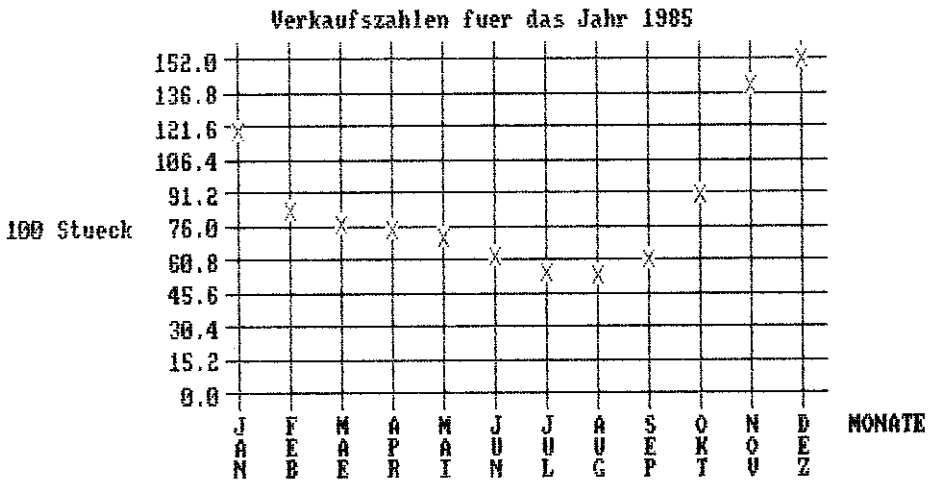
Eine andere Darstellungsform ist das Balkendiagramm, das Sie schon in der Einführung in den Themenkreis der Geschäftsgrafiken kennengelernt haben. Seine Darstellung ergibt sich durch das Zeichnen eines Balkens von dem ermittelten Wert hin zur Achse, auf der die Herkunft des Zahlenmaterials angezeigt ist. Durch die Balken wird den dargestellten Fakten künstlich ein besonderer Nachdruck verliehen.

Doch beurteilen Sie selbst wie unterschiedlich verschiedene Darstellungsformen dasselbe Zahlenmaterial herausstellen. Grundlage für die folgenden drei Diagramme (Punkt-, Kurven-

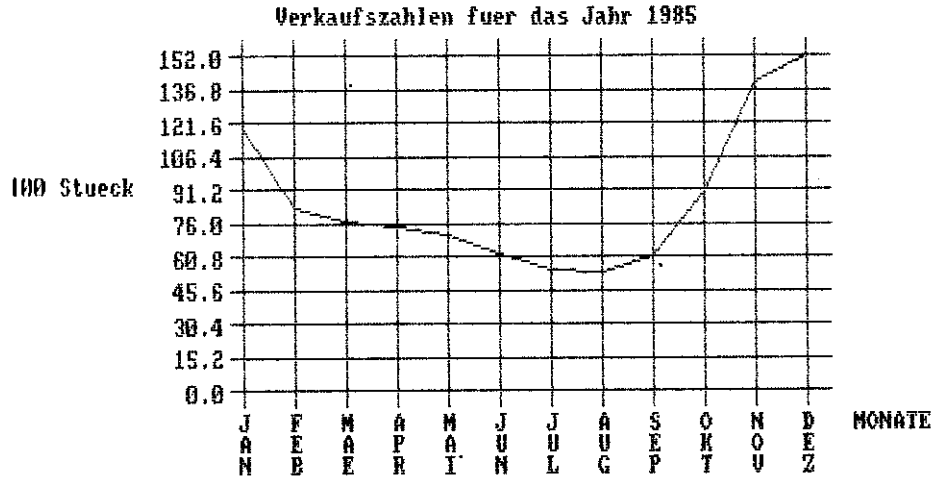
und Balkendiagramm) soll eine Zusammenstellung von Verkaufszahlen sein. Die Aufstellung setzt verkaufte Stückzahlen eines imaginären Produktes in Relation zu den Monaten des Jahres 1985.

Verkaufszahlen für das Jahr 1985

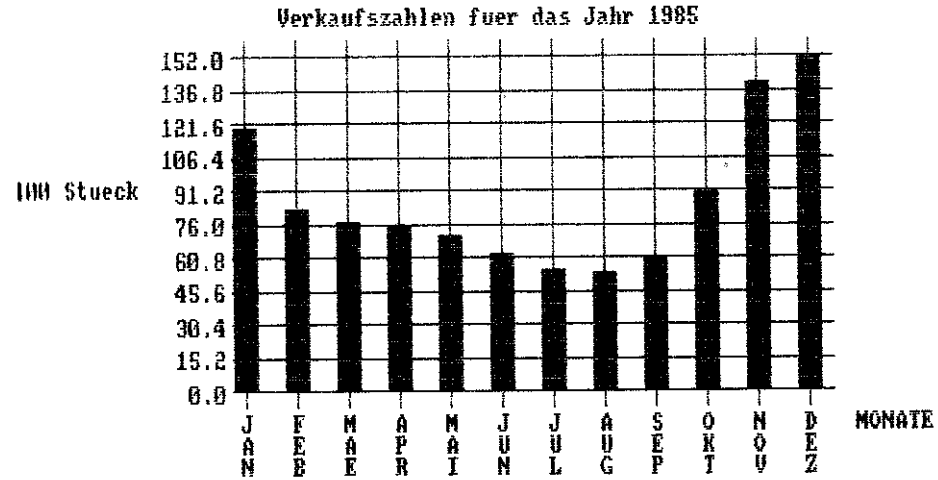
Januar:	12000 Stück
Februar:	8300 Stück
März:	7700 Stück
April:	7400 Stück
Mai:	7100 Stück
Juni:	6200 Stück
Juli:	5500 Stück
August:	5400 Stück
September:	6100 Stück
Oktober:	9000 Stück
November:	14000 Stück
Dezember:	15200 Stück



Hardcopy 8: Punktediagramm zeigt Fakten



Hardcopy 9: Kurvendiagramm unterstreicht Tendenzen



Hardcopy 10: Balkendiagramm verleiht besonderen Nachdruck

Um Ihnen auch die Möglichkeit an die Hand zu geben, in Zukunft Ihr Zahlenmaterial, sei es aus privatem oder geschäftlichem Bereich, mit diesen drei Diagrammformen zu veranschaulichen, haben wir das Programm "CPC-Chart" entwickelt. Es ist in der veröffentlichten Form zwar auf die Darstellung von Werten, die in Beziehung zu den Monaten eines Jahres stehen, zugeschnitten, läßt sich aber auf Grund der ausführlichen Programmbeschreibung sehr leicht an andere Gegebenheiten anpassen.

Die Bedienung von "CPC-Chart" entspricht im groben Rahmen der des bereits bekannten Programms "Punktediagramm". So muß auch hier zunächst die Beschriftung des Diagramms und der Werteskala eingegeben werden, dann folgt das Einlesen der Werte für die einzelnen Monate - auch hier dürfen die eingegebenen Werte nicht kleiner Null sein. Doch dann weicht die Bedienung das Programms von der von "Punktediagramm" ab. An dieser Stelle wird der Benutzer nämlich aufgefordert, eine Darstellungsform zu wählen. Zur Auswahl stehen Punkte-, Kurven- und Balkendiagramm. Ist die Entscheidung über die Art der Darstellung getroffen, so beginnt "CPC-Chart" in bekannter Manier mit dem Erstellen der Grafik. Auch hier werden sämtliche Einteilungen wieder automatisch vorgenommen.

Ist das Diagramm auf dem Bildschirm erstellt, so bietet sich dem Benutzer innerhalb der Grafik ein zweites Menü. Hier kann er wählen, ob er von dem erstellten Diagramm einen Ausdruck auf dem Drucker erzeugen will (Hardcopy), zurück zur Wahl der Darstellungsform möchte oder das Programm beenden will. Hat der Benutzer "Hardcopy" gewählt, so wird die Menüauswahl auf dem Bildschirm gelöscht und eine exakte Kopie des Diagramms auf dem Drucker erzeugt.


```
1000 '************************************************************
1010 '***                                                      ***
1020 '***          CPC-CHART - DER DIAGRAMMGENERATOR           ***
1030 '***                    JST  30.7.1986                    ***
1040 '***                                                      ***
1050 '************************************************************
1060 '
1070 MODE 2
1080 MEMORY &8000
1090 BORDER 0
1100 INK 0,0
1110 INK 1,11
1120 DIM MY(12)
1130 MA=0
1140 '
1150 'GRAFIK-HARDCOPY
1160 '
1170 FOR A=&A000 TO &A0BF
1180 READ D
1190 POKE A,D
1200 Z=Z+D
1210 NEXT A
1220 '
1230 DATA &CD,&BA,&BB,&CD,&E7,&BB,&32,&BD
1240 DATA &A0,&CD,&6C,&A0,&21,&8F,&01,&22
1250 DATA &BE,&A0,&11,&00,&00,&3E,&07,&32
1260 DATA &C0,&A0,&CD,&7C,&A0,&0E,&00,&3A
1270 DATA &C0,&A0,&47,&E5,&D5,&C5,&CD,&F0
1280 DATA &BB,&C1,&D1,&21,&BD,&A0,&BE,&E1
1290 DATA &37,&20,&01,&A7,&CB,&11,&2B,&2B
1300 DATA &10,&E9,&CD,&AF,&A0,&79,&CD,&A6
1310 DATA &A0,&13,&E5,&21,&7F,&02,&37,&ED
1320 DATA &52,&E1,&38,&05,&2A,&BE,&A0,&18
1330 DATA &CC,&23,&7C,&B5,&C8,&2B,&11,&00
1340 DATA &00,&22,&BE,&A0,&3E,&07,&BD,&20
1350 DATA &B9,&7C,&B4,&20,&B5,&3E,&04,&32
1360 DATA &C0,&A0,&18,&AE,&3E,&1B,&CD,&A6
1370 DATA &A0,&3E,&31,&CD,&A6,&A0,&00,&00
1380 DATA &00,&00,&00,&C9,&E5,&3E,&42,&CD
1390 DATA &1E,&BB,&E1,&28,&02,&E1,&C9,&3E
```

```
1400 DATA &0D,&CD,&A6,&A0,&3E,&0A,&CD,&A6
1410 DATA &A0,&3E,&1B,&CD,&A6,&A0,&3E,&4C
1420 DATA &CD,&A6,&A0,&3E,&7F,&CD,&A6,&A0
1430 DATA &3E,&02,&CD,&A6,&A0,&C9,&CD,&2E
1440 DATA &BD,&38,&FB,&CD,&2B,&BD,&C9,&3A
1450 DATA &C0,&A0,&FE,&07,&C8,&AF,&CB,&11
1460 DATA &CB,&11,&CB,&11,&C9,&00,&00,&00
1470 '
1480 IF Z<>23151 THEN PRINT "ERROR IN DATAS":END
1490 '
1500 'PROGRAMMKOPF AUSGEBEN
1510 '
1520 GOSUB 3460
1530 '
1540 'BESCHRIFTUNGEN EINLESEN
1550 '
1560 LOCATE 36,8
1570 PRINT "|------------------------------------|"
1580 LOCATE 1,8
1590 INPUT "UEBERSCHRIFT (MAX. 40 BUCHSTABEN)";UE$
1600 '
1610 LOCATE 36,10
1620 PRINT "|--------|"
1630 LOCATE 1,10
1640 INPUT "WERTESKALA   (MAX. 10 BUCHSTABEN)";WE$
1650 '
1660 'EINLESEN DER EINZELNEN WERTE UND MAXIMALWERT ERMITTELN
1670 '
1680 LOCATE 1,12
1690 PRINT "GEBEN SIE JETZT DIE WERTE FUER DIE EINZELNEN MONATE EI
N!"
1700 PRINT
1710 '
1720 FOR I=1 TO 12
1730 READ MONAT$
1740 PRINT USING "\       \";MONAT$;
1750 INPUT MY(I)
1760 IF MA<MY(I) THEN MA=MY(I)
1770 NEXT I
1780 '
```



```
1790 CLS
1800 '
1810 'PROGRAMMKOPF AUSGEBEN
1820 '
1830 GOSUB 3460
1840 '
1850 LOCATE 1,8
1860 PRINT "WAEHLEN SIE EINE DARSTELLUNGSFORM:"
1870 PRINT
1880 PRINT "PUNKTEDIAGRAMM (P)"
1890 PRINT "KURVENDIAGRAMM (K)"
1900 PRINT "BALKENDIAGRAMM (B)"
1910 LOCATE 1,14
1920 PRINT "IHRE WAHL"
1930 '
1940 DA$=INKEY$
1950 IF DA$<>"P" AND DA$<>"K" AND DA$<>"B" AND DA$<>"p" AND DA$<>"
k" AND DA$<>"b" THEN 1910
1960 '
1970 CLS
1980 '
1990 'POSITION FUER BESCHRIFTUNG ERMITTELN
2000 '
2010 T1=LEN(UE$)
2020 T2=LEN(WE$)
2030 PU=40-(T1/2)
2040 PW=6-(T2/2)
2050 '
2060 'BESCHRIFTUNG AUSGEBEN
2070 '
2080 LOCATE 69,20
2090 PRINT "MONATE"
2100 LOCATE PU,1
2110 PRINT UE$
2120 LOCATE PW,9
2130 PRINT WE$
2140 '
2150 'KOORDINATENSYSTEM ZEICHNEN
2160 '
2170 MOVE 163,375
```

```
2180 DRAW 163,112
2190 DRAW 530,112
2200 '
2210 'SKALIERUNG DER X-ACHSE
2220 '
2230 FOR X=163 TO 515 STEP 32
2240 MOVE X,100
2250 DRAW X,375
2260 NEXT X
2270 '
2280 'BESCHRIFTUNG DER X-ACHSE
2290 '
2300 FOR MY=1 TO 3
2310 MY1=MY+19
2320 RESTORE 2930
2330 '
2340 FOR M=1 TO 12
2350 MX=M*4+17
2360 READ MONAT$
2370 MONAT$=RIGHT$(LEFT$(MONAT$,MY),1)
2380 LOCATE MX,MY1
2390 PRINT MONAT$
2400 NEXT M
2410 '
2420 NEXT MY
2430 '
2440 'SKALIERUNG DER Y-ACHSE
2450 '
2460 FOR Y=362 TO 112 STEP -25
2470 MOVE 155,Y
2480 DRAW 530,Y
2490 NEXT Y
2500 '
2510 'BESCHRIFTUNG DER Y-ACHSE
2520 '
2530 IF INT(MA)<>MA THEN MA=INT(MA)+1
2540 IF INT(MA/2)<>MA/2 THEN MA=MA+1
2550 '
2560 TAG
2570 '
```



```
2580 FOR I=0 TO 10
2590 PY=25*I+117
2600 MOVE 95,PY
2610 PRINT USING "#####.#";MA/10*I;
2620 NEXT I
2630 '
2640 'DARSTELLUNGSFORM
2650 '
2660 IF DA$="P" OR DA$="p" THEN GOSUB 2950
2670 IF DA$="K" OR DA$="k" THEN GOSUB 3110
2680 IF DA$="B" OR DA$="b" THEN GOSUB 3290
2690 '
2700 TAGOFF
2710 '
2720 LOCATE 1,20
2730 PRINT "HARDCOPY (H)"
2740 PRINT "ZURUECK  (Z)"
2750 PRINT "ENDE     (E)"
2760 '
2770 HZE$=INKEY$
2780 IF HZE$<>"H" AND HZE$<>"Z" AND HZE$<>"E" AND HZE$<>"h" AND HZ
E$<>"z" AND HZE$<>"e" THEN 2720
2790 IF HZE$="E" OR HZE$="e" THEN CLS:END
2800 IF HZE$="Z" OR HZE$="z" THEN CLS:GOTO 1790
2810 '
2820 'HARDCOPY ANFERTIGEN
2830 '
2840 LOCATE 1,20
2850 PRINT "              "
2860 PRINT "              "
2870 PRINT "              "
2880 '
2890 CALL &A000
2891 '
2892 GOTO 1790
2900 '
2910 'DATAS MIT MONATSNAMEN
2920 '
2930 DATA JANUAR,FEBRUAR,MAERZ,APRIL,MAI,JUNI
2940 DATA JULI,AUGUST,SEPTEMBER,OKTOBER,NOVEMBER,DEZEMBER
```

```
2950 '
2960 'PUNKTE SETZEN
2970 '
2980 SYMBOL 255,0,65,34,20,8,20,34,65
2990 '
3000 F=131
3010 '
3020 FOR I=1 TO 12
3030 MY=((MY(I)/MA)*250)+112
3040 F=F+32
3050 MY=MY+8
3060 MOVE F-4,MY
3070 PRINT CHR$(255);
3080 NEXT I
3090 '
3100 RETURN
3110 '
3120 'KURVE ZEICHNEN
3130 '
3140 F=131
3150 '
3160 FOR I=1 TO 12
3170 MY=((MY(I)/MA)*250)+112
3180 F=F+32
3190 IF I=1 THEN X2=F:Y2=MY
3200 X1=X2
3210 Y1=Y2
3220 X2=F
3230 Y2=MY
3240 MOVE X1,Y1
3250 DRAW X2,Y2
3260 NEXT I
3270 '
3280 RETURN
3290 '
3300 'BALKEN ZEICHNEN
3310 '
3320 F=131
3330 '
3340 FOR I=1 TO 12
```



```
3350 MY=((MY(I)/MA)*250)+112
3360 F=F+32
3370 '
3380 FOR L=F-7 TO F+7
3390 MOVE L,MY
3400 DRAW L,112
3410 NEXT L
3420 '
3430 NEXT I
3440 '
3450 RETURN
3460 '
3470 'PROGRAMMKOPF AUSGEBEN
3480 '
3490 LOCATE 30,3
3500 PRINT "2-D-GESCHAEFTSGRAFIK"
3510 LOCATE 33,4
3520 PRINT "JST  30.7.1986"
3530 '
3540 'RAHMEN ZEICHNEN
3550 '
3560 MOVE 16,383
3570 DRAW 16,320
3580 DRAW 623,320
3590 DRAW 623,383
3600 DRAW 16,383
3610 '
3620 RETURN
```

Programmbeschreibung:

1000-1130 Definitionen, Dimensionierungen und Vorbelegungen. Der Bildschirm wird in den 80-Zeichen-Modus versetzt, und der RAM-Bereich, der BASIC zu Verfügung steht, durch MEMORY &8000 begrenzt. Oberhalb dieser Grenze wird ein Maschinenprogramm zu Stehen kommen, das durch diese Maßnahme gegen Überschreiben geschützt ist. Eine kontrastreiche Farbkombination wird gewählt und

die indizierte Variable MY auf 13 Elemente dimensioniert. Unter den Indizes 1 bis 12 sollen die Werte für die zwölf Monate gespeichert werden. Die Variabe MA, in der im Verlauf des Programms der Maximalwert für die Monate entstehen soll, wird mit 0 vorbelegt.

1140-1480 **Grafik-Hardcopy**
Dieser Programmteil schreibt eine einfache Hardcopyroutine in den Speicher, mit der die erstellten Diagramme auf einem mit dem Rechner verbundenen Drucker ausgegeben werden können. Die Routine steht ab &A000 im RAM-Speicher und kann mit CALL &A000 aufgerufen werden. Eine genaue Beschreibung dieser Routine finden Sie in Kapitel 8.3.

1490-1520 **Programmkopf ausgeben**
Der Name und die Eignerkennung des Programms werden auf dem Bildschirm ausgegeben und umrandet. Um dies zu erreichen, wird ein Unterprogramm in Zeile 3460 aufgerufen.

1530-1640 **Beschriftungen einlesen**
Überschrift und Beschriftung der Werteskala des zu erzeugenden Diagramms werden mit der INPUT-Funktion eingelesen. Der für die Eingaben zur Verfügung stehende Platz wird auf dem Bildschirm gekennzeichnet.

1650-1790 **Einlesen der einzelnen Werte und Maximalwert ermitteln**
Nach der Meldung, daß nun die Werte für die darzustellenden Monate eingegeben werden sollen, beginnt das Programm mit der Abarbeitung einer FOR-TO-NEXT-Schleife. Innerhalb dieser Schleife, die genau 12mal durchlaufen wird, werden Eingaben für die einzelnen Monate vom Benutzer erfragt. Hierzu wird bei jedem Schleifendurchlauf der Inhalt von MONAT$ mit einem der Monatsnamen gefüllt, die im DATA-Block am Ende des Programms stehen.

Nach der Ausgabe des Monatsnamen, mit einem für alle Monate einheitlichen Format, wird der Wert erwartet, der für die grafische Darstellung des betreffenden Monats zu Grunde gelegt werden soll. In Zeile 1760 wird der aktuelle Inhalt von MY(I) mit dem von MA verglichen und, sollte MA den kleineren Wert enthalten, zugewiesen. Nach dem Durchlauf der FOR-TO-NEXT-Schleife steht in MA der Maximalwert aller zwölf Monate. Der Bildschirm wird für weitere Ausgaben gelöscht.

1800-1970 Nach der erneuten Ausgabe des Programmkopfes (GOSUB 3460) zeigt dieser Programmteil eine Auswahl von drei Menüpunkten auf dem Bildschirm. Durch Betätigen der Tasten "P", "K" oder "B" kann der Benutzer zwischen den Darstellungsformen Punkte-, Kurven und Balkendiagramm wählen. Nach erfolgter Auswahl wird der Bildschirm zum Zeichnen des Diagramms gelöscht.

1980-2040 **Position fuer Beschriftung ermitteln**
Die Länge der Überschrift und der Beschriftung der Werteskala werden mit der Funktion LEN ermittelt und den Variablen T1 und T2 zugewiesen. Der durch 2 dividierte Inhalt dieser Variablen wird von der X-Position subtrahiert, die später genau in der Mitte der Beschriftung liegen wird. Auf diese Weise kann auf die X-Position des Textcursors geschlossen werden, an der mit der Ausgabe der Texte begonnen werden muß, um sie an der gewünschten Position zentriert erscheinen zu lassen. Die X-Koordinaten für den Start der Textausgaben befinden sich in den Variablen PU (Überschrift) und PW (Werteskala).

2050-2130 **Beschriftung ausgeben**
Die Beschriftung für das Diagramm erscheint auf dem Bildschirm, wobei die errechneten Positionen für die Textausgabe Verwendung finden.

2140-2190 **Koordinatensystem zeichnen**
Mit einem MOVE- und zwei DRAW-Kommandos wird das Koordinatensystem gezeichnet, in dem das Diagramm ruhen wird.

2200-2260 **Skalierung der X-Achse**
Nach dem Durchlauf dieses Programmteils ist das Koordinatensystem von der X-Achse aus durch 12 senkrechte Linien aufgeteilt. Diese an den zwölf Monaten orientierte Unterteilung wird zur besseren Übersichtlichkeit des Diagramms durchgeführt.

2270-2420 **Beschriftung der X-Achse**
Innerhalb dieses Programmteils werden die ersten drei Buchstaben der Monatsnamen senkrecht untereinander an die für sie bestimmten Positionen der X-Achse geschrieben. Diese Aufgabe wird von zwei ineinander verschachtelten FOR-TO-NEXT-Schleifen gelöst.

Die äußere Schleife hat genau drei Durchläufe, von denen jeder einen der drei Buchstaben der Monatsnamen behandelt. Aus der Indexvariablen dieser Schleife (MY) wird in Zeile 860 die Y-Position für die Ausgabe der Buchstaben des aktuellen Schleifendurchlaufs ermittelt. Vor dem Eintritt in die innere FOR-TO-NEXT-Schleife wird der DATA-Zeiger auf den DATA-Block mit Monatsnamen zurückgesetzt und damit ein Auslesen der Monatsnamen bei jedem Schleifendurchlauf ermöglicht.

Jeder der 12 Durchläufe der inneren Schleife handelt einen Monat ab. Zunächst wird in der Schleife aus der Indexvariablen M die X-Koordinate für

die Ausgabe des Buchstaben ermittelt und MX zugewiesen. Damit steht die Position für die Textausgabe in den Variablen MX und MY1. Welcher Buchstabe auszugeben ist, wird in den Zeilen 2360 und 2370 bestimmt.

Aus dem Monatsnamen, der dem DATA-Block entnommen wird, isoliert die Programmzeile 2370 den entsprechenden Buchstaben. Dazu werden entsprechend der Indexvariablen MY der erste, die ersten beiden oder die ersten drei Buchstaben des Monatsnamen abgetrennt (LEFT$(MONAT$,MY)) und der jeweils letzte für die Ausgabe bestimmt. Es erfolgt die Positionierung des Textcursors auf die ermittelte Stelle und die Ausgabe des isolierten Buchstaben, der sich in der Variablen MONAT$ befindet.

2430-2490 **Skalierung der Y-Achse**
Nach dem Durchlauf dieses Programmteils ist das Koordinatensystem von der Y-Achse aus durch 11 waagerechte Linien aufgeteilt. Die unterste Linie (X-Achse) steht gleichzeitig für den Nullpunkt auf der Werteskala. Alle weiteren Unterteilungen, dienen zur besseren Zuordnungsmöglichkeit der Angaben der Werteskala zu den Punkten des Diagramms und sind deshalb, wie auch die senkrechten Unterteilungen, über das gesamte Diagramm durchgeführt.

2500-2620 **Beschriftung der Y-Achse**
Der erste Schritt dieses Programmteils ist die Aufbereitung des Maximalwertes (MA), der sich bei der Eingabe der Werte für die einzelnen Monate ergeben hat. Zeile 2530 trennt eventuell vorhandene Nachkommastellen ab und erhöht den Maximalwert um eins, falls es zu einem Abtrennen kommt. Dies geschieht, damit sich keine Werte ergeben können, die größer sind als die größte Einteilung der Skala. Sollte der Maximalwert eine un-

gerade Zahl sein, so wird sie in Zeile 2540 inkrementiert, um eine allzu komplexe Einteilung der Werteskala auszuschließen.

Nach dem Koppeln der Textausgabe mit dem Grafikcursor (TAG) wird in einer FOR-TO-NEXT-Schleife die Beschriftung der Y-Achse vorgenommen. Bei jedem der 11 Schleifendurchläufe wird zuerst die Position ermittelt, an der die Ausgabe der Beschriftung für den aktuellen Durchlauf erfolgen soll, und der Variablen PY zugewiesen. Die Beschriftung wird von unten nach oben vorgenommen, wobei, unter Berücksichtigung der Formatangabe, das Zehntel des Maximalwertes multipliziert mit dem Schleifenindex ausgegeben wird.

2630-2680 **Darstellungsform**
In diesem Programmteil wird, entsprechend der Eingabe bei der Frage nach der Darstellungsform, in die Unterprogramme zum Erstellen eines Punkte- (Zeile 2950), Kurven- (Zeile 3110) oder Balkendiagramms (Zeile 3290) verzweigt.

2690-2800 Befindet sich das gewünschte Diagramm auf dem Bildschirm, so erscheint eine zweite Menüauswahl. Sie bietet die Möglichkeiten, eine Hardcopy des Diagramms, das sich auf dem Bildschirm befindet, anzufertigen, zurück zur Auswahl der Darstellungsform zu gehen oder das Programm zu beenden.

Nach erfolgter Auswahl eines der drei möglichen Punkte wird das Programm beendet (Zeile 2790) oder zur weiteren Verarbeitung nach Zeile 1790 gesprungen. Der Menüpunkt "Hardcopy" wird nicht mit einer IF-Abfrage ausgesondert, sondern immer ausgeführt, wenn keiner der anderen Punkte ausgewählt wurde.

2810-2890 **Hardcopy anfertigen**
Um das Diagramm von überflüssigen Angaben zu befreien, wird in den Zeilen 2840 bis 2870 die Menüauswahl auf dem Bildschirm gelöscht und dann die Hardcopyroutine aufgerufen (CALL &A000). Nach dem Ausdruck der Hardcopy setzt das Programm seine Verarbeitung bei Zeile 1790 fort.

2900-2940 **DATAs mit Monatsnamen**
Hier stehen, zu zwei DATA-Statements zusammengefaßt, die Namen der zwölf Monate, die von den Programmteilen EINLESEN DER EINZELNEN WERTE UND MAXIMALWERT ERMITTELN und BESCHRIFTUNG DER X-ACHSE benötigt werden.

2950-3100 **Punkte setzen**
Nach der Umdefinition des Zeichen 255 in Gestalt eines Kreuzes (Zeile 2980) wird in einer FOR-TO-NEXT-Schleife das Punktediagramm gezeichnet. In Zeile 3030 wird zunächst aus dem Wert für den Monat, der sich durch die Indizierung ergibt, unter Einbeziehung des Maximalwertes und zweier Konstanten, die Y-Position für den Mittelpunkt des Kreuzes ermittelt (MY).

Die X-Position für das Kreuz ergibt sich durch Hinzuaddieren von 32 - bei jedem Schleifendurchlauf - zu der Variablen F, die vor der Schleife auf den Defaultwert 131 gesetzt wurde. Die beiden rechnerischen Koordinaten müssen, da kein Punkt, sondern das umdefinierte Zeichen 255 ausgegeben werden soll, so umgewandelt werden, daß sich die Koordinaten der linken oberen Ecke der Zeichenmatrix ergeben.

Für die Y-Position geschieht das in Zeile 3050. Die X-Position wird beim Versetzen des Grafikcursors (Zeile 3060) angepaßt, denn sie darf nicht zurück-

geschrieben werden. An der so erreichten Position des Grafikcursors wird das unter Zeichennummer 255 definierte Kreuz auf den Bildschirm gesetzt. Am Schnittpunkt der beiden Linien des Kreuzes liegt der Wert für den jeweiligen Monat.

Nachdem das Diagramm erstellt ist, erfolgt der Rücksprung aus diesem Unterprogramm (Zeile 3100).

3110-3280 **Kurve zeichnen**
Der Aufbau des Unterprogramms KURVE ZEICHNEN entspricht in groben Zügen dem Aufbau von PUNKTE SETZEN. Einer der Hauptunterschiede ist es, daß die ermittelten Punkte durch einen Linienzug miteinander verbunden werden. Um dies zu erreichen, werden die Koordinaten des vorherigen Punktes als Startkoordinaten für die Verbindung mit dem aktuellen Punkt gewählt. Bei ersten Durchlauf der FOR-TO-NEXT-Schleife kann es noch keine Startkoordinaten geben, so daß diese künstlich erzeugt werden müssen (Zeile 3190).

3290-3450 **Balken zeichnen**
Auch BALKEN ZEICHNEN entspricht dem Aufbau von PUNKTE SETZEN, nur daß statt der Kreuze Balken von der X-Achse bis zu den ermittelten Werten der Monate gezogen werden. Zu diesem Zweck wird eine zweite FOR-TO-NEXT-Schleife in die Routine integriert, der die Aufgabe zukommt die zwölf Balken zu zeichnen.

Als Startwert für die zweite Schleife wird die X-Position, an der der gerade bearbeitete Monat steht, um sieben vermindert und als Endwert um ebenfalls sieben erhöht. Der Grafikcursor wird bei jedem Schleifendurchlauf an die Position bewegt, die sich aus dem Schleifenindex (L) und dem errechneten Y-Wert für den Monat zusammensetzt.

Von dieser Position wird dann eine Linie auf die X-Achse gefällt. Hat die innere FOR-TO-NEXT-Schleife ihre Arbeit verrichtet, so ist der Balken für den Monat, der sich aus dem Index der äußeren Schleife (I) ergibt, gezeichnet.

3460-Ende **Programmkopf ausgeben**
In diesem Unterprogramm wird der Name und die Eignerkennung des Programms auf dem Bildschirm ausgegeben und umrandet. Danach Rücksprung ins Hauptprogramm.

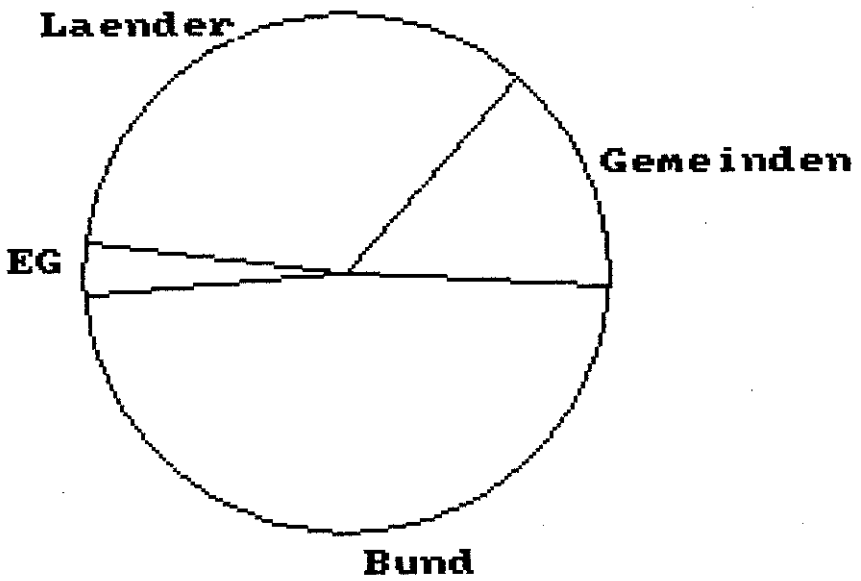
5.3 Kuchendiagramm

Eine Form der grafischen Darstellung von Zahlenmaterial, die von den bisher besprochenen vollkommen abweicht, ist das Kuchendiagramm. Das Kuchendiagramm ermöglicht es, Zahlen, die zusammen ein Ganzes ergeben, in ihrem prozentualen Verhältnis darzustellen. Dabei werden die Prozentzahlen als Segmente eines 360-Grad-Vollkreises veranschaulicht, wodurch auch der Name "Kuchendiagramm" entstanden ist.

Wie man mit einem Kuchendiagramm Zahlenmaterial[1] auswertet, soll ein Beispiel aus dem Staatshaushalt der BRD verdeutlichen:

Im Jahre 1984 nahmen der Bund, die Länder, die Gemeinden und die EG Steuern im Wert von rund 415 Milliarden DM ein. Dabei entfielen 48% (199,2 Mrd.) auf den Bund, 34,7% (144,005 Mrd.) auf die Länder, 13% (53,95 Mrd.) auf die Gemeinden und 3,5% (14,525 Mrd.) auf die EG. Für die Darstellung als Kuchendiagramm werden nun die 415 Mrd. DM Gesamtsteueraufkommen gleich 100% gesetzt, was in der Darstellung dem 360-Grad-Vollkreis entspricht. Entsprechend der prozentualen Verteilung wird nun jedem am Steueraufkommen Beteiligten ein unterschiedlich großes Stück des "Steuerkuchens" zugedacht, so daß sich diese Darstellung ergibt:

1 Quelle: Kaufmännische Betriebslehre (H), Europa Lehrmittel, 1986, Seite 493



Hardcopy 11: **Kuchendiagramm zeigt prozentuale Verhältnisse**

Das Programm "Kuchendiagramm", das diese Grafik erzeugte, erwartet vom Benutzer nach der Eingabe eine Überschrift für das Diagramm, die Eingabe der darzustellenden Werte sowie ihrer Bezeichnungen. Es spielt für die Verarbeitung des Programms keine Rolle, ob Prozentzahlen oder absolute Größen eingegeben werden, da das Programm ohnehin nur ihren Anteil am Vollkreis ermittelt. Will man die Eingabe von Werten beenden, so kann dies durch einfaches Drücken der RETURN-Taste bei der Frage nach Wert und Bezeichnung erreicht werden.

Sind alle Werte ordnungsgemäß eingegeben und mindestens zwei - für eine sinnvolle Darstellung nötige - vorhanden, so beginnt das Programm mit der Erstellung der Kuchengrafik. Zunächst werden die Überschrift und der Vollkreis ausgegeben. Dann wird nach und nach der Vollkreis in einzelne Segmente unterteilt

und diese werden beschriftet. Die fertig erstellte Grafik bleibt solange auf dem Bildschirm erhalten, bis das Programm mit der ESC-Taste unterbrochen wird.

```
100 MODE 2
110 DEG
120 DIM WERT(25)
130 DIM GRAD(25)
140 DIM P(25)
150 DIM BEZ$(25)
160 WINDOW#1,1,80,10,25
170 GES=0
180 INDEX=1
190 '
200 'PROGRAMMKOPF AUSGEBEN
210 '
220 LOCATE 32,3
230 PRINT "KUCHENDIAGRAMM
240 LOCATE 34,4
250 PRINT "JST 2.8.1986"
260 '
270 'RAHMEN ZEICHNEN
280 '
290 MOVE 16,383
300 DRAW 16,320
310 DRAW 623,320
320 DRAW 623,383
330 DRAW 16,383
350 '
360 'BESCHRIFTUNG EINLESEN
370 '
380 LOCATE#1,36,1
390 PRINT#1,"|--------------------------------|"
400 LOCATE#1,1,1
410 INPUT#1,"UEBERSCHRIFT (MAX. 38 BUCHSTABEN)";UE$
420 T1=LEN(UE$)
430 PU=20-(T1/2)
440 '
450 CLS#1
```

```
460 '
470 LOCATE 1,8
480 PRINT "GEBEN SIE JETZT DIE WERTE UND BEZEICHNUNGEN EIN!"
490 '
500 PRINT#1," ";USING"##";INDEX;
510 PRINT#1,"TER WERT:";
520 INPUT#1,WERT(INDEX)
530 '
540 PRINT#1,"BEZEICHNUNG:";
550 INPUT#1,BEZ$(INDEX)
560 '
570 IF WERT(INDEX)=0 AND BEZ$(INDEX)="" THEN CLS:GOTO 650
580 '
590 GES=GES+WERT(INDEX)
600 INDEX=INDEX+1
610 PRINT#1
620 '
630 GOTO 500
640 '
650 IF INDEX<=2 THEN END
660 '
670 MODE 1
680 '
690 'BESCHRIFTUNG AUSGEBEN
700 '
710 LOCATE PU,1
720 PRINT UE$
730 '
740 'GRAFIK ZEICHNEN
750 '
760 GOSUB 1110
770 '
780 ORIGIN 320,200
790 TAG
800 '
810 FOR I=1 TO INDEX
820 GRAD(I)=INT(WERT(I)*360/GES)
830 P(I)=INT(GRAD(I)/2)
840 NEXT I
850 '
```



```
860 I=1:ALT=0
870 FOR GRAD=1 TO 360
880 MOVE 0,0
890 IF GRAD=ALT+P(I) THEN ALT=ALT+P(I)*2:GOSUB 930
900 IF GRAD(I)=GRAD THEN DRAW 150*COS(GRAD),150*SIN(GRAD):GRAD(I+1
)=GRAD(I+1)+GRAD(I):I=I+1
910 NEXT GRAD
920 GOTO 920
930 '
940 'BESCHRIFTUNG DER KREISSEGMENTE
950 '
960 O=(LEN(BEZ$(I))*16)+10
970 X=150*COS(GRAD)
980 Y=150*SIN(GRAD)
990 MOVE X,Y
1000 '
1010 IF X>=0 THEN MOVER 10,0
1020 IF X<0  THEN MOVER -O,0
1030 IF Y>=0 THEN MOVER 0,10
1040 IF Y<0  THEN MOVER 0,-10
1050 '
1060 PRINT BEZ$(I);
1070 '
1080 MOVE 0,0
1090 '
1100 RETURN
1110 '
1120 'KREIS ZEICHNEN
1130 '
1140 'MITTELPUNKT UND RADIUS
1150 '
1160 XP%=320
1170 YP%=200
1180 XR%=150
1190 YR%=150
1200 '
1210 'ERSTE KOORDINATE
1220 '
1230 J%=0
1240 GOSUB 1490
```

```
1250 X1%=X2%
1260 Y1%=Y2%
1270 '
1280 'WEITERE KOORDINATEN
1290 '
1300 FOR J%=0 TO 15 STEP 6
1310 GOSUB 1490
1320 '
1330 'KREIS ZEICHNEN
1340 '
1350 MOVE XP%+X1%,YP%+Y1%
1360 DRAW XP%+X2%,YP%+Y2%
1370 MOVE XP%+X1%,YP%-Y1%+1
1380 DRAW XP%+X2%,YP%-Y2%+1
1390 MOVE XP%-X1%+1,YP%-Y1%+1
1400 DRAW XP%-X2%+1,YP%-Y2%+1
1410 MOVE XP%-X1%+1,YP%+Y1%
1420 DRAW XP%-X2%+1,YP%+Y2%
1430 '
1440 X1%=X2%:Y1%=Y2%
1450 '
1460 NEXT
1470 '
1480 RETURN
1490 '
1500 'KOORDINATEN ERMITTELN
1510 '
1520 X2%=INT(COS(J%)*XR%+0.5)
1530 Y2%=INT(SIN(J%)*YR%+0.5)
1540 '
1550 RETURN
```

Programmbeschreibung:

100-180 Definitionen, Dimensionierungen und Vorbelegungen. MODE 2 wird gesetzt und der Computer mit DEG ins Winkelmaß gesetzt. Die Variablen WERT, GRAD, P und BEZ$ werden auf je 26 Elemente dimensioniert, wodurch die Anzahl

der maximal in einer Kuchengrafik darstellbaren Werte begrenzt ist.

Unter #1 wird ein Fenster definiert, in dessen Grenzen die darzustellenden Werte und deren Bezeichnungen eingelesen werden können, ohne daß die Überschrift zerstört wird.

Die Variablen GES und INDEX werden auf ihren Defaultwert gebracht. In GES wird die Summe aller darzustellenden Werte gebildet, und INDEX dient als Zugriffsschlüssel auf die verschiedensten Feldvariablen.

190-250 **Programmkopf ausgeben**
Der Name des Programms und die Eignerkennung werden auf den Bildschirm gebracht.

260-330 **Rahmen zeichnen**
Die Ausgaben auf dem Bildschirm werden mit einer Linie umrandet.

350-670 **Beschriftung einlesen**
Auf dem Bildschirm wird ein 38 Zeichen großes Feld markiert, in dem der Benutzer den Text für die Überschrift des Kuchendiagramms eingeben soll. Ist der Text eingegeben, so wird in Zeile 420 seine Länge ermittelt und mit dieser Angabe die X-Position errechnet, an der mit der Ausgabe des Textes auf dem Bildschirm begonnen werden muß, damit der Text zentriert erscheint. Alle bisherigen Eingaben wurden im Fenster #1 vorgenommen, das mit Zeile 450 wieder gelöscht wird.

Mit der Aufforderung "GEBEN SIE JETZT DIE WERTE UND BEZEICHNUNGEN EIN!" wird der Benutzer zur Eingabe der darzustellenden Daten veranlaßt. Diese Aufforderung soll während der gesamten Eingabephase auf dem Bildschirm sichtbar bleiben und erfolgt deshalb in Window #0. In

Fenster #1 werden dann die Werte und deren Bezeichnungen eingelesen. Ist die letzte Zeile des Fensters beschrieben, so kann innerhalb von #1 gescrollt werden, ohne die Überschrift zu zerstören.

Zeile 570 wird überprüft, ob der aktuell eingegebene Wert 0 und dessen Bezeichnung Leerstring ist, was zum Verlassen der Eingabeschleife führt.

Wurde die Abbruchbedingung nicht erfüllt, so addiert das Programm den aktuellen Wert zu den schon eingegebenen Werten hinzu, die aufsummiert in der Variablen GES vorliegen. Nach dem Inkrementieren des Index (INDEX) wird die Abarbeitung der Eingabeschleife von vorne begonnen.

In Zeile 650, die nach dem Verlassen der Eingabeschleife angesprungen wird, trifft das Programm auf END, wenn weniger als zwei Werte eingegeben wurden. Mit weniger als zwei Werten läßt sich kein sinnvolles Kuchendiagramm erstellen!

680-720 **Beschriftung ausgeben**
An der vorberechneten Stelle wird die Beschriftung des Kuchendiagramms auf dem zuvor durch MODE 1 (Zeile 670) gelöschten Bildschirm, ausgegeben.

730-920 **Grafik zeichnen**
Erster Schritt bei Erstellen der Kuchengrafik ist der Aufruf der Unterroutine ab Zeile 1110, die einen Kreis auf den Bildschirm zeichnet. In den Mittelpunkt dieses Kreises wird dann der Koordinatenursprung gesetzt und die Textausgabe an den Grafikcursor gebunden.

In einer FOR-TO-NEXT-Schleife, die so viele Durchläufe macht, wie Werte eingegeben wurden,

wird ermittelt, wieviel Grad des 360-Grad-Vollkreises auf die Darstellung der einzelnen Werte entfallen. Des weiteren wird die Hälfte dieser Gradangaben gebildet, um eine Position für die Bezeichnung der Kreissegmente gewinnen zu können.

Die FOR-TO-NEXT-Schleife von Zeile 870 bis Zeile 910 nimmt die Unterteilung des Vollkreises und die Beschriftung der einzelnen Segmente vor. Die Schleife hat genau 360 Durchläufe und kann so für jedes einzelne Grad des Vollkreises überprüfen, ob eine Beschriftung vorzunehmen oder ein Kreissegment abzugrenzen ist. Als Hilfsvariablen dienen innerhalb der Schleife I und ALT, die vor dem Eintritt in die Schleife auf ihren jeweiligen Defaultwert gebracht wurden. I stellt einen Index dar, mit dem auf die Elemente der indizierten Variablen P und GRAD zugegriffen werden kann. In der Variablen ALT werden steht immer die Gradangabe der äußeren Begrenzung des Kreissegments, das zuletzt beschriftet wurde.

Ist die Kuchengrafik auf dem Bildschirm entstanden, so bringt Zeile 920 den Computer in eine Endlosschleife, damit die Ready-Meldung die Grafik nicht zerstören kann.

930-1100 **Beschriftung der Kreissegmente**
Dieses Unterprogramm wird bei Zeichnen der Kuchengrafik immer dann aufgerufen, wenn die Hälfte eines Kreissegmentes erreicht ist. Das Unterprogramm errechnet die Position, an der die Bezeichnung des Segments ausgegeben werden soll, bewegt den Grafikcursor und gibt den Text, den es entsprechend der Indizierung von BEZ$ findet, aus. Ist diese Arbeit verrichtet, setzt die Subroutine noch den Grafikcursor an den Koordinatenursprung, bevor der Rücksprung erfolgt.

1110-Ende **Kreis zeichnen**
Die Integervariablen XP% (X-Koordinate Mittelpunkt), YP% (Y-Koordinate Mittelpunkt), XR% (Radius in X-Ausdehnung) und YR% (Radius in Y-Ausdehnung) werden mit den Angaben zum Zeichnen des Kreises versorgt.

Da beim Zeichnen des Kreises nicht mit dem BASIC-Kommando PLOT gearbeitet wird, sondern das Kommando DRAW Verwendung findet, muß jeweils ein Start- und ein Endpunkt für die Linie errechnet werden. Der Startpunkt für die erste Linie (0 Grad) wird im Programmteil ERSTE KOORDINATE ermittelt, wobei die Koordinaten sich schließlich in den Variablen X1% und Y1% befinden. In der FOR-TO-NEXT-Schleife von Zeile 1300 bis 1460 werden dann in 6-Grad-Schritten alle weiteren Koordinaten (Von 6 bis 90 Grad) erzeugt.

Zu den Koordinaten sei noch angemerkt, daß sich ausnahmslos alle im ersten Viertel des Kreises befinden. Die Koordinaten für die drei übrigen Viertel des Kreises werden durch Spiegelung am Kreismittelpunkt erzeugt. Wie das Zeichnen des Kreises exakt vonstatten geht, wird bei Betrachtung des Programmteils KREIS ZEICHNEN schnell deutlich. Ist eine Linie gezogen, so wird ihr Endpunkt zum Startpunkt für die nächste Linie erklärt (Zeile 1440). Nachdem der Kreis vollständig gezeichnet ist, erfolgt der Rücksprung zur Hauptschleife.

Das Unterprogramm KOORDINATEN ERMITTELN wird ausschließlich von KREIS ZEICHNEN verwendet. In diesem Unterprogramm können mit dem Handwerkszeug der Geometrie die Koordinaten von Punkten, die sich auf der Peripherie des ersten Viertelkreises befinden, errechnet werden.

In J% befindet sich eine Gradangabe, zu der die entsprechenden Koordinaten (X2%, Y2%) ermittelt werden sollen.

6. Grafische Darstellung mathematischer Funktionen

Erinnern Sie sich noch an Ihre Schulzeit? Wenn ja, dann ist Ihnen bestimmt auch noch eine Episode aus dem Mathematikunterricht in der Erinnerung haften geblieben, die man überschreiben könnte mit: "Wer zeichnet den schönsten Funktionsgraphen?" Wenn Sie sich nicht mehr an diese Zeit erinnern, dann lassen Sie sich einen Hauch Schüleralltag um die Nasen wehen.

Ein bei Schülern wie Lehrern gleichsam beliebtes Gebiet der Mathematik ist die grafische Darstellung mathematischer Funktionen - bei den Schülern ist dieses Gebiet so beliebt, weil Malen viel schöner ist als Rechnen, und die Lehrer lieben die gezeichneten Funktionen so, weil sie auf diese Weise endlich die Sauberkeit der Schüler mit einer Note belegen können. Doch klären wir fern ab von allen Emotionen, die mit der Schulzeit verwoben sind, erst einmal die Grundlage für dieses Kapitel, die Funktion.

Das Mathematikbuch definiert den Begriff der Funktion folgendermaßen[1]: "Wenn eine eindeutige Zuordnung zwischen den Elementen zweier Mengen A und B vorliegt, dann spricht man in der Mathematik von einer 'Funktion'. Zur Bezeichnung von Funktionen sind grundsätzlich alle Buchstaben zugelassen; häufig verwendet man jedoch den Buchstaben f, manchmal auch g, h, ...usw. Ist durch eine Funktion f einem Element x aus A das Element y aus B zugeordnet, so nennt man y auch den 'Funktionswert von x' und bezeichnet diesen mit 'f(x)', gelesen: 'f von x'. Die Variable x nennt man auch das 'Argument' der Funktion."

Wem diese Definition kein Jota weitergeholfen hat, dem sei der Sachverhalt noch einmal anschaulicher, mit Hilfe eines Beispiels, vor Augen geführt. Gegeben ist als Definitionsbereich, also der

1 Quelle: Mathematik Sekundarstufe II Analysis Grundkurse Neu, Schwann, 1982, Seite 11

Menge A aus der Mathematikbuchdefinition, die Zahlenmenge {1;2;3;4;5}. Dieser Zahlenmenge werden nun eindeutig, durch die Funktionsgleichung f(x)=2*x Elemente des Wertebereiches B zugeordnet, wobei B als die Menge der natürlichen Zahlen definiert ist. Zur Erinnerung: Die Menge der natürlichen Zahlen ist die Menge, die sich aus den Zahlen 1, 2, 3, ... bis unendlich zusammensetzt.

Welche Zuordnungen sich nach der Funktionsgleichung ergeben, läßt sich recht anschaulich an einer *Wertetabelle* zeigen. Die Wertetabelle kann einfach dadurch gewonnen werden, daß für x nacheinander alle Elemente des Definitionsbereiches in die Funktionsgleichung eingesetzt werden und die Gleichung nach f(x) aufgelöst wird. Sowohl x als auch f(x) werden dann im Zusammenhang aufgelistet, das Ergebnis ist die Wertetabelle.

x	1	2	3	4	5
f(x)	2	4	6	8	10

Um nun wieder den Bogen zur einleitenden Frage, "Wer zeichnet den schönsten Funktionsgraphen?", zu schlagen, sei ausgeführt, daß sich der gerade aufgezeigte mathematische Zusammenhang auch grafisch darstellen läßt. Um nicht zu tief in die Materie einzudringen, bleiben wir bei unserem Beispiel f(x)=2*x.

Will man diesen Sachverhalt zeichnerisch erfassen, bedarf es im Grunde nur zweier *Geraden*, die aufeinander senkrecht stehen. Die beiden Geraden stellen Zahlenstrahle dar, von denen der waagerechte die Elemente der Definitionsmenge beherbergt, der senkrechte die des Wertebereiches.

Die beiden Zahlenstrahle werden von ihrem Schnittpunkt aus in gleichen Abständen unterteilt. Diese Unterteilung beschriften wir nun wie folgt: Der waagerechte Strahl, auch X-Achse genannt, vom Schnittpunkt, den wir mit Null bezeichnen, aus nach rechts mit 1, 2, 3, usw. Ebenso verfahren wir mit der zweiten Linie, nur daß hier von Schnittpunkt aus nach oben gezählt

wird. Die nun entstandene Zeichnung nennt man in der Mathematik "Das karthesische Koordinatensystem".

In dieses Koordinatensystem wollen wir nun den Graphen unserer Funktion, also ihr gezeichnetes Abbild, einbringen. Nehmen wir das erste Element des Definitionsbereichs, die 1. Von der X-Achse, also dem Zahlenstrahl, der die Elemente des Wertebereiches beherbergt, aus denken wir uns an der mit 1 bezeichneten Stelle eine zur X-Achse senkrecht stehende Geraden. Aus der Wertetabelle wissen wir, daß der 1 aus der Definitionsmenge das Element 2 des Wertebereiches zugeordnet wird. Also auf der senkrechten Achse die mit 2 markierte Stelle gesucht und eine waagerechte Linie gedacht. Am Schnittpunkt der beiden imaginären Linien liegt ein Punkt, den wir markieren wollen.

Sind wir in gleicher Weise mit allen fünf Elementen des Definitionsbereiches verfahren, so ergibt sich das Bild, das auf Abbildung 8 zu sehen ist.

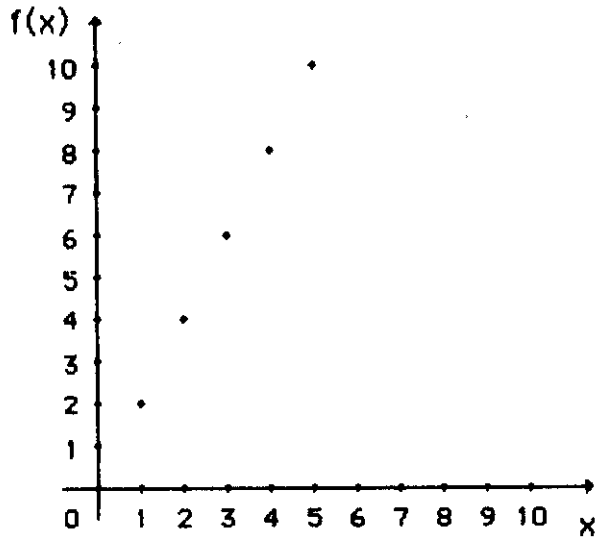


Abb. 8: **Funktionsgraph der Funktion f(x)=2*x (x aus {1;2;3;4;5})**

Die fünf Punkte stellen die Zuordnung der Elemente des Definitionsbereiches der Elemente des Wertebereiches in grafischer Form dar und werden deshalb als Funktionsgraph bezeichnet.

6.1 Das Koordinatensystem

Aus den vorangegangenen Betrachtungen über Funktionen und deren Darstellung wissen wir, daß ein Koordinatensystem Grundlage für die grafische Darstellung von mathematischen Funktionen ist. Deshalb soll der erste Schritt zum Zeichnen von Funktionsgraphen mit dem Computer - was das Ziel dieses Kapitels ist - das Erstellen eines geeigneten Koordinatensystems sein.

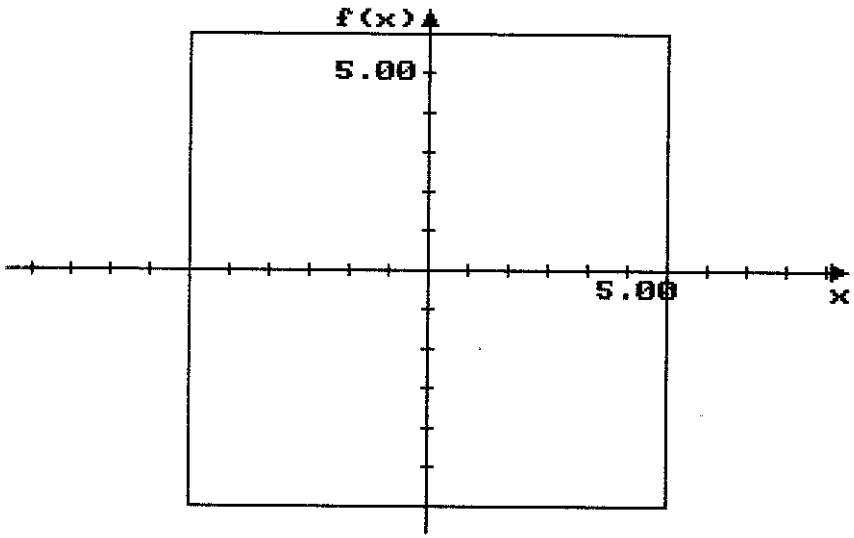
Welche Anforderungen an ein Koordinatensystem zu stellen sind, in das hinein sich möglichst alle Funktionsgraphen zeichnen lassen, erfährt man am einfachsten, wenn man sich die Vorgehensweise eines Menschen veranschaulicht, der ein solches System von Hand zeichnet.

Ohne jede vorhergehende Überlegung lassen sich, für jedes Koordinatensystem gleich die beiden Achsen zeichnen. An das rechte Ende der X-Achse und an das obere Ende der f(x)-Achse (auch *Y-Achse* genannt) werden Pfeilspitzen gesetzt. Diese sollen andeuten, daß die gezeichneten Linien nur einen Ausschnitt den Zahlenstrahlen darstellen, sie sich aber bis unendlich fortsetzen. Beide Achsen können auch schon beschriftet werden, nämlich mit "x" bzw. "f(x)". Der nächste Schritt ist die Skalierung der Achsen, die beliebig, aber für beide gleich, vorgenommen werden kann. Soweit kann das Koordinatensystem für alle Funktionen als gleich angesehen werden.

Damit die Skalierung beide Achsen nicht nur in kleinere Einheiten unterteilt, sondern die Einteilung durch eine Größenordung an Aussagekraft gewinnt, ist es notwendig, die Skalierung zu beschriften. Dazu werden einfach an einige der Einteilungen die Elemente des Definitionsbereiches (X-Achse) oder des Wertebereiches (Y-Achse) geschrieben, die an dieser Stelle des durch die Achse sysmbolisierten Zahlenstrahls liegen. Diese

Beschriftung kann erst dann sinnvoll vorgenommen werden, wenn bekannt ist, für welchen Definitionsbereich die Funktion gezeichnet werden soll.

Generell beschriftet man die Einteilung so, daß der gewünschte Definitionsbereich möglich groß im Koordinatensystem zu sehen ist. Will man eine Funktion im Definitionsbereich von -6 bis 6 betrachten, so würde man, bei zehn Einteilungen der Achsen vom Nullpunkt aus, die fünfte mit der Zahl "5" beschriften. Die angestellten Überlegungen wurde in der Hardcopy 12 bereits verwirklicht.



Hardcopy 12: Koordinatensystem zur Darstellung einer Funktion im Definitionsbereich -6 bis 6

Das Programm mit dem die voranstehende Bildschirmausgabe erzeugt wurde, zeichnet die Achsen, skaliert und teilt sie ein. Es benötig von Benutzer lediglich zwei Angaben, bevor es seine Arbeit aufnehmen kann. Die erste Angabe ist die des Definitionsbereiches, die wir schon von unseren theoretischen Überle-

gungen her kennen. Die zweite Angabe muß ein *Vergrößerungsfaktor* sein. Mit Hilfe des Vergrößerungsfaktors kann der zu betrachtende Definitionsbereich optimal in das Bildschirmfenster eingepaßt werden.

Zur Erzeugung von Hardcopy 12 wurde die Zahl 1 als Vergrößerungsfaktor angegeben, was einer Ausgabe im Maßstab 1 zu 1 entspricht. Diese Maßstabsangabe ist geeignet, weil das Programm die X-Achse generell vom Nullpunkt aus in zehn Einheiten unterteilt und somit der Definitionsbereich -6 bis 6 dargestellt werden kann. Möchte man aber einen Definitionsbereich von -1 bis 1 betrachten, so würde diese Bereich nur einen kleinen Teil auf dem Bildschirm in Anspruch nehmen. Die Eingabe des Vergrößerungsfaktors 10 vergrößert die Darstellung des gewünschten Definitionsbereichs auf das 10fache, so daß der Bildschirm nahezu ausgefüllt wird.

Selbstverständlich kann auch eine Zahl kleiner als 1 Vergrößerungsfaktor werden, nur mit dem Unterschied, daß die Darstellung auf dem Bildschirm dann schrumpft. So kann aber ein Definitionsbereich von beispielsweise -100 bis 100 immer noch auf dem Bildschirm sichtbar gemacht werden.

Bei der Wahl des entsprechenden Vergrößerungsfaktors soll der Rahmen, der in das Koordinatensystem eingezeichnet wird behilflich sein (vgl. Hardcopy 12). Dieser Rahmen begrenzt den gewählten Definitionsbereich auf der X-Achse und teilt noch einmal ein entsprechendes Stück auf der Y-Achse ab. Ist der Rahmen nur als kleines Quadrat in der Mitte des Bildschirms sichtbar, so sollte eine größere Zahl als Vergrößerungsfaktor gewählt werden. Ist der Rahmen nicht sichtbar, so muß der Faktor verkleinert werden.

```
100 MODE 1
102 SYMBOL 255,8,8,28,28,62,62,127,8
104 '
106 'WELCHER BEREICH SOLL BETRACHTET WERDEN
108 '
110 INPUT "WELCHEN DEF.-BEREICH WUENSCHEN SIE";DEFI
```



```
112 INPUT "WELCHE VERGROESSERUNG WUENSCHEN SIE";FF
114 FF=FF*30
116 '
118 CLS
120 '
122 'KOORDINATENSYSTEM ZEICHNEN
124 '
126 TAG
128 ORIGIN 320,200
130 '
132 'X-ACHSE
134 '
136 PLOT -320,0,1
138 DRAW 320,0
140 MOVER -16,6
142 PRINT CHR$(246);
144 MOVER -16,-16
146 PRINT "x";
148 '
150 'SKALIERUNG DER X-ACHSE
152 '
154 FOR I=-300 TO 300 STEP 30
156 MOVE I,4
158 DRAW I,-4
160 NEXT I
162 '
164 'EINTEILUNG DER X-ACHSE
166 '
168 MOVE 78,-8
170 PRINT USING "####.##";150/FF;
172 '
174 'Y-ACHSE
176 '
178 MOVE 0,-200
180 DRAW 0,200
182 MOVER -8,-2
184 PRINT CHR$(255);
186 MOVER -80,0
188 PRINT "f(x)";
190 '
```

```
192 'SKALIERUNG DER Y-ACHSE
194 '
196 FOR I=-180 TO 180 STEP 30
198 MOVE 4,I
200 DRAW -4,I
202 NEXT I
204 '
206 'EINTEILUNG DER Y-ACHSE
208 '
210 MOVE -120,158
212 PRINT USING "####.##";150/FF;
214 '
216 TAGOFF
218 '
220 'AUSSCHNITT KENNZEICHNEN
222 '
224 MOVE -DEFI*FF,-DEFI*FF
226 DRAWR 2*DEFI*FF,0,2
228 DRAWR 0,2*DEFI*FF
230 DRAWR -2*DEFI*FF,0
232 DRAWR 0,-2*DEFI*FF
```

Programmbeschreibung:

100-102 Nach dem Setzen von MODE 1 wird, da sich kein geeignetes Zeichen im Charaktergenerator befindet, unter der Zeichennummer 255 ein Pfeil nach oben definiert, der beim Zeichnen des Koordinatensystems Verwendung findet.

104-118 **Welcher Bereich soll betrachtet werden**
Dieser Programmteil erfragt vom Benutzer, welchen Definitionsbereich er gerne betrachten möchte, und mit welchem Vergrößerungsfaktor das Programm arbeiten soll. Der Vergrößerungsfaktor wird mit 30 multipliziert, um den Faktor an die Einteilung des Koordinatensystems anzupassen.

Sind beide Werte eingegeben, wird der Bildschirm zum Zeichnen des Koordinatensystems gelöscht (Zeile 118).

120-128 **Koordinatensystem zeichnen**
Als Vorbereitung für das Zeichnen des Koordinatensystems wird die Anpassung der Textausgabe an den Grafikcursor (TAG) und der Koordinatenursprung in die Mitte des Bildschirms gesetzt.

130-146 **X-Achse**
Der Programmteil X-ACHSE zeichnet mit dem Farbstift 1 in die Mitte des Bildschirms eine waagerechten Linie. An das rechte Ende der Linie wird eine Pfeilspitze und die Bezeichnung "x" gesetzt. Diese Linie stellt die X-Achse des rechtwinkligen Koordinatensystems dar.

148-160 **Skalierung der X-Achse**
Die gezeichnete X-Achse wird in Abstand von 30 Bildpunkten durch kleine senkrechte Striche unterteilt.

162-170 **Einteilung der X-Achse**
Unter Einbeziehung des Vergrößerungsfaktors FF wird an den fünften Skalierungsstrich, auf dem positiven Teil der X-Achse, der Wert geschrieben, der ihm auf dem Koordinatensystem entspricht.

172-188 **Y-Achse**
Der Programmteil Y-ACHSE zeichnet in die Mitte des Bildschirms eine senkrechte Linie. An das obere Ende der Linie wird die Pfeilspitze, die unter Zeichennummer 255 definiert wurde, und die Bezeichnung "f(x)" gesetzt. Diese senkrechte Linie stellt die Y-Achse des Koordinatensystems dar.

190-202 **Skalierung der Y-Achse**
Die gezeichnete Y-Achse wird in Abstand von 30 Bildpunkten durch kleine waagerechte Striche unterteilt.

204-216 **Einteilung der Y-Achse**
Unter Einbeziehung des Vergrößerungsfaktors FF wird an den fünften Skalierungsstrich, auf dem positiven Teil der Y-Achse, der Wert geschrieben, der ihm auf dem Koordinatensystem entspricht.

Nachdem das Koordinatensystem vollständig gezeichnet, skaliert und beschriftet ist, wird die Textausgabe wieder an die Position des Textcursors geleitet (TAGOFF).

218-232 **Ausschnitt kennzeichnen**
Mit Farbstift 2 wird der Ausschnitt des Koodinatensystems durch einen Rahmen markiert, der vom Benutzer für die Ausgabe bestimmt wurde. Die Eckpunkte des Rahmens ergeben sich durch Spiegeln des für DEFI eingegebenen Wertes, mulitipliziert mit dem Vergrößerungsfaktor FF, in alle vier Quadranten des Koordinatensystems.

6.2 Lineare Funktionen

Sie erinnern sich doch noch an Abbildung 6, den Funktionsgraph der Funktion f(x)=2*x (x aus {1,2,3,4,5}). An diesem Funktionsgraphen haben wir Ihnen die Darstellung mathematischer Funktionen im karthesischen Koordinatensystem nahegebracht. Da es unser Ziel ist, Funktionsgraphen mit dem Schneider-CPC zu zeichnen, wollen wir mit dieser Funktion, deren Abbild Sie ja schon kennen, einen ersten Versuch unternehmen.

Grundlage für das Zeichnen des Funktionsgraphen ist das Programm zum Zeichnen eines Koordinatensystems aus Kapitel 6.1. An dieses Programm sollen einige Zeilen angefügt werden, die

den Graphen der Funktion f(x)=2*x auf dem Bildschirm erscheinen lassen.

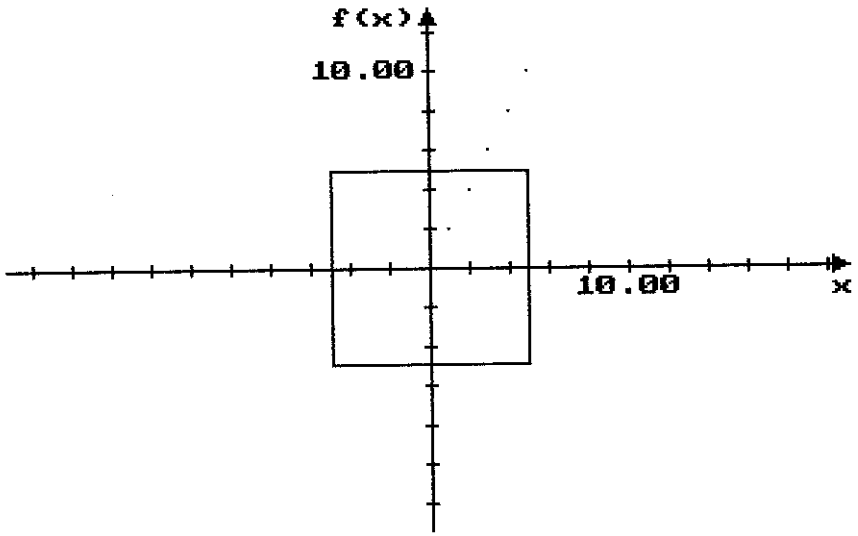
Nachdem das Koordinatensystem sich bereits auf dem Bildschirm befindet, obliegt den zu ergänzenden Programmzeilen nur noch das Zeichnen der Funktion. Da die Funktion als Definitionsbereich nur fünf Elemente, nämlich 1, 2, 3, 4, 5, besitzt, lassen sich alle Elemete des Definitionsbereichs einfach mit einer FOR-TO-NEXT-Schleife erzeugen. Diese Schleife hat von 1 bis 5 genau fünf Durchläufe, und der Schleifenindex nimmt nacheinander die Werte aller Elemente des Definitionsbereichs an.

Zu den Werten des Definitionsbereiches können die Werte des Wertebereiches über die Zuordnungsvorschrift der Funktionsgleichung gefunden werden.

Um schließlich den Graphen der Funktion im gleichen Maßstab wie das Koordinatensystem entstehen zu lassen, müssen die X-Werte des Definitionsbereiches und die Y-Werten des Wertebereiches mit dem Vergrößerungsfaktor FF mulitipliziert werden.

```
236 'FUNKTION f(x)=2*x (x AUS {1,2,3,4,5}) ZEICHNEN
238 '
240 FOR X=1 TO 5
242 Y=2*X
244 PLOT X*FF,Y*FF
246 NEXT X
```

Wurden die voranstehenden Programmzeilen zum Programm aus Kapitel 6.1 hinzugefügt, so erzeugt das Gesamtprogramm die Bildschirmausgabe, die durch Hardcopy 13 wiedergegeben wird. Zur Angabe des Definitionsbereichs wurde die Zahl 5 und als Vergrößerungsfaktor 0,5 eingegeben, um den Funktionsgraphen vollständig sichtbar auf dem Bildschirm zu haben.



Hardcopy 13: Graph der Funktion f(x)=2*x (x aus {1;2;3;4;5})
Definitionsbereich 5 Vergrößerungsfaktor 0,5

Der vorgestellte Programmteil liefert für den Spezialfall, daß die Funktion für eine begrenzte Anzahl von Argumenten definiert ist, den Funktionsgraphen, versagt aber gänzlich, wenn die Funktion für den Definitionsbereich aller rationalen Zahlen gezeichnet werden soll. Die rationalen Zahlen haben nämlich die Eigenschaft, daß zwischen zwei dieser Zahlen immer eine dritte existiert und es so unmöglich ist, alle Elemente des Definitionsbereiches, wie im oben stehenden Beispiel, mit einer Schleife zu generieren.

Wir suchen also nach einem Verfahren, wie der Graph der Funktion f(x)=2*x, definiert für alle x aus der Menge der rationalen Zahlen, gezeichnet werden kann. Dabei können wir uns eine Eigenschaft dieser Funktion zunutze machen - die Funktion f(x)=2*x fällt unter die Gruppe der linearen Funktionen.

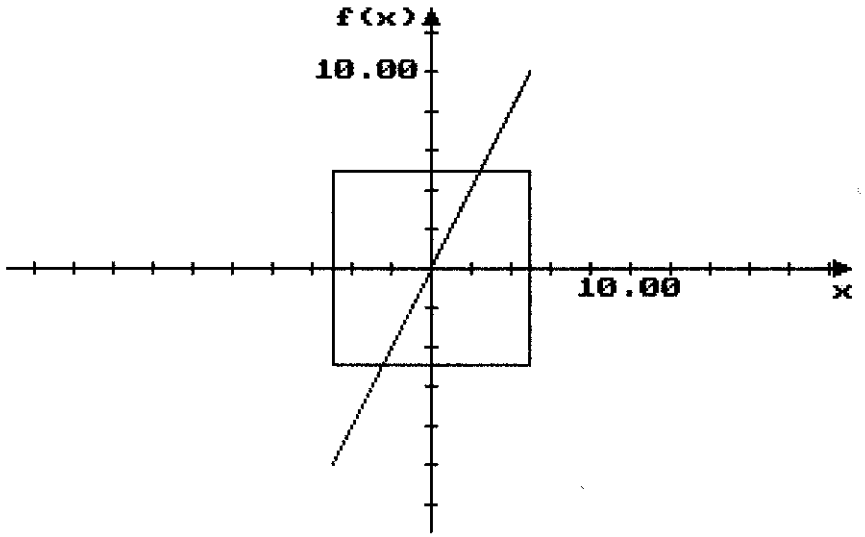
Lineare Funktionen haben immer eine Gerade als Funktionsgraphen. Eine Gerade hat die Eigenschaft, wie aus der Geometrie bekannt sein dürte, daß sie durch zwei Punkte, die auf ihr liegen, eindeutig bestimmt wird. Das heißt, eine Gerade läßt sich immer zeichnerisch darstellen, wenn man zwei Punkte, die auf ihr liegen, miteinander verbindet und sich die entstehende Linie in die Unendlichkeit verlängert vorstellt.

Diese Eigenschaft der linearen Funktionen kann man sich bei erzeugen ihrer Graphen mit dem Computer zu Nutze machen. Man ermittelt einfach die Funktionswerte am unteren und oberen Ende des Definitionsbereichs und hat so die Koordinaten zweier Punkte. Die Punkte zu verbinden ist für den Computer ein leichtes.

Die Ergänzung, die sich zum Programm in Kapitel 6.1 ergibt, und mit der der Graph der Funktion f(x)=2*x für einen beliebigen Definitionsbereich gezeichnet werden kann, ist im folgenden aufgelistet.

```
236 'FUNKTION f(x)=2*x ZEICHNEN
238 '
240 X=-DEFI
242 Y=2*X
244 MOVE X*FF,Y*FF
246 '
248 X=DEFI
250 Y=2*X
252 DRAW X*FF,Y*FF
```

Die Hardcopy 14 zeigt, daß sich mit dem Verfahren, aus zwei Punkte auf dem Funktionsgraphen auf den gesamten Graphen zu schließen, bewährt hat. Durch Austauschen der Funktionsgleichung in den Zeilen 242 und 250 kann selbstverständlich auch jede andere Funktion dargestellt werden - vorausgesetzt, es handelt sich um eine lineare Funktion.



Hardcopy 14: Graph der Funktion f(x)=2*x
Definitionsbereich 5 Vergrößerungsfaktor 0,5

Eine Art von Funktionen, bei denen das gerade eingeführte Verfahren nicht mehr zum Funktionsgraphen führt, soll im folgenden Kapitel vorgestellt werden.

6.3 Quadratische Funktionen

Grundsätzlich jede Funktion, die nicht unter die Definition einer linearen Funktion fällt, läßt das am Ende von Kapitel 6.2 vorgestellte Verfahren zum Zeichnen von Funktionsgraphen versagen. Dies liegt ganz offensichtlich an der Tatsache, daß der Graph dieser Funktionen eben keine Gerade ist und so auch nicht eindeutig durch zwei Punkte bestimmt werden kann. Ein Beispiel für eine Funktion dieser Art ist die quadratische Funktion.

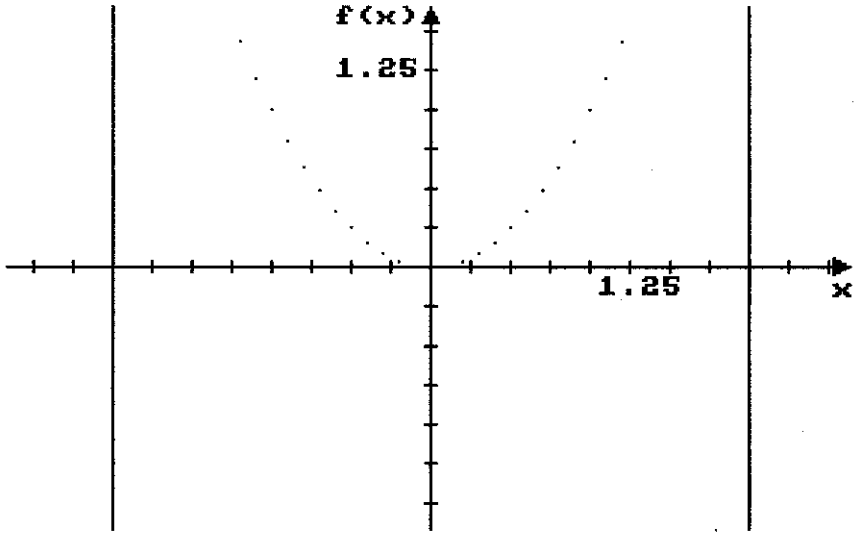
Quadratische Funktionen, die ihren Namen vom Potenzieren des Arguments mit zwei beziehen, haben immer das grafische Abbild einer *Parabel*. Eine Parabel kann man sich als eine Art Hufeisen vorstellen, dessen Schenkel ein wenig nach außen gebogen sind und sich in die Unendlichkeit fortsetzen. Schon von der Anschauung her ist dieser Funktionsgraph vollkommen anders als der einer linearen Funktion.

Um der Graphen einer quadratischen Funktion, die über die Menge der rationalen Zahlen definiert ist, zeichnen zu können, muß man einen Kompromiß eingehen. Einen Kompromiß deshalb, weil es vollkommen unmöglich ist, alle Elemente des Definitionsbereiches zum Zeichnen des Funktionsgraphen heranzuziehen, sieht man, wie die Menge der rationalen Zahlen definiert ist. Der Kompromiß sieht konkret so aus, daß man in gleichmäßigen Abständen Elemente des Definitionsbereiches herausgreift, zu ihnen die Funktionswerte ermittelt und den Graphen zeichnet. Das Ergebnis dieser Verfahrensweise erinnert sehr stark an die Darstellung der Funktion f(x)=2*x für x aus {1;2;3;4;5}, da das Prinzip, den Funktionsgraphen zu erstellen, identisch ist. Nur mit dem Unterschied, daß im Fall der linaren Funktion, definiert für fünf Argumente, der gezeichnete Funktionsgraph den Tatsachen entsprach, bei der quadratischen Funktion aber nur eine Näherung darstellt.

Die unten abgedruckte Ergänzung zum Koordinatensystemprogramm ermittelt im Abstand von 0,1 Einteilungseinheiten einen Funktionswert und zeichnet damit den Graphen der Funktion f(x)=x*x, den man in der Mathematik als Normalparabel bezeichnet.

```
236 'FUNKTION f(x)=x*x ZEICHNEN
238 '
240 FOR X=-DEFI TO DEFI STEP 0.1
242 Y=X*X
244 PLOT X*FF,Y*FF
246 NEXT X
```

Das mit diesem Programm erstellte Abbild der Funktion f(x)=x*x für den Definitionsbereich -2 bis 2 und einer Vergrößerung von 1 zu 4 (Vergrößerungsfaktor=4) ist in Hardcopy 15 wiedergegeben.



Hardcopy 15: Graph der Funktion f(x)=x*x
Definitionsbereich 2 Vergrößerungsfaktor 4

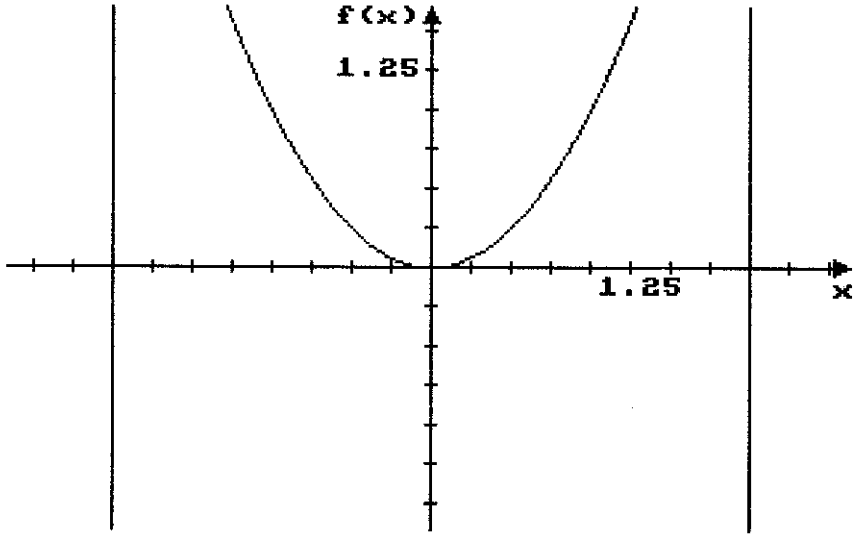
Durch Verringern der Schrittweite in Zeile 240 der Programmergänzung läßt sich der Abstand der einzelnen Punkte, die den Funktionsgraphen ausmachen, praktisch beliebig verringern. Damit kann zum einen ein äußerst genaues Bild der Funktion gewonnen werden, und zum anderen macht sich der Kompromiß, nicht alle Elemente des Definitionsbereiches beim Zeichnen zu berücksichtigen, nicht mehr bemerkbar. Doch auch diese Methode hat einen entscheidenden Nachteil. Je geringer der Punktabstand gewählt wird, desto mehr Funktionswerte müssen berechnet werden und die Verarbeitungsgeschwindigkeit des Programms sinkt schnell auf ein nicht mehr erträgliches Maß ab.

