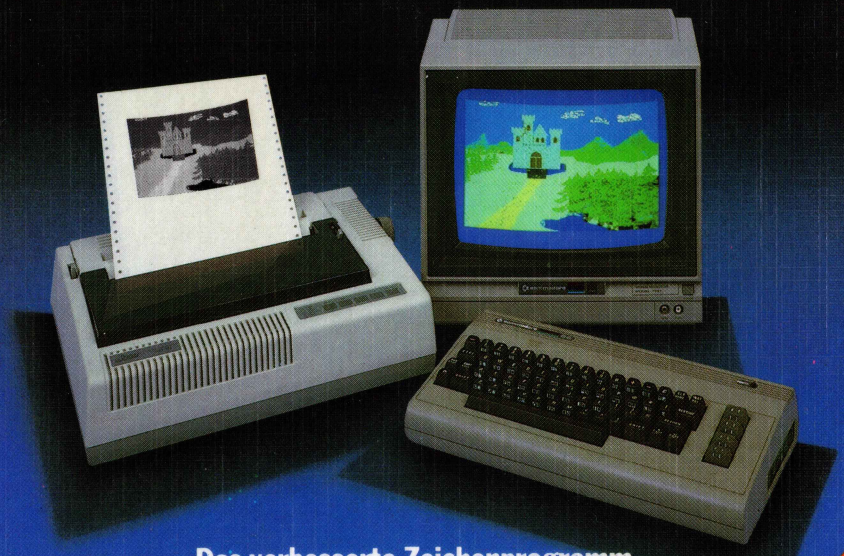
Markt&Technik

Hans Haberl

Mini-CAD mit Hi-Eddi plus auf dem C64 / C128



Das verbesserte Zeichenprogramm aus der Zeitschrift 64er. Mit ausführlicher Dokumentation, vielen Anwendungsbeispielen und neuen Features wie ein komfortables Druckeranpassungsprogramm oder verschiedene Zeichensätze und Construction Sets.

Inklusive Disk mit Hi-Eddi plus und allen Beispielen

Mini-CAD mit Hi-Eddi plus auf dem C64/C128

Hans Haberl

Mini-CAD mit Hi-Eddi plus auf dem C64/C128

Das verbesserte Zeichenprogramm aus der Zeitschrift 64er.
Mit ausführlicher Dokumentation, vielen Anwendungsbeispielen und neuen Features, wie ein komfortables Druckeranpassungsprogramm oder verschiedene Zeichensätze und Construction Sets.

Markt & Technik Verlag

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Haberl, Hans:
Mini-CAD mit Hi-Eddi plus auf dem C64/C128 : d. verb. Zeichenprogramm aus d. Zeitschr. 64er ;
mit ausführl. Dokumentation, vielen Anwendungsbeispielen u. neuen Features,
wie e. komfortables Druckeranpassungsprogramm oder verschiedene Zeichensätze u. construction sets / Hans Haberl. –
Haar bei München : Markt-und-Technik-Verlag, 1986. & Diskette
ISBN 3-89090-136-0

15 14 13 12 11 10 9 8 7 6
89 88 87 86

ISBN 3-89090-136-0

Einbandgestaltung: Grafikdesign Heinz Rauner
Druck: Schoder, Gersthofen
Printed in Germany

Inhaltsverzeichnis

Vorwort

Während bis vor kurzem Homecomputer noch als bessere Telespiele angesehen und oft auch nur als solche verwendet wurden, dringen sie heute mehr und mehr in ernsthafte Anwendungsbereiche vor. Textverarbeitung, Dateiverwaltung, Kalkulation und Lernen sind nur die populärsten der Anwendungen, für die es auf dem Markt bereits eine Fülle von Software gibt. Möglich wurden diese Anwendungen, die bislang noch den teuren Personal Computern vorbehalten waren, erst durch die rasante technische Entwicklung, die es mit sich brachte, daß heute eine Speicherkapazität von 64 kByte RAM auch bei Computern unter 1000 DM schon Standard ist. (Bald werden wohl 128 kByte das Minimum sein!)

Doch nicht nur die Speicherkapazität wurde wesentlich verbessert, auch die grafischen Fähigkeiten vieler Homecomputer erreichen oder übertreffen sogar die diesbezüglichen Fähigkeiten der meisten PCs. Insbesondere die Nummer Eins auf dem deutschen Homecomputer-Markt, der Commodore 64, setzte mit seiner Auflösung von 320*200 Punkten und 16 Farben neue Maßstäbe. Damit wird für Homecomputer ein Anwendungsbereich zugänglich, der sogar für die PCs noch keineswegs selbstverständlich ist: CAD.

CAD oder Computer Aided Design bedeutet: computerunterstütztes Entwerfen oder Konstruieren. Natürlich ist damit nicht nur die Grafik gemeint, wie oft angenommen wird. So gehört zum Beispiel bei der Entwicklung einer elektrischen Schaltung von der Berechnung der Elemente über die Simulation, das Erstellen des Schaltbildes und des Layouts, der Dokumentation und des Tests alles zu CAD. Und weiter geht es dann mit CAM oder Computer Aided Manufacturing, der computerunterstützten Fertigung. Alles zusammen bezeichnet man als CAE oder Computeraided Engineering.

Darauf wollen wir jedoch im vorliegenden Buch nicht näher eingehen. Vielmehr werden wir uns auf den grafischen Aspekt konzentrieren, d.h. das Erstellen von technischen Plänen und Skizzen aller Art, wie zum Beispiel Schaltpläne, Layouts, Konstruktionszeichnungen, Programmlaufpläne und Struktogramme, Blockschaltbilder, Grundrisse, Zeichnungen zur Dokumentation und anderes mehr. Sogar Glückwunschkarten, Briefköpfe, bewegte Schaufensterwerbung oder Bilder für Grafik-Adventures können mit dem Commodore 64 und der geeigneten *Software* erstellt werden. Damit sind wir beim Stichwort *Software.*

Eine gute Software ist die erste Voraussetzung, und als Besitzer eines Commodore 64 wissen Sie sicherlich auch, daß seine grafischen Fähigkeiten nicht allzu umfangreich sind. Während in der Werbung die grafischen Fähigkeiten des C64 groß angepriesen werden, ist im *Handbuch* kein Wort darüber zu lesen. Der Grund dafür ist, daß das im C64 eingebaute Minimal-BASIC die Grafik überhaupt nicht unterstützt. Eine gute BASIC-Erweiterung ist deshalb unbedingt erforderlich. Die bekannteste und auch eine der besten ist Simon's BASIC. Damit kann man jedoch - was CAD anbelangt - die Möglichkeiten des C64 nicht voll ausschöpfen. Erstens stellt Simon's BASIC nur einen Grafik-Bildschirm zur Verfügung und zweitens sind viele wichtige Befehle, wie etwa das Verschieben von Bildschirmausschnitten, nicht enthalten und müßten daher erst in BASIC programmiert werden. Das ist jedoch extrem langsam!

Da wir jedoch keine Kompromisse eingehen wollen und die CAD-Möglichkeiten des Commodore 64 alle ausnutzen wollen, bleibt nur eines:

Maschinensprache!

Nur in Maschinensprache oder Assembler, was dasselbe bedeutet, kann man einen Rechner optimal programmieren und alle Möglichkeiten ausnutzen. Da jedoch das Eingeben von größeren Assemblerprogrammen eine äußerst mühsame Angelegenheit ist, liegt diesem Buch eine Diskette bei, auf der Sie nicht nur das CAD-Programm selbst, sondern eine ganze Reihe von Construction-Sets, Zeichensätzen und anderen nützlichen Hilfen zum computerunterstützten Zeichnen finden.

Allerdings soll nicht unerwähnt bleiben, daß man auch mit der besten Software auf einem C64 nie das erreichen wird, was die professionellen CAD-Systeme bieten. Wer würde sonst schon ein viele tausend Mark teueres System kaufen, wenn ein Homecomputer den gleichen Zweck erfüllen würde! Das Programm, das diesem Buch zugrunde liegt, geht allerdings schon weit über eine reine Einführung in CAD hinaus.

Sicherlich werden Sie die Fähigkeiten dieses Programms überzeugen. Falls Sie das Zeichenprogramm **Hi-Eddi** aus der Zeitschrift *64'er* kennen, wird Ihnen das diesem Buch beigefügte Grafikprogramm **Hi-Eddi plus** sicherlich bekannt vorkommen. Falls Sie jedoch der Meinung sind, der Hi-Eddi aus der 64er wäre ein Programm, das die Möglichkeiten und Fähigkeiten des C64 bereits weitgehend ausschöpft, dann muß ich Sie eines Besseren belehren: Der Hi-Eddi plus bietet noch eine ganze Menge mehr! Wenn Sie einen kurzen Blick in die Einleitung dieses Buches werfen, erfahren Sie dort bereits in Stichpunkten, was der neue Hi-Eddi alles kann.

Einleitung

In der Zeitschrift 64'er, Ausgabe 1/85 wurde bereits das Programm **Hi-Eddi** veröffentlicht. Die Abkürzung steht für *High-Resolution-Grafik-Editor*, also Editor für die hochauflösende Grafik.

Mit diesem Programm stand bereits ein sehr komfortables Mittel zur Erstellung von Grafiken auf dem C64 zur Verfügung. Neben den *Standardbefehlen* zum Setzen und Löschen von Punkten, dem Zeichnen von Linien, Kreisen und Rechtecken sowie dem Ausfüllen unregelmäßiger Flächen und dem Verschieben und Duplizieren von Bildschirmbereichen bietet Hi-Eddi eine Reihe von Besonderheiten, die dieses Programm von anderen Grafikprogrammen abhebt. Das hervorragendste Merkmal ist wohl die Tatsache, daß bis zu sieben Grafikbildschirme gleichzeitig zur Verfügung stehen. Da diese auf einem Epson-Drucker oder einem Epson-kompatiblen Drucker über- und nebeneinander ausgedruckt werden können, lassen sich Zeichnungen erstellen, die die Auflösung des Computers um ein Vielfaches übertreffen. Auch die Möglichkeiten, Text in die Grafik einzufügen, die Bildschirme zu verknüpfen oder in schneller Folge durchzuschalten, womit ein Trickfilm-Effekt entsteht, zählen zu den nicht alltäglichen Fähigkeiten dieses Programms.

Die Resonanz auf das Programm war äußerst rege: Ich erhielt weit über hundert Zuschriften und Anrufe, viele mit Verbesserungsvorschlägen und Wünschen. An Ideen mangelte es mir also nicht mehr. Folglich setzte ich mich hin und erweiterte den *alten* Hi-Eddi zu einem Superprogramm, von dem ich ohne Übertreibung behaupten kann, daß es derzeit an CAD für den C64 nichts Besseres gibt und wohl so schnell auch nicht geben wird.

Das Ergebnis ist der diesem Buch beiliegende "Hi-Eddi plus". Und nun wird Sie natürlich interessieren, was dieses Programm gegenüber dem alten Hi-Eddi alles dazugelernt hat. Hier ist eine Aufstellung seiner zusätzlichen Fähigkeiten. Die Fähigkeiten des alten Hi-Eddi bleiben natürlich voll erhalten!

* Neben der Joystick-Steuerung können Sie den Cursor nun auch mit einem Koala-Pad steuern. Denken Sie dabei aber nicht an die zittrige und wacklige Koala-Pad-Steuerung anderer Zeichenprogramme, denn beim Hi-Eddi plus ist dies wesentlich besser gelöst!

* Sie können nun beliebige Zeichensätze erstellen und Ihre Grafiken damit beschriften - in alle vier Richtungen, wie es für technische Zeichnungen notwendig ist.

* Eine echte Zoom-Funktion sorgt dafür, daß Sie zum punktgenauen Zeichnen kleiner Details keine Lupe brauchen. Und damit Sie beim Zoomen nicht die Übersicht verlieren, sehen Sie neben dem vergrößerten Ausschnitt noch den entsprechenden Bereich des Bildschirms in Originalgröße, in dem Sie die Wirkung Ihrer Manipulationen sofort verfolgen können.

* Wie bisher bereits im Sprite-Editor, können Sie nun auch im Grafikeditor beliebige Bereiche oder ganze Bildschirme spiegeln und um 90 oder 180 Grad drehen. Das Verknüpfen, bisher nur für ganze Bildschirme möglich, funktioniert jetzt auch für beliebige Bereiche.

* Der Paint-Befehl füllt unregelmäßige Flächen wahlweise auch mit einem Raster auf.

* Daneben gibt es einige vollkommen neue Befehle, wie eine Spray-Funktion und einen Befehl zum schnellen Kopieren ganzer Bildschirme.

* Auch die Drucker-Routine hat ihre Fähigkeiten erweitert. Sie kann jetzt auch die Commodore-Drucker MPS801/VC1525 ansteuern, mit Epson- und Epson-kompatiblen Druckern ist nun auch Double-Strike-Druck möglich, und der Plot-Modus, mit dem Kreise auch wirklich rund werden, kann angesteuert werden.

* Erwähnenswert sind außerdem die vielen Construction-Sets und Zeichensätze auf der Diskette, wie zum Beispiel die Construction-Sets zum Erstellen von Schaltplänen für analoge und digitale Schaltungen; oder die Zierschrift-Zeichensätze, die Sie für Glückwunschkarten, Einladungen zur Geburtstagsparty oder für ein Namensschild für Ihren Computerarbeitsraum verwenden können.

Wem das alles noch zu wenig ist, für den gibt es auf der Diskette ein Zusatzprogramm, das auf Kosten eines Grafikspeichers einige ganz besonders wertvolle Hilfen bietet. Darunter fällt zum Beispiel die Koordinatenanzeige, die das Erstellen maßstabgerechter Zeichnungen erleichtert; oder die erweiterten Load- und Save-Befehle, die eine Reihe zusätzlicher Formate zur Diskettenspeicherung zur Verfügung stellen; oder die Definition von Makro-Befehlen, die fast schon eine kleine Programmiersprache darstellen. Schließlich gibt es noch als ganz besondere Einrichtung den Scroller, der sechs Bildschirme wie einen Riesenbildschirm mit 640*600 Punkten verwaltet. Das Zusammenstückeln der Bilder für einen mehrteiligen Ausdruck hat damit ein Ende, denn nun fahren Sie den sichtbaren Bildschirm wie ein Fenster über das gesamte Bild.

Im einzelnen behandelt das vorliegende Buch folgende Themenbereiche:

Kapitel 1 enthält eine sehr ausführliche Bedienungsanleitung, wobei alle Befehle des Hi-Eddi plus erklärt werden. Dieses Kapitel ist auch als Nachschlagewerk gedacht, wenn Sie einmal nicht mehr genau wissen, wie ein bestimmter Befehl funktioniert.

In Kapitel 2 möchte ich Ihnen anhand einer Reihe von konkreten Beispielen schrittweise aufzeigen, wie man mit Hi-Eddi plus arbeitet. Dabei werden Sie nicht nur eine Menge Anregungen über die vielfältigen Verwendungsmöglichkeiten von Hi-Eddi plus erhalten, sondern ich werde Ihnen auch viele Tricks verraten, wie Sie Hi-Eddi plus voll ausnutzen und einige scheinbare Schwächen des Programms umgehen oder sogar für Ihre Zwecke einsetzen können. Wer mit der theoretischen Beschreibung der Befehle in Kapitel 1 noch nicht zurechtkommt, der wird in Kapitel 2 über deren praktische Anwendung informiert. Damit wird es auch dem Laien möglich, in kürzester Zeit anspruchsvolle Computergrafiken zu erstellen.