Abhilfe schafft hier die Verknüpfung der Verfahren, einzelne Punkte zu berechnen und zwei Punkte miteinander durch eine Linie zu verbinden. Beide Verfahren isoliert sind schon bekannt, doch die Fusion beider führt zum Zeichnen beliebiger Funktionen mit beliebiger Genaugigkeit. Wobei sich die Genauigkeit aus der Anzahl der berechneten Funktionswerte ergibt und so konträr zur Verarbeitungsgeschwindigkeit steht.

```
236 'FUNKTION f(x)=x*x ZEICHNEN
238 '
240 X=-DEFI
242 Y=X*X
244 PLOT X*FF,Y*FF
246 '
248 FOR X=-DEFI TO DEFI STEP 0.1
250 Y=X*X
252 DRAW X*FF,Y*FF
254 NEXT X
```

Die Funktionsweise des Programms ist denkbar einfach: Am unteren Ende des Definitionsbereichs wird der Funktionswert ermittelt. Er dient als Startpunkt für das Ziehen der ersten Linie. Dann werden innerhalb einer FOR-TO-NEXT-Schleife, im Abstand der eingegebenen Schrittweite, alle weiteren Funktionswerte bis zum oberen Ende des Definitionsbereichs errechnet. Jedesmal wenn ein neuer Funktionswert ermittelt worden ist, wird durch Multiplikation mit dem Vergrößerungsfaktor seine Position auf dem Bildschirm bestimmt und eine Linie dorthin gezogen.

Das Ergebnis, die Normalparabel, sieht bei gleichem Definitionsbereich und gleichem Vergrößerungsfaktor wie bei Hardcopy 15 schon so gut aus, daß keine Verbesserung mehr anzubringen ist.



Hardcopy 16: Graph der Funktion f(x)=x*x
Definitionsbereich 2 Vergrößerungsfaktor 4

6.4 Der Funktionsplotter

Alle Erkenntnisse, die wir auf den vorangegangenen Seiten dieses Kapitels zusammengetragen haben, sollen nun in ein vollständiges Anwendungprogramm einfließen, den "Funktionsplotter". Mit diesem Programm soll es möglich sein, beliebige Funktionen mit beliebiger Genauigkeit auf dem Bildschirm darzustellen, wobei neben dem zu betrachtenden Definitionsbereich und der verwendeten Vergrößerung auch die Abstände, die zwischen den einzelnen berechneten Funktionswerten liegen, frei wählbar ist. Diese zusätzliche Option macht es möglich, sich mit dem Programm einen schnellen Überblick über den ungefähren Verlauf des Funktionsgraphen zu machen, aber auch die Funktion mit maximal auf dem Bildschirm darstellbarer Auflösung zu zeichnen.

Folgende Bemerkung muß an den Anfang der Betrachtung des Funktionsplotters gestellt werden: Ganz zu Anfang des Funktionsplotters steht ein Unterprogramm, das die Funktionsgleichung enthält. Dieser Platz innerhalb des Programms wurde gewählt, damit die Funktionsgleichung - soll sie durch eine andere ersetzt werden - schnell aufgefunden werden kann. Vor dem Starten des Programms muß in Zeile 104 die Funktionsgleichung eingetragen werden, deren Graph mit dem Programm erstellt werden soll. Bei der Funktionsgleichung muß immer die Form gewahrt sein, daß die Variable X des Argument der Funktion und Y der Funktionswert von X ist, da der Funktionsplötter anderenfalls nicht funktionieren kann.

Bedingt durch das Unterprogramm am Anfang des Funktionsplotters tritt beim Versuch, das Programm durch "RUN" zu starten, die Fehlermeldung "Unexpected RETURN in 108" auf. Wie im Listing angemerkt, muß der Funktionsplotter mit "RUN 112" gestartet werden, wodurch die Zeilen vor 112 ihre Bedeutung verlieren.

Nachdem das Programm ordnungsgemäß gestartet wurde und den Programmkopf aufgebaut hat, erfragt es vom Benutzer drei Angaben:

```
1.  Welcher Definitionsbereich
2.  Welcher Vergrößerungsfaktor
3.  Grad der Feinheit
```

Die ersten beiden Angaben werden Ihnen - vorausgesetzt Sie haben das Kapitel bis hierhin aufmerksam verfolgt - sicher keinerlei Schwierigkeiten bereiten. Die dritte Angabe ist nichts anderes als der Abstand zwischen den Punkten, die berechnet werden. Diese Angabe sollten Sie unter Berücksichtigung des gewählten Vergrößerungsfaktors immer so wählen, daß bei der Darstellung der Funktion jeder Bildpunkt berechnet wird, der für den Funktionsgraphen von Bedeutung ist. Eine Auflösung, die über die auf dem Bildschirm darstellbare hinausgeht, ist nicht mehr sinnvoll, da sie das Ergebnis auch nicht verbessert.

Hat der Funktionsplotter vom Benutzer die nötigen Angaben erhalten, so löscht das Programm den Bildschirm und beginnt in MODE 1 mit dem Zeichnen, Skalieren und Beschriften des Koordinatensystems. In das Koordinatensystem wird in bekannter Weise der Rahmen gezeichnet, der den gewählten Auschnitt kennzeichnet.

Ist soweit alles vorbereitet, beginnt der Funktionsplotter mit der Berechnung der Funktionswerte. Dabei wird mit roten Punkten auf der X-Achse angezeigt, an welcher Stelle das Programm gerade die zugehörigen Funktionswerte ermittelt. An dieser Anzeige läßt sich aber nicht nur die Position ablesen, an der gerade zu den Werten des Definitionsbereichs Elemente des Wertebereichs ermittelt werden, sondern sie zeigt auch, ob der Grad der Feinheit richtig gewählt wurde. Bleiben beim Zeichnen der Funktion zwischen den Punkten des Indikators noch Stellen der X-Achse frei, so ist die maximal darstellbare Feinheit noch nicht erreicht. Ebenso ist mit den roten Punkten auf der X-Achse ein Herantasten an das maximale Auflösungsvermögen von der anderen Seite her möglich - so kann, wird die Funktion zu langsam gezeichnet, was ein untrügliches Merkmal für übertrieben hohe Auflösung ist, der Grad der Feinheit gröber gewählt werden. Die roten Punkte zeigen an, ob die Auflösung stimmt.

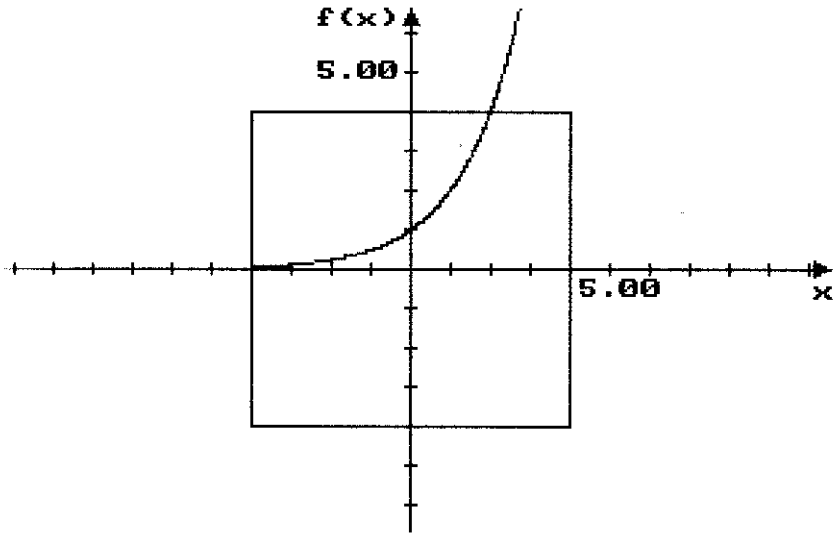
Doch damit der Hinweise nicht genug, die der Indikator auf der X-Achse dem Benutzer gibt. Da das Band immer am unteren Ende des Definitionsbereiches seine Arbeit beginnt und am oberen damit endet, zeigt er durch ein Erreichen der oberen Begrenzung an, daß die Funktion fertig gezeichnet ist. Auch wenn auf dem Bildschirm kein Funktionsgraph zu erblicken ist. Der Benutzer hat dann lediglich den falschen Betrachtungsausschnitt gewählt.

Die Arbeit des Funktionsplotters läßt sich grundsätzlich immer durch das Betätigen einer beliebigen Taste unterbrechen, was das Programm zur Frage nach dem Definitionsbereich zurückkehren läßt. An dieselbe Stelle kehrt das Programm auch zurück, nachdem die Funktion gezeichnet und eine Taste gedrückt wurde. Von sich aus zerstört das Programm die erstellte Grafik nie,

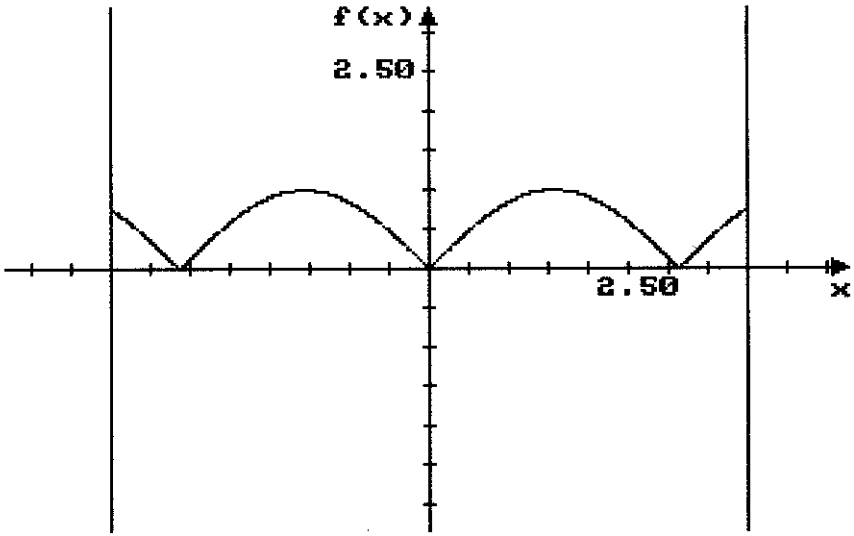
sondern stellt sie dem Anwender zur ungestörten Betrachtung zur Verfügung, bis der seine Betrachtungen durch einen Tastendruck als beendet anzeigt.

Sollten Sie während eine Funktion gezeichnet wird, aus dem Lautsprecher Ihres CPCs einen Ton vernehmen, so deutet das an, daß die Ermittlung eines Funktionswertes zu einem Ergebnis geführt hat, welches für den Computer nicht mehr darstellbar ist. Die Fehlermeldung führt zwar nicht zum Abbruch des Programms, sollte Ihnen aber signalisieren, daß mit dem entstehenden Funktionsgraphen etwas nicht in Ordnung ist.

Bevor wir Sie in das Abtippen des Programmlistings entlassen, wollen wir Ihnen noch zwei Funktionsgraphen präsentieren, zusammen mit den Gleichungen und Angaben, die zu ihrer Entstehung geführt haben. Mit diesen beiden Funktionen können Sie sogleich das ordnungsgemäße Arbeiten Ihres Funktionsplotters überprüfen.



Hardcopy 17: Graph der Funktion f(x)=2^x Definitionsbereich 4
Vergrößerungsfaktor 1 Grad der Feinheit 0,01



Hardcopy 18: Graph der Funktion f(x)=ABS(SIN(X)) Definitionsbereich 4 Vergrößerungsfaktor 2 Grad der Feinheit 0,01

```
100 'ACHTUNG! DER FUNKTIONSPLOTTER MUSS MIT 'RUN 112' GESTARTET WE
RDEN, DENN HIER STEHT DIE FUNKTIONSGLEICHUNG
102 '
104 Y=COS(X)
106 '
108 RETURN
110 '
112 BORDER 0
114 INK 0,0
116 INK 1,26
118 INK 2,6
120 INK 3,11
122 '
124 SYMBOL 255,8,8,28,28,62,62,127,8
126 '
128 ON ERROR GOTO 340
130 '
132 'PROGRAMMKOPF AUSGEBEN
134 '
```



```
136 MODE 2
138 ORIGIN 0,0
140 '
142 LOCATE 32,3
144 PRINT "FUNKTIONSPLOTTER"
146 LOCATE 33,4
148 PRINT "JST  1.10.1986"
150 '
152 PLOT 16,383,1
154 DRAW 16,320
156 DRAW 623,320
158 DRAW 623,383
160 DRAW 16,383
162 '
164 'WELCHER BEREICH SOLL BETRACHTET WERDEN
166 '
168 LOCATE 5,10
170 INPUT "WELCHEN DEFINITIONSBEREICH WUENSCHEN SIE";DEFI
172 LOCATE 5,10
174 INPUT "WELCHEN VERGROESSERUNGSFAKTOR WUENSCHEN SIE";FF
176 FF=FF*30
178 LOCATE 5,10
180 INPUT "BESTIMMEN SIE FUER DAS ZEICHNEN DEN GRAD DER FEINHEIT";
FH
182 '
184 'KOORDINATENSYSTEM ZEICHNEN
186 '
188 MODE 1
190 TAG
192 ORIGIN 320,200
194 '
196 'X-ACHSE
198 '
200 PLOT -320,0,1
202 DRAW 320,0
204 MOVER -16,6
206 PRINT CHR$(246);
208 MOVER -16,-16
210 PRINT "x";
212 '
```

```
214 'SKALIERUNG DER X-ACHSE
216 '
218 FOR I=-300 TO 300 STEP 30
220 MOVE I,4
222 DRAW I,-4
224 NEXT I
226 '
228 'EINTEILUNG DER X-ACHSE
230 '
232 MOVE 78,-8
234 PRINT USING "####.##";150/FF;
236 '
238 'Y-ACHSE
240 '
242 MOVE 0,-200
244 DRAW 0,200
246 MOVER -8,-2
248 PRINT CHR$(255);
250 MOVER -80,0
252 PRINT "f(x)";
254 '
256 'SKALIERUNG DER Y-ACHSE
258 '
260 FOR I=-180 TO 180 STEP 30
262 MOVE 4,I
264 DRAW -4,I
266 NEXT I
268 '
270 'EINTEILUNG DER Y-ACHSE
272 '
274 MOVE -120,158
276 PRINT USING "####.##";150/FF;
278 '
280 TAGOFF
282 '
284 'AUSSCHNITT KENNZEICHNEN
286 '
288 MOVE -DEFI*FF,-DEFI*FF
290 DRAWR 2*DEFI*FF,0,2
292 DRAWR 0,2*DEFI*FF
```



```
294 DRAWR -2*DEFI*FF,0
296 DRAWR 0,-2*DEFI*FF
298 '
300 'FUNKTIONSGRAPH ZEICHNEN
302 '
304 X=-DEFI
306 GOSUB 104
308 PLOT X*FF,Y*FF,3
310 '
312 FOR X=-DEFI TO DEFI STEP FH
314 GOSUB 104
316 DRAW X*FF,Y*FF,3
318 '
320 PLOT X*FF,0,2
322 MOVE X*FF,Y*FF
324 '
326 IF INKEY$<>"" THEN 336
328 '
330 NEXT X
332 '
334 IF INKEY$="" GOTO 334
336 CLS
338 GOTO 130
340 '
342 'FEHLERBEHANDLUNG
344 '
346 TAGOFF
348 PRINT CHR$(7);
350 TAG
352 '
354 RESUME NEXT
```

Programmbeschreibung:

100-108 Ganz zu Anfang des Funktionsplotters steht ein Unterprogramm, das die Funktionsgleichung enthält. Dieser Platz innerhalb des Programms wurde gewählt, damit die Funktionsgleichung - soll sie durch eine andere ersetzt werden - schnell aufge-

funden werden kann. Bei Start des Funktionsplotter durch "RUN" tritt die Fehlermeldung "Unexpected RETURN in 108" auf; Das Programm muß also wie im Listing angemerkt mit "RUN 112" gestartet werden.

112-128 Die Rahmenfarbe und die Farben für die Stifte 0 bis 3 werden gesetzt. Da sich kein geeignetes Zeichen im Charaktergenerator befindet, wird unter der Zeichennummer 255 ein Pfeil nach oben definiert, der beim Zeichnen des Koordinatensystems Verwendung finden wird.

Bei den Berechnungen, die während der Arbeit des Funktionsplotter anfallen, kann nicht ausgeschlossen werden, daß es zu Fehlern (z.B. Bereichsüberschreitung) kommt. Damit diese Fehler nicht zum Abbruch des Programms durch Fehlermeldung führen, wird mit ON ERROR GOTO festgelegt, was im Falle eines Fehlers zu tun ist.

130-160 **Programmkopf ausgeben**
Nach dem Setzen von MODE 2 und Koordinatenursprung wird der Name des Programms und die Eignerkennung auf den Bildschirm gebracht. Um diese Ausgaben wird ein Rahmen gezeichnet.

162-180 **Welcher Bereich soll betrachtet werden**
Dieser Programmteil erfragt vom Benutzer, welchen Definitionsbereich der Funktion er gerne betrachten möchte, mit welchem Vergrößerungsfaktor das Programm arbeiten und mit welcher Schrittweite der Funktionsgraph gezeichnet werden soll, was sich durch die Eingabe von FH ergibt. Alle Angaben können unmittelbar vom Programm verwertet werden, nur der Vergrößerungsfaktor wird mit 30 multipliziert, um das Zeichnen der Funktion an die Einteilung des Koordinatensystems anzupassen.

182-192 **Koordinatensystem zeichnen**
Als Vorbereitung für das Zeichnen des Koordinatensystems wird MODE 1, die Anpassung der Textausgabe an den Grafikcursor (TAG) und der Koordinatenursprung in die Mitte des Bildschirms gesetzt.

194-210 **X-Achse**
Der Programmteil X-ACHSE zeichnet mit dem Farbstift 1 in die Mitte des Bildschirms eine waagerechten Linie. An das rechte Ende der Linie wird eine Pfeilspitze und die Bezeichnung "x" gesetzt. Diese Linie stellt die X-Achse des rechtwinkligen Koordinatensystems dar.

212-224 **Skalierung der X-Achse**
Die gezeichnete X-Achse wird in Abstand von 30 Bildpunkten durch kleine senkrechte Striche unterteilt.

226-234 **Einteilung der X-Achse**
Unter Einbeziehung des Vergrößerungsfaktors FF wird an den fünften Skalierungsstrich, auf dem positiven Teil der X-Achse, der Wert geschrieben, der ihm auf dem Koordinatensystem entspricht.

236-252 **Y-Achse**
Der Programmteil Y-ACHSE zeichnet in die Mitte des Bildschirms eine senkrechte Linie. An das obere Ende der Linie wird die Pfeilspitze, die unter Zeichennummer 255 definiert wurde, und die Bezeichnung "f(x)" gesetzt. Diese senkrechte Linie stellt die Y-Achse des Koordinatensystems dar.

254-266 **Skalierung der Y-Achse**
Die gezeichnete Y-Achse wird in Abstand von 30 Bildpunkten durch kleine waagerechte Striche unterteilt.

268-280 **Einteilung der Y-Achse**
Unter Einbeziehung des Vergrößerungsfaktors FF wird an den fünften Skalierungsstrich, auf dem positiven Teil der Y-Achse, der Wert geschrieben, der ihm auf dem Koordinatensystem entspricht.

Nachdem das Koordinatensystem vollständig gezeichnet, skaliert und beschriftet ist, wird die Textausgabe wieder an die Position des Textcursors geleitet (TAGOFF).

282-296 **Ausschnitt kennzeichnen**
Mit Farbstift 2 wird der Ausschnitt des Koodinatensystems durch einen Rahmen markiert, der vom Benutzer für die Ausgabe bestimmt wurde. Die Eckpunkte des Rahmens ergeben sich durch Spiegeln des für DEFI eingegebenen Wertes, mulitipliziert mit dem Vergrößerungsfaktor FF, in alle vier Quadranten des Koordinatensystems.

298-338 **Funktionsgraph zeichnen**
Da der Funktionsgraph, um immer einen geschlossenen Linienzug zu erhalten, nicht mit dem BASIC-Kommando PLOT, sondern mit DRAW erstellt wird, muß der Startpunkt für die erste Linie ermittelt werden. Dies und das Positionieren des Grafikcursor sowie das Auswählen des Zeichenstiftes 3 wird in den Zeilen 304 bis 308 vollzogen.

Der Funktiongraph wird in der FOR-TO-NEXT-Schleife von Zeile 312 bis 330 gezeichnet. Die Schleife läuft vom negativen bis zum positiven Ende des Definitionsbereiches, wobei sich die Schrittweite durch die Variable FH, die zu Beginn des Programms erfragt wurde, bestimmt. Während des gesamten Zeichenvorgangs wird überwacht, ob eine Taste betätigt wird, was zum Abbruch des Zeichnens und zur Verzweigung an den Anfang des Programms führt.

Den Zeilen 320 und 322 in der FOR-TO-NEXT-Schleife kommt die Bedeutung zu, anzuzeigen, an welcher Stelle des Definitionsbereichs der Funktionsplotter gerade die zugehörigen Werte berechnet. Dieses Hilfsmittel schiebt sich, in Gestalt einer Reihe von roten Punkten, vom negativen Ende des Definitionsbereiches zum positiven auf der X-Achse entlang.

Nach dem Durchlaufen der FOR-TO-NEXT-Schleife bleibt die erstellte Grafik, bis zum Betätigen einer Taste auf dem Bildschirm erhalten. Wird eine Taste niedergedrückt, so löscht das Programm den Bildschirm und springt zurück nach Zeile 130, wo andere Angaben zum Zeichnen des Funktionsgraphen gemacht werden können.

340-Ende **Fehlerbehandlung**
Tritt im Verlauf der Berechnung der Funktionswerte ein Fehler auf (Bereichsüberschreitung), so verzweigt das Programm automatisch an diese Stelle, gibt einen Warnton aus und fährt mit den nächsten Wert fort.

7. 3-D-Vektorgrafik

Sicher erinnern sich noch viele von Ihnen an den Science-Fiction-Film "Krieg der Sterne". Viele Kinobesucher waren von den dort gezeigten, dreidimensionalen Computergrafiken fasziniert, die George Lucas 1977 in Zusammenarbeit mit dem heutigen Spezialeffekt-Giganten "Industrial Light and Magic" auf dem Bildschirm gezaubert hatte. Die meisten Grafiken, die damals über die Leinwände flimmerten, wurden mit einer Technik erzeugt, die sich der *vektoriellen Darstellung* bedient. Die so erstellten Computerbilder bestehen ausschließlich aus einer Vielzahl von Linien, die die Umrisse eines Körpers symbolisieren. Das Ergebnis einer derartigen Darstellung ist ein Drahtgerüst, das in der Fachsprache auch *Wire-Frame* genannt wird.

Aber wie lassen sich derartige Wire-Frames berechnen? Nun - die vektorielle Darstellung bedient sich - wie der Name schon vermuten läßt - sogenannter *Vektoren*. Ein Vektor ist eine gerichtete, orientierte Strecke im Raum. Man kann ihn auch als unsichtbaren Pfeil bezeichnen, der auf einen beliebigen Punkt in einem imaginären Universum zeigt. Und genau diese Eigenschaft machen sich viele Programmierer bei der Entwicklung einer 3-D-Grafik zunutze. Das durch die Vektoren erzeugte Wire-Frame ist nämlich die Basis für jede dreidimensional mathematisch definierte Animation. Selbst ein komplexes Vollflächenmodell ist die konsequente Weiterentwicklung eines, mittels Vektor-Technik generierten, Drahtgerüstes.

Heute bedient man sich in Werbung, Film und Wissenschaft verstärkt der dreidimensional animierten Grafik. In den USA gibt es inzwischen spezielle Grafikstudios, die sich auf die Herstellung derartiger Computerfilme spezialisiert haben. Mit gigantischen Rechenanlagen und einer Grafikauflösung von durchschnittlich 1024 mal 1024 Punkten entstehen dort Bilder, die oft unser Vorstellungsvermögen sprengen. Die von diesen Computern generierten Sequenzen bestechen immer wieder durch ihren nahezu unglaublichen Realismus und Perfektionismus. Von einem dieser amerikanischen Grafikstudios stammt

übrigens auch die neue Signation der ARD. Preis für diese Qualitätsarbeit: 4000 Dollar pro Sekunde!

Nun aber zurück zu unserem CPC. Auch bei Home-Computern wird die vektorielle Darstellung in professionellen Softwareprodukten als plastisches und visuelles Medium eingesetzt. Computerspiele wie beispielsweise ELITE[1] beziehen ihren starken Realismus erst durch die sich ständig in der Perspektive ändernden Wire-Frames.

Derartige 3-D-Animationen sind nicht nur den professionellen Softwarehäusern vorbehalten. Mit Hilfe der Vektor-Technik können wir sogar in BASIC überraschende und zugleich verblüffende 3-D-Effekte erzielen. Fließende Rotationen definierter Objekte sind selbstverständlich nur in Maschinensprache möglich. Trotzdem werden wir alle folgenden Programmbeispiele in BASIC vorstellen, denn erstens ist die Maschinenprogrammierung der notwendigen Algorithmen so kompliziert, daß ihnen ein ganzes Buch gewidmet werden müßte und zweitens hat ein BASIC-Interpreter den Vorteil, daß mit ihm leichter experimentiert werden kann. Am Ende dieses Kapitels finden Sie dann ein komplettes, in BASIC geschriebenes, 3-D-Animationsprogramm mit integriertem Editor und Demo-Mode. Da wir die Z80-Programmierer unter Ihnen nicht alleine lassen wollen, werden wir einige Tips zur Programmumsetzung in Maschinensprache vorstellen.

Damit Sie in Zukunft Ihr eigenes 3-D-Programm verwirklichen können, folgt an dieser Stelle etwas Theorie. Aber keine Angst! Diese soll nicht in einen kompletten Lehrgang der "analytischen Geometrie" ausarten. Wie beschränken uns vielmehr auf die wesentlichen Fakten und erläutern fertige Formeln, damit Sie schnellstmöglich zum gewünschten Erfolg kommen.

1 ELITE ist eingetragenes Warenzeichen der Acornsoft Ltd.

7.1 Das dreidimensionale Koordinatensystem

Wie Sie aus dem letzten Kapitel entnehmen konnten, ist das zweidimensionale Koordinatensystem aus zwei senkrecht aufeinander stehenden Achsen aufgebaut, die als X- und Y-Achse bezeichnet werden und die Dimensionen "Breite" und "Höhe" symbolisieren. Im dreidimensionalen Koordinatensystem kommt noch eine weitere Dimension dazu - die Tiefe. Sie wird durch die sogenannte *Z-Achse* dargestellt. Die folgende Abbildung soll Ihnen diese Zusammenhänge verdeutlichen:

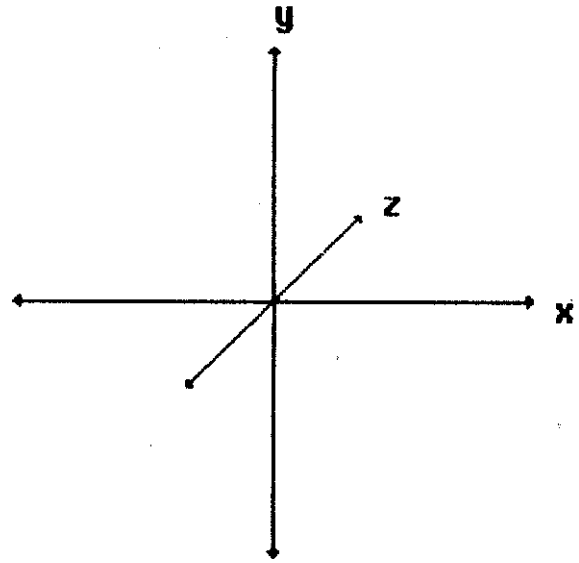


Abb. 9: Das dreidimensionale Koordinatensystem

Mit einem 3-D-Koordinatensystem ist es möglich, anhand der X-, Y-, und Z-Koordinaten jeden beliebigen Punkt in einem definierten Raum zu bestimmen. Soll beispielsweise ein Punkt im Zentrum des Koordinatensystems definiert werden, so geschieht die Koordinatenbestimmung mit Hilfe der Matrix

P(0,0,0). Die drei Martixelemente repräsentieren die X-, Y-, und Z-Koordinaten. Daraus resultiert, daß sie für die Position eines Körpers auf der ihnen zugeordneten Achse zuständig sind. P(0,0,100) bestimmt beispielsweise einen Punkt, der sich in dem vom Betrachter zugewandten Teil der Z-Achse befindet, während P(0,0,-100) ihren negativen Teil adressiert.

Die Größe des dreidimensionalen Raums hängt dabei selbstverständlich von der Länge der einzelnen Achsen des Koordinatensystems ab. Die Anzahl darin zu definierenden Koordinaten werden am besten mit Hilfe von drei eindimensionalen Feldern (Vektoren) adressiert, die problemlos mit dem DIM-Befehl ihres BASIC-Interpreters definiert werden können. Die Größe jedes Vektors ist bei diesem Verfahren nur durch den Speicherplatz Ihres Rechners begrenzt.

Beispiel:

```
100 XN=300:'Maximale Anzahl der X-Koordinaten
110 YN=300:'Maximale Anzahl der Y-Koordinaten
120 ZN=300:'Maximale Anzahl der Z-Koordinaten
130 DIM X(XN), Y(YN), Z(ZN)
```

Es gibt zwei Möglichkeiten, ein Objekt in unserem Koordinatensystem zu definieren.

1. Der gesamte Flächeninhalt eines Objekts wird in das Koordinatensystem übertragen.

2. Die Definition eines Objekts beschränkt sich auf die Koordinaten aller Eckpunkte und deren Verbindungen.

Die erste der beiden Alternativen hat den Nachteil, daß sehr viele Punkte für die Objekt-Definition in unsere Vektoren übertragen werden müssen. Eine Objekt-Rotation hätte somit einen sehr hohen Rechenaufwand zur Folge. Des weiteren sind bei dieser Methode spätere Objektänderungen so gut wie unmöglich.

Die zweite Alternative ist hingegen hervorragend für unser Vorhaben geeignet. Da alle Vektoren nur die Eckpunkte eines Objekts beinhalten, beschränken sich auf sie alle späteren Rechenoperationen der Objektanimation. Dabei kann jedoch nicht das Objekt als Ganzes dargestellt werden. Um dies zu erreichen, bedient man sich eines kleinen Tricks. Alle gespeicherten Eckpunkte werden einfach mit Linien verbunden. Um dies zu erreichen, muß ein permanenter Zugriff auf die Verbindungsdaten der Eckpunkte gewährleistet sein.

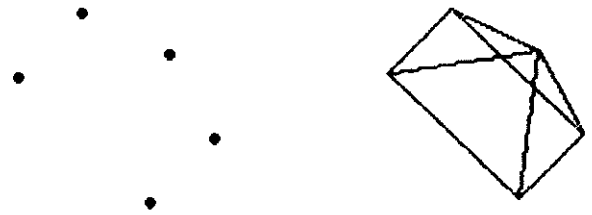


Abb 10: Eckpunkte/verbundene Eckpunkte

Unser 3-D-Programm muß also in seiner endgültigen Version neben den drei Vektoren einen weiteren Datenblock beinhalten, der die Verbindungen zwischen den einzelnen Koordinaten enthält. Er kann mit Hilfe einer zweidimensionalen Matrix erzeugt werden. Da der Umfang der Matrix kaum vorhergesagt werden kann, wählt man im Dimensionierungs-Block sinnvollerweise eine große Zahl. Der erste Parameter der Anweisung

```
DIM V(255,1)
```

gibt in diesem Beispiel die Anzahl der Verbindungen (256) an. Mit dem zweiten Parameter werden jeweils 2 Speicherstellen (0

und 1) für die zwei zu verbindenen Eckpunktnummern reserviert.

Eine zweidimensionale Matrix und drei Vektoren ermöglichen also die vollständige Definition eines Objekts im dreidimensionalen Raum. Aber wie wird es auf den Bildschirm projeziert? Die Mathematik bietet diesbezüglich zwei Lösungen an:

- Die Parallelprojektion und
- die Zentralprojektion.

Beide Projektionsarten sind für unser Vorhaben zu verwenden. Welche Sie später einmal bevorzugen werden, hängt vom Einsatzgebiet der 3-D-Grafik und Ihrem persönlichen Geschmack ab. Im Folgenden wird jedenfalls die Funktionsweise beider Verfahren, sowie die damit verbundenen Vor- und Nachteile erläutert.

7.2 Die Parallelprojektion

Diese Projektionsart ist die einfachere unserer zwei Methoden. Sie stellt Objekte in der sogenannten *Parallelperspektive* dar. Objekte, die mittels Parallelperspektive auf dem Bildschirm projiziert worden sind, beziehen sich auf keinen Fluchtpunkt; d.h. in unserem Fall, keinen definierten Punkt, worauf alle Linien zustreben. Daraus resultiert, daß die Größe eines Körpers optisch nicht mehr im dreidimensionalen Raum erfaßbar ist.

Die Parallelprojektion bietet aber auch Vorteile. Da sie mit relativ wenig Aufwand in 3-D-Programme implementiert werden kann, erreichen selbst BASIC-Programme beachtliche Projektionsgeschwindigkeiten. Darüber hinaus können Objekte in parallelperspektivischer Darstellung selbst bei geringer Bildschirmauflösung (z.B. MODE 0) plastisch wirken.

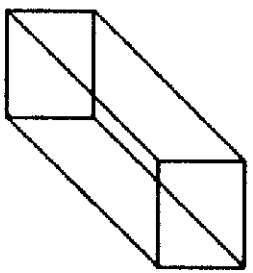


Abb. 11: Beispiel für eine Parallelprojektion

Aber nun zum eigentlichen Projektionsverfahren. Da der Bildschirm nur ein zweidimensionales Medium ist, können Grafiken folglich nur zweidimensional ausgegeben werden. Um einen "Quasi-3-D-Effekt" zu erzielen, muß die Grafik auf dem Bildschirm verzerrt werden. Das bedeutet in unserem Fall, die X- und Y-Koordinaten der Start- und Endpunkte aller Linien eines definierten Objekts mit den ihnen zugeortneten Z-Parametern zu manipulieren. Folgende Formeln ermöglichen eine derartige Transformation:

```
XP=X(V(I,J))+Z(V(I,J))*SIN(ALPHA)
YP=Y(V(I,J))+Z(V(I,J))*SIN(BETA)
```

In beiden Fällen wird zum aktuellen, durch die Verbindungsmatrix V(I,J) indizierten Vektor-Element, das Produkt aus Z-Koordinate und Betrachtungswinkel addiert. Er wird durch die Variablen ALPHA und BETA angegeben. Damit Sie sich eine bessere Vorstellung von der Wirkung der Formeln machen können, präsentieren wir Ihnen an dieser Stelle ein kleines Demonstrationsprogramm, mit einem bereits fertig definierten Objekt. Dessen Eckpunkte und Verbindungslinien sind in DATA-Blökken gespeichert und können von Ihnen leicht manipuliert wer-

den. Experimentieren Sie ruhig einmal mit verschiedenen Blickwinkeln und Objekten und Sie werden erkennen, wie faszinierend die 3-D-Grafik sein kann.

```
10 '##############################
20 '##                          ##
30 '## PARALLELPROJEKTIONS-DEMO ##
40 '##       7.9.'86 TAV        ##
50 '##                          ##
60 '##############################
70 '
80 '
100 'INITIALISIERUNG
110 '
120 MODE 1
130 DEG
140 ORIGIN 320,200
150 DEFINT A-Z
160 DIM X(255),Y(255),Z(255),V(255,1)
170 '
180 'KOORDINATEN EINLESEN
190 '
200 READ J
210 FOR I=0 TO J:READ X(I),Y(I),Z(I):NEXT
220 '
230 'VERBINDUNGEN EINLESEN
240 '
250 READ VB
260 FOR I=0 TO VB:READ V(I,0),V(I,1):NEXT
270 '
280 'BLICKWINKEL EINGEBEN
290 '
300 CLS
310 PRINT"BITTE BETRACHTUNGSWINKEL EINGEBEN"
320 PRINT:PRINT
330 INPUT"ALPHA (0-360)";ALPHA
340 INPUT"BETA (0-360)";BETA
350 CLS
360 '
```



```
370 'PROJEKTIONS-SCHLEIFE
380 '
390 FOR I=0 TO VB
400 J=0:GOSUB 510
410 MOVE XP,YP
420 J=1:GOSUB 510
430 DRAW XP,YP
440 NEXT
450 '
460 LOCATE 8,25:PRINT"BITTE EINE TASTE DRUECKEN!";
470 IF INKEY$="" THEN 470 ELSE 300
480 '
490 '3-D-TRANSFORMATION
500 '
510 XP=X(V(I,J))+Z(V(I,J))*SIN(ALPHA)
520 YP=Y(V(I,J))+Z(V(I,J))*SIN(BETA)
530 RETURN
540 '
550 'KOORDINATEN
560 '
570 DATA 7:'ANZAHL DER KOORDINATEN MINUS EINS
575 '
580 DATA -50,-50,-50
590 DATA -50,50,-50
600 DATA 50,50,-50
610 DATA 50,-50,-50
620 DATA -50,-50,50
630 DATA -50,50,50
640 DATA 50,50,50
650 DATA 50,-50,50
660 '
670 'VERBINDUNGEN
680 '
690 DATA 13:'ANZAHL DER VERBINDUNGEN MINUS EINS
695 '
700 DATA 0,1
710 DATA 1,2
720 DATA 2,3
730 DATA 3,0
740 DATA 4,5
```

```
750 DATA 5,6
760 DATA 6,7
770 DATA 7,4
780 DATA 0,4
790 DATA 1,5
800 DATA 2,6
810 DATA 3,7
820 DATA 0,2
830 DATA 1,3
```

Programmbeschreibung

100-160 **Initialisierung**
Die Grafikauflösung 320x200 Punkte wird gewählt, die Bogenmaß-Kalkulation ausgeschaltet und alle Vektoren und Matrizen dimensioniert. Weiterhin werden alle Variablen als Integer deklariert (DEFINT) und der Koordinatenursprung in die Mitte des Bildschirms verlegt.

170-260 **Daten einlesen**
Die DATA-Elemente werden in die Eckpunkt-, Verbindungstabelle geladen. Mit dem READ-Befehl vor jeder Schleife werden jeweils die Anzahl der Datenelemente-1, welche die in der nächsten Zeile stehende FOR-NEXT-Schleife einzulesen hat, in die zugeordnete Maxline-Variable geladen (J und VB).

270-350 **Blickwinkel definieren**
Der Blickwinkel in Grad wird mittels INPUT-Befehl in die Variablen ALPHA und BETA eingelesen. Danach wird der Bildschirm gelöscht.

360-470 **Projektions-Schleife**
Sie holt alle Linien eines Objekts aus den Tabellen und gibt die Indizes der Start- und Endkoordinaten an die 3-D-Transformationsroutine (GOSUB 510) weiter. Die in XP und YP gewonnenen Koordinaten X1, X2, Y1 und Y2 werden in den Zeilen 410 und 430 für das Zeichnen der Linien verwendet.

Nach Fertigstellung des Bildes wartet die Routine in Zeile 470 so lange, bis eine beliebige Taste gedrückt wird. Wurde eine Taste betätigt, erfolgt ein Rücksprung in die Blickwinkel-Definitionsroutine.

480-530 **3-D-Transformation**
Diese Routine berechnet die X- und Y-Position eines Eckpunktes auf dem Bildschirm in Parallelperspektive.

540-650 **Koordinaten**
Das erste DATA-Element gibt die Anzahl der folgenden DATA-Zeilen-1 an.

660-Ende **Verbindungen**
Das erste DATA-Element gibt die Anzahl der folgenden DATA-Zeilen-1 an.

Soll das Objekt zusätzlich im Raum verschoben werden, müssen alle Koordinaten mit einem entsprechenden Offset versehen werden. Dieser Offset kann sowohl positiv als auch negativ sein. Nach seiner Implementierung ergeben sich folgende Formeln:

```
XP=X(V(I,J))+XOFF+(Z(V(I,J))+ZOFF)*SIN(ALPHA)
YP=Y(V(I,J))+YOFF+(Z(V(I,J))+ZOFF)*SIN(BETA)
```

Falls ein verschobendes Objekt wieder in seine Ausgangsposition zurückgebracht werden soll, muß XOFF, YOFF und ZOFF eine 0 zugewiesen werden.

7.2.1 Rotation im Raum

Bis jetzt war es uns nur möglich, ein dreidimensionales Objekt auf dem Bildschirm zu verschieben und es dabei aus verschiedenen Blickwinkeln zu betrachten. Obwohl mit den daraus resultierenden grafischen Ergebnissen ein starker dreidimensionaler Eindruck erzielt wird, kann dieser noch weiter verbessert werden. Zu einer realistischen dreidimensionalen Animation gehört nämlich die freie Rotation im Raum. Damit Sie ein Objekt um eine der drei Achsen rotieren lassen können, müssen die 3-D-Koordinaten in entsprechende gedrehte Koordinaten umgerechnet werden. Diese Transformation kann pro Achse mit zwei Formeln realisiert werden.

Rotation um die X-Achse:

```
YR=Y(V(I,J))*COS(ALPHA)+Z(V(I,J))*SIN(ALPHA)
ZR=-Y(V(I,J))*SIN(ALPHA)+Z(V(I,J))*COS(ALPHA)
```

Rotation um die Y-Achse:

```
XR=X(V(I,J))*COS(BETA)+Z(V(I,J))*SIN(BETA)
ZR=-X(V(I,J))*SIN(BETA)+Z(V(I,J))*COS(BETA)
```

Rotation um die Z-Achse:

```
XR=X(V(I,J))*COS(GAMMA)+Y(V(I,J))*SIN(GAMMA)
YR=-X(V(I,J))*SIN(GAMMA)+Y(V(I,J))*COS(GAMMA)
```

Den Variablen ALPHA, BETA und GAMMA muß vor Ansprung der Berechnungsroutine ein Rotationswinkel zugewiesen werden. In diesem Zusammenhang ist unbedingt zu beachten, daß die Rotation nicht gleichzeitig über alle Achsen erfolgen kann. Falls Sie dies dennoch wünschen, müssen die vom ersten Formelpaar gewonnenen Daten in die nächsten beiden Gleichungen einbezogen werden.

7.3 Die Zentralprojektion

Obwohl unser Projektionsverfahren an dieser Stelle perfekt zu sein scheint, wirkt die Objekt-Rotation in einigen Fällen unrealistisch. Warum? Nun - wenn wir einen Körper aus unterschiedlichen Entfernungen und Perspektiven betrachten, werden wir feststellen, daß seine Entfernung vom menschlichen Auge in einer Beziehung zu seinem gesamten äußeren Erscheinungsbild steht. Dieses Phänomen basiert auf der Tatsache, daß alle erfaßbaren Konturen eines Gegenstandes auf einen von unseren Augen definierten Punkt am Horizont - dem sogenannten *Fluchtpunkt* - zustreben. Das wiederum hat zur Folge, daß uns ein Objekt umso kleiner erscheint, desto näher es sich an diesem fiktiven Fluchtpunkt zu befinden scheint. Und gerade dieser Effekt, der in der Fachsprache auch "track-effect" genannt wird, hilft dem menschlichen Gehirn die Ausmaße eines Gegenstands und seine Entfernung vom Beobachter zu ermitteln. Er ist folglich auch für den fehlenden Realismus bei unserem bisherigen Projektionsverfahren verantwortlich.

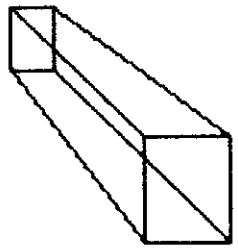


Abb. 12: Beispiel für eine Zentralprojektion

Die Zentralprojektion ermöglicht uns im Gegensatz zur Parallelprojektion, einen fiktiven Fluchtpunkt in unserem dreidimensionalen Koordinatensystem festzulegen. Dieses Verfahren ist leider

extrem komplex und somit sehr rechen- und zeitintensiv. Es ist aber letztendlich die einzige Möglichkeit, eine perfekte dreidimensionale Struktur auf dem Bildschirm zu erzeugen. Mit den folgenden Formeln kann ein definiertes Objekt in Zentralperspektive auf dem Display projiziert werden. Bitte lassen Sie sich nicht durch die vielen benötigen Formeln und deren Komplexität abschrecken! Wir werden später einen kleinen Trick vorstellen, der die hohe Rechenzeit stark minimiert.

```
P0 = COS(GAMMA)*COS(BETA)
P1 = SIN(GAMMA)*COS(BETA)
P2 =-SIN(BETA)
P3 =-SIN(GAMMA)*COS(ALPHA)+COS(GAMMA)*SIN(BETA)*SIN(ALPHA)
P4 = COS(GAMMA)*COS(ALPHA)+SIN(GAMMA)*SIN(BETA)*SIN(ALPHA)
P5 = COS(BETA)*SIN(ALPHA)
P6 = SIN(GAMMA)*SIN(ALPHA)+COS(GAMMA)*SIN(BETA)*COS(ALPHA)
P7 =-COS(GAMMA)*SIN(ALPHA)+SIN(GAMMA)*SIN(BETA)*COS(ALPHA)
P8 = COS(BETA)*COS(ALPHA)

ZP=1-(X(V(I,J)*P2+Y(V(I,J))*P5+Z(V(I,J))*P8+ZOFF)/FP
XP=(X(V(I,J)*P0+Y(V(I,J))*P3+Z(V(I,J))*P6+XOFF)/ZP
YP=(X(V(I,J)*P1+Y(V(I,J))*P4+Z(V(I,J))*P7+YOFF)/ZP
```

Eingabeparameter:

ALPHA = Winkel der Rotation um die X-Achse (0 bis 360 Grad)
BETA = Winkel der Rotation um die Y-Achse (0 bis 360 Grad)
GAMMA = Winkel der Rotation um die Z-Achse (0 bis 360 Grad)

FP = Fluchtpunkt

V = Verbindungslinien-Matrix
I = Verbindungslinien-Nummer (0 bis n-1)
J = Index für Start- und Endpunkt einer Verbindung (0 bis 1)

X(n-1) = X-Vektor
Y(n-1) = Y-Vektor
Z(n-1) = Z-Vektor

Ausgabeparameter:

XP = X-Koordinate auf Bildschirm
YP = Y-Koordinate auf Bildschirm

7.4 Tricks zur schnelleren Objektanimation

Winkelfunktions-Tabellen

Da die dreidimensionale Vektorgrafik mit Hilfe von Winkelfunktionen erzeugt wird, steigt die Rechenzeit, die für den Aufbau eines Bildes notwendig ist, schnell auf mehrere Sekunden. Dieser unerwünschte Nebeneffekt tritt insbesondere im Zusammenhang mit der Zentralperspektive auf. Um die zwangsläufig entstehenden Rechenzeiten auf ein vernünftiges Maß zu reduzieren, kann auf Winkelfunktions-Tabellen zurückgegriffen werden. Sie verhindern, daß während der Objektprojektion zeitintensive Sinus- und Cosinus-Berechnungsroutinen angesprungen werden müssen und machen somit erst die Objektanimation auf Microcomputern möglich.

Die Implementierung der benötigen Sinus- und Cosinus-Tabellen ist sehr einfach. Im Initialisierungsblock Ihres Programms müssen mittels DIM-Anweisung zwei weitere Vektoren mit dem Wert 360 dimensioniert werden: SI! und CO!.

```
DIM SI!(360),CO!(360)
```

SI! repräsentiert in unserem Beispiel die Sinus- und CO! die Cosinus-Tabelle. Der Wert 360 entspricht 360 Grad. Unmittelbar

danach empfiehlt es sich, beide Tabellen zu generieren. Die BASIC-Zeile

```
FOR I=0 TO 360:SI!(I)=SIN(I):CO!(I)=COS(I):NEXT
```

erledigt für uns diese Aufgabe. Jetzt muß in der Berechnungsroutine nur noch SIN und COS durch SI! und CO! ersetzt werden - fertig!

Von diesem Zeitpunkt an braucht Ihr 3-D-Programm nie wieder eine Winkelfunktion zu berechnen. Falls ein derartiger Wert benötigt wird, holt es sich einfach automatisch den benötigten Wert aus einer der beiden Tabellen. Den sich daraus ergebenden Geschwindigkeitsvorteil werden Sie sehr bald zu schätzen wissen.

Flackerfreie Animation

Die dreidimensionale Wirkung einer Vektorgrafik wird leider durch den bei der Objektanimation entstehenden Bildauf- und -abbau stark eingeschränkt. Bei komplizierten Gitterstrukturen kann dieser Effekt den Bewegungseindruck sogar völlig zerstören. Dem kann - wenn mit zwei voneinander unabhängigen Bildschirmseiten gearbeitet wird - abgeholfen werden. Zu diesem Zweck müssen die 16K der zweiten Bank (&4000 bis &7FFF) mittels MEMORY-Befehl reserviert und als alternative Bildschirmseite definiert werden.

Hinweis: Die Besitzer eines CPC-6128 können selbstverständlich mit den BANK-Befehlen |SCREENCOPY und |SCREENSWAP denselben Effekt erzielen. Aus Gründen der Kompatibilität möchten wir Ihnen jedoch an dieser Stelle ein allgemein gültiges Verfahren vorstellen. Es arbeitet genauso schnell wie eine mit Screen-Befehlen realisierte Routine und ist auf allen CPC-Rechnern lauffähig.

Nachdem der RAM-Bereich &4000 bis &7FFF mit dem Memory-Befehl geschützt wurde, schaltet die Sequenz

```
OUT &BC00,12:OUT &BD00,16
```

auf die alternative Bildschirmseite (&4000) und

```
OUT &BC00,12:OUT &BD00,52
```

zurück auf die normale Bildschirmseite (&C000). Durch beide OUT-Sequenzen wird eine gezielte Manipulation des Video-Controllers vorgenommen. Das ist in unserem Fall wesentlich effektiver, als einen Vektor zu benutzen, da letzterer zusätzlich alle Grafikfunktionen auf den neuen Bildschirmbereich fixiert - und genau das soll ja vermieden werden.

Nun brauchen wir nur noch eine kleine Maschinenroutine, die den gesamten Bildschirminhalt der normalen Bildschirmseite in unsere alternative Bildschirmseite kopiert. Der Befehlsschatz der Z80-CPU beinhaltet erfreulicherweise einen Block-Move-Befehl, - den LDIR-Befehl welcher diese Arbeit für uns erledigt. Er muß nur wie folgt initialisiert werden:

Der Z80-Code:

```
LD HL,&C000     'Kopiere die normale Screen
LD DE,&4000     'in die alternative Screen
LD BC,&4000     'Kopiere insgesamt 16K
LDIR            'und zwar jetzt!
RET             'Fertig!
```

Der BASIC-Lader:

```
100 FOR I=0 TO 11:READ J:POKE &3FF4+I,J:NEXT
200 DATA &21,&00,&C0,&11,&00,&40,&01,&00,&40,&ED,&B0,&C9
```

Nun müssen die einzelnen Schritte nur noch in der folgenden Reihenfolge aufgerufen werden:

1. auf normale Bildschirmseite umschalten
2. kopiere mittels LDIR-Befehl die normale Bildschirmseite (&C000) nach Bank 1 (&4000)
3. auf alternative Bildschirmseite umschalten
4. normalen Bildschirm löschen
5. Grafik auf normale Bildschirmseite erstellen
6. Schritte 1 bis 5 so lange wiederholen, bis die Animation beendet ist.

Eine äquivalente Routine läßt sich so mit diesen Informationen leicht entwickeln:

```
100 OUT &BC00,12:OUT &BD00,52
110 CALL LDIR
120 OUT &BC00,12:OUT &BD00,16
130 REM *** Animation ***
            .
            .
            .
999 GOTO 100
```


7.5 CPCs-World

An dieser Stelle möchten wir Ihnen ein komplettes Programm für die dreidimensional-animierte Vektorgrafik vorstellen. Es ermöglicht, ein definiertes Drahtmodell um alle Achsen des Koordinatensystems zu rotieren und zu verschieben. Als Projektionsverfahren wurde die Zentralprojektion gewählt. Das Programm trägt den Namen "CPCs World" und ist in fünf Sektionen unterteilt.

1. Objekt-Generator
2. Objekt-Editor
3. Animations-Editor
4. 3-D-Animationsmodus
5. Laden, Sichern und Löschen eines Objekts

Wie Sie aus dieser Aufstellung entnehmen können, wurde bei der Programmentwicklung besonderen Wert auf die möglichst einfache Generierung einer Animationssequenz eines Objekts gelegt. Das Programm ist so aufgebaut, daß einzelne Routinen in eigene Programme transferiert werden können. Es soll ferner alle grundlegenden Möglichkeiten demonstrieren, die in der Programmiersprache BASIC realisiert werden können.

Nach Starten des Programms erscheint das Hauptmenü auf dem Bildschirm:

```
########################################
##                                    ##
##       CPC's WORLD/TAV 8.9.86       ##
##                                    ##
########################################


              0 Objekt-Generator
              1 Eckpunkt-Editor
              2 Verbindungs-Editor
              3 Animations-Editor
              4 Animation
              5 Objekt sichern
              6 Objekt laden
              7 Objekt loeschen
              8 Demo laden
              9 Programm beenden


          Bitte waehlen Sie eine Funktion
```

Mit den Ziffertasten 0 bis 9 kann in einem der Zehn zur Verfügung stehenden Menüpunkte verzweigt werden. Nach Beendigung jeder Routine erfolgt ein automatischer Rücksprung in das Hauptmenü.

Der nun folgende Text befaßt sich mit den Möglichkeiten, sowie der Anwendung jedes einzelnen Menüpunktes. Eine komplette Programmbeschreibung finden Sie im Anschluß des danach folgenden Listings.

Objekt-Generator

Der Objekt-Generator ermöglicht auf einfachste Weise, ein komplexes Drahtmodell zu definieren. Nach der Menüauswahl erscheint ein zweidimensionales Koordinatensystem sowie ein Fadenkreuz auf dem Bildschirm. Das Kreuz kann mit den Cursortasten in alle Richtungen bewegt werden.

Durch Betätigen der SPACE-Taste werden die aktuellen Koordinaten des Fadenkreuzes gespeichert und mit dem Buchstaben "X" markiert. Danach wird von ihnen automatisch eine Linie zu der zuletzt mit SPACE definierten Position gezogen. Mit dieser Methode kann die zweidimensionale Hälfte eines dreidimensionalen Körpers definiert werden. Die folgende Abbildung verdeutlicht eine derartige Definition anhand der fetten Linien. Die Sterne "*" geben an, an welchen Stellen die SPACE-Taste gedrückt wurde.

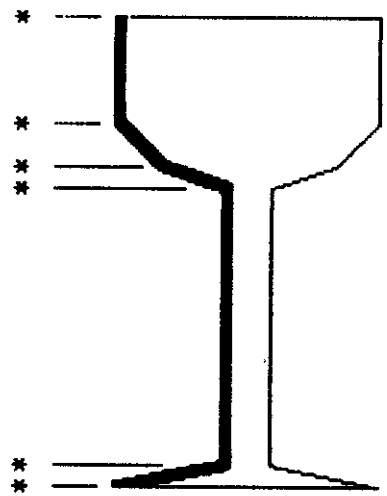


Abb. 13: Definition eines Kelches

Die Generierung eines Objekts wird durch das Betätigen der RETURN-Taste abgeschlossen. Es wird nun um die Y-Achse gespiegelt und in einem anschließenden Arbeitsgang automatisch vernetzt.

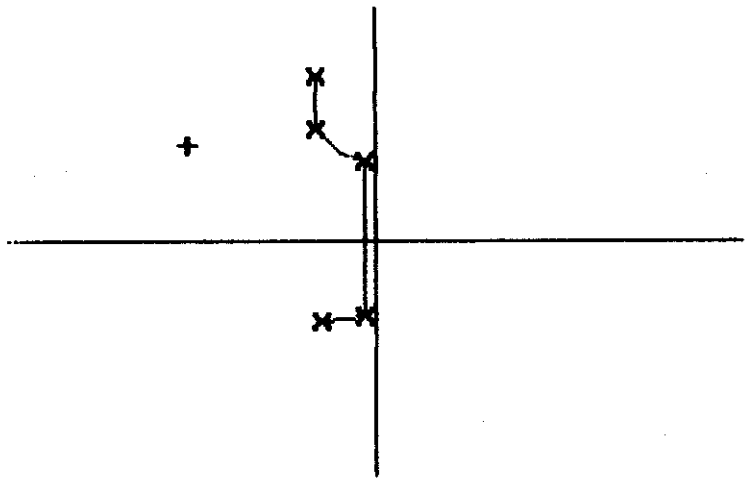


Abb 14: Objekt-Generator

Eckpunkt-Editor

Objekte können ebenfalls mit dem Eckpunkt-Editor definiert werden. Nach Auswahl des Menüpunktes "1" erscheint die Frage

```
Ab Eckpunkt?
```

auf dem Bildschirm. Das Programm erwartet an dieser Stelle eine Zahl zwischen 0 und der Nummer des ersten, noch nicht definierten Eckpunktes. Ist sie zu groß, wird dies durch die Fehlermeldung

```
*** Eckpunkt nicht vorhanden! ***
```


kenntlich gemacht. Falls die RETURN-Taste ohne vorhergehende Eingabe betätigt wird, springt die Routine immer nach Eckpunkt Null. Nach der Definition des ersten zu bearbeitenden Eckpunktes erscheint eine Eckpunkt-Tabelle auf dem Bildschirm.

ECKPUNKT	X	Y	Z	
0	-85	94	0	Aendern?
1	-66	64	0	Aendern?
2	-66	30	0	Aendern?
3	-28	30	0	Aendern?
4	-28	-39	0	Aendern?
5	-62	-39	0	Aendern?

Tabelle 2: Typische Eckpunkt-Tabelle

Auf der linken Seite der Tabelle steht die aktuelle Eckpunktnummer. Daneben wird ihr Inhalt, der aus einer X-, Y-, und Z-Position besteht, angezeigt. Weiterhin fragt Sie die Routine mit der Bildschirmmeldung "Aendern?", ob der Inhalt des Eckpunktes geändert werden soll. Als Eingabeparameter akzeptiert das Programm folgende Antworten:

- "J" (Ja) bestätigt Ihren Änderungswunsch und verzweigt in eine Editor-Routine.
- "Q" (Quitt) bricht die Funktion ab und verzweigt zurück in das Hauptmenü.
- RETURN oder eine andere Taste verzweigt zum nächsten Tabelleneintrag.

Mit dem "Eckpunkt-Editor" können maximal 128 Eckpunkte definiert werden.

Verbindungs-Editor

Diese Routine wird benötigt, um Verbindungen zwischen den mittels Eckpunkt-Editor definierten Punkten herzustellen. Nach Auswahl des Menüpunktes "2" erscheint die Frage

```
Ab Verbindung?
```

auf dem Bildschirm. Das Programm erwartet an dieser Stelle eine Zahl zwischen 0 und der Nummer der ersten, noch nicht definierten Verbindung. Ist sie zu groß, wird dies durch die Fehlermeldung

```
*** Verbindung nicht vorhanden! ***
```

kenntlich gemacht. Falls die RETURN-Taste ohne vorhergehende Eingabe betätigt wird, springt die Routine immer nach Verbindung Null. Nach der Definition der ersten zu bearbeitenden Verbindung erscheint eine Tabelle auf dem Bildschirm.

VERBINDUNG	P1	P2	
0	0	6	Aendern?
1	6	12	Aendern?
2	12	18	Aendern?
3	18	0	Aendern?
4	1	7	Aendern?
5	7	13	Aendern?

Tabelle 3: **Typische Verbindungslinien-Tabelle**

Auf der linken Seite der Tabelle steht die aktuelle Verbindungslinien-Nummer. Daneben wird ihr Inhalt, der aus Eckpunktnummern besteht, angezeigt. Weiterhin fragt Sie das Programm mit der Bildschirmmeldung "Aendern?", ob die Verbindung geändert werden soll. Die Routine akzeptiert die gleichen Antworten wie der oben beschriebende Eckpunkt-Editor. Mit dem "Verbindungs-Editor" können maximal 128 Verbindungen definiert werden.

Animations-Editor

Der Bewegungsablauf eines fertig generierten Objekts muß mit Hilfe des Animations-Editors definiert werden. Er ermöglicht die Definition von 128 Animationsschritten, wobei jeder Animationsschritt aus 6 Teilschritten besteht. Der Animations-Editor wird durch den Punkt "3" im Hauptmenü aufgerufen. Unmittelbar danach erscheint die Frage

```
Ab Zeile?
```

auf dem Bildschirm. Das Programm erwartet an dieser Stelle eine Zahl zwischen 0 und der Nummer der ersten, noch nicht definierten Programmzeile. Ist sie zu groß, wird dies durch die Fehlermeldung

```
*** Zeile nicht vorhanden! ***
```

kenntlich gemacht. Falls die RETURN-Taste ohne vorhergehende Eingabe betätigt wird, springt die Routine immer nach Zeile Null. Nach der Angabe einer gültigen Zeilennummer erscheint eine Animations-Tabelle auf dem Bildschirm.

ZEILE	XROT	YROT	ZROT	XPOS	YPOS	ZPOS	
0:	0	0	0	0	0	0	Aendern?
1:	1	1	0	0	0	15	Aendern?
2:	2	2	0	0	0	30	Aendern?
3:	3	3	0	0	0	45	Aendern?
4:	4	4	0	0	0	60	Aendern?
5:	5	5	0	0	0	75	Aendern?
6:	6	6	0	0	0	90	Aendern?
7:	7	7	0	0	0	105	Aendern?
8:	8	8	0	0	0	120	Aendern?
9:	9	9	0	0	0	135	Aendern?
10:	10	10	0	0	0	150	Aendern?

Tabelle 4: Typische Animations-Tabelle

Auf der linken Seite der Tabelle steht die aktuelle Zeilennummer. Daneben wird ihr Inhalt, der aus jeweils drei Rotationswinkeln und Verschiebeparametern besteht, angezeigt. Weiterhin fragt Sie die Routine mit der Bildschirmmeldung "Zeile Aendern?", ob der Inhalt der Zeile geändert werden soll. Als Eingabeparameter akzeptiert das Programm folgende Antworten:

- "J" (Ja) bestätigt Ihren Änderungswunsch und verzweigt in eine Editor-Routine.
- "Q" (Quittieren) bricht die Funktion ab und verzweigt zurück in das Hauptmenü.
- RETURN oder eine andere Taste verzweigt zum nächsten Tabelleneintrag.

Falls durch Betätigung der Taste "J" die Animations-Zeile geändert werden soll, müssen als erstes die Rotationswinkel XROT, YROT und ZROT definiert werden. Jeder Winkel im Bereich von 0 bis 360 Grad beschreibt die Rotation des Objekts um die X-, Y-, oder Z-Achse. Die Routine erwartet als Eingabeparameter jeweils einen positiven Wert zwischen 0 und 23. Er repräsentiert eine Konstante, die den Rotationswinkel in 15-Grad-Schritten angibt. Mit Hilfe folgender Umrechnungstabelle kann eine benötigte Konstante problemlos ermittelt werden.

Winkel	Konstante	Winkel	Konstante
0	0	195	13
15	1	210	14
30	2	225	15
45	3	240	16
60	4	255	17
75	5	270	18
90	6	285	19
105	7	300	20
120	8	315	21
135	9	330	22
150	10	345	23
165	11	360	0
180	12		

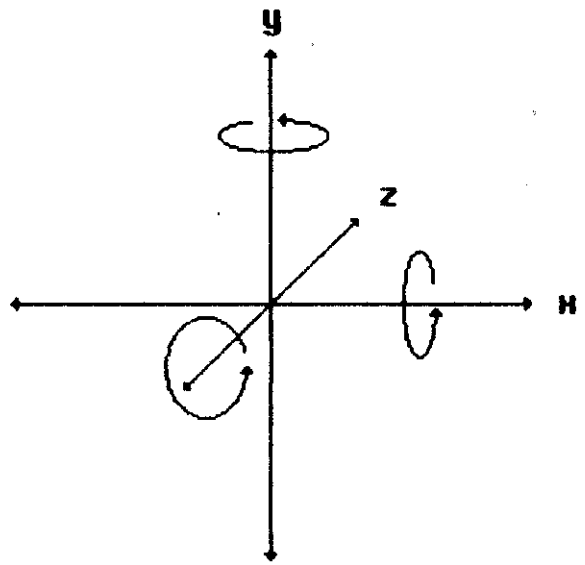


Abb. 15: Wirkung der Rotationswinkel im Koordinatensystem

Als nächstes verlangt der Editor die Eingabe der XPOS-, YPOS- und ZPOS-Verschiebeparameter. Sie geben den Abstand des Objekts auf der X-, Y-, und Z-Achse vom Nullpunkt an und können sowohl positiv als auch negativ sein.

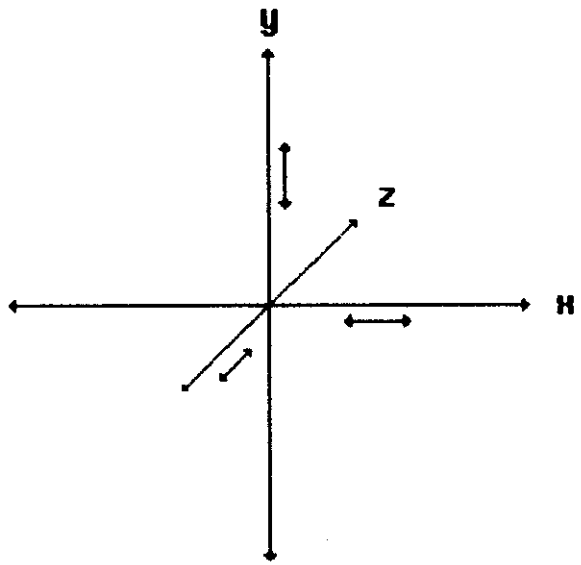


Abb. 16: Wirkung der Verschiebeparameter im Koordinatensystem

Eine auf diese Weise definierte Animation wird durch die folgende Funktion ausgeführt.

Animation

Diese Routine animiert ein definiertes Objekt mit den im Animations-Editor vorhandenen Daten. Nach Abarbeitung der letzten Animationssequenz erfolgt ein automatischer Rücksprung zum Hauptmenü.

Objekt sichern

Dieser Menüpunkt ermöglich die dauerhafe Sicherung eines Objekts, d.h. alle Eckpunkte und Verbindungen sowie die von Ihnen erstellte Animationssequenz auf Diskette. Nach Anwahl von "Objekt sichern" erscheint die Aufforderung

```
Bitte Dateinamen eingeben?
```

auf dem Bildschirm. Nach Eingabe des Dateinamens erfolgt die Objektsicherung mit einem anschließendem Rückspung in das Haupmenü.

Objekt laden

Durch diese Routine wird ein Objekt, das mit der "Objekt sichern"-Funktion gespeichert wurde geladen. Nach Anwahl von "Objekt laden" erscheint die Aufforderung

```
Bitte Dateinamen eingeben?
```

auf dem Bildschirm. Nach Eingabe des Dateinamens erfolgt der Ladevorgang mit einem anschließendem Rückspung in das Haupmenü.

Objekt löschen

Diese Fuktion setzt die Variablen der Eckpunkt-, Verbindungs- und Animations-Tabellen sowie aller Pointer auf ihre Default-Werte.

Demo laden

Lädt ein Objekt sowie eine einfache Animation in der Speicher.

Programm beenden

Beendet "CPCs World" mittels END-Befehl.

```
100 '##############################
110 '##                          ##
120 '##       CPCs  WORLD        ##
130 '##       8.9.'86 TAV        ##
140 '##                          ##
150 '##############################
160 '
170 '
180 'INITIALISIERUNG
190 '
200 MEMORY &3FF3
210 DEG
220 DEFINT A-Z
230 DIM X(127),Y(127),Z(127),V(127,1),SI!(23),CO!(23),ALPHA(127),BETA(12
7),GAMMA(127),XOFF(127),YOFF(127),ZOFF(127)
240 FP=640
250 '
260 'BLOCK-MOVE-ROUTINE IN DEN SPEICHER LADEN
270 '
280 FOR I=0 TO 11:READ J:POKE &3FF4+I,J:NEXT
290 '
300 DATA &21,&00,&C0:    'LD HL,&C000 [1]
310 DATA &11,&00,&40:    'LD DE,&4000
320 DATA &01,&00,&40:    'LD BC,&4000
330 DATA &ED,&B0:        'LDIR
340 DATA &C9:            'RET
350 '
360 'SIN- UND COS-TABELLEN GEGERIEREN
```

1 Remarks in den DATA-Zeilen verursachen beim 464 einen Syntax-Error.


```
370 '
380 FOR I=0 TO 345 STEP 15:SI!(I/15)=SIN(I):CO!(I/15)=COS(I):NEXT
390 '
400 'HAUPMENUE
410 '
420 MODE 1
430 PRINT"########################################";
440 PRINT"##                                    ##";
450 PRINT"##";:PEN 3:PRINT"       CPCs WORLD/TAV 8.9.86       ";:PEN 1:
PRINT"##";
460 PRINT"##                                    ##";
470 PRINT"########################################";
480 PRINT:PRINT:PRINT:PRINT:PRINT
490 PEN 3:PRINT TAB(10)"0 ";:PEN 1:PRINT"Objekt-Generator"
500 PEN 3:PRINT TAB(10)"1 ";:PEN 1:PRINT"Eckpunkt-Editor"
510 PEN 3:PRINT TAB(10)"2 ";:PEN 1:PRINT"Verbindungs-Editor"
520 PEN 3:PRINT TAB(10)"3 ";:PEN 1:PRINT"Animations-Editor"
530 PEN 3:PRINT TAB(10)"4 ";:PEN 1:PRINT"Animation"
540 PEN 3:PRINT TAB(10)"5 ";:PEN 1:PRINT"Objekt sichern"
550 PEN 3:PRINT TAB(10)"6 ";:PEN 1:PRINT"Objekt laden"
560 PEN 3:PRINT TAB(10)"7 ";:PEN 1:PRINT"Objekt loeschen"
570 PEN 3:PRINT TAB(10)"8 ";:PEN 1:PRINT"Demo laden"
580 PEN 3:PRINT TAB(10)"9 ";:PEN 1:PRINT"Programm beenden"
590 PEN 3:PRINT:PRINT:PRINT:PRINT TAB(4)"Bitte waehlen Sie eine Funktion
";:PEN 1
600 Y$=INKEY$
610 IF Y$<"0" OR Y$>"9" THEN 600
620 CLS:ON VAL(Y$)+1 GOTO 660,1310,1420,1200,1670,990,870,1110,1530,2680
630 '
640 'OBJEKT GENERIEREN
650 '
660 GOSUB 1120:MOVE 0,200:DRAW 639,200,3:MOVE 320,0:DRAW 320,399,3
670 ORIGIN 320,200:K=0:I=-160:J=100:PRINT CHR$(23);CHR$(1) ;
680 TAG
690 GRAPHICS PEN 3:MOVE I,J:PRINT "+";
700 Y$=INKEY$:IF Y$="" THEN 700
710 MOVE I,J:PRINT "+";
720 IF Y$=CHR$(240) THEN J=J+1
730 IF Y$=CHR$(241) THEN J=J-1
740 IF Y$=CHR$(242) THEN I=I-1
```

```
750 IF Y$=CHR$(243) THEN I=I+1
760 IF Y$=" " THEN X(K)=I+8:Y(K)=J-6:K=K+1:MOVE I,J:GRAPHICS PEN 2:PRINT
"X";:IF K>1 THEN MOVE X(K-2),Y(K-2):DRAW X( K-1),Y(K-1),1
770 IF Y$=CHR$(13) THEN 790
780 GOTO 690
790 FOR I=0 TO K-1:X(2*K+I)=X(I)*-1:Y(2*K+I)=Y(I):Z(K+I)=X
(I):Y(K+I)=Y(I):Z(3*K+I)=X(I)*-1:Y(3*K+I)=Y(I):NEXT
800 AK=K*4-1
810 X0=0:FOR I=0 TO AK STEP 4:FOR J=0 TO
3:V(I+J,0)=J*K+X0:V(I+J,1)=(J+1)*K+X0:NEXT:V(I+J-1,1)=X0:X0=X0+1:NEXT
820 VB=AK:FOR J=0 TO AK STEP K:FOR I=0 TO K-2:VB=VB+1:V(VB,0)=I+J:V(VB,1
)=I+J+1:NEXT:NEXT
830 AK=AK+1:VB=VB+1:TAGOFF:PRINT CHR$(23);CHR$(0);:GOTO 420
840 '
850 'OBJEKT LADEN
860 '
870 PEN 3:PRINT TAB(10)"*** OBJEKT LADEN ***":PEN 1
880 LOCATE 1,12:INPUT"Bitte Dateinamen eingeben";Y$
890 GOSUB 1120:OPENIN Y$
900 INPUT #9,AA,AK,VB
910 FOR I=0 TO AA:INPUT #9,ALPHA(I),BETA(I),GAMMA(I),XOFF(I),YOFF(I),ZOF
F(I):NEXT
920 FOR I=0 TO AK:INPUT #9,X(I),Y(I),Z(I):NEXT
930 FOR I=0 TO VB:INPUT #9,V(I,0),V(I,1):NEXT
940 CLOSEIN
950 GOTO 420
960 '
970 'OBJEKT SPEICHERN
980 '
990 PEN 3:PRINT TAB(10)"*** OBJEKT SICHERN ***":PEN 1
1000 LOCATE 1,12:INPUT"Bitte Dateinamen eingeben";Y$
1010 OPENOUT Y$
1020 WRITE #9,AA,AK,VB
1030 FOR I=0 TO AA:WRITE #9,ALPHA(I),BETA(I),GAMMA(I),XOFF(I),YOFF(I),ZO
FF(I):NEXT
1040 FOR I=0 TO AK:WRITE #9,X(I),Y(I),Z(I):NEXT
1050 FOR I=0 TO VB:WRITE #9,V(I,0),V(I,1):NEXT
1060 CLOSEOUT
1070 GOTO 420
1080 '
```



```
1090 OBJEKT LOESCHEN
1100 '
1110 GOSUB 1120:GOTO 420
1120 AA=0:AK=0:VB=0
1130 FOR I=0 TO 127
1140 ALPHA(I)=0:BETA(I)=0:GAMMA(I)=0:XOFF(I)=0:YOFF(I)=0:ZOFF(I)=0:V(I,0
)=0:V(I,1)=0:X(I)=0:Y(I)=0:Z(I)=0
1150 NEXT
1160 RETURN
1170 '
1180 'ANIMATIONS-EDITOR
1190 '
1200 INPUT"Ab Zeile";I:IF I>AA OR I<0 THEN PRINT"*** Zeile nicht vorhand
en! ***":PRINT:GOTO 1200
1210 MODE 2:PRINT"ZEILE    XROT    YROT    ZROT    XPOS    YPOS    ZPOS"
1220 PRINT"  ";USING"###";I;:PRINT":     ";USING"##";ALPHA(I);:PRINT"
  ";USING"##";BETA(I);:PRINT"     ";USING"##";GAMMA(I);:PRINT"    ";US
ING"####";XOFF(I);:PRINT"    ";USING"####";YOFF(I);:PRINT"    ";USING"##
##";ZOFF(I);
1230 INPUT"    Zeile aendern";Y$:Y$=UPPER$(Y$)
1240 IF Y$<>"J" AND I+1>AA OR Y$="Q" THEN 420 ELSE IF Y$<>"J" THEN 1260
1250 PRINT TAB(9);:INPUT ALPHA(I):PRINT CHR$(11);STRING$(16,9);:INPUT BE
TA(I):PRINT CHR$(11);STRING$(24,9);:INPUT GAMMA(I):PRINT CHR$(11);STRING
$(32,9);:INPUT XOFF(I):PRINT CHR$(11);STRING$(40,9);:INPUT YOFF(I):PRINT
 CHR$(11);STRING$(48,9);:INPUT ZOFF(I)
1260 I=I+1:IF I>AA THEN AA=I
1270 IF I>127 THEN AA=127:GOTO 420 ELSE 1220
1280 '
1290 'ECKPUNKT EDITOR
1300 '
1310 INPUT"Ab Eckpunkt";I:IF I>AK OR I<0 THEN PRINT"*** Eckpunkt nicht
vorhanden! ***":PRINT:GOTO 1310
1320 CLS:PEN 3:PRINT"ECKPUNKT   X      Y      Z":PEN 1
1330 PRINT"  ";USING"###";I;:PRINT"   ";USING"####";X(I);:PRINT"   ";USI
NG"####";Y(I);:PRINT"   ";USING"####";Z(I);
1340 INPUT"   Aendern";Y$:Y$=UPPER$(Y$)
1350 IF Y$<>"J" AND I+1>AK OR Y$="Q" THEN 420 ELSE IF Y$<>"J" THEN 1370
1360 PRINT TAB(9);:INPUT X(I):PRINT CHR$(11);STRING$(15,9);:INPUT Y(I):P
RINT CHR$(11);STRING$(22,9);:INPUT Z(I)
1370 I=I+1:IF I>AK THEN AK=I
```

```
1380 IF I>127 THEN AK=127:GOTO 420 ELSE 1330
1390 '
1400 'VERBINDUNGS-EDITOR
1410 '
1420 INPUT"Ab Verbindung";I:IF I>VB OR I<0 THEN PRINT"*** Verbindung
nicht vorhanden! ***":PRINT:GOTO 1420
1430 CLS:PEN 3:PRINT"VERBINDUNG    P1      P2":PEN 1
1440 PRINT"  ";USING"###";I;:PRINT"        ";USING"###";V(I,0);:PRINT"
   ";USING"###";V(I,1);
1450 INPUT"   Aendern";Y$:Y$=UPPER$(Y$)
1460 IF Y$<>"J" AND I+1>VB OR Y$="Q" THEN 420 ELSE IF Y$<>"J" THEN 1480
1470 PRINT TAB(13);:INPUT V(I,0):PRINT CHR$(11);STRING$(20,9);:INPUT V(I
,1)
1480 I=I+1:IF I>VB THEN VB=I
1490 IF I>127 THEN VB=127:GOTO 420 ELSE 1440
1500 '
1510 'DEMO EINLESEN
1520 '
1530 GOSUB 1120:RESTORE 2120
1540 READ AK
1550 FOR I=0 TO AK-1:READ X(I),Y(I),Z(I):NEXT
1560 '
1570 READ VB
1580 FOR I=0 TO VB-1:READ V(I,0),V(I,1):NEXT
1590 '
1600 READ AA
1610 FOR I=0 TO AA-1:READ ALPHA(I),BETA(I),GAMMA(I),XOFF(I),YOFF(I),ZOFF
(I):NEXT
1620 '
1630 GOTO 420
1640 '
1650 'ANIMATIONS-SCHLEIFE
1660 '
1670 ORIGIN 320,200
1680 FOR K=0 TO AA-1
1690 '
1700 'DISPLAY MANAGEMENT
1710 '
1720 OUT &BC00,12:OUT &BD00,52
1730 CALL &3FF4
```



```
1740 OUT &BC00,12:OUT &BD00,16
1750 CLS
1760 '
1770 P0!=CO!(GAMMA(K))*CO!(BETA(K))
1780 P1!=SI!(GAMMA(K))*CO!(BETA(K))
1790 P2!=-SI!(BETA(K))
1800 P3!=-SI!(GAMMA(K))*CO!(ALPHA(K))+CO!(GAMMA(K))*SI!(BETA(K))*SI!(ALP
HA(K))
1810 P4!=CO!(GAMMA(K))*CO!(ALPHA(K))+SI!(GAMMA(K))*SI!(BETA(K))*SI!(ALPH
A(K))
1820 P5!=CO!(BETA(K))*SI!(ALPHA(K))
1830 P6!=SI!(GAMMA(K))*SI!(ALPHA(K))+CO!(GAMMA(K))*SI!(BETA(K))*CO!(ALPH
A(K))
1840 P7!=-CO!(GAMMA(K))*SI!(ALPHA(K))+SI!(GAMMA(K))*SI!(BETA(K))*CO!(ALP
HA(K))
1850 P8!=CO!(BETA(K))*CO!(ALPHA(K))
1860 '
1870 'PROJEKTIONS-SCHLEIFE
1880 '
1890 FOR I=0 TO VB-1
1900 J=0:GOSUB 2040
1910 MOVE XP,YP
1920 J=1:GOSUB 2040
1930 DRAW XP,YP
1940 NEXT
1950 '
1960 NEXT
1970 '
1980 OUT &BC00,12:OUT &BD00,52
1990 FOR I=1 TO 10000:NEXT
2000 GOTO 420
2010 '
2020 '3-D-TRANSFORMATION
2030 '
2040 X0=X(V(I,J)):Y0=Y(V(I,J)):Z0=Z(V(I,J))
2050 ZP!=1-(X0*P2!+Y0*P5!+Z0*P8!+ZOFF(K))/FP
2060 XP=(X0*P0!+Y0*P3!+Z0*P6!+XOFF(K))/ZP!
2070 YP=(X0*P1!+Y0*P4!+Z0*P7!+YOFF(K))/ZP!
2080 RETURN
2090 '
```

```
2100 'KOORDINATEN
2110 '
2120 DATA 8
2130 DATA -50,-50,-50
2140 DATA -50,50,-50
2150 DATA 50,50,-50
2160 DATA 50,-50,-50
2170 DATA -50,-50,50
2180 DATA -50,50,50
2190 DATA 50,50,50
2200 DATA 50,-50,50
2210 '
2220 'VERBINDUNGEN
2230 '
2240 DATA 14
2250 DATA 0,1
2260 DATA 1,2
2270 DATA 2,3
2280 DATA 3,0
2290 DATA 4,5
2300 DATA 5,6
2310 DATA 6,7
2320 DATA 7,4
2330 DATA 0,4
2340 DATA 1,5
2350 DATA 2,6
2360 DATA 3,7
2370 DATA 0,2
2380 DATA 1,3
2390 '
2400 ANIMATION
2410 '
2420 DATA 25
2430 DATA 0,0,0,0,0,0
2440 DATA 1,0,0,0,0,0
2450 DATA 2,0,0,0,0,0
2460 DATA 3,0,0,0,0,0
2470 DATA 4,0,0,0,0,0
2480 DATA 5,0,0,0,0,0
2490 DATA 6,0,0,0,0,0
```



```
2500 DATA 7,0,0,0,0,0
2510 DATA 8,0,0,0,0,0
2520 DATA 9,0,0,0,0,0
2530 DATA 10,0,0,0,0,0
2540 DATA 11,0,0,0,0,0
2550 DATA 12,0,0,0,0,0
2560 DATA 13,0,0,0,0,0
2570 DATA 14,0,0,0,0,0
2580 DATA 15,0,0,0,0,0
2590 DATA 16,0,0,0,0,0
2600 DATA 17,0,0,0,0,0
2610 DATA 18,0,0,0,0,0
2620 DATA 19,0,0,0,0,0
2630 DATA 20,0,0,0,0,0
2640 DATA 21,0,0,0,0,0
2650 DATA 22,0,0,0,0,0
2660 DATA 23,0,0,0,0,0
2670 DATA 0,0,0,0,0,0
2680 END
```

Programmbeschreibung:

200-240 **Initialisierungs-Block**
Der Speicher wird geschützt, die Bogenmaß-Kalkulation ausgeschaltet und alle Vektoren und Matrizen dimensioniert. Weiterhin werden alle Variablen als Integer deklariert (DEFINT) und die Variable FP (Fluchtpunkt) auf ihren Default-Wert gesetzt.

250-340 **Block-Move-Routine in den Speicher laden**
Die in den BASIC-Zeilen 300 bis 340 enthaltenen DATA-Elemente werden in den geschützten RAM-Bereich ab &3FF4 gePOKEt.

350-380 **Sinus- und Cosinus-Tabelle generieren**
Dabei werden die SIN- und COS-Werte in 15-Grad-Intervallen ermittelt. Der Laufindex wird da-

bei mittels Division durch 15 im Bereich 0 bis 23 gehalten.

390-620 **Hauptmenü generieren**
Der CPC wird in Grafikmodus 1 gebracht und damit der Bildschirm gelöscht. Danach wird das Hauptmenü mit PRINT-Anweisungen auf den Bildschirm geschrieben. In den Zeilen 600 und 610 erfolgt ein Keybordcheck mir anschließender Prüfung auf gültige Eingabeparameter. Bei Gültigkeit verzweigt Zeile 620 in die angewählte Routine.

630-830 **Objekt generieren**
Die Zeilen 660 bis 680 sind für das Zeichnen des Koordinatensystems sowie der Initialisierung aller Default-Werte zuständig. Die Zeilen 690 bis 780 beinhalten die Animierungsschleife des Fadenkreuzes sowie die Abfrage und Auswertung der SHIFT- und RETURN-Tasten. In den folgenden Zeilen werden die eingegebenen Koordinaten um die Y-Achse gespiegelt und das Ergebnis vernetzt.

840-950 **Objekt laden**
Nach dem Löschen aller Tabellen mittels GOSUB 1120 werden die ersten drei Byte der in Y$ definierten Daten in die Variablen AA, AK und VB transferiert. AA enthält dann die Anzahl der Animationssequenzen, AK die Anzahl der Eckpunkte und VB die Anzahl der Verbindungen, die in den Zeilen 910 bis 930 den einzelnen Tabellen zugeordnet werden.

960-1070 **Objekt sichern**
Die Variablen AA, AK und VB, die als erstes in die mittels Y$ definierte Datei geschrieben werden, enthalten jeweils die Anzahl der Datenelemente, die in der Animation-, Eckpunkt- und Verbindungstabelle gespeichert sind. In den Zeilen 1030 bis 1050 werden sie aus diesem Grund auch als

Laufvariablen in den FOR-NEXT-Schleifen, die die Tabelleninhalte auf Diskette schreiben, eingesetzt.

1080-1160 **Objekt löschen**
Der GOSUB-Befehl in Zeile 1110 ruft die eigentliche Lösch-Routine auf und beendet die Funktion mit einem Rücksprung in das Hauptmenü (GOTO 420). Da die Lösch-Funktion als Unterprogramm definiert wurde, kann sie von anderen Programmodulen aus aufgerufen werden.

Die Zeilen 1120 bis 1160 befindet sich die eigentliche Lösch-Routine. In ihr werden die Variablen AA, AK und VB auf Null gesetzt sowie alle Tabellenwerte innerhalb der FOR-NEXT-Schleife in den Zeilen 1130 bis 1150 gelöscht.

1170-1270 **Animations-Editor**
Nachdem die Nummer der ersten Animations-Zeile in Zeile 1200 als Index-Variable I definiert wurde, wird Ihr Inhalt in Form der Variablen ALPHA(I), BETA(I), GAMMA(I), XOFF(I), YOFF(I) und ZOFF(I) mittels USING-Anweisungen auf dem Bildschirm ausgegeben. In Zeile 1230 bis 1240 erfolgt dann die Eingabe des EDIT-Parameters nach Y$ und die damit verbundene Meldung "Zeile Aendern?" sowie seine anschließende Überprüfung auf Gültigkeit.

- Falls sich "J" in Y$ befindet, erfolgt in Zeile 1250 eine formatierte Eingabe der Variablen ALPHA(I), BETA(I), GAMMA(I), XOFF(I), YOFF(I) und ZOFF(I) mit INPUT-Anweisungen.

- Bei Identifizierung des Parameter "Q" erfolgt ein Rücksprung in das Hauptmenü. Falls sich in Y$ nicht "J" befindet und I+1 größer als der Inhalt

der Variablen AA (Maxlines) ist, wird ebenfalls in das Hauptmenü verzweigt.

- Wenn sich ein beliebiges anderes Zeichen in Y$ befindet, wird die Zeile 1260 angesprungen.

In den Zeilen 1260 bis 1270 wird der Laufindex I inkrementiert und überprüft, ob sein Inhalt den Wert 127 überschreitet. Des weiteren wird - falls nötig - die Variable AA (Maxline) dem Laufindex I angepaßt.

1280-1380 **Eckpunkt-Editor**
Nachdem die Nummer des ersten Eckpunktes in Zeile 1310 als Index-Variable I definiert wurde, wird Ihr Inhalt in Form der Variablen X(I), Y(I) und Z(I) mittels USING-Anweisungen auf dem Bildschirm ausgegeben. In Zeile 1340 bis 1350 erfolgt dann die Eingabe des EDIT-Parameters nach Y$ und die damit verbundene Meldung "Aendern?", sowie seine anschließende Überprüfung auf Gültigkeit.

- Falls sich "J2 in Y$ befindet, erfolgt in Zeile 1360 eine formatierte Eingabe der Variablen X(I), Y(I) und Z(I) mit INPUT-Anweisungen.

- Bei Identifizierung des Parameter "Q" erfolgt ein Rücksprung in das Hauptmenü. Falls sich in Y$ nicht "J" befindet und I+1 größer als der Inhalt der Variable AK (Maxlines) ist, wird ebenfalls, in das Hauptmenü verzweigt.

- Wenn sich ein beliebiges anderes Zeichen in Y$ befindet, wird die Zeile 1370 angesprungen.

In den Zeilen 1370 bis 1380 wird der Laufindex I inkrementiert und überprüft, ob sein Inhalt den Wert 127 überschreitet. Des weiteren wird - falls

nötig - die Variable AK (Maxline) dem Laufindex I angepaßt.

1390-1490 **Verbindungs-Editor**
Nachdem die Nummer der ersten Verbindung in Zeile 1420 als Index-Variable I definiert wurde, wird Ihr Inhalt in Form der Variablen V(I,0) und V(I,1) mittels USING-Anweisungen auf dem Bildschirm ausgegeben. In Zeile 1450 bis 1460 erfolgt dann die Eingabe des EDIT-Parameters nach Y$ und die damit verbundene Meldung "Aendern?" sowie seine anschließende Überprüfung auf Gültigkeit.

- Falls sich "J" in Y$ befindet, erfolgt in Zeile 1470 eine formatierte Eingabe der Variablen X(I), Y(I) und Z(I) mit INPUT-Anweisungen.

- Bei Identifizierung des Parameter "Q" erfolgt ein Rücksprung in das Hauptmenü. Falls sich in Y$ nicht "J" befindet und I+1 größer als der Inhalt der Variable VB (Maxlines) ist, wird ebenfalls in das Hauptmenü verzweigt.

- Wenn sich ein beliebiges anderes Zeichen in Y$ befindet, wird die Zeile 1480 angesprungen.

In den Zeilen 1480 bis 1490 wird der Laufindex I inkrementiert und überprüft, ob sein Inhalt den Wert 127 überschreitet. Desweiteren wird - falls nötig - die Variable VB (Maxline) dem Laufindex I angepasst.

1500-1630 **Demo einlesen**
Nach dem Löschen aller Tabellen (GOSUB 1120) und der Positionierung des DATA-Pointers auf den Anfang der Demo-Datenblöcke, werden in den Zeilen 1540 bis 1610 die DATA-Elemente in die Eckpunkt-, Verbindungs-, und Animationstabelle geladen. Mit den READ-Befehle vor jeder Schleife

werden jeweils Anzahl der Datenelemente+1, die die in der nächsten Zeile stehende FOR-NEXT-Schleife einzulesen hat, die zugeordnete Maxline-Variable geladen (AA, AK oder VB).

1640-1680 **Animationsschleife**
Die FOR-NEXT-Schleife in Zeile 1680 ist für die Steuerung der Animationssequenzen verantwortlich. Die Laufvariable K repräsentiert in diesem Zusammengang den Animations-Zugriffschlüssel (Index). Das in Zeile 1670 stehende ORIGIN 320,200 fixiert das Koordinatensystem auf dem Bildschirm-Mittelpunkt.

1690-1750 **Display-Management**
Diese Routine ist für die zweiseitige Bildschirmverwaltung verantwortlich. Bei diesem Verfahren wird die Grafik auf einer unsichtbaren Bildschirmseite erstellt, um unnötiges Flackern des Bildes zu verhindern. Zeile 1720 schaltet auf die normale Bildschirmseite (&C000 bis &FFFF). Danach wird mit CALL &3FF4 der Inhalt des Bildschirms in BANK 1 (&4000 bis &7FFF) kopiert. Der OUT-Befehl in Zeile 1740 legt den Bildschirmspeicher nach &4000 und macht somit den Inhalt von BANK 1 sichtbar. Nun kann durch den CLS-Befehl die normale Bildschirm (&C000 bis &FFFF) gelöscht werden.

1760-1850 **Rotations-Parameterdefinition**
In diesen Zeilen werden mit Hilfe von Winkelfunktions-Tabellen die für die Rotation benötigten Parameter P0 bis P8 ermittelt.

1860-2000 **Projektions-Schleife**
Sie holt alle Linien eines Objekts aus den Tabellen und gibt die Indizes der Start- und Endkoordinaten an die 3-D-Transformationsroutine (GOSUB 2040) weiter. Die in XP und YP gewonnenen Koordinaten X1, X2, Y1 und Y2 werden in den Zeilen 1910 und 1930 für das Zeichnen der Linien verwendet.

Nach dem Beendigung der Animationssequenz bewirken die Zeilen 1980 bis 1990, daß das letzte im Bildschirmpuffer vorhandene Bild für einige Sekunden auf dem Bildschirm stehen bleibt und betrachtet werden kann.

2010-2080 **3-D-Transformation**
Diese Routine berechnet unter Einbeziehung der Rotationsparameter P0 bis P8, sowie die Fluchtpunkt-Variable FP, die X- und Y-Position eines Eckpunktes auf dem Bildschirm in Zentralperspektive.

2090-2200 **Eckpunkte für Demo**
Das erste DATA-Element gibt die Anzahl der folgenden DATA-Zeilen an.

2210-2380 **Verbindungen für Demo**
Das erste DATA-Element gibt die Anzahl der folgenden DATA-Zeilen an.

2390-2670 **Animationssequenzen für Demo**
Das erste DATA-Element gibt die Anzahl der folgenden DATA-Zeilen an.

2680 **Programm beenden**
Dieser END-Befehl darf nicht entfernt werden, da ihn die Funktion "Programm beenden" anspringt.

7.5.1 Programmerweiterungen

An dieser Stelle möchten wir an Sie einige Anregungen für 3-D-Erweiterungsmodule weitergeben. Viele dieser Ideen lassen sich leider nur in Maschinensprache realisieren. Wir hoffen trotz alledem, daß der interressierte Leser "CPCs World" oder ein eigens 3-D-Programm um einige dieser Ideen bereichern kann.

Versteckte Kanten (Hidden Lines)

Alle vorgestellten 3-D-Algorithmen projizierten auch die normalerweise nicht sichtbaren Kanten eines Objekts auf den Bildschirm. Um dies zu verhindern, muß nicht das gewählte Projektionsverfahren geändert, sondern ein sogenanntes "Hidden-Line-Modul" in das 3-D-Programm implementiert werden. Das "Hidden-Line-Modul" hat die Aufgabe, alle Koordinaten so zu manipulieren, daß unsere Projektionsroutine automatisch keine "versteckten Kanten" zeichnet.

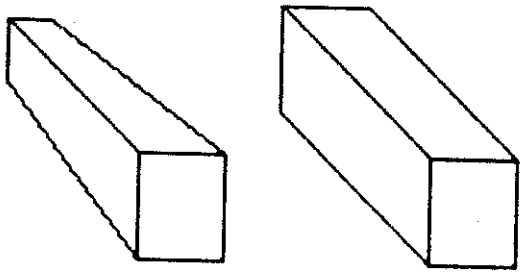


Abb. 17: **Objekt mit versteckten Kanten: rechts Parallel-, links Zentralperspektive**

Es existieren vielfältige Möglichkeiten, einen derartigen Hidden-Line-Algorithmus zu realisieren. Beispielsweise können alle im Objekt vorkommenden Flächen, mit einem Index versehen, in Bezug auf die Z-Achse sortiert und in eine Tabelle eingetragen werden. Anhand dieser Tabelle kann dann entschieden werden, welche Koordinaten an die Projektionsroutine weitergegeben werden dürfen, und welche ignoriert werden müssen.

Mehrfarbige Objekte

Es gibt mehrere Möglichkeiten, die Farbvielfalt des Schneider-CPC in 3-D-Programme einzubauen. Falls Sie mehrere Objekte simultan auf dem Bildschirm animieren, kann beispielsweise jedem Wire-Frame eine andere Farbe zugewiesen werden. Wenn sich die Animation weiterhin auf ein Objekt begrenzt, besteht die Möglichkeit den Verbindungseditor in solch einer Weise zu ändern, daß neben den Verbindungsparametern auch ein Farbparameter für jede einzelne Linie definiert werden kann. Eine Kombination beider Verfahren ist selbstverständlich ebenfalls möglich.

Die vielleicht interessanteste Farbgebung ist die Einfärbung alle im Objekt vorhandenen Flächen in verschiedene Farben. Dieses Verfahren setzt allerdings eine Fill-Funktion sowie einen Hidden-Line-Algorithmus voraus.

Licht und Schatten

Besonders faszinierend ist die Definition einer imaginären Lichtquelle im dreidimensionalen Raum. Zu diesem Zweck müssen die Flächen aller projizierten Objekte mit verschiedenen Farben ausgefüllt werden. Flächen, die unter direkter Lichteinwirkung stehen, werden mit einer hellen Farbe gefüllt. Ein Halbschatten simuliert man am besten mit einer dunklen Farbe. Vollschatten werden mit Schwarz und Glanzlichter mit leuchtend Weiß dargestellt.

Neben der komplexen Berechnung von Schatten, Halbschatten, diffusen Lichtquellen und Glanzlichern gibt es noch eine weitere und gleichzeitig einfach zu realisierende Alternative. Die Farbhelligkeit muß nur in ein richtiges Verhälnis zur Z-Achse gebracht werden. D.h., daß der Fläche mit dem größten Z-Wert die hellste, und der Fläche mit dem niedrigsten Z-Wert die dunkelste Farbe zugeordnet wird. Eine derartige Schattensimulation kann, obwohl ihr mathematischer Ursprung irreal ist, den drei-

dimensionalen Effekt Ihrer Grafik vervielfachen. Sie setzt allerdings eine Fill-Funktion sowie einen Hidden-Line-Algorithmus vorraus.

Fließende Animation

Eine fließende Animation dreidimensionaler Wire-Frames ist - wenn überhaupt - nur in Maschinensprache möglich. Das dazu notwendige Know-How kann und soll Ihnen dieses Buch nicht vermitteln. Das Programmieren in Z80-Maschinensprache ist ein Kapitel für sich und muß mit entsprechender Spezialliteratur erlernt werden. Für diejenigen, die diesen Schritt bereits geschafft haben, wollen wir die wichtigsten Regeln für eine möglichst effektive Umsetzung der in BASIC vorgestellten 3-D-Algorithmen aufführen.

1. Bedenken Sie, daß die Z80-CPU ein registerorientierter Prozessor ist. Greifen Sie aus diesem Grund, wenn möglich, immer auf ein Register und nicht auf Speicherzellen zurück.

2. Erstellen Sie zeitintensive Routinen immer anhand eines Z80-Prozessorhandbuchs. Darin finden Sie die Ausführungszeit jedes einzelnen Z80-Befehls. Versuchen Sie, die Gesamtausführungszeit Ihrer Routine so gering wie möglich zu halten, indem Sie die Taktzyklen aller verwendeten Befehle addieren und überprüfen, ob Befehlsketten durch andere Befehle mit geringerer Zyklusrate ersetzt werden können. Geitzen Sie in diesem Zusammenhang mit jeder Mikrosekunde, da sich diese insbesondere in Programmschleifen schnell addieren!

3. Schreiben Sie einen eigenen Line-Algorithmus. Der DRAW-Befehl des CPC ist zwar schnell, aber es geht noch viel schneller. Verzichten Sie im Idealfall auf alle Betriebssystemfunktionen (auch auf die Division und Multiplikation).

4. Realisieren Sie jede Form der Bildschirmausgabe immer mit ganzen Bytes (Bit-Masken).

5. Arbeiten Sie mit soviel Tabellen wie möglich. Bedenken Sie, daß ein Tabellenelement im Idealfall ein Byte lang sein sollte. Auch die Winkelfunktionstabellen sollten aus ein-Byte-Daten bestehen.

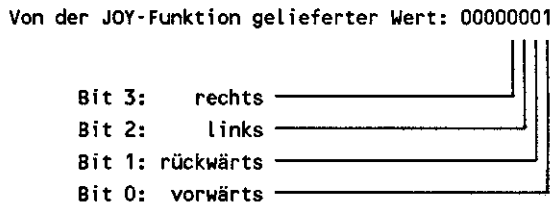
Und zum Schluß noch ein Tip für denjenigen, dem die Programmumsetzung in Maschinensprache zu aufwendig ist. Eine fließende Objektanimtion kann auch im eingeschränktem Maße mit einem BASIC-Programm und zwei kleinen Maschinenroutinen erzeugt werden. Wie? Ganz einfach: Nach Berechnung jeder Sequenz wird eine kleine Maschinenroutine angesprungen, welche die Daten des 16K-Bildschirms auf ein Minimum komprimiert und das Ergebnis auf Diskette wegschreibt. Ist auf diese Weise eine vollständige Sequenz gespeichert, braucht lediglich ein weiteres Maschinenprogramm alle abgespeicherten und kompremierten Bilder in den Speicher zu laden und sie nacheinander unter Expansion in den Bildschirmspeicher zu transferieren.

8. Darf's auch etwas mehr sein - Peripheriegeräte

Die Erfahrung zeigt, daß Computer selten allein bleiben, sondern im Regelfall einen ganzen Park von Gerätschaften nach sich ziehen. Angefangen bei Speichererweiterungen und zusätzlichen Schnittstellen, die direkt in das Gerät eingebaut werden, setzt sich die Expansion bei Peripheriegeräten fort. Um Peripheriegeräte, die im Zusammenhang mit Grafik einen Nutzen bringen - wenn man das Wissen um ihre Einsetzbarkeit hat - soll es in diesem Kapitel gehen.

8.1 Joystickabfrage

Angeschlossene Joysticks werden vom CPC wie ein Teil der Tastatur behandelt und können deshalb mit denselben BASIC-Funktionen wie die Tastatur abgefragt werden. Neben dieser Abfragemöglichkeit kann der Zustand der beiden Joysticks aber auch mit JOY(0) und JOY(1) ermittelt werden. Diese Funktionen liefern ein sogenanntes bitsignifikantes Ergebnis, das die Bewegungsrichtung der Joysticks zum Zeitpunkt der letzten Abfrage wiederspiegelt. Bitsignifikant heißt nichts anderes, als daß den einzelnen Bits in dem von der JOY-Funktion zurückgelieferten Wert eine Bedeutung zukommt. Am Beispiel eines nach vorne gedrückten Joysticks soll die folgende Tabelle den Zusammenhang beleuchten:



Der Weg von der Joystickabfrage zu einer Anwendung des Joysticks im Bereich der grafikbezogenen Programmierung ist nicht weit. So läßt sich mit dem Joystick eine Markierung - wir wollen sie *Joystickzeiger* nennen - über den Bildschirm bewegen. Der Joystickzeiger kann bei der Programmierung eines Mal- oder Zeichenprogramms die Stelle kennzeichnen, an der der Pinsel oder Zeichenstift sich befindet.

Im folgenden ist ein Programm abgedruckt, das das Prinzip der Steuerung eines Joystickzeigers veranschaulicht. Es setzt die Bewegungen des Joysticks in pixelgenaue Positionsänderungen des Joystickzeigers um. Der Joystickzeiger wird durch ein "X" dargestellt, das sich innerhalb des Bildschirmfenster beliebig bewegen läßt. Bewegungen, die den Joystickzeiger aus dem Bildschirm verschwinden lassen würden, werden vom Programm erkannt und unterbunden.

```
100 DEFINT A-Z
110 MODE 1
120 TAG
130 X=320
140 Y=200
150 XALT=X
160 YALT=Y
170 MOVE X,Y
180 PRINT "X";
190 '
200 J=JOY(0)
210 IF J=0 THEN 200
220 '
230 IF (J AND 1)=1 THEN Y=Y+1
240 IF (J AND 2)=2 THEN Y=Y-1
250 IF (J AND 4)=4 THEN X=X-1
260 IF (J AND 8)=8 THEN X=X+1
270 '
280 IF X<0 THEN X=0
290 IF X>623 THEN X=623
300 IF Y<15 THEN Y=15
310 IF Y>399 THEN Y=399
```



```
320 '
330 MOVE XALT,YALT
340 PRINT " ";
350 '
360 MOVE X,Y
370 PRINT "X";
380 '
390 XALT=X
400 YALT=Y
410 '
420 GOTO 200
```

Programmbeschreibung:

100-180 Im Initialisierungsteil des Programms werden alle auftretenden Variablen als Integervariablen definiert, MODE 1 gesetzt und die Position für die Textausgabe aus die Koordinaten des Grafikcursors gebunden. Die Variablen X und Y, die die Grafikkoordinaten in sich bergen, an denen sich der Joystickzeiger befindet, werden auf ihren Defaultwert (Bildschirmmitte) gebracht. Ihre Werte werden den Variablen XALT und YALT zugewiesen, in denen die alte Position des Joystickzeigers gepuffert wird. Am Ende dieses Programmteils wird der Grafikcursor mit den Parametern X und Y gesetzt und der Joystickzeiger in Gestalt eines "X" auf den Bildschirm geschieben.

190-210 Mit der BASIC-Funktion JOY(0) wird der Status des Joysticks ermittelt und der Variablen J zugewiesen. Hat J den Wert 0, i.e. der Joystick verharrt in Ruhestellung, dann erfolgt der Rücksprung zur erneuten Joystickabfrage.

220-260 Die Bits 0 bis 3 der Variablen J werden auf gesetzt bzw. nicht gesetzt überprüft und gemäß der durch sie wiedergespiegelten Joystickbewegung die Koordinaten für den Joystickzeiger aktualisiert.

270-310 In diesen Programmzeilen wird überprüft, ob die entstandene Position außerhalb des Bildschirms liegen würde, und berichtigt, sollte dies zutreffen.

320-Ende Die verbleibenden Programmzeilen löschen den Joystickzeiger an den alten Bildschirmposition (XALT, YALT) und setzen ihn mit den aktuellen Koordinaten (X, Y). Ist der Joystickzeiger auf den Bildschirm gebracht, so werden die aktuellen Koordinaten zu alten und das Programm geht in eine Schleife.

8.2 Der Lightpen

Ein Peripheriegerät, dem immer noch ein etwas exklusiver Touch anhaftet, und von dem man sagt, daß es professionellen Systemen vorbehalten ist, ist der Lightpen. Dieses Gerät ist dem äußeren Anschein nach nicht viel mehr als ein Stift aus dem ein Kabel ragt, dessen Ende mit dem Computer verbunden ist. Betrachtet man die Spitze des Stifts genauer, so entdeckt man dort einen kleinen optischen Sensor, der das entscheidende Teil im Gefüge des Lightpens ist.

Die Funktionsweise des Lightpens ist einfach. Wird er mit der Spitze, in der sich der optische Sensor befindet, auf den Bildschirm des Computers gehalten, so erzeugt er jedesmal, wenn der Kathodenstrahl der Bildröhre seine Position bestreicht, einen Impuls. Dieser Impuls kann vom Computer mit einem geeigneten Programm ausgewertet und so auf die Position des Lightpens auf dem Bildschirm geschlossen werden. Aus dieser Funktionsweise ergeben sich eine Reihe von Anwendungsmöglichkeiten für den Lightpen. Sie reichen von einfacher Menüauswahl bis hin zum Zeichnen auf dem Bildschirm.

Daß der Lightpen nicht nur professionellen Systemen vorbehalten ist, sondern sich auch hervorragend für den CPC einsetzen läßt, soll dieses Kapitel zeigen. Am Anfang unserer Exkursion soll der Selbstbau eines einfachen Lightpens stehen, der sich leider nur für den Betrieb mit einem Farbmonitor eignet, und an deren Ende ein Programm, das den Einsatz des Lightpens am Beispiel einer Menüauswahl verdeutlichen soll.

Zum Aufbau des Lightpens sind nur ein paar Bauteile nötig, die in jedem Elektronikladen erhältlich sein dürften. Die Schaltung kann mit geringem Aufwand auch von Ungeübten auf einem Stück Lochrasterplatte, die ebenfalls im Elektronikladen zu bekommen ist, aufgebaut werden. Nicht mit auf die Platine darf der Fototransistor BPX 81 gelötet werden; er soll vielmehr durch ein Stück zweiadrige Litze mit der übrigen Schaltung verbunden werden.

Der Fototransistor, dessen lichtempfindliche Seite beim Betrieb zum Bildschirm weisen muß, sollte, da er klein und leicht zu zerstören ist, in ein Gehäuse eingebracht werden. Hervorragend geeignet für der BPX 81 ist ein Kunststoffkugelschreiber, der zuvor von allen Innereien befreit wurde. Das Loch an der Spitze des Stifts muß mit einer kleinen Feile oder einem spitzen Messer so ausgearbeitet werden, daß das Gehäuse des Fototransistors mit zuvor angelöteten Anschlüssen darin eingeklebt werden kann. Vorsicht beim Einkleben, daß kein Klebstoff auf die lichtempfindliche Stelle des Bauteils tropft, was seine Funktionstüchtigkeit erheblich beeinträchtigen würde.

Ist die Schaltung aufgebaut, der Fototransistor eingeklebt, so verbleibt im Grunde nur der Anschluß des Ganzen an den Schneider-CPC. Dazu sollte unbedingt eine geeignete Steckverbindung verwendet werden, auch wenn sie der größte Kostenfaktor beim Lightpen ist. Lassen Sie sich beim Anlöten der Kabel zwischen Platine und Steckverbindung nicht von den Beschriftungen der Pins leiten, die auf einigen Steckverbindern aufgedruckt sind. Der Schneider weißt eine Benennung der Pins auf, die vom üblichen Standard abweicht.

Drei Anschlüsse der 50-poligen-Steckverbindung sind für den Lightpen relevant: da ist Pin 27 mit der positiven Betriebsspannung (5 Volt), Pin 49 Masse und Pin 47, der die Bezeichnung Lightpen trägt. Mit der Erwähnung dieses Anschlusses sind wir zur Beschreibung der Schaltung des Lightpens vorgedrungen.

Beim Betrieb der Schaltung wird der Fototransistor auf den Bildschirm des CPC gehalten. Jedesmal wenn der Kathodenstrahl beim Bildaufbau die Stelle erreicht, an die der Lightpen gehalten wird, registiert der Fototransistor den Lichtimpuls des vorbeistreichenden Kathodenstrahls. Da der vom Fototransistor ausgehende Impuls für eine weitere Auswertung zu schwach ist, wird er durch den Darlington-Transistor BC 517 verstärkt. Mit dem verstellbaren 10-K-Widerstand kann die Empfindlichkeit des Lightpens eingestellt werden. Der so aufbereitete Impuls gelangt schließlich an den TTL-Schwellwertschalter, der dem Impuls einen TTL-Pegel verleiht. Um aus dem Impuls einen Impuls mit Low-High-Flanke zu gewinnen, muß der Impuls mit dem zweiten Gatter invertiert werden. Soweit der Weg vom durch den Kathodenstrahl erregten Impuls durch die Elektronik des Lightpens zu Pin 47 (Lightpen) des CPC.

Tritt an Pin 47 eine Low-High-Flanke auf, dann wird der aktuelle Zustand der Memory-Adress-Lines in das Lightpen-Register des Video-Controllers übertragen und gespeichert. Über die Memory-Adress-Lines werden die Speicherplätze im Bildschirmspeicher adressiert und so ist es umgekehrt möglich über ihren Zustand auf die Position des Lightpens auf dem Bildschirm zum Zeitpunkt des Impulses zu schließen. Der im Lightpen-Register gespeicherte Zustand der Memory-Adress-Lines kann von einem Programm ausgelesen und verwertet werden.

Sollte Ihnen die Beschreibung der Vorgänge, die mit dem Betrieb des Lightpens verbunden sind etwas unwohl sein, so können wir Sie beruhigen: Auch ohne ihre genaue Kenntnis können Sie sich an den Bau des Lightpens wagen.

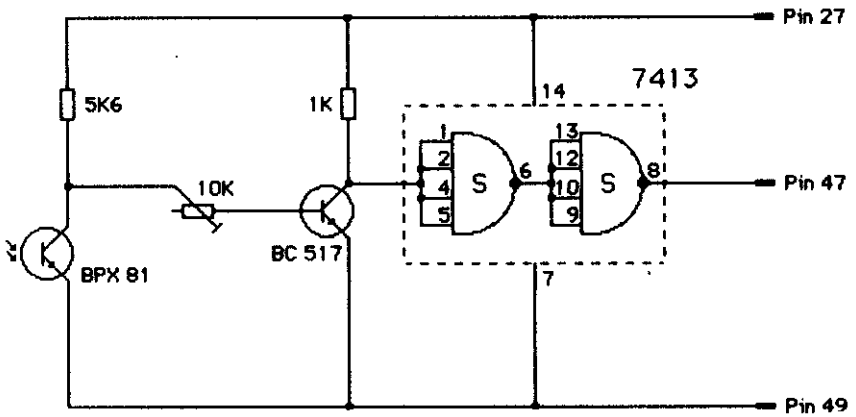


Abb. 18: Der Schaltplan des Lightpen

Bauteilliste:

TTL-IC 7413
14-pol.-IC-Sockel

Transistor BC 517

Fototransistor BPX 81

10K-Trimmer

1K-Festwiderstand
5K6-Festwiderstand

Steckverbindung für Expansionport

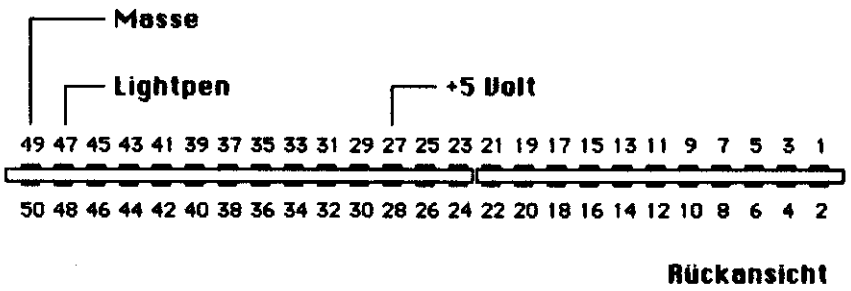


Abb. 19: Der Expansionport des CPC

Nachdem Sie nun den Lightpen - hoffentlich mit Erfolg - aufgebaut haben, wollen wir Ihnen ein kleines Programm vorstellen, mit dem Sie zum einen den Lightpen testen können und mit dessen Hilfe Sie zum anderen erfahren, wie er von einem Programm abgefragt werden kann. Das Programm ist wie schon erwähnt eine Menüauswahl. Wir haben nur das grobe Gerüst einer Menüauswahl realisiert, das Sie selbst zu einem vollständigen Benutzerprogramm mit Lightpen gesteuerter Auswahl der Optionen ausbauen können.

Das Programm bringt fünf zur Auswahl stehende Menüpunkte auf den Bildschirm und beginnt dann unmittelbar mit der Abfrage des Lightpen. Tippt man mit dem Lightpen auf einen der Menüpunkte - der Farbbildschirm sollte auf maximale Helligkeit und der Trimmer auf der Lightpenplatine auf maximale Empfindlichkeit stehen - und betätigt gleichzeitig eine beliebige Taste auf der Tastatur des CPC, dann verzweigt das Programm in das durch die Menüauswahl bestimmte Unterprogramm. Dort führt es eine Warteschleife aus, die für die in diesem Unterprogramm ablaufenden Vorgänge steht, und kehrt danach wieder zu Abfrage des Lightpens zurück.


```
100 MODE 1
110 INK 0,26
120 INK 1,0
130 BORDER 26
140 '
150 FOR I=1 TO 5
160 LOCATE 15,I*4+1
170 PRINT "MENUEPUNKT";I
180 NEXT
190 '
200 GOSUB 270
210 '
220 V=INT((P+82)/164)
230 IF INKEY$<>"" THEN CLS:ON V GOSUB 360,390,420,450,480 ELSE GOT
O 200
240 CLS
250 '
260 GOTO 150
270 '
280 'LIGHTPEN ABFRAGEN
290 '
300 OUT &BC00,16: PH=INP(&BF00)
310 OUT &BC00,17: PL=INP(&BF00)
320 '
330 P=(256*PH+PL) AND &3FF
340 '
350 RETURN
360 PRINT "UNTERPROGRAMM 1"
370 FOR I=1 TO 1000:NEXT
380 RETURN
390 PRINT "UNTERPROGRAMM 2"
400 FOR I=1 TO 1000:NEXT
410 RETURN
420 PRINT "UNTERPROGRAMM 3"
430 FOR I=1 TO 1000:NEXT
440 RETURN
450 PRINT "UNTERPROGRAMM 4"
460 FOR I=1 TO 1000:NEXT
470 RETURN
```

```
480 PRINT "UNTERPROGRAMM 5"
490 FOR I=1 TO 1000:NEXT
500 RETURN
```

Der entscheidender Bestandteil der Menüauswahl mit dem Lightpen sind die Programmzeilen 270-330, hier findet die Abfrage des Lightpens statt. Wenn Sie das Programm nur nutzen wollen, dann können Sie die vorgestellte Methode zur Abfrage des Lightpen in der vorgestellten Weise übernehmen, wenn Sie aber genau wissen wollen, wozu die vom Programm ausgelesenen Register gut sind, so sollten Sie in Kapitel 9.2 nachschlagen.

8.3 Grafik-Hardcopy

Besonders oft kommt es im Zuge der Grafikprogrammierung vor, daß man etwas auf dem Bildschirm erstellt hat und gern schwarz auf weiß auf den Drucker ausgeben möchte. Gemeint ist "Eine Hardcopy vom Bildschirm machen", wie dieser Vorgang in Fachkreisen genannt wird. Dabei wird mit einem geeigneten Programm der Bildschirminhalt gelesen und auf dem Drucker ausgegeben.

Grundsätzlich unterscheidet man zwei Arten von Hardcopyroutinen: Da ist als erste die sogenannte Text-Hardcopy, die ihren Name daher trägt, weil mit ihr nur die Bildschirminhalt auf dem Drucker ausgegeben werden können, die aus Textzeichen bestehen. Diese Art der Hardcopyroutinen hat durch die begrenzte Anzahl von Zeichen den Vorteil, schneller zu sein, als die zweite, die Grafik-Hardcopy genannt wird. Eine Grafik-Hardcopyroutine zeichnet sich aber dadurch aus, daß mit ihr der Bildschirminhalt, egal ob Text oder Grafik, auf dem Drucker ausgegeben werden - zumindest gilt dies uneingeschränkt für die Schneider-CPCs, da sie bekanntlich auch Text als Grafik darstellen.

Trotz des komfortablen Betriebsystems hat der Schneider keine Hardcopyroutine mit auf den Weg bekommen, so daß es an uns ist, Sie mit einem solchen Utility zu versorgen. Natürlich haben

wir eine Routine[1] gewählt, die mit den Merkmalen einer Grafik-Hardcopyroutine behaftet ist, um Ihnen die Möglichkeit zu bieten, sowohl Text als auch Grafiken auszudrucken.

Weil die Grafik-Hardcopyroutine ganz massiv auf die Fähigkeiten des Druckers eingehen muß und bekanntermaßen nicht alle Drucker gleich sind, mußte unter den sich anbietenden Druckern eine Auswahl getroffen werden. Die Entscheidung fiel zunächst auf den Drucker NLQ 401. Da dieser Drucker aber erstaunlich kompatibel zu den Druckern von EPSON ist, arbeitet das Programm auch mit den Drucker EPSON MX/RX/FX sowie allen Kompatiblen. Sollten Sie keinen dieser Drucker besitzen, so muß das nicht heißen, daß die Routine auf Ihrem Drucker nicht läuft, sondern nur, daß wir es nicht ausprobiert haben - das müssen wir dann schon Ihnen überlassen.

Betrachtet man das was eine Hardcopyroutine leisten muß, so ist unmittelbar einsichtig, daß sie nur in nensprache geschrieben werden kann. Damit die BASIC-Programmierer unter Ihnen aber nicht leer ausgehen, wollen wir neben dem Assemblercode auch mit einem BASIC-Lader aufwarten, der die Hardcopyroutine in den RAM-Speicher des CPCs einschreibt. Dort kann sie dann auch von BASIC aus aufgerufen werden.

Nach dem Start des abgedruckten BASIC-Laders wird die in hexadezimaler Form in den DATA-Statements enthaltene Hardcopyroutine in den RAM-Speicher des CPC gebracht. Dort steht sie dann ab Adresse &A000 und sollte gegebenenfalls gegen Überschreiben durch BASIC geschützt werden. Will man vom aktuellen Bildschirminhalt eine Hardcopy anfertigen, so genügt ein Aufruf der Routine mit CALL &A000.

```
100 'GRAFIK-HARDCOPY
110 '
120 FOR A=&A000 TO &A0BF
130 READ D
140 POKE A,D
```

1 Quelle: CPC 664/6128 Intern, DATA BECKER, 1985, Seite 158

```
150 Z=Z+D
160 NEXT A
170 '
180 DATA &CD,&BA,&BB,&CD,&E7,&BB,&32,&BD
190 DATA &A0,&CD,&6C,&A0,&21,&8F,&01,&22
200 DATA &BE,&A0,&11,&00,&00,&3E,&07,&32
210 DATA &C0,&A0,&CD,&7C,&A0,&0E,&00,&3A
220 DATA &C0,&A0,&47,&E5,&D5,&C5,&CD,&F0
230 DATA &BB,&C1,&D1,&21,&BD,&A0,&BE,&E1
240 DATA &37,&20,&01,&A7,&CB,&11,&2B,&2B
250 DATA &10,&E9,&CD,&AF,&A0,&79,&CD,&A6
260 DATA &A0,&13,&E5,&21,&7F,&02,&37,&ED
270 DATA &52,&E1,&38,&05,&2A,&BE,&A0,&18
280 DATA &CC,&23,&7C,&B5,&C8,&2B,&11,&00
290 DATA &00,&22,&BE,&A0,&3E,&07,&BD,&20
300 DATA &B9,&7C,&B4,&20,&B5,&3E,&04,&32
310 DATA &C0,&A0,&18,&AE,&3E,&1B,&CD,&A6
320 DATA &A0,&3E,&31,&CD,&A6,&A0,&00,&00
330 DATA &00,&00,&00,&C9,&E5,&3E,&42,&CD
340 DATA &1E,&BB,&E1,&28,&02,&E1,&C9,&3E
350 DATA &0D,&CD,&A6,&A0,&3E,&0A,&CD,&A6
360 DATA &A0,&3E,&1B,&CD,&A6,&A0,&3E,&4C
370 DATA &CD,&A6,&A0,&3E,&7F,&CD,&A6,&A0
380 DATA &3E,&02,&CD,&A6,&A0,&C9,&CD,&2E
390 DATA &BD,&38,&FB,&CD,&2B,&BD,&C9,&3A
400 DATA &C0,&A0,&FE,&07,&C8,&AF,&CB,&11
410 DATA &CB,&11,&CB,&11,&C9,&00,&00,&00
420 '
430 IF Z<>23151 THEN PRINT "ERROR IN DATAS"
```

Wenn Sie über Kenntnisse in Maschinensprache verfügen, wird Ihnen das Assemblerlisting die Arbeitsweise der Routine verdeutlichen. Mit seiner Hilfe wird es Ihnen auch möglich sein, die Routine an Drucker anzupassen, mit denen sie in der vorgestellten Form nicht ausdruckt. Sie sollten bei Ihrer Arbeit unbedingt der Tatsache Rechnung tragen, daß der CPC nur einen 7-Bit-Druckeranschluß besitzt, woraus sich einige Komplikationen für die Hardcopyroutine ergeben.

So ist es bedingt durch den 7-Bit-Anschluß nur möglich, sieben Punkte, die untereinander liegen, gleichzeitig zu drucken. Die Grafik des CPC setzt sich aber aus 200 Punkten in vertikaler Richtung zusammen, die nicht ohne Rest durch sieben teilbar sind. Die verbleibenden vier Punktreihen müssen daher von der Hardcopyroutine gesondert bearbeitet werden.

Ein weiteres Problem, das der 7-Bit-Anschluß mit sich bringt, liegt in der Befehlsübermittlung an den Drucker. Das Einschalten des Grafikmodus mit ESC L würde für die 640 Punkte einer Bildschirmzeile, die von der Hardcopyroutine erfaßt werden müssen, die Steuersequenz

```
PRINT #8,CHR$(27);"L";CHR$(128);CHR$(2)
```

erfordern. Die in der Sequenz enthaltene Zahl 128 läßt sich nicht in 7 Bits darstellen und ist so auch nicht an den Drucker übertragbar. Um dieses Problem zu lösen, haben wir speziell für die Hardcopyroutine den Bildschirm künstlich um eine senkrechte Punktreihe verkürzt. Die letzten Punkte werden beim Ausdruck einfach ignoriert wird - so ist aus der 128, zum Einschalten des Grafikmodus eine 127 geworden, die über den 7-Bit-Anschluß an den Drucker übertragen werden kann.

```
A000  CD BA BB     CALL  &BBBA          ;GRAFIKMODUS EINSCHALTEN
A003  CD E7 BB     CALL  &BBE7          ;HINTERGRUNDFARBE BESTIMMEN
A006  32 BD A0     LD    (&A0BD),A
A009  CD 6C A0     CALL  &A06C          ;DRUCKER AUF 7/72 ZOLL
A00C  21 8F 01     LD    HL,&018F       ;WIR FANGEN OBEN UND
A00F  22 BE A0     LD    (&A0BE),HL
A012  11 00 00     LD    DE,&0000       ;LINKS AN ZU DRUCKEN
A015  3E 07        LD    A,&07          ;UND ZWAR MIT 7 NADELN
A017  32 C0 A0     LD    (&A0C0),A
A01A  CD 7C A0     CALL  &A07C          ;ESC-SEQUENZ FÜR GRAFIK
A01D  0E 00        LD    C,&00          ;C ENTHÄLT BITMUSTER
A01F  3A C0 A0     LD    A,(&A0C0)
A022  47           LD    B,A            ;B=DOTREIHENZÄHLER
```

```
A023 E5          PUSH HL
A024 D5          PUSH DE
A025 C5          PUSH BC
A026 CD F0 BB    CALL &BBF0          ;FARBE DES PIXELS BEI (HL,DE)
A029 C1          POP  BC
A02A D1          POP  DE
A02B 21 BD A0    LD   HL,&A0BD
A02E BE          CP   (HL)           ;PIXELFARBE=HINTERGRUNDFARBE?
A02F E1          POP  HL
A030 37          SCF                 ;WENN PIXEL<>PAPER, DANN
A031 20 01       JR   NZ,$ 3 >&A034  ;CARRY SETZEN SONST
A033 A7          AND  A              ;CARRY LÖSCHEN
A034 CB 11       RL   C              ;CARRY INS UNTERSTE BIT
A036 2B          DEC  HL             ;DES C-REGISTERS SCHIEBEN,
A037 2B          DEC  HL             ;HL=HL-2, NÄCHSTER PUNKT
A038 10 E9       DJNZ $-21 >&A023    ;ALLES 7 MAL
A03A CD AF A0    CALL &A0AF          ;SONDERBEHANDLUNG DER LETZTEN
A03D 79          LD   A,C            ;BITMUSTER, IN AKKU ÜBERTRAGEN
A03E CD A6 A0    CALL &A0A6          ;UND DRUCKEN
A041 13          INC  DE
A042 E5          PUSH HL
A043 21 7F 02    LD   HL,&027F       ;EINE ZEILE GEDRUCKT?
A046 37          SCF
A047 ED 52       SBC  HL,DE
A049 E1          POP  HL
A04A 38 05       JR   C,$ 7 >&A051
A04C 2A BE A0    LD   HL,(&A0BE)
A04F 18 CC       JR   $-50 >&A01D
A051 23          INC  HL             ;SONDERBEHANDLUNG DER
A052 7C          LD   A,H            ;LETZTEN 4
A053 B5          OR   L
A054 C8          RET  Z
A055 2B          DEC  HL
A056 11 00 00    LD   DE,&0000       ;VORBEREITEN DER NÄCHSTEN
A059 22 BE A0    LD   (&A0BE),HL     ;DRUCKZEILE
A05C 3E 07       LD   A,&07
A05E BD          CP   L              ;LETZTE 7ER REIHE?
A05F 20 B9       JR   NZ,$-69 >&A01A
A061 7C          LD   A,H
A062 B4          OR   H
```



```
A063 20 B5       JR   NZ,$-73 >&A01A
A065 3E 04       LD   A,&04          ;DANN NUR NOCH 4 REIHEN
A067 32 C0 A0    LD   (&A0C0),A
A06A 18 AE       JR   $-80 >&A01A
A06C 3E 1B       LD   A,&1B          ;FÜR NLQ/MX/RX/FX
A06E CD A6 A0    CALL &A0A6          ;ESC A 7, UM DEN RICHTIGEN
A071 3E 31       LD   A,&31          ;ZEILENVORSCHUB ZU BEKOMMEN
A073 CD A6 A0    CALL &A0A6
A076 00          NOP
A077 00          NOP
A078 00          NOP
A079 00          NOP
A07A 00          NOP
A07B C9          RET
A07C E5          PUSH HL             ;DEL-TASTE GEDRÜCKT?
A07D 3E 42       LD   A,&42          ;WENN JA, DANN ABBRUCH
A07F CD 1E BB    CALL &BB1E
A082 E1          POP  HL
A083 28 02       JR   Z,$ 4 >&A087   ;DEL WAR NICHT GEDRÜCKT
A085 E1          POP  HL             ;STACK MANIPULIEREN
A086 C9          RET                 ;UM AN RET ZU GELANGEN
A087 3E 0D       LD   A,&0D          ;CR/LF AUSGEBEN
A089 CD A6 A0    CALL &A0A6
A08C 3E 0A       LD   A,&0A
A08E CD A6 A0    CALL &A0A6
A091 3E 1B       LD   A,&1B          ;ESC L 127 2=GRAFIK
A093 CD A6 A0    CALL &A0A6          ;MIT 639 PUNKTEN
A096 3E 4C       LD   A,&4C
A098 CD A6 A0    CALL &A0A6
A09B 3E 7F       LD   A,&7F
A09D CD A6 A0    CALL &A0A6
A0A0 3E 02       LD   A,&02
A0A2 CD A6 A0    CALL &A0A6
A0A5 C9          RET
A0A6 CD 2E BD    CALL &BD2E          ;DRUCKER BUSY?
A0A9 38 FB       JR   C,$-3 >&A0A6
A0AB CD 2B BD    CALL &BD2B          ;ZEICHEN DRUCKEN
A0AE C9          RET
A0AF 3A C0 A0    LD   A,(&A0C0)      ;BEHANDLUNG DER LETZTEN
A0B2 FE 07       CP   &07            ;VIER DOTREIHEN
```

```
A0B4  C8          RET   Z
A0B5  AF          XOR   A
A0B6  CB 11       RL    C              ;DREI MAL 0 ÜBER CARRY
A0B8  CB 11       RL    C              ;IN C-REGISTER SCHIEBEN
A0BA  CB 11       RL    C
A0BC  C9          RET
A0BD  00          DEFB  00
A0BE  00 00       DEFW  0000
A0C0  00          DEFB  00
```

8.4 Spruchband

Wenn Sie Ihren Worten zu gegebenen Anlaß einmal besonderen Nachdruck verleihen wollen, so können Sie das mit diesem Programm wirkungsvoll machen. Es erzeugt auf dem Drucker ein Spruchband mit übergroßen Lettern, dessen Text Sie frei bestimmen können. Die einzelnen Buchstaben sind im Extremfall 80 normale Druckzeichen hoch und werden quer zur üblichen Schreibrichtung des Druckers ausgegeben. Da das Programm je nach Länge des eingegebenen Textes ganz beachtliche Papiermengen verschlingen kann, ist es unbedingt ratsam, Endlospapier zu verwenden.

Die Bedienung des Programms ist denkbar einfach: Nach dem Start mit RUN wird der Benutzer aufgefordert, den Text einzugeben. Die eingetippten Buchstaben werden ist der ersten Zeile des Bildschirms mitgeschrieben. Hat man sich bei der Eingabe einmal vertan oder möchte den bereits eingegebenen Text verändern, so kann mit der DEL-Taste der Text Zeichen für Zeichen gelöscht werden.

Ist der Text vollständig eingegeben, so wird durch Betätigen der RETURN-Taste die Druckausgabe eingeleitet. Belegt der Text die 80 Zeichen der ersten Zeile vollständig, so beginnt das Programm bei Überschreiten der letzten Zeichenposition automatisch mit der Druckausgabe.

Während der Druckausgabe tastet das Programm die gesetzten Bildpunkte in der ersten Bildschirmzeile ab und bringt sie in

vergrößerter Form auf den Drucker. Jeder gesetzte Bildpunkt wird zu einem Feld von 10 mal 5 Doppelkreuzen. Durch dieses Abtastverfahren kann jeder Inhalt der ersten Bildschirmzeile ausgedruckt werden; Sie sind also bei diesem Programm nicht an die Zeichen des im CPC integrierten Zeichensatzes gebunden, sondern können Ihren eigenen Zeichensatz verwenden. Mit einem modifizierten Zeichensatz und kleinen Grafikelementen, die sich in der Größe eines Charakters in den Text einbringen lassen, können mit dem Programm "Spruchband" recht reizvolle Ergebnisse erzielt werden.

```
100 '*************************************************************
110 '***                                                       ***
120 '***              SPRUCHBAND  JST 5.8.1986                 ***
130 '***                                                       ***
140 '*************************************************************
150 '
160 MODE 2
170 '
180 CLS
190 '
200 DIM MATRIX(639,7)
210 '
220 START=1
230 PUNKT$=STRING$(10,"#")
240 BLANK$=SPACE$(10)
250 '
260 'TEXT EINGEBEN
270 '
280 LOCATE 1,10
290 PRINT "GEBEN SIE DEN TEXT EIN:"
300 '
310 IF START<=1 THEN START=1
320 '
330 FOR POSITION=START TO 80
340 '
350 LOCATE POSITION,1
360 PRINT " ";
370 '
```

```
380 ZEICHEN$=INKEY$
390 IF ZEICHEN$="" THEN 380
400 IF ASC(ZEICHEN$)=127 THEN START=POSITION-1:GOTO 310
410 IF ASC(ZEICHEN$)=13 THEN GOTO 480
420 '
430 LOCATE POSITION,1
440 PRINT ZEICHEN$;
450 '
460 NEXT POSITION
470 '
480 'SPRUCHBAND ERZEUGEN
490 '
500 ENDE=((POSITION-1)*8)-8
510 IF ENDE<0 THEN END
520 '
530 FOR INDEX=0 TO ENDE STEP 8
540 '
550 'ZEICHEN VOM BILDSCHIRM LESEN
560 '
570 FOR SPALTE=INDEX TO INDEX+7
580 FOR ZEILE=0 TO 7
590 '
600 MATRIX(SPALTE,ZEILE)=TEST(SPALTE,399-ZEILE*2)
610 '
620 NEXT ZEILE
630 NEXT SPALTE
640 '
650 'DRUCKERAUSGABE
660 '
670 FOR ZEILE=INDEX TO INDEX+7
680 FOR MULTI=1 TO 5
690 FOR SPALTE=7 TO 0 STEP -1
700 '
710 IF MATRIX(ZEILE,SPALTE)=1 THEN PRINT #8,PUNKT$; ELSE PRINT #8,
BLANK$;
720 '
730 NEXT SPALTE
740 '
750 PRINT #8
760 '
```



```
770 NEXT MULTI
780 NEXT ZEILE
790 '
800 NEXT INDEX
```

Programmbeschreibung:

100-240 Definitionen, Dimensionierungen und Vorbelegungen. Zur Bildschirmausgabe wird MODE 2 mit 80-Zeichen-Darstellung gewählt. Die doppeltindizierte Variable MATRIX wird auf 640 mal 8 Felder dimensioniert; sie soll im Verlauf des Programms die vom Bildschirm gelesenen Zeichenmatrizen aufnehmen.

Die Hilfsvariable START wird mit dem Wert 1 vorbelegt, der einen Hinweis auf die Position gibt, an der mit der Ausgabe des Spruchbandes auf dem Bildschirm begonnen werden soll.

Den Variablen PUNKT$ und BLANK$ werden zehn Doppelkreuze bzw. zehn Leerzeichen zugewiesen. Diese Variablen werden bei der geeignet vergrößerten Ausgabe der Zeichenmatrizen auf dem Drucker Verwendung finden.

250-460 **Text eingeben**
Dieser Programmteil dient dazu, den Text, der später auf dem Spruchband erscheinen soll, auf dem Bildschirm auszugeben und gegebenfalls zu editieren. Genau dazu fordert das Programm seinen Benutzer in Zeile 290 auf. Die Eingabe des Textes erfolgt Buchstabe für Buchstabe innerhalb der FOR-TO-NEXT-Schleife, die von Zeile 330 bis Zeile 460 reicht. Zunächst wird der Buchstabe an der aktuellen Schreibposition auf dem Spruchband (POSITION) gelöscht (wichtig für das Editieren) und danach eine Eingabe von der Tastatur entge-

gengenommen. Diese Eingabe wird in vier verschiedene Fälle unterteilt:

1. Es wurde keine Taste betätigt - erneuten Wert von Tastatur holen (INKEY$).

2. Das eingegebene Zeichen hat den Code 127 (DEL-Taste), was zur Folge hat, daß die Schreibposition auf dem Spruchband dekrementiert und der Variablen START zugewiesen wird. Danach wird zur Zeile 310 verzweigt und dort ermittelt, ob das Erniedrigen von POSITION zum Erreichen des linken Randes geführt hat. Trifft das zu, so wird als aktuelle Schreibposition der linke Rand angenommen (START=1). Mit der so manipulierten Startposition wird die FOR-TO-NEXT-Schleife zur Eingabe der Zeichen wieder durchlaufen. Jetzt macht auch das Löschen des Zeichens an der aktuellen Schreibposition einen Sinn.

3. Es wurde die RETURN-Taste betätigt (Code 13), was bewirkt, daß das Programm das Spruchband als vollständig eingegeben betrachtet und den Programmteil TEXT EINGEBEN verläßt (GOTO 480).

4. Die Eingabe eines anderen Zeichens führt zur Ausgabe des Zeichens an der aktuellen Schreibposition auf dem Spruchband und zum ungehinderten Durchlaufen der FOR-TO-NEXT-Schleife.

470-Ende **Spruchband erzeugen**

Mit diesem Programmteil wird der fertig eingegebene Text des Spruchbandes pixelweise aus dem Bildschirmspeicher des CPC gelesen und ebenfalls an den einzelnen Pixeln orientiert in der indizierten Variablen MATRIX(X,Y) gespeichert, von der aus die Ausgabe auf den angeschlossenen

Drucker erfolgen kann. Nur durch diese Verfahrensweise, die auf den ersten Blick etwas umständlich anmuten mag, ist es möglich, jedes Zeichen, auch das eines umdefinierten Zeichensatzes, auf den Drucker zu bringen.

In Zeile 500 wird aus der letzen Position, an der ein Zeichen des Spruchbandes geschrieben wurde, auf die entsprechende X-Position des Grafikcursors geschlossen (linke Spalte der Zeichenmatrix). Ist der so ermittelte Wert kleiner als Null, so bedeutet das, es wurde kein Text eingegeben und das Programm trifft auf END. Von Zeile 530 bis zum Ende des Programms reicht eine alles andere umklammernde FOR-TO-NEXT-Schleife, in der unter der Bezeichnung "INDEX" Werte erzeugt werden, die jeweils die kleinste X-Position eines Zeichens des Spruchbandes - bezogen auf den Grafikbildschirm - darstellen. In dieser FOR-TO-NEXT-Schleife befinden sich zwei Programmteile, von denen der erste die Benennung ZEICHEN VOM BILDSCHIRM LESEN und der zweite DRUKKERAUSGABE trägt.

ZEICHEN VOM BILDSCHIRM LESEN besteht aus zwei ineinander verschachtelten FOR-TO-NEXT-Schleifen, mit deren Hilfe die Matrix eines Zeichens, an der durch INDEX bestimmten Position, vom Bildschirm gelesen wird. Aus den beiden verschachtelten Schleifen ergeben sich Bildschirmpositionen, an denen mit der BASIC-Funktion TEST die Nummer des verwendeten Farbstiftes ermittelt und der entsprechend indizierten Variablen MATRIX(SPALTE,ZEILE) zugewiesen wird.

Im Programmteil DRUCKERAUSGABE sind drei FOR-TO-NEXT-Schleifen ineinander verschachtelt, von denen zwei, wie in ZEICHEN VOM BILDSCHIRM LESEN, einen Zugriffsschlüssel auf die doppeltindizierte Variable MATRIX erzeugen -

hier ist der Zugriff entsprechend den Belangen der Druckausgabe modifiziert. Die eigentliche Ausgabe auf den Drucker erfolgt in Zeile 710; dabei wird PUNKT$ ausgegeben, wenn das korrespondierende Pixel bei der Bildschirmausgabe gesetzt war und BLANK$, wenn nicht. Ist eine Zeile vollständig auf dem Drucker ausgegeben, so sorgt die dritte FOR-TO-NEXT-Schleife (Zeile 680) dafür, daß dieselbe Zeile insgesamt fünfmal ausgegeben wird, um eine maßstabsgetreue Wiedergabe der Zeichen zu erwirken.

Hat die alles umklammernde FOR-TO-NEXT-Schleife ihren Endwert erreicht, so ist das Spruchband vollständig auf dem Drucker ausgegeben.

9. CPC-Maschinenprogrammierung

Wie Sie bis jetzt gesehen haben, bietet das Schneider-BASIC eine Fülle von Grafikfunktionen, mit deren Hilfe erstaunlich viele programmtechnische Probleme gelöst werden können. Die Ausführungsgeschwindigkeiten der Programme war, obwohl wir mit einer Interpretersprache und 16 KByte Video-RAM gearbeitet haben, in nahezu allen Fällen ausreichend. Trotz alledem gibt es mehr als genug grafische Problemstellungen, deren Umsetzung den BASIC-Interpreter um "Lichtjahre" überfordert. Erste Anzeichen einer solchen Überlastung habe Sie schon im Zusammenhang mit der dreidimensionalen Grafik kennengelernt. Weiterhin gibt es Programme, deren Realisierung aus der BASIC-Ebene unmöglich ist. Denken Sie beispielsweise an die Entwicklung eines leistungsfähigen Sprite-Generators, Soft-Scrolling oder das blitzschnelle Verwalten zwei voneinander unabhängiger Bildschirmseiten.

Die Maschinenprogrammierung ermöglicht es hingegen, die technischen Gegebenheiten der uns zur Verfügung stehenden Hardware voll auszunutzen und die Ausführungszeiten unserer Programme bzw. Programmodule durch eine grafikspezifische Programmierung extrem gering zu halten. Programme wie Profi-Painter haben gezeigt, daß mit dem CPC all die interessanten Dinge möglich sind, die anfangs nur den zehn Mal so teuren Personal Computern vorbehalten wahren.

9.1 Das Herz des CPC - die Z80-CPU

Dieser Teil ist in erster Linie für den Leser gedacht, der entweder allgemeine Vorkenntnisse in der Maschinenprogrammierung besitzt oder sein Z80-Wissen wieder einmal auffrischen möchte. Eine umfangreiche Einführung sowie das Abdrucken kompletter Befehlstabellen kann hier leider nicht erfolgen, da diese Themen erstens nicht Gegenstand dieses Buches sind, und zweitens den Rahmen desselben sprengen würden. Hier soll vielmehr die grundlegende Prozessororganisation zwecks besserer Anschaulichkeit der folgenden Assemblerlistings aufgeführt werden. An

dieser Stelle möchten wir den interessierten Anfänger auf die zahlreiche, im Fachhandel erhältliche Z80-Spezialliteratur aufmerksam machen, die sowohl eine Step-by-Step-Einführung als auch ein umfangreiches, in der Zukunft als Nachschlagewerk weiter verwendbares All-Round-Wissen vermitteln kann. In diesem Zusammenhang sind die ebenfalls im DATA BECKER Verlag erschienenen Bücher "Das Maschinensprache-Buch zum CPC" und "Das Z80-Prozessorbuch" erwähnenswert.

Die Z80-CPU wurde 1974 von der amerikanischen Firma ZILOG entworfen. Sie ist eine konsequente Weiterentwicklung des bis dahin bewährten 8080-Mikroprozessors, der den Grundbaustein des CP/M-Betriebssystems - das auch mit Ihrem Schneider CPC ausgeliefert wird - repräsentiert. Die Z80 ist 8080 aufwärtskompatibel, d.h., das alle Programme, die auf einem 8080 geschrieben wurden, problemlos von der Z80 verarbeitet werden können - aber nicht umgekehrt. Aus diesem Grund entschließen sich heute nahezu alle Computerhersteller, die erheblich leistungsfähigeren Z80-Prozessoren in ihre CP/M-Systeme zu implementieren.

Schneider entschied sich für den Einsatz eines Z80-A, einem mit 4 MHz getakteten Z80-Mikroprozessor. Er gehört - wie die 8080-CPU - zur Gruppe der 8-Bit-Prozessoren.

Z80-Registersatz

Die Z80 besitzt insgesamt 22 Register. Darunter befinden sich neben Programcounter, Stackpointer und Index-Registern zwei Akkumulatoren und zwei Flag-Register. Diese, auf den ersten Blick etwas unsinnige Tatsache beruht auf einer simplen Ursache. Die Z80 beinhaltet zwei komplette Registersätze, zwischen denen mit einem Spezialbefehl gewechselt werden kann. Dieser Zweitregistersatz macht die Z80-CPU extrem leistungsfähig, da der prozessorinterne Datentransfer immer schneller als ein RAM-Zugriff ist.

Nachfolgend sollen Ihnen alle Register des Z80-Mikroprozessors vorgestellt werden.

8-Bit-Register

Die Register A, B, C, D, E, F, H und L repräsentieren den doppelt vorhandenen Grundregistersatz unseres Mikroprozessors. In ihnen können Werte von 0 bis 255 gespeichert werden. In diesem Zusammenhang haben das A- und F-Register entgegen den übrigen Registern besondere Funktionen.

A (Akkumulator): Der Akkumulator oder Akku enthält immer das Ergebnis einer arithmetischen oder logischen 8-Bit-Operation. Er kann selbstverständlich auch als Zwischenspeicher dienen.

F (Flag-Register): Die Bits im Flag-Register (Status-Register) werden nach jeder arithmetischen oder logischen Operation vom Prozessor beeinflußt, und zeigen beispielsweise an, ob ein Registerüberlauf stattgefunden hat oder das Ergebnis der Operation Null ist. So können mit Hilfe des F-Registers logische Entscheidungen getroffen werden.

8-Bit-Spezialregister

Die verbleibenden 8-Bit-Register I und R werden vom Betriebssystem der CPC-Rechner nicht eingesetzt. Obwohl diese Register normalerweise für Spezialzwecke verwendet werden, können sie dennoch sinnvoll in unseren Programmen verwendet werden.

R (Refresh-Reg.): Das R-Register benötigt die Z80-CPU, um einen automatischen Refresh durchzuführen. In diesem Zusammenhang werden die relativen Bits 0-6 ständig inkrementiert. Das relative Bit Nummer 7 wird nicht beeinflußt. Es kann vom Benutzer nach Bedarf manipuliert werden. Das Refresh-Register kann übrigens für eine sinnvolle Anwendung "mißbraucht" werden. Falls Sie bei-

spielsweise ein eigenes Videospiel entwerfen möchten, sind Sie mit größter Warscheinlichkeit auf einige Zufallszahlen angewiesen. Da der Zustand des Refresh-Register nie vorhersehbar ist, kann es somit hervorragend als Zufallsgenerator dienen.

I (Interrupt-Reg.): Das 8 Bit breite Interrupt-Register wird nur im Zusammenhang mit dem Interrupt-Mode IM2 verwendet. Es repräsentiert dann das High-Byte einer 256 Byte langen Memory-Page. In ihr können bis zu 128 Interrupt-Vektoren gespeichert werden. Das I-Register kann, falls IM2 in Ihrem Programm keine Verwendung finden sollte, zum schnellen 8-Bit-Zwischenspeicher degradiert werden. Der Zugriff auf das Interrupt-Register, ist zwar langsamer als auf ein normales Register jedoch immer schneller als ein direkter Speicherzugriff.

16-Bit-Doppelregister

Die Z80 ist einer der wenigen 8-Bit-Prozessoren, der 16-Bit-Register enthält. Diese 16-Bit-Register bieten enorme Geschwindigkeitsvorteile. Sie erlauben uns beispielsweise zwei Zahlen im Bereich von 0 bis 65535 mit einem einzigen Befehl zu addieren. Selbst eine Multiplikation mit 2 ist auf 16-Bit-Ebene auf einen Schlag möglich!

Die Z80 kennt zwei verschiedene Arten von 16-Bit-Registern:

- echte 16-Bit-Register und
- simulierte 16-Bit-Register - sogenannte Doppelregister

Die Doppelregister bestehen aus den 8-Bit-Register-Gruppen BC, DE und HL. Da sie sich im Grunde aus den Elementarregistern zusammensetzen, sind auch diese Register erfreulicherweise doppelt vorhanden (Zweitregistersatz!). Das HL-Register übernimmt in dieser Adressierungsart übrigens die Rolle eines 16-Bit-Akkus. Das Flag-Register wird wie gehabt verwendet.

16-Bit-Indexregister

Die Z80-CPU stellt uns für die indizierte Adressierung zwei weitere 16-Bit-Register (IX und IY) zur Verfügung. Diese Register können mit einem zusätzlichen 8-Bit-Offset (Distanzadresse) versehen werden, der automatisch zum Inhalt des entsprechenden Indexregisters addiert wird. Dieser Offset wird im Zweierkomplement dargestellt. Auf diese Weise können auch negative Distanzadressen addiert werden.

16-Bit-Spezialregister

Die noch verbleibenden 16-Bit-Register werden unter anderem von der Z80 selbst beeinflußt.

SP (Stackpointer): Das SP-Register (Stapelregister) beinhaltet die Adresse eines von uns reservierten Speicherbereichs, in dem 16-Bit-Daten zwischengespeichert werden können. Die Z80-CPU legt in diesem Bereich die Rücksprungadressen aller CALL-Befehle ab. Das Transferieren eines Datenwortes in diesen Bereich hat ein doppeltes Dekrementieren des Stackpointers zur Folge.

PC (Prg.-Counter): Das PC-Register (Programmzähler) enthält immer die aktuelle Adresse des von der CPU zu verarbeitenden Befehls. Nach Befehlsausführung addiert der Prozessor automatisch die Bytelänge des gerade verarbeiteten Operationscode und möglichen Operanden zum Inhalt des PC-Registers. Es kann auch durch einen Jump- oder Call-Befehl direkt vom Benutzer geladen werden.

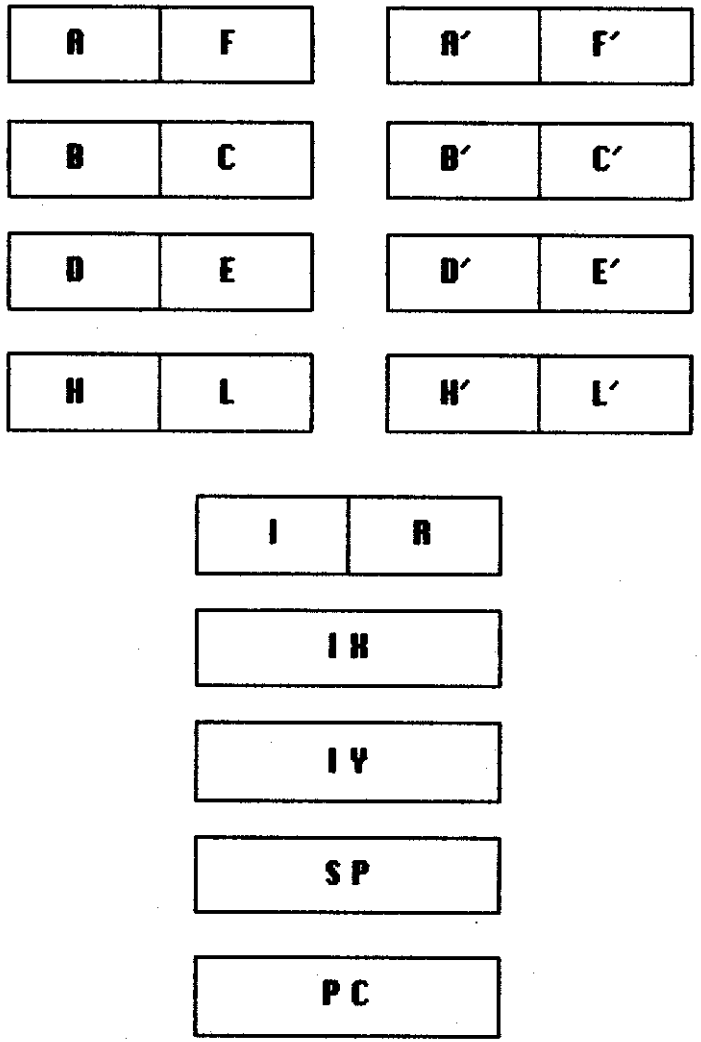


Abb. 20: Z80-Registersatz

9.2 Der Video-Controller

Der Video-Controller ist für jegliche Darstellung auf dem Bildschirm verantwortlich. Es handelt sich bei allen CPCs um den HD 6845 - einen leistungsstarken Vertreter seiner Art, der auch in wesentlich teureren Systemen anzutreffen ist.

Der Video-Controller wird nach bester Z80-Manier durch seine I-/O-Ports adressiert. Diese Port-Adressierung hat gegenüber der Memory-Maped-Methode den Vorteil, daß kein zusätzlicher RAM-Bereich verloren geht. Apropos Port-Adressierung: Bekanntermaßen stellt die Z80 dem Benutzer 256 I-/O-Ports zur Verfügung. Das Betriebssystem des Schneider-CPC verwaltet hingegen 65536 Ein-/Ausgabe-Kanäle. Wie? Nun - die Konstruktoure der Z80 haben sich zwecks universeller Einsetzbarkeit Ihrer CPU einen einfachen, jedoch sehr effektiven Trick einfallen lassen. Sie beinhaltet nämlich Befehle, mit deren Hilfe durch ein 16 Bit breites Datenwort der entsprechende I-/O-Port selektiert werden kann. Zu diesem Zweck muß das BC-Register mit der gewünschten Port-Adresse geladen werden. Diese Adressierungsart hat leider auch Nachteile. So können nicht alle zur Verfügung stehenden Port-Befehle in diesem Modus verwendet werden.

Für die Adressierung des HD 6845 stehen 18 Daten-Register sowie ein Adreß-Register zur Verfügung. Bevor ein Datenregister beschrieben oder gelesen werden kann, muß seine Nummer in den unteren 5 Bits des Adreß-Register abgelegt werden. Die Register werden dabei über folgende Ports adressiert:

Adreß-Register:	&BC00
Daten-Register:	&BD00

Die gezielte Programmierung des Video-Controllers ermöglicht letztendlich eine wesentlich schnellere, elegantere und einfachere Grafik-Programmierung. So ist beispielsweise der Lightpen-Anschluß sowie das Scrollen des Bildschirminhalts mit seiner Hilfe ohne Probleme möglich.

Der Registersatz des Video-Controllers

AR Adreß-Register (5-Bit-WRITE)
Das Adreß-Register muß mit der Nummer des zu adressierenden Datenregisters geladen werden. Es akzeptiert die Register-Nummern 0 bis 17. Die Werte 18 bis 31 werden ignoriert. Das Adreß-Register kann nur beschrieben werden.

R0 Horizontal Total (8-Bit-WRITE)
Dieses Register enthält die Anzahl der Zeichen pro totale Zeile. Das Wort "totale" bezieht sich nicht auf die sichtbaren Zeichen, sondern auf den vom Betriebssystem erwarteten Wert. Da es auch Zeichen für den Rand und Strahlenrücklauf mit einbezieht, muß der Wert das 1,5fache der real dargestellten Zeichen betragen. R0 kann nur beschrieben werden.

R1 Horizontal Displayed (8-Bit-WRITE)
R1 enthält die reale (sichtbare) Anzahl der Zeichen pro Zeile. Dieser Wert muß immer kleiner als der Wert in R0 sein. R1 kann nur beschrieben werden.

R2 Horizontal Sync-Position (8-Bit-WRITE)
Mit diesem Register kann ein ganz besonderer Effekt erzielt werden. Es enthält nämlich den Zeitpunkt des HSync-Impulses. Eine Verringerung dieses Wertes verschiebt das gesamte Monitorbild nach rechts, eine Erhöhung nach links. R2 kann nur beschrieben werden.

R3 Sync Width (4-Bit-WRITE)
Die relativen Bits 0 bis 3 bestimmen die Breite der HSync- und VSync-Impulse. Die verbleibenden Bits werden nicht benötigt. R3 kann nur beschrieben werden.

R4 Vertical Total (7-Bit-WRITE)
Der Inhalt von R4 enthält die Anzahl der Rasterzeilen pro Bild. Er bestimmt ferner, ob die Bildwiederholfrequenz in 50- oder 60-Herz-Intervallen synchronisiert wird. R4 kann nur beschrieben werden.

R5 Vertical Total Adjust (6-Bit-WRITE)
Die unteren 6 Bit dieses Registers sind für den Feinausgleich (tuning) der Bildwiederholfrequenz verantwortlich. R5 kann nur beschrieben werden.

R6 Vertical Displayed (7-Bit-WRITE)
Die relativen Bits 0 bis 7 repräsentieren die reale Anzahl der Rasterzeilen auf dem Display. Der hier eingetragene Wert muß immer kleiner als der in R4 sein. R6 kann nur beschrieben werden.

R7 Vertical Sync-Position (7-Bit-WRITE)
Mit dem Inhalt von R7 kann analog zu R2 ein vertikales Verschieben des Gesamtbildes erreicht werden. Es bestimmt den Zeitpunkt des VSync-Impulses. Wird dieser Wert verringert, verschiebt sich das Monitorbild nach unten, eine Erhöhung schiebt es nach oben. R7 kann nur beschrieben werden.

R8 Interlace (2-Bit-WRITE)
Die relativen Bits 0 und 1 legen fest, ob die Bildschirmdarstellung mit oder ohne Interlace (Zeilensprung-Verfahren) erfolgen soll. R8 kann nur beschrieben werden.

R9 Maximum Register Adress (5-Bit-WRITE)
R9 enthält die Anzahl der Rasterzeilen, die für ein darzustellendes Zeichen verwendet werden. Dieses Register kann nur beschrieben werden.

R10 Cursor Start Raster (7-Bit-WRITE)
R10 ist für das Cursor-Management verantwortlich. Die unteren 5 Bits des Registers enthalten die Rasterzeile, ab der der Cursor gezeichnet werden soll. Zusätzlich

wird mit den relativen Bits 5 und 6 einer von vier Cursor-Modi festgelegt. Die Zuordnung ist wie folgt:

Bits 6	5	
0	0	normaler Cursor
0	1	kein Cursor
1	0	Cursor blinkt etwa 3 mal pro Sekunde
0	1	Cursor blinkt etwa 1,5 mal pro Sekunde

R10 kann nur beschrieben werden.

R11 Cursor End Raster (5-Bit-WRITE)
R11 ist für das Cursor-Management verantwortlich. Die unteren 5 Bits des Registers enthalten die Rasterzeile, ab der der das Zeichnen des Cursors endet. R11 kann nur beschrieben werden.

R12 Start Adress High (6-Bit-READ/WRITE)
Eine besonders interessante Möglichkeit eröffnet uns das Register R12. Die unteren 6 Bits legen das High-Byte der Adresse fest, ab welcher der Videobereich im 16KByte-Adreßbereich des Video-Controllers beginnt. R12 kann sowohl gelesen als auch beschrieben werden. Wird es gelesen, so befinden sich die beiden obersten Bits immer in Low-Zustand.

R13 Start Adress Low (8-Bit-READ/WRITE)
Das Register R13 leget das Low-Byte der Adresse fest, ab welcher der Videobereich im 16KByte-Adreßbereich des Video-Controllers beginnt. R13 kann sowohl gelesen als auch beschrieben werden.

R14 Cursor High (6-Bit-READ/WRITE)
Die relativen Bits 0 bis 5 von R14 repräsentieren das High-Byte der aktuellen Cursor-Position. R14 kann sowohl beschrieben als auch gelesen werden.

R15 Cursor Low (8-Bit-READ/WRITE)
Dieses Register enthält das Low-Byte der aktuellen Cursor-Position. R15 kann sowohl beschrieben als auch gelesen werden.

R16 Light-Pen-High (6-Bit-READ)
R16 ist ein Lightpen-Register und kann in Verbindung mit R17 genutzt werden. Es enthält nach jedem positiven Strobe-Impuls das High-Byte der derzeit aktiven Bildschirmadresse. R16 kann nur gelesen werden.

R17 Light-Pen-Low (8-Bit-READ)
Dieses Register ist ein Lightpen-Register und kann in Verbindung mit R16 genutzt werden. Es enthält nach jedem positiven Strobe-Impuls das Low-Byte der derzeit aktiven Bildschirmadresse. R17 kann nur gelesen werden.

9.3 Das Gate Array

Das Gate Array ist ein weiterer wichtiger Baustein Ihres CPC-Computers. Neben wichtigen System-Koordinationen löst es auch zahlreiche grafikbezogene Aufgaben. Darunter fallen beispielsweise die Erzeugung von Farben oder das Umschalten des Bildschirm-Modus.

Wie auch der Video-Controller wird der Registersatz des Gate Arrays über die Ein-/Ausgabe-Ports der Z80 adressiert. Die dabei zu verwendende Adresse ist &7F00. Mit ihrer Hilfe können drei verschiedene Register angesteuert werden. Diese Register können nur beschrieben werden. Falls Sie dennoch Rückmeldungen in Ihren Programmen benötigen, müssen Sie jede Manipulation des Gate-Arrays in Variablen übertragen. Diese Methode funktioniert selbstverständlich nur bei Ihren eigenen Routinen.

Die Register des Gate Arrays

PEN: Das PEN-Register ist für die Farbzuordnung verantwortlich. Es wird mit dem Offset der Adresse geladen, die einen neuen Farbwert bekommen soll. Dieser Offset entspricht dem PEN-Parameter im Amstrad-BASIC.