In Kapitel 3 erfahren Sie, wie Sie Hi-Eddi plus an Ihren Drucker anpassen und wie Sie Ihren Drucker optimal einsetzen.

In Kapitel 4 folgt eine Beschreibung der Erweiterung und ihrer Bedienung.

In Kapitel 5 schließlich kommt auch der Profi, also der in Maschinensprache erfahrene Computerfreak, auf seine Kosten. Dieses Kapitel beinhaltet eine Dokumentation des Programms, die es dem Profi ermöglicht, Routinen aus dem Hi-Eddi plus in eigenen Programmen zu verwenden oder, was noch wesentlich interessanter ist, den Hi-Eddi plus individuell zu erweitern.

Doch nun genug der Vorrede, jetzt soll es endlich losgehen!

1 Bedienungsanleitung zu Hi-Eddi plus

Hi-Eddi plus ist ein High-Resolution-Graphic-Editor, der eine Vielzahl von Befehlen und Features zum komfortablen Erstellen von Zeichnungen aller Art zur Verfügung stellt. Technische Zeichnungen, elektrische Schaltpläne, Lagepläne, Diagramme, Programmlaufpläne oder Grundrisse sind mit Hi-Eddi plus ebenso perfekt zu erstellen wie Glückwunschkarten oder Briefköpfe.

Aber auch ganz andere Anwendungsmöglichkeiten, wie zum Beispiel Schaufensterwerbung (siehe Walk-Befehl), sind denkbar. Ihrer Phantasie setzt Hi-Eddi plus keine Grenzen!

Da man am besten durch Probieren lernt, sollten Sie das in diesem Buch Erklärte immer gleich am Computer ausprobieren. Laden Sie dazu das Programm Hi-Eddi plus von der beiliegenden Diskette mit 'LOAD"HI-EDDI+",8', und starten Sie es mit 'RUN'. Nachdem der Vorspann mit dem Titelbild abgelaufen und der Rest des Programms nachgeladen ist, stellt Hi-Eddi plus einige Fragen zur Betriebsart. Hi-Eddi plus kann nämlich in drei verschiedenen Betriebsarten arbeiten:

1.1 Betriebsartenwahl

Der Video-Controller im Commodore 64 - kurz VIC genannt - kennt zwei verschiedene Möglichkeiten, hochauflösende Grafiken darzustellen: den High-Resolution-Modus mit einer Auflösung von 320 Punkten horizontal und 200 Punkten vertikal sowie den Multicolormodus mit nur 160*200 Punkten, also der halben Auflösung. Dafür bietet der Multicolormodus - wie der Name schon sagt - erweiterte Möglichkeiten bei der Farbgebung der Grafiken. Dieser Modus wird deshalb von den *bunten* Malprogrammen, zum Beispiel *Paint Magic*, angewendet. Hi-Eddi plus ist jedoch konsequent für Zeichnungen in maximal möglicher Auflösung konzipiert und benutzt deshalb den High-Resolution-Modus. Die Farbmöglichkeiten sind somit gegenüber einem *bunten* Malprogramm zwar eingeschränkt, aber dennoch nicht auf bloße *Schwarzweiß*-Darstellung reduziert!

Normalerweise werden auf dem Bildschirm 25 Zeilen zu je 40 Zeichen dargestellt. Die Auflösung im Text- oder Low-Resolution-Modus beträgt also 40*25. Vergleicht man das mit der Auflösung des High-Resolution-Modus von 320*200 Punkten, so stellt man fest, daß ein Zeichen im Low-Resolution-Modus einem Feld von 8*8 Punkten im High-Resolution-Modus entspricht, da 40*8=320 und 25*8=200 ist.

Was nun die Farbgebung im High-Resolution-Modus betrifft, so besteht hier ein Zusammenhang mit der normalen Auflösung im Low-Resolution-Modus: In jedem 8*8-Punkte-Feld - entsprechend einem Zeichen im Low-Resolution-Modus, wie wir gerade gesehen haben - dürfen nur zwei verschiedene Farben vorkommen. So ist es zum Beispiel nicht möglich, eine gelbe und eine rote Linie auf schwarzem Hintergrund durch dasselbe 8*8-Punkte-Feld zu legen. Das wären drei Farben in einem Feld!

Da jedoch der Bildschirm aus insgesamt 40*25, also 1000 solchen Feldern besteht und in jedem Feld zwei beliebige Farben aus 16 möglichen vorkommen dürfen, kann das Ganze doch noch sehr bunt werden. Für Bilder zu einem selbst programmierten Adventure-Spiel oder für einen farbigen Trickfilm reicht das durchaus!

Allerdings hat diese Farbigkeit auch einen Nachteil: Die Farbinformationen müssen im Rechner abgelegt werden und verbrauchen somit Speicherplatz. Verzichtet man auf die Farbe, so steht mehr Platz für die Bilder selbst zur Verfügung und man bekommt mehrere Bitmaps - das sind die Speicherbereiche für die hochauflösende Grafik - in den Computer.

Hi-Eddi plus überläßt die Wahl, ob man lieber Farbe oder mehrere Bilder möchte, dem Benutzer und stellt dazu verschiedene Betriebsarten zur Auswahl: eine Farb-Betriebsart, in der sechs Bildschirme mit den oben geschilderten Farbmöglichkeiten zur Verfügung stehen, sowie eine Schwarzweiß-Betriebsart, die sieben Bildschirme mit nur zwei Farben bietet.

Sie werden vielleicht fragen, wozu man so viele Bildschirme braucht. Ich bin mir sicher, wenn Sie erst einmal damit gearbeitet haben, werden Sie nicht mehr darauf verzichten wollen. Die Anwendungsmöglichkeiten sind vielfältig:

* Abspeichern von *Zwischenstadien* der Werke, bei denen man wieder ansetzen kann, wenn etwas schiefgegangen ist.

* Arbeiten mit Construction Sets: Aus einem Vorrat von Symbolen, zum Beispiel für elektrische Schaltungen, holen Sie sich die benötigten Elemente und bauen daraus auf einem anderen Bildschirm Ihren Schaltplan auf.

* Erstellen von Superhardcopys: Durch den zusammenhängenden Ausdruck mehrerer Bilder neben- und übereinander können Sie beliebig große Pläne erstellen, wobei die Auflösung des Druckers voll erhalten bleibt.

* Erstellen von Zeichentrickfilmen oder Schaufenster-Werbung mittels Walk-Befehl.

Man wird deshalb die Schwarzweiß-Betriebsart vor allem dann anwählen, wenn man viele Bildschirme möchte. Um diese Betriebsart anzuwählen, geben Sie auf die Frage **FARBE** ein **N** für **Nein** ein.

Neben den beiden bisher beschriebenen Betriebsarten gibt es noch eine dritte, die eigentlich nur eine Unterteilung der Farb-Betriebsart darstellt. Die entsprechende Frage, nämlich **MENUE**, erscheint deshalb auch nur, wenn Sie auf die Frage **FARBE** mit **J** geantwortet haben.

Diese dritte Betriebsart, die Menü-Betriebsart, bezieht sich auf die Form der Befehlseingabe: In den ersten beiden Betriebsarten erfolgt die Befehlseingabe über die Tastatur, zum Beispiel durch ein **L** für Line (Linien zeichnen) oder ein **C** für Circle (Kreise zeichnen). Diese Eingabeart ist zwar schnell und unkompliziert, setzt aber voraus, daß man die Befehle im Kopf hat, anderenfalls muß man ständig in der Bedienungsanleitung nachblättern.

Als Alternative gibt es die Menü-Betriebsart, in der alle Befehle in einer übersichtlichen, farbigen und illustrierten Menütafel dargestellt sind. Um einen Befehl einzugeben, muß man zunächst die Menütafel auf den Bildschirm holen - das geschieht durch Drücken der Leertaste (Space) - dann den Cursor auf den gewünschten Befehl steuern und die Return-Taste oder den Knopf am Joystick oder Pad drücken. Hi-Eddi plus schaltet dann automatisch auf den zuletzt angewählten Bildschirm zurück und führt den Befehl aus. Diese Art der Eingabe ist zwar etwas langsamer, aber sicher sehr nützlich, solange Sie die Befehle noch nicht auswendig kennen. Allerdings hat diese Betriebsart, die Sie übrigens durch ein **J** auf die Frage **MENUE** anwählen können, einen Nachteil: Da die Menütafel einen eigenen Bildschirm belegt, bleiben Ihnen nur noch 5 Bildschirme übrig.

Jedoch hat die Menüeingabe auch Vorteile: Auf diese Weise können die Farben, die bei der Tastatureingabe nur eine nach der anderen durchgeschaltet werden, direkt angewählt werden. Auch andere Befehle, die bei der Tastatureingabe das Betätigen mehrerer Tasten erfordern, sind im Menü als einzelner Befehl vorhanden.

Eine Menüeingabe im Schwarzweiß-Betrieb ist nicht vorgesehen, da erstens der Vorteil der direkten Farbanwahl entfällt (das Menü könnte nicht mehrfarbig sein) und zweitens der Schwarzweiß-Modus viele Bildschirme zur Verfügung stellen soll, was jedoch dem für die Menütafel nötigen Speicherplatzbedarf widerspricht.

Übrigens bleibt die Tastatureingabe neben der Menüeingabe voll erhalten. Sie können also zum Beispiel die Befehle, die Sie schon gelernt haben, direkt über die Tastatur eingeben, während Sie für die, bei denen Sie noch nicht sicher sind, die Menütafel zu Rate ziehen können.

Nachdem Sie sich für eine Betriebsart entschieden haben, erscheint die Frage **LOESCHEN**. Hier werden Sie in der Regel **J** eingeben, was bewirkt, daß alle Bildschirme gelöscht werden und Sie auf *sauberem* Hintergrund zeichnen können. Falls Sie jedoch zuvor mit einem anderen Grafikprogramm oder einer BASIC-Erweiterung gearbeitet haben und das damit erstellte Bild mit Hi-Eddi plus weiterbearbeiten möchten, ohne es erst auf Diskette abzuspeichern, beantworten Sie die Frage **LOESCHEN** mit **N**. Das Bild bleibt dann im Speicher und ist in einem der Bildschirme im Hi-Eddi plus zu finden; in den anderen werden Sie dagegen nur unbrauchbare Daten vorfinden. Allerdings müssen Sie eine BASIC-Erweiterung o.ä. abschalten, bevor Sie Hi-Eddi plus laden. Dazu dürfen Sie den Computer natürlich nicht ausschalten, da sonst das Bild verloren wäre!

1.2 Cursorsteuerung

Nachdem Sie nun alle Fragen beantwortet haben, erscheint der erste Bildschirm oder, falls Sie die Menü-Betriebsart gewählt haben, die Menütafel, die allerdings erst von der Diskette nachgeladen wird. In jedem Fall werden Sie in der Mitte des Bildschirms ein blinkendes Kreuz sehen. Das ist der Zeichencursor, der Ihnen immer anzeigt, wo Sie sich auf dem Bildschirm gerade befinden.

Der Zeichencursor kann wahlweise mit einem Joystick oder einem Koala-Pad gesteuert werden; beides muß in Port 2 stecken. Hi-Eddi plus erkennt selbst, was eingesteckt ist. Die beiden Tasten am Pad haben die gleiche Funktion wie der Feuerknopf am Joystick und die RETURN-Taste.

Die Koala-Pad-Steuerung ist dann überlegen, wenn es darum geht, ungleichmäßige Linien schnell zu zeichnen (zum Beispiel Handschrift). Das Besondere an der Pad-Steuerung ist ihre Auslegung als relative Steuerung: Der Cursor reagiert nur auf Bewegung des Zeichenstiftes über das Pad! Außerdem deckt das Pad mit seiner kleinen Zeichenfläche nicht den ganzen Bildschirm ab, sondern nur ca. 1/6 (120*120 Punkte). Dies hat den Vorteil, daß Sie äußerst präzise zeichnen können. Daß Sie dennoch überall hinkommen, liegt an der relativen Steuerung: Um zum Beispiel vom linken zum rechten Rand zu gelangen, fahren Sie dreimal über das Pad. Aber nicht zu schnell, zu hastigen Bewegungen folgt der Cursor nämlich nicht! Damit werden die von anderen Programmen bekannten und gefürchteten *Ausrutscher* unterdrückt! Falls Sie die Koala-Pad-Steuerung von anderen Programmen, bei denen das Pad den ganzen Bildschirm abdeckt, schon gewohnt sind, wird Ihnen die relative Steuerung des Hi-Eddi plus zunächst einige Schwierigkeiten bereiten. Mit etwas Übung werden Sie jedoch mit dem Hi-Eddi plus besser zeichnen können als mit den absoluten und daher unpräzisen Steuerungen anderer Programme.

Die Joystick-Steuerung ist von Vorteil, wenn es darum geht, den Cursor exakt horizontal oder vertikal zu bewegen, also in der Regel beim Zeichnen von technischen Skizzen und Plänen. Die Besonderheit hierbei ist der beschleunigende Cursor: Er ist langsam genug, um durch Antippen des Joysticks pixelweise zu rangieren; bei längeren Strecken wird er jedoch ohne lästiges Umschalten schneller. Durch Drücken der Minus-Taste kann die Endgeschwindigkeit herabgesetzt werden, so daß der Cursor dann fast nicht mehr beschleunigt. Die Plus-Taste stellt wieder den Einschalt-Zustand her.

Beim Koala-Pad bewirkt die Minus-Taste eine Halbierung der Pad-Empfindlichkeit; das Pad deckt dann nur noch eine Fläche von ca. 60*60 Punkten ab. Damit ist pixelgenaues Rangieren noch leichter und das gelegentlich auftretende *Zittern* des Cursors wird vollständig unterdrückt.

Der Cursor kann auch über die Cursortasten gesteuert werden, die dafür geltenden Besonderheiten wie beispielsweise die programmierbare Schrittweite werden in Kapitel 1.10 erklärt.

1.3 Die Zeichenmodi

Nun kommen wir zur Beschreibung der zahlreichen Befehle von Hi-Eddi plus, die sich in zwei Gruppen einteilen lassen:

1) die Modus-Befehle, die einen Betriebsmodus anwählen und bestimmen, was bei einem Druck auf den Feuerknopf am Joystick (oder auf die Return-Taste oder auf eine der Tasten am Koala-Pad) geschieht.

2) die direkten Befehle, die nur eine unmittelbare Wirkung haben und den aktuellen Modus nicht verändern.

Die folgenden Befehle zum Zeichnen von Punkten, Linien, Rechtecken, Kreisen, Ausfüllen von Flächen und zum Sprühen sind Modus-Befehle. Ich nenne sie deshalb die *Zeichenmodi.* In der Menütafel sind diese Zeichenmodi in dem gelben Feld links oben zusammengefaßt. Sie geben die Befehle ein, indem Sie den entsprechenden vor dem Befehl angegebenen Buchstaben eintippen oder wenn Sie, nachdem Sie die Menübetriebsart gewählt haben, den Cursor im Menü auf das entsprechende Feld setzen und den Knopf drücken. Mit **Knopf** meine ich im folgenden immer den Feuerknopf des Joysticks, die Tasten am Pad oder die Return-Taste.