Die beiden obersten Bits des Adreßbyte sind für die Selektion des PEN-Registers verantwortlich. Sie müssen sich beide im Low-Zusatnd befinden. Die relativen Bits 0-3 enthalten den Offset der zu ändernden Adresse (0-15). Während das 5. Bit nicht benötigt wird, hat das relative Bit Nummer 4 eine besondere Bedeutung. Falls es gesetzt sein sollte, werden die unteren 4 Bits ignoriert und der Wert des unmittelbar folgenden Out-Befehls, der das INK-Register adressieren muß, als neue Rahmenfarbe interpretiert.

INK: Dieses Register kann nur im Zusammenhang mit dem PEN-Register verwendet werden. Es lädt die Adresse, die mit dem PEN-Register selektiert wurde, mit einem neuen Farbwert. Der Farbwert ist mit den Farbnummern des BASIC-INK-Befehls identisch.

Die relativen Bits 7 und 6 des Adreßbyte sind für das Auswählen des INK-Registers verantwortlich. Bit 7 muß sich dabei immer im Low- und Bit 6 im High-Zustand befinden. Die unteren 5 Bits enthalten die Nummer der zu ladenden Farbe (0-26). Das relative Bit Nummer 5 wird nicht benötigt.

MFR: Das Multifunktionsregister hat verschiedene Aufgaben. Es kann beschrieben werden, wenn sich das relative Bit 7 des Adreßbyte im High- und Bit 6 im Low-Zustand befindet. Während das Bit Nummer 5 nicht benötigt wird, sind die restlichen Bits für die Funktionsauswahl verantwortlich. Sie haben folgende Bedeutung:

Bit 4: Löschen des V-Sync-Zählers
Bit 3: ROM ein-/ausschalten (&C000 bis &FFFF)
Bit 2: ROM ein-/ausschalten (&0000 bis &3FFF)
Bit 1: Ändern des Bildschirm-Modus
Bit 0: "

Ist bei Ausgabe des Adreßbytes Bit 4 gesetzt, so wird der System-Pointer, der auf den abzuarbeitenden Interrupt-Vektor zeigt, auf Null gesetzt. Der Interrupt-Vektor mit der höchsten Priorität wird folglich beim nächsten Interrupt abgearbeitet.

Die relativen Bits 2 und 3 bestimmen mit ihrem aktuellen Zustand die ROM-/RAM-Speicherkonfiguration. Falls eines der beiden Bits gesetzt ist, so wird aus der entsprechenden RAM-Bank gelesen - ansonsten aus dem ROM-Bereich.

Die letzten beiden Bits beziehen sich auf die Definition der Bildschirm-Modi. Durch ihre vier verschiedenen Kombinationsmöglichkeiten können mit ihrer Hilfe alle Bildschirmmodi sowie ein Spezial-Mode aufgerufen werden. Die Bitzuweisung ist wie folgt:

Bit 1	Bit 0	Funktion
0	0	Mode 0 (160*200 Punkte/16 Farben/Blinken)
0	1	Mode 1 (320*200 Punkte/4 Farben/Blinken)
1	0	Mode 2 (640*200 Punkte/2 Farben/Blinken)
1	1	Mode 0 (160*200 Punkte/16 Farben)

Und hier noch einmal die Zusammenfassung der oberen Bit-Zustände aller Register:

Bit 7	Bit 6	Funktion
0	0	PEN-Register mit Farbadressen-Offset laden
0	1	INK-Register mit Farbnummer laden
1	0	Funktion im MFN-Register initialisieren
0	0	wird vom System für Speicherumschaltung des 6128 verwendet

So - das wäre alles! Und wie kann das Gate Array sinnvoll eingesetzt werden? Nun - falls Sie innerhalb Ihrer Maschinenprogramme in erster Linie auf ROM-Routinen zurückgreifen wollen, ist es in den meisten Fällen sicherlich sinnvoll, auf alle vorhandenen Routinen zurückzugreifen. Diesbezügliche ROM-Zugriffe über Vektoren werden etwas später vorgestellt. Wenn Sie jedoch, sei es aus Speicherplatz- oder Geschwindigkeitsgründen, auf eigene Routinen zurückgreifen möchten, ist eine genaue Kenntnis des eben beschriebenen unumgänglich. Wie einfach die Benutzung der Gate-Array-Register ist, soll folgendes Beispiel aufzeigen:

```
; ##################################
; ##                              ##
; ## GATE ARRAY ANWENDUNGSBEISPIEL ##
; ##                              ##
; ##################################
;
;
9000        ORG &9000
;
9000        DI                  ;System abkoppeln
9001 START: LD   BC,&7F00       ;Port definieren
9004        LD   A,&02          ;Farb-Register 2 wählen
9006        OUT  (C),A          ;PEN-Register
                                 ;adressieren
9008        LD A,&43            ;Farbe 3 wählen
900A        OUT (C),A           ;INK-Register
                                 ;adressieren
900C        EI                  ;Interrupt zulassen
900D        RET                 ;Fertig!
```


Diese kleine Routine lädt das Farb-Adreßregister Nummer 2 mit dem Farbwert 3. Sie entspricht strenggenommen dem BASIC-Befehl INK 2,3. Falls Sie versuchen sollten, dasselbe Beispiel mit den BASIC-Befehlen

```
OUT &7F00,2:OUT &7F00,&43      <RETURN>
```

zu realisieren, werden Sie eine Überraschung erleben. Ein kurzes Aufblitzen der gewählten Farbe ist alles, was Sie erreichen werden. Dieses Phänomen hängt mit der Tatsache zusammen, daß das Betriebssystem grundsätzlich alle Farben des CPC mittels Interruptsteuerung blinkend darstellt. Bei jedem Interrupt wird automatisch von der System-Software einer der beiden Farbwerte in das Gate Array übertragen. Da im Normalfall beide für die Blinksequenz benötigten Farben identisch sind, merkt der Benutzer den ständigen Farbwechsel nicht. Wenn jedoch - wie in unserem Beispiel - aus der BASIC-Ebene eine Farbmanipulation mittels Gate Array vorgenommen wurde, ist die neue Farbe genau bis zum nächsten Interrupt zu sehen und äußert sich durch ein kurzes Aufblitzen auf Ihrem Bildschirm.

9.4 Grafikbezogene Z80-Programmierung

Die Z80-CPU ist ein registerorientierter Prozessor. Das bedeutet, daß die Ausführungsgeschwindigkeit unserer Programme um so größer wird, je mehr Register verwendet werden. Da viele Grafikroutinen mit höchstmöglicher Geschwindigkeit durchlaufen werden sollen, gilt es, alle Vorteile der Z80 so optimal wie möglich zu nutzen. Gerade auf dem Gebiet der Grafikprogrammierung lohnt es sicht, jeden unnötigen Taktzyklus zu vermeiden, denn nur so können fließende Bewegungen auf dem Bildschirm dargestellt werden.

In Kapitel 9.1 haben wir erfahren, daß der Z80-Prozessor zwei Hauptregistersätze beinhaltet. Der Zweitregistersatz wird jedoch

vom Betriebssystem des CPC genutzt und kann aus diesem Grund leider nicht in eigenen Routinen verwendet werden. Falls dies dennoch geschehen sollte, ist ein unabwendbarer Systemabsturz die Folge.

Diese Tatsache schränkt die auf Geschwindigkeit ausgelegte Grafikprogrammierung natürlich stark ein. Mit einem kleinen Trick ist es dennoch möglich, das Schneider-Betriebssystem zu überlisten und doch noch an die begehrten Zweitregister heranzukommen.

Folgende Routine vermeidet einen Systemabsturz:

```
; #################
; ##             ##
; ## USE 2ND REG ##
; ##             ##
; #################
;
;
DI              ;SYSTEM "ABKOPPELN"
EX   AF,AF'     ;ALLE ZWEITREGISTER HOLEN
EXX             ;      "
PUSH AF         ;UND RETTEN
PUSH BC         ;      "
PUSH DE         ;      "
PUSH HL         ;      "
;
CALL PROGRAM    ;MAIN-PROGRAM AUFRUFEN
;
POP  HL         ;GERETTETE REGISTER HOLEN
POP  DE         ;      "
POP  BC         ;      "
POP  AF         ;      "
EXX             ;ARBEITSREGISTER HOLEN
EX   AF,AF'     ;      "
EI              ;SYSTEM "EINHÄNGEN"
;
RET             ;RÜCKSPRUNG INS BASIC
```


Unser kleines Startprogramm verhindert den Interrupt, rettet den Inhalt aller Zweitregister auf den Stack und ruft Ihr Programm oder Ihre Routine auf. Danach werden die ursprünglichen Werte in die Zweitregister zurückgeschrieben und der Interrupt wieder freigegeben - das ist alles.

Da während der ganzen Zeit der Interrupt ausgeschaltet ist, merkt das Betriebssystem die Benutzung der Zweitregister nicht. Bitte beachten Sie, daß Ihr Programm den Interrupt nicht freigeben darf. Falls das dennoch erforderlich sein sollte, muß der System-Interrupt-Vektor in Adresse &38 zwischengespeichert werden. Danach können Sie einen beliebigen neuen Vektor eintragen und den Interrupt wieder freigeben. Kurz vor Beendigung Ihres Programms muß schließlich der Original-Vektor wieder zurückgeschrieben werden.

Aufbau des Video-RAMs

Die Grafik des Schneider CPC gestattet eine maximale Auflösung von 640 mal 200 Punkten bzw. 80 mal 200 Bytes. Das Video-RAM beginnt normalerweise ab Adresse &C000 und endet bei &FFFF. Leider kennt das Schneider-Betriebssystem keine Analogie zwischen der internen Organisation des Video-RAMs und der reellen Bildschirmausgabe. Der vertikale Grafikaufbau erfogt nicht zeilen- sondern zeichenweise, d.h. es werden immer 7 Zeilen übersprungen. Das folgende kleine BASIC-Programm soll das verdeutlichen:

```
10 REM ######################################
20 REM ##                                  ##
30 REM ## Bildschirmspeicher vollschreiben ##
30 REM ##                                  ##
40 REM ######################################
50 '
55 '
60 MODE 2
```

```
70 For I=&C000 to &FFFF
80 POKE I,&FF
90 NEXT
```

Aber warum verwendet das Schneider-Betriebssystem eine derartige - auf den ersten Blick unsinnige - Bildschirmorganisation? Nun - sie eignet sich hervorragend zur Darstellung eines 8 mal 8 Zeichens. Da der CPC auch Text jeglicher Art im Grafik-Modus erzeugt, bietet dieses Verfahren gewisse Geschwindigkeitsvorteile. In vielen anderen Fällen erweist es sich jedoch als ein programmiertechnisch komplexes Übel, das die Grafikausgabegeschwindigkeit Ihrer Z80-Programme unerbittlich verlangsamt.

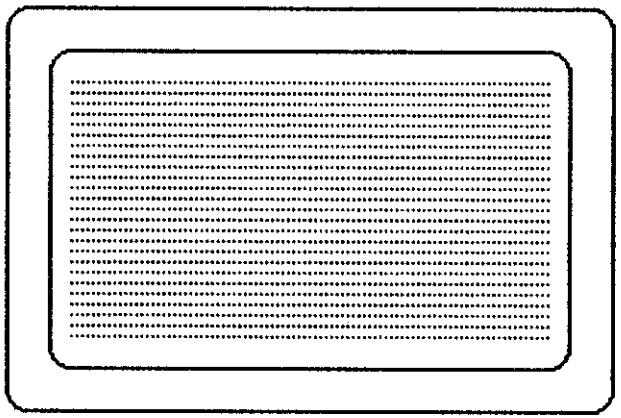


Abb. 21: Bildschirmaufbau

Obwohl die Z80-CPU ein 8-Bit-Prozessor ist, stellt sie uns - wie in Kapitel 9.1 erwähnt - 16-Bit-Register und eine 16-Bit-Arithmetik zur Verfügung. Die letztere bezieht sich auf die Addition und die Subtraktion. Diese Möglichkeit erlaubt uns einen sowohl einfachen als auch effektiven 16-Bit-Algorithmus zu entwickeln, der das Ziehen einer vertikalen Linie ermöglicht.


```
        ; ###################
        ; ##               ##
        ; ## VERTICAL-LINE ##
        ; ##               ##
        ; ###################
        ;
        ;
START:  LD   HL,VSTART          ;16-Bit Startadresse
        LD   DE,&0800           ;Zeilenoffset 1
        LD   BC,&3FAF           ;Zeilenoffset 2
        LD   A,VCOUNT           ;Anzahl der Punkte (1-200)
;
NXTLN:  LD   (HL),COLOR         ;Gewünschte Farbe plotten
        ADD  HL,DE              ;Nächste Zeile
        JP   NC,NOOFF           ;Zeilenoffset 2?
        SBC  HL,BC              ;Entgültige Zeile
;
NOOFF:  DEC  A                  ;Alle Punkte geplottet?
        RET  Z                  ;Wenn ja fertig, ansonsten
        JP   NXTLN              ;plotte nächsten Punkt
```

COLOR enthält Informationen über die aktuelle Liniendicke und Linienfarbe, welche zugleich "modebezogen" sind, d.h., daß alle drei Grafikmodi spezifische Parameter verlangen.

MODE 2: (620 mal 200 Pixel/2 Farben)

Die Parameter für diesen Mode sind am einfachsten zu ermitteln. Jedes gesetzte Bit repräsentiert einen Punkt der PEN-Farbe 1 - ein rückgesetztes Bit die PAPER-Farbe. Es können maximal 8 Linien gleichzeitig gezeichnet werden.

MODE 1: (320 mal 200 Pixel/4 Farben)

Hier wird es schon etwas komplizierter. Die relativen Bitpaare (7,3), (6,2), (5,1) und (4,0) repräsentieren jeweils ein Pixel. Durch LD A,1 wird beispielsweise der relativen Linie 0 die PEN-Farbe 1 zugeordnet.

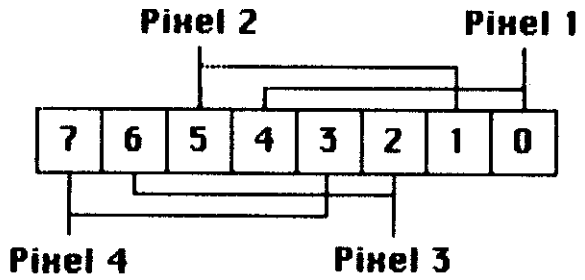


Abb. 22: Pixel-Organisation (MODE 1)

MODE 0: (160 x 200 Pixel/16 Farben)

Dieser Modus stellt uns 4 Farbinformationsbits zur Verfügung. Dies ermöglicht es uns, einem Pixel 24 = 16 Farben zuzuordnen. Die PEN-Farbe der linksbündigen Bildpunkte wird durch die relativen Bits (7,3,5,1) definiert. Analog dazu repräsentieren die relativen Bits (6,2,4,0) die PEN-Farben der rechtsbündigen Pixel. Bei der Farbdefinition ist unbedingt zu beachten, daß die Wertigkeit der Bits nicht mit der reellen Reihenfolge eines üblichen Binärwortes übereinstimmen. In der von AMSTRAD verwendeten Verschlüsselung entsprechen nämlich die Bits 1, 5, 3 und 7 aller linksbündigen Pixel der Binär-Kodierung 20 bis 23. Analog dazu bilden die Bits 0, 4, 2 und 6 die Farbkodierung aller rechtsbündigen Pixel.

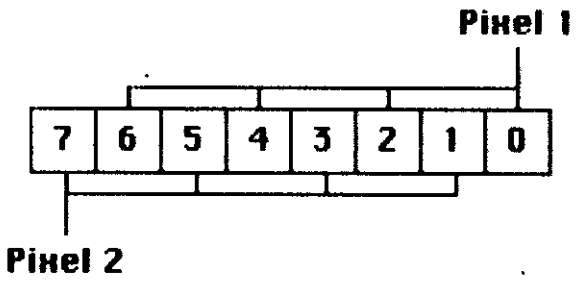


Abb. 23: Pixel-Organisation (MODE 0)

Das Ziehen einer horizontalen Linie ist hingegen viel einfacher. Dies ist sogar mit einem einzigen Z80-Befehl möglich! Sie glauben mir nicht? Nun, dann studieren Sie doch einmal das nachstehende Programm:

```
; ###############
; ##            ##
; ## LDIR-Demo ##
; ##            ##
; ###############
;
;
START:  LD   HL,VSTART      ;Start des Video-RAM
        LD   DE,VSTART+1    ;2. Wert für Transfer
        LD   BC,COUNT       ;Linienlänge
        LD   A,COLOR        ;Linienfarbe
        LD   (HL),A         ;Erstes Byte plotten
;
        LDIR                ;Ziehe Linie!
;
        RET                 ;Fertig!
```

Der LDIR-Befehl zeichnet die horizontale Linie nach der Initialisierung mit einer erstaunlichen Geschwindigkeit! Er benötigt pro Byte nur 21 Taktzyklen. Bitte beachten Sie, daß dieses Verfahren byteweise arbeitet.

Das Ziehen einer horizontalen Linie kann sogar noch erheblich scheller erfolgen. Um das zu erreichen, behelfen wir uns mit einem kleinen Trick. Da die Z80-CPU einen 16-Bit-Stack-Pointer beinhaltet, kann er mit der letzten Bildschirmadresse einer Linie geladen werden. Wenn wir jetzt das High- und Low-Register eines Doppelregisters mit dem Wert der gewünschten, modebezogenen Farbe laden, können durch einfaches "PUSHen" zwei Bytes gleichzeitig auf dem Bildschirm geplottet werden. In diesem Zusammenhang ist unbedingt zu beachten, daß der ursprüngliche Wert des Stack-Pointer vor Ausführung der Routine zwischengespeichert und nach Beendigung des Zeichenvorgangs wieder mit demselben geladen werden muß. Weiterhin muß der Interrupt während der gesamten Ausführungsperiode deaktiviert sein.

Dieses Verfahren ist zweifelsfrei die schnellste Methode, um eine horizontale Linie mittels Z80-Prozessor zu generieren. Das nun folgende Maschinenprogramm soll Ihnen zeigen, wie die Realisation einer derartigen Routine aussehen kann.

```
; ######################################
; ##                                  ##
; ##  ZIEHEN EINER HORIZONTALEN LINIE ##
; ##        MITTELS STACK-POINTER     ##
; ##                                  ##
; ######################################
;
;
VEND:   EQU &C050          ;ENDE DER LINIE
COLOR:  EQU &FFFF          ;PLOTFARBE
COUNT:  EQU 40             ;PLOTTE 80 BYTES
;
;
START:  DI                 ;INTERRUPT SPERREN
```



```
        LD   (STACK),SP    ;STACK-POINTER RETTEN
;
        LD   SP,VEND       ;SP = LINIENENDE
        LD   HL,COLOR      ;HL = ZU PLOTTENDES WORT
        LD   B,COUNT       ;B  = LINIENLAENGE
;
LINE:   PUSH HL            ;ZWEI BYTE PLOTTEN
        DJNZ LINE          ;FERTIG?
;
        LD   SP,(STACK)    ;STACK-POINTER HOLEN
        EI                 ;INTERRUPT FREIGEBEN
;
        RET                ;FERTIG!
;
STACK:  DEFS 2             ;PLATZ FUER STACK-POINTER
```

Tabellenorientierte Bildschirmorganisation

Das Lokalisieren einer Bildschirmkoordinate im Video-RAM ist trotz alledem nur mit einem zeitaufwendigen Algorithmus möglich.

Dieser Rechenaufwand kann jedoch mit einem einfachen Trick - dem *tabellenorientierten Lokalisieren* - auf ein Minimum reduziert werden. Sie können mittels eines einfachen BASIC-Programms eine Tabelle aller Y-Adressen erstellen lassen und diese in einen beliebigen Speicherbereich POKEen. Ihr Maschinenprogramm muß dann lediglich die gewünschte Adresse aus der Y-Tabelle lesen und den X-Wert addieren - fertig!

Die Y-High/Low-Tabelle kann mit folgendem BASIC-Programm generiert werden:

```
 1 REM ########################
 2 REM ##                    ##
 3 REM ## Y-Tabellengenerator ##
 4 REM ##                    ##
 5 REM ########################
 6 '
 7 '
10 AD=&C000:REM Video Start
20 YH=&6000:REM Y-High-Byte Tabelle
30 YL=&6100:REM Y-Low-Byte Tabelle
40 '
50 FOR I=0 to 199
60 POKE YH+I,INT(AD/256)
70 POKE YL+I,AD-INT(AD/256)*256
80 AD=AD+&800
90 IF AD>-1 THEN AD=AD-&3FB0
100 NEXT
110 '
120 SAVE "YTAB",B,&6000,&1C7
```

Die High-Byte-Tabelle haben wir in unserem Beispiel nach &6000 und die Low-Byte-Tabelle nach &6100 gelegt. Sie können die Tabellen selbstverständlich in jeden anderen Speicherbereich transferieren. Beachten Sie jedoch in diesem Zusammenhang, daß der Inhalt der Low-Bytes immer &00 betragen sollte. Dies gestattet Ihnen einen wesentlich schnelleren Tabellenzugriff Ihres Maschinenprogramms, da wir uns zwei 16-Bit-Additionen sparen. Aber nun zur eigentlichen Maschinenroutine:

```
; ########################
; ##                    ##
; ## TABGET Demoprogramm ##
; ##                    ##
; ########################
;
;
TBYH:   EQU  &6000        ;Y-High-Tabelle
TBYL:   EQU  &6100        ;Y-Low-Tabelle
```



```
;
        LD   E,27         ;Übergebe X
        LD   A,55         ;Übergebe Y
        CALL CALC         ;Berechne Adresse
;
; *** Hier kann Ihre eigene Routine eingebaut
; *** werden!
;
        RET               ;Fertig!
;
CALC:   LD   (OFF1+1),A   ;Übergebe Y
        LD   (OFF2+1),A   ;    "
OFF1:   LD   HL,(TBYL)    ;Hole Wert aus
OFF2:   LD   A,(TBYH)     ;Tabelle
        LD   H,A          ;    "
        SRL  E            ;X=X/4 (MODE 1)
        SRL  E            ;    "
        LD   D,0          ;Addiere X/4
        ADD  HL,DE        ;    "
        RET               ;Ergebnis in HL
```

In diesem Beispiel wurde die Adresse eines Pixels in MODE 1 berechnet. Der X-Wert muß diesbezüglich durch 4 dividiert werden. Für die verschiedenen Modi gilt folgendes:

MODE 0: X = X/2 (160*200 Pixel)
MODE 1: X = X/4 (320*200 Pixel)
MODE 2: X = X/8 (640*200 Pixel)

Sie werden vielleicht bemerkt haben, daß das Ergebnis der X-Division immer 80 ist. Das kommt daher, weil eine horizontale Bildschirmzeile immer aus 80 Bytes aufgebaut ist.

Mit Hilfe dieser Erkenntnisse können wir jetzt ohne weiteres eine PLOT-Funktion aufbauen, deren Ausführungsgeschwindigkeit dem CPC-Äquivalent um ein Mehrfaches überlegen ist. Sie verlangt im BC-Register die X- (0 bis 319) und im Akku-

mulator die Y-Koordinaten (0 bis 199). In der Adresse FARBE+1 wird die aktuelle Plotfarbe definiert. Zu diesem Zweck muß FARBE+1 vor dem Anspringen der PLOT-Routine mit

```
&00, &0F, &F0 oder &FF
```

geladen werden. Diese Werte repräsentieren die INKs 0 bis 3 (Mode 1) und manipulieren durch eine AND-Verknüpfung die Nibbles der benötigten Bitmasken.

Unsere Routine verlangt neben unseren Y-High/Low-Tabellen eine Bitmasken-Tabelle. Sie ist vier Bits lang und dient zur Negativ-Maskenerstellung für den Hintergrund. Die Bitmasken-Tabelle sollte - wie die Y-High/Low-Tabellen - aus Geschwindigkeitsgründen auf einer Adresse mit dem Low-Byte &00 liegen.

```
; ###################
; ##               ##
; ## PLOT-FUNKTION ##
; ##               ##
; ###################
;
; Eingabeparameter: BC = X-Pos (0-319)
;                 :  A = Y-Pos (0-199)
;                 : (FARBE+1) = Punktfarbe
;                   (&00, &0F, &F0, &FF)
;
; zerstörte Register: AF, BC, HL
;
;
        ORG  &6200
;
BITNR:  DEFB %01110111          ;negativ-Masken
        DEFB %10111011          ;    "
```



```
        DEFB %11011101          ;    "
        DEFB %11101110          ;    "
;
;
;       ORG  &7000              ;Start Adresse
;
TBYH:   EQU  &6000              ;Y-Low-Tabelle
TBYL:   EQU  &6100              ;Y-High-Tabelle
BITNR:  EQU  &6200              ;Bitmasken
;
PLOT:   LD   (OFF1+1),A         ;Übergebe Y
        LD   (OFF2+1),A         ;    "
OFF1:   LD   HL,(TBYL)          ;Hole Wert aus
OFF2:   LD   A,(TBYH)           ;Tabelle
        LD   H,A                ;    "
        LD   A,&03              ;Bit-Nummer aus-
        AND  C                  ;blenden
        LD   (OFF3+1),A         ;Pointer auf Bitmaske
        SRL  C                  ;X=X/4 (MODE 1)
        SRL  C                  ;    "
        ADD  HL,BC              ;    "
;
OFF3:   LD   A,(BITNR)          ;Hole Bitmaske
        LD   C,A                ;    "
        AND  (HL)               ;Hintergrundmaske
                                ;erstellen
        LD   B,A                ;und sichern
        LD   A,C                ;Bitmaske holen
        CPL                     ;Negativ eliminieren
FARBE:  AND  &FF                ;Colorieren
        OR   B                  ;und mit Hintergrund-
                                ;maske verknüpfen
        LD   (HL),A             ;Punkt ins Video-RAM
                                ;schreiben
;
        RET                     ;Fertig!
```

Analog zu unserer PLOT-Funktion können wir eine tabellenorientierte TEST-Funktion entwerfen. Sie verlangt wieder im BC-

Register die X- (0 bis 319) und im Akkumulator die Y-Testkoordinaten (0 bis 199). Nach der Programmausführung steht entweder

&00, &0F, &F0 oder &FF

im Akkumulator. Diese Werte geben die aktuelle Punktfarbe der adressierten Koordinate (INK 0 bis 3).

```
; ##################
; ##              ##
; ## TEST-FUNKTION ##
; ##              ##
; ##################
;
; Eingabeparameter: BC = X-Pos (0-319)
;                 :  A = Y-Pos (0-199)
;
; Ausgabeparameter:A   = Punktfarbe
;                        (&00, &0F, &F0, &FF)
;
; zerstörte Register:    AF, BC, HL
;
;
        ORG  &6200
;
BITNR:  DEFB %01110111          ;negativ-Masken
        DEFB %10111011          ;     "
        DEFB %11011101          ;     "
        DEFB %11101110          ;     "
;
;
;       ORG  &7000              ;Start Adresse
;
TBYH:   EQU  &6000              ;Y-Low-Tabelle
TBYL:   EQU  &6100              ;Y-High-Tabelle
BITNR:  EQU  &6200              ;Bitmasken
```



```
;
TEST:   LD   (OFF1+1),A         ;Übergebe Y
        LD   (OFF2+1),A         ;     "
OFF1:   LD   HL,(TBYL)          ;Hole Wert aus
OFF2:   LD   A,(TBYH)           ;Tabelle
        LD   H,A                ;     "
        LD   A;&03              ;Bit-Nummer aus-
        AND  C                  ;blenden
        LD   (OFF3+1),A         ;Pointer auf Bitmaske
        SRL  C                  ;X=X/4 (MODE 1)
        SRL  C                  ;     "
        ADD  HL,BC              ;     "
;
;OFF3:  LD   A,(BITNR)          ;Hole Bitmaske
        CPL                     ;Blende überflüssige
        AND  (HL)               ;Bits aus
;
        CP   &10                ;Bit im MSN gesetzt?
        JR   C,NXT              ;Wenn nein, dann prü-
                                ;fe nächstes Nibble
        LD   B,&F0              ;Zwischenergebnis
                                ;merken
NXT:    AND  &0F                ;Bit im LSN gesetzt?
        JR   Z,NXT2             ;Wenn nein, dann
                                ;hole Ergebnis nach A
        LD   A,&0F              ;Zwischenergebnis
                                ;merken
        OR   B                  ;Punktfarbe errechnen
        LD   B,A                ;und retten
NXT2:   LD   A,B                ;Punktfarbe im Akku
;
        RET                     ;Fertig!
```

Wie Sie sehen, ist das Prinzip der tabellenorientierten Programmierung immer das gleiche. Da derartige Funktionen durch ihre Tabellen immer speicheraufwendig sind, sollten sie in Ihren Programmen nur dann Verwendung finden, wenn ein Höchtmaß an

Geschwindigkeit gefordert ist. In allen anderen Fällen kann getrost auf die ROM-Routinen des Schneider-CPC zurückgegriffen werden.

9.5 Die wichtigsten ROM-Routinen

An dieser Stelle wollen wir Ihnen einige wichtige Grafik-Routinen des Betriebssystems vorstellen. Die Routinen werden über die Vektor-Tabelle im RAM angesprungen. Es ist wichtig, daß die Routinen nicht direkt angesprungen werden, da ansonsten die Software-Kompatibilität zwischen dem 464, 664 und 6128 nicht mehr gegeben ist, des weiteren und Systemabstürze auftreten können.

Die hier vorgestellten Routinen sind in 6 Sektionen aufgeteilt:

1. Routinenname und Einspung-Vektor
2. Einsprung-Adressen aller CPCs
3. Kurzbeschreibung der Funktion
4. Eingangsbedingungen
5. Ausgangsbedingungen
6. Komplette Funktionsbeschreibung

Alle Grafik-Routinen sind Bestandteil des Graphics-, Text- oder Screen-Packs des Schneider Betriebssystems (ein darauf Bezug nehmendes, komplett dokumentiertes ROM-Listing finden Sie übrigens im Anschluß diese Routinenbeschreibungen). Der Übersicht halber werden die ROM-Routinen erst einmal grob den einzelnen Packs zugeordnet. Innerhalb eines solchen Packs erfolgt dann eine weitere, nach aufsteigenden Adressen sortierte Unterteilung. Obwohl sich dieses Verfahren auf den ersten Blick vielleicht etwas kompliziert anhört, gewinnt es doch mit zunehmenden Gebrauch an Tranzparenz, da Querverweise zum ROM-Listing ohne Probleme gebildet werden können.

Doch zuerst eine vollständige Liste aller aufgeführten ROM-Routinen in alphabetischer Reihenfolge:

ASK CURSOR	(GRA)	&BBC6
CLEAR WINDOW	(GRA)	&BBDB
CONVERT POS	(GRA)	&BD4F
DOT POSITION	(SCR)	&BC1D
EXTENDED INITIALISE	(GRA)	&BD43
FILL	(GRA)	&BD52
FIRST POINT	(GRA)	&BD49
GET CURSOR	(TXT)	&BB78
GET ORIGIN	(GRA)	&BBCC
GET PAPER	(GRA)	&BBE7
GET PEN	(GRA)	&BBE1
GET WINDOW	(TXT)	&BB69
GET W HEIGHT	(GRA)	&BBD8
GET W WIDTH	(GRA)	&BBD5
HORIZONTAL	(SCR)	&BC5F
INITIALISE	(GRA)	&BBBA
INITIALISE	(TXT)	&BB4E
LINE ABSOLUTE	(GRA)	&BBF6
LINE RELATIVE	(GRA)	&BBF9
MOVE ABSOLUT	(GRA)	&BBC0

MOVE RELATIVE	(GRA)	&BBC3
OUTPUT	(TXT)	&BB5A
PIXELS	(SCR)	&BC5C
PLOT ABSOLUTE	(GRA)	&BBEA
PLOT RELATIVE	(GRA)	&BBED
RESET	(GRA)	&BBBD
RESET	(TXT)	&BB51
SET CURSOR	(TXT)	&BB75
SET GRAPHIC	(TXT)	&BB63
SET MASK	(GRA)	&BD4C
SET ORIGIN	(GRA)	&BBC9
SET PAPER	(GRA)	&BBE4
SET PEN	(GRA)	&BBDE
TEST ABSOLUTE	(GRA)	&BBF0
TEST RELATIVE	(GRA)	&BBF3
TRANS SWITCH	(GRA)	&BD46
VERTICAL	(SCR)	&BC62
WIN ENABLE	(TXT)	&BB66
WIN HIGHT	(GRA)	&BBD2
WIN WIDTH	(GRA)	&BBCF
WR CHAR	(GRA)	&BBFC

TXT INITIALISE **&BB4E**

ROM-Adressen:

464	&1078
664	&1070
6128	&1074

Funktion: TXT-Pack initialisieren

Eingaberegister: keine

Ausgaberegister: keine

Zerstörte Register: AF, BC, DE, HL

Funktionsbeschreibung:

TXT INITIALISE ist für die Initialisierung des TXT-Packs verantwortlich. Dabei werden die PAPER- und PEN-Farben auf ihre Anfangswerte gebracht. Zusätzlich werden alle Indirections in den RAM-Bereich kopiert und die Parameterblöcke (14 Bytes) aller WINDOWS initialisiert. Da die Routine beim Kopiervorgang den LDIR-Blocktransfer-Befehl der Z80-CPU verwendet, werden alle Doppelregister zerstört.

TXT RESET **&BB51**

ROM-Adressen:

464	&1088
664	&1080
6128	&1084

Funktion: TXT-Pack zurücksetzen

Eingaberegister: keine

Ausgaberegister: keine

Zerstörte Register: AF, BC, DE, HL

Funktionsbeschreibung:

TXT RESET kopiert die gesamte Steuerzeichen-Sprungtabelle sowie die Indirections des TXT-Packs in den zugehörigen RAM-Bereich. Da die Routine beim Kopiervorgang den LDIR-Blocktransfer-Befehl der Z80-CPU verwendet, werden alle Doppelregister zerstört.

TXT OUTPUT **&BB5A**

ROM-Adressen:		
	464	&1400
	664	&13FA
	6128	&13FE

Funktion: Zeichen auf dem Bildschirm ausgeben

Eingaberegister: A = ASCII-Code des Zeichens

Ausgaberegister: keine

Zerstörte Register: keine

Funktionsbeschreibung:

TXT OUTPUT bringt das im Akkumulator befindliche Zeichen auf das aktuelle Bildschirmfenster oder führt es aus, falls es sich um ein Steuerzeichen handelt. Beachten Sie, daß diese Funktion das Indirection TXT OUT ACTION benutzt. Sollten Sie diese manipuliert haben, wird TXT OUTPUT auch Ihre Routine statt der im ROM stehenden benutzen.

TXT SET GRAPHIC **&BB63**

ROM-Adressen:		
	464	&13A7
	664	&13A4
	6128	&13A8

Funktion: Textausgabe im Grafik-Window aktivieren oder deaktivieren

Eingaberegister: A = 0 - deaktiviert, <> 0 - aktiviert

Ausgaberegister: keine

Zerstörte Register: AF

Funktionsbeschreibung:

TXT SET GRAPHIC ermöglicht die Textausgabe im Grafik-Window. Wird beim Funktionsaufruf keine Null im Akkumulator übergeben, erscheint der Text von diesem Zeitpunkt ab an der aktuellen Grafikcursor-Position.

TXT WIN ENABLE **&BB66**

ROM-Adressen:		
	464	&120C
	664	&1204
	6128	&1208

Funktion: Größe des aktuellen Textfensters definieren.

Eingaberegister:
D = rechte Fenstergrenze
E = untere Fenstergrenze
H = linke Fenstergrenze
L = obere Fenstergrenze

Ausgaberegister: keine

Zerstörte Register: AF, BC, DE, HL

Funktionsbeschreibung:

TXT WIN ENABLE legt die Größe des aktuellen Textfensters fest. Vor dem Aufruf der Routine werden die Registerpaare HL und DE mit den Werten für die Fenstergröße versorgt.

TXT GET WINDOW **&BB69**

ROM-Adressen:		
	464	&1256
	664	&124E
	6128	&1252

Funktion: Größe des aktuellen Textfensters holen.

Eingaberegister: keine

Ausgaberegister:
D = rechte Fenstergrenze
E = untere Fenstergrenze
H = linke Fenstergrenze
L = obere Fenstergrenze

Zerstörte Register: AF

Funktionsbeschreibung:

TXT GET WINDOW gibt Anwort auf die Frage nach der Größe des aktuellen Textfensters. Beim Verlassen der Routine enthalten die Registerpaare HL und DE die Werte für die Fenstergröße.

TXT SET CURSOR **&BB75**

ROM-Adressen:		
	464	&1174
	664	&116C
	6128	&1170

Funktion: Cursor positionieren

Eingaberegister:
H = Spalte
L = Zeile

Ausgaberegister: keine

Zerstörte Register: AF, HL

Funktionsbeschreibung:

TXT SET CURSOR positioniert den Cursor im aktuellen Text-Window. Vor dem Aufruf der Routine muß H mit dem Wert für die Spalte und L mit dem Wert für die Zeile geladen werden.

TXT GET CURSOR **&BB78**

ROM-Adressen:		
	464	&1180
	664	&1178
	6128	&117C

Funktion: Ermitteln der aktuellen Cursor-Position

Eingaberegister: keine

Ausgaberegister:
A = lfd. ROLL-COUNT
H = Spalte
L = Zeile

Zerstörte Register: keine

Funktionsbeschreibung:

TXT GET CURSOR ermittelt die aktuelle Cursor-Position. Beim Verlassen der Routine enthält das H-Register den Wert für die Spalte und L den Wert für die Zeile. Darüber hinaus enthält der Akkumulator den Scolling-Zähler.

SCR DOT POSITION **&BC1D**

ROM-Adressen:		
	464	&0BA9
	664	&0BAB
	6128	&0BAF

Funktion: Pixelmaske testen

Eingaberegister:
DE = X-Koordinate (MODE-bezogen)
HL = Y-Koordinate (0 bis 199)

Ausgaberegister:
B = Anzahl der Pixel
C = Pixelmaske
HL = Adresse im Bildschirmspeicher

Zerstörte Register: AF, DE

Funktionsbeschreibung:

SCR DOT POSITION ist eine Routine, die wichtige Informationen über ein Byte im Bildschirmspeicher liefert. Diese Informationen dienen dem maskenorientierten Bildschirmaufbau und können verhindern, daß der Hintergrund ungewollt manipuliert wird. SCR DOT POSITION arbeitet MODE-orientiert und verlangt aus diesem Grund die X-Koordinate in den Mode-spezifischen Bereichen [MODE 0 (0-159), MODE 1 (0-319), MODE 2 (0-639)].

SCR PIXELS (Force Mode) **&BC5C**

ROM-Adressen:		
	464	&0C6B
	664	&0C70
	6128	&0C74

Funktion: Pixel setzen

Eingaberegister:
B = Farbnummer
C = Pixelmaske
HL = Adresse im Bildschirmspeicher

Ausgaberegister: keine

Zerstörte Register: AF

Funktionsbeschreibung:

SCR PIXEL setzt ein oder mehrere Punkte der im B-Register angegebenen Farbe in den Bildschirmspeicher. Der aktivierte Schreib-Modus wird bei diesem Vorgang ignoriert.

SCR HORIZONTAL **&BC5F**

ROM-Adressen:		
	464	&0FC4
	664	&0F8F
	6128	&0F93

Funktion: Horizontale Linie zeichnen

Eingaberegister:
A = Farbnummer
BC = X-Endposition (MODE-bezogen)
DE = X-Startposition (MODE-bezogen)
HL = Y-Position (0 bis 199)

Ausgaberegister: keine

Zerstörte Register: AF, BC, DE, HL

Funktionsbeschreibung:

SCR HORIZONTAL zieht mit hoher Geschwindigkeit eine waagerechte Linie (von DE nach BC) auf dem Bildschirm. Die Endkoordinate in BC muß dabei immer größer oder gleich der Startkoordinate in DE sein. Alle X-Parameter sind MODE-abhängig [MODE 0 (0-159), MODE 1 (0-319), MODE 2 (0-639)].

SCR VERTICAL **&BC62**

ROM-Adressen:

464	&102F
664	&0F97
6128	&0F9B

Funktion: Vertikale Linie zeichnen

Eingaberegister:
A = Farbnummer
BC = Y-Endposition (0 bis 199)
DE = X-Position (MODE-bezogen)
HL = Y-Startposition (0 bis 199)

Ausgaberegister: keine

Zerstörte Register: AF, BC, DE, HL

Funktionsbeschreibung:

SCR VERTICAL zieht mit hoher Geschwindigkeit eine senkrechte Linie (von HL nach BC) auf dem Bildschirm. Die Endkoordinate in BC muß dabei immer größer oder gleich der Startkoordinate in HL sein. Alle X-Parameter sind MODE-abhängig [MODE 0 (0-159), MODE 1 (0-319), MODE 2 (0-639)].

GRA INITIALISE **&BBBA**

ROM-Adressen:

464	&15B0
664	&15A4
6128	&15A8

Funktion: Initialisierung des GRA-Packs

Eingaberegister: keine

Ausgaberegister: keine

Zerstörte Register: AF, BC, DE, HL

Funktionsbeschreibung:

Als erstes werden die Indirections GRA PLOT, GRA TEST und GRA LINE in den RAM-Bereich kopiert. Danach werden Hintergrundfarbe (PAPER) und Stiftfarbe (PEN) auf ihre Standardwerte gebracht. ORIGIN auf linke untere Ecke des Bildschirms bringen (0,0). Grafik-Cursor auf 0,0 positionieren. WINDOW(0) definieren und dessen Inhalt nicht zerstören.

GRA RESET **&BBBD**

ROM-Adressen:

464	&15DF
664	&15D3
6128	&15D7

Funktion: Kopieren aller Indirections

Eingaberegister: keine

Ausgaberegister: keine

Zerstörte Register: AF, BC, DE, HL

Funktionsbeschreibung:

Die Indirections GRA PLOT, GRA TEST und GRA LINE werden in den RAM-Bereich kopiert.

GRA MOVE ABSOLUTE **&BBC0**

ROM-Adressen:		
	464	&15F4
	664	&15FA
	6128	&15FE

Funktion: Absolute Positionierung des Grafik-Cursors

Eingaberegister: DE = Cursorposition (Spalten)
HL = Cursorposition (Zeilen)

Ausgaberegister: keine

Zerstörte Register: AF, BC, DE, HL

Funktionsbeschreibung:

Der Grafik-Cursor wird an die von DE und HL angegebene absolute Bildschirmposition gesetzt.

GRA MOVE RELATIVE **&BBC3**

ROM-Adressen:		
	464	&15F1
	664	&15F7
	6128	&15FB

Funktion: Relative Positionierung des Grafik-Cursors bezogen auf aktuelle Position

Eingaberegister: DE = Cursorposition (Spalten)
HL = Cursorposition (Zeilen)

Ausgaberegister: keine

Zerstörte Register: AF, BC, DE, HL

Funktionsbeschreibung:

Der Grafik-Cursor wird an die von DE und HL angegebene relative Bildschirmposition gesetzt. Diese Position bezieht sich auf die aktuellen Koordinaten des Grafik-Cursors. Die entstehenden Koordinaten können auch außerhalb des definierten Grafik-Windows liegen.

GRA ASK CURSOR **&BBC6**

ROM-Adressen:		
	464	&15FC
	664	&1602
	6128	&1606

Funktion: Ermittelt die gegenwärtige Position des Grafik-Cursors.

Eingaberegister: keine

Ausgaberegister: DE = Cursorposition (Spalte)
HL = Cursorposition (Zeile)

Zerstörte Register: keine

Funktionsbeschreibung:

Ermittelt die gegenwärtige Position des Grafik-Cursors.

GRA SET ORIGIN **&BBC9**

ROM-Adressen:	464	&1604
	664	&160A
	6128	&160E

Funktion: Koordinatenursprung definieren

Eingaberegister: DE = X-Koordinate
HL = Y-Koordinate

Ausgaberegister: keine

Zerstörte Register: DE, HL

Funktionsbeschreibung:

Der vom Anwender definierte Koordinatenursprung bezieht sich immer auf die linke untere Ecke des Bildschirms.

GRA GET ORIGIN **&BBCC**

ROM-Adressen:	464	&1612
	664	&1618
	6128	&161C

Funktion: Koordinatenursprung ermitteln

Eingaberegister: keine

Ausgaberegister: DE = X-Koordinate
HL = Y-Koordinate

Zerstörte Register: keine

Funktionsbeschreibung:

Die Routine ermittelt den Koordinatenursprung, analog zu GRA SET ORIGIN, bezogen auf die linke untere Ecke des Bildschirms.

GRA WIN WIDTH **&BBCF**

ROM-Adressen:	464	&1734
	664	&16A1
	6128	&16A5

Funktion: Linke und rechte Begrenzung des Grafik-Fensters setzen.

Eingaberegister: DE = linker Rand des Windows
HL = rechter Rand des Windows

Ausgaberegister: keine

Zerstörte Register: AF, DE, HL

Funktionsbeschreibung:

GRA WIN WIDTH testet als erstes, welche der in DE und HL übergebenen Koordinaten größer ist. Falls die in HL befindliche Koordinate kleiner ist, werden die Registerinhalte vertauscht. Danach werden sie gerundet, damit immer ein ganzes Byte in

dem vom Anwender definierten Fenster liegt. Je nach Bildschirmmodus (MODE 0 bis 2) werden folgende Manipulationen an den Koordinaten durchgeführt:

MODE 2: Die Koordinaten werden nicht manipuliert.
MODE 1: Alle Koordinaten werden durch 2 dividiert.
MODE 0: Alle Koordinaten werden durch 4 dividiert.

GRA WIN HEIGHT **&BBD2**

ROM-Adressen:		
	464	&1779
	664	&16E6
	6128	&16EA

Funktion: Obere und untere Begrenzung des Grafik-Fensters setzen.

Eingaberegister: DE = oberer Rand des Windows
HL = unterer Rand des Windows

Ausgaberegister: keine

Zerstörte Register: AF, DE, HL

Funktionsbeschreibung:

GRA WIN HEIGHT testet als erstes, welche der in DE und HL übergebenen Koordinaten größer ist. Falls die in DE befindliche Koordinate kleiner ist, werden die Registerinhalte vertauscht.

GRA GET W WIDTH **&BBD5**

ROM-Adressen:		
	464	&17A6
	664	&1713
	6128	&1717

Funktion: Linke und rechte Begrenzung des Grafik-Fensters ermitteln.

Eingaberegister: keine

Ausgaberegister: DE = linker Rand des Windows
HL = rechter Rand des Windows

Zerstörte Register: AF

Funktionsbeschreibung:

GRA GET W WIDTH ermittelt die rechte und linke Fenstergrenze. Vor der Ausgabe werden die Koordinaten an den gegenwärtigen Bildschirm-Modus wie folgt angepaßt:

MODE 2: Die Koordinaten werden nicht manipuliert.
MODE 1: Alle Koordinaten werden mit 2 multipliziert.
MODE 0: Alle Koordinaten werden mit 4 multipliziert.

GRA GET W HEIGHT **&BBD8**

ROM-Adressen:		
	464	&17BC
	664	&1729
	6128	&172D

Funktion: Obere und untere Begrenzung des Grafik-Fensters ermitteln.

Eingaberegister: keine

Ausgaberegister: DE = oberer Rand des Windows
HL = unterer Rand des Windows

Zerstörte Register: AF

Funktionsbeschreibung:

GRA GET W HEIGHT ermittelt die obere und untere Fenstergrenze. Die Funktion steht im direkten Zusammenhang mit GRA GET W WIDTH.

GRA CLEAR WINDOW **&BBDB**

ROM-Adressen:		
	464	&17C5
	664	&1732
	6128	&1736

Funktion: Löschen eines Grafikfensters.

Eingaberegister: keine

Ausgaberegister: keine

Zerstörte Register: AF, BC, DE, HL

Funktionsbeschreibung:

GRA CLEAR WINDOW löscht das Grafikfenster und positioniert den Grafik-Cursor auf den Ursprung des Koordinatensystems.

GRA SET PEN **&BBDE**

ROM-Adressen:		
	464	&17F6
	664	&1763
	6128	&1767

Funktion: Definieren der Farbe eines Grafikstiftes.

Eingaberegister: A = Stiftfarbe

Ausgaberegister: keine

Zerstörte Register: AF

Funktionsbeschreibung:

Der in A angegebene Farbwert wird automatisch dem aktuellen MODE angepaßt. Danach erfolgt die Stiftfarben-Definition.

GRA GET PEN **&BBE1**

ROM-Adressen:		
	464	&1804
	664	&1771
	6128	&1775

Funktion: Farbe eines Grafikstiftes holen

Eingaberegister: keine

Ausgaberegister: A = Stiftfarbe

Zerstörte Register: AF

Funktionsbeschreibung:

GRA GET PEN lädt den Akkumulator mit der Farbe des aktuellen Grafikstiftes. Bei diesem Vorgang werden alle Flags zerstört.

GRA SET PAPER **&BBE4**

ROM-Adressen:		
	464	&17FD
	664	&176A
	6128	&176E

Funktion: Hintergrundfarbe (PAPER) definieren.

Eingaberegister: A = Hintergrundfarbe

Ausgaberegister: keine

Zerstörte Register: AF

Funktionsbeschreibung:

Mittels GRA SET PAPER wird der Hintergrundfarbe (PAPER) der im Akkumulator befindliche Farbwert zugewiesen. Bei diesem Vorgang werden alle Flags zerstört.

GRA GET PAPER **&BBE7**

ROM-Adressen:		
	464	&180A
	664	&1776
	6128	&177A

Funktion: Hintergrundfarbe (PAPER) auslesen.

Eingaberegister: keine

Ausgaberegister: A = Hintergrundfarbe

Zerstörte Register: AF

Funktionsbeschreibung:

Mittels GRA GET PAPER wird der Akkumulator mit der aktuellen Hintergrundfarbe (PAPER) geladen. Bei diesem Vorgang werden alle Flags zerstört.

GRA PLOT ABSOLUTE **&BBEA**

ROM-Adressen:		
	464	&1813
	664	&177F
	6128	&1783

Funktion: Punkt an einer absoluten Koordinate im aktuellen Window ausgeben.

Eingaberegister: DE = X-Position
HL = Y-Position

Ausgaberegister: keine

Zerstörte Register: AF, BC, DE, HL

Funktionsbeschreibung:

GRA PLOT ABSOLUTE ist für die Ausgabe eines Punktes im dem vom Benutzer definierten Koordinatensystems zuständig. Falls eine Koordinate außerhalb des aktuellen Windows liegt, wird die Funktion zwar ausgeführt - die Bildschirmausgabe hingegen ignoriert.

GRA PLOT RELATIVE **&BBED**

ROM-Adressen:		
	464	&1810
	664	&177C
	6128	&1780

Funktion: Punkt an einer relativen Koordinate im aktuellen Window ausgeben.

Eingaberegister: DE = relativer X-Offset
HL = relativer Y-Offset

Ausgaberegister: keine

Zerstörte Register: AF, BC, DE, HL

Funktionsbeschreibung:

GRA PLOT RELATIVE ist für die Ausgabe eines Punktes in dem vom Benutzer definierten Koordinatensystems zuständig. Falls eine relative Koordinate außerhalb des aktuellen Windows liegt, wird die Funktion zwar ausgeführt - die Bildschirmausgabe hingegen ignoriert.

GRA TEST ABSOLUTE **&BBF0**

ROM-Adressen:

464	&1827
664	&1793
6128	&1797

Funktion: Farbe eines Punktes an einer absoluten Koordinate im aktuellen Window auslesen.

Eingaberegister: DE = X-Position
HL = Y-Position

Ausgaberegister: A = Farbnummer des getesteten Punktes

Zerstörte Register: BC, DE, HL

Funktionsbeschreibung:

GRA PLOT ABSOLUTE testet die Farbe eines Punktes im dem vom Benutzer definierten Koordinatensystems und übergibt dessen Farbwert an den Akkumulator. Falls eine Koordinate außerhalb des aktuellen Windows liegt, wird diese Funktion ignoriert.

GRA TEST RELATIVE **&BBF3**

ROM-Adressen:

464	&1824
664	&1790
6128	&1794

Funktion: Farbe eines Punktes an einer relativen Koordinate im aktuellen Window auslesen.

Eingaberegister: DE = relativer X-Offset
HL = relativer Y-Offset

Ausgaberegister: A = Farbnummer des getesteten Punktes

Zerstörte Register: BC, DE, HL

Funktionsbeschreibung:

GRA PLOT RELATIVE testet die Farbe eines Punktes in dem vom Benutzer definierten Koordinatensystems und übergibt dessen Farbwert an den Akkumulator. Falls eine relative Koordinate außerhalb des aktuellen Windows liegt, wird diese Funktion ignoriert.

GRA LINE ABSOLUTE **&BBF6**

ROM-Adressen:	464	&1839
	664	&17A5
	6128	&17A9

Funktion: Zieht eine Linie von der aktuellen zur angegebenen absoluten Position im definierten Window.

Eingaberegister: DE = X-Position
HL = Y-Position

Ausgaberegister: keine

Zerstörte Register: AF, BC, DE, HL

Funktionsbeschreibung:

GRA LINE ABSOLUTE zieht eine Linie von der aktuellen Position des Grafikcursors zur absoluten in DE und HL angegebenen Position im definierten Koordinatensystem. Die Funktion wird auch ausgeführt, wenn eine Koordinate außerhalb des aktuellen Windows liegt. Die Linie ist jedoch nur im definierten Window sichtbar. Zum Zeichnen der Linie wird der aktuelle Grafikstift verwendet.

GRA LINE RELATIVE **&BBF9**

ROM-Adressen:	464	&1836
	664	&17A2
	6128	&17A6

Funktion: Zieht eine Linie von der aktuellen zur angegebenen relativ zum Grafikcursor befindlichen Position im definierten Window.

Eingaberegister: DE = relativer X-Offset
HL = relativer Y-Offset

Ausgaberegister: keine

Zerstörte Register: AF, BC, DE, HL

Funktionsbeschreibung:

GRA LINE RELATIVE zieht eine Linie von der aktuellen Position des Grafikcursors zur relativ von ihm in DE und HL angegebenen Position im definierten Koordinatensystem. Die Funktion wird auch ausgeführt, wenn eine relative Koordinate außerhalb des aktuellen Windows liegt. Die Linie ist jedoch nur im definierten Window sichtbar. Zum Zeichnen der Linie wird der aktuelle Grafikstift verwendet.

GRA WR CHAR **&BBFC**

ROM-Adressen:	464	&1945
	664	&193C
	6128	&1940

Funktion: Gibt ein Zeichen auf dem Bildschirm aus

Eingaberegister: A = Zeichen (ASCII-Code)

Ausgaberegister: keine

Zerstörte Register: AF, BC, DE, HL

Funktionsbeschreibung:

GRA WR CHAR gibt das Zeichen, dessen ASCII-Code sich im Akkumulator befindet, ab der linken oberen Ecke an der Posi-

tion des Grafik-Cursors aus. Danach wird der Grafik-Cursor um die Breite des Zeichens nach rechts verschoben. Diese Verschiebung ist MODE-Abhängig und kann 8 (MODE 2), 16 (MODE 1) oder 32 (MODE 0) Punkte betragen. Die Zeichenfarbe entspricht der aktuellen Stiftfarbe. Die Hintergrundfarbe, die mit ausgegeben wird und den Hintergrund überschreibt, ist mit der PAPER-Farbe identisch. Falls sich die Position des Grafikcursors außerhalb des definierten Windows befinden sollte, wird GRA WR CHAR ignoriert.

GRA EXTENDED INITIALISE **&BD43**

ROM-Adressen:	464	-
	664	&15E8
	6128	&15EC

Funktion: Zusätzliche Grafikfunktionen initialisieren.

Eingaberegister: keine

Ausgaberegister: keine

Zerstörte Register: AF, HL

Funktionsbeschreibung:

GRA EXTENDED INITIALISE initialisiert äquivalent zu GRA INITIALISE alle Zusatzfunktionen des CPC 664 und 6128. Diese Funktion ist nicht auf dem CPC 464 verfügbar.

GRA TRANS SWITCH **&BD46**

ROM-Adressen:	464	-
	664	&19D1
	6128	&19D5

Funktion: Ein- oder Ausschalten des Transparent-Modus im Grafik-Modus.

Eingaberegister: A = Schalter

Ausgaberegister: keine

Zerstörte Register: keine

Funktionsbeschreibung:

GRA TRANS SWITCH schaltet den Transparentmodus bei der Zeichenausgabe im Grafik-Modus an oder aus. Falls der Hintergrund mit den Zeichen verknüpft werden soll, muß im Akkumulator eine 0 übergeben werden. Wenn nicht, muß der Inhalt des Akkus ungleich Null sein. Diese Funktion ist nicht auf dem CPC 464 verfügbar.

GRA FIRST POINT **&BD49**

ROM-Adressen:	464	-
	664	&17B0
	6128	&17AC

Funktion: Selektiert, ob der erste Punkt einer Linie geplottet werden soll.

Eingaberegister: A = Schalter

Ausgaberegister: keine

Zerstörte Register: keine

Funktionsbeschreibung:

GRA FIRST POINT selektiert, ob der erste Punkt einer Linie geplottet werden soll oder nicht. Falls der erste Punkt nicht gesetzt werden soll, muß im Akkumulator beim Funktionsaufruf eine 0 übergeben werden. Wenn nicht, muß der Inhalt des Akkus ungleich Null sein. Diese Funktion ist nicht auf dem CPC 464 verfügbar.

GRA SET MASK **&BD4C**

ROM-Adressen:	464	-
	664	&17A8
	6128	&17AC

Funktion: Maske für LINE-Funktion definieren

Eingaberegister: A = Maske

Ausgaberegister: keine

Zerstörte Register: keine

Funktionsbeschreibung:

Mittels GRA SET MASK kann das äußere Erscheinungsbild einer Linie definiert werden. Dazu muß eine 8-Bit-Maske beim Funktionsaufruf in dem Akkumulator übergeben werden. Jedes gesetzte Bit entspricht dabei einem Punkt auf dem Bildschirm. Hier einige Beispiele:

Striche:	%11110000 (&F0)
Punkte:	%10101010 (&AA)
Strich-Punkt:	%11100100 (&E4)
Voll-Linie:	%11111111 (&FF)

Diese Funktion ist nicht auf dem CPC 464 verfügbar.

GRA CONVERT POS **&BD4F**

ROM-Adressen:	464	-
	664	&1626
	6128	&162A

Funktion: Absolute Koordinaten in physikalische Koordinaten konvertieren.

Eingaberegister: DE = absolute X-Koordinate (0 bis 639)
HL = absolute Y-Koordinate (0 bis 399)

Ausgaberegister: DE = phys. X-Koordinate (MODE)
HL = phys. Y-Koordinate (0-199)

Zerstörte Register: AF

Funktionsbeschreibung:

GRA CONVERT POS wandelt die in DE und HL angegebenen absoluten Koordinaten (auf den Ursprung bezogen) in physikalische Koordinaten (MODE-bezogen) um. Das heißt, daß der Y-Wert immer durch 2 und der X-Wert MODE-abhängig durch 4, 2 oder 1 dividiert wird [MODE 0 (0-159), MODE 1 (0-319), MODE 2 (0-639)]. Diese Funktion ist nicht auf dem CPC 464 verfügbar.

GRA FILL **&BD52**

ROM-Adressen:	464	-
	664	&19D5
	6128	&19D9

Funktion: Beliebige Fläche mit Farbe füllen.

Eingaberegister:
A = Füllfarbe
DE = maximale FILL-Puffer Größe
HL = FILL-Puffer Startadresse

Ausgaberegister: keine

Zerstörte Register: AF, BC, DE, HL

Funktionsbeschreibung:

GRA FILL füllt eine umrahmte Fläche beliebiger Struktur ab der Grafikcursor-Position mit der im Akkumulator angegebenen Farbe aus. Dabei muß die Farbe der Umrahmung mit der aktuellen Stiftfarbe sowie der Füllfarbe übereinstimmen. Da GRA FILL rekursiv arbeitet, benötigt sie einen Puffer, dessen Größe und Position in den Doppelregistern DE und HL angegeben werden muß. Falls der definierte Puffer zu klein sein sollte, wird der Füllvorgang automatisch abgebrochen. Diese Funktion ist nicht auf dem CPC 464 verfügbar.

9.6 ROM-Listing

Der in diesem Teil dokumentierte ROM-Auszug bezieht sich auf GRA-, TXT-, und SCR-Pack des Schneider-ROMs und gestattet eine einfachere Maschinenprogrammierung. Villeicht ist Ihnen bekannt, daß die Unterschiede im ROM zwischen den Schneider Rechnern minimal sind. Deshalb haben wir uns entschieden - um keine Seitenschinderei zu betreiben - bei der Betrachtung des Betriebssystems unser besonderes Augenmerk auf den CPC 6128 zu richten. Das bedeutet nicht, daß alle anderen CPC-Besitzer jetzt das Buch schließen, sondern lediglich, daß diese ein wenig aufpassen müssen, ob die abgedruckten Kommentare unmittelbar zu Ihrem ROM-Listing passen oder ob sich die Adressen um ein paar Bytes verschoben haben. Da Sie ja soweit in der Maschinensprache zuhause sind, daß Sie es verstehen mit dem Betriebssystem Ihre Experimente anzustellen, dürfte Ihnen diese Transferleistung nicht schwerfallen. Sind Sie aber noch nicht so sattelfest, sollten Sie ohnehin ausschließlich die Vektoren verwenden.

Leider das Abdrucken der Op-Codes und Mnemonics des Schneider-ROMs in dieser Form urheberrechtlich untersagt. So haben wir uns entschlossen, die Dokumentation auf Kommentare und Adressen zu beschränken. Diese Kommentare sind, betrachtet man sie isoliert, nicht sehr aussagekräftig; erzeugt man sich aber mit dem nachstehenden Disassembler ein Listing von dem zu untersuchenden ROM-Bereich und bringt dann die Adressen des Listings mit denen der Kommentare zur Deckung, so fügt sich beides zu einem sinnvollen Ganzen zusammen.

9.6.1 Z80-Disassembler für CPC

Der Zugriff auf dem ROM-Speicher stellt beim Schneider-CPC ein aus der BASIC-Ebene unlösbares Problem dar; Anwenderprogramme, die von Daten aus dem ROM leben, wie zum Beispiel ein Disassembler, lassen sich aber nur mit ungeheurem Aufwand vollständig in Maschinensprache realisieren. Deshalb

wollen wir Ihnen an dieser Stelle einen Z80-Disassembler[1] vorstellen, der zwar in BASIC geschrieben ist, aber mit einer kleinen Maschinenroutine das Auslesen des ROMs ermöglicht. Er soll zum einen den Zugriff auf das ROM verdeutlichen, zum anderen aber eine Utility zur Arbeit mit dem Schneider-CPC sein, die sich zur Untersuchung des Betriebssystems als unentbehrlich erweist.

Gleich nach dem Start des Disassemblers schreibt des Programm eine kleine Maschinenroutine ab Adresse &AB70 in den Speicher des CPC, die in Assemblerschreibweise folgendermaßen aussieht:

```
AB70  01 82 7F     LD   BC,&7F82    ;RAM-ROM-KONFIGURATION
AB73  ED 49        OUT  (C),C       ;UMSCHALTEN
AB75  1A           LD   A,(DE)      ;BYTE AUS ROM LESEN
AB76  32 7F AB     LD   (&AB7F),A   ;UND INS RAM SCHREIBEN
AB79  C9           RET              ;FERTIG!
```

Auf dieses Maschinenprogramm greift der Disassembler immer dann zu, wenn er aus dem ROM lesen will. Beim Aufruf der Routine mit CALL wird als Parameter die Adresse von der gelesen werden soll übergeben. Die Adresse wird vom BASIC in das DE-Register des Z80 geschrieben und steht dort für die Maschinenroutine zur Verfügung. Das aus dem ROM gelesene Byte schreibt die Routine ihrerseits an die Adresse &AB7F in den RAM-Speicher, wo es mit der BASIC-Funktion gelesen werden kann. Auf diesem Umweg sind die im ROM stehenden Informationen doch aus der BASIC-Ebene erreichbar.

```
100 'RAM-ROM-UMSCHALTUNG
101 '
102 DATA &01,&82,&7F,&ED,&49,&1A,&32,&7F,&AB,&C9
103 MEMORY &9FFF
104 FOR A=&AB70 TO &AB79
105 READ D
```

1 Das Maschinensprache-Buch zum CPC, DATA BECKER, 1984, Seite 247


```
106 POKE A,D
107 NEXT A
108 '
109 MODE 2
110 GOTO 227
111 LOCATE 18,4:PRINT"Z 8 0  -  D I S A S S E M B L E R"
112 LOCATE 5,7:INPUT"DRUCKER (J/N) ",E$
113 IF E$="J" THEN AUS=8 ELSE AUS=0
114 LOCATE 5,10:INPUT"STARTADRESSE : &",A$
115 GOSUB 213:ANFA=A
116 LOCATE 5,12:INPUT"ENDADRESSE   : &",A$
117 GOSUB 213:ENDE=A
118 IF ANFA>ENDE THEN 109
119 PC=ANFA
120 MODE 2
121 ADR=PC
122 PRINT#AUS,HEX$(ADR,4);"  ";
123 IFLAG=0
124 GOSUB 219
125 GOSUB 137
126 IF IFLAG THEN 178
127 IF (W=&CF OR W=&D7 OR W=&DF OR W=&EF) AND (LEFT$(PR$,3)="RST") THEN
PR$=PR$+" /DW:NN"
128 IF INSTR(PR$,"N")<>0 THEN 191
129 IF INSTR(PR$,"E")<>0 THEN 203
130 PO=INSTR(PR$," ")
131 IF PR$="" THEN PR$="???"
132 IF PO=0 THEN PRINT#AUS,TAB(21);PR$;:GOTO 134
133 PRINT#AUS,TAB(21);LEFT$(PR$,PO-1);TAB(27);RIGHT$(PR$,LEN(PR$)-PO);
134 PRINT#AUS
135 IF PC<=ENDE THEN 121
136 END
137 '
138 'INTERPRETIEREN
139 '
140 IF (W=&DD OR W=&FD) AND NOT IFLAG THEN 163
141 IF W=&ED THEN 158
142 IF W=&CB THEN 150
143 GOSUB 170
144 ON CO1 GOTO 146,148,145
```

```
145 PR$=BEF$(W):RETURN
146 IF W=&76 THEN PR$="HALT":RETURN
147 PR$="LD "+REGTAB$(CO2)+","+REG$:RETURN
148 IF CO2=0 OR CO2=1 OR CO2=3 THEN A$=" A," ELSE A$=" "
149 PR$=ARILOG$(CO2)+A$+REG$:RETURN
150 '
151 'CB
152 '
153 GOSUB 219
154 IF IFLAG THEN DIS=W:GOSUB 219
155 GOSUB 170
156 IF CO1=0 THEN PR$=ROTSCHI$(CO2)+" "+REG$ ELSE PR$=BITTI$(CO1)+STR$(C
O2)+","+REG$
157 RETURN
158 '
159 'ED
160 '
161 GOSUB 219
162 IF W<&40 OR W>&BF THEN PR$="???":RETURN ELSE GOTO 145
163 '
164 'XY
165 '
166 IFLAG=-1
167 IF W=&DD THEN I$="IX" ELSE I$="IY"
168 GOSUB 219
169 GOTO 137
170 '
171 'CODE ZERLEGEN
172 '
173 CO1=(W AND &X11000000)/64
174 CO2=(W AND &X111000)/8
175 CO3=W AND &X111
176 REG$=REGTAB$(CO3)
177 RETURN
178 '
179 'INDIZIERT
180 '
181 PO=INSTR(PR$,"HL")
182 IF PO=0 THEN PR$="???":GOTO 130
183 IF INSTR(PR$,"(HL)")<>0 THEN 187
```



```
184 IF PR$="EX DE,HL" THEN PR$="???":GOTO 130
185 IF PR$="ADD HL,HL" THEN PR$="ADD "+I$+","+I$:GOTO 130
186 PR$=LEFT$(PR$,PO-1)+I$+RIGHT$(PR$,LEN(PR$)-PO-1):GOTO 127
187 IF LEFT$(PR$,2)="JP" THEN 186
188 IF PC-ADR<3 THEN GOSUB 219:DIS=W
189 IF DIS>127 THEN DIS$=STR$(DIS-
256) ELSE DIS$="+"+RIGHT$(STR$(DIS),LEN(STR$(DIS))-1)
190 I$=I$+DIS$:GOTO 186
191 'N ERSETZEN
192 PO=INSTR(PR$,"NN")
193 IF PO<>0 THEN 198
194 PO=INSTR(PR$,"N")
195 GOSUB 219
196 PR$=LEFT$(PR$,PO-1)+"&"+HEX$(W,2)+RIGHT$(PR$,LEN(PR$)-PO)
197 GOTO 130
198 GOSUB 219:LB=W
199 GOSUB 219
200 WERT=W*256+LB
201 PR$=LEFT$(PR$,PO-1)+"&"+HEX$(WERT,4)+RIGHT$(PR$,LEN(PR$)-PO-1)
202 GOTO 130
203 '
204 'E ERSETZEN
205 '
206 PO=INSTR(PR$,"E")
207 GOSUB 219
208 IF W>127 THEN W=W-256:REM 2ER-KOMP.
209 W=W+2
210 A$="$"+STR$(W)+" >"+"&"+HEX$(PC+W-2,4)
211 PR$=LEFT$(PR$,PO-1)+A$+RIGHT$(PR$,LEN(PR$)-PO)
212 GOTO 130
213 '
214 'UMWANDLUNG HEX-DEZ
215 '
216 IF A$="" THEN A=0:RETURN
217 A=VAL("&"+A$)
218 RETURN
219 '
220 'BYTE AUS ROM LESEN
221 '
222 CALL &AB70,PC
```

```
223 W=PEEK(&AB7F)
224 PC=PC+1
225 PRINT#AUS,HEX$(W,2);" ";
226 RETURN
227 '
228 'INIT
229 '
230 DIM REGTAB$(7),ROTSCHI$(8),BITTI$(3),ARILOG$(7),BEF$(255)
231 FOR I=0 TO 7:READ REGTAB$(I):NEXT
232 FOR I=0 TO 7:READ ROTSCHI$(I):NEXT
233 FOR I=1 TO 3:READ BITTI$(I):NEXT
234 FOR I=0 TO 7:READ ARILOG$(I):NEXT
235 FOR I=0 TO &7F:READ BEF$(I):NEXT
236 FOR I=&80 TO &9F:BEF$(I)="":NEXT
237 FOR I=&A0 TO &FF:READ BEF$(I):NEXT
238 GOTO 111
239 '
240 'DATAS
241 '
242 DATA B,C,D,E,H,L,(HL),A
243 DATA RLC,RRC,RL,RR,SLA,SRA,???,SRL
244 DATA BIT,RES,SET
245 DATA ADD,ADC,SUB,SBC,AND,XOR,OR,CP
246 DATA NOP,"LD BC,NN","LD (BC),A",INC BC,INC B,DEC B,"LD B,N",RLCA
247 DATA "EX AF,AF'","ADD HL,BC","LD A,(BC)",DEC BC,INC C,DEC C,"LD C,N"
,RRCA
248 DATA DJNZ E,"LD DE,NN","LD (DE),A",INC DE,INC D,DEC D,"LD D,N",RLA
249 DATA JR E,"ADD HL,DE","LD A,(DE)",DEC DE,INC E,DEC E,"LD E,N",RRA
250 DATA "JR NZ,E","LD HL,NN","LD (NN),HL",INC HL,INC H,DEC H,"LD H,N",D
AA
251 DATA "JR Z,E","ADD HL,HL","LD HL,(NN)",DEC HL,INC L,DEC L,"LD L,N",C
PL
252 DATA "JR NC,E","LD SP,NN","LD (NN),A",INC SP,INC (HL),DEC (HL),"LD (
HL),N",SCF
253 DATA "JR C,E","ADD HL,SP","LD A,(NN)",DEC SP,INC A,DEC A,"LD A,N",CC
F
254 DATA "IN B,(C)","OUT (C),B","SBC HL,BC","LD (NN),BC",NEG,RETN,IM 0,"
LD I,A"
255 DATA "IN C,(C)","OUT (C),C","ADC HL,BC","LD BC,(NN)",,RETI,,"LD R,A"
256 DATA "IN D,(C)","OUT (C),D","SBC HL,DE","LD (NN),DE",,,IM 1,"LD A,I"
```



```
257 DATA "IN E,(C)","OUT (C),E","ADC HL,DE","LD DE,(NN)",,,IM 2,"LD A,R"
258 DATA "IN H,(C)","OUT (C),H","SBC HL,HL","LD (NN),HL",,,,RRD
259 DATA "IN L,(C)","OUT (C),L","ADC HL,HL","LD HL,(NN)",,,,RLD
260 DATA ,,"SBC HL,SP","LD (NN),SP",,,,
261 DATA "IN A,(C)","OUT (C),A","ADC HL,SP","LD SP,(NN)",,,,
262 DATA LDI,CPI,INI,OUTI,,,,,LDD,CPD,IND,OUTD,,,,
263 DATA LDIR,CPIR,INIR,OTIR,,,,,LDDR,CPDR,INDR,OTDR,,,,
264 DATA RET NZ,POP BC,"JP NZ,NN",JP NN,"CALL NZ,NN",PUSH BC,"ADD A,N",R
ST &00
265 DATA RET Z,RET,"JP Z,NN",->,"CALL Z,NN",CALL NN,"ADC A,N",RST &08
266 DATA RET NC,POP DE,"JP NC,NN","OUT (N),A","CALL NC,NN",PUSH DE,"SUB
N",RST &10
267 DATA RET C,EXX,"JP C,NN","IN A,(N)","CALL C,NN",->,"SBC A,N",RST &18
268 DATA RET PO,POP HL,"JP PO,NN","EX (SP),HL","CALL PO,NN",PUSH HL,"AND
 N",RST &20
269 DATA RET PE,JP (HL),"JP PE,NN","EX DE,HL","CALL PE,NN",-
>,"XOR N",RST &28
270 DATA RET P,POP AF,"JP P,NN",DI,"CALL P,NN",PUSH AF,"OR N",RST &30
271 DATA RET M,"LD SP,HL","JP M,NN",EI,"CALL M,NN",->,"CP N",RST &38
```

9.6.2 SCREEN PACK (SCR)

Das SCREEN PACK ist dem TEXT- und dem GRAPHICS PACK untergeordnet. Es ist praktisch die Exekutive für diese beiden Packs und damit für die unmittelbare Handhabung des Bildschirms zuständig.

```
0ABF ------------------------------------------------- SCR INITIALISE
```

Beim Aufruf dieser Routine erfolgt eine vollständige Initialisierung des Screen-Packs. Der SCREEN START wird auf &C000 gelegt.

```
0ABF                    ;BASISADRESSE DEFAULT FARBEN NACH DE
0AC2                    ;MC CLEAR INKS
0AC5                    ;ANFANGSWERT HIGH BYTE SCREEN START
0AC7                    ;(HIGH BYTE SCREEN START)
0ACA                    ;SCR RESET
0ACD                    ;BILDSCHIRM MODE 1 SETZEN


0AD0 ------------------------------------------------------ SCR RESET
```

Rücksetzen des Screen-Packs.

```
0AD0                    ;AKKU LÖSCHEN UND FLAGS ZURÜCKSETZEN
0AD1                    ;SCR ACCESS
0AD4                    ;RESTORE SCR INDIRECTIONS
0AD7                    ;MOVE (HL+3) NACH ((HL+1)), CNT=(HL)
0ADA                    ;RESET FARBEN
0ADD                    ;9 BYTES
0ADE                    ;ZIELADRESSE

0AE0                    ;CODE FÜR JP
0AE1                    ;SCR READ

0AE3                    ;CODE FÜR JP
0AE4                    ;SCR WRITE
```



```
0AE6                    ;CODE FÜR JP
0AE7                    ;SCR CLEAR


0AE9 ------------------------------------------------- SCR SET MODE
```

SCR SET MODE versetzt den Bildschirm in einen neuen Modus. Beim Ansprung dieser Routine muß im Akkumulator der Wert für den neuen Bildschirm-MODE übergeben werden. Zulässig ist einer der Werte 0, 1 oder 2.

```
0AE9                    ;BIT 2 BIS 7 AUSBLENDEN
0AEB                    ;IST MODE GRÖSSER ODER GLEICH 3
0AED                    ;WENN JA, DANN ZURÜCK
0AEE                    ;BILDSCHIRM-MODE RETTEN
0AEF                    ;TERMINATE COLOUR EVENT BLOCK
0AF2                    ;INHALT VON DE HOLEN
0AF3                    ;FARBEN DECODIEREN
0AF6                    ;AKKU UND FLAGS AUF STACK SICHERN
0AFA                    ;INHALT VON HL-REGISTER HOLEN
0AFB                    ;MODE HOLEN
0AFC                    ;MASKEN HOLEN
0AFF                    ;SCR CLEAR
0B02                    ;PEN UND PAPER HOLEN
0B06                    ;WINDOWNUMMER HOLEN
0B07                    ;FENSTERPARAMETER AUF DEFAULT
0B0A                    ;INSERT COLOUR EVENT BLOCK


0B0C ------------------------------------------------- SCR GET MODE
```

SCR GET MODE liefert im Akkumulator den gegenwärtigen Bildschirm-Modus. Die Flags werden je nach Modus wie folgt gesetzt:

```
      Mode 0: ZERO-Flag=0, CARRY-Flag=1
      Mode 1: ZERO-Flag=1, CARRY-Flag=0
      Mode 2: ZERO-Flag=0, CARRY-Flag=0

0B0C                    ;(CURR. SCREEN MODE)
0B0F                    ;FLAGS MANIPULIEREN
0B11                    ;ZURÜCK
```

```
0B12 ------------------------------------------------- MODE 1 SETZEN
```

Bringt den Bildschirm in MODE 1

```
0B12                    ;WERT FÜR CURR. SCREEN MODE
0B14                    ;DIESEN WERT WEGSCHREIBEN
```

```
0B17 ------------------------------------------------- SCR CLEAR
```

Bildschirm löschen.

```
0B17                    ;TERMINATE COLOUR EVENT BLOCK
0B1A                    ;ÜBERGABEPARAMETER FÜR SCR SET OFFSET
0B1D                    ;SCR SET OFFSET
0B20                    ;(ADRESSE SCREEN START)
0B23                    ;LSB AUF NULL SETZEN
0B25                    ;HL=BASISADRESSE
0B26                    ;DE=BASISADRESSE+1
0B28                    ;16 KILOBYTE
0B2B                    ;BYTE WIRD AUF NULL GEBRACHT
0B2C                    ;BILDSCHIRM LÖSCHEN
0B2E                    ;INSERT COLOUR EVENT BLOCK
0B31                    ;(CURR. SCREEN MODE)
0B34                    ;MC SET MODE (BILDSCHIRMMODUS SETZEN)
```

```
0B37 ------------------------------------------------- SCR SET OFFSET
```

Startadresse des ersten Zeichens relativ zur Basisadresse des Video-RAMs setzen.

```
0B37                    ;(SCR HIGH BYTE SCREEN START)
0B3A                    ;HIER GEHT ES WEITER
```

```
0B3C ------------------------------------------------- SCR SET BASE
```

Basisadresse des Video-RAMs.


```
0B3C                    ;(POSITION IN EINER ZEILE)
0B3F                    ;SCR VERÄNDERUNG SCREEN START
0B42                    ;MC SCREEN OFFSET
```

```
0B45 ------------------------------------ SCR VERÄNDERUNG SCREEN START
```

Als Eingabeparameter erhält die Routine in HL die Position in einer Zeile und im Akkumulator das Highbyte der Bank, die als Bildschirmspeicher definiert wurde.

```
0B45                    ;BIT 0-5 AUSBLENDEN
0B47                    ;(HIGH BYTE SCREEN START)
0B4A                    ;AKKU UND FLAGS SICHERN
0B4B                    ;LADE AKKUMULATOR MIT INHALT VON H
0B4C                    ;BIT 3-7 AUSBLENDEN
0B4E                    ;ERGEBNIS ZURÜCK NACH H
0B4F                    ;BIT 0 LÖSCHEN
0B51                    ;(POSITION IN EINER ZEILE)
0B54                    ;ALTEN ZUSTAND AKKU UND FLAGS
0B55                    ;ZURÜCK
```

```
0B56 ------------------------------------------------- SCR GET LOCATION
```

Beim Verlassen dieser Routine steht im Akkumulator das Highbyte des Screenstart und im HL-Register die Position in einer Zeile.

```
0B56                    ;(POSITION IN EINER ZEILE)
0B59                    ;HIGH BYTE SCREEN START
0B5C                    ;ZURÜCK
```

```
0B5D ------------------------------------------------- SCR CHAR LIMITS
```

SCR CHAR LIMITS liefert die größtmögliche Zeilen- und Spaltenzahl abhängig vom Bildschirm-Modus (MODE). Die Zahl der im aktuellen Modus mögli-

chen Zeilen wird im C-Register, die der Spalten im B-Register, zur weiteren Verwendung bereitgestellt.

```
0B5D                    ;SCR GET MODE
0B60                    ;B: ANZAHL DER SPALTEN MODE 0
                        ;C: ANZAHL DER ZEILEN
                        ;(POSITION 0,0 IST ZULÄSSIG!)
0B63                    ;MODE 0 IST GESETZT
0B64                    ;ANZAHL DER SPALTEN MODE 1
0B66                    ;MODE 1 IST GESETZT
0B67                    ;ANZAHL DER SPALTEN MODE 2
0B69                    ;ZURÜCK
```

```
0B6A ---------------------------------------------- SCR CHAR POSITION
```

Übersetze phys. Koordinaten in eine Bildschirmposition. Als Eingabeparameter erwartet SCR CHAR POSITION im Doppelregister HL die Koordinaten (H = Spalte und L = Zeile). Als Ausgabe befindet sich die Adresse der Zeichenposition im HL-Register und die Länge eines Zeichens in X-Richtung im B-Register.

```
0B6A                    ;INHALT VON DE AUF STACK RETTEN
0B6B                    ;SCR GET MODE
0B6E                    ;ZEICHENLÄNGE = VIER BYTES (MODE 0)
0B70                    ;MODE 0 AKTIV?
0B72                    ;ZEICHENLÄNGE = ZWEI BYTES (MODE 1)
0B74                    ;MODE 1 AKTIV?
0B76                    ;ZEICHENLÄNGE = EIN BYTE (MODE 2)
0B77                    ;RETTE X-AUSDEHNUNG DER ZEICHEN
0B78                    ;HOLE SPALTE
0B79                    ;HIGHBYTE SPALTE LÖSCHEN
0B7B                    ;H-REGISTER WIRD NULL
0B7C                    ;SPALTE AUF STACK RETTEN
0B7D                    ;D-REGISTER MSB ZEILE
0B7E                    ;E-REGISTER LSB ZEILE
0B7F                    ;HL=HL*80
0B80                    ;VGL. 0B7F
0B81                    ;VGL. 0B7F
0B82                    ;VGL. 0B7F
```



```
0B83                    ;VGL. 0B7F
0B84                    ;VGL. 0B7F
0B85                    ;VGL. 0B7F
0B86                    ;HOLE SPALTE
0B87                    ;ADDIERE SPALTE
0B88                    ;JE NACH MODE SPALTE, SPALTE*2 ODER
                        ;SPALTE*4 ZU HL ADDIEREN
0B8A                    ;HOLE SCREEN-OFFSET
0B8E                    ;SCREEN-OFFSET ADDIEREN
0B8F                    ;MSB NACH AKKUMULATOR
0B90                    ;BIT 3 BIS 7 AUSBLENDEN
0B92                    ;ERGEBNIS ZURÜCHSCHREIBEN
0B93                    ;(HIGH BYTE SCREEN START)
0B96                    ;BILDSCHIRMADRESSE + MSB SCREEN START
0B97                    ;ERGEBNIS NACH H-REGISTER
0B98                    ;HOLE X-AUSDEHNUNG DES ZEICHENS
0B99                    ;ALTEN INHALT VON DE HERSTELLEN
0B9A                    ;FERTIG!
```

```
0B9B ---------------------------------------------- COMPUTE WINDOW PARAMS
```

Als Eingabeparameter erwartet die Routine im H-Register die linke und im L-Register die obere Fenstergrenze. Analog dazu muß sich im D-Register die rechte Fenstergrenze und im E-Register die untere Fenstergrenze befinden. Bei der Ausgabe befindet sich linke obere Bildschirmadresse des Fensters im HL-Register, des weiteren gibt das D-Register die Anzahl der Bytes pro Fensterzeile aus. Im E-Register befindet sich die Anzahl der Rasterzeilen des Fensters.

```
0B9B                    ;UNTERE FENSTERGRENZE NACH AKKUMULATOR
0B9C                    ;FENSTERHÖHE BERECHNEN
0B9D                    ;FENSTERHÖHE UM EINS ERHÖHEN
0B9E                    ;ERGEBNIS MAL ACHT
0B9F                    ;ERGIBT DIE ANZAHL DER
0BA0                    ;RASTERZEILEN DES FENSTERS
0BA1                    ;ANZAHL DER RASTERZEILEN NACH E-REGISTER
0BA2                    ;RECHTE FENSTERGRENZE NACH AKKUMULATOR
0BA3                    ;FENSTERBREITE BERECHNEN
0BA4                    ;ERGEBNIS UM EINS ERHÖHEN
```

```
0BA5                    ;NACH D-REGISTER ZURÜCKSCHREIBEN
0BA6                    ;SCR CHAR POSITION
0BA9                    ;AKKU LÖSCHEN UND FLAGS ZURÜCKSETZEN
0BAA                    ;JE NACH MODE D
0BAB                    ;MIT 2, 4 ODER 8 MULIPLIZIEREN
0BAD                    ;ERGEBNIS ZURÜCK NACH D-REGISTER
0BAE                    ;ZURÜCK


0BAF ---------------------------------------------- SCR DOT POSITION
```

Bildschirmposition für ein Pixel ermitteln. Beim Aufruf erwartet SCR DOT POSITION im HL-Register die X und im DE-Register die Y-Koordinate. Beim Verlassen stellte die Routine im HL-Register die Bildschirmadresse für das Pixel zur Verfügung.

```
0BAF                    ;X-KOORDINATE AUF STACK SICHERN
0BB0                    ;Y-KOORDINATE NACH DE
0BB1                    ;MAXIMALWERT FÜR Y-KOORDINATE
0BB4                    ;CARRY FÜR SBC ZURÜCKSETZEN
0BB5                    ;MAXIMUM MINUS Y-KOORDINATE
0BB7                    ;LSB ERGEBNIS NACH AKKU
0BB8                    ;RASTERADRESSE FREISTELLEN
0BBA                    ;ALLE BITS DREIMAL
0BBB                    ;NACH LINKS ROTIEREN
0BBC                    ;(RA INTIALISIERUNG FÜR GATE-ARRAY)
0BBD                    ;ERGEBNIS NACH C-REGISTER
0BBE                    ;LBS MODIFIZIERTE Y-KOORDINATE NACH A
0BBF                    ;BITS 0 BIS 2 AUSBLENDEN
0BC1                    ;ERGEBNIS ZURÜCK NACH L-REGISTER
0BC2                    ;MODIFIZIERTES MSB DER Y-KOORDINATE
0BC3                    ;DE = MODIFIZIERTE Y-KOORDINATE
0BC4                    ;Y-KOORDINATE
0BC5                    ;MIT 10
0BC6                    ;MULIPLIZIEREN UND
0BC7                    ;SO ZEILE ERRECHNEN
0BC8                    ;X-KOORDINATE VON STACK HOLEN
0BC9                    ;BC FREISTELLEN
0BCA                    ;GET PIXEL MASK
0BCD                    ;AKKU = PUNKTMASKE
```



```
0BCE                    ;CALC MASK Y-INDEX
0BCF                    ;ERGEBNIS NULL?
0BD1                    ;X-INDEX HALBIEREN
0BD3                    ;MASK-INDEX-1
0BD4                    ;ERGEBNIS NULL?
0BD6                    ;HIGH- UND LOW-BYTE SWAP MIT STACK
0BD7                    ;VERGL. 0BD6
0BD8                    ;VERGL. 0BD6
0BD9                    ;VERGL. 0BD6
0BDA                    ;HOLE PUNKTMASKE
0BDB                    ;UND VERSCHIEBE PIXELS IN MASKE
0BDC                    ;X-POS = X-POS/2 (MSB)
0BDE                    ;X-POS = X-POS/2 (LSB)
0BE0                    ;UND VERSCHIEBE PIXELS IN MASKE
0BE1                    ;LOW BIT IN CARRY?
0BE3                    ;BERECHNE BILDSCHIRMADRESSE
0BE4                    ;(POSITION IN EINER ZEILE)
0BE8                    ;ADDIERE VIDEO-START
0BE9                    ;HOLE MSB VON BILDSCHIRMADRESSE
0BEA                    ;UND MODIFIZIERE ES MIT AKTUELLER
                        ;RASTER ADRESS
0BEC                    ;ERGEBNIS ZURÜCKSCHREIBEN
0BED                    ;(HIGH BYTE SCREEN START)
0BF0                    ;A = MSB DER BILDSCHIRMADRESSE
0BF1                    ;ADDIERE SCROFF
0BF2                    ;UND ERGEBNIS ZURÜCKSCHREIBEN
0BF3                    ;VIDEO-START ADRESS
0BF4                    ;HIGH-BYTE NACH C
0BF5                    ;DAS WAR'S!


0BF6 ---------------------------------------------- GET PIXEL MASK

0BF6                    ;SCR GET MODE
0BF9                    ;PIXELMASKE FÜR MODE 0
0BFC                    ;MODE 0 AKTIV?
0BFD                    ;PIXELMASKE FÜR MODE 1
0C00                    ;MODE 1 AKTIV?
0C01                    ;PIXELMASKE FÜR MODE 2
0C04                    ;MODE 2 AKTIV!
```

```
0C05 ---------------------------------------------------- SCR NEXT BYTE
```

Liefert in HL die Bildschirmadresse der nächsten Byteposition zurück, wenn Sie vor dem Ansprung HL mit der alten Adresse versorgt haben. So überflüssig das scheinen mag, so practisch ist es. Es ist nämlich, aufgrund der auf Grafikbetrieb ausgerichteten Organisation des Bildschirms, nicht einfach, die Byteposition zu ermitteln. Zudem ist die Distanz vom Modus abhängig.

Beachten Sie, daß, wenn die nächste Position icht mehr innerhalb des Bildschirms läge, die zurückgelieferte Adresse unsinnig ist. Sie liegt dann im Bereich der letzten (für die Darstellung unbenutzten) Bytes des Video-RAMs.

```
0C05                    ;LOW BYTE BILDSCHIRMADRESSE +1
0C06                    ;ZURÜCK WENN KEIN ÜBERLAUF
0C07                    ;HIGH BYTE BILDSCHIRMADRESSE +1
0C08                    ;NEUES HIGH BYTE IN AKKU
0C09                    ;BITS 3 BIS 7 AUF NULL SETZEN
0C0B                    ;ZURÜCK WENN AKKU <> 0
0C0C                    ;HIGH BYTE ZU WEITERER BERECHNUNG IN
                        ;DEN AKKU
0C0D                    ;&08 VOM AKKU SUBTRAHIEREN
0C0F                    ;NEU BERECHNETER WERT FÜR HIGH BYTE
0C10                    ;ZURÜCK MIT BILDSCHIRMADRESSE FÜR
                        ;NÄCHSTE BYTEPOSITION IN HL
```

```
0C11 ---------------------------------------------------- SCR PREV BYTE
```

Liefert in HL die Bildschirmadresse der vorigen Byteposition zurück, wenn Sie vor dem Ansprung HL mit der alten Adresse geladen haben. Vergleichen Sie mit SCR NEXT BYTE.

```
0C11                    ;LOW BYTE BILDSCHIRMADRESSE NACH AKKU
0C12                    ;LOW BYTE-1
0C13                    ;AKKUMULATOR GLEICH 0?
0C14                    ;ZURÜCK WENN AKKU <> 0
```



```
0C15                    ;HIGH BYTE BILDSCHIRMADRESSE NACH AKKU
0C16                    ;HIGH BYTE-1
0C17                    ;BITS 3 BIS 7 AUF NULL SETZEN
0C19                    ;ZURÜCK WENN AKKU <> 0
0C1A                    ;HIGH BYTE ZU WEITERER BERECHNUNG IN
                        ;DEN AKKU
0C1B                    ;&08 ZUM AKKU ADDIEREN
0C1D                    ;NEU BERECHNETER WERT FÜR HIGH BYTE
0C1E                    ;ZURÜCK MIT BILDSCHIRMADRESSE FÜR
                        ;VORIGE BYTEPOSITION IN HL
```

```
0C1F ---------------------------------------------------- SCR NEXT LINE
```

Arbeitet analog zu SCR NEXT BYTE, nur daß die Bildschirmadresse um eine ganze Zeile vorgerechnet wird. Auch hier ist die Adresse beim Verlassen des darstellbaren Bereiches ungültig.

```
0C1F                    ;HIGH BYTE BILDSCHIRMADRESSE NACH AKKU
0C20                    ;&08 ZUM AKKU ADDIEREN
0C22                    ;ERGEBNIS ZURÜCK NACH H
0C23                    ;BITS 0 BIS 2 UND 6 BIS 7 AUSSCHALTEN
0C25                    ;UNTER DIESER VORAUSSETZUNG ZURÜCK
0C26                    ;H WIEDER IN DEN AKKU
0C27                    ;&40 VOM AKKU SUBTRAHIEREN
0C29                    ;ERGEBNIS ZURÜCK NACH H
0C2A                    ;LOW BYTE BILDSCHIRMADRESSE NACH AKKU
0C2B                    ;&50 ZUM AKKU ADDIEREN
0C2D                    ;ERGEBNIS ZURÜCK NACH L
0C2E                    ;ZURÜCK, WENN BEDINGUNG ERFÜLLT
0C2F                    ;H+1
0C30                    ;H WIEDER IN DEN AKKU
0C31                    ;BITS 3 BIS 7 AUF NULL SETZEN
0C33                    ;UNTER DIESER PRÄMISSE ZURÜCK
0C34                    ;H WIEDER IN DEN AKKU
0C35                    ;&08 VOM AKKU SUBTRAHIEREN
0C37                    ;ERGEBNIS ZURÜCK NACH H
0C38                    ;UNBEDINGTER RÜCKSPRUNG
```

```
0C39 ---------------------------------------------------- SCR PREV BYTE
```

Arbeitet analog zu SCR PREV BYTE, nur daß die Bildschirmadresse um eine ganze Zeile zurückgerechnet wird. Vergleichen Sie mit SCR NEXT LINE und SCR PREV BYTE.

```
0C39                    ;HIGH BYTE BILDSCHIRMADRESSE NACH AKKU
0C3A                    ;&08 VOM AKKU SUBTRAHIEREN
0C3C                    ;ERGEBNIS ZURÜCK NACH H
0C3D                    ;BITS 0 BIS 2 UND 6 BIS 7 AUSSCHALTEN
0C3F                    ;BITS 3 BIS 5 GESETZT?
0C41                    ;ZURÜCK UNTER DIESER PRÄMISSE
0C42                    ;H WIEDER IN DEN AKKU
0C43                    ;&40 ZUM AKKU ADDIEREN
0C45                    ;ERGEBNIS WIEDER NACH H
0C46                    ;LOW BYTE BILDSCHIRMADRESSE NACH AKKU
0C47                    ;&50 VOM AKKU SUBTRAHIEREN
0C49                    ;ERGEBNIS ZURÜCK NACH L
0C4A                    ;UNTER DIESER VORAUSSETZUNG ZURÜCK
0C4B                    ;H WIEDER IN DEN AKKU
0C4C                    ;H-1
0C4D                    ;BITS 3 BIS 7 AUF NULL SETZEN
0C4F                    ;ZURÜCK, WENN BEDINGUNG ERFÜLLT
0C50                    ;H WIEDER IN DEN AKKU
0C51                    ;&08 ZUM AKKU ADDIEREN
0C53                    ;ERGEBNIS ZURÜCK NACH H
0C54                    ;UNBEDINGTER RÜCKSPRUNG
```

```
0C55 ---------------------------------------------------- SCR ACCESS
```

Steuerzeichen sichtbar/unsichtbar. Bei der Eingabe erwartet SCR ACCESS den Bildschirmmodus im Akkumulator.

```
0C55                    ;UNNÖTIGE BITS ABSCHALTEN
0C57                    ;SCR PIXELS (FORCE MODE)
0C5A                    ;FORCE MODE?
0C5C                    ;XOR-MODE DEFINIERT?
0C5E                    ;LOW BYTE XOR-MODE
0C60                    ;SPRINGE WENN XOR-MODE
```



```
0C62                    ;LOW BYTE AND-MODE
0C64                    ;SPRINGE WENN AND-MODE
0C66                    ;LOW BYTE OR-MODE
0C68                    ;OPCODE FÜR JP
0C6A                    ;OPCODE WEGSCHREIBEN
0C6D                    ;(SCR WRITE INDIRECTION)
0C70                    ;ZURÜCK
```

```
0C71 ---------------------------------------------------- SCR WRITE
```

```
0C71                    ;SCR WRITE INDIRECTION
```

```
0C74 ---------------------------------------- SCR PIXELS (FORCE MODE)
```

Beim Aufruf dieser Routine muß im B-Register das Pen-Byte und im D-Register die Bitmaske für das Pixel stehen. Im HL-Register steht die Adresse für das Pixel im Bildschirmspeicher.

```
0C74                    ;PEN-BYTE NACH AKKUMULATOR
0C75                    ;MIT BILDSCHIRMSPEICHER XOR-VERKNÜPFEN
0C76                    ;BITMASKE FÜR PIXEL
0C77                    ;MIT BILDSCHIRMSPEICHER XOR-VERKNÜPFEN
0C78                    ;IN DEN BILDSCHIRMSPEICHER SCHREIBEN
0C79                    ;ES GIBT NICHTS MEHR ZU TUEN
```

```
0C7A ---------------------------------------------------- XOR-MODE
```

Beim Aufruf dieser Routine muß im B-Register das Pen-Byte und im D-Register die Bitmaske für das Pixel stehen. Im HL-Register steht die Adresse für das Pixel im Bildschirmspeicher.

```
0C7A                    ;PEN-BYTE NACH AKKUMULATOR
0C7B                    ;BITMASKE FÜR PIXEL
0C7C                    ;MIT BILDSCHIRMSPEICHER XOR-VERKNÜPFEN
0C7D                    ;IN DEN BILDSCHIRMSPEICHER SCHREIBEN
0C7E                    ;DAS WAR'S
```

```
0C7F ---------------------------------------------------- AND-MODE
```

Beim Aufruf dieser Routine muß im B-Register das Pen-Byte und im D-Register die Bitmaske für das Pixel stehen. Im HL-Register steht die Adresse für das Pixel im Bildschirmspeicher.

```
0C7F                    ;BITMASKE FÜR PIXEL NACH AKKU
0C80                    ;NEGATMASKE ERSTELLEN
0C81                    ;MIT PEN-BYTE OR-VERKNÜPFEN
0C82                    ;UND MIT BILDSCHIRMSPEICHER UNDIEREN
0C83                    ;ERGEBNIS IN DEN BILDSCHIRM-
                        ;SPEICHER SCHREIBEN
0C84                    ;ERLEDIGT
```

```
0C85 ---------------------------------------------------- OR-MODE
```

Beim Aufruf dieser Routine muß im B-Register das Pen-Byte und im D-Register die Bitmaske für das Pixel stehen. Im HL-Register steht die Adresse für das Pixel im Bildschirmspeicher.

```
0C85                    ;PEN-BYTE NACH AKKU
0C86                    ;MIT BITMASKE FÜR PIXEL UNDIEREN
0C87                    ;UND MIT BILDSCHRIMSPEICHER ODERIEREN
0C88                    ;IN DEN BILDSCHIRMSPEICHER SCHREIBEN
0C89                    ;FERTIG
```

```
0C8A ---------------------------------------------------- SCR READ
```

Beim Aufruf erwartet SCR READ eine Adresse aus dem Video-RAM im HL-Register vorzufinden.

```
0C8A                    ;HOLE BYTE AUS VIDEO-RAM
0C8B                    ;BERECHNE FARBSTIFT
```



```
0C8E ---------------------------------------------- SCR INK ENCODE
```

Codieren einer Ink, so daß alle Bildpunkte auf diese Ink gesetzt werden. Diese Routine erwartet bei der Eingabe den Farbstift im Akkumulator und schreibt bei der Ausgabe die Farbmaske in den Akku.

```
0C8E                    ;BC-REGISTER FREISTELLEN
0C8F                    ;DE-REGISTER FREISTELLEN
0C90                    ;CNV PEN-NUMBER
0C93                    ;PEN-NUMMER NACH E-REGISTER
0C94                    ;GET PIXEL MASK
0C97                    ;ZÄHLER FÜR MASKENMANIPULATION
0C99                    ;BIT 0 DER PEN-NUMMER INS CARRY
0C9B                    ;UND NACH BIT 7 DES AKKUS
0C9C                    ;PIXEL-BITMASKE MANIPULIEREN
0C9E                    ;WENN BIT 0 = 1 NÄCHSTES BIT HOLEN
0CA0                    ;ANDERENFALLS MASKE REKONSTRUIEREN
0CA2                    ;NÄCHSTES BIT HOLEN
0CA4                    ;ALTEN INHALT VON DE-REGISTER
0CA5                    ;ALTEN INHALT VON BC-REGISTER
0CA6                    ;DAS WAR'S
```

```
0CA7 ---------------------------------------------- SCR INK DECODE
```

Entschlüsseln einer Ink. Diese Routine erwartet bei der Eingabe die Farbmaske im Akkumulator und schreibt bei der Ausgabe den Farbstift in den Akku.

```
0CA7                    ;BC FREISTELLEN
0CA8                    ;INHALT VON AKKUMULATOR RETTEN
0CA9                    ;GET PIXEL MASK
0CAC                    ;INHALT VON AKKUMULATOR HOLEN
0CAD                    ;CALC PEN NUMBER
0CB0                    ;INHALT VON BC HOLEN
0CB1                    ;FERTIG!
```

```
0CB2 ---------------------------------------------- CALC PEN-NUMBER
```

Im Akkumulator wird an CALC PEN-NUMBER die Farbmaske übergeben und im C-Register die Pixel-Bitmaske. Beim Verlassen der Routine steht im Akkumulator die Farbstiftnummer.

```
0CB2                    ;DE-REGISTER FREISTELLEN
0CB3                    ;BITZÄHLER INITIALISIEREN
0CB6                    ;HOLE BIT AUS FARBMASKE
0CB7                    ;UND NACH D ÜBERTRAGEN
0CB9                    ;HOLE BIT AUS PIXELAUSWAHLMASKE
0CBB                    ;BIT = 0?
0CBD                    ;ÜBERTRAGE BIT NACH D
0CBF                    ;BITCOUNTER - 1
0CC0                    ;WEITERE BIT VORHANDEN?
0CC2                    ;AKKU WIRD PEN-NUMBER
0CC3                    ;CNV PEN-NUMBER
0CC6                    ;ALTEN INHALT VON DE HERSTELLEN
0CC7                    ;FERTIG!
```

```
0CC8 ---------------------------------------------- CNV PEN-NUMBER
```

```
0CC8                    ;AKKU NACH D-REGISTER ÜBERSTELLEN
0CC9                    ;SCR GET MODE
0CCC                    ;WERT ZURÜCK NACH AKKUMULATOR
0CCD                    ;ZURÜCK WENN UNGLEICH MODE 0
0CCE                    ;AKKUMULATOR INHALT
0CCF                    ;DURCH VIER TEILEN
0CD0                    ;CARRY ADDIEREN FALLS VORHANDEN
0CD2                    ;ERGEBNIS DURCH ZWEI TEILEN
0CD3                    ;ISOLIERE CARRY (CARRY = &FF)
0CD4                    ;MIT SECHS UNDIEREN
0CD6                    ;A = PEN NUMMER
0CD7                    ;ERLEDIGT!
```



```
0CD8 ---------------------------------------------- RESET FARBEN
```

```
0CD8                    ;QUELLE: DEFAULT FARBEN
0CDB                    ;ZIEL: FARBSPEICHER ZWEITE FARBEN
0CDE                    ;ZÄHLER: &22 BYTES
0CE1                    ;KOPIERVORGANG AUSFÜHREN
0CE3                    ;AKKU LÖSCHEN UND FLAGS ZURÜCKSETZEN
0CE4                    ;(FLAG LFD FARBSATZ)
0CE7                    ;STARTWERT FÜR FLASH PERIODS
```

```
0CEA ---------------------------------------------- SCR SET FLASHING
```

Blinkzeiten zur Farbdarstellung für alle Inks und den Rahmen setzen. Im Registerpaar HL muß der Wert für die Systemvariable FLASH PERIODS an SCR SET FLASHING übergeben werden.

```
0CEA                    ;(FLASH PERIODS)
0CED                    ;SCHON FERTIG
```

```
0CEE ---------------------------------------------- SCR GET FLASHING
```

```
0CEE                    ;(FLASH PERIODS)
0CF1                    ;DAS WAR'S
```

```
0CF2 ---------------------------------------------- SCR SET INK
```

Zuordnung der beiden Farben, die zur Darstellung einer Ink verwendet werden. Beim Aufruf der Routine muß die Nummer des Farbstifts im Akkumulator stehen. Im B-Register erwachtet SCR SET INK die Nummer der ersten Farbe und im C-Register die Nummer der zweiten Farbe.

```
0CF2                    ;BIT 4 BIS 7 AUSSCHALTEN
0CF4                    ;AKKUMULATOR UM 1 ERHÖHEN
                        ;DIE FARBSTIFTNUMMER LIEGT NUN IM
                        ;BEREICH 1 BIS 16
0CF5                    ;SET COLOUR
```

```
0CF7 ---------------------------------------------------- SCR SET BORDER
```

Zuordnung der beiden Farben, die zur Darstellung eines Rahmens verwendet werden. Im B-Register erwachtet SCR SET BORDER die Nummer der ersten Farbe und im C-Register die Nummer der zweiten Farbe.

```
0CF7                        ;0 IST FARBSTIFT FÜR BORDER


0CF8 ---------------------------------------------------- ;SET COLOUR
```

Im B-Register erwachtet diese Routine die Nummer der ersten Farbe und im C-Register die Nummer der zweiten Farbe.

```
0CF8                        ;NUMMER DES FARBSTIFTS BUFFERN
0CF9                        ;NUMMER ERSTE FARBE NACG AKKU
0CFA                        ;FARBMATRIX EINTRAG HOLEN
0CFD                        ;ERSTER FARBWERT NACH B
0CFE                        ;NUMMER ZWEITE FARBE NACH AKKU
0CFF                        ;FARBMATRIX EINTRAG HOLEN
0D02                        ;ZWEITER FARBWERT NACH C
0D03                        ;NUMMER DES FARBSTIFTS ZURÜCK IN A
0D04                        ;INK ADRESSE HOLEN
0D07                        ;ZWEITER FARBWERT IM RAM ABLEGEN
0D08                        ;ZEIGER TAUSCHEN
0D09                        ;ERSTER FARBWERT IM RAM ABLEGEN
0D0A                        ;FLAG FÜR NEUE FARBWERTE
0D0C                        ;FLAG WIRD GESETZT
0D0F                        ;ERLEDIGT


0D10 ------------------------------------------ FARBMATRIX EINTRAG HOLEN
```

Diese Routine ermittelt die Adresse eines Farbwerts innerhalb der Farbmatrix. Als Eingabeparameter wird im Akkumulator eine Farbnummer erwartet. Beim Verlassen von FARBMATRIX EINTRAG HOLEN steht im HL-Register die Adresse des Farbwerts in der Farbmatrix.


```
0D10                        ;IM BEREICH 0 BIS 31
0D12                        ;LSB DER BASISADRESSE ADDIEREN
0D14                        ;ERGEBNIS NACH L BRINGEN
0D15                        ;MSB BASISADRESSE UND CARRY ADDIEREN
0D17                        ;LSB WIEDER ABZIEHEN
0D18                        ;MSB DER ADRESSE NACH H
0D19                        ;DAS WAR'S


0D1A ---------------------------------------------------- SCR GET INK
```

Holen der beiden Farben, die zur Darstellung einer Ink verwendet werden. Als Eingabe wird die Nummer des Farbstifts im Akkumulator erwachtet. Als Ausgabe liefert SCR GET INK die Nummer der ersten Farbe im B-Register und die Nummer der zweiten Farbe im C-Register.

```
0D1A                        ;BITS 4 BIS 7 AUSBLENDEN
0D1C                        ;DANN UM EINS ERHÖHEN
                            ;DIE FARBSTIFTNUMMER LIEGT NUN IM
                            ;BEREICH 1 BIS 16
0D1D                        ;GET COLOUR


0D1F ---------------------------------------------------- SCR GET BORDER
```

Holen der beiden Farben, die zur Darstellung eines Rahmens verwendet werden. Als Ausgabe liefert SCR GET BORDER die Nummer der ersten Farbe im B-Register und die Nummer der zweiten Farbe im C-Register.

```
0D1F                        ;0 IST FARBSTIFT FÜR BORDER


0D20 ---------------------------------------------------- GET COLOUR
```

Als Ausgabe liefert GET COLOUR die Nummer der ersten Farbe im B-Register und die Nummer der zweiten Farbe im C-Register.

```
0D20                        ;INK ADRESSE HOLEN
0D23                        ;ERSTER FARBWERT NACH A
0D24                        ;ZWEITER FARBWERT NACH E
```

```
0D25                    ;FARBWERT IN TABELLE LOKALISIEREN
0D28                    ;ERSTE FARBNUMMER NACH B-REGISTER
0D29                    ;ZWEITER FARBWERT NACH AKKU
0D2A                    ;COUNTER INITIALISIEREN
0D2C                    ;ADRESSE DER FARBMATRIX NACH HL
0D2F                    ;FARBWERT LOKALISIERT?
0D30                    ;DANN FERTIG!
0D31                    ;TABELLENPOINTER UND
0D32                    ;FARBNUMMER ERHÖHEN
0D33                    ;SUCHE FORTSETZEN!
```

0D35 ---------- INK ADRESSE HOLEN

Als Eingabe erwartet INK ADRESSE HOLEN die Nummer des Farbstifts im Akkumulator. Mit dieser Angabe ist es der Routine möglich, zu ermitteln, wo im RAM die beiden Farbwerte stehen, die dem Farbstift zugeordnet sind. Bei Verlassen von INK ADRESSE HOLEN befindet sich in DE die Adresse, an der sich der erste Farbwert befindet, und HL enthält die Adresse des zweiten Farbwerts.

```
0D35                    ;NUMMER DES FARBSTIFTS
0D36                    ;MSB ERZEUGEN
0D38                    ;FARBSPEICHER 1. FARBEN
0D3B                    ;NUMMER DES FARBSTIFTS ADDIEREN
0D3C                    ;ERGEBNIS NACH DE TAUSCHEN
0D3D                    ;HILFSSUMMAND FÜR ZWEITEN FARBWERT
0D40                    ;HL ZEIGT AUF ZWEITEN FARBWERT
0D41                    ;FERTIG
```

0D42 ---------- INSERT COLOUR EVENT BLOCK

```
0D42                    ;EVENT BLOCK SET INKS
0D45                    ;ADRESSE EVENT BLOCK AUF STACK
0D46                    ;KL DEL FRAME FLY
0D49                    ;FLASH INKS
0D4C                    ;SET INKS ON FRAME FLY
0D4F                    ;LADE B-REGISTER MIT &81
```



```
0D51                    ;ADRESSE EVENT BLOCK VOM STACK
0D52                    ;KL NEW FRAME FLY
```

0D55 ---------- TERMINATE COLOUR EVENT BLOCK

```
0D55                    ;EVENT BLOCK: SET INKS
0D58                    ;KL DEL FRAME FLY
0D5B                    ;PARAMS D. LFD FARBSATZ HOLEN
0D5E                    ;MC CLEAR INKS
```

0D61 ---------- SET INKS ON FRAME FLY

```
0D61                    ;CURR. FLASH PERIOD
0D64                    ;DEKREMENTIERE FLASH PERIOD
0D65                    ;FLASH INKS
0D67                    ;ZEIGER AUF COLVAL
0D68                    ;HOLE COLVAL-FLAGS
0D69                    ;COLVAL-FLAG POSITIV?
0D6A                    ;WENN NEIN, DANN FERTIG!
0D6B                    ;PARAMS D. LFD FARBSATZ HOLEN
0D6E                    ;MC SET INKS
0D71                    ;DELETE COLVAL
```

0D73 ---------- FLASH INKS

```
0D73                    ;PARAMS D. LFD FARBSATZ HOLEN
0D76                    ;(CURR FLASH PERIOD)
0D79                    ;MC SET INKS
0D7C                    ;FLAG LFD. FARBSATZ
0D7F                    ;LFD. FARBSATZ NACH AKKU
0D80                    ;BILDE EINERKOMPLEMENT
0D81                    ;ERGEBNIS ZURÜCK NACH FLAG LFD. FARBSATZ
0D82                    ;AKKU LÖSCHEN UND FLAGS ZURÜCKSETZEN
0D83                    ;ALLE COLVAL-FLAGS LÖSCHEN
0D86                    ;ZURÜCK
```

```
0D87 ---------------------------------- PARAMS DES LFD FARBSATZ HOLEN
```

Als Ausgaben liefert diese Routine im Akkumulator die Flash Periods und im DE-Register die Adresse der Farbtabelle.

```
0D87                    ;FARBSPEICHER 1. FARBEN
0D8A                    ;(FLAG LFD. FARBSATZ)
0D8D                    ;AKKU = 0?
0D8E                    ;(FLASH PERIOD 1. FARBE)
0D91                    ;WENN AKKU = 0, DANN FERTIG!
0D92                    ;FARBSPEICHER 2. FARBE
0D95                    ;(FLASH PERIODS)
0D98                    ;ZURÜCK


0D99 ------------------------------------------------------- FARBMATRIX

0D99                    14 04 15 1C 18 1D 0C 05
0DA1                    0D 16 06 17 1E 00 1F 0E
0DA9                    07 0F 12 02 13 1A 19 1B
0DB1                    0A 03 0B 01 08 09 10 11


0DB9 ------------------------------------------------------ SCR FILL BOX
```

Vorgegebenes Fenster mit einer Farbe füllen (Position Zeichenbezogen, Mode-abhängig). Im Akkumulator wird die Farbmaske an SCR FILL BOX übergeben.

```
0DB9                    ;FARBMASKE NACH C-REGISTER
0DBA                    ;COMPUTE WINDOW PARAMS


0DBD ------------------------------------------------------ SCR FLOOD BOX
```

Vorgegebenes Fenster mit einer Farbe füllen (Positionen sind Bildschirmadressen, Mode-unabhängig). Zum Füllen des Fenster muß der Routine im

HL-Register die Bildschirmadresse übergeben werden, ab der zu Füllen ist. Der Inhalt des DE-Register legt die Größe des zu füllenden Bereiches fest und das C-Register enthält die Farbmaske, mit der gefüllt werden soll.

```
0DBD                    ;BILDSCHIRMADRESSE AUF STACK LEGEN
0DBE                    ;ANZAHL BYTES PRO ZEILE
0DBF                    ;ÜBERTRAG ERMITTELN
0DC2                    ;KEIN ÜBERTRAG
0DC4                    ;ANZAHL BYTES PRO ZEILE
0DC5                    ;FARBMASKE IN BILDSCHIRMSPEICHER
0DC6                    ;SCR NEXT BYTE
0DC9                    ;WEITER SOLANGE NOCH BYTES
0DCB                    ;WEITERE RASTERZEILE
0DCD                    ;BC-REGISTER FREISTELLEN
0DCE                    ;DE-REGISTER FREISTELLEN
0DCF                    ;BYTE IN BILDSCHIRMSPEICHER SCHREIBEN
0DD0                    ;SPRINGE, WENN DIESE
0DD1                    ;ZEILE IST ABGEHANDELT
0DD3                    ;LSB ZÄHLER ZU KOPIERENDE BYTES
0DD4                    ;MSB ZÄHLER
0DD6                    ;MSB BILDSCHIRMADRESSE
0DD7                    ;LSB BILDSCHIRMADRESSE
0DD8                    ;ZIELADRESSE FÜR LDIR
0DD9                    ;BILDSCHIRMSPEICHER FÜLLEN
0DDB                    ;ALTEN INHALT VON DE-REGISTER
0DDC                    ;ALTEN INHALT VON BC-REGSITER
0DDD                    ;BILDSCHIRMADRESSE VOM STACK HOLEN
0DDE                    ;SCR NEXT LINE
0DE1                    ;WENN ES NOCH RASTERZEILEN GIBT,
0DE2                    ;DANN MACHE WEITER
0DE4                    ;AUFGABE ERLEDIGT


0DE5 ------------------------------------------------- SCR CHAR INVERT
```

Bei einem Zeichen Vorder- und Hintergrundfarbe vertauschen. Im HL-Register wird die Position des Zeichen an SCR CHAR INVERT übergeben (H=Spalte, L=Zeile).

```
0DE5                    ;HINTERGRUNDFARBE NACH AKKU
0DE6                    ;XOR-VERKNÜPFUNG MIT VORDERGRUNDFARBE
0DE7                    ;INVERTIERMASKE NACH C BRINGEN
0DE8                    ;SCR CHAR POSITION
0DEB                    ;ES GIBT ACHT RASTERZEILEN
0DED                    ;BILDSCHIRMADRESSE AUF STACK
0DEE                    ;B=ANZAHL DER BYTES PRO ZEICHEN
                        ;C=ERMITTELTE INVERTIERMASKE
0DEF                    ;BYTE VON BILDSCHIRMADRESSE LESEN
0DF0                    ;BYTE INVERTIEREN
0DF1                    ;BYTE WIEDER AN BILDSCHIRMADRESSE
0DF2                    ;SCR NEXT BYTE
0DF5                    ;WEITER SOLANGE NOCH BYTES VORHANDEN
0DF7                    ;B=ANZAHL DER BYTES PRO ZEICHEN
                        ;C=ERMITTELTE INVERTIERMASKE
0DF8                    ;BILDSCHIRMADRESSE VOM STACK HOLEN
0DF9                    ;SCR NEXT LINE
0DFC                    ;RASTERZEILE=RASTERZEILE-1
0DFD                    ;WEITER SOLANGE RASTERZEILEN
0DFF                    ;FERTIG
```

0E00 ---------- SCR HW ROLL

Schiebt den Bildschirm (hardwaremäßig) um eine Zeile nach unten, wenn B=0 ist und um eine Zeile nach oben, wenn B<>0 ist. Im Akku muß der Wert für die Farben stehen, die die neue (leere) Zeile annehmen soll.

```
0E00                    ;HINTERGRUNDFARBE NACH C-REGISTER
0E01                    ;SCROLL-RICHTUNG UND FARBE AUF STACK
0E02                    ;DE MIT DEFAULTWERT LADEN
0E05                    ;B-REGISTER MIT DEFAULTWERT LADEN
0E07                    ;FILL AREA
0E0A                    ;SCROLL-RICHTUNG UND FARBE VOM STACK
0E0B                    ;MC WAIT FLYBACK
0E0E                    ;SCROLL-RICHTUNG NACH AKKUMULATOR
0E0F                    ;ZERO-FLAG MANIPULIEREN
0E10                    ;SCROLL-RICHTUNG OBEN
0E12                    ;DE = OFFSET
0E15                    ;ADD OFFSET
```



```
0E18                    ;DE INITIALISIEREN
0E1B                    ;BYTES PRO ZEILE
0E1D                    ;ADD OFFSET
0E1F                    ;DE = OFFSET
0E22                    ;ADD OFFSET
0E25                    ;DE = OFFSET
0E28                    ;BYTES PRO ZEILE
0E2A                    ;POSITION IN EINER ZEILE
0E2D                    ;ADDIERE OFFSET
0E2E                    ;A = HIGH BYTE
0E2F                    ;RASTER ADRESS EINBEZIEHEN
0E31                    ;UND ERGEBNIS ZURÜCKSCHREIBEN
0E32                    ;(HIGH BYTE SCREEN START)
0E35                    ;MSB BILDSCHIRMADRESSE
0E36                    ;ERRECHNEN UND INS H-REGISTER
0E37                    ;BYTES PRO ZEILE
0E38                    ;ES GIBT ACHT RASTERZEILEN
0E3A                    ;SCR FLOOD BOX
```

0E3D ---------- ADD OFFSET

```
0E3D                    ;(POSITION IN EINER ZEILE)
0E40                    ;BASISADRESSE UND OFFSET ADDIEREN
0E41                    ;SCR SET OFFSET
```

0E44 ---------- SCR SW ROLL

Verschiebt einen Bildschirmbereich softwaremäßig. A und B sind wie bei SCR HW ROLL zu beschicken. Zusätzlich muß H die Spaltennummer des linken Randes des zu verschiebenden Bereiches enthalten, L die oberste Zeile, D die rechte Spalte und E die unterste Zeile des Bereiches.

Beachten Sie, daß Spalte und Zeile 0 die linke obere Ecke des Bildschirms darstellt. Achten Sie auch unbedingt selbst darauf, daß die übergebenen Parameter tatsächlich einen Bereich innerhalb des Video-RAMs markieren.

```
0E44                    ;HINTERGRUNDFARBE RETTEN
0E45                    ;A = SCROLL-RICHTUNG
0E46                    ;ERGEBNIS NULL?
0E47                    ;WENN 0 ABWÄRTS SCROLLEN
0E49                    ;RETTE X1,Y1
0E4A                    ;COMPUTE WINDOW PARAMS
0E4D                    ;HOLE X1 UND Y1
0E4E                    ;Y1=Y1+1
0E4F                    ;SCR CHAR POSITION
0E52                    ;C = X2
0E53                    ;A = Y2
0E54                    ;Y2=Y2 MINUS BILDSCHIRMZEILE (8
                        ;RASTERZEILE)
0E56                    ;ERGEBNIS (COUNTER) NACH B
0E57                    ;ERGEBNIS NULL?
0E59                    ;HOLE X2 UND Y2
0E5A                    ;MC WAIT FLYBACK
0E5D                    ;RETTE X2,YCOUNTER
0E5E                    ;RETTE X1 UND Y1
0E5F                    ;RETTE X2 UND Y2
0E60                    ;COPY RASTER LINE
0E63                    ;HOLE X2 UND Y2
0E64                    ;SCR NEXT LINE
0E67                    ;DE = X2,Y2
0E68                    ;HOLE X1 UND Y1
0E69                    ;SCR NEXT LINE
0E6C                    ;HOLE X2 UND MODY2 (COUNTER)
0E6D                    ;COUNTER=0? (RASTERZEILEN)
0E6F                    ;ÜBERTRAGE X2,Y2
0E70                    ;NACH HL
0E71                    ;BYTE PRO RASTERZEILE
0E72                    ;E = EINE ZEILE (8 RASTERZEILEN)
0E74                    ;HOLE PAPIERFARBE
0E75                    ;C = PAPIERFARBE
0E76                    ;SCR FLOOD BOX
0E79                    ;X1 UND Y1 RETTEN
0E7A                    ;X2 UND Y2 RETTEN
0E7B                    ;CALC WIN PARAMS
0E7E                    ;ANZAHL DER BYTES IM FENSET (X-RICHTUNG)
0E7F                    ;A = ZEILEN (Y-RICHTUNG)
```



```
0E80                    ;MINUS 8 RASTERZEILEN
0E82                    ;ERGEBNIS NACH B
0E83                    ;HOLE X2 UND Y2
0E84                    ;MIT X1,Y1 TAUSCHEN
0E85                    ;RASTERZEILE KOPIEREN
0E87                    ;ANZAHL DER ZEILEN RETTEN (COUNTER)
0E88                    ;L = X1
0E89                    ;D = Y1
0E8A                    ;X1 = X1 MINUS 1
0E8B                    ;SCR CHAR POSITION
0E8E                    ;DE = X1,Y1
0E8F                    ;SCR CHAR POSITION
0E92                    ;HOLE COUNTER
0E93                    ;MC WAIT FLYBACK
0E96                    ;SCR PREV LINE
0E99                    ;X2,Y2 RETTEN
0E9A                    ;DE = X2,Y2
0E9B                    ;SCR PREV LINE
0E9E                    ;X1,Y1 RETTEN
0E9F                    ;COUNTER RETTEN
0EA0                    ;COPY RASTER LINE
0EA3                    ;COUNTER HOLEN
0EA4                    ;HOLE X1,Y1
0EA5                    ;HOLE X2,Y2
0EA6                    ;FERTIG?
0EA8                    ;KILL UPPER LINE (8 RASTERZEILEN)
```

0EAA ---------- COPY RASTER LINE

COPY RASTER LINE kopiert eine vollständige Rasterzeile in einen anderen Bereich. HL muß dabei die Adresse der Quellzeile beinhalten, und DE die Adresse der Zielzeile. Das C-Register ist für die Anzahl der zu kopierenden Bytes verantwortlich.

```
0EAA                    ;COUNTER INIT
0EAC                    ;RA-TEST AUFRUFEN
0EAF                    ;AUFGETRETEN
0EB1                    ;RA-TEST AUFRUFEN
0EB4                    ;INDIRECT COPY?
```

```
0EB6                    ;ANZAHL DER BYTES RETTEN
0EB7                    ;BERECHNE LÄNGE
0EB8                    ;VERGL. 0EB7
0EB9                    ;LÄNGE NACH C
0EBA                    ;ERSTEN TEIL DER ZEILE KOPIEREN
0EBC                    ;HOLE ANZAHL DER BYTES
0EBD                    ;BERECHNE ZAHL DER KOPIERTEN BYTES
0EBE                    ;VERGL 0EBE
0EBF                    ;VERGL 0EBE
0EC0                    ;VERGL 0EBE
0EC1                    ;ERGEBNIS NACH AKKU
0EC2                    ;RASTER ADRESS BITS REKONSTRUIEREN
0EC4                    ;ERGEBNIS NACH H
0EC5                    ;ZWEITEN TEIL DER ZEILE KOPIEREN
0EC7                    ;RA-TEST AUFRUFEN
0ECA                    ;COPY STRAIGHT LINE
0ECC                    ;ANZAHL DER BYTES RETTEN
0ECD                    ;BERECHNE LÄNGE
0ECE                    ;VERGL. 0ECE
0ECF                    ;LÄNGE NACH C
0ED0                    ;ZWEITEN TEIL DER ZEILE KOPIEREN
0ED2                    ;HOLE ANZAHL DER BYTES
0ED3                    ;BERECHNE ZAHL DER KOPIERTEN BYTES
0ED4                    ;VERGL. 0ED4
0ED5                    ;VERGL. 0ED4
0ED6                    ;VERGL. 0ED4
0ED7                    ;ERGEBNIS NACH AKKU
0ED8                    ;RASTER ADRESS BITS REKONSTRUIEREN
0EDA                    ;ERGEBNIS NACH D
0EDB                    ;NORMAL KOPIEREN
0EDD                    ;FERTIG
```

0EDE ---------- COPY STRAIGHT LINE

COPY STRAIGHT LINE kopiert eine vollständige Rasterzeile in einen anderen Bereich. HL muß dabei die Adresse der Quellzeile beinhalten, und DE die Adresse der Zielzeile. Das C-Register ist für die Anzahl der zu kopierenden Bytes verantwortlich.


```
0EDE                    ;HOLE LÄNGE DER ZEILE
0EDF                    ;HOLE QUELL-BYTE
0EE0                    ;KOPIERE ES IN ZIELBYTE
0EE1                    ;SCR NEXT BYTE
0EE4                    ;HL = ZIEL-POINTER
0EE5                    ;SCR NEXT BYTE
0EE8                    ;HL = QUELL-POINTER
0EE9                    ;ZEILE KOPIERT?
0EEB                    ;FERTIG
```

0EEC ---------- RA-TEST

RA-TEST stellt fest, ob ein Übertrag der Raster Adress Bits stattgefunden hat. Als Eingabeparameter erwartet die Routine jeweils in HL und DE eine Bildschirmadresse sowie die Zeilenlänge in C. Falls ein Übertrag stattgefunden hat, befindet sich nach dem Verlassen von RA-TEST das Carry-Bit im High-Zustand.

```
0EEC                    ;A = ANZAHL DER BYTES
0EED                    ;DE = 1. VIDEO-ADRESS
0EEE                    ;COUNTER -1
0EEF                    ;ADDIERE LSB DER 2. VIDEO-ADRESS ZUM
                        ;COUNTER
0EF0                    ;FERTIG?
0EF1                    ;HOLE MSB DER 2. BILDSCHIRMADRESSE
0EF2                    ;ÜBERTRAG ?
0EF4                    ;VERGL 0EF2
0EF6                    ;FERTIG?
0EF7                    ;CARRY = HIGH (OUT PARAMS)
0EF8                    ;FERTIG!
```

0EF9 ---------- SCR UNPACK

SCR UNPACK vergrößert dir Zeichenmatrix für MODE 0 und 1. Die Routine erwartet im HL-Register die Adsresse der kompremierten Matrix (MODE 2), und im DE-Register die Zieladresse für die errechnete Matrix.

```
0EF9                    ;SCR GET MODE
0EFC                    ;MODE 0 GESETZT
0EFE                    ;MODE 1 GESETZT
0F00                    ;ZÄHLER: ACHT BYTES
0F03                    ;KOPIERVORGANG DURCHFÜHREN
0F05                    ;DAS WAR'S

0F06                    ;BIT-COUNTER UND MASKE NACH BC (MODE 0)
0F09                    ;JUMP START
0F0B                    ;BIT-COUNTER UND MASKE NACH BC (MODE 1)
0F0E                    ;COUNTER INITIALISIEREN (8 BYTES
                        ;KOPIEREN)
0F10                    ;COUNTER RETTEN
0F11                    ;POINTER FÜR GEPACKTE MATRIX RETTEN
0F12                    ;L = KOMPREMIERTES MATRIXELEMENT
0F13                    ;BIT-COUNTER MERKEN
0F14                    ;AKKU LÖSCHEN
0F15                    ;HOLE BIT AUS KOMPREMIERTER MATRIX
0F17                    ;BIT VORHANDEN?
0F19                    ;MERGE MATIX
0F1A                    ;HOLE BIT AUS MATRIX
0F1C                    ;BIT VORHANDEN?
0F1E                    ;SPEICHER UNGEPACKTES BYTE
0F1F                    ;POINTER AUF NÄCHSTES BYTE
0F20                    ;ALLE BITS VERARBEITET?
0F22                    ;HOLE BIT-COUNTER
0F23                    ;HOLE POINTER FÜR GEPACKTE MATRIX
0F24                    ;AUF NÄCHSTE ZEILE DER MATRIX ZEIGEN
0F25                    ;HOLE COUNTER
0F26                    ;COUNTER-1
0F27                    ;MATRIX KOPIERT UND KONVERTIERT?
0F29                    ;FERTIG!
```

0F2A ---------- SCR REPACK

Zeichenmatrix wieder auf Originalform stauchen. Die Routine erwartet im H-Register die Spalte, und im L-Register die Reihe der Matrix. Im Akkumulator muß sich weiterhin die PEN-Nummer, und DE-Register die Zieladresse für die kompremierte Matrix befinden.


```
0F2A                    ;HINTERGRUNDFARBE NACH C-REGISTER
0F2B                    ;SCR CHAR POSITION
0F2E                    ;SCR GET MODE
0F31                    ;ES GIBT ACHT RASTERZEILEN
0F33                    ;MODE 0 GESETZT
0F35                    ;MODE 1 GESETZT
0F37                    ;AKKU AUS BILDSCHIRMSPEICHER FÜLLEN
0F38                    ;XOR-VERKNÜPFUNG MIT HINTERGRUNDFARBE
0F39                    ;EINERKOMPLEMENT BILDEN
0F3A                    ;ERGEBNIS NACH (DE) SCHREIBEN
0F3B                    ;ZEIGER ERHÖHEN
0F3C                    ;SCR NEXT LINE
0F3F                    ;WEITER SOLANGE NOCH RASTERZEILEN
0F41                    ;FERTIG

0F42                    ;RETTE DEN INHALT ALLER REGISTER
0F43                    ;VERGL. 0F42
0F44                    ;VERGL. 0F42
0F45                    ;ERSTES BYTE KONVERTIEREN
0F48                    ;SCR NEXT BYTE
0F4B                    ;ZWEITES BYTE KONVERTIEREN
0F4E                    ;AKKU = MATRIXELEMENT
0F4F                    ;HOLE ZIELADRESSE
0F50                    ;MATRIXELEMENT IN ZIELADRESSE SCHREIBEN
0F51                    ;AUF NÄCHSTE ADRESSE ZEIGEN
0F52                    ;HOLE SPALTE UND REIHE
0F53                    ;SCR NEXT LINE
0F56                    ;HOLE COUNTER
0F57                    ;ALLE BYTES ÜBERTRAGEN?
0F59                    ;WENN JA DANN FERTIG

0F5A                    ;C = MASKE (MODE 0)
0F5C                    ;B = BIT-COUNTER
0F5E                    ;HOLE BYTE AUS VIDEO-RAM
0F5F                    ;HINTERGRUNDFARBE AUSMASKIEREN
0F60                    ;MIT MASKE VERBINDEN
0F61                    ;PIXEL = PEN FARBE?
0F63                    ;WENN NEIN, DANN PIXEL AKTIVIEREN
0F64                    ;PIXEL IN MATRIX IMPLEMENTIEREN
```

```
0F66                    ;VERGL. 0F64
0F68                    ;ALLE BITS ÜBERTRAGEN
0F6A                    ;RÜCKSPRUNG

0F6B                    ;RETTE DEN INHALT ALLER REGISTER
0F6C                    ;VERGL. 0F6B
0F6D                    ;VERGL. 0F6B
0F6E                    ;INITIALISIERE COUNTER
0F70                    ;HOLE BYTE AUS VIDEO-RAM
0F71                    ;HINTERGRUNDFARBE AUSMASKIEREN
0F72                    ;MIT MASKE VERBINDEN
0F74                    ;PIXEL = PEN FARBE?
0F76                    ;WENN NEIN, DANN PIXEL AKTIVIEREN
0F77                    ;PIXEL IN MATRIX IMPLEMENTIEREN
0F79                    ;VERGL. 0F77
0F7A                    ;VERGL. 0F77
0F7B                    ;VERGL. 0F77
0F7D                    ;PIXEL VORHANDEN?
0F7F                    ;WENN NICHT, PIXEL AKTIVIEREN
0F80                    ;UND IN MATRIX IMPLEMENTIEREN
0F82                    ;SCR NEXT BYTE
0F85                    ;ALLE BITS ÜBERTRAGEN?
0F87                    ;A = MATRIXELEMENT
0F88                    ;HOLE MATRIX-POINTER
0F89                    ;BYTE IN MATRIX SCHREIBEN
0F8A                    ;POINTER AUF NÄCHSTES BYTE DER MATRIX
0F8B                    ;HOLE VIDEO-ADRESS
0F8C                    ;SCR NEXT LINE
0F8F                    ;HOLE COUNTER
0F90                    ;ALLE RASTERZEILEN ÜBERTRAGEN?
0F92                    ;FERTIG!
```

0F93 ---------- SCR HORIZONTAL

Horizontale Linie ziehen. Die Routine erwartet X1 im DE-Register, X2 im BC-Register und die Y-Koordinate im HL-Register. Im Akkumulator muß sich der Farbmaske befinden.


```
0F93                    ;MASK HANDLING
0F96                    ;DRAW H-LINE
0F99                    ;FORTSETZUNG FOLGT
```

0F9B ---------- SCR VERTICAL

Vertikale Linie ziehen. Die Routine erwartet Y1 im HL-Register, Y2 im BC-Register und die X-Koordinate im DE-Register. Im Akkumulator muß sich der Farbmaske befinden.

```
0F9B                    ;MASK HANDLING
0F9E                    ;DRAW V-LINE
0FA1                    ;ALTE MASK NACH HL-REGISTER
0FA4                    ;LSB ALTE MASK NACH AKKU
0FA5                    ;(GRA PEN)
0FA8                    ;MSB ALTE MASK NACH AKKU
0FA9                    ;MASK-PARAM
0FAC                    ;ERLEDIGT!
```

0FAD ---------- MASK HANDLING

```
0FAD                    ;HL-REGISTER FREISTELLEN
0FAE                    ;(GRA PEN)
0FB1                    ;(GRA PEN)
0FB4                    ;MASK-PARAM NACH AKKU
0FB7                    ;MASK-PARAM NACH H
0FB8                    ;ALLE BITS SETZEN ERGIBT
0FBA                    ;GESCHLOSSENEN LINIENZUG
0FBD                    ;ALTE MASK KONSERVIEREN
0FC0                    ;ALTEN INHALT VON HL-REGISTER
0FC1                    ;DAS WAR'S
```

0FC2 ---------- DRAW H-LINE

```
0FC2                    ;CARRY FÜR H-LINE SETZEN
0FC3                    ;HOLE PARAMERTER
0FC6                    ;LINIENMASKE ERSTELLEN
```

```
0FC8                ;PUNKTMASKE HOLEN
0FC9                ;BIT = PEN-COL
0FCB                ;PIXELANZAHL-1
0FCC                ;ALLE PIXEL ÜBERTRAGEN?
0FCE                ;PIXELANZAHL+1
0FCF                ;ALLE PIXEL ÜBERTRAGEN?
0FD1                ;PIXELMASKE ERSTELLEN
0FD3                ;GRENZE ERREICHT?
0FD5                ;PIXEL HIGH?
0FD7                ;PENCOL OK?
0FD9                ;WENN NICHT PENCOL SETZEN
0FDA                ;PIXELMASKE ERSTELLEN
0FDC                ;HOLE NÄCHSTES PIXEL
0FDE                ;PIXELANZAHL-1
0FDF                ;ALLE PIXEL ÜBERTRAGEN?
0FE1                ;PIXELANZAHL+1
0FE2                ;PENCOL OK?
0FE4                ;PIXELMASKE ERSTELLEN
0FE6                ;PENCOL OK?
0FE8                ;PIXEL HIGH?
0FEA                ;PENCOL OK?
0FEC                ;WENN NICHT PENCOL SETZEN
0FED                ;PIXELMASKE ERSTELLEN
0FEF                ;BIT = PEN-COL
0FF1                ;MASKEN RETTEN
0FF2                ;HOLE PUNKMASKE
0FF3                ;(GRA PAPER)
0FF6                ;IN B MERKEN
0FF7                ;HOLE HINTERGRUND-MODUS
0FFA                ;MODUS LOKALISIEREN
0FFB                ;FERTIGES BYTE IN VIDEO-RAM ÜBERTRAGEN
0FFD                ;RETTE MASKEN
0FFE                ;HOLE PUNKTMASKE
0FFF                ;(GRA PAPER)
1002                ;NACH B ÜBERTRAGEN
1003                ;FLAG-INIT
1007                ;HOLE MASKEN
1008                ;GRENZE ÜBERSCHRITTEN?
100A                ;SCR NEXT BYTE
100D                ;ALLE PIXEL ÜBERTRAGEN
```



```
100E                ;VERGL. 100D
100F                ;VERGL. 100D
1011                ;A=LINE-MASK
1012                ;LINE-MASK MERKEN
1015                ;FERTIG!


1016 ---------------------------------------------------- DRAW V-LINE

1016                ;CARRY FÜR H-LINE SETZEN
1017                ;HOLE PARAMETER
101A                ;LINIENMASKE ERSTELLEN
101C                ;(GRA PEN)
101F                ;BIT = PEN-COL
1021                ;HINTERGRUNDMODUS?
1024                ;TRANZPARENT MODUS AKTIVIERT?
1025                ;WENN NEIN WECHSLE MASKE
1027                ;(GRA PAPER)
102A                ;MASKEN RETTEN
102B                ;COLMASK HOLEN
102C                ;SCR WRITE
102F                ;MASKEN HOLEN
1030                ;SCR PREV LINE
1033                ;NEXT PIXEL
1034                ;VERGL. 1033
1036                ;ALLE PIXEL ÜBERTRAGEN?
1037                ;VERGL. 1036
1039                ;VERGL. 1036
103B                ;RETTE Y-KOORDINATE
103C                ;V-LINE?
103E                ;MSB DER X-KOORDINATE NACH HL
                    ;TRANSFERIEREN
103F                ;LSB DER X-KOORDINATE NACH HL
                    ;TRANSFERIEREN
1040                ;CARRY FÜR SBC ELEMINIEREN
1041                ;KOORDINATE ERMITTELN
1043                ;VERGL. 1041
1046                ;LÄNGE SYNCHRONISIEREN
1047                ;VERGL. 1046
1048                ;UND MERKEN
```

```
1049                    ;SCR DOT POSITION
104C                    ;LINE-MASK HOLEN
104F                    ;IN B-REGISTER ÜBERTRAGEN
1050                    ;SYNCHRONISIERTE LÄNGE HOLEN
1051                    ;DAS WAR'S


1052 ---------------------------------------------- DEFAULT FARBEN

1052                    04 04 0A 13 0C 0B 14 15
105A                    0D 06 1E 1F 07 12 19 04
1062                    17 04 04 0A 13 0C 0B 14
106A                    15 0D 06 1E 1F 07 12 19
1072                    0A 07
```


9.6.3 Text Screen (TXT)

Dieses Pack ist, wie der Name schon sagt, für das Handling von Texten verantwortlich. Zu dieser Arbeit gehört die Organisation der achts Windows, die Darstellung von Zeichen auf dem Bildschirm sowie die Verwaltung des Cursors. Bei der Durchführung dieser Arbeiten bedient sich der TEXT SCREEN weitestgehend der Möglichkeiten des SCREEN PACKs. Das SCREEN PACK ist somit die Exekutive des TEXT SCREEN.

Unmittelbar nach dem Aufruf beginnt der TEXT SCREEN mit der vollständigen Initialisierung des Packs. Dabei werden die Indirektions des TEXT SCREEN in den RAM-Speicher kopiert. Diese Indirektions sind:

```
BDCD                    ;TXT DRAW/UNDRAW CURSOR
                        ;Setzen/Löschen des Cursors.

BDD0                    ;TXT DRAW/UNDRAW CURSOR
                        ;dito

BDD3                    ;TXT WRITE CHAR
                        ;Ein Zeichen auf den Bildschirm
                        ;schreiben.

BDD6                    ;TXT UNWRITE CHAR
                        ;Ein Zeichen vom Bildschirm lesen.

BDD9                    ;TXT OUT ACTION
                        ;Ausgabe eines Zeichens auf dem
                        ;Bildschirm oder Ausführung eines
                        ;Steuercodes.
```

Als zweite Tabelle wird die Steuerzeichen-Sprungtabelle ins RAM kopiert. Diese Tabelle steht ab Adresse &B763.

Zu jedem der acht Windows gehört ein &0E Byte großer Parameterblock, der ebenfalls vom TEXT SCREEN auf seinen Default-Wert gebracht wird. Zuerst werden mit der Routine DEFAULT PARAMS SETZEN die Fensterparameter auf ihren

"Anfangswert" gebracht. Im Registerpaar HL werden die nötigen Informationen zur Farbwahl an DEFAULT PARAMS SETZEN übergeben. H enthält den Wert für die Hintergrundfarbe und L den Wert für die Vordergrundfarbe. Ist auf diese Weise ein Parameterblock initialisiert, so wird dieser Parameterblock mit RESET PARAMS (ALLE FENSTER) auf alle acht Windows bezogen. Damit ist die Initialisierung des TEXT SCREEN durchgeführt.

Die in den Cursor-Routinen verlangten oder gelieferten Koordinaten sind als logische Angeben zu verstehen, d.h. sie beziehen sich auf das laufende Fenster. Die Koordinate 1,1 ist dabei die linke obere Ecke des Fensters. Wollen Sie, z.B. mit TXT SET CURSOR, den Cursor außerhalb des Fensters positionieren, wird er automatisch auf die nächste mögliche Position innerhalb des Fensters gesetzt, falls der Cursor eingeschaltet ist oder nachfolgend ein Zeichen dargestellt werden soll. Dadurch wird auch die laufende Position (die Sie mit TXT GET CURSOR zurückbekommen) geändert.

Ist der Cursor ausgeschaltet, wird die gewünschte neue Position zunächst akzeptiert, bis entweder ein Zeichen dargestellt oder der Cursor eingeschaltet wird.

```
1074 ------------------------------------------------- TXT INITIALISE

Vollständige Initialisierung des TEXT PACKs.

1074                    ;TXT RESET
1077                    ;AKKUMULATOR WIRD GELÖSCHT
1078                    ;FLAG USER-MATRIX LÖSCHEN
107B                    ;PAPER=0 UND PEN=1
107E                    ;DEFAULT PARAMS SETZEN
1081                    ;RESET PARAMS (ALLE FENSTER)
```



```
1084 ------------------------------------------------- TXT RESET

Die Indirektions des TEXT PACKs werden ins RAM (ab Adresse &BDCD)
kopiert.

1084                    ;RESTORE TXT INDIRECTIONS
1087                    ;MOVE (HL+3) NACH ((HL+1)), CNT=(HL)
108A                    ;DEFAULT STEUERZEICHEN SPRÜNGE KOPIEREN
108D                    ;15 BYTES
108E                    ;ZIELADRESSE

1090                    ;CODE FÜR JP
1091                    ;TXT DRAW/UNDRAW CURSOR

1093                    ;CODE FÜR JP
1094                    ;TXT DRAW/UNDRAW CURSOR

1096                    ;CODE FÜR JP
1097                    ;TXT WRITE CHAR

1099                    ;CODE FÜR JP
109A                    ;TXT UNWRITE CHAR

109C                    ;CODE FÜR JP
109D                    ;TXT OUT ACTION

109F ------------------------------------ RESET PARAMS (ALLE FENSTER)

Bei DEFAULT PARAMS SETZEN (&1139) werden die Fensterparameter auf ihren
Anfangswert gebracht. Ist das geschehen, so werden diese Parameter auf
alle acht Fenster bezogen (vgl. &107E und &1081).

109F                    ;ANZAHL DER FENSTER
10A1                    ;ZIEL: START PARAMETER FENSTER 0
10A4                    ;QUELLE: DIVERSE PARAMETER FÜR FENSTER
                        ;&B726  CURSOR POSITION (ROW, COL)
                        ;&B728  SCROLL FLAG
                        ;&B729  FENSTER OBEN
                        ;&B72A  FENSTER LINKS
```

```
                    ;&B72B  FENSTER UNTEN
                    ;&B72C  FENSTER RECHTS
                    ;&B72D  ROLL COUNT
                    ;&B72E  CUSOR FLAG
                    ;&B72F  PEN
                    ;&B730  PAPER
                    ;&B731  BACKGROUND MODE
                    ;&B733  GRAPH CHAR WRITE MODE
                    ;&B734  1. ZEICHEN USER MATRIX
10A7                ;ZÄHLER: ANZAHL ZU KOPIERENDE BYTES
10AA                ;KOPIERVORGANG AUSFÜHREN
10AC                ;NÄCHSTES FENSTER
10AD                ;SPRINGE, WENN NOCH EIN FENSTER
10AF                ;(LFD. BILDSCHIRMFENSTER)
10B2                ;ERLEDIGT
```

10B3 ---------- FARBEN DECODIEREN

Beim Verlassen dieser Routine steht im C-Register die Kennziffer für das aktuelle Bildschirmfenster.

```
10B3                ;(LFD. BILDSCHIRMFENSTER)
10B6                ;AKKU FREI FÜR WEITERE VERWENDUNG
10B7                ;ES GIBT ACHT WINDOWS
10B9                ;AKKU DIENT ALS ZÄHLER
10BA                ;ZÄHLER-1 (FENSTER 0 BIS 7)
10BB                ;TXT STR SELECT
10BE                ;TXT DRAW/UNDRAW CURSOR
10C1                ;TXT GET PAPER
10C4                ;LFD. PAPER
10C7                ;TXT GET PEN
10CA                :(LFD. PEN)
10CD                ;SPRINGE, WENN NOCH EIN FENSTER
10CF                ;LFD. BILDSCHIRMFENSTER NACH AKKU
10D0                ;FERTIG
```


10D1 ---------- FENSTERPARAMETER AUF DEFAULT

Die Fensternummer wird im Akkumulator an die Routine übergeben

```
10D1                ;FENSTERNUMMER NACH C
10D2                ;ES GIBT ACHT FENSTER
10D4                ;AKKU DIENT ALS ZÄHLER
10D5                ;ZÄHLER-1 (FENSTER 0 BIS 7)
10D6                ;TXT STR SELECT
10D9                ;REGISTERINHALTE SICHERN
10DA                ;(TXT LFD PEN)
10DD                ;DEFAULT PARAMS SETZEN
10E0                ;REGISTERINHALTE WIEDER HERSTELLEN
10E1                ;SPRINGE, WENN NOCH EIN FENSTER
10E3                ;FENSTERNUMMER NACH AKKU
```

10E4 ---------- TXT STR SELECT

Mit TXT STR SELECT kann ein Textfenster ausgewählt werden. Die Nummer des neuen Textfensters muß beim Aufruf von TXT STR SELCET im Akkumulator stehen. Beim Verlassen der Routine steht die Nummer des Textfensters im Akku, welches vor dem Aufruf aktiv war.

```
10E4                ;FENSTERNUMMER IM BEREICH 0 BIS 7
10E6                ;LFD. BILDSCHIRMFENSTER
10E9                ;IST DIESES FENSTER SCHON AKTIVIERT?
10EA                ;DANN GIBT'S NICHTS MEHR ZU TUN
10EB                ;REGISTERINHALTE SICHERN
10EC                ;REGISTERINHALTE SICHERN
10ED                ;NUMMER DES AKTUELLEN BILDSCHIRMFENSTERS
                    ;NACH HL
10EE                ;NUMMER DES NEUEN FENSTERS
10EF                ;NUMMER AUCH NACH B
10F0                ;ALTE NUMMER IN DEN AKKUMULATOR
10F1                ;ADR. FENSTER PARAMS NACH DE
10F4                ;LDIR CNT=14
10F7                ;NEUE NUMMER IN DEN AKKUMULATOR
10F8                ;ADR. FENSTER PARAMS NACH DE
10FB                ;FÜR DEN KOPIERVORGANG VERTAUSCHEN
```

```
10FC                    ;LDIR CNT=14
10FF                    ;ALTE FENSTERNUMMER WIRD IM AKKU
                        ;BEREITGESTELLT
1100                    ;REGISTERINHALTE WIEDER HERSTELLEN
1101                    ;REGISTERINHALTE WIEDER HERSTELLEN
1102                    ;ZURÜCK


1103 ---------------------------------------------- TXT SWAP STREAMS
```

Die Parameter (Farben, Fenstergrenzen usw.) zweier Fenster werden miteinander vertauscht. Dazu benötigt TXT swap STREAMS die Nummer der beiden Fenster, deren Parameter vertauscht werden sollen. Die Nummer des einen Fensters seht beim Aufruf der Routine im B-Register, die des anderen im C-Register.

```
1103                    ;(LFD. BILDSCHIRMFENSTER)
1106                    ;AUF DEN STACK SICHERN
1107                    ;2. FENSTERNUMMER NACH AKKU
1108                    ;TXT STR SELECT
110B                    ;1. FENSTERNUMMER NACH AKKU
110C                    ;(LFD. BILDSCHIRMFENSTER)
110F                    ;ADR. FENSTER PARAMS NACH DE
1112                    ;ADR. FENSTER PARAMS SICHERN
1113                    ;2. FENSTERNUMMER NACH AKKU
1114                    ;ADR. FENSTER PARAMS NACH DE
1117                    ;ADR. FENSTER PARAMS HOLEN
1118                    ;LDIR CNT=14
111B                    ;HOLEN LFD. BILDSCHIRMFENSTER
111C                    ;TXT STR SELECT


111E ------------------------------------------------------ LDIR CNT=14
```

Kopiert 14 Bytes (Parameter für Fenster).