D **Draw - Freihändig zeichnen**
SHIFT D **Zeichnen mit dickerem Pinsel**

Im Draw-Modus kann mit dem Cursor *freihändig* gezeichnet werden. Wenn Sie den Knopf drücken wird an der Stelle, an der sich der Cursor befindet, ein Punkt gesetzt. Bewegen Sie den Cursor bei gleichzeitig gedrücktem Knopf, so hinterläßt er eine Spur. Um Punkte zu löschen, muß zusätzlich zum Knopf noch die SHIFT-Taste gedrückt werden. Wollen Sie einen einzelnen Punkt löschen, so setzen Sie den Cursor darauf, drücken die SHIFT-Taste und zugleich den Knopf. Da es beim Löschen mehrerer Punkte sehr umständlich wäre, die SHIFT-Taste ständig niedergedrückt zu halten, benutzen Sie dafür die SHIFT-LOCK-Taste, die Sie nur einmal zu drücken brauchen. Sie bleibt dann eingerastet und Sie können nun mit dem Cursor, bei zugleich gedrücktem Knopf, Punkte löschen.

Wenn Sie mit dem Löschen fertig sind, rasten Sie die SHIFT-LOCK-Taste wieder aus. Das ist wichtig, da viele Befehle die SHIFT-Taste ebenfalls benutzen und es daher bei einer eingerasteten SHIFT-Taste, die man leicht übersieht, zu einer Fehlbedienung kommen kann.

Ein Beispiel dafür ist bereits der nächste Befehl, nämlich **SHIFT D**: Wenn Sie zum **D** gleichzeitig die SHIFT-Taste drücken, wird ein Modus angewählt, in dem Sie ebenfalls freihändig zeichnen können, jedoch mit einem dickeren Pinsel. Bei Kopfdruck werden nun gleichzeitig 3*3 Punkte gesetzt bzw. bei Knopfdruck und gedrückter SHIFT-Taste 3*3 Punkte gelöscht.

Sie sehen also, daß die SHIFT-Taste zwei verschiedene Funktionen hat: Zusammen mit einer anderen Taste dient sie zum Anwählen verschiedener Befehle; zusammen mit dem Knopf hat sie die Funktion, einen bereits angewählten Modus weiter zu differenzieren wie beispielsweise in Punkte zu setzen oder zu löschen.

Apropos Löschen: Wenn Sie nun das bisher Beschriebene ausprobieren, werden Sie feststellen, daß es recht schwierig ist, mit dem dünnen Pinsel zu löschen, da man das zu Löschende sehr schlecht trifft.

Leichter geht es mit dem dicken Pinsel. Hi-Eddi plus bietet jedoch noch eine Reihe weiterer Möglichkeiten, um etwas wegzuradieren. Darauf werde ich bei der Erklärung der weiteren Befehle näher eingehen.

Der Zeichenmodus mit dem dicken Pinsel eignet sich übrigens besonders gut zum Setzen der Verbindungspunkte in elektrischen Schaltungen. Pinsel in beliebiger Form lassen sich im Append-Modus definieren, doch dazu später.

Wenn durch das viele Probieren der Bildschirm bereits ausgefüllt ist, können Sie ihn mit **C= CLR/HOME** löschen. Das **C=** steht für die Commodore-Taste ganz links unten auf der Tastatur. Sie müssen zum Löschen des Bildschirms diese Taste niederdrücken und zugleich die Taste **CLR/HOME** betätigen. Ich habe hier absichtlich nicht die im BASIC übliche und auch vom alten Hi-Eddi her gewohnte Eingabe von **SHIFT CLR/HOME** gewählt, da dies unangenehme Folgen haben könnte: Mit **CLR/HOME** setzt man nämlich im Hi-Eddi plus den Cursor in die linke obere Bildschirmecke (diese Position nenne ich deshalb HOME-Position). Will man dies tun, und die SHIFT-LOCK-Taste ist noch eingerastet, wird der gesamte Bildschirm gelöscht. Bei **C= CLR/HOME** ist diese fatale Fehlbedienung dagegen nicht möglich.

L Line - Linien ziehen

In diesem Modus wird mit dem ersten Knopfdruck der Anfangspunkt und mit dem zweiten der Endpunkt einer Linie festgelegt. Nach dem ersten Knopfdruck erscheint ein *Merkcursor*, der Ihnen den Anfangspunkt anzeigt, während Sie den Endpunkt markieren. Nach dem zweiten Knopfdruck wird die Linie gezeichnet. Mit dem dritten Knopfdruck wird dann wieder ein Anfangspunkt markiert, mit dem vierten ein Endpunkt und so weiter. Um einen zusammenhängenden Linienzug zu zeichnen, müssen Sie also an jedem Knickpunkt zweimal drücken.

Um Strahlen zu zeichnen, drücken Sie, nachdem Sie eine Linie gezeichnet haben, die Funktionstaste F7. Der Cursor springt dann zurück an den Anfangspunkt der zuletzt gezeichneten Linie. Sie können dort nun durch Knopfdruck den Anfangspunkt der nächsten Linie setzen. Auf der Funktionstaste F7 wird nämlich immer der Anfangspunkt der zuletzt gezeichneten Linie gespeichert. Auch bei anderen Befehlen, wie beim Zeichnen von Rechtecken oder Kreisen, wird der erste Knopfdruck auf dieser Taste gespeichert.

Eine solche Position, an die man per Tastendruck zurückkehren kann, nennt man Tabulator. Genaueres darüber erfahren Sie in einem späteren Kapitel, bei der Beschreibung der Funktionstasten.

Das Löschen von Linien geht analog zum Draw-Befehl, indem Sie während des Zeichnens der Linie, also beim zweiten Knopfdruck, die SHIFT-Taste drücken. Allerdings ist es, ebenso wie bei **Draw** mit dünnem Pinsel, sehr schwierig, eine Linie genau zu löschen. Das **Löschen** einer Linie mittels SHIFT-Taste ist jedoch dann sinnvoll, wenn Sie in einer bereits ausgefüllten Fläche eine Linie zeichnen wollen.

R Rectangle - Rechtecke zeichnen

Mit dem ersten Knopfdruck legen Sie einen beliebigen Eckpunkt des Rechtecks fest, der zweite Knopfdruck gibt die dazu diagonal liegende Ecke an. Was die SHIFT-Taste betrifft, gilt das bereits bei **Line** Gesagte. Das gleiche gilt für F7. Auf dieser Taste wird auch bei **Rectangle** der erste Eckpunkt gespeichert.

Besonders schnell läßt sich mit diesem Befehl ein Rahmen um das ganze Bild zeichnen. Setzen Sie dazu erst den Cursor in die linke obere Bildschirmecke - durch Antippen der **CLR/HOME**-Taste geht es am schnell-

sten, wie Sie bereits wissen - und drücken Sie den Knopf. Um nun schnell in die rechte, untere Bildschirmecke zu kommen, drücken Sie die Funktionstaste F1. Auch diese Taste dient, ebenso wie F7, zum Anspringen einer Tabulatorposition, und im Einschaltzustand ist auf F1 die rechte untere Bildschirmecke als Tabulator gespeichert. Dort drücken Sie nun zum zweiten Mal den Knopf, und schon wird der Rahmen gezeichnet.

C Circle - Kreise Zeichnen

Der erste Knopfdruck ergibt den Mittelpunkt, der zweite einen beliebigen Randpunkt des zu zeichnenden Kreises. Auch hier gilt für SHIFT und F7 wieder das gleiche wie bei *Line*. Durch das Speichern des ersten Knopfdruckes, also des Mittelpunktes, auf F7 können leicht konzentrische Kreise gezeichnet werden. Die Vorgehensweise entspricht der beim Zeichen von Strahlen.

Sollen mehrere exakt gleich große Kreise gezeichnet werden, so empfiehlt es sich, den Radius als Schrittweite zu speichern. Wie das geschieht, erfahren Sie bei den Befehlen H und V.

Es können auch Kreise gezeichnet werden, die nicht ganz auf den Bildschirm passen. Allerdings darf der Radius nicht größer als 256 Punkte werden!

P Paint - Ausfüllen begrenzter Flächen

Um begrenzte Flächen auszumalen, genügt es, den Paint-Modus anzuwählen, den Cursor irgendwo auf die auszufüllende Fläche zu setzen und den Knopf zu drücken. Sollte durch ein Loch in der Umrandung der ganze Bildschirm ausgefüllt werden, so kann der Vorgang durch Drücken der STOP-Taste abgebrochen werden. Dazu ist allerdings eine gute Reaktion nötig! Meist ist es schon zu spät, wenn man ein Loch übersehen oder die falsche Fläche ausgemalt hat. Ich werde Sie deshalb an anderer Stelle noch daran erinnern, sich Sicherheitskopien Ihrer Bilder zu machen, bevor Sie diesen Befehl anwenden. Dazu kommen uns dann die vielen Bildschirmspeicher des Hi-Eddi plus zugute.

Die SHIFT-Taste hat hier eine andere Funktion als bei den vorangegangenen Befehlen. Bei gedrückter SHIFT-Taste wird die Fläche nicht vollständig, sondern mit einem Raster aufgefüllt. Falls Sie einen Farbfernseher als

Sichtgerät benutzen, werden Sie in diesem Raster möglicherweise schwache, senkrechte Farbbalken erkennen. Das liegt nicht am Programm und auch nicht daran, daß Ihr Fernseher schlecht ist, sondern ist systembedingt: Die Auflösung des Computers deckt sich nicht mit der des Fernsehers.

Auch das Löschen beliebig großer, zusammenhängender Flächen ist leicht möglich. Sie invertieren dazu zunächst den Bildschirm durch Eingabe von I. Aus der zu löschenden Fläche ist jetzt ein *Loch* geworden, das Sie, wie oben beschrieben, auffüllen. Anschließend wird der Bildschirm durch nochmalige Eingabe von I zurückinvertiert und die Fläche ist verschwunden.

J Jots - Pünktchen versprühen

Mit dieser Funktion wird eine Spraydose simuliert. Je länger an einer Stelle der Knopf gedrückt wird, desto dichter werden die Punkte. Bei gedrückter SHIFT-Taste werden Pünktchen gelöscht. Das ist vor allem zum *Auflockern* nötig, wenn man an einer Stelle zu dicht gesprüht hat.

Da die Pünktchen beim Sprühen zufällig gesetzt werden, eignet sich diese Funktion hervorragend, um Bildern den spezifischen *Computercharacter* zu nehmen. So kann zum Beispiel Rasen mit Blümchen übersät oder Mauern können *angenagt* werden und es lassen sich fließende Farbübergänge herstellen.

1.4 Bildschirmverwaltung

Die folgenden direkten Befehle dienen dazu, zwischen den Bildschirmen umzuschalten - uns stehen bekanntlich sieben Schwarzweiß- oder sechs farbige Bildschirme zur Verfügung -, sie zu kopieren, zu löschen oder zu invertieren.

1 bis 7 - Bildschirm anwählen

Durch Eingabe einer Nummer wird der entsprechende Bildschirm angewählt, d.h. er wird auf den Monitor geholt. Im Schwarzweiß-Betrieb sind alle Nummern von 1 bis 7 möglich. Im Farb-Betrieb gibt es nur die Bildschirme 1 bis 6. In der Menü-Betriebsart schließlich sind nur die Nummern 2 bis 6 verfügbar, da in Speicher 1 die Menütafel liegt. Bildschirm 1 kann zwar angewählt werden, aber hier ist Vorsicht geboten: Holt man die Menütafel mittels Space-Taste auf den Bildschirm, so befindet man sich im Menü-Modus und bei einem Knopfdruck wird der Befehl, auf dem der Cursor steht, ausgeführt. Wählt man die Menütafel dagegen über die Taste 1 an, so wird dieser Bildschirm wie jeder andere behandelt und bei einem Knopfdruck erfolgt die dem aktuellen Modus entsprechende Aktion. Zum Beispiel wird ein Punkt gesetzt, wenn Sie sich im Draw-Modus befinden. Damit können Sie natürlich die Menütafel nach Ihren eigenen Vorstellungen ändern; andererseits kann jedoch die Funktion der Menütafel gestört werden. Was man beim Ändern oder Erstellen der Menütafel beachten muß, erfahren Sie in Kapitel 1.13. Im Menü ist die Anwahl des Bildschirms 1 gar nicht möglich, um dadurch Fehler zu vermeiden.

C=1 bis C=7 Copy - Bildschirme kopieren

Das C= steht für die Commodore-Taste rechts unten auf der Tastatur, die von Hi-Eddi plus, ähnlich wie die SHIFT-Taste, für die Eingabe von Befehlen verwendet wird. Wird diese Taste zusammen mit einer Nummer gedrückt, so wird damit der Bildschirm mit der angegebenen Nummer in den aktuellen, also sichtbaren Bildschirmspeicher kopiert.

Sie sollten sich immer in bestimmten Abständen und besonders vor *riskanten* Eingriffen eine Kopie Ihrer Arbeit machen, um dort wieder ansetzen zu können, falls etwas schiefgegangen ist. Wenn Sie zum Beispiel auf Bildschirm 1 zeichnen, dann schalten Sie zwischendurch auf 2 um, drücken dort

gleichzeitig die Commodore-Taste und die **1** und schalten durch nochmaliges Drücken der **1** wieder auf Ihren *Arbeits-Bildschirm* zurück. Wollen Sie dann auf die Sicherheitskopie zurückgreifen, drücken Sie einfach die Commodore-Taste und **2**.

Falls Sie sich nicht merken können, ob beim Kopieren nun die aktuelle Bildschirmseite in die angewählte oder umgekehrt die angewählte in die aktuelle kopiert wird, so hilft Ihnen sicherlich die folgende Grundregel, die bei der Konzeption von Hi-Eddi plus zugrunde gelegt wurde: Es wird immer nur der aktuelle, sichtbare Bildschirm beeinflußt, die unsichtbaren Bildschirme bleiben unangetastet. Daraus folgt, daß der angewählte Bildschirm in den sichtbaren kopiert wird und somit der alte Inhalt des sichtbaren Bildschirms verloren geht.

C= CLR/HOME Clear - Bildschirm löschen

Die Pixels des Bildschirms werden gelöscht, nicht dagegen die Farben (im Farb-Betrieb). Dafür sind die Farbbefehle zuständig, die in Kapitel 1.8 behandelt werden.

I Invert - Bildschirm invertieren

Die Pixels des Bildschirms werden invertiert, d.h., die gesetzten Pixels werden gelöscht und umgekehrt. Dies ergibt besonders im Zusammenhang mit anderen Befehlen eine Vielzahl von Möglichkeiten. So können Sie damit zum Beispiel den Paint-Befehl, wie bereits beschrieben, zum Löschen verwenden. Auch kann dieser Befehl im Zusammenhang mit dem Print-Befehl sinnvoll sein, wenn Sie eine inverse Hardcopy möchten.

1.5 Verschieben, Verknüpfen, Spiegeln

Hi-Eddi plus bietet eine ganze Reihe von Befehlen zum Verschieben, logischen Verknüpfen, Spiegeln und Drehen von Bildschirmbereichen oder ganzen Bildern.

M Move - Verschieben von Bildschirmbereichen

Bei **M** handelt es sich um einen Modus-Befehl. Mit dem ersten und zweiten Knopfdruck im Move-Modus werden, ähnlich dem Befehl Rectangle, zwei beliebige, diagonale Ecken des zu transportierenden Bereichs gesetzt. Nach dem 2. Knopfdruck erscheint eine farbliche Markierung dieses Bereichs. Der 3. Knopfdruck gibt die linke, obere Ecke des Zielbereichs an. Der Zielbereich muß noch ganz auf den Bildschirm passen (sonst wird der Knopfdruck nicht akzeptiert), er darf jedoch den Quellbereich überlappen oder auch in einem anderen Bildschirm liegen. Sie können also zum Beispiel einen Bereich aus Bild 3 in Bild 5 transportieren. Dazu müssen Sie lediglich nach dem zweiten Knopfdruck auf Bildschirm 5 umschalten.

Der verschobene Ausschnitt überdeckt dabei den Zielbereich. Es gibt jedoch auch Möglichkeiten, den Zielbereich mit dem verschobenen Ausschnitt zu verknüpfen. Dieses wird bei den Befehlen **O**,**X** und **U** erklärt.

Sie werden bald merken, daß sich der Quellbereich nicht pixelgenau festlegen läßt. Grund dafür ist die bereits bei der Betriebsartenwahl beschriebene beschränkte Farbauflösung des Video-Controllers. Quell- und Zielbereich lassen sich nur entsprechend der Farbauflösung von 40*25 Feldern (zu je 8*8 Punkten) festlegen. Wie man dennoch zum Beispiel die Blöcke eines Planes nahtlos aneinanderfügen kann, erfahren Sie in Kapitel 2 bei den Anwendungen.

Pfeil nach oben - Recall Move

Soll ein Bereich mehrmals kopiert werden, muß er nicht jedesmal neu markiert werden. Ein Druck auf die Pfeil-nach-oben-Taste schaltet den Move-Modus ein, sofern er nicht schon aktiv ist, und holt die letzte Markierung wieder auf den Bildschirm. Danach kann sofort der Zielbereich bestimmt werden.

Pfeil nach links - Korrektur

Hat man sich beim Markieren vertan, so löscht ein Druck auf die Pfeil-nach-links-Taste die Markierung wieder (gilt übrigens auch für L,R,C). Dabei ist es gleichgültig, ob die Korrektur nach dem ersten oder zweiten Knopfdruck vorgenommen wird.

O Oder-Verknüpfung
X Exor-(Exclusiv-Oder-)Verknüpfung
U Und-Verknüpfung

Nun kommen wir zu einer Gruppe sehr leistungsfähiger Befehle, zu deren Erklärung ich jedoch etwas ausholen muß.

Wie Sie vielleicht schon wissen, gibt es in der Logik nur zwei mögliche Zustände, nämlich **0** und **1**. Übertragen auf die Grafik kann man nun die **0** als gelöschten und die **1** als gesetzten Punkt betrachten. So lassen sich Bilder oder Ausschnitte nach den Regeln der Boolschen Algebra verknüpfen. Zur Veranschaulichung diene uns Abb.1.1, in der die möglichen Verknüpfungen eines Balkens und eines Ringes dargestellt sind:

* Bei der **Oder**-Verknüpfung ist ein Punkt im Ergebnis gesetzt, wenn in einem oder im anderen zu verknüpfenden Gebilde ein Punkt gesetzt ist.

* Bei der **Und**-Verknüpfung dagegen ist ein Punkt nur gesetzt, wenn er im ersten und zweiten Gebilde gesetzt ist.

* Bei der **Exclusiv-Oder**-Verknüpfung schließlich ist ein Punkt dann gesetzt, wenn er entweder im ersten oder im zweiten Gebilde gesetzt ist, nicht jedoch, wenn er in beiden gesetzt ist.

Die **Oder**-Verknüpfung entspricht also einer Überlagerung. Man wird sie benutzen, wenn man zwei Bilder oder Ausschnitte wie zwei Folien übereinanderlegen möchte. Die **Und**-Verknüpfung wird zum *Maskieren* benutzt, um zum Beispiel von einem Gebilde unerwünschte Teile wegzuschneiden. Bei der **Exor**-Verknüpfung ist vor allem die Eigenschaft interessant, daß sich zweimalige Verknüpfung mit demselben Gebilde aufhebt. Bei der Verknüpfung des Ringes mit dem Balken in Abb.1.1 zum Beispiel erhalten wir das Gebilde in Abb.1.1 rechts unten. Verknüpft man dieses Gebilde nochmals mit dem Balken, so entsteht wieder der Ring, wie in Abb.1.2 dargestellt.

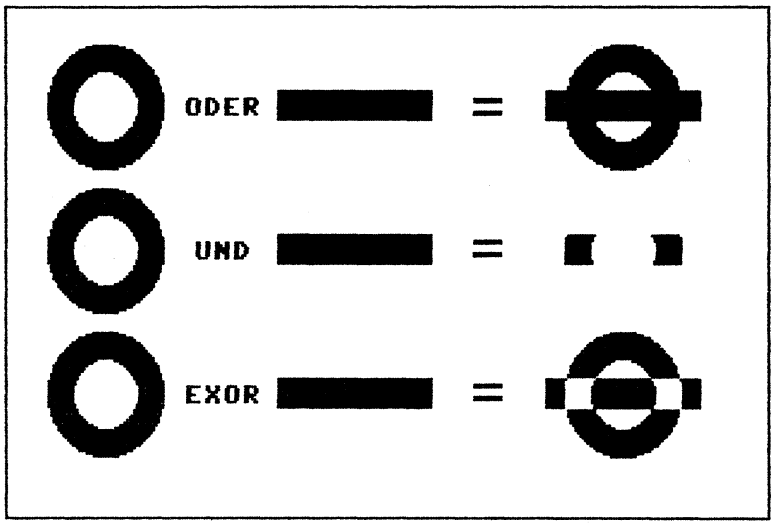


Bild 1.1: Die Verknüpfungen ODER, UND, EXOR

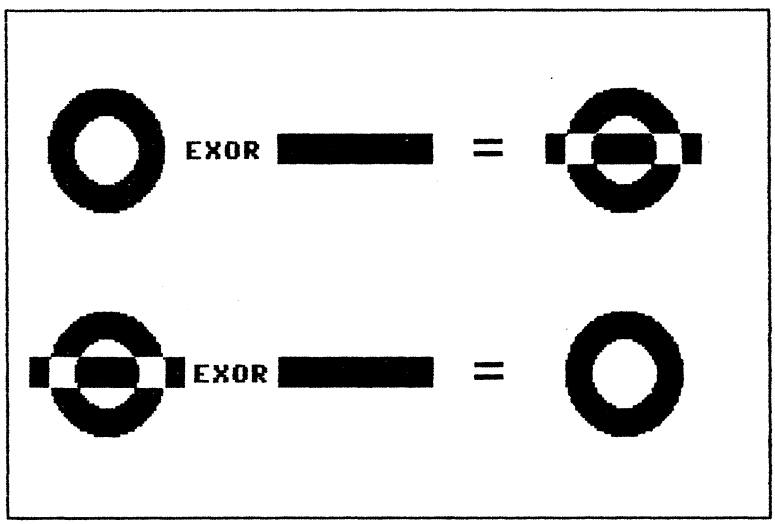


Bild 1.2: Die Wirkung zweimaliger EXOR-Verknüpfung

Eine weitere für uns interessante Eigenschaft ist die Verknüpfung eines beliebigen Gebildes mit einer Fläche gesetzter Punkte, wie in Abb.1.3 dargestellt. Sie erkennen sicherlich, daß das Gebilde invertiert wurde, was ebenfalls der Befehl **I** bewirkt. Allerdings kann das der Befehl **I** nur mit dem ganzen Bildschirm, während sich die Verknüpfungsbefehle auch auf beliebige Bereiche anwenden lassen.

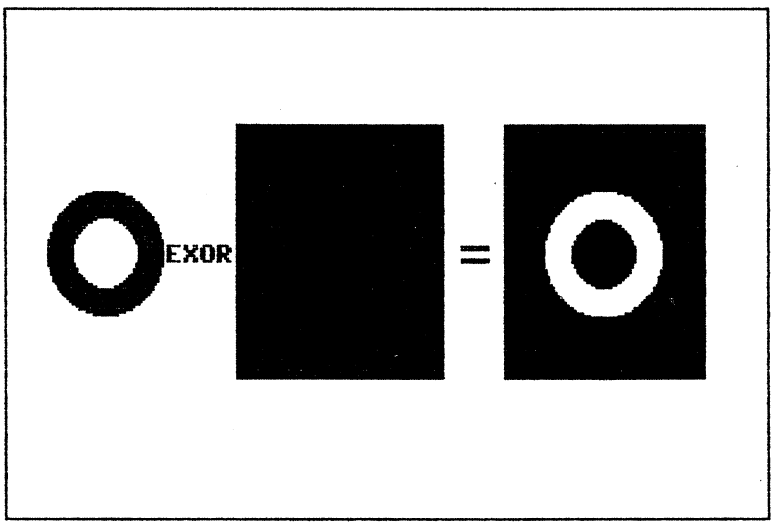


Bild 1.3: Die EXOR-Verknüpfung zum Invertieren

In Kapitel 2 werde ich eine Reihe von Anwendungsbeispielen zu den Verknüpfungsbefehlen aufführen. Sie werden dann sehen, daß es dabei eine Menge nützlicher und effektvoller Tricks gibt.

Doch nun genug der Vorrede; wir wollen als nächstes die Befehle einmal eingeben. Wie bereits erwähnt, können entweder Ausschnitte oder ganze Bildschirme verknüpft werden. Um letzteres zu erreichen geben Sie den gewünschten Buchstaben, also **O**, **U** oder **X** ein und anschließend die Nummer des Bildschirms, der mit dem sichtbaren verknüpft werden soll. Ein Beispiel: Sie haben Bildschirm 2 angewählt und wollen ihn mit Bildschirm 4 Exclusiv-Oder verknüpfen. Dazu geben Sie **X** und anschließend **4** ein. Nun können Sie auch gleich ausprobieren was passiert, wenn Sie dasselbe nochmals wiederholen: Es muß dann wieder der ursprüngliche Bildschirm 2 erscheinen, wie wir schon gelernt haben.

In der Menütafel ist für Bildschirmanwahl, Kopieren und Verknüpfen das große Feld rechts oben bestimmt. Dort setzen Sie den Cursor in die Zeile mit der gewünschten Verknüpfung und in die Spalte mit dem zu verknüpfenden Bildschirm und drücken den Knopf.

Zum Verknüpfen von Ausschnitten benötigen Sie den **Move**-Befehl. Bevor Sie durch den dritten Knopfdruck im Move-Modus den Zielbereich für eine Verschiebung festlegen, drücken Sie die gewünschte Verknüpfung (**O**, **U** oder **X**) oder wählen im Menü das entsprechende Feld im Block für Verschieben und Verknüpfen, rechts neben den Zeichenmodi, an.

Dazu nochmals ein Beispiel: Sie wollen einen Bereich verschieben, wobei das, was im Zielbereich zu sehen ist, nicht gelöscht, sondern überlagert werden soll.

Markieren Sie, nachdem Sie den Move-Modus angewählt haben, den Quellbereich, indem Sie zweimal den Knopf dürcken, positionieren Sie dann den Cursor auf die linke obere Ecke des Zielbereichs, drücken Sie die Taste **O** und zuletzt den Knopf.

Und nun noch eine Denksportaufgabe: Wie kann man rechteckige Flächen beliebiger Größe am schnellsten löschen? Die Lösung: Führen Sie eine Exclusiv-Oder-Verknüpfung dieses Bereichs mit sich selbst durch, d.h. Quell- und Zielbereich müssen identisch sein.

SHIFT M Mirror - Spiegeln
SHIFT T Turn - Drehen

Um einen ganzen Bildschirm zu spiegeln oder zu drehen, tippen Sie **SHIFT M** oder **SHIFT T** ein. Mit SHIFT M wird der gesamte Bildschirm zur Senkrechten gespiegelt (links oben wird zu rechts oben), und mit **SHIFT T** wird er um 180 Grad gedreht (links oben zu rechts unten). Beides ergibt eine Spiegelung zur Waagerechten (links oben zu links unten).

Ähnlich wie bei den Verknüpfungsbefehlen ist es auch bei beliebigen Ausschnitten. Dazu müssen diese Befehle unmittelbar nach einem Move-Vorgang (also nach dem dritten Knopfdruck) eingegeben werden. Es wird dann nicht mehr der ganze Bildschirm, sondern nur noch der soeben verschobene Bereich behandelt. Falls Sie einen Bereich auf der Stelle drehen oder spiegeln wollen, *verschieben* Sie ihn einfach auf der Stelle, also Quellbereich gleich Zielbereich.

1.6 Sprite-Befehle

In den Betriebsmodi, die durch die folgenden vier Modus-Befehle angewählt werden, erscheint ein spritegroßer Rahmen (24*21 Punkte) als Cursor. Ein Sprite ist ein bewegliches, grafisches Objekt, das im Hi-Eddi plus dazu verwendet wird, kleine Grafiken aufzunehmen, zu transportieren und beliebig oft auf den Bildschirm zu drucken. Es dient somit als Stempel oder Pinsel.

G Get Sprite - Sprite aus Bildschirm kopieren

Auf Knopfdruck wird der Bildschirmausschnitt, auf dem der Rahmen sitzt, in das Sprite hineinkopiert. Dieser Inhalt ist nun auch im Rahmen sichtbar, so daß ein genaues Positionieren sehr einfach ist. Der sichtbare Inhalt hat die Farbe des Bildschirmrahmens. Die Rahmenfarbe ist also so einzustellen, (siehe Farbbefehle), daß sich sowohl zum Hintergrund als auch zum Vordergrund ein guter Kontrast ergibt.

Anschließend geht Hi-Eddi plus automatisch in den Append-Modus, das Sprite kann an anderer Stelle wieder eingesetzt oder im Sprite-Editor bearbeitet werden.

A Append - Sprite in Bildschirm einfügen

Auf Knopfdruck wird der Sprite-Inhalt in den Bildschirm eingefügt, ohne jedoch den Bildschirmausschnitt vorher zu löschen (dies entspricht einer Oder-Verknüpfung). Sie können also den Sprite-Inhalt so oft Sie wollen auf das Bild *stempeln*. Allerdings können Sie das Sprite auch als Pinsel betrachten: Wenn Sie es bewegen und dabei den Knopf gedrückt halten, hinterläßt es eine Spur entsprechend der Form des Sprite-Inhalts.

Auf diese Art können Sie sich beliebig geformte Pinsel selbst erstellen. Auch das Löschen mit diesen Pinseln ist möglich; man verwendet dabei denselben Trick wie beim Paint-Befehl. Zunächst wird der Bildschirm mit I invertiert. Die zu löschenden Teile erscheinen nun als *Löcher*, die man mit dem Pinsel übermalt und anschließend wird der Bildschirm mit I zurückinvertiert. Wollen Sie zum Beispiel einen waagerechten, 15 Punkte breiten Streifen löschen, so zeichnen Sie zunächst einen mindestens 15 Punkte langen senkrechten Strich ins Sprite (siehe Erklärung des Sprite-Editors im

nächsten Kapitel). Dann wählen Sie den Append-Modus an, invertieren den Bildschirm und überfahren mit dem Strich wie mit einem Scheibenwischer den zu löschenden Streifen. Danach invertieren Sie den Bildschirm wieder.

Die Befehle **G** und **A** sind auch die wichtigsten bei der Arbeit mit Construction-Sets. Mit **G** holen Sie ein Element aus dem Set und anschließend setzen Sie es im Zielbildschirm ein. Den Append-Modus brauchen Sie dazu gar nicht anzuwählen, da dies nach einem Knopfdruck im Get-Modus automatisch geschieht. In Abb. 1.4 ist ein solches Construction-Set zu sehen - nämlich das Set zum Erstellen von Schaltplänen für analoge Schaltungen von der beiliegenden Diskette. Die genaue Vorgehensweise wird in Kapitel 2 anhand der auf der Diskette befindlichen Construction-Sets noch ausführlicher beschrieben.