Die Anzahl der zu kopierenden Bytes ist auf 14 festgelegt (BC), hingegen Quelle und Ziel frei wählbar sind. Beim Aufruf der Routine muß HL mit der Quelladresse geladen sein und DE die Zieladresse enthalten. Nur so

kann der LDIR sinnvoll ausgeführt werden. Der Inhalt von BC vor dem Aufruf der Routine ist bei ihrem Verlassen wieder hergestellt.

```
111E                    ;BC ZUR WEITEREN VERWENDUNG SICHERN
111F                    ;ZÄHLER: ANZAHL ZU KOPIERENDE BYTES
1122                    ;KOPIERVORGANG DURCHFÜHREN
1124                    ;BC WIE VOR DEM ANSPRUNG
1125                    ;ZURÜCK


1126 ------------------------------------ ADR. FENSTER PARAMS NACH DE
```

Wird dieser Routine im Akkumulator eine Fensternummer übergeben, so stellt sie beim Verlassen im DE-Register die Adresse der zugehörigen Parameter zur Verfügung. Ebenso befindet sich im HL-Register die Adresse der zum aktuellen Fenster gehörigen Parameter.

```
1126                    ;FENSTERNUMMER IM BEREICH 0 BIS 7
1128                    ;A NACH E SICHERN
1129                    ;2A (ERGEBNIS IN A)
112A                    ;3A (ERGEBNIS IN A)
112B                    ;6A (ERGEBNIS IN A)
112C                    ;7A (ERGEBNIS IN A)
112D                    ;14A (ERGEBNIS IN A)
112E                    ;ADDIERE LSB DER BASISADRESSE
1130                    ;LSB DER PARAMETERTABELLE
1131                    ;EVENTUELLER LSB ÜBERLAUF
1133                    ;BEI ERMITTLUNG DES MSB DER
                        ;PARAMETERTABELLE EINBEZIEHEN
1134                    ;DE=ANFANGSADRESSE DER PARAMETERTABELLE
                        ;(&0E BYTES)
1135                    ;LFD. CURSOR POSITION (ROW, COL)
1138                    ;FERTIG


1139 ------------------------------------------ DEFAULT PARAMS SETZEN
```

Hier werden die Fensterparameter auf ihren "Anfangswert" gebracht. Im Registerpaar HL werden die nötigen Informationen zur Farbwahl an DEFAULT

PARAMS SETZEN übergeben. H enthält den Wert für die Hintergrundfarbe und L den Wert für die Vordergrundfarbe.

```
1139                    ;PAPER NACH D UND PEN NACH E
113A                    ;WERT FÜR LFD. CURSOR-FLAG
113C                    ;(TXT LFD. CURSOR-FLAG)
113F                    ;PAPER NACH AKKU
1140                    ;TXT SET PAPER
1143                    ;PEN NACH AKKU
1144                    ;TXT SET PEN
1147                    ;AKKU LÖSCHEN UND FLAGS ZURÜCKSETZEN
1148                    ;TXT SET GRAPHIC
114B                    ;TXT SET BACK
114E                    ;FENSTERGRENZEN LINKS OBEN
1151                    ;FENSTERGRENZEN RECHTS UNTEN
1154                    ;TXT WIN ENABLE
1157                    ;TXT VDU ENABLE
```

115A ---------- TXT SET COLUMN

TXT SET COLUMN setzt die horizontale Position des Cursors. Im Akkumulator muß der Wert für die Spalte übergeben werden.

```
115A                    ;UM EINS ERNIEDRIGEN
115B                    ;LFD. FENSTER LINKS
115E                    ;ERGIBT ABSOLUTE POSITION
115F                    ;(LFD. CURSOR POSITION (ROW, COL))
1162                    ;NEUE POSITION IN DER SPALTE
1163                    ;AUF ZUR CURSORPOSITIONIERUNG
```

1165 ---------- TXT SET ROW

TXT SET ROW setzt die vertikale Position des Cursors. Im Akkumulator muß der Wert für die Zeile übergeben werden.

```
1165                    ;UM EINS ERNIEDRIGEN
1166                    ;LFD. FENSTER OBEN
1169                    ;ERGIBT ABSOLUTE POSITION
```



```
116A                    ;(LFD. CURSOR POSITION (ROW, COL))
116D                    ;NEUE POSITION IN DER ZEILE
116E                    ;ZUR CURSORPOSITIONIERUNG
```

1170 ---------- TXT SET CURSOR

TXT SET CURSOR positioniert den Cursor. Vor dem Aufruf der Routine muß H mit dem Wert für die Spalte und L mit dem Wert für die Zeile geladen werden.

```
1170                    ;LFD. FENSTER OBEN, LINKS + HL
1173                    ;TXT DRAW/UNDRAW CURSOR
1176                    ;(LFD. CURSOR POSITION (ROW, COL))
1179                    ;TXT DRAW/UNDRAW CURSOR
```

117C ---------- TXT GET CURSOR

Mit dieser Routine kann die monentane Cursorposition ermittelt werden. Beim Verlassen von TXT GET CURSOR enthält H der Wert für die Spalte und L den Wert für die Zeile. Darüber hinaus enthält der Akku LFD. ROLL COUNT (Scrolling-Zähler).

```
117C                    ;(LFD. CURSOR POSITION (ROW, COL))
117F                    ;LFD. FENSTER OBEN, LINKS - HL
1182                    ;(LFD. ROLL COUNT)
1185                    ;FERTIG
```

1186 ---------- LFD. FENSTER OBEN, LINKS + HL

Diese Routine verwandelt eine relative in eine absolute Position. Sowohl beim Aufruf als auch beim Verlassen der Routine enthält H die Angabe für die Spalte und L die Angabe für die Zeile.

```
1186                    ;(LFD. FENSTER OBEN)
1189                    ;LFD. FENSTER OBEN - 1 ERGIBT OFFSET
118A                    ;OFFSET + ZEILE
118B                    ;ERGIBT ABSOLUTE ZEILENPOSITION
```

```
118C                    ;(LFD. FENSTER LINKS)
118F                    ;LFD. FENSTER LINKS - 1 ERGIBT OFFSET
1190                    ;OFFSET + SPALTE
1191                    ;ERGIBT ABSOLUTE SPALTENPOSITION
1192                    ;ZURÜCK
```

1193 ---------- LFD. FENSTER OBEN, LINKS - HL

Diese Routine verwandelt eine absolute in eine relative Position. Sowohl beim Aufruf als auch beim Verlassen der Routine enthält H die Angabe für die Spalte und L die Angabe für die Zeile.

```
1193                    ;(LFD. FENSTER OBEN)
1196                    ;LFD. FENSTER OBEN - ZEILE
1197                    ;ZWEIERKOMPLEMENT BILDEN
1198                    ;ZUM INHALT DES AKKUMULATOR ZWEI
1199                    ;HINZUADDIEREN
119A                    ;ERGIBT RELATIVE ZEILENPOSITION
119B                    ;(LFD. FENSTER LINKS)
119E                    ;LFD. FENSTER LINKS - SPALTE
119F                    ;ZWEIERKOMPLEMENT BILDEN
11A0                    ;ZUM INHALT DES AKKUMLATOR ZWEI
11A1                    ;HINZUADDIEREN
11A2                    ;ERGIBT RELATIVE SPALTENPOSITION
11A3                    ;ZURÜCK
```

11A4 ---------- CURSOR INVERTIEREN

Vor dem Durchlaufen der Routine POSITION DES CURSORS ÜBERPRÜFEN wird hier mittels TXT DRAW/UNDRAW CURSOR der Cursor invertiert.

```
11A4                    ;TXT DRAW/UNDRAW CURSOR
```

11A7 ---------- POSITION DES CURSORS ÜBERPRÜFEN

Beim Verlassen von POSITION DES CURSORS ÜBERPRÜFEN steht im Registerpaar HL die Position des Cursors zu Verfügung (H=Spalte, L=Zeile).


```
11A7                    ;(LFD. CURSOR POSITION (ROW,COL))
11AA                    ;HL INNERHALB FENSTERGRENZEN
11AD                    ;(LFD. CURSOR POSITION (ROW,COL))
11B0                    ;HL LAG IM ZULÄSSIGEN BEREICH
11B1                    ;POSITION DES CURSORS AUF DEN STACK
11B2                    ;LFD. ROLL COUNT
11B5                    ;B HAT EINEN DER WERTE &00 ODER &FF
                        ;B=&00 HEISST NACH UNTEN SCROLLEN
                        ;B=&FF HEISST NACH OBEN SCROLLEN
11B6                    ;A=&00 ODER A=&FE
11B7                    ;A=&01 ODER A=&FF
11B8                    ;LFD. ROLL COUNT HINZUADDIEREN
11B9                    ;NEUER ROLL COUNT
11BA                    ;TXT GET WINDOW
11BD                    ;(LFD. PAPER)
11C0                    ;LFD. PAPER UND CARRY SICHERN
11C1                    ;SCR SW ROLL
11C4                    ;LFD. PAPER UND CARRY HOLEN
11C5                    ;SCR HW ROLL
11C8                    ;POSITION DES CURSORS VOM STACK
11C9                    ;FERTIG
```

11CA ---------- TXT VALIDATE

TXT VALIDATE ermittelt, ob sich der Cursor innerhalb des Textfensters befindet. Sowohl beim Aufruf, als auch beim Verlassen dieser Routine, enthält das H-Register den Wert für die Spalte und das L-Register den Wert für die Zeile. Ist beim Verlassen von TXT VALTIDATE das CARRY-Flag gesetzt, so befindet sich der Cursor im zulässigen Bereich. Nicht gesetztes CARRY bedeutet Scrolling.

```
11CA                    ;LFD. FENSTER OBEN, LINKS + HL
11CD                    ;HL INNERHALB FENSTERGRENZEN
11D0                    ;AKKU UND FLAGS SICHERN
11D1                    ;LFD. FENSTER OBEN, LINKS - HL
11D4                    ;AKKU UND FLAGS WIE GEWESEN
11D5                    ;ZURÜCK
```

11D6 ---------- HL INNERHALB FENSTERGRENZEN

Diese Routine bringt die im Registerpaar HL übergebene Spalten-/Zeilen-Position in die Fenstergrenzen und stellt die Position ebenfalls wieder in HL zur Verfügung. Ist beim Verlassen von HL INNERHALB FENSTERGRENZEN das Carry-Flag gesetzt, so ist kein Scrollen des Bildschirms notwendig, um die gegebene Position darzustellen. Ist CARRY nicht gesetzt und enthält das B-Register &FF, so muß nach oben gescrollt werden; enthält das B-Register &00, so muß nach unten gescrollt werden.

```
11D6                    ;(TXT LFD. FENSTER RECHTS)
11D9                    ;SPALTE NICHT GRÖSSER ALS RECHTER RAND
11DA                    ;DANN SPRINGE
11DD                    ;(TXT LFD. FENSTER LINKS)
11E0                    ;WERT NACH H WEITERREICHEN
11E1                    ;ZEILE UM 1 ERHÖHEN
11E2                    ;(TXT LFD. FENSTER LINKS)
11E5                    ;LFD. FENSTER LINKS ERNIEDRIGEN
11E6                    ;SPALTE GRÖSSER AKKUMULATOR?
11E7                    ;DANN SPRINGE
11EA                    ;(TXT LFD. FENSTER RECHTS)
11ED                    ;WERT NACH H WEITERREICHEN
11EE                    ;ZEILE UM 1 ERNIEDRIGEN
11EF                    ;(TXT LFD. FENSTER OBEN)
11F2                    ;LFD. FENSTER OBEN ERNIEDRIGEN
11F3                    ;ZEILE KLEINER ODER GLEICH AKKU
11F4                    ;DAS DARF SO NICHT SEIN! KORREKTUR
                        ;DURCHFÜHREN
11F7                    ;(TXT LFD. FENSTER UNTEN)
11FA                    ;ZEILE KLEINER ODER GLEICH AKKU
11FB                    ;CARRY SETZEN (KEIN SCROLLING)
11FC                    ;UNTER DIESER PRÄMISSE WAR'S DAS
11FD                    ;TXT LFD. FENSTER UNTEN WIRD ZEILE
11FE                    ;ES WIRD NACH OBEN GESCROLLT
1200                    ;CARRY ZURÜCKSETZEN (SCROLLING)
1201                    ;FERTIG
```


1202 ---------- KORREKTUR DURCHFÜHREN

```
1202                    ;TXT LFD. FENSTER OBEN WIEDERHERSTELLEN
1203                    ;TXT LFD. FENSTER OBEN WIRD ZEILE
1204                    ;ES WIRD NACH UNTEN GESCROLLT
1206                    ;CARRY ZURÜCKSETZEN (SCROLLING)
1207                    ;FERTIG
```

1208 ---------- TXT WIN ENABLE

Hier wird die Größe des aktuellen Textfensters festlegen. Vor dem Aufruf von TXT WIN ENABLE werden die Registerpaare HL und DE mit Werten für die Fenstergröße versorgt. H = linke Grenze, L = obere Grenze, D = rechte Grenze und E = untere Grenze.

```
1208                    ;SCR CHAR LIMITS
120B                    ;LINKE GRENZE NACH AKKUMULATOR
120C                    ;BEREICHSÜBERSCHREITUNG SPLATE
120F                    ;MANIPULIERTEN WERT NACH LINKE GRENZE
1210                    ;RECHTE GRENZE NACH AKKUMULATOR
1211                    ;BEREICHSÜBERSCHREITUNG SPLATE
1214                    ;MANIPULIERTEN WERT NACH RECHTE GRENZE
1215                    ;GRENZEN RECHTS LINKS IN ORDNUNG?
1216                    ;VERTAUSCHEN ÜBERSPRINGEN
1218                    ;LINKE GRENZE NACH RECHTE GRENZE
1219                    ;RECHTE GRENZE NACH LINKE GRENZE
121A                    ;OBERE GRENZE NACH AKKUMULATOR
121B                    ;BEREICHSÜBERSCHREITUNG ZEILE
121E                    ;MANIPULIERTER WERT NACH OBERE GRENZE
121F                    ;UNTERE GRENZE NACH AKKUMULATOR
1220                    ;BEREICHSÜBERSCHREITUNG ZEILE
1223                    ;MANIPULIERTER WERT NACH UNTERE GRENZE
1224                    ;GRENZEN OBEN UNTEN IN ORDNUNG
1225                    ;VERTAUSCHEN ÜBERSPRINGEN
1227                    ;OBERE GRENZE NACH UNTERE GRENZE
1228                    ;UNTERE GRENZE NACH OBERE GRENZE
1229                    ;LINKE GRENZE, OBERE GRENZE SCHREIBEN
122C                    ;RECHTE GRENZE, UNTERE GRENZE SCHREIBEN
1230                    ;LINKE GRENZE NACH AKKUMULATOR
```

```
1231                    ;LINKE GRENZE OR OBERE GRENZE
1232                    ;SW ROLL
1234                    ;RECHTE GRENZE NACH AKKUMULATOR
1235                    ;RECHTE GRENZE XOR GRÖSSTER ZULÄSSIGER
                        ;WERT
1236                    ;SW ROLL
1238                    ;OBERE GRENZE NACH AKKUMULATOR
1239                    ;OBERE GRENZE XOR GRÖSSTER ZULÄSSIGER
                        ;WERT
123A                    ;(TXT SCROLL FLAG)
123D                    ;AUF ZUR CURSORPOSITIONIERUNG
```

1240 ---------- BEREICHSÜBERSCHREITUNG SPALTE

Für die Ein- und Ausgabe sind die Register A und B relevant. Im Akkumulator befindet sich sowohl bei der Ein- als auch bei der Ausgabe die Spaltengrenze eines Fensters; Das B-Register enthält den größten zulässigen Wert für die Spaltengrenze.

```
1240                    ;SPALTENGRENZE GRÖSSER ODER GLEICH NULL
1241                    ;UNTER DIESER PRÄMISSE SPRINGE
1244                    ;NULL FÜR SPALTENGRENZE
1245                    ;SPALTENGRENZE KLEINER GRÖSSTER
                        ;ZULÄSSIGER WERT?
1246                    ;TRIFFT ZU, DANN FERTIG
1247                    ;GRÖSSTER ZULÄSSIGER WERT WIRD
                        ;SPALTENGRENZE
1248                    ;FERTIG
```

1249 ---------- BEREICHSÜBERSCHREITUNG ZEILE

Für die Ein- und Ausgabe sind die Register A und C relevant. Im Akkumulator befindet sich sowohl bei der Ein- als auch bei der Ausgabe die Zeilengrenze eines Fensters; Das C-Register enthält den größten zulässigen Wert für die Zeilengrenze.

```
1249                    ;ZEILENGRENZE GRÖSSER ODER GLEICH NULL
124A                    ;UNTER DIESER PRÄMISSE SPRINGE
```



```
124D                    ;NULL FÜR ZEILENGRENZE
124E                    ;ZEILENGRENZE KLEINER GRÖSSTER
                        ;ZULÄSSIGER WERT
124F                    ;TRIFFT ZU, DANN FERTIG
1250                    ;GRÖSSTER ZULÄSSIGER WERT WIRD
                        ;ZEILENGRENZE
1251                    ;ZURÜCK
```

1252 ---------- TXT GET WINDOW

TXT GET WINDOW gibt Antwort auf die Frage nach der Größe des aktuellen Textfensters. Beim Verlassen der Routine enthalten die Registerpaare HL und DE die Werte für die Fenstergröße. H = linke Grenze, L = obere Grenze, D = rechte Grenze und E = untere Grenze.

```
1252                    ;(TXT LFD. FENSTER OBEN)
1255                    ;(TXT LFD. FENSTER OBEN)
1259                    ;(TXT SCROLL FLAG)
125C                    ;CARRY GELÖSCHT FÜR HW-ROLL
125E                    ;DAS WAR'S!
```

125F ---------- TXT DRAW/UNDRAW CURSOR

Setzen/Löschen des Cursors.

```
125F                    ;(TXT LFD. CURSOR FLAG)
1262                    ;CURSOR NICHT ERLAUBT ODER VERRIEGELT
1264                    ;DANN ZURÜCK
```

1265 ---------- TXT PLACE/REMOVE CURSOR

Cursor auf den Bildschirm setzen/Cursor vom Bildschirm nehmen

```
1265                    ;REGISTERPAAR BC SICHERN
1266                    ;REGISTERPAAR DE SICHERN
1267                    ;REGISTERPAAR HL SICHERN
1268                    ;POSITION DES CURSORS ÜBERPRÜFEN
```

```
126B                    ;(TXT LFD. PEN)
                        ;B=LFD. PAPER
                        ;C=LFD. PEN
126F                    ;SCR CHAR INVERT
1272                    ;ALTEN INHALT VON HL
1273                    ;ALTEN INHALT VON DE
1274                    ;ALTEN INHALT VON BC
1275                    ;DAS WAR'S
```

1276 ---------- TXT CUR ON

Cursor erlauben (Betriebssystem).

```
1276                    ;AKKU UND FLAGS AUF STACK SICHERN
1277                    ;BIT NUMMER 1 IST GELÖSCHT
1279                    ;CUR ENABLE CONT'D
127C                    ;AKKU UND FLAGS VOM STACK HOLEN
127D                    ;FERTIG
```

127E ---------- TXT CUR OFF

Cursor verriegeln (Betriebssystem, höhere Priorität als TXT CUR ENABLE und TXT CUR DISABLE).

```
127E                    ;AKKU UND FLAGS AUF STACK SICHERN
127F                    ;BIT NUMMER 1 IST GESETZT
1281                    ;CUR DISABLE CONT'D
1284                    ;AKKU UND FLAGS VOM STACK HOLEN
1285                    ;FERTIG
```

1286 ---------- TXT CUR ENABLE

Cursor erlauben (Anwenderprogramm).

```
1286                    ;BIT NUMMER 0 IST GELÖSCHT
```


1288 ---------- CUR ENABLE CONT'D

```
1288                    ;AKKU UND FLAGS AUF STACK SICHERN
1289                    ;TXT DRAW/UNDRAW CURSOR
128C                    ;AKKU UND FLAGS VOM STACK HOLEN
128D                    ;REGISTERPAAR HL SICHERN
128E                    ;TXT LFD. CURSOR FLAG
1291                    ;BITMANIPLUATION
1292                    ;TXT LFD. CURSOR FLAG ZURÜCKSCHREIBEN
1293                    ;REGISTERPAAR HL HOLEN
1294                    ;TXT DRAW/UNDRAW CURSOR
```

1297 ---------- TXT CUR DISABLE

Cursor verriegeln (Anwenderprogramm).

```
1297                    ;BIT NUMMER 0 SETZEN
```

1299 ---------- CUR DISABLE CONT'D

```
1299                    ;AKKU UND FLAGS AUF STACK SICHERN
129A                    ;TXT DRAW/UNDRAW CURSOR
129D                    ;AKKU UND FLAGS VOM STACK
129E                    ;REGISTERPAAR HL SICHERN
129F                    ;TXT LFD. CURSOR FLAG
12A2                    ;BITMANIPULATION
12A3                    ;TXT LFD. CURSOR FLAG ZURÜCKSCHREIBEN
12A4                    ;WERT FÜR HL VOM STACK
12A5                    ;DAS WAR'S
```

12A6 ---------- TXT SET PEN

TXT SET PEN setzt die Vordergrundfarbe. Im Akkumulator wird der Wert für die Vordergrundfarbe an die Routine übergeben.

```
12A6                    ;TXT LFD. PEN
12A9                    ;ZUR DURCHFÜHRUNG
```

12AB ---------- TXT SET PAPER

TXT SET PAPER setzt die Hintergrundfarbe. Im Akkumulator wird der Wert für die Hintergrundfarbe an die Routine übergeben.

```
12AB                    ;TXT LFD. PAPER
12AE                    ;FARBWERT AUF DEM STACK SICHERN
12AF                    ;TXT DRAW/UNDRAW CURSOR
12B2                    ;FARBWERT WIEDERHOLEN
12B3                    ;SCR INK ENCODE
12B6                    ;BERECHNETEN FARBWERT SETZEN
12B7                    ;TXT DRAW/UNDRAW CURSOR
```

12BA ---------- TXT GET PEN

TXT GET PEN ermittelt die Vordergrundfarbe, deren Nummer beim Verlassen der Routine im Akku steht.

```
12BA                    ;(TXT LFD. PEN)
12BD                    ;SCR INK DECODE
```

12C0 ---------- TXT GET PAPER

TXT GET PAPER ermittelt die Hintergrundfarbe, deren Nummer beim Verlassen der Routine im Akku steht.

```
12C0                    ;(TXT LFD. PAPER)
12C3                    ;SCR INK DECODE
```

12C6 ---------- TXT INVERSE

Diese Routine vertauscht die aktuelle Vordergrundfarbe mit der aktuellen Hintergrundfarbe.

```
12C6                    ;TXT DRAW/UNDRAW CURSOR
```



```
12C9                    ;LFD. PAPER/LFD. PEN NACH HL
12CC                    ;PAPER NACH AKKU
12CD                    ;PEN NACH PAPER
12CE                    ;AKKU NACH PEN
12CF                    ;VERTAUSCHTES PAPER/PEN
12D2                    ;TXT DRAW/UNDRAW CURSOR
```

12D4 ---------- TXT GET MATRIX

Adresse des Punktmusters eines Zeichens holen. Im Akkumulator wird die Nummer des Zeichens an TXT GET MATRIX übergeben. Als Ergebnis kommt im HL-Register die Adresse der Punktmatrix zurück.

```
12D4                    ;DE-REGISTER BEREITHALTEN
12D5                    ;NUMMER DES ZEICHENS IN E BUFFERN
12D6                    ;TXT GET M TABLE
12D9                    ;SPRINGE, WENN KEINE MATRIX
12DB                    ;ERSTES ZEICHEN DER MATRIX NACH D
12DC                    ;GEBUFFERTES ZEICHEN ZURÜCK NACH A
12DD                    ;ERSTES ZEICHEN VON ZEICHEN SUBTRAHIEREN
12DE                    ;KOMPLEMENTIERE CARRY-FLAG
12DF                    ;SPRINGE, WENN ZEICHEN NICHT VORHANDEN
12E1                    ;NUMMER IN DER MATRIX
12E2                    ;ZUR BERECHNUNG DER ADRESSE
12E4                    ;HIER BEGINNT DER ZEICHENSATZ IM ROM
12E7                    ;AKKU UND FLAGS AUF STACK SICHERN
12E8                    ;D ZURÜCKSETZEN
12EA                    ;DE=BASISADRESSE ROM-ZEICHENSATZ H=&00
                        ;UND L=NUMMER IN DER MATRIX
12E8                    ;2HL (ERGEBNIS IN HL)
12EC                    ;4HL (ERGEBNIS IN HL)
12ED                    ;8HL (ERGEBNIS IN HL)
12EE                    ;BASISADRESSE HINZUADDIEREN
12EF                    ;AKKU UND FLAGS VOM STACK HOLEN
12F0                    ;DE WIEDER HERSTELLEN
12F1                    ;FERTIG
```

12F2 ---------- TXT SET MATRIX

TXT SET MATRIX bindet eine vom Benutzer definierte Zeichenmatrix in den Zeichensatz ein. Beim Aufruf von TXT SET MATRIX steht im Akkumulator die Nummer des Zeichens und im Registerpaar HL die Adresse der zugehörigen Zeichenmatrix.

```
12F2                    ;VOR DEM AUFRUF VON TXT GET MATRIX HL-
                        ;REGISTER FREISTELLEN
12F3                    ;TXT GET MATRIX
12F6                    ;ZURÜCK, WENN KEINE MATRIX
12F7                    ;DE IST ZIEL FÜR ZEICHENMATRIX HL IST
                        ;QUELLE
12F8                    ;ZÄHLER: ACHT BYTES KOPIEREN
12FB                    ;KOPIERVORGANG DURCHFÜHREN
12FD                    ;DAS WAR'S
```

12FE ---------- TXT SET M TABLE

Startadresse und erstes Zeichen einer vom Anwender definierten Punktmatrix setzen. Beim Aufruf der Routine TXT SET M TABLE steht im HL-Register der Pointer auf die vom Benutzer definierte Zeichen-Matrix. Enthält das D-Register den Wert Null, dann gibt es eine Benutzer-Matrix. Das E-Register enthält den Wert des ersten Zeichens.

Beim Verlassen von TXT SET M TABLE ist das CARRY-Flag gesetzt, wenn eine alte Benutzer-Matrix vorhanden war. Im Registerpaar HL befindet sich die Adresse dieser alten Benutzer-Matrix und im Akkumulator ihr erstes Zeichen.

```
12FE                    ;POINTER AUF MATRIX SICHERN
12FF                    ;D NACH A (FLAG)
1300                    ;GIBT ES EINE BENUTZER-MATRIX?
1301                    ;D-REGISTER WIRD &00 (KEINE MATRIX)
1303                    ;SPRINGE, WENN KEINE MATRIX
1305                    ;D-REGISTER WIRD &FF (MATRIX)
1306                    ;D (FLAG) UND E (ERSTES ZEICHEN
                        ;INNERHALB DER BENUTZER-MATRIX) SICHERN
1307                    ;ERSTES ZEICHEN NACH C
```



```
1308                    ;POINTER AUF MATRIX NACH DE BRINGEN
1309                    ;ERSTES ZEICHEN NACH AKKUMULATOR
130A                    ;TXT GET MATRIX
130D                    ;MSB MATRIX NACH AKKUMULATOR
130E                    ;XOR MIT MSB POINTER AUF MATRIX
130F                    ;QUELL- UND ZIELADRESSE SIND UNGLEICH
                        ;ES KANN KOPIERT WERDEN!
1311                    ;LSB MATRIX NACH AKKUMULATOR
1312                    ;XOR MIT LSB POINTER AUF MATRIX
1313                    ;QUELL- UND ZIELADRESSE SIND GLEICH
1315                    ;BC-REGISTER ZUR VERFÜGUNG STELLEN
1316                    ;ACHT BYTES MATRIX MIT LDIR KOPIEREN
1319                    ;BC-REGISTER WIEDER ALTER WERT
131A                    ;NÄCHSTES ZEICHEN
131B                    ;SPRINGE SOLANGE ZEICHEN
131D                    ;INHALT VON DE VOM STACK HOLEN
131E                    ;TXT GET M TABLE
1321                    ;(TXT 1. ZEICHEN USER MATRIX)
1325                    ;ADRESSE DER BENUTZER-MATRIX
1326                    ;(TXT ADR. USER MATRIX
132A                    ;DAS WAR'S
```

132B ---------- TXT GET M TABLE

TXT GET M TABLE ermittelt die Startadresse und das erste Zeichen einer Anwendermatrix. Beim verlassen der Routine befindet sich im Akkumulator die Nummer des ersten Zeichens der Anwendermatrix und im Registerpaar HL die Adresse der Anwendermatrix. Des weiteren zeigt ein gesetztes CARRY-Flag das Vorhandensein einer Anwendermatrix an.

```
132B                    ;(TXT ERSTES ZEICHEN USER MATRIX)
132E                    ;MSB NACH AKKUMULATOR
132F                    ;CARRY BEEINFLUSSEN
1330                    ;LSB NACH AKKUMULATOR
1331                    ;(TXT ADR. USER MATRIX)
1334                    ;FERTIG
```

```
1335 ---------------------------------------------------- TXT WR CHAR
```

Zeichen darstellen. Im Akku wird übergeben, welches Zeichen dargestellt werden soll.

```
1335                    ;DARZUSTELLENDES ZEICHEN NACH B
1336                    ;(TXT LFD. CURSOR FLAG)
1339                    ;CARRY-FLAG BEEINFLUSSEN
133A                    ;ZURÜCK, WENN CARRY GESETZT IST
133B                    ;ZEICHEN IN B AUF STACK SICHERN
133C                    ;CURSOR INVERTIEREN
133F                    ;1 ZU SPALTE FÜR CURSOR ADDIEREN
1340                    ;(TXT LFD. CURSOR POS. (ROW, COL))
1343                    ;ALTER WERT FÜR SPALTE CURSOR
1344                    ;ZEICHEN VOM STACK NACH AKKUMULATOR
1345                    ;TXT WRITE CHAR
1348                    ;TXT DRAW/UNDRAW CURSOR
```

```
134B ---------------------------------------------------- TXT WRITE CHAR
```

Ein Zeichen auf den Bildschirm schreiben. Um diese Aufgabe erfüllen zu können, benötigt die Routine die Angaben, welches Zeichen und an welche Position es geschrieben werden soll. Im Akkumulator wird die Nummer des darzustellenden Zeichens übergeben. Im Registerpaar HL befindet sich die Position, an der das Zeichen ausgegeben werden soll (H=Spalte, L=Zeile).

```
134B                    ;ZEICHENPOSITION RETTEN
134C                    ;TXT GET MATRIX
134F                    ;ADRESSE DER MATRIX (UNGEPACKT)
1352                    ;DIESE AUF DEN STACK SICHERN
1353                    ;SCR UNPACK
1356                    ;ADRESSE DER MATRIX (UNGEPACKT)
1357                    ;ZEICHENPOSITION HOLEN
1358                    ;SCR CHAR POSITION
135B                    ;BELEGT ACHT RASTERZEILEN
135D                    ;INHALT VON BC (ZÄHLER) RETTEN
135E                    ;INHALT VON HL RETTEN
135F                    ;ZÄHLER EIN ZWEITES MAL RETTEN
1360                    ;ADRESSE DER MATRIX UNGEPACKT SICHERN
```



```
1361                    ;UND NACH HL TAUSCHEN
1362                    ;ADRESSIERTES BYTE AUS MATRIX HOLEN
1363                    ;VERZWEIGUNG
1366                    ;SCR NEXT BYTE
1369                    ;MATRIX BYTE HOLEN
136A                    ;NÄCHSTES BYTE DER MATRIX
136B                    ;ALTER INHALT VON BC (ZÄHLER)
136C                    ;SOLANGE NOCH BYTES VORHANDEN
136E                    ;ALTER INHALT VON HL
136F                    ;SCR NEXT LINE
1372                    ;ALTER INHALT VON BC (ZÄHLER)
1373                    ;TEST AUF NULL
1374                    ;WEITER WENN NOCH RASTERZEILEN
1376                    ;FERTIG
```

```
1377 ---------------------------------------------------- VERZWEIGUNG
```

```
1377                    ;(TXT LFD. BACKGROUND MODE)
137A                    ;IN DIE GEWÜNSCHTE ROUTINE VERZWEIGEN
```

```
137B ---------------------------------------------------- TXT SET BACK
```

TXT SET BACK ist für das Setzen des Transparentmodus zuständig. Wird dieser Routine im Akkumulator der Wert 0 übergeben, so wählt sie die herkömmliche Darstellungsweise des CPC (Transparentmodus aus). Wir hingegen im Akku eine 1 zur Verfügung gestellt, so schaltet die Routine den Transparentmodus ein.

```
137B                    ;ADRESSE TRANSPARENTMODUS AUS
137E                    ;MODUS FESTSTELLEN
137F                    ;NORMALEINSTELLUNG SETZEN
1381                    ;ADRESSE TRANSPARENTMODUS EIN
1384                    ;(TXT LFD. BACKGROUND MODE)
1387                    ;DAS WAR'S
```

1388 ---------- TXT GET BACK

TXT GET BACK ermittelt welcher Transparentmodus gesetzt ist. Ist beim Verlassen der Routine das ZERO-Flag nicht gesetzt, so ist der Transparentmodus ausgeschaltet; Ein gesetztes ZERO signalisiert eingeschalteten Transparentmodus.

```
1388                    ;(TXT LFD. BACKGROUND MODE)
138B                    ;HILFSSUMMAND NACH DE
138E                    ;HL KANN ENTHALTEN &1392 ODER &13A0
                        ;&1392+HILFSSUMMAND=&0000
                        ;&13A0+HILFSSUMMAND=&000E
138F                    ;MSB NACH AKKUMULATOR
1390                    ;FLAGBEEINFLUSSUNG:
                        ;ZERO=0 TRANSPARENTMODUS AUS
                        ;ZERO=1 TRANSPARENTMODUS EIN
1391                    ;DAS WAR'S
```

1392 ---------- TRANSPARENTMODUS AUS (NORMALEINSTELLUNG)

Im DE-Register wird die Bildschrimadresse an die Routine übergeben und in C steht das Byte der Zeichenmatrix.

```
1392                    ;L=TXT LFD. PEN  H=TXT LFD. PAPER
1395                    ;A=BYTE DER ZEICHENMATRIX
1396                    ;NEGAT DER ZEICHENMATRIX ERSTELLEN
1397                    ;MIT PAPER-FARBE VERKNÜPFEN
1398                    ;NEGAT-MASKE NACH B-REGISTER
1399                    ;BYTE DER ZEICHENMATRIX HOLEN
139A                    ;MIT PEN-FARBE EINFÄRBEN
139B                    ;MIT NEGAT-MASKE (HINTERGRUNDFARBE)
                        ;VERKNÜPFEN
139C                    ;ALLE BITS SETZEN UM AKKU IN SCR PIXELS
                        ;(FORCE MODE) NICHT ZU ZERSTÖREN
139E                    ;AB HIER GLEICHE VERARBEITUNG
```


13A0 ---------- TRANSPARENTMODUS EIN

In Registerpaar DE wird die Bildschirmadresse an TRANSPARENTMODUS EIN übergeben. Im C-Register steht das Byte der Zeichenmatrix.

```
13A0                    ;(TXT LFD. PEN)
13A3                    ;TXT LFD. PEN NACH B-REGISTER
13A4                    ;TAUSCHE BILDSCHIRMADRESSE NACH HL
13A5                    ;SCR PIXELS (FORCE MODE)
```

13A8 ---------- TXT SET GRAPHIC

Diese Routine bestimmt ob die Ausgabe von Zeichen an der Position des Text- oder Grafikcursors erfolgen soll. Wird im Akkumulator eine Null an TXT SET GRAPHIC übergeben, so erfolgt die Ausgabe von Zeichen (wie üblich) an der Position des Text-Cursors. Wird etwas anderes als Null übergeben, so wird die Ausgabe an der Position des Grafik-Cursors erfolgen (BASIC-Befehl TAG).

```
13A8                    ;(TXT GRAPH CHAR WRITE MODE (0=DIS.))
13AB                    ;FERTIG
```

13AC ---------- TXT RD CHAR

TXT RD CHAR ließt ein Zeichen vom Bildschirm. Beim Verlassen befindet sich das Zeichen im Akkumulator. Wurde ein Blank gefunden, so ist das ZERO-Flag gesetzt. Ein gelöschtes CARRY-Flag zeigt an, daß kein Zeichen identifiziert werden konnte.

```
13AC                    ;HL-REGISTERPAAR BEREITSTELLEN
13AD                    ;DE-REGISTERPAAR BEREITSTELLEN
13AE                    ;BC-REGISTERPAAR BEREITSTELLEN
13AF                    ;CURSOR INVERTIEREN
13B2                    ;TXT UNWRITE CHAR
13B5                    ;ZEICHEN UND FLAGS AUF STACK
13B6                    ;TXT DRAW/UNDRAW CURSOR
13B9                    ;ZEICHEN UND FLAGS VOM STACK
13BA                    ;ALTEN ZUSTAND VON BC
```

```
13BB                    ;ALTEN ZUSTAND VON DE
13BC                    ;ALTEN ZUSTAND VON HL
13BD                    ;DAS WAR'S
```

13BE ---------- TXT UNWRITE CHAR

Ein Zeichen vom Bildschirm lesen. Im Registerpaar HL wird die Position des Zeichen an TXT UNWRITE CHAR übergeben (H=Spalte, L=Zeile). Als Ergebnis liefert die Routine das gefundene Zeichen im Akkumulator. Ist das CARRY-Flag nicht gesetzt, so konnte das Zeichen nicht bestimmt werden. Ein gesetztes ZERO-Flag steht für ein gefundenes Leerzeichen.

```
13BE                    ;(TXT LFD. PAPER)
13C1                    ;ADRESSE DER MATRIX
13C4                    ;ZEICHEN-POSITION SICHERN
13C5                    ;ADRESSE DER MATRIX SICHERN
13C6                    ;SCR REPACK
13C9                    ;MATRIX-ADRESSE VOM STACK HOLEN
13CA                    ;UND SOFORT WIEDER SICHERN
13CB                    ;ANZAHL DER RASTERZEILEN NACH B
13CD                    ;MATRIX-BYTE HOLEN
13CE                    ;KOMPLEMENT BILDEN
13CF                    ;KOMPLEMENT ZURÜCKSCHREIBEN
13D0                    ;AUF NÄCHSTES BYTE ZEIGEN
13D1                    ;SPRINGE SOLANGE NOCH RASTERZEILEN
13D3                    ;VERGLEICHE MATRIX
13D6                    ;MATRIX-ADRESSE VOM STACK HOLEN
13D7                    ;ZEICHEN-POSITION VOM STACK HOLEN
13D8                    ;SPRINGE, WENN KEIN ZEICHEN GEFUNDEN
13DA                    ;ZURÜCK, WENN ETWAS ANDERES ALS
                        ;LEERZEICHEN GEFUNDEN
13DB                    ;(TXT LFD. PEN)
13DE                    ;SCR REPACK
```


13E1 ---------- VERGLEICHE MATRIX

Als Ergebnis liefert die Routine das Zeichen im Akkumulator. Ist das CARRY-Flag nicht gesetzt, so konnte das Zeichen nicht identiviziert werden. Ein gesetztes ZERO-Flag steht für ein Leerzeichen.

```
13E1                    ;ZÄHLER INITIALISIEREN
13E3                    ;ZÄHLERSTAND NACH AKKUMULATOR
13E4                    ;TXT GET MATRIX
13E7                    ;ADRESSE DER MATRIX
13EA                    ;ANZAHL DER RASTERZEILEN NACH B
13EC                    ;MATRIX-BYTE HOLEN
13ED                    ;VERGLEICHE MIT VORGEGEBENEM BYTE
13EE                    ;SPRINGE WENN UNGLEICH
13F0                    ;AUF NÄCHSTES BYTE
13F1                    ;AUF NÄCHSTES ZU VERGLEICHENDES BYTE
13F2                    ;SPRINGE SOLANGE NOCH RASTERZEILEN
13F4                    ;NUMMER DES ZEICHENS NACH AKKU
13F5                    ;LEERZEICHEN GEFUNDEN (ZERO=1)
13F7                    ;CARRY=1 FÜR SUCHE ERFOLGREICH
13F8                    ;FERTIG
13F9                    ;NÄCHSTES ZEICHEN
13FA                    ;SPRINGE SOLANGE ZEICHEN
13FC                    ;CARRY=0 FÜR SUCHE ERFOLGLOS
13FD                    ;FERTIG
```

13FE ---------- TXT OUTPUT

(Steuer-) Zeichen darstellen oder ausführen.

Bringt das Zeichen im Akku auf das aktuelle Bildschirmfenster, bzw. führt es aus, falls es sich um ein Steuerzeichen handelt.

Beachten Sie, daß diese Routine die Indirection TXT OUT ACTION benutzt. Sollten Sie diese manipuliert haben, wird TXT OUTPUT auch Ihre Routine, statt der im ROM stehenden, benutzen.

```
13FE                    ;REGISTERPAAR AF SICHERN
13FF                    ;REGISTERPAAR BC SICHERN
```

```
1400                    ;REGISTERPAAR DE SICHERN
1401                    ;REGISTERPAAR HL SICHERN
1402                    ;TXT OUT ACTION
1405                    ;ALTEN INHALT VON HL HOLEN
1406                    ;ALTEN INHALT VON DE HOLEN
1407                    ;ALTEN INHALT VON BC HOLEN
1408                    ;ALTEN INHALT VON AF HOLEN
1409                    ;ZURÜCK
```

140A ---------- TXT OUT ACTION

Ausgabe eines Zeichens auf dem Bildschirm oder Ausführung eines Steuercodes. Im Akkumulator wird die Nummer des Zeichens an die Routine übergeben.

```
140A                    ;NUMMER DES ZEICHENS NACH C RETTEN
140B                    ;(TXT GRAPH CHAR WRITE MODE)
140E                    ;ZERO-FLAG BEEINFLUSSEN
140F                    ;NUMMER DES ZEICHENS IN AKKU
1410                    ;GR WRITE CHAR
1413                    ;(TXT ZEICHENZÄHLER CONTROL BUFFER)
1416                    ;ZÄHLERINHALT NACH B-REGISTER
1417                    ;ZÄHLERINHALT NACH AKKUMULATOR
1418                    ;VERGLEICHE MIT MAX. LÄNGE
141A                    ;CONTROL BUFFER LÖSCHEN
141C                    ;GIBT ES ZEICHEN IM CONTROL BUFFER?
141D                    ;SPRINGE UNTER DIESER PRÄMISSE
141F                    ;NUMMER DES ZEICHENS NACH AKKU
1420                    ;HANDELT ES SICH UM STEUERZEICHEN?
1422                    ;KEIN STEUERZEICHEN, DANN TXT WR CHAR
1425                    ;ZÄHLER=ZÄHLER+1
1426                    ;(TXT ZEICHENZÄHLER CONTROL BUFFER)
1427                    ;ZÄHLERSTAND NACH E BRINGEN
1428                    ;ZU DOPPELREGISTER AUFSTOCKEN
142A                    ;BASISADRESSE+ZÄHLERSTAND
142B                    ;ZEICHEN AN DIESER STELLE SICHERN
142C                    ;(POINTER CONTROL BUFFER)
142F                    ;ENTSPRECHENDES STEUERZEICHEN NACH E
1430                    ;ADRESSE SPRUNGTABELLE STEUERZEICHEN
```



```
1433                    ;ADRESSE=ADRESSE+ZÄHLERSTAND
1434                    ;+ZÄHLERSTAND
1435                    ;+ZÄHLERSTAND
1436                    ;BYTE AN ERMITTELTER STELLE NACH AKKU
1437                    ;BITS 4 BIS 7 AUSBLENDEN
1439                    ;VERGLEICHE MIT ZÄHLERSTAND
143A                    ;MEHR ZEICHEN!
143B                    ;(TXT CURSOR FLAG)
143E                    ;AND MIT BYTE AN ERMITTELTER STELLE
143F                    ;BIT 7 NACH CARRY-FLAG
1440                    ;CONTROL BUFFER LÖSCHEN
1442                    ;AUF LSB FÜR CALL-ADRESSE
1443                    ;LSB FÜR CALL-ADRESSE
1444                    ;AUF MSB FÜR CALL-ADRESSE
1445                    ;MSB FÜR CALL-ADRESSE
1446                    ;POINTER CONTROL BUFFER
1449                    ;ZEICHEN NACH AKKUMULATOR
144A                    ;BEDEUTET SOVIEL WIE: CALL (DE)
144D                    ;AKKU LÖSCHEN UND FLAG ZURÜCKSETZEN
144E                    ;(TXT ZEICHENZÄHLER CONTROL BUFFER)
1451                    ;FERTIG
```

1452 ---------- TXT VDU DISABLE

Zeichendarstellung unterbinden.

```
1452                    ;BIT 7 UND BIT 0 WIRD GESETZT
1454                    ;CUR DISABLE CONT'D
1457                    ;WEITER BITTE
```

1459 ---------- TXT VDU ENABLE

Es können Zeichen auf den Bildschirm geschrieben werden.

```
1459                    ;BIT 1 BIS BIT 6 WIRD GESETZT
145B                    ;CUR ENABLE CONT'D
145E                    ;FORTSETZUNG FOLGT
```

```
1460 ------------------------------------ LFD. CURSOR FLAG NACH AKKU

1460                        ;(TXT LFD. CURSOR FLAG)
1463                        ;SCHON FERTIG


1464 --------------------------- DEFAULT STEUERZEICHEN SPRÜNGE KOPIEREN

1464                        ;AKKU LÖSCHEN UND FLAGS ZURÜCKSETZEN
1465                        ;(ZEICHENZÄHLER CONTROL BUFFER)
1468                        ;QUELLE: DEFAULT STEUERZEICHEN SPRÜNGE
146B                        ;ZIEL: SPRUNGTABELLE STEUERZEICHEN
146E                        ;ZÄHLER: ANZAHL ZU KOPIERENDE BYTES
1471                        ;KOPIERVORGANG AUSFÜHREN
1473                        ;FERTIG


1474 ---------------------------------- DEFAULT STEUERZEICHEN SPRÜNGE

1474                        &80
1475                        ;00 KEIN EINFLUSS (CHR$(0))

1477                        &81
1478                        ;01 TXT WR CHAR (CHR$(1))

147A                        &80
147B                        ;02 TXT CUR DISABLE (CHR$(2))

147D                        &80
147E                        ;03 TXT CUR ENABLE (CHR$(3))

1480                        &81
1481                        ;04 SCR SET MODE (CHR$(4))

1483                        &81
1484                        ;05 GRA WR CHAR (CHR$(5))

1486                        &00
1487                        ;06 TXT VDU ENABLE (CHR$(6))
```



```
1489                        &80
148A                        ;07 KLINGEL (CHR$(7))

148C                        &80
148D                        ;08 CRSR LEFT (CHR$(8))

148F                        &80
1490                        ;09 CRSR RIGHT (CHR$(9))

1492                        &80
1493                        ;0A CRSR DOWN (CHR$(10))

1495                        &80
1496                        ;0B CRSR UP (CHR$(11))

1498                        &80
1499                        ;0C TXT CLEAR WINDOW (CHR$(12))

149B                        &80
149C                        ;0D CRSR AUF ZEILENANFANG (CHR$(13))

149E                        &81
149F                        ;0E TXT SET PAPER (CHR$(14))

14A1                        &81
14A2                        ;0F TXT SET PEN (CHR$(15))

14A4                        &80
14A5                        ;10 ZEICHEN AUF CURSORPOSITION LÖSCHEN
                            ;(CHR$(16))

14A7                        &80
14A8                        ;11 ZEILE BIS CURSORPOSITION LÖSCHEN
                            ;(CHR$(17))

14AA                        &80
14AB                        ;12 ZEILE AB CURSORPOSITION LÖSCHEN
                            ;(CHR$(18))
```

```
14AD                    &80
14AE                    ;13 FENSTER BIS CURSORPOSITION LÖSCHEN
                        ;(CHR$(19))

14B0                    &80
14B1                    ;14 FENSTER AB CURSORPOSITION LÖSCHEN
                        ;2(CHR$(20))

14B3                    &80
14B4                    ;15 TXT VDU DISABLE (CHR$(21))

14B6                    &81
14B7                    ;16 TRANSPARENTMODUS EIN/AUS (CHR$(22))

14B9                    &81
14BA                    ;17 SCR ACCESS (CHR$(23))

14BC                    &80
14BD                    ;18 TXT INVERSE (CHR$(24))

14BF                    &89
14C0                    ;19 SYMBOL-BEFEHL (CHR$(25))

14C2                    &84
14C3                    ;1A FENSTER DEFINIEREN (CHR$(26))

14C5                    &00
14C6                    ;1B KEIN EFFEKT (CHR$(27))

14C8                    &83
14C9                    ;1C INK-BEFEHL (CHR$(28))

14CB                    &82
14CC                    ;1D BORDER-BEFEHL (CHR$(29))

14CE                    &80
14CF                    ;1E CRSR HOME (CHR$(30))

14D1                    &82
14D2                    ;1F LOCATE-BEFEHL (CHR$(31))
```


14D4 ---------- TXT GET CONTROLS

TXT GET CONTROLS holt die Adresse der Steuerzeichen-Sprungtabelle. Diese Adresse befindet sich beim Verlassen der Routine im Registerpaar HL.

```
14D4                    ;TXT SPRUNGTABELLE STEUERZEICHEN
14D7                    ;ZURÜCK
```

14D8 ---------- PARAMETERTABELLE FÜR KLINGEL (CHR$(7))

```
14D8                    87 00 00 5A 00 00 0B 14
14E0                    00
```

14E1 ---------- KLINGEL (CHR$(7))

Die Routine KLINGEL zeichnet für die Erzeugung des Tones CHR$(7) verantwortlich.

```
14E1                    ;IX-REGISTER SICHERN
14E3                    ;POINTER AUF PARAMETERTABELLE
14E6                    ;SOUND QUEUE
14E9                    ;ALTEN ZUSTAND VON IX HOLEN
14EB                    ;DAS WAR'S
```

14EC ---------- TRANSPARENTMODE EIN/AUS (CHR$(22))

```
14EC                    ;PARAMETER NACH CARRY ROTIEREN
14ED                    ;AKKU WIRD &00 ODER &FF
14EE                    ;TXT SET BACK
```

14F1 ---------- INK-BEFEHL (CHR$(28))

```
14F1                    ;ZEIGER AUF ERSTEN PARAMETER
14F2                    ;Nummer des Farbstifts
```

```
14F3                    ;ZEIGER AUF ZWEITEN PARAMETER
14F4                    ;Parameter erster Farbwert
14F5                    ;ZEIGER AUF DRITTEN PARAMETER
14F6                    ;Parameter zweiter Farbwert
14F7                    ;SCR SET INK


14FA ---------------------------------------- BORDER-BEFEHL (CHR$(29))

14FA                    ;ZEIGER AUF ERSTEN PARAMETER
14FB                    ;PARAMETER ERSTER FARBWERT
14FC                    ;ZEIGER AUF ZWEITEN PARAMETER
14FD                    ;PARAMETER ZWEITER FARBWERT
14FE                    ;SCR SET BORDER


1501 ---------------------------------- FENSTER DEFINIEREN (CHR$(26))

1501                    ;ZEIGER AUF ERSTEN PARAMETER
1502                    ;FENSTERGRENZE RECHTS/LINKS
1503                    ;ZEIGER AUF ZWEITEN PARAMETER
1504                    ;FENSTERGRENZE RECHTS/LINKS
1505                    ;ZEIGER AUF DRITTEN PARAMETER
1506                    ;FENSTERGRENZE OBEN/UNTEN
1507                    ;ZEIGER AUF VIERTEN PARAMETER
1508                    ;FENSTERGRENZE OBEN/UNTEN
1509                    ;FENSTERGRENZE RECHTS/LINKS
150A                    ;TXT WIN ENABLE


150D ---------------------------------------- SYMBOL-BEFEHL (CHR$(25))

150D                    ;HL ZEIGT AUF NUMMER DES ZEICHENS
150E                    ;NUMMER DES ZEICHENS NACH AKKU
150F                    ;HL ZEIGT AUF MATRIX-ADRESSE
1510                    ;TXT SET MATRIX
```



```
1513 ---------------------------------------------------- CHR$(0)

1513                    ;CURSOR INVERTIEREN
1516                    ;TXT DRAW/UNDRAW CURSOR


1519 ------------------------------------------ CRSR LEFT (CHR$(8))

1519                    ;KENNUNG FÜR SPALTE ERNIEDRIGEN
151C                    ;ES GEHT WEITER


151E ------------------------------------------ CRSR RIGHT (CHR$(9))

151E                    ;KENNUNG FÜR SPALTE ERHÖHEN
1521                    ;HIER GEHT'S WEITER


1523 ------------------------------------------ CRSR DOWN (CHR$(10))

1523                    ;KENNUNG FÜR ZEILE ERHÖHEN
1526                    ;HIER GEHT'S WEITER


1528 ------------------------------------------ CRSR UP (CHR$(11))

1528                    ;KENNUNG FÜR ZEILE ERNIEDRIGEN
152B                    ;KENNUNG AUF DEN STACK SICHERN
152C                    ;CURSOR INVERTIEREN
152F                    ;KENNUNG VOM STACK HOLEN
1530                    ;CURSORPOSITION ZEILE
1531                    ;KENNUNG ZEILE ADDIEREN
1532                    ;NEUE CURSORPOSITION (ROW)
1533                    ;CURSORPOSITION SPALTE
1534                    ;KENNUNG SPALTE ADDIEREN
1535                    ;NEUE CURSORPOSITION (COL)
1536                    ;NEUE CURSORPOSITION WEGSCHREIBEN, DANN
                        ;TXT DRAW/UNDRAW CURSOR
```

```
1539 ---------------------------------------- CRSR HOME (CHR$(30))

1539                    ;(TXT LFD. FENSTER OBEN)
153C                    ;AUF ZUR CURSORPOSITIONIERUNG


153F ---------------------------------- CRSR AUF ZEILENANFANG (CHR$(13))

153F                    ;CURSOR INVERTIEREN
1542                    ;(TXT LFD. FENSTER LINKS)
1545                    ;DORT GEHT ES WEITER


1547 ---------------------------------------- LOCATE-BEFEHL (CHR$(31))

1547                    ;POINTER AUF CURSORSPALTE
1548                    ;CURSORPOSITION SPALTE
1549                    ;POINTER AUF CURSORZEILE
154A                    ;CURSORPOSITION ZEILE
1548                    ;IN DE ZUSAMMENGESTELLTE POSITION NACH
                        ;HL TAUSCHEN
154C                    ;TXT SET CURSOR


154F ---------------------------------- TXT CLEAR WINDOW (CHR$(12))

154F                    ;TXT DRAW/UNDRAW CURSOR
1552                    ;H=TXT LFD. FENSTER LINKS
                        ;L=TXT LFD. FENSTER OBEN
1555                    ;(TXT LFD. CURSORPOS. (ROW, COL))
1558                    ;D=TXT LFD. FENSTER RECHTS
                        ;E=TXT LFD. FENSTER UNTEN
155C                    ;ZUR DURCHFÜHRUNG


155E ------------------------------ ZEICHEN AUF C-POS LÖSCHEN (CHR$(16))

155E                    ;CURSOR INVERTIEREN
1561                    ;CURSORPOSITION SPALTE NACH D
```



```
1562                    ;CURSORPOSITION ZEILE NACH E
1563                    ;ZUR DURCHFÜHRUNG


1565 ------------------------------ FENSTER AB C-POS LÖSCHEN (CHR$(20))

1565                    ;ZEILE AB CURSORPOSITION LÖSCHEN
1568                    ;H=TXT LFD. FENSTER LINKS
                        ;L=TXT LFD. FENSTER OBEN
156B                    ;D=TXT LFD. FENSTER RECHTS
                        ;E=TXT LFD. FENSTER UNTEN
156F                    ;LFD. CURSORPOSITION NACH AKKU
1572                    ;CURSORZEILE NACH L BRINGEN
1573                    ;WIRD STARTZEILE
1574                    ;STARTZEILE KLEINER FENSTER UNTEN
1575                    ;ZURÜCK UNTER DIESER PRÄMISSE
1576                    ;HIER GEHT'S WEITER


1578 ------------------------------ FENSTER BIS C-POS LÖSCHEN (CHR$(19))

1578                    ;ZEILE BIS CURSORPOSITION LÖSCHEN
157B                    ;(TXT LFD. FENSTER OBEN)
157E                    ;(TXT LFD. FENSTER RECHTS)
1581                    ;FENSTERGRENZE RECHTS NACH AKKU
1582                    ;(TXT LFD. CURSOR POS. (ROW,COL))
1585                    ;CURSORPOSITION (ROW) -1
1586                    ;WIRD ENDPOSITION
1587                    ;START KLEINER ODER GLEICH ENDPOSITION?
1588                    ;UNTER DIESER PRÄMISSE ZURÜCK
1589                    ;(TXT LFD. PAPER)
158C                    ;SCR FILL BOX


158F ------------------------------ ZEILE AB C-POS LÖSCHEN (CHR$(18))
2
158F                    ;CURSOR INVERTIEREN
1592                    ;CURSORPOSITION (ROW) ALS ZEILE
1593                    ;(TXT LFD. FENSTER RECHTS)
```

```
1596                    ;RECHTE FENSTERGRENZE NACH D
1597                    ;ZUR WEITERFÜHRUNG


1599 ------------------------------ ZEILE BIS C-POS LÖSCHEN (CHR$(17))

1599                    ;CURSOR INVERTIEREN
159C                    ;CURSORPOSITION NACH DE BRINGEN
159D                    ;CURSORPOSITION (ROW) ALS ZEILE
159E                    ;(TXT LFD. FENSTER LINKS)
15A1                    ;LINKE FENSTERGRENZE
15A2                    ;DORT GEHT ES WEITER
15A5                    ;TXT DRAW/UNDRAW CURSOR
```


9.6.4 Graphics Screen (GRA)

Dieses Pack dient ausschließlich der Handhabung des Graphikfensters.

Zu den Koordinatenangaben, die von den verschiedenen Routinen verlangt werden, ist folgendes zu bemerken:

Die Koordinaten werden in drei Stufen übersetzt. Die anwendernächste Stufe ist die Position bezüglich des von ihm gesetzten Koordinatenursprungs (ORIGIN). Diese wird umgerechnet in eine Position relativ zum Bildschirmursprung (unten links). Diese beiden Stufen sind vom Mode unabhängig!
Die letzte Stufe ist die physikalische Adresse des Punktes. Diese ist abhängig vom aktuellen Modus!

Diesen drei Stufen wird dann noch eine vierte Stufe vorangestellt, wenn ein relatives Koordinatenpaar in eine absolute Position, relativ zu ORIGIN, umgerechnet werden muß.

```
15A8 ------------------------------------------------ GRA INITIALISE

Vollständige Initialisierung des Grafikpacks.

15A8                    ;GRA RESET
15AB                    ;PEN 1, PAPER 0
15AE                    ;MSB NACH AKKU
15AF                    ;GRA SET PAPER
15B2                    ;LSB NACH AKKU
15B3                    ;GRA SET PEN
15B6                    ;ORIGIN AUF 0,0 SETZEN
15B9                    ;MSB NACH D-REGISTER
15BA                    ;LSB NACH E-REGISTER
15BB                    ;GRA SET ORIGIN
15BE                    ;KLEINSTEN WERT NACH DE
15C1                    ;GRÖSSTEN WERT NACH HL
15C4                    ;GRÖSSTEN WERT AUF STACK
15C5                    ;KLEINSTEN WERT AUF STACK
15C6                    ;GRA WIN WIDTH
```

```
15C9                    ;ALTE INHALTE VOM
15CA                    ;STACK HOLEN
15CB                    ;GRA WIN HEIGHT
```

15DE ---------- DECODE PEN AND PAPER

Als Ausgabe liefert diese Routine den Wert für den Hintergrund im H-Register und den Wert für den Vordergrund im L-Register.

```
15CE                    ;GRA GET PAPER
15D1                    ;FARBSTIFT FÜR HINTERGRUND NACH H
15D2                    ;GRA GET PEN
15D5                    ;FARBSTIFT FÜR VORDERGRUND NACH L
15D6                    ;FERTIG!
```

15D7 ---------- GRA RESET

Zurücksetzen des Grafikpacks und kopieren der Indirections.

```
15D7                    ;INITIALISE
15DA                    ;RESTORE GRA INDIRECTIONS
15DD                    ;MOVE (HL+3) NACH ((HL+1)), CNT=(HL)
15E0                    ;9 BYTES
15E1                    ;ZIELADRESSE

15E3                    ;CODE FÜR JP
15E4                    ;GRA PLOT

15E6                    ;CODE FÜR JP
15E7                    ;GRA TEST

15E9                    ;CODE FÜR JP
15EA                    ;GRA LINE
```

15EC ---------- GRA EXTENDED INITIALISE

```
15EC                    ;FORCE MODE EINSCHALTEN
```



```
15ED                    ;SCR ACCESS
15F0                    ;FORCE MODE EINSCHALTEN
15F1                    ;GRA FILL
15F4                    ;PARAMETER FÜR ERSTEN PUNKT GENERIEREN
15F5                    ;GRA FIRST POINT
15F8                    ;GRA SET MASK
```

15FB ---------- GRA MOVE RELATIVE

Bewegung zu einer relativen Position.

```
15FB                    ;ADD LFD KOORD. + REL KOORD.
```

15FE ---------- GRA MOVE ABSOLUTE

Bewegung zu einer absoluten Position.

```
15FE                    ;(LFD X-KOORDINATE)
1602                    ;(LFD Y-KOORDINATE)
1605                    ;ERLEDIGT!
```

1606 ---------- GRA ASK CURSOR

Wo ist der lfd. Grafikcursor? Als Ausgabeparameter liefert GRA ASK CURSOR im HL-Register die Y-Koordinate und im DE-Register die X-Koordinate des Grafikcursors.

```
1606                    ;(LFD X-KOORDINATE)
160A                    ;(LFD Y-KOORDINATE)
160D                    ;DAS WAR'S
```

160E ---------- GRA SET ORIGIN

Ursprung der Anwenderkoordinaten setzen. Als Eingabeparameter erwartet die Routine im HL-Register die Y-Koordinate und im DE-Register die X-Koodinate.

```
160E                    ;(X ORIGIN)
1612                    ;(Y ORIGIN)
1615                    ;DE-REGISTER INITIALISIEREN
1618                    ;HL-REGISTER
1619                    ;INITIALISIEREN
161A                    ;GRA MOVE ABSOLUTE


161C ------------------------------------------------- GRA GET ORIGIN
```

Ursprung der Anwenderkoordinaten holen. Als Ausgabeparameter liefert die Routine im HL-Register die Y-Koordinate und im DE-Register die X-Koodinate.

```
161C                    ;(X ORIGIN)
1620                    ;(Y ORIGIN)
1623                    ;DAS WAR'S SCHON!


1624 ---------------------------------------- PHYS STARTPOSITION HOLEN

1624                    ;GRA ASK CURSOR


1627 ---------------------------- PHYS ZIELPOS. HOLEN UND CURSOR SETZEN

1627                    ;GRA MOVE ABSOLUTE


162A ------------------------------------------------- GRA CONVERT POS

162A                    ;Y-KOORDINATE AUF STACK RETTEN
162B                    ;SCR GET MODE
162E                    ;MODE DEFINIEREN
1630                    ;MASKE GENERIEREN
1632                    ;H-REGISTER INITIALISIEREN
1634                    ;MASKE NACH L-REGISTER
1635                    ;BIT SIEBEN GESETZT?
1637                    ;SPRINGE UNTER DIESER PRÄMISSE
```



```
1639                    ;ADDITION VORBEREITEN
163A                    ;MASKE ADDIEREN
163B                    ;ERGEBNIS NACH DE
163C                    ;MASKE INVERTIEREN
163D                    ;UND MIT DEM E-REGISTER VERBINDEN
163E                    ;ERGEBNIS ZURÜCK INS E-REGISTER
                        ;SCHREIBEN
163F                    ;RETTE MASKE
1640                    ;(X ORIGIN)
1643                    ;MIT ERRECHNETER
1644                    ;MASKE HALBIEREN
1645                    ;HL DURCH ZWEI TEILEN
1648                    ;MASKE HALBIEREN
1649                    ;HL DURCH ZWEI TEILEN
164C                    ;HOLE Y-KOORDINATE
164D                    ;UND LEGE X-KOORDINATE AUF STACK
164E                    ;HOLE MSB VON Y-KOORDINATE
164F                    ;MSB*2
1650                    ;GRENZE ÜBERSCHRITTEN
1652                    ;Y-KOORDINATE +1
1653                    ;IN RICHTIGES FORMAT BRINGEN
1655                    ;(Y ORIGIN)
1658                    ;Y-ORIGIN ADDIEREN
1659                    ;X-KOORDINATE HOLEN
165A                    ;Y DURCH ZWEI TEILEN


165D ------------------------------------ ADD LFD KOORD. + REL KOORD.

165D                    ;INHALT DES HL-REGISTERS RETTEN
165E                    ;(LFD X-KOORDINATE)
1661                    ;KOORDINATEN ADDIEREN
1662                    ;DE-REGISTER VOM STACK LADEN
1663                    ;ERGEBNIS DER ADDITION AUF DEN STACK
1664                    ;(LFD Y-KOORDINATE)
1667                    ;KOORDINATEN ADDIEREN
1668                    ;DE-REGISTER VOM STACK LADEN
1669                    ;FERTIG!
```

166A ---------- XPOS IM FENSTER?

Als Eingabeparameter erwartet die Routine im DE-Register eine X-Koordinate. Ein bei der Ausgabe gesetztes CARRY-Flag zeigt an, daß die Koordinate sich innerhalb der Fenstergrenzen befindet. Ist das CARRY nicht gesetzt, so zeigt ein gesetztes ZERO und eine 0 im Akkumulator an, das die X-Koordinate kleiner als die Fenstergrenze ist. Ist das ZERO nicht gesetzt und enthält der Akku &FF, so ist die Koordinate größer als die Fenstergrenze.

```
166A                    ;(X-KOORDINATE GRA FENSTER LINKS)
166D                    ;CARRY FÜR SBC SETZEN
166E                    ;LINKE FENSTERGRENZE - X-KOORD. (CARRY)
1670                    ;UNTER DIESER VORAUSSETZUNG SPRINGE
1673                    ;(X-KOORDINATE GRA FENSTER RECHTS)
1676                    ;CARRY FÜR SBC ZURÜCKSETZEN
1677                    ;RECHTE FENSTERGRENZE - X-KOORD.
1679                    ;KOORDINATE IM FENSTER
167A                    ;BEDINGTER RÜCKSPRUNG
167B                    ;KOORDINATE GRÖSSER ALS FENSTERGRENZE
167D                    ;RÜCKSPRUNG
167E                    ;KOORDINATE KLEINER ALS FENSTERGRENZE
167F                    ;RÜCKSPRUNG
```

1680 ---------- YPOS IM FENSTER?

Als Eingabeparameter erwartet die Routine im HL-Register eine Y-Koordinate. Ein bei der Ausgabe gesetztes CARRY-Flag zeigt an, daß die Koordinate sich innerhalb der Fenstergrenzen befindet. Ist das CARRY nicht gesetzt, so zeigt ein gesetztes ZERO und eine 0 im Akkumulator an, das die X-Koordinate kleiner als die Fenstergrenze ist. Ist das ZERO nicht gesetzt und enthält der Akku &FF, so ist die Koordinate größer als die Fenstergrenze.

```
1680                    ;(Y-KOORDINATE GRA FENSTER OBEN)
1683                    ;CARRY FÜR SBC ZURÜCKSETZEN
1684                    ;FENSTERGRENZE - Y-KOORDINATE
1686                    ;SPRINGE, WENN Y-KOORD. GRÖSSER
1689                    ;(Y-KOORDINATE GRA FENSTER UNTEN)
```



```
168C                    ;CARRY FÜR SBC SETZEN
168D                    ;UNTERE FENSTERGRENZE - Y-KOORD. (CARRY)
168F                    ;UNTER DIESER BEDINGUNG SPRINGE
1692                    ;KOORDINATE IM FENSTER
1693                    ;ERLEDIGT!
```

1694 ---------- POS IN WINDOW

Als Eingabeparameter erwartet POS IN WINDOW eine Y-Koordinate im HL-Register und eine X-Koordinate im DE-Register. Ein bei der Ausgabe gesetztes CARRY zeigt an, daß sich die Koordinaten innerhalb der Fenstergrenzen befinden.

```
1694                    ;PHYS ZIELPOS HOLEN UND CURSOR SETZEN
1697                    ;Y-KOORDINATE AUF STACK
1698                    ;XPOS IM FENSTER?
169B                    ;Y-KOORDINATE VOM STACK
169C                    ;POSITION AUSSERHALB DES FENSTERS
169D                    ;X-KOORDINATE AUF STACK
169E                    ;REGISTER FÜR YPOS IM FENSTER? TAUSCHEN
169F                    ;YPOS IM FENSTER?
16A2                    ;REGISTER ZURÜCKTAUSCHEN
16A3                    ;X-KOORDIANTE VOM STACK
16A4                    ;DAS WAR'S
```

16A5 ---------- GRA WIN WIDTH

Linke und rechte Begrenzung des Grafikfensters setzen. Dazu muß sich im HL-Register die rechte, und im DE-Register die linke Fenstergrenze befinden.

```
16A5                    ;RECHTE FENSTERGRENZE RETTEN
16A6                    ;ALTE LINKE FENSTERGRENZE
16A9                    ;RECHTE FENSTERGRENZE HOLEN
16AA                    ;LINKE FENSTERGRENZE RETTEN
16AB                    ;ALTE RECHTE FENSTERGRENZE
16AE                    ;LINKE FENSTERGRENZE HOLEN
16AF                    ;INKE FENSTERGRENZE<RECHTE GRENZE
```

```
16B0                    ;VERGL.16AF
16B1                    ;VERGL.16AF
16B2                    ;VERGL.16AF
16B3                    ;SPRINGE WENN KLEINER
16B5                    ;VERTAUSCHE GRENZEN
16B6                    ;LINKE GRENZE HOLEN
16B7                    ;UND MASKIEREN
16B9                    ;ERGEBNIS NACH A
16BA                    ;RECHTE GRENZE HOLEN
16BB                    ;UND MASKIEREN
16BD                    ;ERGEBNIS NACH A
16BE                    ;SCR GET MODE
16C1                    ;MODE 0?
16C2                    ;SPLIT VALUE
16C5                    ;MODE 1?
16C6                    ;SPLIT VALUE
16C9                    ;(X-KOORDINATE GRA FENSTER LINKS)
16CD                    ;(X-KOORDINATE GRA FENSTER RECHTS)
16D0                    ;DAS WAR'S

16D1                    ;MSB LINKE FENSTERGRENZE NACH AKKU
16D2                    ;C-FLAG ELEMINIEREN
16D3                    ;HL INIT
16D6                    ;NEGATIV?
16D7                    ;GRÖSSTER WERT
16DA                    ;LSB LINKE FENSTERGRENZE NACH AKKU
16DB                    ;REALE GRÖSSE ERMITTELN
16DC                    ;HOLE MSB LINKE FENSTERGRENZE NACH AKKU
16DD                    :REALE GRÖSSE ERMITTELN
16DE                    ;FERTIG, WENN KEIN ÜBERLAUF
16DF                    ;WIEDER ALTER WERT SETZEN
16E0                    ;FERTIG!
```

16E1 ---------- SPLIT VALUE

Die bei der Eingabe in den Registern HL und DE stehenden Werte stehen bei der Ausgabe halbiert in diesen Registern.


```
16E1                    ;INHALT VON DE-REGISTER
16E3                    ;DURCH ZWEI TEILEN
16E5                    ;INHALT VON HL-REGISTER
16E7                    ;DURCH ZWEI TEILEN
16E9                    ;FERTIG!
```

16EA ---------- GRA WIN HEIGHT

Obere und untere Begrenzung des Grafikfensters setzen.

```
16EA                    ;WERTE FÜR UNTERE FENSTERGRENZE RETTEN
16EB                    ;FENSTERGRENZE OK?
16EE                    ;HOLE UNTERE FENSTERGRENZE
16EF                    ;HL FREISTELLEN
16F0                    ;FENSTERGRENZE OK?
16F3                    ;UNTERE GRENZE HOLEN
16F4                    ;UNTERE GRENZE = OBERE GRENZE
16F5                    ;VERGL.16F4
16F6                    ;VERGL.16F4
16F7                    ;VERGL.16F4
16FA                    ;WERTE VERTAUSCHEN
16FB                    ;(Y-KOORDINATE GRA FENSTER OBEN)
16FF                    ;(Y-KOORDINATE GRA FENSTER UNTEN)
1702                    ;DAS WAR'S
```

1703 ---------- FENSTERGRENZE OK?

Als Ein-/Ausgabeparameter hat FENSTERGRENZE OK? eine Fenstergrenze in Y-Ausdehnung.

```
1703                    ;MSB OBERE FENSTERGRENZE NACH AKKU
1704                    ;CARRY ZURÜCKSETZEN
1705                    ;KLEINSTE FENSTERGRENZE
1708                    ;UNTERE GRENZE WIRD &0000 GESETZT
1709                    ;FENSTERGRENZE DURCH ZWEI
170B                    ;DIVIDIEREN ERGIBT REALE GRENZE
170D                    ;GRÖSSTE FENSTERGRENZE
1710                    ;LSB REALE FENSTERGRENZE NACH AKKU
```

```
1711                    ;LSB OK?
1712                    ;MSB REALE FENSTERGRENZE NACH AKKU
1713                    ;MSB OK?
1714                    ;GRÖSSTE FENSTERGRENZE
1715                    ;WIEDER ALTE FENSTERGRENZEN
1716                    ;SCHON FERTIG!
```

1717 ---------- GRA GET W WIDTH

Linke und rechte Begrenzung des Grafikfensters? Dazu muß sich im HL-Register die rechte, und im DE-Register die linke Fenstergrenze befinden.

```
1717                    ;(X-KOORDINATE GRA FENSTER LINKS)
171B                    ;(X-KOORDINATE GRA FENSTER RECHTS)
171E                    ;SCR GET MODE
1721                    ;MODE-1
1722                    ;MODE 1?
1725                    ;MODE-1
1726                    ;MODE 0?
1727                    ;LINKE X-KOORDINATE * 2
1728                    ;+1
1729                    ;HL = RECHTE X-KOORDINATE
172A                    ;RECHTE X-KOORDINATE * 2
172B                    ;HL = LINKE X-KOORDINATE
172C                    ;FERTIG!
```

172D ---------- GRA GET W HEIGHT

Obere und untere Begrenzung des Grafikfensters?

```
172D                    ;(Y-KOORDINATE GRA FENSTER OBEN)
1731                    ;(Y-KOORDINATE GRA FENSTER UNTEN)
1734                    ;FORTSETZUNG FOLGT
```


1736 ---------- GRA CLEAR WINDOW

Grafikfenster löschen.

```
1736                    ;GRA GET W WIDTH
1739                    ;CARRY LÖSCHEN
173A                    ;BREITE DES FENSTERS FÜR
173C                    ;HL-REGISTER BERECHNEN
173D                    ;WERT IN HL DURCH ZWEI TEILEN
1740                    ;WERT IN HL DURCH ZWEI TEILEN
1743                    ;LSB DURCH ZWEI TEILEN
1745                    ;ERGEBNIS INS B-REGISTER
1746                    ;(Y-KOORDINATE GRA FENSTER UNTEN)
174A                    ;(Y-KOORDINATE GRA FENSTER OBEN)
174D                    ;KOORDINATE AUF STACK RETTEN
174E                    ;CARRY FÜR SBC ZURÜCKSETZEN
174F                    ;FENSTERHÖHE FÜR
1751                    ;HL-REGISTER ERRECHNEN
1752                    ;LSB NACH C-REGISTER
1753                    ;(X-KOORDINATE GRA FENSTER LINKS)
1757                    ;Y-KOORDINATE OBEN
1758                    ;INHALT VOM BC-REGISTER AUF STACK RETTEN
1759                    ;SCR DOT POSITION
175C                    ;DE-REGISTER VOM STACK FÜLLEN
175D                    ;(GRA PAPER)
1760                    ;GRA PAPER INS C-REGISTER
1761                    ;SCR FLOOD BOX
1764                    ;GRAFIKCURSOR AUF DEFAULTPOSITION
```

1767 ---------- GRA SET PEN

Schreibfarbe setzen. Im Akkumulator erwartet GRA SET PEN die Farbstiftnummer.

```
1767                    ;SCR INK ENCODE
176A                    ;(GRA PEN)
176D                    ;FERTIG!
```

176E ---------- GRA SET PAPER

Hintergrundfarbe setzen. Im Akkumulator erwartet GRA SET PEN die Farbstiftnummer.

```
176E                    ;SCR IN ENCODE
1771                    ;(GRA PAPER)
1774                    ;DAS WAR'S
```

1775 ---------- GRA GET PEN

Welche Schreibfarbe? Diese Routine liefert den Farbstift der Schreibfarbe im Akkumulator.

```
1775                    ;(GRA PEN)
1778                    ;FORTSETZUNG FOLGT!
```

177A ---------- GRA GET PAPER

Welche Hintergrundfarbe? Diese Routine liefert den Farbstift der Hintergrundfarbe im Akkumulator.

```
177A                    ;(GRA PAPER)
177D                    ;SCR INK DECODE
```

1780 ---------- GRA PLOT RELATIVE

Grafikpunkt relativ zur aktuellen Cursorposition setzen.

```
1780                    ;ADD LFD KOORD. + REL KOORD.
```

1783 ---------- GRA PLOT ABSOLUTE

Grafikpunkt setzen (absolut).

```
1783                    ;GRA PLOT
```


1786 ---------- GRA PLOT

GRA PLOT erwartet beim Aufruf die Koordinaten für das Setzen des Punktes in den Registern HL und DE (HL = Y-Koordinate, DE = X-Koordinate).

```
1786                    ;POS IN WINDOW
1789                    ;POSITION AUSSERHALB DES FENSTER
178A                    ;SCR DOT POSITION
178D                    ;(GRA PEN)
1790                    ;GRA PEN INS B-REGISTER ÜBERSTELLEN
1791                    ;SCR WRITE
```

1794 ---------- GRA TEST RELATIVE

Punkt gesetzt (relativ zum lfd. Cursor)?

```
1794                    ;ADD LFD KOORD. + REL KOORD.
```

1797 ---------- GRA TEST ABSOLUTE

Punkt gesetzt (absolut)?

```
1797                    ;GRA TEST
```

179A ---------- GRA TEST

Gib die INK an der momentanen Grafikcursorposition im Akkumulator aus. Die Position des Grafikcursors wird durch die Registerpaare HL und DE bestimmt (HL = Y-Koordinate, DE = X-Koordinate).

```
179A                    ;POS IN WINDOW
179D                    ;GRA GET PAPER
17A0                    ;SCR DOT POSITION
17A3                    ;SCR READ
```

```
17A6 ---------------------------------------------- GRA LINE RELATIVE
```

Linie von der lfd. zur relativen Distanz ziehen.

```
17A6                    ;ADD LFD KOORD. + REL KOORD.
```

```
17A9 ------------------------------------------------ GRA LINE ABSOLUTE
```

Linie von der lfd. zur absoluten Position ziehen.

```
17A9                    ;GRA LINE
```

```
17AC ------------------------------------------------- GRA SET MASK
```

Parameter aus dem BASIC-Befehl MASK aus dem Akku retten.

```
17AC                    ;MASK PARAMETER
17AF                    ;FERTIG!
```

```
17B0 ------------------------------------------------ GRA FIRST POINT
```

Parameter aus dem BASIC-Befehl MASK aus dem Akku retten. Wird im Akku der Wert &FF übergeben, so soll der erste Punkt einer Linie gezeichnet werden. &00 steht für ersten Punkt nicht zeichnen.

```
17B0                    ;&FF ERSTEN PUNKT ZEICHNEN
                        ;&00 ERSTEN PUNKT NICHT ZEICHNEN
17B3                    ;DAS WAR'S
```

```
17B4 ------------------------------------------------------ GRA LINE
```

```
17B4                    ;ENDKOORDINATE ZEILE RETTEN
17B8                    ;ENDKOORDINATE ZEILE HOLEN
17B9                    ;PHYS ZIELPOS.
17BC                    ;HL-REGISTER SICHERN
```



```
17BD                    ;(RECHENPUFFER X-KOORD.)
17C0                    ;CARRY-FLAG ZURÜCKSETZEN
17C1                    ;DIFFERENZ ERRECHNEN
17C3                    ;MSB NACH AKKUMULATOR
17C4                    ;ERGEBNIS WEGSCHREIBEN
17C7                    ;KOORDINATE GRÖSSER?
17CA                    ;DE-REGISTER AUF STACK RETTEN
17CB                    ;HL-REGISTER VOM STACK LADEN
17CC                    ;(RECHENPUFFER Y-KOORD.)
17CF                    ;CARRY-FLAG ZURÜCKSETZEN
17D0                    ;DIFFERENZ ERMITTELN
17D2                    ;MSB NACH AKKUMULATOR
17D3                    ;UND INS RAM SCHREIBEN
17D6                    ;KOORDINATE GRÖSSER?
17D9                    ;Y-ENDKOORD. HOLEN
17DA                    ;CARRY FÜR SBC ZURÜCKSETZEN
17DB                    ;Y-DIFFERENZ BILDEN
17DD                    ;ENDKOORDINATE ADDIEREN
17DE                    ;CARRY ERMITTELN
17DF                    ;UND WEGSCHREIBEN
17E2                    ;AKKU MIT NEUEM WERT VERSORGEN
17E5                    ;Y-DIFF. GRÖSSER X-DIFF.?
17E7                    ;CHANGE DIFF.
17E8                    ;CHANGE SIGN
17EB                    ;VGL. &17E8
17EC                    ;VGL. &17E8
17F0                    ;VGL. &17E8
17F1                    ;VGL. &17E8
17F2                    ;ERSTES PIXEL
17F5                    ;PIXEL = LOW?
```

```
1940 ------------------------------------------------------ GRA WR CHAR
```

Ein Zeichen an der lfd. Grafikcursorposition schreiben.

```
1942                    ;TXT GET MATRIX
1962                    ;SCR DOT POSITION
1973                    ;SCR NEXT BYTE
197B                    ;SCR NEXT LINE
```

```
1985                    ;GRA ASK CURSOR
1998                    ;GRA MOVE ABSOLUTE
19AC                    ;SCR DOT POSITION
19C4                    ;(GRA PEN)
19CE                    ;(GRA PAPER)
19D2                    ;SCR WRITE

19D5 ---------------------------------------- GRA TRANS SWITCH

19D5                    ;PARAMETER FÜR HINTERGRUNDMODUS
19D8                    ;FERTIG
```


9.7 Die Macht der Maschinensprache

Maschinensprache - das ist Macht! Keine Hochsprache kann ihr in Punkto Geschwindigkeit und Kompaktheit das Wasser reichen, denn bis heute vermag keine Software eine Programmidee so zu optimieren wie der menschliche Geist. Doch leider ist die Maschinenprogrammierung - selbst für das menschliche Gehirn - eine zeitaufwendige und äußerst komplexe Angelegenheit. Dieser Umstand ist mit Sicherheit auch der Grund, warum viele Systemprogrammierer immer häufiger auf Hochsprachen zurückgreifen. Die Programmierung eines Problems läßt sich nämlich so in fast allen Fällen auf einen Bruchteil der sonst benötigten Zeit minimieren. Da jedoch selbst industrielle Compiler oft bei grafischen Problemen auf ihre Grenzen stoßen, behelfen sich viele Programmierer mit einer Verbindung aus Hochsprache und Maschinensprache. Betrachten wir die Struktur eines derartigen Programms genauer, werden wir feststellen, daß die Maschinensprache stets dann ihren Einsatz findet, wenn das mit ihrer Hilfe gelöste Problem entweder gar nicht oder nur schlecht mit Hochsprachen realisiert werden kann.

Im diesem Teil finden Sie eine Sammlung von Maschinenroutinen, die sowohl von Maschinenprogrammen als von der BASIC-Ebene aus in eigene Programme implementiert werden können. Das bedeutet, daß auch der reine BASIC-Programmierer die Macht der Maschinensprache in gewissen Grenzen nutzen kann.

9.7.1 Sprites auf dem CPC

Sprites sind selbst definierbare, überdimensional große Sonderzeichen, die oft aus mehreren Farben bestehen und an jeden beliebigen Punkt einer Rastergrafik projiziert werden können. Jedes Sprite wird dabei wie eine eigenständige Rastergrafik verwaltet und kann auf dem Bildschirm beliebig ein- oder ausgeblendet werden. Sprites spielen insbesondere auf dem Gebiet der Spieleprogrammierung eine bedeutende Rolle, da sie nicht den durch sie überlagerten Hintergrund, oder aber auch andere Sprites bei Kollisionen zerstören.

In der Regel werden Sprites hardwaremäßig mit einem Video-Controller generiert und belasten somit kaum die Rechenzeit des Hauptprozessors. Dieser muß nämlich nur noch die aktuellen Sprite-Positionen an den Video-Controller weitergeben.

Der HD 6845 des Schneider-CPC bietet uns hingegen - trotz seiner enormen Leistungsfähigkeit - nicht die Möglichkeit der interaktiven Sprite-Generierung. Viele CPC-Besitzer behelfen sich aus diesem Grund mit umdefinierten Charactern, die sie mittels TAG-Kommando über den Bildschirm gleiten lassen (siehe dazu auch Anhang). Damit sie den von ihnen überlagerten Hintergrund nicht zerstören, muß während der Animationsphase der XOR-Modus aktiviert sein. Obwohl dieses Verfahren bei einigen Anwendungen durchaus befriedigende Ergebnisse liefert, weist es doch bei objektiver Betrachtung erhebliche Schwächen auf.

Da ein Charakter nicht aus mehreren Farben bestehen kann, ist die Farbgebung umdefinierter Zeichenketten durch die Anzahl der verwendeten Charakter begrenzt. Weiterhin sinkt die Animationsgeschwindigkeit bei simultaner Bewegung mehrerer Character so stark ab, daß beispielsweise ein action-geladenes Spiel entweder kaum oder nur mit Verlust der fließenden Charakter-Bewegung zu realisieren ist. Bedauerlich ist auch die Tatsache, daß die eigentliche Form einer Zeichenkette oder eines Zeichens durch den XOR-Modus so stark entstellt werden kann, daß sie kaum wieder zu erkennen ist.

Um diese Schwachpunkte zu beseitigen, möchten wir Ihnen an dieser Stelle einen softwaremäßigen Sprite-Generator vorstellen. Er ist komplett in Maschinensprache geschrieben und läßt sich sowohl von BASIC als auch von Maschinenprogrammen ansteuern. Der Sprite-Generator kann maximal acht Sprites gleichzeitig verwalten und nimmt dabei inklusive Sprite-Daten weniger als 1,5 KByte ihres Hauptspeichers in Anspruch.

Spritespezifizierung

Die Größe eines generierten Sprites begrenzt sich auf eine 8 mal 16-Punkt-Matrix. Da der Sprite-Generator grundsätzlich in MODE 0 arbeitet, kann jedem dieser 128 Punkte eine von maximal 16 Farben zugewiesen werden. Um Ihnen die doch recht komplizierte Sprite-Definition zu erleichtern, werden wir zu einem späteren Zeitpunkt einen komfortablen Sprite-Editor zum Abtippen vorstellen. Selbstverständlich können die Sprites auch ohne dieses Programm definiert werden. Wo welches Bit wie gesetzt bzw. rückgesetzt werden muß, findet der interessierte Leser in Kapitel 9.4. Wir empfehlen Ihnen aber trotzdem, den Sprite-Editor zu verwenden, da selbst unter Zuhilfenahme eines Taschenrechners die vollständige Definition eines einzigen Sprite länger dauert als das Abtippen des gesamten Editors.

Die fertigen Sprite-Daten, die pro Sprite 64 Byte umfassen, müssen in einen fest definierten RAM-Bereich abgelegt werden. Dieser Bereich kann mit der Formel

```
X=&A0C8+SN*64
```

ermittelt werden, wobei &A0C8 die Basisadresse und SN die Sprite-Nummer (0 bis 7) repräsentiert. Einmal auf diese Weise definierte Sprites können selbstverständlich auf Diskette abgespeichert und bei Bedarf direkt in den zuständigen RAM-Bereich geladen werden. Die BASIC-Sequenz

```
SAVE"filename",B,&A0C8+SN*64,64
```

kann einen durch SN spezifizierten Sprite abspeichern. Zum Laden eines auf diese Weise abgespeicherten Sprite genügt ein einfacher LOAD-Befehl ohne Adreß-Angabe.

Die aktuelle Bildschirmposition eines Sprite muß in reservierte Speicherzellen, die sich innerhalb unseres Sprite-Generator-Pro-

gramms befinden, mit POKE-Befehlen übergeben werden. Diese Position (Koordinaten-Ursprung) bezieht sich auf die linke obere Ecke der Sprites. Ihre X- und Y-Koordinaten müssen im folgenden Werte-Bereich übergeben werden:

X = 0 bis 151
Y = 0 bis 183

Falls der übermittelte Wert den zulässigen Wertebereich überschreitet, wird dieser, um Fehlfunktionen zu vermeiden, automatisch in Position 0 umgewandelt.

Die reservierten Adressen, in denen die Sprite-Koordinaten übergeben werden müssen, können mit den beiden nachstehenden Formeln errechnet werden:

Adresse für X-Koordinate = &A559+2*SN
Adresse für Y-Koordinate = &A55A+2*SN

Die Adressen &A559 und &A55A sind dabei die Basisadressen bezogen auf Sprite 0. Die Variable SN enthält wieder unsere Sprite-Nummer (0 bis 7), die mit zwei multipliziert den erforderlichen Offset bildet, der zum Erreichen aller acht Sprite-Koordinatenpaare erforderlich ist.