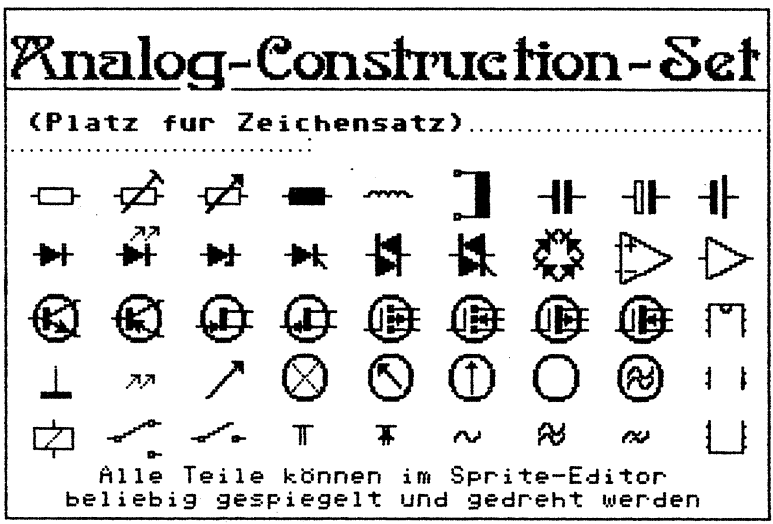


Bild 1.4: Das Analog-Constructions-Set auf der Diskette

S Stamp - Sprite auf Bildschirm "kleben"

Wie A, jedoch wird vor dem Einfügen der Untergrund gelöscht. Das Sprite wird wie eine Briefmarke auf den Bildschirm geklebt. Diesen Befehl wird man wohl nicht so häufig benötigen wie A.

E Erase - Löschen

Der Rahmen wird zum *Radiergummi*, der alles löscht, was er bei gedrücktem Knopf überfährt. Sollte der Erase-Radiergummi zu groß sein, so kann, wie bereits beim Append-Befehl beschrieben, auch mit einem kleineren, beliebig geformten Radiergummi gelöscht werden. Daneben gibt es die Möglichkeit, mit dem dicken Pinsel im Draw-Modus zu löschen sowie die bei Paint beschriebene Möglichkeit zum Löschen zusammenhängender Flächen und die bei Exor beschriebene Möglichkeit für Rechtecke. Sie sehen, es gibt die unterschiedlichsten Wege dabei.

1.7 Zoom-Funktion und Sprite-Editor

Wie bereits bei Append angedeutet, kann der Inhalt des Sprites in einem eigenen Sprite-Editor in vergrößerter Darstellung bearbeitet werden. In einer abgewandelten Form dient dieser Sprite-Editor auch als Zoom-Funktion, mit der Bildschirmausschnitte vergrößert und damit leichter und exakter bearbeitet werden können.

Aufgerufen werden sowohl Sprite-Editor als auch Zoom-Funktion mit der Space-Taste. Das gilt zumindest für Farb- und Schwarzweiß-Betriebsart. In der Menü-Betriebsart, in der ja die Space-Taste schon zum Aufruf des Menüs dient, werden Sprite-Editor und Zoom-Funktion mittels SHIFT-Space oder aber über das Menü aufgerufen - über das Feld in der Mitte, unter den Sprite-Befehlen.

Wie kann man nun zwischen Sprite-Editor und Zoom-Funktion auswählen, wenn beide in gleicher Weise aufgerufen werden? Was man tatsächlich aufruft, hängt vom gerade eingestellten Modus ab: Befindet man sich in einem der Zeichenmodi (Draw, Line, Circle, Rectangle, Paint und Jots), so wird zweckmäßigerweise die Zoom-Funktion aufgerufen. Dasselbe gilt für die Modi Move, Text, Fore und Back. Dabei wird die Umgebung der momentanen Cursorposition stark vergrößert dargestellt. Zum besseren Überblick erscheint daneben ein Teil des Bildschirms in Originalgröße, in dem der vergrößerte Ausschnitt farblich markiert ist. Alle Manipulationen, die Sie nun im vergrößerten Ausschnitt vornehmen, werden direkt in das Originalbild übertragen.

Aus den übrigen Modi, und das sind nur noch die vier Sprite-Modi Append, Get, Stamp und Erase, wird der Sprite-Editor aufgerufen, der, im Gegensatz zur Zoom-Funktion, keinerlei Einfluß auf den Grafikbildschirm hat. Im Sprite-Editor können Sie entweder ein Sprite bearbeiten, das Sie mittels Get (siehe vorigen Abschnitt) aus dem Bildschirm geholt haben, oder Sie können ein Sprite neu erstellen. Dabei haben Sie sogar zwei Sprites zur Verfügung, die Sie austauschen und verknüpfen können. Die so im Sprite-Editor erstellten Sprites können Sie anschließend im Grafik-Bildschirm mittels Append oder Stamp als Stempel oder Pinsel benutzen.

Kommen wir nun zu den Befehlen, die Ihnen im Sprite-Editor und in der Zoom-Funktion zur Verfügung stehen. Sprite-Editor und Zoom-Funktion sind gewissermaßen ein eigenes, in sich abgeschlossenes Programm inner-

halb des Hi-Eddi plus. Die Befehle des Grafik-Editors, zum Beispiel Draw, Line etc. sind hier nicht zugänglich, dafür jedoch die im folgenden beschriebenen.

Zunächst einmal können Sie Punkte setzen und löschen wie gewohnt - bei Knopfdruck werden Punkte gesetzt, bei SHIFT und Knopfdruck werden sie gelöscht. Den Cursor können Sie außer mit Joystick oder Pad auch mit den Cursortasten in gewohnter Weise bewegen. Nun zu den Befehlen in Sprite-Editor und Zoom-Funktion:

M **Mirror** - Das Sprite wird zur Senkrechten gespiegelt.

T **Turn** - das Sprite wird um 180 Grad gedreht. Mirror und Turn ergeben eine Spiegelung zur Waagerechten.

R **Rotate** - das Sprite wird um 90 Grad im Uhrzeigersinn gedreht. Das ist wichtig für Elemente aus Construction Sets, um Sie in die gewünschte Lage zu bringen. Da das Sprite 24 Punkte breit, aber nur 21 Punkte hoch ist, gehen die drei rechten Spalten verloren; außerdem ist zweimal **R** nicht dasselbe wie **T**.

G **Grid** - Zum besseren Abzählen von Punkten wird ein Schachbrettmuster eingeblendet, nach nochmaliger Eingabe von **G** verschwindet es wieder.

I **Invert** - Das Sprite wird invertiert.

SHIFT CLR/HOME **Clear** - Sprite löschen

B **Border** - Die Rahmenfarbe wird weitergeschaltet. Die übrigen Sprite-Editor-Farben entsprechen denen des Grafik-Bildes und können nur dort geändert werden.

Der Sprite-Editor arbeitet übrigens nicht nur in *Schwarzweiß*, sondern auch im Farb-Betrieb. In diesem Fall richten sich die Farben des Sprite-Editors nach der linken oberen Ecke des Bildschirms. Möchte man also die Farben des Sprite-Editors im Farb-Betrieb ändern, so muß man nur die Farben der linken oberen Bildschirmecke entsprechend einstellen (siehe nächstes Kapitel).

* Cursorblinken ein- und ausschalten.

C Change - Sprite wechseln
O,U,X Sprites verknüpfen

Im Sprite-Editor stehen, wie schon erwähnt, zwei Sprites zur Verfügung, zwischen denen mit **C** umgeschaltet wird und die sogar verknüpft werden können. So kann zum Beispiel ein häufig benötigtes Element aus einem Construction Set im zweiten Sprite untergebracht werden, von wo es bei Bedarf schneller geholt werden kann als aus dem Construction Set.

Die Verknüpfungen funktionieren ebenso wie die entsprechenden im Grafik-Editor.

Space Zurück zum Grafik-Editor.

1.8 Farbbefehle

F Foreground-Colormode - Vordergrund einfärben
B Background-Colormode - Hintergrund einfärben

Diese beiden Modus-Befehle sind nur wirksam, wenn Hi-Eddi plus im Farb-Betrieb arbeitet. In Abschnitt 1.1 über die Betriebsartenwahl wurde auch schon beschrieben, welche Farbmöglichkeiten Hi-Eddi plus bietet. Kurz zur Wiederholung: In jedem Feld von 8*8 Punkten - entsprechend einem der 40*25 Zeichen im Low-Resolution-Modus - dürfen nur zwei Farben, nämlich je eine für Vorder- und Hintergrund, vorkommen.

Bei den Befehlen **F** und **B** werden Sie nun eine weitere Besonderheit von Hi-Eddi plus kennenlernen, die dieses Programm von den bunten Multicolor-Malprogrammen unterscheidet. Während Sie bei den bunten Malprogrammen bereits beim Zeichnen - oder besser gesagt Malen - die Farbe, in der Sie zeichnen wollen, festlegen, sind Zeichnen und Einfärben bei Hi-Eddi plus zwei vollkommen getrennte Vorgänge. Sie zeichnen also zuerst etwas in Schwarzweiß und kolorieren es anschließend, wie in einem Malbuch! Das bedeutet zwar etwas mehr Arbeit, hat aber auch einen bedeutenden Vorteil: Sie können beliebige, bereits vorhandene Schwarzweiß-Bilder, zum Beispiel aus Dia-Shows oder mit einem Scanner oder Digitalisierer erstellt, nachträglich einfärben!

Nun zu den Befehlen **F** und **B**: Mit **F** wählen Sie den Foreground-Colormode an, also einen Modus, in dem auf Knopfdruck die gesetzten Punkte (=Vordergrund) des 8*8-Punkte-Feldes, auf dem sich der Cursor befindet, mit der momentanen Rahmenfarbe eingefärbt werden. Analog wird mit **B** der **Background-Colormode** aktiviert, der die gelöschten Pixels des 8*8 Punkte-Feldes einfärbt. Zusätzlich wird mit jeder Eingabe von **F** oder **B** die Rahmenfarbe weitergeschaltet, und zwar in der Reihenfolge, wie sie im Commodore-Handbuch angegeben ist.

Das klingt vielleicht ein wenig kompliziert, deshalb zur Verdeutlichung ein Beispiel: Wollen Sie die gesetzten Punkte, also den Vordergrund, gelb einfärben, so geben Sie so oft **F** ein, bis der Rahmen gelb ist. Jetzt können Sie per Knopfdruck die gesetzten Punkte des 8*8-Feldes, auf dem sich der Cursor befindet, einfärben.

Sehr viel einfacher geht das Anwählen einer bestimmten Farbe im Menü-Betrieb. Dort brauchen Sie lediglich in der Zeile **Brush Foreground** (Pinsel Vordergrund) oder **Brush Background** (Pinsel Hintergrund) den Cursor auf die gewünschte Farbe setzen und den Knopf drücken.

Einige Tricks, wie Sie trotz der gegenüber Multicolor-Programmen eingeschränkten Farbmöglichkeiten zu hübschen, bunten Bildern gelangen, erfahren Sie in Kapitel 2.

Im Schwarzweiß-Betrieb schalten die Befehle **F** und **B** lediglich die Rahmenfarbe weiter, den aktuellen Modus beeinflussen sie jedoch nicht.

Da Sie beim *Anpinseln* eines Bildes die genaue Lage der Grenzen der 8*8-Punkte-Felder oft nicht kennen, kann es leicht vorkommen, daß Sie ein Feld zuviel anmalen. Hi-Eddi plus bietet deshalb eine Möglichkeit, solche Fehler schnell zu korrigieren. Ähnlich wie bei den meisten Zeichenmodi, wird bei einem Knopfdruck und gleichzeitig gedrückter SHIFT-Taste die Colorierung des 8*8-Punkte-Feldes, auf dem sich der Cursor befindet, wieder rückgängig gemacht. Genauer gesagt, es erscheint wieder die Farbe, die beim letzten Bildschirmwechsel dort war. Als *Bildschirmwechsel* gelten dabei:

1. Bildschirmanwahl, also die Eingabe einer Nummer von 1 bis 6.
2. Ausschalten des High-Resolution-Bildschirms. Das geschieht bei den Befehlen **H**, **V**, **SHIFT W** sowie den Disk- und Druckerbefehlen.
3. die Zoom-Funktion oder der Sprite-Editor.
4. die Befehle Move, Mirror und Turn.

Das Betätigen der Pfeil-nach-links-Taste hat dieselbe Wirkung wie SHIFT + Knopfdruck, jedoch für den ganzen Bildschirm sowie Vorder- und Hintergrund. Dabei werden alle Farbveränderungen seit dem letzten Bildschirmwechsel rückgängig gemacht.

SHIFT F Total Foreground - Vordergrund-Farbe
SHIFT B Total Background - Hintergrund-Farbe

Diese Befehle funktionieren im Farb- und Schwarzweiß-Betrieb gleich. Sie färben den gesamten Vorder- oder Hintergrund eines Bildschirms mit der momentanen Rahmenfarbe ein. Wollen Sie zum Beispiel einen schwar-

zen Hintergrund, dann tippen Sie zunächst so oft **F** oder **B**, bis der Rahmen schwarz ist und dann **SHIFT B**. Falls Sie das Ganze im Farb-Betrieb tun, müssen Sie beachten, daß Sie durch **F** oder **B** natürlich auch den entsprechenden Modus angewählt haben. Waren Sie zum Beispiel vorher im Draw-Modus und wollen damit weiterarbeiten, so müssen Sie erst wieder **D** eingeben.

Einfacher geht es auch hier wieder in der Menü-Betriebsart: Sie wählen dabei nur in den Zeilen **Total Foreground** oder **Total Background** die gewünschte Farbe.

Allerdings kann es im Farb-Betrieb oder Menü-Betrieb auch leicht passieren, daß man durch diese Befehle eine mühsam erstellte Kolorierung zerstört, da ja der ganze Bildschirm gleich eingefärbt wird. In diesem Fall genügt ein Druck auf die Pfeil-nach-links-Taste, und die alte Kolorierung ist wieder da!

1.9 Text in der Grafik

Natürlich muß ein Programm, das vorwiegend für technische Anwendungen konzipiert ist, auch die Möglichkeitbieten, die Zeichnungen zu beschriften. Dabei ist es mit dem normalen Commodore-Zeichensatz nicht getan. Hi-Eddi plus erlaubt daher die Verwendung beliebiger, selbst zu erstellender Zeichensätze, die zum Beispiel Sonderzeichen für alle möglichen Anwendungen beinhalten können. Einige Zeichensätze sind auch auf der Diskette schon vorhanden. Als weitere Besonderheit ist die Beschriftung in alle vier Richtungen möglich. Wie das beispielsweise in einer Schaltung aussehen kann, zeigt Ihnen Abb. 1.5.

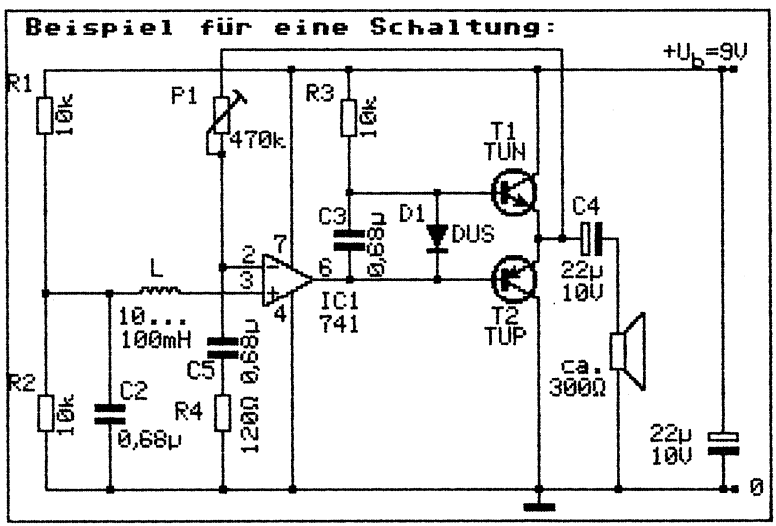


Bild 1.5: Das Schaltungsbeispiel auf der Diskette

T Text - Buchstaben und Grafiksymbole einfügen

Dieser Befehl ist der wichtigste, was Beschriftung anbelangt. Er aktiviert den Text-Modus, in dem als Cursor ein 8*8 Punkte großer Rahmen erscheint, der sich ähnlich verhält wie der Cursor im BASIC-Editor:

Sie können Buchstaben und - zusammen mit der SHIFT- oder Commodore-Taste - Grafikzeichen eingeben, die in gewohnter Weise auf dem Bildschirm erscheinen. Sie können auch durch gleichzeitiges Drücken der SHIFT- und Commodore-Taste zwischen den beiden Zeichensätzen umschalten. Alle 512 Zeichen der beiden Commodore-Zeichensätze lassen sich gleichzeitig darstellen. Mit **CTRL 9** und **CTRL 0** kann inverse Schrift ein- und ausgeschaltet werden. Schließlich läßt sich der Cursor in gewohnter Weise mit den Cursortasten steuern und auch die INST/DEL-Taste funktioniert, allerdings etwas anders als gewohnt: Bei DEL wird das zuletzt eingetippte Zeichen, also das Zeichen links vom Cursor, gelöscht, ohne jedoch die Zeichen hinter dem Cursor nachzuziehen. Ebenso wird bei INST nur das Zeichen, auf dem der Cursor steht, gelöscht, die Zeichen danach jedoch nicht verschoben. INST wird gebraucht, um für einzugebenden Text Platz zu schaffen. Die Buchstaben werden nämlich nur in den Bildschirm eingefügt, ohne den Untergrund zu löschen (ähnlich Append).

Neben der Cursorsteuerung mittels Cursortasten bleibt auch die Steuerung mittels Joystick oder Pad erhalten. Damit kann man, im Gegensatz zum normalen Texteditor, die Schrift punktgenau plazieren. Auch alle sonstigen Befehle des Grafik-Editors bleiben zugänglich, allerdings muß für die Eingabe zusätzlich die CTRL-Taste gedrückt werden, da sonst das entsprechende Zeichen gedruckt würde. Dies gilt jedoch nur für Buchstaben und Zahlen; Steuerzeichen dagegen, wie zum Beispiel **C= CLR/HOME** oder Funktionstasten, können weiterhin ohne CTRL eingegeben werden. Eine Ausnahme bildet auch die Space-Taste, die ja zur Menü-Anwahl im Menübetrieb besonders häufig gebraucht wird. Da die Eingabe von **CTRL Space** dafür wohl zu umständlich wäre, wird das Menü im Textmodus durch Knopfdruck (oder RETURN-Taste) angewählt. Dasselbe trifft auch für die Anwahl des Sprite-Editors im Farb- oder Schwarzweiß-Betrieb zu.

Durch Anwahl eines anderen Modus, zum Beispiel **CTRL D** für Draw, wird der Textmodus verlassen und es ist wieder die *Ein-Hand-Eingabe* der Befehle möglich, d.h. die CTRL-Taste muß nicht mehr gedrückt werden.

Z Schreibrichtung drehen

Mit **Z** - oder besser besagt **CTRL Z**, da man diesen Befehl nur im Textmodus verwenden wird - wird die Schreibrichtung um 90 Grad im Uhrzeigersinn gedreht. Es wird somit in der Reihenfolge normal - abwärts - auf dem Kopf stehend - aufwärts durchgeschaltet.

Die Orientierung der INST/DEL-Taste wird entsprechend mitgedreht, nicht dagegen die der Cursortasten.

SHIFT Z Zeichensatz ins RAM kopieren

Der gerade angewählte Commodore-Zeichensatz - also der Groß-Kleinschrift- oder der Großschrift-Grafik-Zeichensatz - wird aus dem ROM (Nur-Lese-Speicher) ins RAM (Schreib-Lese-Speicher) kopiert und für den Textmodus angewählt. Da der gesamte RAM-Speicher schon von den Grafikbildschirmen belegt ist, muß der Zeichensatz in eben einen solchen kopiert werden. Er wird in die ersten sieben Zeilen des aktuellen Bildschirmspeichers gelegt.

Die Zeichen, die nun im Text-Modus gedruckt werden, holt Hi-Eddi plus nicht mehr aus dem ROM, sondern aus dem RAM. Man kann nun diesen Zeichensatz im RAM ändern und die geänderten Zeichen im Text-Modus verwenden. So ist es möglich, sich Zeichensätze nach eigenem Geschmack und Bedarf zu erstellen, zum Beispiel mit wissenschaftlichen Sonderzeichen, mit neuem Schriftbild oder wie immer man es gern möchte. Als Beispiel mögen die drei auf der Diskette befindlichen Zeichensätze dienen, die Sie in Abb. 1.6 sehen.

Bild 1.6: Die Zeichensätze GRIECHISCH.ZS, PICA.ZS, PICA/GR.ZS

Natürlich lassen sich die so erstellten Zeichensätze auch auf Diskette abspeichern und wieder laden. Weiteres dazu erfahren Sie bei den Diskbefehlen.

C= Z Commodore-Zeichensatz anwählen

Durch gleichzeitiges Drücken der Commodore-Taste und Z sagen Sie Hi-Eddi plus, daß er die Zeichen wieder aus dem ROM nehmen soll. Ein noch im RAM befindlicher Zeichensatz kann dann gelöscht werden, ohne daß es im Text-Modus zu Komplikationen kommt.

1.10 Befehle zur Cursorsteuerung

Nun werden einige direkte Befehle erklärt, die ich weiter oben bereits angesprochen habe.

F1-F8 Tabulatoren

Die 4 Funktionstasten dienen als Speicher für 4 Cursorpositionen: Durch gleichzeitiges Drücken der SHIFT-Taste und einer Funktionstaste wird die momentane Cursorposition gespeichert; durch Drücken einer Funktionstaste allein springt der Cursor wieder genau an die gespeicherte Stelle. Im Einschaltzustand sind auf den Tasten F1 bis F5 schon einige Positionen, die man häufig benötigt, gespeichert. Dies sind insbesondere die Eckpunkte, vor allem wohl die rechte untere Ecke (auf F1 gespeichert), die der HOME-Position genau gegenüberliegt. Man braucht sie zum Beispiel zum Zeichnen eines Rahmens um das ganze Bild, wie bereits beim Rectangle-Befehl beschrieben.

Eine besonders nützliche Anwendung dieser Tabulatoren haben wir auch schon bei den Befehlen Line, Rec und Circle kennengelernt. Dort wird automatisch der jeweils erste Knopfdruck auf der Funktionstaste F7/F8 gespeichert. Durch einen Druck auf die Taste F7 springt der Cursor wieder an die Stelle, wo er stand, als der Knopf zum ersten Mal gedrückt wurde. Übrigens wird auch der erste Knopfdruck bei Move sowie die Cursorposition beim Aufruf der Zoom-Funktion auf F7 gespeichert.

Die anderen drei Funktionstasten können Sie natürlich bei Bedarf selbst belegen, um zu bestimmten Positionen schnell und ohne langes Rangieren zurückzukommen.

Eine weitere, sozusagen *fest programmierte* Tabulator-Taste ist die Taste CRL/HOME. Wird sie gedrückt, springt der Cursor in die linke obere Bildschirmecke.

H **Horizontal Step - Horizontale Schrittweite**
V **Vertical Step - Vertikale Schrittweite**
F1-F8 **Schrittweiten speichern**

Normalerweise bewegt sich der Cursor in Ein-Punkte-Schritten bei der Joystick- und Pad-Steuerung sowie in 8-Punkte-Schritten bei Steuerung mittels Cursortasten und bei der Zeicheneingabe im Textmodus. Diese Schrittweiten sind jedoch - getrennt für horizontale und vertikale Bewegung - frei programmierbar. Vier solcher Schrittweitenpaare können auf den Funktionstasten gespeichert und bei Bedarf schnell angewählt werden.

Dabei muß unterschieden werden zwischen Joystick- oder Pad-Steuerung einerseits und Steuerung mittels Cursortasten andererseits. Die Joystick- und Pad-Steuerung arbeitet immer mit der Schrittweite, die auf F1 gespeichert ist. Im Einschaltzustand ist das 1 für horizontal und vertikal, also pixelweise Bewegung des Cursors. Vergrößert man diese Schrittweite, so können damit im Draw-Modus punktierte Linien gezeichnet werden.

Welches Schrittweitenpaar für die Steuerung mittels Cursortasten herangezogen wird, kann man selbst bestimmen. Dazu muß die Commodore-Taste und eine Funktionstaste gedrückt werden. Im Einschaltzustand ist die Funktionstaste F3 angewählt, auf der die Schrittweite 8 für horizontal und vertikal gespeichert ist. Das ist die Schrittweite für den Textmodus! Durch Vergrößern dieser Schrittweite würde eine gedehnte Schrift oder ein größerer Zeilenabstand entstehen. Für besonders schmale Zeichensätze, wie die auf der Dikette enthaltenen Pica-Zeichensätze, ist auch eine Verringerung dieser Schrittweite sinnvoll. Auf F5 sind die Schrittweiten 24 und 21 gespeichert, die genau den Abmessungen eines Sprites entsprechen. Damit ist flächendeckende Bearbeitung mittels Sprite-Befehl oder Zoom-Funktion möglich. Auf F7 schließlich sind die Schrittweiten 160 und 96 abgelegt, deren Funktion im Zusammenhang mit dem Walk-Befehl besprochen wird.

Um nun zum Beispiel das Schrittweitenpaar 24/21, das auf F5 gespeichert ist, anzuwählen, drücken Sie gleichzeitig die Commodore-Taste und F5. Beachten Sie, der Cursor bewegt sich nur bei Steuerung mittels Cursortasten in diesen Schritten, die Joystick- oder Pad-Steuerung bleibt davon unbeeinflußt.

Falls Ihnen diese vier im Einschaltzustand vorgegebenen Schrittweiten nicht zusagen, können Sie sie umprogrammieren. Dazu wählen Sie zunächst die Funktionstaste an, deren Schrittweiten Sie ändern wollen. Zum Beispiel soll die Schrift auf einen Zeichenabstand von 10 Punkten gedehnt werden. Drücken Sie dazu die Commodore-Taste und die Funktionstaste F3, da wir

darauf die Textschrittweiten speichern wollen. Nun geben Sie **H** für horizontale Schrittweite ein. Hi-Eddi zeigt Ihnen nun an, daß F3 angewählt ist und die Schrittweite 8 beträgt. Tippen Sie nun **10** ein und schließen Sie die Eingabe mit RETURN ab. Wenn Sie jetzt in den Textmodus umschalten (**T** eingeben) und einige Zeichen schreiben, werden Sie sehen, daß der Zeichenabstand größer geworden ist. Drehen Sie aber die Schriftrichtung (**CTRL Z**) und geben erneut einige Zeichen ein, dann stellen Sie fest, daß der Zeichenabstand hier noch der alte ist. Wir müssen also auch die vertikale Schrittweite ändern. Sie ahnen sicherlich schon, wie das geschieht. Aber nicht vergessen: Sie müssen jetzt **CTRL V** eingeben, da Sie sich im Text-Modus befinden!

Haben Sie nun auch vertikal mit vergrößertem Zeichenabstand einige Zeichen eingegeben, wollen wir nun einmal in den Draw-Modus wechseln (**CTRL D**), mit Commodore-Taste und **F1** das Schrittweitenpaar der Joysticksteuerung anwählen und horizontale (**H**) und vertikale (**V**) Schrittweite auf 3 vergrößern. Und schon können Sie mit dem Joystick oder Pad punktierte Linien zeichnen! Das bleibt auch so, wenn Sie die Cursortasten-Steuerung, die ja jetzt ebenfalls in 3-Punkte-Schritten arbeitet, wieder auf eine andere Funktionstaste, zum Beispiel F3, umstellen.

Nachdem Sie nun gesehen haben, wie man mit den programmierbaren Schrittweiten umgeht, will ich Ihnen noch einige nützliche Anwendungsmöglichkeiten nennen:

Maßstäbe, Skalierungen oder exakte Diagramme sind damit problemlos zu erstellen. Ebenso Gitterraster, die man als Konstruktionsgitter mittels X-Befehls in Bildschirme ein- und wieder ausblenden kann. Auch das Positionieren des Cursors nach Koordinaten ist möglich. Dazu speichern Sie die Koordinaten als Schrittweiten, stellen dann den Cursor ins linke obere Bildschirmeck (CLR/HOME) und bewegen den Cursor nun mittels Cursortasten einmal abwärts und nach rechts. Die Erweiterung bietet dafür allerdings noch eine komfortablere Möglichkeit. Näheres dazu aber in Kapitel 4.

Einige weitere Anwendungsmöglichkeiten der programmierbaren Schrittweite werde ich Ihnen in Kapitel 2 aufzeigen. Auch wenn Sie sich im Augenblick noch nicht so recht vorstellen können, was man damit anfangen soll, werden Sie vermutlich bald nicht mehr darauf verzichten wollen.

*** Cursorblinken ein- und ausschalten.**

Sollte Sie das Blinken des Cursors stören, können Sie es mit * ausschalten. Im Menü ist diese Funktion als Glühbirne dargestellt. Ist das Blinken aus, so hat der Cursor die aktuelle Rahmenfarbe. Sie müssen also selbst darauf achten, den Cursor durch ungünstige Farbwahl nicht unsichbar zu machen. Andererseits können Sie ihn auch absichtlich verstecken. Durch nochmalige Eingabe von * blinkt der Cursor wieder.

+ Cursorgeschwindigkeit und Pad-Empfindlichkeit erhöhen
- Cursorgeschwindigkeit und Pad-Empfindlichkeit erniedrigen

Diese Befehle wurden bereits bei der Cursorsteuerung in Abschnitt 1.2 erläutert.

1.11 Disk- und Druckerbefehle

Hi-Eddi plus bietet zwar im Computer selbst schon eine ganze Menge Platz für Bilder, aber irgendwann werden Sie Ihre Werke auch auf Diskette abspeichern oder auf den Drucker ausgeben wollen, da sie sich nur so dauerhaft aufbewahren lassen. Sehen wir uns deshalb als nächstes die Befehle für die Speicherung auf Diskette an:

C= D Directory (C= bedeutet Commodore-Taste)

Durch gleichzeitiges Drücken der Commodore-Taste und **D** wird das Directory, also das Disketten-Inhaltsverzeichnis angezeigt. Da es möglicherweise so lang ist, daß es nicht auf den Bildschirm paßt, kann die Auflistung durch Drücken einer beliebigen Taste angehalten werden. Nach dem Loslassen der Taste wird die Ausgabe fortgesetzt. Am Ende des Directory wartet Hi-Eddi plus auf einen Tastendruck, nach dem es wieder in den Grafikeditor zurückkehrt.