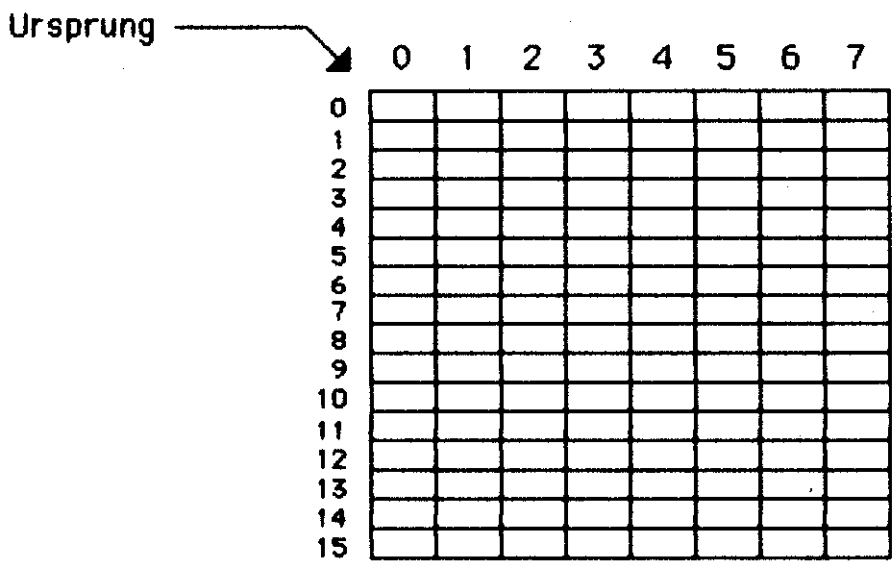


Abb. 24: Grundstruktur eines Sprite

Das Assemblerlisting:

```
100 ; ####################################
110 ; ##                                ##
120 ; ##   SPRITE-GENERATOR/3.10.86 TAV  ##
130 ; ##                                ##
140 ; ####################################
150 ;
160 ;
170         ORG  &A0C0
180 ;
190 FRAME:  EQU  &BD19      ;FRAME FLY BACK
200 ;
210 INIT:   LD   IX,NEWSP   ;IX=POS-COUNTER
220         CALL GBACK      ;SPRITEHINTERGRUND SICHERN
230         RET             ;RUECKSPRUNG ZU BASIC
```

```
240 ;
250 ;
260 ; ***** SPRITEDATEN *****
270 ;
280 SPR:    DEFS 64         ;DATEN FUER SPRITE 0
290         DEFS 64         ;DATEN FUER SPRITE 1
300         DEFS 64         ;DATEN FUER SPRITE 2
310         DEFS 64         ;DATEN FUER SPRITE 3
320         DEFS 64         ;DATEN FUER SPRITE 4
330         DEFS 64         ;DATEN FUER SPRITE 5
340         DEFS 64         ;DATEN FUER SPRITE 6
350         DEFS 64         ;DATEN FUER SPRITE 7
360 ;
370 BACK:   DEFS 80         ;SPEICHER FUER SPRITE-HINTERGR. 0
380         DEFS 80         ;SPEICHER FUER SPRITE-HINTERGR. 1
390         DEFS 80         ;SPEICHER FUER SPRITE-HINTERGR. 2
400         DEFS 80         ;SPEICHER FUER SPRITE-HINTERGR. 3
410         DEFS 80         ;SPEICHER FUER SPRITE-HINTERGR. 4
420         DEFS 80         ;SPEICHER FUER SPRITE-HINTERGR. 5
430         DEFS 80         ;SPEICHER FUER SPRITE-HINTERGR. 6
440         DEFS 80         ;SPEICHER FUER SPRITE-HINTERGR. 7
450 ;
460 ONOFF:  DEFB 0          ;SPRITE EIN-/AUS-SCHALTER
470 ;
480 OLDSP:  DEFW 0          ;ALTE POSITION (SPRITE 0)
490         DEFW 0          ;ALTE POSITION (SPRITE 1)
500         DEFW 0          ;ALTE POSITION (SPRITE 2)
510         DEFW 0          ;ALTE POSITION (SPRITE 3)
520         DEFW 0          ;ALTE POSITION (SPRITE 4)
530         DEFW 0          ;ALTE POSITION (SPRITE 5)
540         DEFW 0          ;ALTE POSITION (SPRITE 6)
550         DEFW 0          ;ALTE POSITION (SPRITE 7)
560 ;
570 NEWSP:  DEFW 0          ;NEUE POSITION (SPRITE 0)
580         DEFW 0          ;NEUE POSITION (SPRITE 1)
590         DEFW 0          ;NEUE POSITION (SPRITE 2)
600         DEFW 0          ;NEUE POSITION (SPRITE 3)
610         DEFW 0          ;NEUE POSITION (SPRITE 4)
620         DEFW 0          ;NEUE POSITION (SPRITE 5)
```



```
630         DEFW 0          ;NEUE POSITION (SPRITE 6)
640         DEFW 0          ;NEUE POSITION (SPRITE 7)
650 ;
660 ;
670 ; ***** SPRITE-HINTERGRUND ZURUESCHSCHREIBEN *****
680 ;
690 START:  CALL FRAME      ;AUF STRAHLENRUECKLAUF WARTEN
700         LD   C,&08      ;SCHLEIFE INITIALISIEREN
710         LD   IX,OLDSP   ;IX = POSITIONS-POINTER
720         LD   HL,BACK    ;POINTER AUF HINTERGRUNDSPEICHER
730 ;
740 GETB:   CALL CLCPOS     ;BERECHNE START-BYTE
750 ;
760         LD   B,&10      ;LINE-COUNTER INIT
770 PLINE:  PUSH BC         ;RETTE COUNTER
780         LD   BC,&0005   ;5 BYTES PRO ZEILE
790         PUSH DE         ;VIDEO-POINTER RETTEN
800         LDIR            ;ZEILE ZURUECKSCHREIBEN
810         POP  DE         ;VIDEO-POINTER HOLEN
820         LD   A,&08      ;DE = DE + &0800
830         ADD  A,D        ;      "
840         LD   D,A        ;      "
850         JP   NC,NOOFF   ;NEGATIVER ZEILENOFFSET?
860         EX   DE,HL      ;HL = VIDEO-POINTER
870         LD   BC,&3FAF   ;ZEILENOFFSET FUER SBC
880         SBC  HL,BC      ;ENTGUELTIGE ZEILE
890         EX   DE,HL      ;HL = HINTERGR. POINTER
900 ;
910 NOOFF:  POP  BC         ;HOLE LINE-COUNTER
920         DJNZ PLINE      ;NAECHSTE ZEILE
930 ;
940         DEC  C          ;NAECHSTER SPRITEHINTERGRUND
950         JP   NZ,GETB    ;      "
960 ;
970 ;
980 ; ***** SPRITE-HINTERGRUND HOLEN *****
990 ;
1000        CALL GBACK      ;HOLE HINTERGRUND
1010 ;
```

```
1020 ;
1030 ; ***** SPRITES ZEICHNEN *****
1040 ;
1050         LD   C,&01       ;SCHLEIFENZAEHLER INIT
1060         LD   IX,NEWSP    ;POINTER AUF SPRITEKOORDINATEN
1070         LD   HL,SPR      ;HL = SPRITE-POINTER
1080 ;
1090 PUTSPR: LD   A,(ONOFF)   ;SPRITE AKTIVIERT?
1100         AND  C           ;      "
1110         JR   Z,NOSPR     ;      "
1120 ;
1130         CALL CLCPOS      ;BERECHNE START-BYTE
1140 ;
1150         LD   B,&10       ;LINE-COUNTER
1160 ;
1170 DSPRLN: PUSH BC          ;RETTE COUNTER
1180         PUSH DE          ;RETTE VIDEO-POINTER
1190         LD   BC,&0004    ;4 BYTES PRO ZEILE
1200         LD   A,I         ;HOLE X
1210         AND  %00000001   ;BILDSCHIRMADR. GERADE?
1220         JR   Z,EVEN      ;      "
1230 ;
1240         LD   B,C         ;COUNTER NACH B
1250         LD   A,(DE);     ;HOLE BYTE VON DISPLAY
1260 NGSN:   AND  %10101010   ;UND BLENDE RECHTES PIXEL AUS
1270         LD   C,A         ;ERGEBNIS NACH C
1280         LD   A,(HL)      ;HOLE SPRITEDATEN
1290         RRCA             ;VERSCHIEBE PIXEL NACH RECHTS
1300         AND  %01010101   ;UND BLENDE FALSCHE PIXEL AUS
1310         OR   C           ;VERBINDE MIT HINTERGRUND
1320         LD   (DE),A      ;FERTIGES BYTE ZURUECKSCHREIBEN
1330         INC  DE          ;POINTER AUF NAECHSTES BYTE
1340         LD   A,(DE)      ;HOLE BYTE VON DISPLAY
1350         AND  %01010101   ;UND BLENDE LINKES PIXEL AUS
1360         LD   C,A         ;ERGEBNIS NACH C
1370         LD   A,(HL)      ;HOLE SPRITEDATEN
1380         RLCA             ;VERSCHIEBE PIXEL NACH LINKS
1390         AND  %10101010   ;UND BLENDE FALSCHE PIXEL AUS
1400         OR   C           ;VERBINDE MIT HINTERGRUND
```



```
1410         INC  HL          ;NAECHSTE SPRITEDATEN
1420 ;
1430         DJNZ NGSN        ;NAECHSTES NIBBLE VERARBEITEN
1440         LD   (DE),A      ;LETZTES BYTE ZURUECKSCHREIBEN
1450         JR   OFFS        ;UND NAECHSTE ZEILE HOLEN
1460 ;
1470 EVEN:   LDIR             ;ZEILE KOPIEREN
1480 ;
1490 OFFS:   POP  DE          ;VIDEO-POINTER HOLEN
1500         LD   A,&08       ;DE = DE + &0800
1510         ADD  A,D         ;      "
1520         LD   D,A         ;      "
1530         JP   NC,NOOFF2   ;NEGATIVER ZEILENOFFSET?
1540         EX   DE,HL       ;HL = VIDEO-POINTER
1550         LD   BC,&3FAF    ;ZEILENOFFSET FUER SBC
1560         SBC  HL,BC       ;ENTGUELTIGE ZEILE
1570         EX   DE,HL       ;HL = HINTERGR. POINTER
1580 ;
1590 NOOFF2: POP  BC          ;HOLE COUNTER
1600         DJNZ DSPRLN      ;NAECHSTE ZEILE
1610 MVEIN:  SLA  C           ;HOLE NAECHSTEN SPRITE
1620         RET  Z           ;FERTIG???
1630         JR   PUTSPR      ;WENN NICHT...
1640 ;
1650 NOSPR:  LD   DE,&40      ;UEBERSPRINGE SPRITE
1660         ADD  HL,DE       ;      "
1670         INC  IX          ;      "
1680         INC  IX          ;      "
1690         JR   MVEIN       ;      "
1700 ;
1710 ;
1720 ; ***** SPRITE-HINTERGRUND SPEICHERN *****
1730 ;
1740 GBACK:  LD   C,&08       ;SCHLEIFE INITIALISIEREN
1750         LD   HL,BACK     ;POINTER AUF HINTERGRUNDSPEICHER
1760 ;
1770 GETB2:  CALL CLCPOS      ;BERECHNE START-BYTE
1780 ;
1790         LD   B,&10       ;LINE-COUNTER INIT
```

```
1800         EX   DE,HL       ;HL = VIDEO-POINTER
1810 GLINE:  PUSH BC          ;RETTE COUNTER
1820         LD   BC,&0005    ;5 BYTES PRO ZEILE
1830         PUSH HL          ;VIDEO-POINTER RETTEN
1840         LDIR             ;ZEILE ZURUECKSCHREIBEN
1850         POP  HL          ;VIDEO-POINTER HOLEN
1860         LD   BC,&0800    ;HL = HL + &0800
1870         ADD  HL,BC       ;      "
1880         JP   NC,NOOFF3   ;NEGATIVER ZEILENOFFSET?
1890         LD   BC,&3FAF    ;ZEILENOFFSET FUER SBC
1900         SBC  HL,BC       ;ENTGUELTIGE ZEILE
1910 ;
1920 NOOFF3: POP  BC          ;HOLE LINE-COUNTER
1930         DJNZ GLINE       ;NAECHSTE ZEILE
1940 ;
1950         EX   DE,HL       ;HL = (BACK)
1960         DEC  C           ;NAECHSTER SPRITEHINTERGRUND
1970         JP   NZ,GETB2    ;          "
1980 ;
1990         LD   DE,OLDSP    ;ALTE KOORDINATEN
2000         LD   HL,NEWSP    ;= NEUE KOORDINATEN
2010         LD   BC,16       ;INSGES. 16 STUECK
2020         LDIR             ;KOPIEREN
2030 ;
2040         RET              ;GESCHAFFT!
2050 ;
2060 ;
2070 ; ***** BILDSCHIRMADRESSE BERECHNEN *****
2080 ;
2090 CLCPOS: PUSH HL          ;HL UND BC RETTEN
2100         PUSH BC          ;      "
2110         LD   A,(IX+&00)  ;A = X
2120         CP   152         ;X <= 151 ?
2130         JR   C,SVRR      ;   "
2140         XOR  A           ;X = 0
2150 SVRR:   LD   C,A         ;C = X
2160         LD   I,A         ;I = X
2170 ;
2180         LD   A,(IX+&01)  ;A = Y
```



```
2190         CP   184         ;Y <= 183 ?
2200         JR   C,SVOR      ;   "
2210         XOR  A           ;X = 0
2220 SVOR:   LD   B,A         ;C = X
2230 ;
2240         AND  %00000111   ;BLENDE BIT-NUMMER AUS
2250 ;
2260         SRL  B           ;Y = Y/8
2270         SRL  B           ;   "
2280         SRL  B           ;   "
2290 ;
2300         LD   HL,&C000    ;HL = VIDEO-START
2310         JR   Z,NOMUL1    ;MULTIPLIKATOR = 0?
2320         LD   DE,80       ;HL = Y/8 * 80
2330 MUL1:   ADD  HL,DE       ;      "
2340         DJNZ MUL1        ;      "
2350 ;
2360 NOMUL1: AND  A           ;MULTIPLIKATOR = 0?
2370         JR   Z,NOMUL2    ;      "
2380         LD   B,A         ;SCHLEIFEN INIT
2390         LD   DE,&0800    ;HL = HL+(Y-Y/8*8)*&800
2400 MUL2:   ADD  HL,DE       ;      "
2410         DJNZ MUL2        ;      "
2420 ;
2430 NOMUL2: LD   E,C         ;E = X
2440         SRL  E           ;X=X/2
2450         LD   D,&00       ;16-BIT ADDITION VORBEREITEN
2460         ADD  HL,DE       ;HL = VIDEO-ADRESSE
2470         EX   DE,HL       ;DE = VIDEO-ADRESSE
2480 ;
2490         INC  IX          ;POINTER AUF NAECHSTE KOORDINATEN
2500         INC  IX          ;    "
2510 ;
2520         POP  BC          ;BC UND HL HOLEN
2530         POP  HL          ;      "
2540         RET              ;FERTIG!
2550 ;
2560         END              ;PROGRAMMENDE
```

Und für unsere BASIC-Freunde der BASIC-Lader:

```
100 '######################################
110 '##                                  ##
120 '## BASIC-LADER FUER SPRITE-GENERATOR ##
130 '##          10.10.1986/TAV          ##
140 '##                                  ##
150 '######################################
160 '
170 '
180 MEMORY &A0BF
190 FOR I=&A0C0 TO &A0C7:READ X:POKE I,X:NEXT
200 FOR I=&A0C8 TO &A568:POKE I,0:NEXT
210 FOR I=&A569 TO &A66A:READ X:POKE I,X:NEXT
220 END
230 '
240 DATA 221,33,89,165,205,247,165,201,205,25,189,14,8,221,33
250 DATA 73,165,33,200,162,205,42,166,6,16,197,1,5,0,213,237
260 DATA 176,209,62,8,130,87,210,144,165,235,1,175,63,237,66,235
270 DATA 193,16,231,13,194,117,165,205,247,165,14,1,221,33,89,165
280 DATA 33,200,160,58,72,165,161,40,68,205,42,166,6,16,197,213
290 DATA 1,4,0,237,87,230,1,40,27,65,26,230,170,79,126,15
300 DATA 230,85,177,18,19,26,230,85,79,126,7,230,170,177,35,16
310 DATA 234,18,24,2,237,176,209,62,8,130,87,210,229,165,235,1
320 DATA 175,63,237,66,235,193,16,198,203,33,200,24,182,17,64,0
330 DATA 25,221,35,221,35,24,241,14,8,33,200,162,205,42,166,6
340 DATA 16,235,197,1,5,0,229,237,176,225,1,0,8,9,210,22
350 DATA 166,1,175,63,237,66,193,16,233,235,13,194,252,165,17,73
360 DATA 165,33,89,165,1,16,0,237,176,201,229,197,221,126,0,254
370 DATA 152,56,1,175,79,237,71,221,126,1,254,184,56,1,175,71
380 DATA 230,7,203,56,203,56,203,56,33,0,192,40,6,17,80,0
390 DATA 25,16,253,167,40,7,71,17,0,8,25,16,253,89,203,59
400 DATA 22,0,25,235,221,35,221,35,193,225,201
```


Organisation und Arbeitsweise des Sprite-Generators

Unser Sprite-Generator arbeitet nach einem einfachen, aber sehr effektiven Verfahren. Unmittelbar nach dem Start Ihres Trägerprogramms muß er einmalig durch einen

```
CALL &A0C0
```

initialisiert werden. Dies ist notwenig, um den Hintergrund aller acht möglichen Sprites in einen speziellen, im Generator-Programm vorhandenen Speicherbereich zu transferieren. Im nächsten Schritt werden die Hintergrund-Koordinaten aller acht Sprites für einen späteren Zugriff kopiert.

Jetzt ist das Sprite-Generator-Programm prinzipiell einsatzbereit. Doch bevor Ihre Sprites animiert werden können, muß erst spezifiziert werden

- wieviel Sprites definiert wurden,
- wieviel Sprites gleichzeitig bewegt werden sollen und
- welche Priorität einem Sprite zugeordnet wird.

Die dafür benötigten Parameter werden in einem einzigen Byte abgelegt - der RAM-Adresse &A548. Die Bits 0 bis 7 ihres Inhalts bestimmen, ob ein Sprite an- oder ausgeschaltet ist sowie seine Priorität. Unter Priorität verstehen wir die Position eines Sprite auf einer imaginären Z-Achse. Imaginär deshalb, weil sich die Z-Achse nur auf Sprites und nicht auf die übrige Grafik bezieht. Da maximal acht Sprites definiert werden können, gibt es auch acht Prioritäten mit den Stufen 0 bis 7. Diese Prioritätsstufen entsprechen den Bit-Nummern unseres Steuerbytes. Falls zwei Sprites kollidieren, überlappt immer der Sprite mit der höheren Prioritätsstufe den mit der niedrigeren. Die nachstehende Tabelle verdeutlicht noch einmal diese Zusammenhänge:

Bit 0: (POKE &A548,&01) Sprite 1 aktivieren (Prioritätsstufe 0)
Bit 1: (POKE &A548,&02) Sprite 2 aktivieren (Prioritätsstufe 1)
Bit 2: (POKE &A548,&04) Sprite 3 aktivieren (Prioritätsstufe 2)
Bit 3: (POKE &A548,&08) Sprite 4 aktivieren (Prioritätsstufe 3)
Bit 4: (POKE &A548,&10) Sprite 5 aktivieren (Prioritätsstufe 4)
Bit 5: (POKE &A548,&20) Sprite 6 aktivieren (Prioritätsstufe 5)
Bit 6: (POKE &A548,&40) Sprite 7 aktivieren (Prioritätsstufe 6)
Bit 7: (POKE &A548,&80) Sprite 8 aktivieren (Prioritätsstufe 7)

Wie aus dieser Tabelle ersichtlich ist, entspricht einem gesetzten Bit ein aktivierter Sprite. Sollen mehrere Sprites gleichzeitig eingeschaltet werden, müssen die oben aufgeführten POKE-Argumente der betreffenen Sprite-Nummern addiert werden. Beispielsweise aktiviert

```
POKE &A548,&42
```

Sprite Nummer zwei und sieben. Die übrigen Sprites bleiben dabei ausgeschaltet.

Sind auf diese Weise alle Sprites spezifiziert, erwartet der Sprite-Generator die aktuelle Position der zu bewegenden Sprites in den oben bereits beschriebenen Adressen, die für deren X- und Y-Koordinaten reserviert sind. Unmittelbar danach sollten mit

```
CALL &A569
```

alle aktivierten Sprites animiert, d.h. den neuen Koordinaten angepaßt werden. Das Sprite-Generator-Programm wartet bei diesem Aufruf zuerst auf den Strahlenrücklauf, damit die nachfolgenden Schritte so flackerfrei wie möglich ausgeführt werden können. Danach schreibt eine Routine den zuvor geretteten Hintergrund zurück und löscht somit alle alten Sprites. Nun wird der Hintergrund, der durch die gerade übergebene aktuelle

Position definiert ist, erneut gerettet. Weiterhin werden die aktuellen Koordinaten kopiert, um ein späteres Zurückschreiben des Hintergrundes zu ermöglichen. Im letzen Schritt transferiert unser Programm alle aktivierten Sprites gemäß ihrer Prioritätsstufe auf dem Bildschirm.

Um Ihnen an einem praktischen Beispiel den Einsatz des Sprite-Generators zu demonstrieren, folgt an dieser Stelle ein kleines Demo-Programm. Bitte beachten Sie, daß sich vor Starten dieses Programms der Code des Sprite-Generators bereits im Speicher befinden muß.

```
100 '####################
110 '##                  ##
120 '##   SPRITE-DEMO    ##
130 '## 10. 10. 86/TAV   ##
140 '##                  ##
150 '####################
160 '
170 '
180 MEMORY &A0BF
190 DEFINT A-Z
200 '
210 '
220 '***** BILDSCHIRMPARAMETER SPEZIFIZIEREN
230 '
240 MODE 0
250 BORDER 0:PAPER 5:CLS
260 INK 14,13:INK 15,11
270 '
280 '
290 '***** KOMMANDOS GENERIEREN
300 '
310 ONOFF=&A548
320 INIT=&A0C0
330 SPRITE=&A569
340 SPRDAT=&A0C8
350 SPRPOS=&A559
360 '
```

```
370 '
380 '***** SPRITES EINLESEN
390 '
400 FOR I=SPRDAT TO SPRDAT+191
410 READ X
420 POKE I,X
430 NEXT
440 '
450 '
460 '***** SPRITEGENERATOR INITIALISIEREN
470 '
480 CALL INIT
490 '
500 '
510 '***** STERNE ERZEUGEN
520 '
530 FOR I=1 TO 100
540 PLOT(RND(1)*639),(RND(1)*399),14
550 NEXT
560 '
570 '
580 '***** SPRITES PROJIZIEREN
590 '
600 POKE ONOFF,3
610 Z=0
620 '
630 FOR I=0 TO 151
640 POKE SPRPOS,I:POKE SPRPOS+1,I
650 POKE SPRPOS+2,151-I:POKE SPRPOS+3,151-I
660 IF Z=2 AND I=75 THEN 750
670 CALL SPRITE
680 NEXT
690 Z=Z+1
700 GOTO 630
710 '
720 '
730 '***** EXPLOSION
740 '
750 POKE SPRPOS+4,I:POKE SPRPOS+5,I
760 POKE ONOFF,4
```



```
770 CALL SPRITE
780 FOR I=1 TO 500:NEXT
790 POKE ONOFF,0
800 CALL SPRITE
810 FOR I=1 TO 1000:NEXT
820 GOTO 600
830 '
840 '
850 '***** SPRITEDATEN FUER 1. JAEGER
860 '
870 DATA 240,240,240,240
880 DATA 240,240,240,240
890 DATA 240,240,240,240
900 DATA 240,240,240,240
910 DATA 210,240,240,240
920 DATA 3,240,240,240
930 DATA 139,3,82,240
940 DATA 204,63,35,82
950 DATA 161,23,63,63
960 DATA 240,240,240,240
970 DATA 240,240,240,240
980 DATA 240,240,240,240
990 DATA 240,240,240,240
1000 DATA 240,240,240,240
1010 DATA 240,240,240,240
1020 DATA 240,240,240,240
1030 '
1040 '
1050 '***** SPRITEDATEN FUER 2. JAEGER
1060 '
1070 DATA 240,240,240,240
1080 DATA 240,240,240,240
1090 DATA 240,240,240,240
1100 DATA 240,240,240,240
1110 DATA 240,240,240,240
1120 DATA 240,240,240,176
1130 DATA 240,240,240,18
1140 DATA 240,165,11,101
1150 DATA 245,58,48,204
1160 DATA 60,60,41,82
```

```
1170 DATA 240,240,240,240
1180 DATA 240,240,240,240
1190 DATA 240,240,240,240
1200 DATA 240,240,240,240
1210 DATA 240,240,240,240
1220 DATA 240,240,240,240
1230 '
1240 '
1250 '***** SPRITEDATEN FUER EXPLOSION
1260 '
1270 DATA 240,240,240,240
1280 DATA 240,240,240,240
1290 DATA 240,240,240,240
1300 DATA 240,216,205,240
1310 DATA 240,229,228,240
1320 DATA 228,240,218,240
1330 DATA 240,206,206,240
1340 DATA 240,156,109,240
1350 DATA 240,228,109,216
1360 DATA 240,229,218,240
1370 DATA 240,216,216,216
1380 DATA 240,240,240,240
1390 DATA 240,229,229,240
1400 DATA 240,240,240,240
1410 DATA 240,240,240,240
1420 DATA 240,240,240,240
```

Vielleicht haben Sie bemerkt, daß die Sprites während der Animationsphase leicht flackern. Das liegt daran, daß die Ausführungszeit, die unser Sprite-Generator für einen Berechnungsvorgang benötigt, mehr Zeit in Anspruch nimmt, als ein Strahlenrücklauf des Kathodenstrahls. Um diesen Effekt zu eliminieren, muß das Programm mit zwei voneinander unabhängigen Bildschirmseiten arbeiten. Dabei werden alle Sprites auf einer unsichtbaren Bildschirmseite erstellt und danach die sichtbare Bildschirmseite mit der unsichtbaren vertauscht. Wie eine derartige Verwaltung programmtechnisch zu realisieren ist, erfahren Sie in einem späteren Abschnitt.

Anmerkung: Falls Sie einen CPC 464 besitzen und den Sprite-Generator mit einem BASIC-Programm ansteuern, ist von einer zweiseitigen Bildschirmverwaltung abzuraten, da die Organisation der zweiten Bildschirmseite 16 KByte in Anspruch nimmt.

Der Sprite-Editor

Sprites sind - insbesondere in MODE 0 - aufwendig zu definieren. Um diesen Aufwand auf ein Minimum zu reduzieren, möchten wir zum Abschluß noch einen Sprite-Editor vorstellen. Er ist vollkommen in BASIC geschrieben, und ermöglicht die komfortable und schnelle Definition einer 8 mal 16-Punkte-Matrix. Weiterhin enthält das Programm einen DATA-Generator, der auf Wunsch die Daten des fertiggestellten Sprite errechnet, in DATA-Zeilen umwandelt und diese als BASIC-Programm abspeichert. Die so entstandenen Programm-Files können dann mit dem MERGE-Befehl mit Ihrem Hauptprogramm verbunden werden.

Befehlssatz:

CURSOR-Tasten

Mit den Cursor-Tasten wird der Cursor der 8 mal 16 Punkte-Matrix in alle Richtungen bewegt.

Pfeil oben oder Pfeil unten + SHIFT

Wechselt die PEN-Nummer (Cursor-Farbe) in auf- oder absteigender Reihenfolge.

Leertaste (Space)

Durch das Betätigen der Leertaste wird der Punkt unter dem Cursor in der eingestellten Farbe gesetzt.

Taste "<"

Verschiebt alle Pixel um eine Stelle nach links. Die dabei aus der Matrix geschobenen Punkte werden rechts angefügt.

Taste ">"

Verschiebt alle Pixel um eine Stelle nach rechts. Die dabei aus der Matrix geschobenen Punkte werden links angefügt.

Taste "A"

Verschiebt alle Pixel um eine Stelle nach oben. Die dabei aus der Matrix geschobenen Punkte werden unten angefügt.

Taste "Z"

Verschiebt alle Pixel um eine Stelle nach unten. Die dabei aus der Matrix geschobenen Punkte werden oben angefügt.

Taste "S"

Sprite sichern. Diese Routine öffnet ein Fenster und fragt nach einem Dateinamen. Danach wird der Sprite unter dem eingegebenen Namen auf Diskette abgelegt.

Taste "L"

Sprite laden. Diese Routine öffnet ein Fenster und fragt nach einem Dateinamen. Danach wird der Sprite unter dem eingegebenen Namen von Diskette geladen.

Taste"D"

DATA-Zeilen generieren. Diese Routine öffnet ein Fenster und fragt nach einem Dateinamen. Danach werden die Daten des Sprite automatisch in DATA-Zeilen umgewandelt und diese unter dem eingegebenen Namen auf Diskette als BASIC-Programm abgespeichert. (Das so erzeugte Programm kann mit dem MERGE-Befehl mit jedem anderen BASIC-Programm verknüpft werden.)

Taste "N"

Neu - der aktuelle Sprite wird gelöscht.

Taste "E"

Verlassen des Sprite-Editors.

```
100 '####################
110 '##                ##
120 '## SPRITE-EDITOR ##
130 '##  8.10.86/TAV   ##
140 '##                ##
150 '####################
160 '
170 '
172 '***** INITIALISIERUNG *****
175 '
180 MODE 0:INK 14,13:INK 15,11:PEN 1:PAPER 0:DEFINT A-Z
190 DIM P(7,15)
```

```
200 C=4:FOR I=0 TO 15:FOR J=0 TO 7:P(J,I)=5:NEXT:NEXT
210 WINDOW #1,3,17,15,18
220 SYMBOL AFTER 254
230 SYMBOL 255,0,127,127,127,127,127,127,0
240 SYMBOL 254,0,85,42,85,42,85,42,0
242 '
244 '
246 '***** BILDSCHIRMMASKE AUFBAUEN *****
248 '
250 PRINT "####################";
260 PRINT "##                  ##";
270 PRINT "##";:PEN 3:PRINT" SPRITE-EDITOR  ";:PEN 1:PRINT"##";
280 PRINT "##";:PEN 3:PRINT"  8.10.86/TAV   ";:PEN 1:PRINT"##";
290 PRINT "##                  ##";
300 PRINT "####################";
310 PEN 3:LOCATE 1,11:GOSUB 550:LOCATE 1,19:GOSUB 550
320 FOR I=0 TO 15:LOCATE 7,9+I:FOR J=0 TO 7:PLOT 32+J*4,94-I*2,P(J,I):PE
N P(J,I):PRINT CHR$(255);:NEXT
330 PEN 1:PRINT" P";:PEN 3:PRINT USING"##";I;:PEN I:PRINT " ";CHR$(143);
:NEXT
340 LOCATE X+7,Y+9:PEN C:PRINT CHR$(254);:LOCATE 2,12:PRINT CHR$(143);:L
OCATE 2,13:PRINT CHR$(143);
350 LOCATE 17,C+9:PEN 4:PRINT USING"##";C;
352 '
354 '
356 '***** ABFRAGE DER KOMMANDOS UND DEREN DEKODIERUNG *****
358 '
360 Y$=INKEY$:Y$=UPPER$(Y$)
370 IF Y$=CHR$(240) THEN GOSUB 540:Y=Y-1:IF Y<0 THEN Y=15
380 IF Y$=CHR$(241) THEN GOSUB 540:Y=Y+1:IF Y>15 THEN Y=0
390 IF Y$=CHR$(242) THEN GOSUB 540:X=X-1:IF X<0 THEN X=7
400 IF Y$=CHR$(243) THEN GOSUB 540:X=X+1:IF X>7 THEN X=0
410 IF Y$=" " THEN P(X,Y)=C:PLOT 32+X*4,94-Y*2,C
420 IF Y$=CHR$(244) THEN LOCATE 17,C+9:PEN 3:PRINT USING"##";C;:C=C-1:IF
 C<0 THEN C=15
430 IF Y$=CHR$(245) THEN:LOCATE 17,C+9:PEN 3:PRINT USING"##";C;:C=C+1:IF
 C>15 THEN C=0
440 IF Y$="." THEN FOR J=0 TO 15:TMP=P(7,J):FOR I=6 TO 0 STEP-1:P(I+1,J)
=P(I,J):NEXT:P(0,J)=TMP:NEXT:GOTO 320
```



```
450 IF Y$="," THEN FOR J=0 TO 15:TMP=P(0,J):FOR I=1 TO 7:P(I-1,J)=P(I,J)
:NEXT:P(7,J)=TMP:NEXT:GOTO 320
460 IF Y$="A" THEN FOR I=0 TO 7:TMP=P(I,0):FOR J=1 TO 15:P(I,J-1)=P(I,J)
:NEXT:P(I,15)=TMP:NEXT:GOTO 320
470 IF Y$="Z" THEN FOR I=0 TO 7:TMP=P(I,15):FOR J=14 TO 0 STEP-1:P(I,J+1
)=P(I,J):NEXT:P(I,0)=TMP:NEXT:GOTO 320
480 IF Y$="N" THEN RUN
490 IF Y$="S" THEN 600
500 IF Y$="L" THEN 700
510 IF Y$="D" THEN 800
520 IF Y$="E" THEN WINDOW SWAP 0,1:PAPER 5:PEN 4:CLS:PRINT:INPUT"PROGRAM
MENDE";NM$:NM$=UPPER$(NM$):IF NM$="J" THEN MODE 2:PEN 1:PAPER 0:END ELSE
 750
530 GOTO 340
540 LOCATE X+7,Y+9:PEN P(X,Y):PRINT CHR$(255);:RETURN
550 PRINT CHR$(150);CHR$(154);CHR$(156):PRINT CHR$(149);" ";CHR$(149):PR
INT CHR$(149);" ";CHR$(149):PRINT CHR$(147);CHR$(154);CHR$(153);:RETURN
560 '
570 '
580 '***** SPRITE SICHERN *****
590 '
600 WINDOW SWAP 0,1:PAPER 5:PEN 4:CLS
610 PRINT"SPRITE SICHERN:":PRINT:INPUT"NAME";NM$
620 OPENOUT NM$
630 FOR J=0 TO 15:FOR I=0 TO 7:WRITE #9,P(I,J):NEXT:NEXT
640 CLOSEOUT
650 GOTO 750
660 '
670 '
680 '***** SPRITE LADEN *****
690 '
700 WINDOW SWAP 0,1:PAPER 5:PEN 4:CLS
710 PRINT"SPRITE LADEN:":PRINT:INPUT"NAME";NM$
720 OPENIN NM$
730 FOR J=0 TO 15:FOR I=0 TO 7:INPUT #9,P(I,J):NEXT:NEXT
740 CLOSEIN
750 PAPER 0:PEN C:CLS:WINDOW SWAP 0,1:GOTO 320
760 '
770 '
```

```
780 '***** DATAS GENERIEREN UND SPEICHERN *****
790 '
800 WINDOW SWAP 0,1:PAPER 5:PEN 4:CLS
810 PRINT"DATA GENERATOR:":PRINT:INPUT"NAME";NM$
820 OPENOUT NM$
830 PRINT#9,"30000 'DIES IST EIN AUTOMATISCH ERZEUGTES PROGRAMMFILE"
840 Z=0:TMP=&C5F4:GOSUB 880
850 TMP=&C644:GOSUB 880
860 CLOSEOUT
870 GOTO 750
880 FOR I=0 TO 7
890 D$=STR$(30001+Z)+"DATA ":FOR J=0 TO 3
900 P$=STR$(PEEK(TMP+J+&800*I)):D$=D$+RIGHT$(P$,LEN(P$)-1)+","
910 NEXT
920 PRINT#9,LEFT$(D$,LEN(D$)-1):Z=Z+1
930 NEXT
940 RETURN
```

9.7.2 Das Organisieren mehrerer Bildschirmseiten

Dieses Unterkapitel wendet sich in erster Linie an die Besitzer eines CPC 464 und 664, da der CPC 6128 mit den BANK-Befehlen |SCREENCOPY und |SCREENSWAP eine einfache, und zugleich effektive Programmierung mehrerer Bildschirmseiten aus der BASIC-Ebene heraus gestattet.

Das Arbeiten mit zwei unabhängig von einander organisierten Bildschirmseiten - einer sichtbaren und einer unsichtbaren - ist in einigen Anwendungsbereichen obligatorisch. Beispielsweise kann auf der unsichtbaren Bildschirmseite ein beliebiges Objekt gezeichnet werden, während die sichtbare (aktuelle) Bildschirmseite das gleiche Objekt in einer anderen Form oder auf einer einer anderen Position auf dem Bildschirm zeigt. Wechselt man danach die Bildschirmseiten, kann ein flackerfreier Trickfilmeffekt erzeugt werden. Eine praktische Anwendung dieser Technik finden Sie übrigens in Kapitel 7.4.

Eine weitere, nicht zu unterschätzende Anwendung ist das Speichern von zwei vollkommen verschiedenen und zugleich

kompliziert aufgebauten Grafiken. Möchte der Benutzer auf die unsichtbare Grafik zugreifen, muß das Programm lediglich die sichtbare gegen die unsichtbare Bildschirmseite austauschen und nicht erst die aktuelle Grafik löschen und danach die gewünschte mühevoll berechnen.

Installation

Die Rechner CPC 464 und CPC 664 beinhalten je vier RAM-Banks (0 bis 3). Eine Bank ist ein 16 KByte umfassender Speicherbereich, der mit der Größe des Bildschirmspeichers identisch ist. Aus diesem Grund ist auch Bank 3 (&C000 bis &FFFF) als Bildschirmspeicher reserviert.

Theoretisch kann nun Bank 0, 1 oder 2 zu unserer alternativen Bildschirmseite definiert werden. In der Praxis ist aber in nahezu allen Fällen nur Bank 1 (&4000 bis &7FFF) sinnvoll, da die übrigen Banks Teile des Betriebssystem enthalten und bei deren Überschreibung das einwandfreie Funktionieren des Betriebssystems nicht mehr gewährleistet währe.

Zugriff

Das Kopieren der aktuellen (sichtbaren) Bildschirmseite in die alternative (unsichtbare) und umgekehrt ist in Z80-Maschinensprache extrem einfach, da wir dabei den Blocktransfer-Befehl "LDIR" verwenden können. Die beiden nachstehenden Programme erledigen diese Aufgabe für uns:

```
; ##############################################
; ##                                          ##
; ##              COPY 1 TO 3                 ##
; ## DIESES PROGRAMM KOPIERT BANK 1 NACH BANK 3 ##
; ##                                          ##
; ##############################################
;
;
```

```
START:  LD HL,&4000          ;HL = ERSTES BYTE VON BANK 1
        LD DE,&C000          ;DE = ERSTES BYTE VON BANK 3
        LD BC,&4000          ;KOPIERE DIE GESAMTE BANK
;
;       LDIR                 ;UND ZWAR JETZT

;       RET                  ;FERTIG!
```

```
; ################################################
; ##                                            ##
; ##                COPY 3 TO 1                 ##
; ## DIESES PROGRAMM KOPIERT BANK 3 NACH BANK 1 ##
; ##                                            ##
; ################################################
;
;
START:  LD HL,&C000          ;HL = ERSTES BYTE VON BANK 3
        LD DE,&4000          ;DE = ERSTES BYTE VON BANK 1
        LD BC,&4000          ;KOPIERE DIE GESAMTE BANK
;
;       LDIR                 ;UND ZWAR JETZT

;       RET                  ;FERTIG!
```

Es gibt weiterhin drei Möglichkeiten, um Bank 1 gegen Bank 3 zu vertauschen - eine rein softwaremäßige, eine hardwaremäßige und eine Kombination aus beiden (Jump-Vektor). Jede dieser Lösungen hat Vorteile und kann bei einem optimalen Einsatz nicht - oder nur schwer - durch eine andere ersetzt werden.

Das softwaremäßige Verfahren, das wie Ihnen an dieser Stelle vorstellen, tauscht alle Bytes der beiden Banks nicht nur optisch, sondern real aus. Das bedeutet, daß sich nach Ablauf der folgenden Routine der vollständige Informationsgehalt beider Banks verschoben hat.


```
; ################################################
; ##                                            ##
; ##                SWITCH 1/3                  ##
; ##  DIESES PROGRAMM VERTAUSCHT ALLE BYTES     ##
; ##          VON BANK 1 UND BANK 2             ##
; ##                                            ##
; ################################################
;
;
START:  LD HL,&C000          ;HL = ERSTES BYTE VON BANK 3
        LD DE,&4000          ;DE = ERSTES BYTE VON BANK 1
;
SWITCH: LD A,(HL)            ;HOLE BYTE AUS BANK 3
        LD B,A               ;UND ZWISCHENSPEICHERN
        LD A,(DE)            ;HOLE BYTE AUS BANK 1
        LD (HL),A            ;UND TRANSFERIERE ES NACH BANK 3
        LD A,B               ;HOLE BYTE AUS ZWISCHENSPEICHER
        LD (DE),A            ;UND TRANSFERIERE ES NACH BANK 1
;
        INC HL               ;HOLE NAECHSTE BYTES
        INC DE               ;      "
;
        LD A,H               ;ALLE BYTES VERTAUSCHT?
        OR L                 ;      "
        JR NZ,SWITCH         ;      "
;
;       RET                  ;FERTIG!
```

Die hardwaremäßige Lösung wird mit einer direkten Manipulation am Video-Controller realisiert. Zu diesem Zweck muß in Register 12 des HD 6845 ein Wert abgelegt werden, der bestimmt, ab welcher Adresse der 16-KByte-Bereich des Bildschirmspeichers beginnen soll. Mit der Z80-Sequenz

```
LD   BC,&BC00
LD   A,12
OUT  (C),A
ADD  A,4
```

```
LD   BC,&BD00
OUT  (C),A
RET
```

oder der BASIC-Anweisung

```
OUT &BC00,12:OUT &BD00,16
```

auf die alternative Bildschirmseite (&4000) und mit

```
LD   BC,&BC00
LD   A,12
OUT  (C),A
LD   A,52
LD   BC,&BD00
OUT  (C),A
RET
```

oder der BASIC-Anweisung

```
OUT &BC00,12:OUT &BD00,52
```

wird zurück auf die normale Bildschirmseite (&C000) gewechselt. Dabei ist zu beachten, daß nicht die Bytes beider Bildschirmseiten vertauscht, sondern der Adreßbereich des Video-Controllers manipuliert wird.

Die letzte Alternative ist, eine über den Vektor &BC06 zu erreichende Betriebssystemroutine anzuspringen. Diese Routine manipuliert das Register 12 des Video-Controllers in genau der gleichen Weise, wie wir oben gesehen haben. Zusätzlich werden alle Betriebssystemroutinen, die für die Grafikerstelleung bzw.

deren Berechnung verantwortlich sind, so initialisiert, daß sie von nun an die grafische Ausgabe auf die als Bildschirmbereich definierte Bank umleiten.

Vor Ansprung des Jump-Vektors &BC06 muß das High-Byte der gewünschten Bank (&00, &40, &80 oder &C0) in den Akkumulator übergeben werden. Beachten Sie, daß bei der Übergabe anderer Parameter unerwünschte Nebeneffekte auftreten können.

9.7.3 Scrolling

Srolling - das Verschieben des Bildschirminhalts - ist eine beliebte Technik, die insbesondere in der Spieleprogrammierung Verwendung findet. Die Entwicklung derartiger Scroll-Routinen ist jedoch oft mit sehr großem Aufwand verbunden und können nicht immer von reinen BASIC-Programmierern realisiert werden. Das Betriebssystem des Schneider-CPC bietet glücklicherweise eine Fülle fertiger Scroll-Routinen, die ohne allzu großen Aufwand in BASIC - und selbstverständlich in Maschinenprogramme - eingebunden werden können. Doch bevor wir uns näher mit diesen ROM-Routinen befassen, möchten wir eine Technik vorstellen, die es ermöglicht, den gesamten Bildschirminhalt inklusive Rahmen in kürzester Zeit zu verschieben.

Scrolling mittels H- und VSync-Impulsen

Der Video-Controller des Schneider-CPC enthält zwei Register, die den Zeitpunkt des horizontalen bzw. vertikalen Syncron-Impulses bestimmen. Wird dieser vom Benutzer manipuliert, kann das ganze Bild auf seiner horizontalen oder vertikalen Ebene verschoben werden.

Für die horizontale Verschiebung ist das Register zwei, für die vertikale das Register sieben verantwortlich. Der Default-Wert für Register zwei ist 46, der des Register sieben 30, d.h. daß nach Übergabe dieser Werte der Bildschirm in seiner normalen

Form erscheint. Höhere oder niedrigere Werte verschieben das Bild nach links oder rechts bzw. nach oben oder unten.

Die beiden folgenden Programme zeigen eine praktische Anwendung dieser Scroll-Methode. Dabei ist bemerkenswert, daß das BASIC-Listing ohne Zuhilfenahme von Maschinensprache zu realisieren war.

Das Assemblerlisting:

```
100 ; ##################
110 ; ##              ##
120 ; ## SCROLL-DEMO  ##
130 ; ## 16.10.86/TAV ##
140 ; ##              ##
150 ; ##################
160 ;
170 ;
180         ORG  &A000
190 ;
200 START:  LD   HL,&0207   ;PARAMETER FUER VIDEO-CONTROLLER
210         LD   B,0        ;          "
220 ;
230 BEGIN:  LD   A,43       ;X-STARTWERT
240 M1:     CALL HMOVE      ;HORIZONTALE BEWEGUNG
250         INC  A          ;NAECHSTER VERSCHIEBEWERT
260         CP   50         ;ENDE?
270         JR   NZ,M1      ; "
280 ;
```



```
290         LD   A,27       ;Y-STARTWERT
300 M2:     CALL VMOVE      ;VERTIKALE BEWEGUNG
310         INC  A          ;NAECHSTER VERSCHIEBEWERT
320         CP   34         ;ENDE?
330         JR   NZ,M2      ; "
340 ;
350         LD   A,49       ;X-ENDWERT
360 M3:     CALL HMOVE      ;HORIZONTALE BEWEGUNG
370         DEC  A          ;NAECHTER VERSCHIEBEWERT
380         CP   42         ;ENDE?
390         JR   NZ,M3      ; "
400 ;
410         LD   A,33       ;Y-ENDWERT
420 M4:     CALL VMOVE      ;VERTIKALE BEWEGUNG
430         DEC  A          ;NAECHSTER VERSCHIEBEWERT
440         CP   26         ;ENDE?
450         JR   NZ,M4      ; "
460
470         JR   BEGIN      ;WIEDERHOLUNG
480 ;
490 HMOVE:  LD   B,&BC      ;BC = ADRESS-REGISTER
500         OUT  (C),H      ;ADRESS-REGISTER MIT 2 LADEN
510         JR   PSHDAT     ;PARAMETER AUSGEBEN
520 ;
530 VMOVE:  LD   B,&BC      ;BC = ADRESS-REGISTER
540         OUT  (C),L      ;ADRESS-REGISTER MIT 7 LADEN
550 ;
560 PSHDAT: CALL &BD19      ;WARTE AUF STRAHLENRUECKLAUF
570         LD   B,&BD      ;BC = DATEN-REGISTER
580         OUT  (C),A      ;DATEN-REGISTER MIT PARAMETER LADEN
590         RET             ;FERTIG!
600 ;
610         END             ;PROGRAMMENDE
```

Und nun das Listing für unsere BASIC-Freunde:

```
100 FOR I=43 TO 49:GOSUB 150:NEXT
110 FOR I=27 TO 33:GOSUB 160:NEXT
120 FOR I=49 TO 43 STEP-1:GOSUB 150:NEXT
130 FOR I=33 TO 27 STEP-1:GOSUB 160:NEXT
140 GOTO 100
150 CALL &BD19:OUT &BC00,2:OUT &BD00,I:RETURN
160 CALL &BD19:OUT &BC00,7:OUT &BD00,I:RETURN
170 END
```

Vertikales Scrollen

Hierfür stellt uns das Schneider Betriebssystem eine Routine zur Verfügung, die mit dem Jump-Vektor &BC4D angesprungen werden kann. Die Routine ermöglicht es den kompletten Bildschirm eine Zeile nach oben oder unten zu verschieben. Die Farbe der neu eingefügten Zeile (PAPER) ist dabei frei definierbar.

Die Scroll-Routine erwartet im B-Register einen Verschiebeparameter, der die Verschieberichtung festlegt. Enthält B den Wert Null, wird nach unten gescrollt. Jeder andere Wert bewegt den Bildschirm aufwärts. Der Akkumulator muß den Farbwert der neuen Zeile beinhalten. Dieser muß im Rahmen des aktivierten MODEs definiert werden.

Horizontales Scrolling

Eine Routine, die das horizontale Scrolling ermöglicht, wird über den Jump-Vektor &BC05 angesprungen. Mit Ihr kann der komplette Bildschirm eine Zeile nach links verschoben werden, wobei die aus dem Bildschirm geschobene Zeile rechts angefügt wird.

Die horizontale Verschieberoutine erwartet im HL-Register einen Verschiebeparameter, daß immer duch zwei teilbar sein

muß. Das bedeutet, der sich das relative Bit Null des L-Registers im Low-Zustand befinden muß.

Scrolling beliebiger Bildteile

Eine Betriebssystem-Routine, die beliebige Bildteile verschiebt, soll als letztes vorgestellt werden. Über den Jump-Vektor &BC50 kann sie angesprungen werden und erwartet in den Z80-Registern folgende Parameter:

A: Farbwert
B: Verschieberichtung
H: linke Spalte des Bildschirmbereichs
L: oberste Reihe des Bildschirmbereichs
D: rechte Spalte des Bildschirmbereichs
E: unterste Reihe des Bildschirmbereichs

Die Bildschirmbegrenzungen beziehen sich - wie gewohnt - auf die linke obere Ecke des Bildschirms. Ihr niedrigster Wert ist jedoch im Gegensatz zu BASIC nicht eins, sondern Null.

10. GSX - die grafische Systemerweiterung

10.1 Was ist GSX?

Man liest das Wort in jeder Fachzeitschrift; alle JOYCE-Besitzer kennen es; jeder möchte es in seinen eigenen Programmen verwenden, doch kaum jemand hat dazu die nötigen Hintergrundinformationen. Gemeint ist GSX - das grafische Zauberwort für die Schneider-Computer JOYCE und CPC 6128.

Aber was genau ist GSX? Nun - GSX ist die Abkürzung für Graphics System Extension und bedeutet soviel wie *grafische Systemerweiterung*. Sie wurde von der Firma DIGITAL RESEARCH entwickelt und repräsentiert unter CP/M Plus ein völlig systemunabhängiges Grafik-Ein-/Ausgabesystem. Software, die unter GSX entwickelt wird, ist ungeheuer vielseitig, da sie unter anderem

- auf allen CPM-Plus-Rechnern mit GSX-Erweiterung lauffähig
- und nicht auf ein bestimmtes Ausgabegerät fixiert ist.

Anmerkung: Es gibt Grafik-Kommandos, die sich nicht in gleicherweise auf alle Ausgabegeräte auswirken können. Ein Histogramm kann beispielsweise nur monochrom auf dem Bildschirm, jedoch farbig auf einem Plotter angezeigt werden. Solche hardwarebedingten Differenzen rufen keine Fehlermeldungen hervor. So kann die grafische Information in ungünstigen Fällen zwar verfälscht, aber niemals verloren gehen.

Diese nahezu uneingeschränkte Portabilität eröffnet uns eine Programmflexibilität, die kaum meßbar ist. So ist es auch kein Wunder, daß sich GSX mit der Zeit zur Standard-Grafikerweiterung für Mikrocomputer entwickelt hat.

Die Schneider-Betriebsanleitung gibt leider kaum über GSX und die damit verbundenen Grafikmöglichkeiten Ihres JOYCE oder

CPC-6128 Auskunft. Die meisten 6128-Besitzer wird dieser Umstand warscheinlich wenig stören, weil das Amstrad-BASIC über die vielfältigsten Grafikfunktionen verfügt. Dem JOYCE-Besitzer geht es in dieser Hinsicht sicherlich anders. In vielen Fachzeitschriften wird die außerordentliche Grafikauflösung seines Rechners gelobt - gesehen hat er sie nie. Geht es Ihnen auch so? Dann kann ich Sie beruhigen, denn GSX kann sowohl in eigene Maschinenspracheprogramme, als auch ins Mallard-80-BASIC eingebunden werden. Wie? Nun - GSX ist, ähnlich wie CP/M, modular aufgebaut. Es beinhaltet beispielsweise die Programm-Module GDOS (Graphic Device Operating System) und GIOS (Graphic Input Output System). Letzteres entspricht einem oder mehrerer Devicetreiber[1] für Video-Display, Matrixdrucker oder Plotter.

GDOS ist hingegen für die Entschlüsselung und Weiterleitung von GSX-Kommandos an die Device-Teiber verantwortlich. Die drei wichtigsten Aufgaben sind:

- die Organisation der GSX-Aufrufe
- das Laden der Devicetreiber
- die treibergerechte-Konvertierung aller Parameter

Durch die interne Struktur von CP/M und den darauf abgestimmten modularen Aufbau von GSX ist es möglich, viele unter CP/M Plus geschriebende Programme um GSX zu erweitern. Die Umkonfigurierung bestehender Software ist denkbar einfach, da sich für diese Aufgabe ein spezielles Programm auf einer der mitgelieferten Systemdisketten befindet. Wie das genau funktioniert, werden wir etwas später kennenlernen.

An dieser Stelle möchte ich Ihnen alle mitgelieferten GSX-Systemdateien vorstellen. Die finden sie auf den Seiten zwei, drei und vier Ihrer Systemdisketten.

1 Geräte-Steuerprogramm, das zum Betreiben eines Ausgabegerätes unbedingt erforderlich ist.

Sowohl JOYCE als auch CPC-6128 benötigen die Programme:

GSX.SYS	Dieses Programm enthält GDOS - die eigentliche Grafikerweiterung.
ASSIGN.SYS	beinhaltet bis zu 5 Devicetreiber-Namen und ihre logischen Devicenummern. Diese Datei ist im ASCII-Format abgespeichert und kann somit von Ihnen einfach manipuliert werden.
GENGRAF.COM	Ein Programm, das bestehende Programme so umkonfiguriert, daß GSX automatisch mit ihnen geladen werden kann.
DDHP7470.PRL	Devicetreiber, der es ermöglicht, einen HP-kompatiblen Plotter an ihren JOYCE anzuschließen.

Folgende GSX-Module finden nur auf dem JOYCE Verwendung:

DDSCREEN.PRL	Devicetreiber, der für die Ansteuerung des Displays verantwortlich ist.
DDFXLR8.PRL	Low-Resolution-Devicetreiber für den mitgelieferten JOYCE-Matrixdrucker.
DDFXHR8.PRL	High-Resolution-Devicetreiber für den mitgelieferten JOYCE-Matrixdrucker.

Die folgenden Programme befinden sich nur auf dem CPC-6128:

DRIVERS.GSX	Eine "Read me" File, dessen Inhalt mittels dem TYPE-Befehl angezeigt werden kann und alle Namen der mitgelieferten Devicetreiber beinhaltet.

DDFXLR7.PRL	Low-Resolution-Devicetreiber für Epson und Epson-kompatible Matrixdrucker.
DD-DMP1.PRL	Devicetreiber für den Amstrad DMP-1-Drucker.
DDSHINWA.PRL	Devicetreiber für alle Drucker, die mit einem Shinwa-Mechanismus arbeiten.
DDMODE0.PRL	MODE-0-Devicetreiber für den Bildschirm.
DDMODE1.PRL	MODE 1-Devicetreiber für den Bildschirm.
DDMODE2.PRL	MODE 2-Devicetreiber für den Bildschirm.

10.2 GSX-Installation

Das Installieren von GSX wird durch den eingangs beschiebenen modularen Aufbau stark vereinfacht. Im folgenden Beispiel soll GSX ins Mallard-80-BASIC implementiert werden. Dieses Programm wurde nicht rein zufällig ausgewählt, da jeder JOYCE-Besitzer dieses BASIC besitzt, und die Installation eine Vorraussetzung für GSX-Aufrufe aus der BASIC-Ebene sind.

GSX-Installation in Verbindung mit Mallard BASIC

1. Formattieren Sie eine Leerdiskette.
2. Fertigen Sie darauf mittels PIP eine Kopie von Mallard-BASIC an.
3. Kopieren Sie dazu die Programme GSX.SYS, ASSIGN.SYS, GENGRAF.COM und mindestens einen Devicetreiber. (Device-Teiber sind durch die Extension .PRL kenntlich gemacht.)

4. Geben Sie nun über die Tastatur den Befehl GENGRAF BASIC ein. Dieser Befehl implementiert GSX automatisch in das hinter GENGRAF stehende Programm (in diesem Fall BASIC). Nun kann das Programm GENGRAF.COM von Ihrer Kopie gelöscht werden, da es nur für die GSX-Implementierung, und nicht für den eigentlichen Programmablauf Verwendung findet.

Alle von ihnen verwendeten Devicetreiber müssen nun in der Systemdatei ASSIGN.SYS angemeldet werden. Das Modifizieren dieser Datei ist relativ einfach, da sie im ASCII-Format abgespeichert ist. Sie können sie beispielsweise mit dem mitgelieferten Programm RPED editieren. Bereits angemeldete Treiber zeigt die Befehlsfolge

```
TYPE ASSIGN.SYS
```

direkt auf dem Bildschirm an. Die Datei ASSIGN.SYS wurde von Schneider in sinnvoller Weise vordefiniert. Sie beinhaltet bereits alle mitgelieferten Devicetreiber. Das voreingestellte Format sieht folgendermaßen aus:

```
21 $:DDFXHR8
22 $:DDFXLR8
11 $:DDHP7470
01 $ DDSCREEN
```

Am Anfang jeder Zeile erkennen wir eine zweistellige logische Devicenummer. Durch sie kann der ihr zugewiesende Treiber vom Anwenderprogramm aus aktiviert werden. Üblicherweise werden die Devicenummern

01 bis 10 für Displays
11 bis 20 für Plotter
und 21 bis 30 für Drucker

verwendet. Aus Gründen der Kompatibilität ist es sinnvoll, sich an diese Regelung zu halten. Ist nur ein Gerät einer Kategorie angemeldet, wählt man im allgemeinen den niedrigsten dafür vorgesehenden Wertebereich. Hinter der Devicenummer folgt der Name des Laufwerks (mit anschließendem Doppelpunkt), auf das sich der zu ladende Devicetreiber befindet. Anstelle eines Laufwerksnamen kann auch das Paragraphenzeichen "§" verwendet werden. Es veranlaßt GDOS, den betreffenden Devicetreiber auf allen unter CP/M angemeldeten Laufwerken zu suchen und - falls gefunden - zu laden.

Bitte beachten Sie bei einer eventuellen Modifikation, daß die Devicetreiber nach absteigender Dateigröße in ASSIGN.SYS eingeordnet werden müssen. Die Größe Ihrer Dateien können mit dem CP/M-Plus-Befehl

```
dir d:filename.prl ÄsizeÜ
```

ermittelt werden. Eine fertig erstellte ASSIGN.SYS-Datei kann maximal fünf Devicetreiber enthalten.

So - damit wäre die GSX-Installation beendet. Kehren Sie zu CP/M zurück und starten Sie unser neues, GSX taugliches BASIC, indem Sie wie gewohnt

```
A>BASIC
```

über das Keybord Ihres JOYCE eingeben, und die Eingabe mit der <RETURN>-Taste bestätigen. Falls Sie alles richtig gemacht haben, präsentiert sich Mallard BASIC ab jetzt mit der Meldung:


```
------------------------------------------------
GSX-80 1.1  01 Oct 83    Serial No 5000-1232-654321
Copyright (C) 1983
Digital Research, Inc.          All rights reserved
------------------------------------------------

Mallard-80  BASIC with Jetsam   Version 1.29
(c) Copyright 1984  Locomotive Software Ltd
All rights reserved

17255 free bytes

Ok
```

Vielleicht ist Ihnen aufgefallen, daß uns nach der GSX-Implementierung statt der üblichen 31597 nur noch 17255[1] Byte für unsere BASIC-Programme zur Verfügung stehen. Was ist passiert? GENGRAF.COM hat eine temporäre Datei mit einem kleinen GSX-Lader und BASIC.COM generiert. Danach wurde Mallard-BASIC gelöscht und temporäre Datei in BASIC.COM umbenannt.

Wenn Sie von nun an BASIC.COM aufrufen, wird zunächst der GSX-Lader in den Speicher transferiert und gestartet. Dieser lädt den ersten in ASSIGN.SYS spezifizierten Devicetreiber in den oberen RAM-Bereich. Weiterhin schiebt er das Programm GSX.SYS unmittelbar vor den Treiber und installiert somit GDOS. Dabei wird die BDOS-Funktionsadresse &0005 (Hex) in der Zeropage derart verbogen, daß sich alle Systemaufrufe zwangsläufig über GDOS verzweigen müssen. Dort wird automatisch bei jeden Funktionsaufruf entschieden, ob es sich um einen GSX- oder BDOS-Befehl handelt. Falls letzteres der Fall sein sollte, verzweigt GSX ohne Umwege ins normale BDOS. Zum Schluß transferiert der GSX-Lader BASIC.COM in den normalen Speicherbereich (&0100 HEX).

1 Dieser Wert bezieht sich auf die originale ASSIGN.SYS Datei

Abb. 25: GSX-Speicherorganisation

Wie wir an Abbildung 25 deutlich erkennen können, handelt es sich bei GSX.SYS um ein *relokatibeles* Programm-Modul, d.h., daß es nicht in einen fest definierten RAM-Bereich geladen werden muß. Da GSX - wie alle kommerziellen CP/M-Programme - in 8080-Maschinensprache geschrieben ist, und somit keine relativen Jumps erlaubt, wird hier mit einem kleinen Trick gearbeitet. Im GSX-Lader existiert nämlich eine kleine Routine, die den 8080-Code so manipuliert, daß GSX.SYS in jeden beliebigen Speicherbereich, dessen Low-Byte der Startadresse &00 ist, geschoben werden kann. Zu diesem Zweck wird ein entsprechender Offset zu den High-Bytes der Zieladressen aller Lade-, Jump- und Call-Befehle addiert. Die Low-Bytes bleiben von dieser Methode unberührt. Die so erreichte Relokatibelität ist sehr wichtig, da die Länge des direkt auf GSX.SYS folgenden Devicetreibers nicht immer gleich groß sein muß.

Übrigens kann GSX nicht mehrere Devicetreiber gleichzeitig verwalten. Falls mehrere Ein-/Ausgabegeräte gesteuert werden sollen, muß die Software den jeweils benötigte Treiber während des Programmablaufs nachladen. Da in der Initialisierungsphase der erste in ASSIGN.SYS definierte Devicetreiber in den Speicher gebracht wird, reserviert GSX folglich einen an diesem Treiber orientierten RAM-Bereich. Diese Tatsache erklärt, warum die in ASSIGN.SYS befindlichen Treibernamen nach absteigender Speichergröße sortiert werden müssen. Es liegt auf der Hand, daß beim Nachladen eines Treibers, der größer ist als der für ihn reservierte Speicherplatz, ein Teil des Systembereichs überschreiben und somit zerstört wird. Ein solches Phänomen hätte denn mit größter Warscheinlichkeit einen unweigerlichen Programmabsturz zur Folge.

10.3 Parameterübergabe

Alle GSX-Routinen werden über die geänderte Zeropage-Adresse &0005 (Hex) aufgerufen. In diesem Zusammenhang erwartet GDOS immer das Steuerwort 115 (&73 Hex) im C-Register, weil alle anderen Werte als BDOS-Kommandos abgearbeitet werden. Im DE-Register muß hingegen der 16-Bit-Start-

wert eines 10-Byte langen Datenblocks enthalten sein. Diese Daten repräsentieren seinerseits wieder fünf Startadressen der eigentlich von GSX benötigten Parameterblöcke.

Diese auf den ersten Blick etwas komplizierten Zusammenhänge, sollen die folgende Tabelle verdeutlichen:

GSX-CALL	&0005
C	GDOS-Steuerwort (&73)
DE	Startadresse eines 10-Byte Datenblock
(DE)+&00	Startadresse des CONTROL-Parameterblocks
(DE)+&02	Startadresse des SETPAR-Parameterblocks
(DE)+&04	Startadresse des SETPOS-Parameterblocks
(DE)+&06	Startadresse des GETPAR-Parameterblocks
(DE)+&08	Startadresse des GETPOS-Parameterblocks

Wenn wir die Tabelle näher betrachten, erkennen wir getrennte Ein- und Ausgaberegister einer Variablengattung. Für die Parameterübergabe finden nur die Eingaberegister Verwendung. Die Ausgaberegister werden von den GSX-Routinen beeinflußt. Dessen Werte können nach einem GSX-Aufruf in weitere Berechnungen einbezogen werden.

Der CONTROL-Parameterblock ist für die interne Verwaltung aller GSX-Routinen verantwortlich. CONTROL 0, 1 und 3 verlangen Parameter, die von Ihnen definiert werden müssen. CONTROL 2, 4 unf 5 liefern hingegen nach einem GSX-Aufruf wichtige Informationen.

CONTROL(0)	enthält den Funktionscode, mit der die aufzurufende GSX-Routine selektiert wird.
CONTROL(1)	enthält die Anzahl der Punkte, denen Koordinaten zugewiesen wurden.

CONTROL(2)	wird von GSX beeinflußt und enthält nach dem Aufruf einer Grafikroutine die Anzahl der Parameter, die in GETPAR an den Benutzer übermittelt werden.
CONTROL(3)	enthält die Anzahl der Parameter, die in SETPAR an GSX übergeben werden sollen.
CONTROL(4)	wird von GSX beeinflußt und enthält nach dem Aufruf einer Grafikroutine die Anzahl der Punkte, dessen Koordinaten in GETPOS zurückgemeldet werden.
CONTROL(5)	wird von GSX beeinflußt und enthält einen von Routine zu Routine unterschiedlichen Wert.

Die SETPAR und GETPAR Register enthalten Informationen, die bestimmen, wie die gewählte GSX-Funktion reagieren soll und reagiert hat.

SETPOS und GETPOS beziehen sich auf Spezifikation einer Funktion im Koordinatensystem. SETPOS erwartet die Koordinatenübergabe in der Form x1, y1, x2, y2 ... GETPOS gibt wichtige Koordinaten in der selben Reihenfolge an den Benutzer zurück. Da GSX eine universelle Grafikerweiterung für Computer ist, und dem zufolge die Ein-/Ausgabeparameter nicht auf ein bestimmtes oder eine bestimmte Anzahl von Ausgabegeräten fixiert sind, erwartet GSX alle Koordinaten in einem allgemein gültigen Definitionsbereich. Dieser Bereich umfaßt die Werte &0000 bis &7FFF (0 bis 32767). Um eine derartige allgemeine Koordinate zu ermitteln, muß der Programmierer die X- und Y-Position der gewünschten Koordinate mit je einer Konstanten multiplizieren. Diese Konstanten werden ermittelt, indem jeweils 32767 durch die maximale X- und Y-Auflösung des Ausgabegeräts geteilt wird.

Jedes Parameterblock-Element wird in Form einer 2-Byte-Integer-Variable gespeichert. Diese sinnvolle Datenorganisation

erleichtert uns erfreulicher weise die Parameterübergabe in BASIC, da wir die benötigten Parameter herkömmlichen Integer-Variablen zuweisen können.

Anmerkung: Die nun folgende Beschreibung zeigt eine mögliche Parameterübergabe aus der BASIC-Ebene auf. Der reine Z80-Programmierer sollte jedoch diesen Teil nicht überspringen, weil er viele wertvolle Informationen enthält.

In BASIC können Sie neben den normalen numerischen Variablen auch sogenannte *Integer-Variablen* definieren. Das hat den Vorteil, daß das BASIC-Betriebsysytem für eine solche Variable nur zwei Byte im Speicher reservieren muß. Und genau diese Variablen sind für unser Vorhaben wie geschaffen. Sie haben nämlich das Format, das GSX in den Parametertabellen erwartet. Integer-Variablen werden übrigens entweder mit dem Suffix % kenntlich gemacht, oder durch die DEFINT-Anweisung als solche vereinbart.

Die Integer-Variablen sollten in jedem GSX-BASIC-Programm wie folgt definiert werden:

```
DIM CONTROL%(5),SETPAR%(79),SETPOS%(1,74),GETPAR%(44),GETPOS%(1,74)
```

Bitte beachten Sie bei der Definition, daß das BASIC-Kommando OPTION BASE 1 in Zusammenhang mit GSX nicht im Programm auftauchen darf. OPTION BASE 1 manipuliert nämlich den internen Variablen-Indexzähler in der Form, daß sein niedrigster Wert nicht Null, sondern Eins ist.

Aber wie sollen jetzt die Parametertabellen an GSX übergeben werden? Mallard-BASIC stellt uns diesbezüglich zwei verschiedene Befehle zur Verfügung: den USR und den CALL-Befehl. Der USR-Befehl erweist sich auf den zweiten Blick als ungeeignet. Mit seiner Hilfe können zwar problemlos Maschinenprogramme aufgerufen werden; die Parameterübergabe beschränkt sich jedoch leider nur auf ein Datenelement. Bei dem CALL-

Befehl sieht die ganze Sache ganz anders aus. Er ermöglicht uns auf einfachste Weise die Übermittlung mehrerer Parameter die folgendermaßen an die Register der Z80-CPU weitergeleitet werden.

Bei 3 oder weniger Parameter ist die Parameterübergabe wie folgt:

- HL enthält die Adresse von Parameter Eins
- DE enthält die Adresse von Parameter Zwei[1]
- BC enthält die Adresse von Parameter Drei[1]

Da in der Regel mehr als 3 Werte an GSX transferiert werden, ist es programmtechnisch sinnvoller, wenn wir immer annehmen, daß mindestens 4 Parameter zur Übergabe definiert wurden. Die Adressen von 4 oder mehr Parametern werden folgendermaßen an die Register übermittelt:

- HL enthält die Adresse von Parameter Eins
- DE enthält die Adresse von Parameter Zwei
- BC enthält die Adresse eines Datenblocks, der die Adressen der Parameter drei bis n-1 in aufsteigender Reihenfolge beinhaltet.

Aus Kapitel 10.3 ging hervor, daß GSX die Startadressen der benötigten Parameterblöcke in einer Tabelle erwartet, dessen Startadresse sich wiederum im DE-Register befindet. Wenn wir uns die Organisation des CALL-Befehls näher betrachten, stellen wir fest, daß die Adressen des dritten bis n-ten Datenelements in einem derartigen Datenblock verwaltet werden. Statt des DE-Registers zeigt allerdings das BC-Register auf dessen Startadresse.

1 Falls der Parameter vorhanden ist

Um den CALL-Befehl effektiv einsetzen zu können, müssen wir seinen Variablenblock um zwei vorranstehende Dummy-Elemente erweitern. Ein Dummy-Element ist eine Variable, deren Inhalt zwar beliebig sein kann, ihr Vorhandensein jedoch für die korrekte Ausführung der Funktion untentberlich ist. Beim Aufruf von

```
CALL Adresse,dummy,dummy,CONTROL%(5),SETPAR%(79),SETPOS%(1,74),GETPAR%(4
4),GETPOS%(1,74)
```

wird zwangsläufig die Adresse des ersten Parameterblocks in dem vom BC-Register adressierten Datenblock abgelegt. HL und DE enthalten die völlig bedeutungslosen Adressen unserer Dummy-Elemente.

Der BDOS-Funktionsaufruf &0005 darf jedoch aus folgenden Gründen nicht direkt aus der BASIC-Ebene angesprungen werden:

1. Die Adresse der Paramterblock-Startadressen-Tabelle befindet sich nicht im DE-Register.

2. Das GDOS-Steuerwort &73 befindet sich nicht im C-Register.

Es muß folglich zuerst eine kleine Maschinenroutine angesprungen werden, die diese beiden Punkte für uns erledigt. Eine derartige Routine soll Ihnen an dieser Stelle vorgestellt werden:

```
PUSH BC      ;Startadresse nach DE kopieren
POP  DE      ;              "
LD   C,&73   ;GDOS-Steuerwort definieren
JP   &0005   ;GDOS Funktionsaufruf
```


Nun brauchen wir nur noch unsere Maschinenroutine ins Mallard-BASIC zu implementieren. Zu diesem Zweck ist es notwendig, sieben Byte im Speicher Ihres Schneider zu reservieren. Ansonsten kann leicht der Fall eintreten, daß BASIC unsere Maschinenroutine überschreibt. Das Reservierung der benötigten Byte kann auf einfache Weise mit den beiden BASIC-Befehlen MEMORY und HIMEM realisiert werden. HIMEM ermittelt die letzte im oberen RAM-Bereich benutzte Adresse des BASIC-Programmspeichers. MEMORY ist hingegen für die Definition der obersten von BASIC benutzten Adresse zuständig. Da in BASIC fast alle Befehle beliebig kombiniert werden können, werden durch die Sequenz

```
MEMORY HIMEM-7
```

die benötigten sieben Bytes im BASIC-Speicher reserviert; gleichzeitig erhält HIMEM einen neuen Wert. Jetzt können wir gefahrlos unsere Maschinenroutine in den Speicher POKEen.

```
10 ##############
20 ##          ##
30 ## GSX-INIT ##
40 ##          ##
50 ##############
60 '
70 MEMORY HIMEM-7
80 FOR I%=1 to 7
90 READ GSX%
100 POKE HIMEM+I,GSX%
110 NEXT I
120 DATA &C5,&D1,&0E,&73,&C3,&05,&00
130 DIM CONTROL%(5),SETPAR%(79),SETPOS%(1,74),GETPAR%(44),GETPOS%(
1,74)
140 '
150 REM *** Hauptprogramm ***
```

10.4 Der GSX-Befehlssatz

In dem nun folgendem Teil wird nacheinander der gesamte GSX-Befehlssatz vorgestellt. Es handelt sich dabei um 20 Befehle, die neben den eigentlichen Grafikfunktionen auch Routinen für verwaltungstechnische Aufgaben enthalten. Alle Befehle sind dabei zwecks Übersichtlichkeit in vier Kategorien unterteilt:

1. Routinenname und Kurzbeschreibung ihrer Funktion
2. Eingabeparameter
3. Ausgabeparameter
4. Funktionsbeschreibung

GDOS unterscheidet - wie eingangs erwähnt - die einzelnen Befehle am Inhalt des CONTROL(0)-Registers. Es muß vor jedem GSX-Aufruf einen entsprechende Befehlsnummer enthalten. Diese routinenspeziefische Befehlsnummer ist stehts in der Routinenbeschreibung unter der Kategorie *Eingabe-Parameter* zu finden. Bitte beachten Sie, daß sowohl an die Eingabe- als auch an die Ausgabe-Register nur ganzahlige Werte übergeben werden können.

LOADDR
Neuen Devicetreiber laden

Eingabeparameter:

CONTROL(0) 1
CONTROL(1) 0
CONTROL(3) 10

SETPAR(0)	log. Nummer des zu ladenden Devicetreibers
SETPAR(1)	Linienform: 1 = durchgehend 2 = kurze Striche 3 = gepunktet 4 = Strich-Punkt 5 = lange Striche 6 = Strich-Punkt-Punkt (außer DDSCREEN)
SETPAR(2)	Linienfarbe 0 oder 1
SETPAR(3)	Form des Markers: 1 = . 2 = + 3 = * 4 = O 5 = x
SETPAR(5)	wird im Zusammenhang mit LOADDR nicht verwendet
SETPAR(6)	Textfarbe: 0 oder 1
SETPAR(7)	Füll-Modus: 0 = skelettieren (nur Rahmen zeichnen) 1 = alles ausfüllen 2 = mit Füllmuster 3 = mit Schraffur
SETPAR(8)	Schraffur-Index: 1 bis 6 (siehe Treiberspezifikationstabelle)

SETPAR(9) Füllfarbe:
0 oder 1

Ausgabeparameter:

CONTROL(2) enthält nach dem Aufruf der GSX-Routine die Anzahl der Parameter, die in GETPAR an den Benutzer übermittelt werden.
CONTROL(4) enthält nach dem Aufruf einer GSX-Routine die Anzahl der Punkte, dessen Koordinaten in GETPOS übermittelt werden.

GETPAR(0) enthält die horizontale Auflösung im Bereich 0 bis 32767.
GETPAR(1) enthält die vertikale Auflösung im Bereich 0 bis 32767.
GETPAR(2) nicht definiert.
GETPAR(3) enthält die horizontale Ausdehnung eines Pixels in Mikrometern.
GETPAR(4) enthält die vertikale Ausdehnung eines Pixels in Mikrometern.
GETPOS(1,0) die kleinstmöglich darstellbare Zeichengröße des Ausgabegerätes.
GETPOS(0,2) die kleinstmöglich darstellbare Linienstärke des Ausgabegerätes.
GETPOS(1,4) die kleinstmöglich darstellbare Marker-Größe des Ausgabegerätes.

Funktion: LOADDR lädt einen neuen Devicetreiber, der in ASSIGN.SYS definiert sein muß, in den dafür reservierten Speicherbereich. Der alte Devicetreiber wird durch bei diesem Vorgang überschrieben.

Verwandte Befehle: Closedr

CLOSEDR
Aktivierten Devicetreiber schließen

Eingabeparameter:

CONTROL(0) 2
CONTROL(1) 0
CONTROL(3) nicht definiert

Ausgabeparameter:

keine

Funktion: CLOSEDR beendet die Arbeit mit einem Treiber ohne die durch andere Grafik-Kommandos gesetzten Parameter zu zerstören.

Verwandte Befehle: Loaddr

CLG
Grafik löschen

Eingabeparameter:

CONTROL(0) 3
CONTROL(1) 0
CONTROL(3) nicht definiert

Ausgabeparameter:

keine

Funktion: CLG löscht je nach verwendetem Devicetreiber den Bildschirm oder gibt an den Drucker ein Form-Feed aus und zieht somit ein neues Blatt Papier ein.

Verwandte Befehle: keine

OUTPUT
Grafik-Puffer auslesen

Eingabeparameter:

CONTROL(0) 4
CONTROL(1) 0
CONTROL(3) nicht definiert

Ausgabeparameter:

keine

Funktion: Während der Bildschirmtreiber DDSCREEN alle grafischen Vorgänge direkt sichtbar macht, werden alle Grafikinformationen, die an den Drucker ausgegeben sollen, zwischengepuffert. Das Kommando OUTPUT gibt an GSX den Befehl, diesen Zwischenspeicher auszulesen, und das Ergebnis auf dem Drucker auszugeben. Falls der Bildschirmtreiber DDSCREEN aktiviert sein sollte, wird dieses Kommando ignoriert.

Verwandte Befehle: keine

DRAW
Linie(n) zeichnen

Eingabeparameter:

CONTROL(0) 6
CONTROL(1) Anzahl der in SETPOS definierten Eckpunkte
CONTROL(3) nicht definiert

SETPOS(0,0...74) X1 bis Xn
SETPOS(1,0...74) Y1 bis Yn

Ausgabeparameter:

keine

Funktion: DRAW verbindet die in SETPOS definierten Punkte mit einer Linie. Sie zeichnet je nach Parameterdefinition

- einen Punkt (CONTROL(1)=1),
- eine Linie (CONTROL(1)=2)
- oder ein Polygon (CONTROL(1)>2).

Verwandte Befehle: Linestyle, Linecolor

SETMKR
Marker setzen

Eingabeparameter:

CONTROL(0) 7
CONTROL(1) Anzahl der in SETPOS definierten Punkte
CONTROL(3) nicht definiert

SETPOS(0,0...74) X1 bis Xn
SETPOS(1,0...74) Y1 bis Yn

Ausgabeparameter:

keine

Funktion: SETMKR bringt einen oder mehrere Marker (Symbole), dessen Koordinaten in SETPOS definiert werden müssen, auf den Bildschirm oder in den Drucker-Eingabepuffer.

Verwandte Befehle: Mkrstyle, Mkrsize, Mkrcolor

TEXT
Text zum Ausgabegerät schicken

Eingabeparameter:

CONTROL(0) 8
CONTROL(1) 1
CONTROL(3) Anzahl der Zeichen

SETPAR(0...79)	Text
SETPOS(0,0...74)	X1 bis Xn
SETPOS(1,0...74)	Y1 bis Yn

Ausgabeparameter:

keine

Funktion: TEXT sendt die in SETPAR definierten Zeichen zum angegebenen Ausgabegerät. Die Position des ersten Zeichens muß dabei mit SETPOS festgelegt werden. Mit dem TEXT-Befehl können alle 256 Zeichen der Schneider ausgegeben werden. Die Zeichen 0 bis 31 werden in diesem Zusammenhang nicht als Control-Codes, sondern als Sonderzeichen interpretiert. Jeder Devicetreiber kennt einen speziefischen Schreibmodus. Dieser differiert zwischen Bildschirm- und Druckerausgabe. Auf dem Bilschirm wird der Hintergrund mit der gesammte Zeichenmatrix überschieben (AND-Modus). Drucker oder Plotter können hingegen aus technischen Gründen nur die gesetzten Punkte eines Zeichens zu Papier bringen (OR-Modus). Diese Modi können nicht vom Benutzer geändert werden.

Verwandte Befehle: Txtsize, Txtdir, Txtcolor

FILLPOLY
Ausgefülltes Polygon zeichnen

Eingabeparameter:

CONTROL(0)	9
CONTROL(1)	Anzahl der in SETPOS definierten Eckpunkte
CONTROL(3)	nicht definiert

SETPOS(0,0...74)	X1 bis Xn
SETPOS(1,0...74)	Y1 bis Yn

Ausgabeparameter:

keine

Funktion: FILLPOLY verbindet - änlich wie DRAW - die in SETPOS definierten Eckpunkte mit einer Linie. Entgegen der DRAW-Funktion wird jedoch der erste mit dem letzten Punkt mittels einer Linie verbunden, und das so entstandene Gebilde ausgefüllt. GSX ermöglicht leider nicht das Ausfüllen beliebiger Flächen. Da jedoch bis zu 75 Eckpunkte definiert werden können, ist sogar die Darstellung eines ausgefüllten Kreises ohne weiteres möglich.

Verwandte Befehle: Bar, Fillstyle, Fillindex, Fillcolor

BAR
Rechteck zeichnen

Eingabeparameter:

CONTROL(0) 11
CONTROL(1) 2
CONTROL(3) nicht definiert

SETPOS(0,0) X1 Koordinate der linken unteren Ecke
SETPOS(1,0) Y1 Koordinate der linken unteren Ecke
SETPOS(0,1) X2 Koordinate der rechten oberen Ecke
SETPOS(1,1) Y2 Koordinate der rechten oberen Ecke

Ausgabeparameter:

keine

Funktion: Mittels der Funktion BAR kann auf einfache Weise ein Rechteck konstruiert werden, da nur zwei Eck-Koordinaten zur Definition angegeben werden müssen. Die so erzeugten Rechtecke können auf vielfältigste Weise ausgefüllt werden (siehe dazu auch FILLCOLOR, FILLINDEX, FILLSTYLE).