C= L Load
C= S Save

Nach dem Aufruf einer dieser Funktionen durch gleichzeitiges Drücken der Commodore-Taste und **L** oder **S** werden Ihnen im Schwarzweiß-Betrieb drei und im Farb-Betrieb vier Möglichkeiten zur Auswahl gestellt: Sie können Grafikbilder, Zeichensätze, Sprites oder Farbbilder, letzteres natürlich nur im Farb-Betrieb, abspeichern und wieder laden. Schauen wir uns die einzelnen Möglichkeiten näher an:

Grafikbild:

Hierbei wird der aktuelle, also sichtbare, Bildschirm abgespeichert. Analog wird beim Laden das Bild von der Diskette in den aktuellen Bildschirm geladen und überschreibt damit das in diesem Speicher befindliche Bild. Es wird unabhängig von der Betriebsart - also Farbe oder Schwarzweiß - nur die Bitmap, also die 8-kByte-Bildinformation, abgespeichert. Die Farbe bleibt unberücksichtigt! Somit entspricht die Farbe beim Laden eines Grafikbildes der momentan eingestellten und nicht der beim Abspeichern vorliegenden Farbe. Ein Grafikbild belegt auf der Diskette 33 Blocks.

Farbbild:

Dieses Format steht nur bei Farb-Betrieb zur Verfügung. Analog zum Grafikbild wird ebenfalls der aktuelle Bildschirm abgespeichert bzw. in den aktuellen Bildschirm geladen. Allerdings wird beim Farbbild die Farbinformation (1 kByte) anschließend an die Bitmap abgespeichert. Ein Farbbild ist somit 9 kByte lang und belegt auf der Diskette 37 Blocks. Beim Laden eines Farbbildes entsprechen die Farben nun denen beim Speichern des Bildes. Das gilt insbesondere auch dann, wenn das Bild mit mehreren verschiedenen Farben eingefärbt ist, was ja der Sinn der Farb-Betriebsart ist.

Zeichensatz:

Wenn Sie beim Speichern ein **Z** für Zeichensatz eingeben, werden die ersten sieben Zeilen des aktuellen Bildschirms abgespeichert. Damit dies einen Sinn ergibt, muß dort natürlich ein Zeichensatz stehen. Wie Sie das erreichen, haben Sie in Abschnitt 1.9 beim Befehl **SHIFT Z** schon gelernt. Mit diesem Befehl nämlich wird der Zeichensatz aus dem ROM ins RAM kopiert, wo Sie ihn nach Belieben verändern können. Ein Zeichensatz ist 2 kByte lang und belegt 9 Blocks auf der Diskette. Wenn Sie einen Zeichensatz laden, wird er automatisch angewählt, d.h. die Zeichen, die im Textmodus gedruckt werden, holt sich Hi-Eddi plus aus diesem geladenen Zeichensatz. Drei Zeichensätze befinden sich auf der beiliegenden Diskette (siehe Abb. 1.6). Laden Sie diese doch einmal! Die Zeichensätze sind durch **.ZS** markiert; außerdem können Sie sie an der Länge von 9 Blocks erkennen. Der Zeichensatz **GRIECHISCH.ZS** enthält anstelle der inversen Zeichen das komplette griechische Alphabet. Wenn Sie diesen Zeichensatz verwenden, können Sie durch **CTRL 9** - so, wie Sie sonst auf reverse Schrift umschalten - auf die griechischen Zeichen umschalten. Entsprechend schalten Sie sie mit **CTRL 0** wieder aus. Die griechischen Zeichen benötigt man vor allem bei technischen und wissenschaftlichen Plänen und Zeichnungen, zum Beispiel für *Ohm*, *mikro*, Winkelbezeichnungen etc. Der Zeichensatz **PICA.ZS** entspricht von seiner Belegung her dem Commodore-Groß-Kleinschrift-Zeichensatz, jedoch ist das Schriftbild ein ganz anderes. Anstelle der für den Fernseher optimierten fetten Commodore-Zeichen besitzt der Pica-Zeichensatz schlanke, zierliche Zeichen, die bei der Druckausgabe besser wirken als die dicken Zeichen. Außerdem läßt sich mit ihnen schmaler und somit platzsparender schreiben, was bei der Beschriftung von Plänen und Schaltungen wichtig ist.

Allerding müssen Sie dazu den Zeichenabstand auf 7 oder 6 Punkte verkleinern (kein Zeichen ist breiter als 5 Punkte!). Wie das Ändern der Schrittweite funktioniert, haben Sie bereits bei den Befehlen **H** und **V** gelernt. Vergessen Sie nicht, auch die vertikale Schrittweite zu ändern, wenn Sie auch senkrecht schreiben wollen. Der Zeichensatz **PICA/GR.ZS** entspricht vom Schriftbild her dem Pica-Zeichensatz, besitzt jedoch anstelle der reversen Zeichen wieder das griechische Alphabet. Dieser Zeichensatz wurde auch zur Erstellung des Demo-Bildes **SCHALTUNG.PIC** (Abb. 1.5) verwendet. Laden Sie dieses Bild doch einmal und schauen Sie es sich genau an. An diesem Demo erkennen Sie bereits die wesentlichen Vorteile der Beschriftungsmöglichkeiten von Hi-Eddi plus: schmale, platzsparende und druckerfreundliche Schrift mit beliebigen Sonderzeichen und in alle Richtungen drehbar.

Sprite:

Hierbei wird, analog zu den vorigen Eingabemöglichkeiten, auch das aktuelle Sprite abgespeichert, also dasjenige, das Sie sehen, wenn Sie den Sprite-Editor anwählen. Wollen Sie das zweite Sprite speichern, so müssen Sie es zuerst mittels **C** im Sprite-Editor anwählen. Beachten Sie aber, daß Sie den Sprite-Editor nur aus einem der Spritemodi anwählen können, also Get, Append, Stamp und Erase. Da ein Sprite nur 63 Byte beansprucht, braucht es auf der Diskette nur einen Block.

Nun noch einige Hinweise, die sich auf alle Formate beziehen:

Zunächst empfehle ich Ihnen, Ihre Files durch einen Zusatz zum Namen zu kennzeichnen, um sie leichter identifizieren zu können, wie zum Beispiel **.ZS** für Zeichensätze, **.CS** für Construction Sets, **.PIC** für Grafikbilder, **.COL** für Farbbilder und **.SPR** für Sprites. Natürlich können Sie auch inhaltliche Unterschiede kennzeichnen, zum Beispiel **.PAP** für Programmablaufpläne, **.STG** für Struktogramme oder was immer Sie gerade erstellen.

Wollen Sie aus dem Load/Save-Menü herauskommen, ohne etwas zu laden oder zu speichern, dann geben Sie einfach keinen Filenamen ein. Dazu beantworten Sie die Frage nach dem Filenamen nur mit RETURN.

Außerdem rate ich Ihnen, Files nur in demselben Format zu laden, in dem Sie sie gespeichert haben. Laden Sie dagegen zum Beispiel ein Farbbild (37 Blocks) im Schwarzweiß-Betrieb als Grafikbild, so erscheint zwar tatsächlich das Bild, nur eben ohne Farbe. Durch die Farbinformation wurde jedoch ein Teil eines anderen sich im Speicher befindlichen Bildes über-

schrieben. Laden Sie schließlich ein Grafikbild als Sprite, so führen Sie einen Absturz herbei. Aber wer käme schon auf so unsinnige Gedanken! Es gibt jedoch tatsächlich einen Fall, in dem *falsches* Laden sogar einen Sinn ergibt: Da ein Zeichensatz, wenn er im RAM steht, die ersten sieben Zeilen eines Bildschirms beansprucht, ist dieser Bildschirm zum Zeichnen nicht mehr zu gebrauchen. Damit er jedoch nicht völlig ungenutzt ist, könnte man den freien Teil als Construction-Set benutzen. Laden Sie dazu das Schaltungs-Construction-Set **ANALOG.CS** von der beiliegenden Diskette. Sie werden sehen, daß die oberen sieben Zeilen extra für einen Zeichensatz freigehalten wurden. Dorthin können Sie nun den Zeichensatz **PICA/GR.ZS** laden. Und nun kommem wir zu einem kleinen Trick, mit dem Sie vermeiden können, daß Sie jedesmal das Construction-Set und den Zeichensatz getrennt laden müssen: Speichern Sie den ganzen Bildschirm als Grafikbild ab. Wenn Sie diese so erzeugte Kombination aus Grafikbild und Zeichensatz jetzt erneut laden - und zwar als Zeichensatz!! - dann haben Sie den gleichen Zustand wie nach dem getrennten Laden der entsprechenden *Einzelteile* erreicht. Hi-Eddi plus *denkt* nämlich, es soll einen Zeichensatz laden und wählt ihn deshalb für den Text-Modus an. Daß in Wirklichkeit ein ganzer Grafikbildschirm geladen wurde, hat keine unangenehmen Folgen.

Damit kommen wir gleich zum nächsten Thema, dem Laden fremder Files, was ja eigentlich auch ein *falsches* Laden ist. Die Lade-Routine von Hi-Eddi plus ist so ausgelegt, daß ziemlich alles geladen werden kann - sogar Programme, allerdings geben diese grafisch nicht viel her! Diese Großzügigkeit hat, wie bereits angedeutet, den Nachteil, daß man durch *falsches* Laden Ärger bekommen kann, d.h. im schlimmsten Fall kommt es zum Absturz. (Die Hardware, also den Computer selbst, können Sie dagegen auch durch noch so einfallsreiche Fehlbedienung nicht beschädigen. Also keine Angst, im Notfall müssen Sie das Programm neu laden!) Diese Großzügigkeit hat jedoch auch einen entscheidenden Vorteil: Es können viele Files anderer Grafikprogramme geladen werden. Bei Programmen, die im High-Resolution-Modus arbeiten und nur die Bitmap abspeichern, gibt es überhaupt keine Probleme. Deren Files können als Grafikbilder geladen werden. Zu diesen Programmen zählen vor allem BASIC-Erweiterungen (Simon's BASIC, Hires 3, Grafik 2000) sowie Diashows. Apropos Simon's BASIC: Diese Erweiterung kann eigentlich keine Grafiken abspeichern. Wie Sie es trotzdem bewerkstelligen, steht in der Zeitschrift 64'er, Ausgabe 3/85.

Etwas problematischer wird es bei Programmen, die zwar in High-Resolution arbeiten, aber die Farbe mit abspeichern. Während Hi-Eddi plus die Farbinformation hinten an die Bitmap anhängt, geschieht dies zum Beispiel beim Grafikprogramm *Doodle* genau umgekehrt. Wie man sich solche Programme dennoch zugänglich macht, erfahren Sie in Kapitel 4.

Am schwierigsten schließlich ist es bei den bunten Malprogrammen, die im Multicolormodus arbeiten, wie zum Beispiel Paint Magic oder Koalapainter. Natürlich würde ich diese Programme nicht erwähnen, wenn es nicht doch möglich wäre, sie zu laden. Es sind vor allem zwei Probleme, die dabei im Wege stehen:

Erstens: Durch die gänzlich andere Bilddarstellung lassen sich Multicolor-Bilder nur eingeschränkt in High-Resolution-Bilder umwandeln. Die Farbe kann man dabei vollkommen vergessen, die eingeschränkten Farbmöglichkeiten des High-Resolution-Modus reichen zu einer farbigen Darstellung nicht aus. Doch schon eine Schwarzweiße-Darstellung bereitet Probleme. So ist zum Beispiel die Farborganisation des Koalapainters so kompliziert, daß dessen Bilder im Hi-Eddi plus praktisch unbrauchbar sind. Anders dagegen das Programm Paint Magic. Dessen Farborganistaion sorgt dafür, daß die Farben durch Schattierungen dargestellt werden können. Dazu muß lediglich das Bild im Hi-Eddi plus mit entsprechenden Farben dargestellt und eventuell invertiert (**I**) werden. Danach können Sie es in guter Qualität ausdrucken.

Das zweite Problem beim Laden von Multicolor-Bildern ist die Tatsache, daß es kein einheitliches Format gibt, in dem Bilder auf Diskette abgespeichert werden. Vor allem die Reihenfolge, in der Bitmap und Farbinformationen abgespeichert werden, führt - wie bereits angedeutet - zu Problemen. Da der Koalapainter dieselbe Reihenfolge wie Hi-Eddi plus benutzt, können dessen Bilder direkt geladen werden. (Dazu den Filenamen in der Form **?PIC...** eingeben, um das Steuerzeichen am Anfang zu umgehen.) Nur nützt das nicht viel! Paint-Magic-Bilder werden zusammen mit einem BASIC-Lader abgespeichert und können somit direkt geladen werden. Man muß also erst ein Paint-Magic-Bild in den Rechner laden (mit **LOAD**), dann Hi-Eddi plus laden, in Schwarzweiß-Betriebsart starten und auf die Frage **LOESCHEN** mit **N** antworten. Das Bild findet sich dann in Speicher 7. Deshalb ist auch Schwarzweiß-Betrieb nötig, da es im Farb-Betrieb diesen Speicher 7 nicht gibt und der zugehörige Speicherbereich für die Ablage der Farbinformationen benützt wird. Natürlich ist dieses Vorgehen sehr umständlich. In Kapitel 4 zeige ich Ihnen deshalb einen komfortableren Weg, der es sogar ermöglicht, Hi-Eddi-Bilder im Paint-Magic-Format oder Koalapainter-Format abzuspeichern. Damit ist dann ein vollständiger Austausch der Bilder dieser Programme möglich.

Wollen Sie Hi-Eddi-Bilder in eigenen BASIC-Programmen laden, so hilft Ihnen dabei das Programm **PIC-LOADER**, das sich auf der Diskette befindet. Es erzeugt mittels Data-Zeilen ein kleines Maschinenprogramm im Kassettenpuffer, das Sie mit einem SYS-Befehl aufrufen können. Dieses Programm lädt sowohl Grafikbilder als auch Farbbilder. Wenn Sie das Programm mit **RUN** starten, gibt es zunächst eine kurze Erklärung, liest dann in Zeile 300 die Datas und schreibt sie in den Kassettenpuffer ab Adresse 900. Diese Zeile 300 sowie die Data-Zeilen können Sie in eigene Programme übernehmen, um den Pic-Loader zur Erzeugung des Maschinenprogrammes nicht jedesmal extra laden zu müssen. Die Zeilen 500 bis 590 dienen als Demo der Anwendung des Pic-Loaders. Wenn Sie **RUN500** eingeben, können Sie es gleich ausprobieren: Das Programm fragt nach dem Namen eines Bildes, das es laden soll. In den Zeilen 510 bis 540 wird die Grafik eingeschaltet. Dabei wird die Bitmap unter das Betriebssystem-ROM und das Video-RAM in den RAM-Bereich ab $C000 (=49152) gelegt. Somit geht kein BASIC-Speicher durch die Grafik verloren. In Zeile 550 bis 560 wird das Video-RAM vorbelegt, also die Farben für das Bild festgelegt. Das ist eigentlich nur beim Laden eines Schwarzweiß-Bildes notwendig, da ein Farbbild ja seine eigene Farbinformation mitbringt und der Pic-Loader diese ins Video-RAM schreibt. In Zeile 570 schließlich wird das Bild von Diskette geladen, indem das Maschinenprogramm im Kassettenpuffer aufgerufen wird. Die Syntax dieses Aufrufes ist: **SYS 900, NAME**, wobei **NAME** der Name des zu ladenden Bildes ist. Anstelle eines Namens in Anführungszeichen kann hier auch eine Stringvariable stehen, wie in Zeile 570 zu sehen ist. In Zeile 580 wird auf einen Tastendruck gewartet und in Zeile 590 schaltet die Grafik wieder ab.