Verwandte Befehle: Fillpoly, Fillstyle, Fillindex, Fillcolor

TXTSIZE
Zeichengröße bestimmen

Eingabeparameter:

CONTROL(0) 12
CONTROL(1) 1
CONTROL(3) nicht definiert

SETPOS(0,0) 0
SETPOS(1,0) Textgröße (1 bis 12)

Ausgabeparameter:

keine

Funktion: Mit TXTSIZE kann die Zeichengröße des auszugebenden Textes bestimmt werden. Wenn der Bildschirm-Treiber DDSCREEN Verwendung findet, wird dieses Kommando ignoriert, da er nur Zeichengröße "1" verarbeiten kann.

Verwandte Befehle: Text, Txtdir, Txtcolor, Mkrsize

TXTDIR
Textrichtung festlegen

Eingabeparameter:

CONTROL(0) 13
CONTROL(1) 0
CONTROL(3) nicht definiert

SETPAR(0) 1 = 0 Grad
2 = 270 Grad
3 = 180 Grad
4 = 90 Grad

Ausgabeparameter:

keine

Funktion: TXTDIR legt die Ausgaberichtung eines Textes fest. Diesbezüglich stehen vier in jeweils 90-Grad-Intervallen verschobene Richtungen zur Verfügung. Sie müssen anhand von Funktionsnummern, die den entsprechenden Rotationswinkel angeben, übermittelt werden. TXTDIR ist nicht im Bildschirmtreiber DDSCREEN implementiert und wird somit während eines diesbezüglichen Aufrufs ignoriert.

Verwandte Befehle: Text, Txtsize, Txtcolor

LINESTYLE
Linienform festlegen

Eingabeparameter:

CONTROL(0) 15
CONTROL(1) 0
CONTROL(3) nicht definiert

SETPAR(0) 1 = durchgehend
2 = kurze Striche
3 = gepunktet
4 = Strich Punkt
5 = lange Striche
6 = Strich-Punkt-Punkt

Ausgabeparameter:

keine

Funktion: LINESTYLE legt das äußere Erscheinungsbild einer Linie fest. Mit diesem Befehl können problemlos Symetrielinien, Schnittlinien oder änliches definiert werden. Die Linienform wird dabei mittels SETPAR(0) festgelegt. Es ist bei der Linienauswahl unbedingt zu beachten, daß der Bildschirmtreiber DDSCREEN keine Strich-Punkt-Punkt-Linie darstellen kann.

Verwandte Befehle: Draw, Linecolor

LINECOLOR
Linienfarbe definieren

Eingabeparameter:

CONTROL(0) 17
CONTROL(1) 0
CONTROL(3) nicht definiert

SETPAR(0) 0 = Punkt rücksetzen
1 = Punkt setzen

Ausgabeparameter:

keine

Funktion: LINECOLOR legt fest, ob eine Linie gezeichnet oder gelöscht werden soll. Falls sich während des GSX-Aufrufs eine 1 in SETPAR(0) befindet, werden zukünftig alle folgenden Linien auf dem Bildschirm in grün, und auf Druckern mit der Farbe des benutzten Farbbands erstellt. Falls der Benutzer hingegen eine 0 übergibt, hat dies zur Folge, daß von nun an mit der Hintergrundfarbe gezeichnet wird. Das wirkt sich zwar auf dem Bildschirm, jedoch in keinster Weise auf Ducker oder Plotter aus.

Verwandte Befehle: Draw, Linestyle

MKRSTYLE
Marker-Form festlegen

Eingabeparameter:

CONTROL(0) 18
CONTROL(1) 0
CONTROL(3) nicht definiert

SETPAR(0) 1 = .
2 = +
3 = *
4 = O
5 = x

Ausgabeparameter:

keine

Funktion: MKRSTYLE erlaubt es, die Form eines Symbols (Marker), das zur Markierung in Diagrammen oder ähnlichem verwendet werden kann, zu bestimmen. Der entsprechende Code der Form muß in SETPAR(0) übergeben werden.

Verwandte Befehle: Setmkr, Mkrsize, Mkrcolor

MKRSIZE
Marker-Größe bestimmen

Eingabeparameter:

CONTROL(0)	19
CONTROL(1)	1
CONTROL(3)	nicht definiert
SETPOS(0,0)	0
SETPOS(1,0)	Marker-Größe (1 bis 12)

Ausgabeparameter:

keine

Funktion: Mit MKRSIZE kann die Zeichengröße der Marker (Symbole) bestimmt werden. Wenn der Bildschirmtreiber DDSCREEN Verwendung findet, wird dieses Kommando ignoriert, da er nur Zeichengröße "1" verarbeiten kann.

Verwandte Befehle: Setmkr, Mkrstyle, Mkrcolor, Txtsize

MKRCOLOR
Marker-Farbe definieren

Eingabeparameter:

CONTROL(0)	20
CONTROL(1)	0
CONTROL(3)	nicht definiert
SETPAR(0)	0 = Punkt rücksetzen 1 = Punkt setzen

Ausgabeparameter:

keine

Funktion: MKRCOLOR legt fest, ob ein Marker (Symbol) gezeichnet oder gelöscht werden soll. Falls sich während des GSX-Aufrufs eine 1 in SETPAR(0) befindet, werden zukünftig alle folgenden Marker auf dem Bildschirm in grün, und auf Druckern mit der Farbe des benutzten Farbbands erstellt. Falls der Benutzer hingegen eine 0 übergibt, hat dies zur Folge, daß von nun an mit der Hintergrundfarbe gezeichnet wird. Das wirkt sich zwar auf dem Bildschirm, jedoch in keinster Weise auf Ducker oder Plotter aus.

Verwandte Befehle: Setmkr, Mkrstyle, Mkrsize

TXTCOLOR
Textfarbe definieren

Eingabeparameter:

CONTROL(0) 22
CONTROL(1) 0
CONTROL(3) nicht definiert

SETPAR(0) 0 = Punkt rücksetzen
1 = Punkt setzen

Ausgabeparameter:

keine

Funktion: TXTCOLOR legt fest, ob ein oder mehrere Zeichen gezeichnet oder gelöscht werden sollen. Falls sich während des GSX-Aufrufs eine 1 in SETPAR(0) befindet, werden zukünftig alle folgenden Zeichen auf dem Bildschirm in grün, und auf Druckern mit der Farbe des benutzten Farbbands erstellt. Falls der Benutzer hingegen eine 0 übergibt, hat dies zur Folge, daß von nun an mit der Hintergrundfarbe gezeichnet wird. Das wirkt sich zwar auf dem Bildschirm, jedoch in keinster Weise auf Dukker oder Plotter aus.

Verwandte Befehle: Text, Txtsize, Txtdir

FILLSTYLE
Füll-Modus festlegen

Eingabeparameter:

CONTROL(0) 23
CONTROL(1) 0
CONTROL(3) nicht definiert

SETPAR(0) 0 = skelettieren (Rahmen zeichnen und Fläche mit der Hintergrundfarbe ausfüllen)
1 = gesammte Fläche ausfüllen
2 = Fläche mit Füllmuster ausfüllen
3 = Fläche mit Schraffur ausfüllen

Ausgabeparameter:

keine

Funktion: Mittels FILLSTYLE wird bestimmt, auf welche Weise geschlossende Flächen ausgefüllt werden sollen. Der entsprechende Code des Füll-Modus muß in SETPAR(0) übergeben werden.

Verwandte Befehle: Fillpoly, Bar, Fillindex, Fillcolor

FILLINDEX
Füllmuster oder Schraffur definieren

Eingabeparameter:

CONTROL(0) 24
CONTROL(1) 0
CONTROL(3) nicht definiert

SETPAR(0) Schraffur- oder Muster-Index: 1 bis 6 (siehe Treiberspezifikationstabelle)

Ausgabeparameter:

keine

Funktion: Mittels der GSX-Routine FILLINDEX können Schraffuren oder Füllmuster definiert werden, die von Funktionen mit automatischer Flächenfüllung verwendet werden. Der Füll-Index, der in SETPAR(0) übergeben werden muß, entspricht entweder einem der 6 vorhandenen Füllmuster oder Schraffuren. Mit der FILLSTYLE-Funktion muß vorher spezifiziert werden, ob sich der Füll-Index auf eine Schraffur oder ein Muster beziehen, bzw. ignoriert werden soll. Die verschiedenen Muster finden Sie in der noch folgenden Treiberspezifikationstabelle. FILLINDEX wirkt sich leider nicht auf den Bildschirmtreiber DDSCREEN auf. In allen Fällen wird dieses Kommando bei der Bildschirmausgabe ignoriert, und die zu füllende Fläche mit der Hintergrundfarbe ausgefüllt.

Verwandte Befehle: Fillpoly, Bar, Fillstyle, Fillcolor

FILLCOLOR
Flächenfarbe definieren

Eingabeparameter:

CONTROL(0) 25
CONTROL(1) 0
CONTROL(3) nicht definiert

SETPAR(0) 0 = Punkt rücksetzen
1 = Punkt setzen

Ausgabeparameter:

keine

Funktion: FILLCOLOR legt fest, ob eine Fläche ausgefüllt oder gelöscht werden soll. Falls sich während des GSX-Aufrufs eine 1 in SETPAR(0) befindet, werden zukünftig alle Flächen auf dem Bildschirm in grün, und auf Druckern mit der Farbe des benutzten Farbbands ausgefüllt. Falls der Benutzer hingegen eine 0 übergibt, hat dies zur Folge, daß von nun an mit der Hintergrundfarbe gefüllt wird. Das wirkt sich zwar auf dem Bildschirm, jedoch in keinster Weise auf Ducker oder Plotter aus.

Verwandte Befehle: Fillpoly, Bar, Fillstyle, Fillindex

10.5 GDOS Intern

In diesem Kapitel soll der Maschinensprache-Programmierer mit den internen Strukturen von GDOS vertraut gemacht werden. Dazu wird an dieser Stelle ein komplett dokumentiertes GDOS-Listing vorgestellt. Der darin verwendete Speicherbereich (Startadresse BE00 Hex) bezieht sich auf folgende, von DIGITAL RESEARCH vordefinierte Standardtreiber-Konfiguration:

- DDFXHR8
- DDFXLR8
- DDHP7070
- DDSCREEN

Die Treiber wurden in Verbindung mit dem Programm BASIC.COM installiert. Falls Sie in Ihren Programmen eine andere Softwarekonfiguration bevorzugen, müssen Sie die sich mit größter Warscheinlichkeit verschobene GSX-Startadresse durch experimentelle Suchprogramme ermitteln.

Wenn Sie nun einen ersten Blick in das GDOS-Listing werfen, werden sie sofort bemerken, daß die Z80-Mnemonics fehlen. Wie schon bei dem in Kapitel 9.6 vorgestellten ROM-Listing konnten auch hier aus urheberrechtlichen Gründen weder Op-Code noch Mnemonics abgedruckt werden. Um Ihnen jedoch die Betrachtung des vollständigen GDOS-Listing (Adressen, Op-Codes, Mnemonics umd Dokumentation) zu ermöglichen, wird an dieser Stelle ein Disassember für den JOYCE abgedruckt. Die 6128-Besitzer können weiterhin den in Kapitel 9.6.1 vorgestellten Disassembler verwenden. Lediglich muß für diese Anwendung das Lesen aus dem ROM-Speicher durch eine einfache PEEK-Funktion ersetzt werden.

Der hier vorgestellte Disassembler übersetzt den Maschinencode in Z80-Mnemonics. Auf das 8080-Äquivalent wurde hier gänzlich verzichtet, weil dessen symbolischer Code wesentlich unverständlicher ist. Des weiteren enthält schließlich Ihr JOYCE oder CPC nicht einen 8080-, sondern einen Z80-Mikroprozessor.

Nach erfolgreicher Disassembliereung wird sich der Z80-Maschinenspracheprofi villeicht über den umständlichen Programmierstil wundern. Dieser uneffektive Code ist nicht etwa mit dem Unvermögen der Programmierer, sondern mit der Tatsache, daß GDOS vollständig in 8080-Maschinensprache programmiert wurde, zu erklären. Dieser Prozessor beinhaltet ja bekanntlich nur knapp ein Zehntel des Z80-Befehlssatzes.

```
50 REM DISASSEMBLER FUER JOYCE
60 REM
70 GOTO 1020
80 PRINT"Z 8 0  -  D I S A S S E M B L E R"
90 PRINT:PRINT:INPUT"STARTADRESSE : &H",A$
100 GOSUB 930:ANFA=A
110 PRINT:INPUT"ENDADRESSE   : &H",A$
120 GOSUB 930:ENDE=A
130 IF ANFA>ENDE THEN 60
140 PC=ANFA
150 ADR=PC
160 PRINT HEX$(ADR,4);"  ";
170 IFLAG=0
180 GOSUB 970
190 GOSUB 310
200 IF IFLAG THEN 620
210 IF (W=&HCF OR W=&HD7 OR W=&HDF OR W=&HEF) AND (LEFT$(PR$,3)="RST") T
HEN PR$=PR$+" /DW:NN"
220 IF INSTR(PR$,"N")<>0 THEN 730
230 IF INSTR(PR$,"E")<>0 THEN 850
240 PO=INSTR(PR$," ")
250 IF PR$="" THEN PR$="???"
260 IF PO=0 THEN PRINT TAB(21);PR$;:GOTO 280
270 PRINT TAB(21);LEFT$(PR$,PO-1);TAB(27);RIGHT$(PR$,LEN(PR$)-PO);
280 PRINT
290 IF PC<=ENDE THEN 150
300 END
```

```
310 REM INTERPRETIEREN
320 IF (W=&HDD OR W=&HFD) AND NOT IFLAG THEN 510
330 IF W=&HED THEN 480
340 IF W=&HCB THEN 420
350 GOSUB 560
360 ON CO1 GOTO 380,400,370
370 PR$=BEF$(W):RETURN
380 IF W=&H76 THEN PR$="HALT":RETURN
390 PR$="LD "+REGTAB$(CO2)+","+REG$:RETURN
400 IF CO2=0 OR CO2=1 OR CO2=3 THEN A$=" A," ELSE A$=" "
410 PR$=ARILOG$(CO2)+A$+REG$:RETURN
420 REM CB
430 GOSUB 970
440 IF IFLAG THEN DIS=W:GOSUB 970
450 GOSUB 560
460 IF CO1=0 THEN PR$=ROTSCHI$(CO2)+" "+REG$ ELSE PR$=BITTI$(CO1)+STR$(C
O2)+","+REG$
470 RETURN
480 REM ED
490 GOSUB 970
500 IF W<&H40 OR W>&HBF THEN PR$="???":RETURN ELSE GOTO 370
510 REM XY
520 IFLAG=-1
530 IF W=&HDD THEN I$="IX" ELSE I$="IY"
540 GOSUB 970
550 GOTO 310
560 REM CODE ZERLEGEN
570 CO1=(W AND 192)/64
580 CO2=(W AND 56)/8
590 CO3=W AND 7
600 REG$=REGTAB$(CO3)
610 RETURN
620 REM INDIZIERT
630 PO=INSTR(PR$,"HL")
640 IF PO=0 THEN PR$="???":GOTO 240
650 IF INSTR(PR$,"(HL)")<>0 THEN 690
660 IF PR$="EX DE,HL" THEN PR$="???":GOTO 240
670 IF PR$="ADD HL,HL" THEN PR$="ADD "+I$+","+I$:GOTO 240
```



```
680 PR$=LEFT$(PR$,PO-1)+I$+RIGHT$(PR$,LEN(PR$)-PO-1):GOTO 210
690 IF LEFT$(PR$,2)="JP" THEN 680
700 IF PC-ADR<3 THEN GOSUB 970:DIS=W
710 IF DIS>127 THEN DIS$=STR$(DIS-256) ELSE DIS$="+"+RIGHT$(STR$(DIS),LE
N(STR$(DIS))-1)
720 I$=I$+DIS$:GOTO 680
730 REM N ERSETZEN
740 PO=INSTR(PR$,"NN")
750 IF PO<>0 THEN 800
760 PO=INSTR(PR$,"N")
770 GOSUB 970
780 PR$=LEFT$(PR$,PO-1)+"&"+HEX$(W,2)+RIGHT$(PR$,LEN(PR$)-PO)
790 GOTO 240
800 GOSUB 970:LB=W
810 GOSUB 970
820 WERT=W*256+LB
830 PR$=LEFT$(PR$,PO-1)+"&"+HEX$(WERT,4)+RIGHT$(PR$,LEN(PR$)-PO-1)
840 GOTO 240
850 REM E ERSETZEN
860 PO=INSTR(PR$,"E")
870 GOSUB 970
880 IF W>127 THEN W=W-256:REM 2ER-KOMP.
890 W=W+2
900 A$="$"+STR$(W)+" >"+"&"+HEX$(PC+W-2,4)
910 PR$=LEFT$(PR$,PO-1)+A$+RIGHT$(PR$,LEN(PR$)-PO)
920 GOTO 240
930 REM UMWANDLUNG HEX -> DEZ
940 IF A$="" THEN A=0:RETURN
950 A=VAL("&H"+A$)
960 RETURN
970 REM BYTE LESEN
980 W=PEEK(PC)
990 PC=PC+1
1000 PRINT HEX$(W,2);" ";
1010 RETURN
1020 REM INIT
1030 DIM REGTAB$(12),ROTSCHI$(8),BITTI$(3),ARILOG$(7),BEF$(255)
1040 FOR I=0 TO 7:READ REGTAB$(I):NEXT
```

```
1050 FOR I=0 TO 7:READ ROTSCHI$(I):NEXT
1060 FOR I=1 TO 3:READ BITTI$(I):NEXT
1070 FOR I=0 TO 7:READ ARILOG$(I):NEXT
1080 FOR I=0 TO &H7F:READ BEF$(I):NEXT
1090 FOR I=&H80 TO &H9F:BEF$(I)="":NEXT
1100 FOR I=&HA0 TO &HFF:READ BEF$(I):NEXT
1110 GOTO 80
1120 REM DATAS
1130 DATA B,C,D,E,H,L,(HL),A
1140 DATA RLC,RRC,RL,RR,SLA,SRA,???,SRL
1150 DATA BIT,RES,SET
1160 DATA ADD,ADC,SUB,SBC,AND,XOR,OR,CP
1170 DATA NOP,"LD BC,NN","LD (BC),A",INC BC,INC B,DEC B,"LD B,N",RLCA
1180 DATA "EX AF,AF'","ADD HL,BC","LD A,(BC)",DEC BC,INC C,DEC C,"LD C,N
",RRCA
1190 DATA DJNZ E,"LD DE,NN","LD (DE),A",INC DE,INC D,DEC D,"LD D,N",RLA
1200 DATA JR E,"ADD HL,DE","LD A,(DE)",DEC DE,INC E,DEC E,"LD E,N",RRA
1210 DATA "JR NZ,E","LD HL,NN","LD (NN),HL",INC HL,INC H,DEC H,"LD H,N",
DAA
1220 DATA "JR Z,E","ADD HL,HL","LD HL,(NN)",DEC HL,INC L,DEC L,"LD L,N",
CPL
1230 DATA "JR NC,E","LD SP,NN","LD (NN),A",INC SP,INC (HL),DEC (HL),"LD
(HL),N",SCF
1240 DATA "JR C,E","ADD HL,SP","LD A,(NN)",DEC SP,INC A,DEC A,"LD A,N",C
CF
1250 DATA "IN B,(C)","OUT (C),B","SBC HL,BC","LD (NN),BC",NEG,RETN,IM 0,
"LD I,A"
1260 DATA "IN C,(C)","OUT (C),C","ADC HL,BC","LD BC,(NN)",,RETI,,"LD R,A
"
1270 DATA "IN D,(C)","OUT (C),D","SBC HL,DE","LD (NN),DE",,,IM 1,"LD A,I
"
1280 DATA "IN E,(C)","OUT (C),E","ADC HL,DE","LD DE,(NN)",,,IM 2,"LD A,R
"
1290 DATA "IN H,(C)","OUT (C),H","SBC HL,HL","LD (NN),HL",,,,RRD
1300 DATA "IN L,(C)","OUT (C),L","ADC HL,HL","LD HL,(NN)",,,,RLD
1310 DATA ,,"SBC HL,SP","LD (NN),SP",,,,
1320 DATA "IN A,(C)","OUT (C),A","ADC HL,SP","LD SP,(NN)",,,,
```



```
1330 DATA LDI,CPI,INI,OUTI,,,,,LDD,CPD,IND,OUTD,,,,
1340 DATA LDIR,CPIR,INIR,OTIR,,,,,LDDR,CPDR,INDR,OTDR,,,,
1350 DATA RET NZ,POP BC,"JP NZ,NN",JP NN,"CALL NZ,NN",PUSH BC,"ADD A,N",
RST &00
1360 DATA RET Z,RET,"JP Z,NN",->,"CALL Z,NN",CALL NN,"ADC A,N",RST &08
1370 DATA RET NC,POP DE,"JP NC,NN","OUT (N),A","CALL NC,NN",PUSH DE,"SUB
 N",RST &10
1380 DATA RET C,EXX,"JP C,NN","IN A,(N)","CALL C,NN",-
>,"SBC A,N",RST &18
1390 DATA RET PO,POP HL,"JP PO,NN","EX (SP),HL","CALL PO,NN",PUSH HL,"AN
D N",RST &20
1400 DATA RET PE,JP (HL),"JP PE,NN","EX DE,HL","CALL PE,NN",-
>,"XOR N",RST &28
1410 DATA RET P,POP AF,"JP P,NN",DI,"CALL P,NN",PUSH AF,"OR N",RST &30
1420 DATA RET M,"LD SP,HL","JP M,NN",EI,"CALL M,NN",->,"CP N",RST &38
```

GDOS 1.0

```
BE00 ------------------------------------------------ GDOS-EINSPRUNG

BE00 ;GDOS EINSPRUNG
BE03 ;BDOS EINSPRUNG
;
BE06 ;DEVICENUMMER DES AKTIVEN TREIBERS


BE08 ------------------------------------------------ DEVICETREIBER 1

BE08 ;DEVICENUMMER (21)
BE0A ;LAUFWERK
BE0B ;DEVICE-NAME
BE12 ;BLANK FÜR 8. STELLE


BE13 ------------------------------------------------ DEVICETREIBER 2

BE13 ;DEVICENUMMER (22)
BE15 ;LAUFWERK
BE16 ;DEVICE-NAME
BE1D ;BLANK FÜR 8. STELLE


BE1E ------------------------------------------------ DEVICETREIBER 3

BE1E ;DEVICENUMMER (11)
BE20 ;LAUFWERK
BE21 ;DEVICE-NAME


BE29 ------------------------------------------------ DEVICETREIBER 4

BE29 ;DEVICENUMMER (1)
BE2B ;LAUFWERK
BE2C ;DEVICE-NAME
```



```
BE34 ------------------------------------------------ DEVICETREIBER 5

BE34 ;ENDE-KENNZEICHEN, DA KEIN FÜNFTER TREIBER ANGEMELDET WURDE
BE36 ;LAUFWERK
BE37 ;RESERVIERTER PLATZ
BE39 ;FÜR DEVICE-NAME
BE3B ;         "
BE3D ;         "
;
BE3F ;ENDE-KENNZEICHEN


BE41 ------------------------------------------------ GSX-MESSAGE

BE41 "-----------------------------"
BE5F "--------------------"
;
BE74 ;ZEILENENDE-KENNUNG
;
BE76 "GSX-80 1.1  01 Oct 83    Seria"
BE94 "l No 5000-1232-654321"
;
BEA9 ;ZEILENENDE-KENNUNG
;
BEAB "Copyright (C) 1983            "
BEC9 "                      "
;
BEDE ;ZEILENENDE-KENNUNG
;
BEE0 "Digital Research, Inc.        "
BEFE "    All rights reserved"
;
BF13 ;ZEILENENDE-KENNUNG
;
BF15 "-----------------------------"
BF33 "--------------------
;
BF48 ;ZEILENENDE-KENNUNG
;
```

```
BF4A  ;EM-BYTE
;
BF4B  ;BUFFER

BF6D  ------------------------------------------------------ GDOS START

BF6D  ;HANDELT ES SICH UM
BF6E  ;EINEN GSX AUFRUF?
BF70  ;WENN NEIN - SPRINGE NACH BDOS
;
BF73  ;HL = BUFFERSTART
BF76  ;HOLE DIE STARTADRESSEN ALLER 5 GSX I/O-VEKTOREN (FÜNF ADRESSEN =
10 BYTES)
BF78  ;TRANSBYT AUFRUFEN
;
BF7B  ;HL = ADRESSE VON CONTROL(0)
BF7E  ;A = LSB[1] VON CONTROL(0)
BF7F  ;ZEIGE AUF MSB[2]
BF80  ;H = MSB (CONTROL(0))
BF81  ;HL = CONTROL(0)
BF82  ;FUNKTIONSNUMMER -1 FÜR VERGLEICH
BF83  ;NEUEN DEVICETREIBER
BF84  ;LADEN?
BF85  ;WENN NEIN FAHRE FORT
;
BF88  ;GETCNT AUFRUFEN
BF8B  ;DEVICETREIBER ANSPRINGEN
BF8E  ;SENDPAR AUFRUFEN


BF91  ---------------------------------------- NEUEN DEVICETREIBER LADEN

BF91  ;HL = ADRESSE VON SETPAR(0)
BF94  ;E = LSB VON SETPAR (0)
BF95  ;ZEIGE AUF MSB
BF96  ;DE = NEUE DEVICENUMMER
BF97  ;HOLE AKTUELLE DEVICENUMMER
```

1 niederwertiges Byte
2 höchstwertiges Byte


```
BF9A  ;CMPDRV AUFRUFEN
BF9D  ;WENN NEUER TREIBER = DEM AKTUELLEN TREIBER, FEHLER AUSGEBEN!
BFA0  ;POINTER AUF DEVICENUMMER VON TREIBER 1
BFA3  ;ADRESSE RETTEN.
BFA4  ;HOLE LSB DER DEVICENUMMER
BFA5  ;HOLE MSB DER
BFA6  ;DEVICENUMMER
BFA7  ;HL = DEVICENUMMER
BFA8  ;GÜLTIGE DEVICE-
BFA9  ;NUMMER?
BFAA  ;STACK IN ORDNUNG BRINGEN
BFAB  ;FEHLER AUSGEBEN?
BFAE  ;RETTE POINTER
BFAF  ;CMPDRV AUFRUFEN
BFB2  ;POINTER AUF DEVICENUMMER
BFB3  ;WENN DEVICETREIBER ANGEMELDET, SPRINGE NACH &BFBD
BFB6  ;OFFSET FÜR NÄCHSTE DEVICENUMMER
BFB9  ;BERECHNE ADRESSE DER NÄCHSTEN DEVICE-NR.
BFBA  ;HOLE NÄCHSTE NUMMER
;
BFBD  ;HOLE LAUFWERKSNUMMER
BFBE  ;       "
BFBF  ;HOLE NEUE DEVICENUMMER
BFC0  ;AKTUELLE DEVICENUMMER = NEUE DEVICENUMMER!
BFC3  ;BUFFER FÜR READFILE
BFC6  ;ÜBERTRAGE 9 BYTES (LAUFWERKSNUMMER UND DEVICETREIBER-NAME)
BFC8  ;TRANSBYTE ANSPRINGEN
BFCB  ;READFILE AUFRUFEN (TREIBER LADEN)
BFCE  ;ERROR AUFRUFEN FEHLER AUFGETRETEN?
BFD1  ;HL = ADRESSE VON GETPAR(0)
BFD4  ;HOLE LSB
BFD5  ;ZEIGE AUF MSB
BFD6  ;UND RETTE ADRESSE
BFD7  ;H = MSB
BFD8  ;HL = GETPAR(0)+1
BFD9  ;     "
BFDA  ;UND MERKEN
BFDD  ;HOLE MSB DER ADRESSE VON GETPAR(0)
BFDE  ;ZEIGE AUF LSB DER GETPAR(1) ADRESSE
BFDF  ;HOLE LSB
```

```
BFE0  ;ZEIGE AUF MSB
BFE1  ;H = MSB
BFE2  ;HL = GETPAR(1)+1
BFE3  ;     "
BFE4  ;UND MERKEN
BFE7  ;SPRUNG AUF JUMP


BFEA  ---------------------------------------------------------- GETCNT

BFEA  ;HL = ADRESSE VON CONTROL(0)
BFED  ;POINTER AUF ADRESSE
BFEE  ;VON CONTROL(1)
BFEF  ;E = LSB VON CONTROL(1)
BFF0  ;ZEIGE AUF MSB
BFF1  ;DE = CONTROL(1)
BFF2  ;WERDEN KOORDINATEN
BFF3  ;ÜBERGEBEN?
BFF4  ;WENN NEIN, ZURÜCK ZUM HAUPTPROGRAMM
BFF5  ;OFFSET FÜR VERGLEICH
BFF8  ;WURDEN MEHR ALS 75
BFF9  ;KOORDIN. ÜBERGEBEN?
BFFC  ;WENN JA, DEZIMIERE IHRE ANZAHL AUF 75
BFFE  ;POINTER AUF ADRESSE VON SETPOS(0)
C001  ;SETZEN UND RETTEN
C002  ;GET-BUFFER EINRICHTEN
C005  ;UND STARTADRESSE MERKEN
C008  ;POINTER NACH BC
C009  ;ÜBERTRAGEN
C00A  ;HOLE ADRESSE VON SETPOS(0)
;
C00B  ;ANZAHL DER KOORDINATEN RETTEN
C00C  ;DE = ADRESSE VON SETPOS(0)
C00D  ;HOLE GETPAR(0)
C010  ;PARCONV AUFRUFEN GETPAR(N) UMRECHNEN
C013  ;DE = ADRESSE VON SETPOS(1)
C014  ;HOLE GETPAR(1)
C017  ;PARCONV AUFRUFRN
C01A  ;HOLE ANZAHL DER KOORDINATEN
```



```
C01B  ;ANZAHL = ANZAHL - 1
C01C  ;NOCH KOORDINATEN VORHANDEN?
;
C01F  ;FERTIG!


C020  ---------------------------------------------------------- PARCONV

C020  ;HOLE ADRESSE VOM SETPOS NACH HL
C021  ;AKKU = LSB VON SETPOS(0)
C022  ;POINTER AUF MSB
C023  ;UND RETTEN
C024  ;H = MSB VON SETPOS(0)
C025  ;RETTE POINTER VON GET-BUFFER
C026  ;HL = SETPOS(0)
C027  ;INITIALISIERE ZÄHLER
C029  ;RETTE GETPAR(0) ODER GETPAR(1)
C02A  ;SETPOS(0) NACH DE
C02B  ;HL INITIALISIEREN
;
;****************** SETPOS(N)/15
;
C02E  ;HOLE MSB VON SETPOS(N)
C02F  ;DIVIDIERE ES DURCH 2
C030  ;UND SCHREIBE ERGEBNIS ZURÜCK NACH D
C031  ;HOLE LSB VON SETPOS(N) UND
C032  ;DIVIDIERE ES DURCH 2
C033  ;DE = SETPOS(N)/2
C034  ;SPRINGE FALLS BEI DIVISION KEIN REST AUFGETRETEN IST
C037  ;RETTE ZÄHLER
C038  ;HOLE GETPAR(0) ODER GETPAR(1)
C039  ;UND ZURÜCK AUF STACK
C03A  ;HL <-- BC
C03B  ;ZÄHLER NACH C ZURÜCKSCHREIBEN
;
C03C  ;HOLE MSB
C03D  ;DIVIDIERE ES DURCH 2
C03E  ;UND SCHREIBE ERGEBNIS ZURÜCK NACH H
C03F  ;HOLE LSB
C040  ;DIVIDIERE ES DURCH 2
```

```
C041  ;UND SCHREIBE ERGEBNIS ZURÜCK NACH H
C042  ;ZÄHLER = ZÄHLER - 1
C043  ;SCHLEIFENENDE?
;
C046  ;HOLE GET-BUFFER
C047  ;       "
C048  ;LSB WEGSCHREIBEN
C049  ;       "
C04A  ;POINTER ERHÖHEN
C04B  ;MSB HOLEN
C04C  ;UND IN DEN GET-BUFFER SCHREIBEN
C04D  ;POINTER ERHÖHEN
C04E  ;HOLE SETPOS(N)
C04F  ;ZEIGE AUF N+1
C050  ;ENDE DER ROUTINE


C051  ---------------------------------------------------- SENDPAR

C051  ;HOLE ADRESSE VON CONTROL(0)
C054  ;ZEIGE AUF ADRESSE VON CONTROL(4)
C057  ;       "
C058  ;HOLE LSB VON CONTROL(4)
C059  ;HOLE MSB
C05A  ;BC = CONTROL(4), DIE ANZAHL DER KOORDINATEN, DIE DEM USER
      ;GESENDET WERDEN
C05B  ;HOLE ADRESSE VON GETPOS(0)
;
C05E  ;SOLLEN KOORDINATEN
C05F  ;GESENDET WERDEN?
C060  ;WENN NEIN, ZURÜCK ZUM HAUPTPROGRAMM
C061  ;RETTE CONTROL(4)
C062  ;DE = ADRESSE VON GETPOS(N,N)
C063  ;HOLE GETPAR(0)
C066  ;PARTRANS AUFRUFEN
C069  ;DE = GETPOS(N+1,N)
C06A  ;HOLE GETPAR(1)
C06D  ;PARTRANS AUFRUFEN
C070  ;HOLE ANZAHL DER ZU SENDENDEN KO-
```



```
C071  ;ORDINATEN UND VERINGERE IHRE ANZAHL UM EINS
C072  ;NÄCHSTE PARAMETER VERARBEITEN


C075  ---------------------------------------------------- PARTRANS

C075  ;BC = GETPAR(0) BZW.
C076  ;GETPAR(1)
C077  ;HOLE ADRESSE VON GETPOS(N,N)
C078  ;E = LSB VON GETPOS(N,N)
C079  :RETTE ADRESSE VON GETPOS(N,N)
C07A  ;HOLE MSB
C07B  ;DE = GETPOS(N)
C07C  ;HL = GETPOS(N)
C07D  ;ZÄHLER INITIALISIEREN
C07F  ;NEW.PRM INITIALISIEREN
C082  ;ZÄHLER RETTEN
C083  ;HOLE NEW.PRM
C084  ;UND MULTIPLIZIERE ES MIT 2
C085  ;HOLE GETPOS(N)
C086  ;GETPOS * 2
C087  ;AKKU = LSB
C088  ;MINUS LSB DES VON GETPAR(0) BZW. GETPAR(1)
C089  ;UND ZURÜCKSCHREIBEN
C08A  ;AKKU = MSB
C08B  ;MINUS CARRY UND MSB VON GETPAR(0) BZW. GETPAR(1)
C08C  ;HL = ERGEBNIS
C08D  ;KLEINER NULL?
C090  ;ADDIERE ZU ERGEBNIS GETPAR(0) BZW. GETPAR(1)
C091  ;NEW.PRM FOR MODIFIZIERUNG SCHÜTZEN (INC DE)
C092  ;NEW.PRM + 1
C093  ;HOLE ZÄHLER
C094  ;UND ERHÖHE IHN
C095  ;ZÄHLER UNGLEICH NULL?
C098  ;LÖSCHE CARRY-FLAG
C099  ;UND HOLE NEW.PRM
C09A  ;NEW.PRM/2
C09B  ;       "
C09C  ;       "
C09D  ;       "
```

```
C09E  ;      "
C09F  ;ÜBERLAUF?
C0A2  ;WENN NEIN ERHÖHEN
C0A3  ;HOLE ADRESSE VON GETPOS(N,N)
C0A4  ;NEW.PRM SPEICHERN
C0A5  ;      "
C0A6  ;      "
C0A7  ;ZEIGE AUF GETPOS(1,N) ODER GETPOS(0,N+1)
C0A8  ;ROUTINENENDE
```

C0A9 ---------- TRANSBYT

TRANSBYT überträgt einen maximal 256 langen Datenblock in einen anderen RAM-Bereich.

Einagbeparameter:
DE = Startadresse des Blocks
HL = Zieladresse
C = Blocklänge in Bytes

Ausgabeparameter:
keine

zerstörte Register: A, C, D, E, F, H, L

```
C0A9  ;HOLE BYTE...
C0AA  ;UND ÜBERTRAGE ES IN DEN BUFFER
C0AB  ;AUF NÄCHTES BYTE IM BUFFER ZEIGEN
C0AC  ;AUF NÄCHTES BYTE ZEIGEN
C0AD  ;ALLE BYTES ÜBERTRAGEN?
C0AE  ;WENN JA ZURÜCK ZUM HAUPTPROGRAMM
C0B1  ;      "
```

C0B2 ---------- CMPDRV

CMPDRV vergleicht die Nummer des aktuell verwendeten Devicetreibers, mit der des neuen Treibers. Die aktuelle Devicenummer muß sich in diesem Zusammenhang in HL, und die neue in DE befinden. Sind beide Nummern identisch, so wird am Ende der Routine das Zero-Flag gesetzt - wenn nicht rückgesetzt.

Einagbeparameter:
DE = neue Devicenummer
HL = aktuelle Devicenummer

Ausgabeparameter:
keine

zerstörte Register: A, F, H, L

```
C0B2  ;AKKU = LSB DER AKTUELLEN DEVICENUMMER
C0B3  ;SUBTRAHIERE LSB DES NEUEN DEVICE TREIBER
C0B4  ;ERGEBNIS SICHERN
C0B5  ;AKKU = MSB DER AKTUELLEN DEVICENUMMER
C0B6  ;SUBTRAHIERE MSB + ÜBERTRAG DES NEUEN DEVICETREIBERS
C0B7  ;ERGEBNIS SICHERN
C0B8  ;NEUER TREIBER = AKTUELLER TREIBER?
C0B9  ;FERTIG!
```

C0BA ---------- SYSTEM BUFFER

```
C0BA  ;CONTROL STARTADRESSE
C0BB  ;      "
C0BC  ;SETPAR STARTADRESSE
C0BD  ;      "
C0BE  ;SETPOS STARTADRESSE
C0BF  ;      "
C0C0  ;GETPAR STARTADRESSE
C0C1  ;      "
C0C2  ;GETPOS STARTADRESSE
C0C3  ;      "
C0C4  ;GETPAR(0) STARTADRESSE
C0C5  ;      "
C0C6  ;GETPAR(1) STARTADRESSE
C0C7  ;      "
```

10.6 GSX Fehlermeldungen

GSX enthält eine ganze Palette spezieller Meldungen, die beim Auftreten eines Fehlers automatisch auf dem Bildschirm ausgegeben werden. Die Ausgabe einer derartigen Fehlermeldung kann einen Programmabbruch anschließendem Rücksprung ins CP/M-Betriebssystem zur Folge haben.

Fehlermeldung	*Erläuterung*
d:filename.PRL not found	"Devicetreiber nicht gefunden": Der in ASSIGN.SYS angemeldete Devicetreiber ist nicht auf dem vordefinierten Laufwerk vorhanden. Der entsprechende Treiber muß auf die Diskette des betrefenden Laufwerks kopiert, oder die Laufwerksspezifikation in ASSIGN .SYS geändert werden.
d:dateiname.PRL empty	"Devicetreiber leer": Der in ASSIGN.SYS angemeldete Devicetreiber wurde gefunden - sein Inhalt ist hingegen fehlerhalt oder nicht vorhanden. Der in der Fehlermeldung aufgeführte Treiber muß erneut auf die betreffende Diskette kopiert werden.
d:filename.PRL contains absolute segment	"Devicetreiber enthält inkompatible Daten": Der betreffende Devicetreiber enthält fehlerhafte Daten. Der in der Fehlermeldung aufgeführte Treiber muß erneut auf die betreffende Diskette kopiert werden.

d:filename.PRL closing error	"Devicetreiber ist nicht geschlossen worden": Die Diskette mit dem in der Fehlermeldung angegebenen Devicetreiber ist ausgewechselt worden. Sie muß wieder in das Laufwerk, das in ASSIGN.SYS angemeldet wurde, eingelegt werden.
d:filename.PRL load error	"Devicetreiber Ladefehler": Die in ASSIGN.SYS angemeldeten Devicetreiber sind nicht in absteigender Reihenfolge eingetragen worden. Folglich muß ASSIGN.SYS neu definiert werden.

10.7 Devicetreiber Spezifikationen

Jeder Devicetreiber hat aus hardwarebedingten Gründen unterschiedliche Eigenschaften. Die Programmierer von DIGITAL RESEARCH haben sich zwar bemüht, die Unterschiede so gering wie möglich zu halten, aber auf einigen Gebieten sind größere Tolleranzen leider unvermeidbar; so kann ein Drucker beispielsweise keine Linien löschen. Auf dem Bildschirm ist hingegen ein Form-Feed beim besten Willen nicht realisierbar. Daraus resultiert, das bestimmte Programme nur mit bestimmten Devicetreibern sinnvoll zusammenarbeiten können.

In vielen Fällen bewährt sich sicherlich der Einsatz des Bildschirmtreibers DDSCREEN in Zusammenarbeit mit einem Druckertreiber. Diese Kombination hat den Vorteil, das die zu bearbeitende Grafik auf dem Bildschirm in der Grundstruktur probemlos erfaßt und manipuliert werden kann. Falls alle Schritte korrekt verlaufen sind, steht dann der Ausgabe über Ihr Peripheriegerät nichts mehr im Wege.

Damit Sie sich über die Möglichkeiten der einzelnen Devicetreiber ein Bild machen können, finden Sie alle diesbezüglichen Informationen in den nun folgenden Treiber-Spezifikationstabellen. Sie sind übrigens nicht nur für den Programmierer interessant, sondern helfen auch dem reinen Anwender beim Softwareerwerb weiter, denn einige GSX-Programme verlangen von dem installierten Devicetreiber ganz bestimmte Eigenschaften.

DDSCREEN.PRL

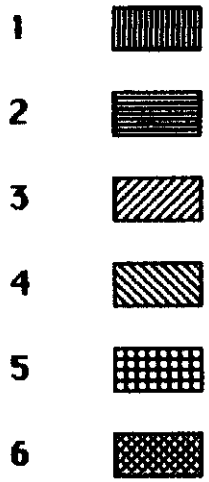
Ausgabegerät:	Bildschirm
Auflösung:	720 x 248 Pixel
Linienform:	1 durchgehend 2 kurze Striche 3 gepunktet 4 Strich-Punkt (Symmetrie-Linie) 5 lange Striche
Strichstärken:	1
Marker (Symbole):	1 . 2 + 3 * 4 o 5 x
Zeichengrößen:	eine (Standardgröße)
Textrichtungen:	eine (0)
Füllmuster:	keine (Fläche wird umrahmt)
Schraffuren:	keine (Fläche wird umrahmt)
Verallgemeinertes Darstellungselement:	Balken-Umriß

Escape-Folgen:	alle, bis auf Grafik-Tablett und Hardcopy
Eingabe:	nein

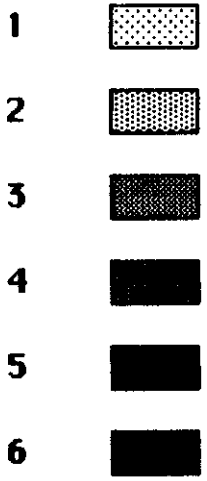
DDFXLR8.PRL

Ausgabegerät:	Drucker (low-resolution)
Auflösung:	480 x 672 Pixel
Linienform:	1 durchgehend 2 kurze Striche 3 gepunktet 4 Strich-Punkt (Symmetrie-Linie) 5 lange Striche 6 Strich-Punkt-Punkt
Strichstärken:	12
Marker (Symbole):	1 . 2 + 3 * 4 o 5 x
Zeichengrößen:	12
Textrichtungen:	0, 90, 180, 270, 360

Füllmuster:



Schraffuren:



Verallgemeinertes Darstellungselement: ausgefüllter Balken

Escape-Folgen: verfügbare Zeichenzellen angeben

Eingabe: nein

DDFXHR8.PRL

Ausgabegerät: Drucker (high-resolution)

Auflösung: 480 x 672 Pixel

Linienform:
1 durchgehend
2 kurze Striche
3 gepunktet
4 Strich-Punkt (Symmetrie-Linie)

5 lange Striche
6 Strich-Punkt-Punkt

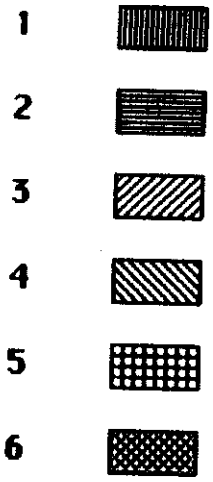
Strichstärken: 12

Marker (Symbole):
1 .
2 +
3 *
4 o
5 x

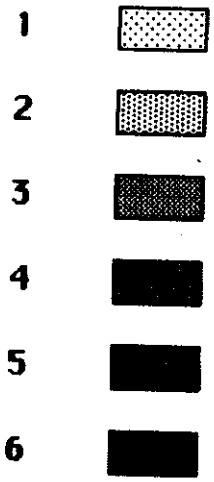
Zeichengrößen: 12

Textrichtungen: 0, 90, 180, 270, 360

Füllmuster:



Schraffuren:



Verallgemeinertes Darstellungselement: ausgefüllter Balken

Escape-Folgen: verfügbare Zeichenzellen angeben

Eingabe: nein

Anhang - Grafikbezogene BASIC-Befehle

BORDER

BORDER Farbe[,Farbe]

Rund um die beschreibbare Fläche auf dem Bildschirm des CPC befindet sich ein Rand (BORDER), auf dem keine Darstellung erfolgen kann. Die Farbe dieses Bildschirmrandes kann mit dem Kommando BORDER verändert werden. Für Farbe kann ein Wert zwischen 0 und 26 eingesetzt werden. Welcher Wert für welche Farbe steht, ist in der folgenden Tabelle aufgeführt.

0 schwarz	14 pastellblau
1 blau	15 orange
2 leuchtend blau	16 rosa
3 rot	17 pastellmagenta
4 magenta	18 leuchtend grün
5 mauve	19 seegrün
6 leuchtend rot	20 leutend blaugrün
7 purpur	21 limonengrün
8 leuchtend magenta	22 pastellgrün
9 grün	23 pastell-blaugrün
10 blaugrün	24 leuchtend gelb
11 himmelblau	25 pastellgelb
12 gelb	26 leuchtend weiß
13 weiß	

Wird zusätzlich zu dem zwingend vorgeschriebenen Parameter ein zweiter Farbwert an das BORDER-Kommando angehängt, so wechselt die Farbe des Bildschirmrands zwischen diesen beiden Farben. Die Geschwindigkeit des Farbwechsels kann mit dem Befehl SPEED INK manipuliert werden.

CLG

CLG [Farbstift]

Das Kommando CLG löscht den Grafikbildschirm. Wurde das CLG-Kommando durchgeführt, so hat der Grafikbildschirm die Farbe angenommen, die dem angegebenen Farbstift zugeordnet ist.

CLS

CLS [#Stream]

Das Kommando CLS (CLear Screen) löscht den gesamten Bildschirm und setzt den Cursor in die linke obere Ecke des Textfensters. Nach einem ausgeführten CLS hat das Textfenster die Farbe des mit PAPER ausgewählten Farbstifts angenommen (vgl. CLG).

Wird zur Spezifikation des CLS-Kommando ein Stream angegeben, so wird das durch den Stream vertretene Fenster nach den oben genannten Vorschriften gelöscht.

COPYCHR$ (nicht auf dem CPC 464)

COPYCHR$(#Stream)

Mit der Funktion COPYCHR$ ist es möglich, ein Zeichen von einer Position des Textbildschirms zu kopieren. Zu diesem Zweck wird der Textcursor auf eine bestimmte Bildschirmposition gebracht, mit COPYCHR$ das Zeichen an dieser Position gelesen und einer Stringvariablen zugewiesen. Der Inhalt dieser Variablen kann dann beliebig weiterverwendet werden. Das folgende Beispielprogramm verdeutlicht den Zusammenhang:


```
10 CLS
20 LOCATE 10,10
30 PRINT "*"
40 LOCATE 10,10
50 ZEICHEN$=COPYCHR$(#0)
60 LOCATE 10,12
70 PRINT ZEICHEN$
80 END
```

Bei dieser BASIC-Funktion muß unbedingt angegeben werden, auf welches Window (Stream) sie sich beziehen soll.

Eine Besonderheit von COPYCHR$ muß noch beachtet werden. Sollte an der Bildschirmposition, von der gelesen werden soll, einmal etwas anderes stehen als das Bitmuster einer Zeichenmatrix, so kann COPYCHR$ natürlich auch kein Zeichen identifizieren. Die Funktion liefert dann einen leeren String.

CURSOR (nicht auf dem CPC 464)

CURSOR [Schalter 1][,Schalter 2]

Das BASIC-Kommando CURSOR dient zum Ein- (1) bzw. Ausschalten (0) der für das Erscheinen des Cursors auf dem Bildschirm zuständigen System- und Benutzerschalter. Der erste Schalter steht für das System, der zweite für den Benutzer. Der Cursor ist auf dem Bildschirm sichtbar, wenn beide Schalter eingeschaltet sind.

DRAW

DRAW X-Position,Y-Position[,[Farbstift][,Modus[1]]]

Mit dem BASIC-Kommando DRAW kann eine Linie gezogen werden. Startpunkt der Linie ist die aktuelle Position des Grafikcursors - Endpunkt, der durch X- und Y-Position bestimmte Punkt. Mit dem optionalen Parameter Farbstift kann bestimmt werden, mit welcher Farbe die Linie gezeichnet wird. Der Parameter kann Werte zwischen 0 und 15 annehmen.

Mit dem Parameter Modus kann festgelegt werden, wie die Linie mit dem Hintergrund interagiert. Zur Auswahl stehen vier verschiedene Modi:

```
Standardeinstellung = 0
   XOR-Verknüpfung = 1
   AND-Verknüpfung = 2
    OR-Verknüpfung = 3
```

DRAWR

DRAWR X-Offset,Y-Offset[,[Farbstift][,Modus[1]]]

Das Kommando DRAWR arbeitet genau wie DRAW, nur daß der Endpunkt der Linie nicht durch absolute Koordinaten bestimmt wird, sondern durch einen X-/Y-Offset, der zu der aktuellen Position des Grafikcursors addiert wird.

1 Nicht auf dem CPC 464

FILL (nicht auf dem CPC 464)

FILL Farbstift

Mit dem BASIC-Kommando FILL können beliebige Flächen innerhalb des Grafikfensters des CPC gefüllt werden. Als Füllfarbe wird die dem Farbstift zugeordnete Farbe verwendet.

Der Prozeß des Füllens beginnt immer an der aktuellen Position des Grafikcursors. Als Begrenzung des zu füllenden Bereichs wird ein Linienzug angesehen, der die gleiche Farbe hat wie der Farbstift, mit dem beim Füllen geschrieben wird. Ebenfalls als Begrenzung gilt eine Linie, die mit dem Farbstift gezogen wurde, der durch den Befehl GRAPHICS PEN bestimmt ist. Steht der Grafikcursor beim Aufruf der Fillroutine auf einer solchen Begrenzung, so wird der Füllvorgang unmittelbar abgebrochen.

```
10 CLS
20 MOVE 100,100
30 DRAWR 0,200
40 DRAWR 200,0
50 DRAWR 0,-200
60 DRAWR -200,0
70 MOVE 200,200
80 FILL 1
90 END
```

Dieses kleine Programm zeichnet zunächst ein Quadrat (Zeilen 20 bis 60) und positioniert dann den Grafikcursor in die Mitte des Quadrats, womit die Voraussetzungen für das in Zeile 80 erfolgende Füllen geschaffen wurden.

FRAME (nicht auf dem CPC 464)

FRAME

Dieses BASIC-Kommando synchronisiert das Schreiben auf dem Bildschirm mit dem Strahlrücklauf. Mit seiner Hilfe kann flimmerfrei auf dem Bildschirm gezeichnet werden.

GRAPHICS PAPER (nicht auf dem CPC 464)

GRAPHICS PAPER Farbstift

Mit dem Kommando GRAPHICS PAPER wird die Farbe für den Hintergrund des Grafikbildschirms festgelegt. Es können Farbstifte aus dem Bereich 0 bis 15 angegeben werden. Das folgende Programm zeigt, wann der Hintergrund des Grafikbildschirms sichtbar wird.

```
10 MODE 0
20 TAG
30 GRAPHICS PAPER 10
40 MOVE 100,100
50 PRINT "A";
```

Das "A" wird in der eingestellten Zeichenfarbe auf den Bildschirm gebracht. Die Pixel der Zeichenmatrix, die bei der Darstellung des "A" nicht in Zeichenfarbe ausgegeben werden, nehmen die Farbe des mit GRAPHICS PAPER beeinflußten Hintergrundes an.

GRAPHICS PEN (nicht auf dem CPC 464)

GRAPHICS PEN [Farbstift][,Modus]

Mit dem BASIC-Kommando GRAPHICS PEN wird die Farbe, mit der auf dem Grafikbildschirm gezeichnet wird, festgelegt. Der Farbstift kann aus dem Bereich von 0 bis 15 stammen.

Der Parameter Modus bestimmt, ob die Bildschirmausgabe mit (0) oder ohne Hintergrund erfolgen soll. Folgendes Programm gibt ein Zeichen ohne seinen Hintergrund auf dem Grafikbildschirm aus.

```
10 MODE 0
20 TAG
30 GRAPHICS PEN 10,1
40 MOVE 100,100
50 PRINT "A";
```

Durch Ersetzen des zweiten Parameter durch eine 0 kann erwirkt werden, daß der Zeichenhintergrund wieder mit ausgegeben wird.

INK

INK Farbstift,Farbnummer[,Farbnummer]

Mit dem BASIC-Kommando INK werden den 16 möglichen Farbstiften des CPC ihre Farben zugeordnet. Als Farbstift kann eine Integerzahl aus dem Bereich 0 bis 15 eingesetzt werden. Abhängig vom eingestellten MODE ist die Anzahl der verfügbaren Farben unterschiedlich. In MODE 2 stehen 2 Farben zur Verfügung (monochrom), in MODE 1 vier Farben und in MODE 0 sechzehn Farben.

Nach den Einschalten des Schneider-CPC sind die Farbstifte mit folgenden Farben vorbelegt:

Farbstift	Farbe (MODE 0)	Farbe (MODE 1)	Farbe (MODE 2)
0	1	1	1
1	24	24	24
2	20	20	1
3	6	6	24
4	26	1	1
5	0	24	24
6	2	20	1
7	8	6	24
8	10	1	1
9	12	24	24
10	14	20	1
11	16	6	24
12	18	1	1
13	22	24	24
14	1,24	20	1
15	16,11	6	24

Mit dem INK-Kommando kann die Defaultbelegung der Farbstifte beliebig verändert werden. Werden beim INK-Kommando zwei Farbnummern angegeben, so wechselt die Farbe des zugeordneten Farbstiftes zwischen den beiden angegebenen Farben. Die Geschwindigkeit des Farbwechsels kann mit SPEED INK verändert werden.

LOCATE

LOCATE[#Stream,]X-Position,Y-Position

Mit dem LOCATE-Kommando kann der Textcursor an eine beliebige Position des Textfensters gebracht werden. Durch Angabe eines Streams kann angezeigt werden, daß der

Textcursor in einem anderen als Fenster #0 bewegt werden soll. Die Anzahl der möglichen Y-Positionen für den Textcursor ist immer 25; die möglichen X-Positionen differieren - abhängig vom eingestellten MODE - zwischen 20, 40 und 80. Der Koordinatenursprung (1,1) für das LOCATE-Kommando liegt an der Home-Position des Textfensters (linke obere Ecke).

MASK (nicht auf dem CPC 464)

MASK [Bitmuster][,Schalter]

Mit dem BASIC-Kommando MASK wird bestimmt, welches Bitmuster beim Zeichnen einer Linie zur Anwendung kommt. Für Bitmuster darf ein Wert zwischen 0 und 255 eingesetzt werden. Rechnet man diesen Wert in eine Dualzahl um (oder trägt unmittelbar die Dualzahl als Parameter ein), so steht jedes gesetzte Bit für ein gesetztes Pixel innerhalb eines acht Bit großen Abschnitts der Linie. Nach jeweils acht Bit wiederholt sich das Muster der Linie.

```
Striche: MASK &X11110000
 Punkte: MASK &X10101010
Strich-Punkt: MASK &X11100100
Voll-Linie: MASK &X11111111
```

Bevor die Schablone verändert wurde, zeichnet der CPC grundsätzlich eine durchgezogene Linie.

Der zweite Parameter des MASK-Kommando (Schalter) kann 0 oder 1 sein. Bei einer eingetragenen Null wird der erste Punkt einer Linie nicht gezeichnet; steht eine Eins, so wird der erste Punkt gezeichnet.

MODE

MODE Parameter

Das MODE-Kommando bestimmt den Bildschirmmodus. Bei Eingabe eines MODE-Kommandos werden alle Fenster, der Text- und der Grafikcursor auf ihre Defaultwerte gebracht. Als Parameter kann die 0, 1 oder 2 gegeben werden.

```
MODE 0: 20 Spalten, 16 Farben
MODE 1: 40 Spalten,  4 Farben
MODE 2: 80 Spalten,  2 Farben
```

Neben dem Auflösungsvermögen des Bildschirms beeinflußt das MODE-Kommando auch die Anzahl der maximal gleichzeitig darstellbaren Farben.

MOVE

MOVE X-Position,Y-Position[,Farbstift[1]][,Modus[1]]

Mit dem MOVE-Kommando wird der Grafikcursor an eine absolute Position auf dem Grafikbildschirm bewegt. Die Koordinaten, zu denen der Grafikcursor bewegt werden soll, sind als Parameter zum MOVE-Kommando angegeben. Soll der aktuelle Grafikfarbstift durch einen anderen ersetzt werden, so kann das durch die Angabe einer Farbstiftnummer für den Parameter Farbstift erreicht werden.

Mit dem Parameter Modus kann festgelegt werden, wie der aktuelle Farbstift mit dem Hintergrund interagiert. Zur Auswahl stehen vier verschiedene Modi:

1 Nicht auf dem CPC 464


```
Standardeinstellung = 0
    XOR-Verknüpfung = 1
    AND-Verknüpfung = 2
     OR-Verknüpfung = 3
```

MOVER

MOVER X-Offset,Y-Offset[,Farbstift[1]][,Modus[1]]

Das Kommando MOVER arbeitet genau wie MOVE, nur daß die Parameter nicht die absoluten Koordinaten für den Grafikcursor darstellen, sondern einen X-/Y-Offset, der zu der aktuellen Position des Grafikcursors addiert werden muß, um die neue Position des Grafikcursors zu gewinnen.

ORIGIN

ORIGIN X,Y[,links,rechts,oben,unten]

Das BASIC-Kommando ORIGIN setzt den Koordinatenursprung (0,0) für das Grafikfenster an den durch die Parameter X und Y bestimmten Punkt. Durch Angabe vier optionaler Parameter kann der Grafikbildschirmbereich begrenzt werden.

PAPER

PAPER [#Stream,]Farbstift

Mit dem PAPER-Kommando kann die Hintergrundfarbe verändert werden. Die Anzahl der zur Verfügung stehenden Farbstifte ist vom aktiven MODE abhängig. Durch die Angabe eines Streams kann die Veränderung der Hintergrundfarbe für ein anderes Fenster als #0 geltend gemacht werden.

1 Nicht auf dem CPC 464

PEN

PEN [#Stream,][Farbstift][Modus[1]]

Mit dem PEN-Kommando kann die Farbe, in der Zeichen auf dem Bildschirm erscheinen, verändert werden. Die Anzahl der zur Verfügung stehenden Farbstifte ist vom aktiven MODE abhängig. Durch die Angabe eines Streams kann die Veränderung der Hintergrundfarbe für ein anderes Fenster als #0 geltend gemacht werden. Der Parameter Modus bestimmt, ob die Bildschirmausgabe transparent (1) oder nicht (0) erfolgen soll.

PLOT

PLOT X-Position,Y-Position[,Farbstift][,Modus[1]]

Mit dem PLOT-Kommando wird der Grafikcursor an eine absolute Position auf dem Grafikbildschirm bewegt und an dieser Stelle ein Pixel ausgegeben. Soll der aktuelle Grafikfarbstift durch einen anderen ersetzt werden, so kann das durch die Angabe einer Farbstiftnummer für den Parameter Farbstift erreicht werden.

Mit dem Parameter Modus kann festgelegt werden, wie der aktuelle Farbstift mit dem Hintergrund interagiert. Zur Auswahl stehen vier verschiedene Modi:

```
Standardeinstellung = 0
    XOR-Verknüpfung = 1
    AND-Verknüpfung = 2
     OR-Verknüpfung = 3
```

1 Nicht auf dem CPC 464

PLOTR

PLOTR X-Offset,Y-Offset[,Farbstift][,Modus[1]]

Das Kommando PLOTR arbeitet genau wie PLOT, nur daß die Parameter nicht die absoluten Koordinaten für die Ausgabe des Pixels darstellen, sondern einen X-/Y-Offset, der zu der aktuellen Position des Grafikcursors addiert werden muß, um die neue Position zu gewinnen, an der das Pixel ausgegeben werden soll.

POS

POS(#Stream)

Die BASIC-Funktion POS liefert die aktuelle X-Koordinate (bezogen auf den linken Rand des Bildschirms), an der sich der Textcursor befindet. Der Stream muß unbedingt als Ergänzung zu dieser Funktion angegeben werden.

SPEED INK

SPEED INK Dauer 1,Dauer 2

Mit den BASIC-Kommandos BORDER und INK ist es möglich, zwei Farben zu definieren, die abwechselnd an einer eingefärbten Stelle erscheinen sollen. Die Geschwindigkeit des Farbwechsels läßt sich mit dem Kommando SPEED INK verändern. Der Parameter Dauer 1 bestimmt, wie lange die erste, der Parameter Dauer 2, wie lange die zweite Farbe sichtbar sein soll. Die Zeitdauer kann in Einheiten von Fünfzigstelsekunden bestimmt werden.

1 Nicht auf dem CPC 464

SYMBOL

SYMBOL Zeichennummer,Zeichenmatrix

Das BASIC-Kommando SYMBOL benötigt neun ergänzende Parameter, deren Werte zwischen 0 und 255 liegen müssen.

Mit dem ersten Wert der auf SYMBOL folgt (Zeichennummer), wird festgelegt, welches Zeichen durch eine neue vom Benutzer definierte Matrix ersetzt werden soll. Die acht dann folgenden Parameter definieren die Matrix.

```
10 SYMBOL AFTER 123
20 SYMBOL 123,&X11011011,&X00111100,&X01100110,&X01100110,&X011111
10,&X01100110,&X01100110,&X00000000
```

Dieses Programm definiert mit Hilfe des BASIC-Kommandos SYMBOL das Zeichen 123 ({) in ein "Ä" um. Die gewählte binäre Notation für die letzten acht Parameter verdeutlicht die Beziehung zur Zeichenmatrix. Ist im ersten dieser acht Parameter das signifikanteste Bit gesetzt, so wird dieses Bit auch in der Zeichenmatrix gesetzt. Analoges gilt für alle übrigen Bits der letzten acht Parameter (vgl. CHR$(25)).

Zu beachten ist vor der Anwendung des SYMBOL-Kommandos, daß mit SYMBOL AFTER die gewünschte Zeichennummer zur Umdefinition vorbereitet wurde.

SYMBOL AFTER

SYMBOL AFTER Parameter

Mit dem Kommando SYMBOL AFTER wird festgelegt, welche Zeichen vom Benutzer umdefiniert werden können. Der Parameter kann Werte zwischen 0 und 255 annehmen, die dann die Zeichennummer widerspiegeln, ab der (einschließlich) alle Zeichen durch vom Benutzer mit SYMBOL definierte Zeichenmatrizen ersetzt werden können. Wird als Parameter der Wert 256 an das SYMBOL AFTER-Kommando angehangen, so können keine neuen Zeichen definiert werden.

TAG

TAG [#Stream]

Das BASIC-Kommando TAG bestimmt, daß ab sofort die Ausgabe von Textzeichen nicht mehr an der Position des Textcursors, sondern an der Position des Grafikcursors erfolgen soll. Dabei wird der Text so ausgegeben, daß die höchstwertigste Pixelposition in einer Zeichenmatrix an den Koordinaten des Grafikcursors steht. Durch die Angabe des optionalen Streamparameters wird festgelegt, aus welchem Stream die auszugebenden Textzeichen bezogen werden sollen.

Wird nach gegebenem TAG mit PRINT eine Textausgabe vorgenommen, so muß sie mit einem ";" abgeschlossen werden, da anderenfalls Steuerzeichen als Textzeichen ausgegeben werden.

TAGOFF

TAGOFF [#Stream]

Das BASIC-Kommando TAGOFF macht ein gesetztes TAG für den angegebenen Stream rückgängig, so daß die Textausgabe wieder an der Position des Textcursors erfolgt.

TEST

TEST(X-Position,Y-Position)

Die Funktion TEST setzt den Grafikcursor auf die als Parameter angegebene Bildschirmposition. Dann liefert TEST die Nummer des Farbstifts, der an dieser Position verwendet wurde.

TESTR

TESTR(X-Offset,Y-Offset)

Die Funktion TESTR setzt den Grafikcursor, gemäß dem angegebenen X- und Y-Offset, relativ zu seiner ursprünglichen Position. Dann liefert TESTR die Nummer des Farbstifts, der an dieser Position verwendet wurde.

VPOS

VPOS(#Stream)

Die BASIC-Funktion VPOS liefert die aktuelle Y-Koordinate (bezogen auf den oberen Rand des Bildschirms), an der sich der Textcursor befindet. Der Stream muß unbedingt als Ergänzung zu dieser Funktion angegeben werden.

WINDOW

WINDOW [#Stream,]links,rechts,oben,unten

Das BASIC-Kommando WINDOW definiert die Größe eines Fensters. Da es sich bei den einzugebenden Koodinaten um Koordinaten des Textbildschirms handelt, ist die Spaltenanzahl des eingestellten MODE bei der Definition eines Fensters zu berücksichtigen. Wird kein Stream angegeben, so bezieht sich das Kommando automatisch auf Stream #0.

WINDOW SWAP

WINDOW SWAP Stream,Stream

Das Kommando WINDOW SWAP vertauscht die durch die beiden Streams bestimmten Bildschirmbereiche miteinander. Den

Streamnummern darf kein "#" vorangestellt werden, um die Syntax von WINDOW SWAP nicht zu verletzen.

XPOS

XPOS

Die BASIC-Funktion XPOS liefert die aktuelle X-Koordinate, an der sich der Grafikcursor befindet.

YPOS

YPOS

Die BASIC-Funktion YPOS liefert die aktuelle Y-Koordinate, an der sich der Grafikcursor befindet.

Glossar

Adresse
ist ein Speicherplatz, der in der Regel als eine 2-Byte-hexadezimale Zahl spezifiziert wird. Der Adreßbereich (0 bis 32767) wird dezimal dargestellt als [0000 to FFFF] [-32768,...,-1].

Akkumulator (Akku)
CPU-Register, in dem Ergebnisse arithmetischer, logischer und Ein-/Ausgabeoperationen gebildet werden.

Algorithmus (Rechenvorschrift)
Eine Schrittweise vorgehende Ausarbeitung eines Lösungsweges für ein gestelltes Problem wird Algorithmus genannt. Ein Algorithmus kann durch beliebige Symbole oder Sprachelemente ausgedrückt werden.

alphabetisch
bezieht sich strikt auf die Buchstaben A bis Z.

alphanumerisch
bezieht sich auf die Buchstaben A bis Z und die Ziffern 0 bis 9.

Anweisung (statement)
ist in BASIC eine komplette Instruktion.

Arbeitsspeicher
(siehe RAM)

Argument
ist ein String (Zeichenfolge) oder eine numerische Menge, die an eine Funktion geht, dort bearbeitet wird und ein Ergebnis bringen soll. Dieses Ergebnis ist der Wert der Funktion.

Array
(siehe Datenfeld)

ASCII (American Standard Code for Information Interchange)
International normierter 7-Bit-Code, der zum Speichern von Textdaten und Datenaustausch verwendet wird.

Assembler
Programmiersprache, bei der der binäre Maschinencode durch einfach zu merkende Kürzel (Mnemonics) dargestellt wird.

BASIC
Abkürzung für Beginners All purpose Symbolic Instruction Code.

Basisadresse
Durch Verknüpfung der Basisadresse mit einem Offset (Relativadresse) wird eine absolute Adresse gewonnen.

Binär (binary)
Es gibt nur zwei mögliche Darstellungen, entweder 0 oder 1. Das binäre Zahlensystem (auf der Basis von 2) benutzt zur Darstellung von Mengen Nullen und Einsen. Dies entspricht der internen Darstellung von Daten im Computer, bei der für 0 und 1 elektrische Zustände verwendet werden.

Bit
Abkürzung für Binary Digit. Ein Bit ist die kleinste Speichereinheit im Computer und stellt die Werte 0 und 1 dar.

Buffer
(siehe Puffer)

Byte
ist die kleinste adressierbare Speichereinheit im Computer. Sie besteht aus 8 Bits und kann 256 verschiedene Werte darstellen, z.B. die Dezimalwerte von 0 bis 255.

Charakter
(siehe Zeichen)

Charakter-Grafik
Computergrafik, die aus einer Vielzahl von Zeichen, und Sonderzeichen, die mit dem Charakter-Generator des Computers erzeugt werden, besteht.

Cursor
wird die Markierung genannt, die die nächste Schreibposition anzeigt.

Daten
sind Informationen, die an ein Programm gehen oder aus einem Programm kommen. Es gibt vier Arten von Daten:

- Ganzzahlen
- Zahlen mit einfacher Genauigkeit
- Zahlen mit doppelter Genauigkeit
- Zeichenfolgen (Strings)

Datenfeld (Array)
ist ein organisierter Satz von Elementen, die insgesamt oder einzeln über den Namen des Datenfeldes und ein oder mehrere Subskripte angesprochen werden können. In BASIC kann jeder variable Namen zur Benennung eines Datenfeldes verwendet werden. Datenfelder können eine oder mehrere Dimensionen haben. AR(x) bedeutet ein eindimensionales Feld mit dem Namen AR; AR(x,y) bedeutet ein zweidimensionales Feld mit dem Namen AR usw.

Default
ist eine Aktion oder ein Wert, der vom Programm geliefert wird, wenn Sie nicht angegeben haben, welche Aktion oder welcher Wert vorgenommen bzw. verwendet werden soll.

Device
(siehe Gerät)

Dezimal
Es gibt 10 mögliche Zahlendarstellungen, nämlich von 0 bis 9. Das Dezimalsystem (auf der Basis von 10) ist uns aus dem Alltag geläufig. Verwendet wird eine Folge von dezimalen Ziffern. Der Schneider-CPC speichert die Dezimalzahlen im Binärcode.

Disassembler
sind Programme, die einen Hexcode in die mnemonische Darstellung übersetzen und die korrekte Befehlslänge erkennen.

Diskette
ist ein magnetisches Aufzeichnungsmedium zur Speicherung großer Datenmengen.

Drahtmodell
So nennt man die mit Hilfe der Vektorgrafik generierten dreidimensionalen Objekte.

Editieren/Bearbeiten
Hierdurch werden die vorhandenen Informationen geändert.

Gerät (Device)
Hierunter versteht man den physikalischen Teil des Computersystems, der für die Ein-/Ausgabe von Daten verwendet wird, z.B. die Tastatur, der Bildschirm oder der Drucker.

Hexadezimal (Hex)
Im Hexadezimalformat sind 16 Darstellungen möglich. Die hexadezimalen Ziffern werden mit 0, 1, 2,..., 9, A, B, C, D, E, F, dargestellt. Hexadezimale Zahlen (auf der Basis von 16) sind Folgen von hexadezimalen Ziffern. Adreß- und Byte-Werte werden häufig in hexadezimaler Form angegeben. Beim Schneider-CPC können hexadezimale Konstanten eingegeben werden, indem man diese mit einem Präfix "&" oder "#" versieht.

Inkrement
ist der Wert, um den der Zähler jedesmal erhöht wird, wenn eine sich wiederholende Prozedur beendet wird.

Interpreter
z.B. BASIC-Interpreter - deutet und führt die Programmierbefehle zeilenweise als Folge von Maschinenbefehlen aus.

Kilobyte oder KByte
sind 1024 Speicher-Bytes. Somit hat ein 64-KByte-System eine Speicherkapazität von 64 x 1024 = 65536 Bytes.

Kompatibilität
nennt man die Austauschbarkeit von Systemteilen oder Programmen, die ohne Modifikation auf einem anderen System verwendet werden können.

LSB
Abkürzung für niederwertiges Byte (Low-Byte).

LSN
Abkürzung für niederwertiges Nibble (Halbbyte).

Maschinensprache
ist z.B. der Z80A-Befehlssatz, der im allgemeinen im Hexadezimal-Code spezifiziert wird. Alle höheren Programmiersprachen müssen in die Maschinensprache übersetzt oder von der Maschinensprache interpretiert werden, um vom Computer ausgeführt werden zu können.

Menü
nennt man Übersichtstafeln, die dem Benutzer anzeigen, was als nächstes zu tun ist bzw. welche Alternativen sich anbieten.

Mnemonic
Unter einem mnemonischen Code versteht man einen Code, dessen Abkürzung einen Hinweis auf seine Eigenschaften enthält.

MSB
Abkürzung für höchstwertiges Byte (High-Byte).

MSN
Abkürzung für höchstwertiges Nibble (Halbbyte).

Nibble
Bezeichnung für eine Gruppe von vier Bits (Halbbyte).

Paramter
sind Informationen, die zusammen mit einem Befehl geliefert werden und spezifizieren, wie der Befehl verarbeitet werden soll.

Peripherie
nennt man alle Ein- und Ausgabegeräte, Speicher und Hilfsspeicher, die sich außerhalb des Rechners befinden und mit ihm zusammenarbeiten.

Pixel
Fachausdruck für Grafikpunkt.

Puffer
sind im Rechner reservierte Bereiche, in denen Daten zwischengespeichert werden können.

Pufferspeicher (Buffer)
ist ein Bereich im RAM-Speicher, in dem Daten für die weitere Verarbeitung akkumuliert werden.

RAM (Random Access Memory)
Schreib-/Lesespeicher mit wahlfreiem Zugriff.

Raster-Grafik
Ein Verfahren, mit dessen Hilfe der Bildschirm Punkte aufgeteilt wird, die einzeln beeinflußt (an- oder ausgeschaltet bzw. umgefärbt) werden können.

ROM (Read Only Memory)
Nur Lesespeicher, der auch nach dem Abschalten der Stromversorgung seinen Inhalt behält.

Routine/Unterprogramme
ist eine Folge von Befehlen zur Ausführung einer bestimmten Funktion. Typischerweise ist eine Routine ein Unterprogramm, das von mehreren Stellen im Programm aufgerufen wird.

Sonderzeichen
In der Datenverarbeitung faßt man alle Zeichen, die weder Buchstaben noch Ziffer sind - z.B. "+", "@", "^" - unter dem Begriff Sonderzeichen zusammen.

Statement
(siehe Anweisung)

Stream
Der Stream bezieht sich auf die Ein-/Ausgabe des Schneider-CPC. Mit der Stream-Nummer kann ein derartige Ein-/Ausgabeschnittstelle angesprochen werden.

Syntax/Schreibweise
Unter Syntax versteht man die "grammatischen" Vorschriften für einen Befehl oder eine Anweisung. Allgemein bezieht sich die Syntax auf die Zeichensetzung und die Reihenfolge der Elemente einer Anweisung.

Unterbrechung (Break)
unterbricht die Ausführung eines Programms. In BASIC bewirkt eine STOP-Anweisung die Unterbrechung des Programms, ebenso die BREAK-Taste.

Vektor
1. Programmspeicheradresse, die dem Rechner abhängig von einem bestimmten Ergebnis mitgeteilt wird.
2. siehe Datenfeld
3. eine gerichtete, orientierte Strecke im Raum.

Vektorgrafik
die Vektorgrafik ist ein Verfahren, das es gestattet, dreidimensionale Körper mit einem Computer darzustellen. Körper, die mit Hilfe von Vektorgrafiken generiert wurden, bestehen aus einer Vielzahl von Linien, die die Kanten des Objekts symbolisieren.

Wire Frame
(siehe Drahtmodell)

Zeichen
Ein Zeichen ist ein vom Computer generiertes Bildelement. Buchstaben und Ziffern sind beispielsweise die häufig verwendete Zeichen.

Index

Literaturverzeichnis

CPC 6128 Benutzerhandbuch (1. Auflage 1985)
Schneider Computer Division, Türkheim

CPC 464 Intern (1. Auflage 1985)
DATA BECKER Verlag, Düsseldorf

CPC 664/6128 Intern (1. Auflage 1985)
DATA BECKER Verlag, Düsseldorf

Das Maschinensprachebuch zum CPC 464 (1. Auflage 1985)
DATA BECKER Verlag, Düsseldorf

GSX - Programmer's Guide/User's Guide (1986)
DIGITAL RESEARCH, Pacific Grove, California

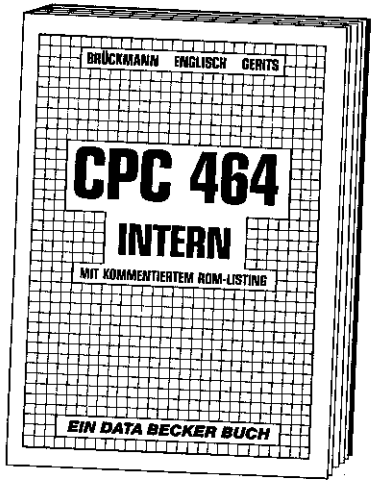
Joyce PCW 8256/8512 Benutzerhandbuch (2. Auflage 1986)
Schneider Computer Division, Türkheim

Kaufmännische Betriebslehre - Hauptausgabe (1986)
Europa Lehrmittel, Wuppertal

Mathematik Sekundarstufe II Analysis Grundkurse Neu (1982)
Schwann, Düsseldorf

Bücher zum CPC

CPC 464 INTERN ist eine unentbehrliche Hilfe für den fortgeschrittenen Basic-Programmierer und ein absolutes Muß für den Assembler-Programmierer. Hier wurde in monatelanger Kleinarbeit ein Standardwerk geschaffen, das wirklich alle Geheimnisse des CPC 464 enthüllt.



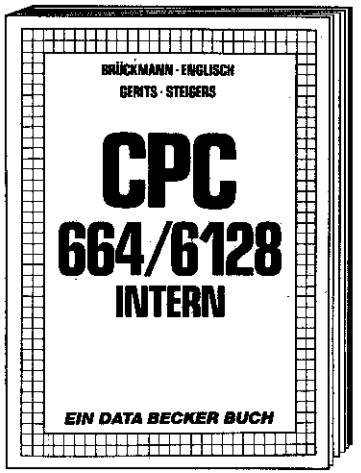
Aus dem Inhalt:

- Das sollten Sie von Ihrem Gerät wissen
- Die Speicheraufteilung
- Der Prozessor
- Besonderheiten des Z80 im CPC 464
- Das Gate Array
- Der Video-Controller
- Das Video-Ram
- Der Soundchip
- Die Schnittstellen
- Das Betriebssystem
  Nutzung der Routinen am Beispiel
  Hardcopy
  Der Character-Generator
- Der BASIC-Interpreter
- BASIC und Maschinensprache
- Das BASIC-ROM-Listing
  und vieles mehr

Brückmann, Englisch, Gerits
CPC 464 Intern mit kommentiertem ROM-Listing
552 Seiten, DM 69,–
ISBN 3-89011-080-0

Bücher zum CPC

Das CPC 664/6128 INTERN ist eine unentbehrliche Hilfe für den ambitionierten BASIC-Programmierer und ein absolutes Muß für den Maschinensprache-Profi.
In monatelanger Kleinarbeit wurde eine Standardwerk geschaffen, das wirklich kein Geheimnis des CPC 664/6128 unergründet läßt.



Aus dem Inhalt:

- Die Speicheraufteilung
- Der Z-80-Prozessor
- Das Gate-Array
- Der Video-Controller
- Der Sound-Chip
- Die Schnittstellen
- Das Betriebssystem
- Der Character-Generator
- Der Basic-Interpreter
- Ein kompletter Disassembler
  und vieles mehr

Brückmann, Englisch, Gerits, Steigers
CPC 664/6128 Intern
456 Seiten, DM 69,–
ISBN 3-89011-135-1

Bücher zum CPC

Technik & Programmierung des Z80-Prozessors sind Thema dieses Buches. Ein unentbehrliches Lehrbuch und Nachschlagewerk für alle, die einen Commodore 128, Schneider CPC, MSX oder anderen Rechner mit Z80-Prozessor haben und in Maschinensprache programmieren wollen!

Aus dem Inhalt:

- Systemarchitektur
- Pinbeschreibung
- Register
- Befehlsausführung
- Flags
- CPU-Software
- Anschluß von Systembauteilen
- serielle/parallele Datenübertragung
- Zähler-/Timerbaustein Z80-CTC mit Befehlssatz

Hausbacher
Das Prozessor-Buch zum Z80
568 Seiten, DM 59,–
ISBN 3-89011-096-7

LADEN, STARTEN – KLAR!

Für alle diejenigen, die sich fleißiges Abtippen ersparen und trotzdem alle Programme nutzen wollen, gibt es einen Ausweg:

Mit der nebenstehenden Post-Zahlkarte einfach die Diskette zum Buch bestellen!

Diskette zum Buch „Das große Grafikbuch zum CPC“ DM 29,–