Um Sprites, die von Hi-Eddi plus erzeugt wurden, in eigene BASIC-Programme zu integrieren, formt man sie am besten in Datazeilen um. Diese Arbeit übernimmt das folgende kleine Programm, das aus dem Heft 64'er, Ausgabe 11/84 entnommen wurde:

```
1 REM SPRITE-DATA-MAKER
2 REM BORIS SCHNEIDER, 64'ER NR11/84, S.30
3 INPUT "STARTZEILE,FILENAME";SZ,F$
4 OPEN 1,8,2,F$+",R,P": GET#1,A$,A$
5 FOR X=1 TO 21: PRINT SZ+X;"DATA";
6 FOR Y=1 TO 3: GET#1,A$: A$=A$+CHR$(0)
7 PRINT RIGHT$("  "+STR$(ASC(A$)),3);
8 PRINT CHR$((1+(Y=3))*ASC(","));
9 NEXT Y: PRINT: NEXT X: CLOSE1: PRINT
```

Das Programm liest ein Sprite-File ein, dessen Name erfragt wird und druckt, beginnend bei der einzugebenden Startzeile, 21 Zeilen mit je drei Datas aus. Sie müssen dann nur noch in die oberste Zeile gehen (**HOME**) und 21 mal RETURN drücken, um die Zeilen in den Speicher zu übernehmen. Die Zeilen dieses Programms können Sie löschen, wenn Sie sie nicht mehr benötigen.

C= C Command to Disk

Nach gleichzeitigem Drücken der Commodore-Taste und **C** kann ein DOS-Befehl an die Floppy gesendet werden. So kann zum Beispiel mit **S:NAME.PIC** ein Bild gelöscht werden, es können Disketten formatiert oder validiert werden. Genaue Beschreibungen der DOS-Befehle finden Sie im Floppy-Handbuch. Nach jedem Kommando oder wenn Sie RETURN drükken, wird der Fehlerkanal ausgelesen. Liegt eine Fehlermeldung vor, zeigt sie Hi-Eddi plus an und wartet auf einen Tastendruck. Liegt dagegen kein Fehler vor, kehrt Hi-Eddi plus sofort in den Grafikeditor zurück.

C= P Print

Mit dieser Eingabe wird die Druckerroutine aufgerufen. Hier ist jedoch etwas ganz Wichtiges zu beachten:

Die Druckerroutine wird in Overlaytechnik von der Diskette nachgeladen, d.h., daß bereits bei Eingabe dieses Befehls die Programmdiskette im Laufwerk liegen muß! Falls Sie das vergessen, passiert zwar nichts, aber der Befehl kann natürlich nicht ausgeführt werden. Außerdem erinnert Sie das Blinken der Fehler-LED am Laufwerk daran, die Programmdiskette einzulegen. Die Programmdiskette muß auch bis zur Rückkehr in den Grafikeditor im Laufwerk bleiben, da nach Beendigung des Druckvorgangs der Teil des Programms, der durch die Druckerroutine überschrieben wurde (daher der Name *Overlay*), wieder geladen werden muß.

Doch nun zu den Fähigkeiten der Druckerroutine. Auf einem Epson- oder Epson-kompatiblen Drucker können Sie ein Bild in einfacher oder doppelter Größe oder zwei Bilder in einfacher Größe nahtlos nebeneinander ausdrucken. Außerdem werden vor und nach einem Ausdruck keine zusätzlichen Zeilenvorschübe ausgegeben, so daß nacheinander gedruckte Bilder zusammenhängen. Damit können Sie aus mehreren Bildern Riesenhardcopys mit einer Breite von 640 Punkten und praktisch unbegrenzter Länge ausdrucken. Mit 6 Bildschirmen, also noch ohne Nachladen von Bildern von

der Diskette, läßt sich eine Superhardcopy mit 640*600 Punkten erstellen. Dabei wird die Auflösung des Druckers voll ausgenutzt! Zusätzlich besteht die Möglichkeit, jede Zeile zweimal, jedoch leicht versetzt, drucken zu lassen. Dieses Feature nennt sich *Double Strike*; es sorgt für eine bessere Qualität der Hardcopy. Verfügt Ihr Drucker über einen Plot-Modus wie zum Beispiel der Epson FX-80, dann kann Hi-Eddi plus auch diese Fähigkeit nutzen. Damit werden Kreise dann wirklich rund!

Bei einem Commodore-Drucker VC1525, MPS801 oder MPS803 sind keine großen oder nebeneinander liegenden Bilder möglich, da nur 480 Punkte in eine Zeile passen. Aber es können immerhin noch Bilder untereinander zusammengefügt werden.

Die Bedienung der Druckerroutine ist recht einfach. Die Routine erfragt die Nummer des ersten und zweiten von zwei nebeneinander auszudruckenden Bildern. Wollen Sie nur eines ausdrucken, so geben Sie bei der Nummer des zweiten Bildes eine **0** ein. In diesem Falle fragt die Routine, ob das Bild groß gedruckt werden soll; antworten Sie hier entweder mit **J** oder N. Anschließend folgen die Fragen nach Double Strike und, sofern Ihr Drukker dafür ausgerüstet ist, für den Plot-Modus. Zuletzt vergewissert sich das Programm noch, ob es auch wirklich drucken soll.

Nach einem Ausdruck folgt die Frage **BILDUNTERSCHRIFT:**. Sie haben hier nun die Möglichkeit, einen maximal 80 Zeichen langen Text einzutippen, der unter das Bild gedruckt wird. Vor und nach dem Text wird eine Leerzeile ausgedruckt. Wünschen Sie keine Bildunterschrift, geben Sie bei dieser Frage einfach RETURN ein. Es werden dann auch keine Zeilenvorschübe ausgegeben, damit aufeinanderfolgende Ausdrucke nahtlos zusammenhängen.

Zur Eingabe dieses Textes schalten Sie am besten, falls nicht sowieso schon geschehen, in den Groß-Kleinschrift-Modus (mittels **SHIFT C=**), damit der Text am Bildschirm so erscheint wie später auf dem Papier. Für Epson- und Epson-kompatible Drucker wird natürlich die Umwandlung von Commodore-ASCII in Standard-ASCII durchgeführt. Ferner wird für den Klammeraffen (rechts neben **P** auf der Tastatur) der ASCII-Code 27, also ESC, ausgesandt. Damit haben Sie die Möglichkeit, die Schriftart der Bildunterschrift festzulegen, zum Beispiel kursiv zu schreiben oder einzelne Worte zu unterstreichen.

Die deutschen Umlaute erreichen Sie folgendermaßen:

Ä:	SHIFT +	ä:	eckige Klammer auf
Ö:	C= -	ö:	Pfund-Zeichen
Ü:	SHIFT -	ü:	eckige Klammer zu
		ß:	Pfeil nach oben

Soll der Text nicht am linken Rand beginnen, so tippen Sie zunächst die gewünschte Anzahl Blanks ein.

Danach erscheint die Frage **NOCHMAL**, die Ihnen die Gelegenheit gibt, gleich mehrere Bilder untereinander auszudrucken.

Falls Sie einen Commodore-Drucker VC1525, MPS801 oder MPS803 besitzen, können Sie gleich mit der Arbeit beginnen, denn für diese Drucker ist die Druckerroutine voreingestellt. Andernfalls müssen Sie vor dem ersten Ausdruck die Routine an Ihren Drucker anpassen. Wie das geht, steht in Kapitel 3.

1.12 Hi-Eddi plus als Kino

Einige Anwendungen der vielen Bildschirmspeicher, die Hi-Eddi plus bietet, haben wir bereits kennengelernt: Riesen-Hardcopys, Construction-Sets, Verknüpfungen oder Sicherheitskopien. Eine Anwendung, die sich geradezu anbietet, folgt jetzt: das sequentielle Durchschalten der Bilder in programmierbarer Reihenfolge. Das kann langsam erfolgen, zum Beispiel alle 20 Sekunden ein neues Bild, etwa für Schaufensterwerbung oder zu Demonstrationszwecken. Oder es kann schnell erfolgen, mit bis zu 24 Bildern pro Sekunde, womit ein richtiger, praktisch flimmerfreier Trickfilmeffekt entsteht.

W **Walk - Bildfolge ablaufen lassen**
SHIFT W **Bildsequenz programmieren**

Mit dem Befehl W werden die Bildschirmspeicher durchgeschaltet. Da jedoch ein Bildschirm als *Leinwand* benutzt wird - nämlich Nummer 7 im Schwarzweiß-Betrieb und Nummer 6 im Farb- und Menübetrieb - stehen nur noch sechs Bildschirme im Schwarzweiß-, fünf im Farb- und vier im Menü-Betrieb zur Verfügung. Diese können mit einer Maximalgeschwindigkeit von 8 Bildern pro Sekunde durchgeschaltet werden. Für Schaufensterwerbung zum Beispiel reicht das voll aus, dort muß es sogar viel langsamer gehen. Für schnelle Bewegungsabläufe ist das jedoch nicht ausreichend. Deshalb können alle Bildschirme in Viertelbilder zerlegt werden. Es steht somit zwar nicht mehr die volle Bildschirmfläche zur Verfügung, dafür jedoch insgesamt maximal 24 Bilder. Diese können außerdem mit einer Geschwindigkeit von maximal 24 Bildern pro Sekunde durchgeschaltet werden.

Die Reihenfolge, in der die Bilder durchgeschaltet werden, teilt man dem Programm durch einen Sequenzstring mit, der aus den Zahlen 1 bis 6 sowie den Buchstaben A bis X bestehen kann. Mit jeder Zahl und jedem Buchstaben wird dabei ein Bild aufgerufen: mit den Zahlen die großen Bildschirme entsprechend ihrer Nummer und mit den Buchstaben die Viertelbilder. So besteht der Bildschirm Nummer 1 aus den Viertelbildern A (links oben), B (rechts oben), C (links unten) und D (rechts unten). Dieses System setzt sich entsprechend fort, nachstehende Tabelle gibt die Zuordnung für alle Bildschirme an:

Bildschirm Nummer	Viertelbilder			
	l.o.	r.o.	l.u.	r.u.
1	A	B	C	D
2	E	F	G	H
3	I	J	K	L
4	M	N	O	P
5	Q	R	S	T
6	U	V	W	X

Nachdem der Sequenzstring vom Programm abgearbeitet wurde, geht es wieder von vorne los. Die Folge läuft also zyklisch in einer Endlosschleife durch. Mit den Tasten + und - kann die Geschwindigkeit in 9 Stufen eingestellt werden. Bei gedrückter SHIFT-Taste läuft der Film rückwärts. Mit der STOP-Taste können Sie die Vorstellung abbrechen.

Nun noch einmal zum Sequenzstring: Wie schon gesagt, besteht er aus den Zahlen 1 bis 6 und den Buchstaben A bis X - zumindest im Schwarzweiß-Betrieb. Bei Farbe dürfen nur die Bildschirme 1 bis 5 oder die Viertelbilder A bis T und bei Menü-Betrieb die Bildschirme 2 bis 5 und die Viertelbilder E bis T aufgerufen werden. Daneben ist noch der Klammeraffe (@) erlaubt: Er bewirkt, unabhängig von der durch + und - eingestellten Geschwindigkeit, eine Verzögerung von ca. 4 Sekunden. Damit kann für Schaufensterwerbung oder Demonstrationszwecke die Zeit, die ein Bild stehenbleiben soll, festgelegt werden. Hier ein Beispiel für einen Sequenzstring, wie er für diesen Anwendungsfall aussehen könnte: **1@@@2@3@@**. Hierbei werden die Bilder 1, 2 und 3 zyklisch durchgeschaltet, wobei Bildschirm 1 etwa zwölf Sekunden, Bildschirm 2 vier Sekunden und Bildschirm 3 acht Sekunden stehenbleibt.

Natürlich dürfen alle Zahlen und Buchstaben auch mehrfach vorkommen, sogar gemischt, sofern dies sinnvoll ist. Einzige Begrenzung: Der Sequenzstring darf nicht länger als 80 Zeichen, also zwei Bildschirmzeilen, sein.

Und nun noch eine Bemerkung zur Einteilung eines Bildschirms in Viertelbilder: Bei einer Auflösung von 320*200 Punkten wäre es logisch, für ein Viertelbild 160*100 Punkte zu nehmen. Dies ist aber nicht möglich! Grund dafür ist wieder die begrenzte Farbauflösung, die, wie in Abschnitt 1.1 dargelegt, der Auflösung im Textbildschirm von 40*25 Zeichen entspricht. Nun ist 25 aber nicht durch 2 teilbar! Folglich kann man für einen Viertel-

bildschirm nur 20*12 Zeichen verwenden, was im High-Resolution-Modus einem Feld von 160*96 Punkten entspricht. Und genau das sind auch die Maße eines Viertelbildes für den Walk-Befehl. Die unterste Zeile des Bildschirms, entsprechend 8 Punktreihen, bleibt also unbenutzt.

Nun noch zur Frage: Wie erstellt man solche Bilder? Dabei gibt es einiges zu beachten. Schließlich soll ein möglichst fließender Bewegungsablauf entstehen, die Bilder sollen nicht wackeln, hüpfen oder zittern. Und dann soll das Ganze noch möglichst wenig Arbeit machen! Man will ja nicht jedes Bild ganz neu zeichnen.

Die naheliegendste Regel beim Erstellen von Bildern für den Walk-Befehl ist wohl die, daß alle *unbeweglichen* Elemente, also das, was in allen Bildern an derselben Stelle steht, nur einmal gezeichnet werden müssen und dann beliebig oft kopiert werden können. Vielleicht ist Ihnen der *alte* Hi-Eddi aus dem 64er, Ausgabe 1/85, bekannt. Dann erinnern Sie sich bestimmt noch an den 4-Takt-Motor, der dort als Beispiel für den Walk-Befehl diente. In Abb. 1.7 sehen Sie nochmals einige Bilder aus dieser Sequenz. In diesen Bildern ist sehr viel *fest*. Man braucht es nur ein einziges Mal zu zeichnen und kann es dann in alle anderen Viertelbilder, sozusagen als *Kulisse*, in die dann die beweglichen Teile eingezeichnet werden, kopieren. Das Kopieren geschieht natürlich mittels Move-Befehl, wobei es hier jetzt von Vorteil ist, daß nur im 8-Punkte-Raster verschoben werden kann. Denn in Ein-Punkte-Schritten exakt zu verschieben, wäre recht schwierig. Und noch ein weiteres Feature kann jetzt sinnvoll genutzt werden: die programmierbare Schrittweite. Wie dort bereits angedeutet, sind auf der Funktionstaste F7 die Schrittweiten 160 und 96 gespeichert. Wählen Sie mit Commodore-Taste und **F7** diese Schrittweiten an. Schalten Sie nun in den Move-Modus (M) und setzen Sie den Cursor mit **HOME** in die linke obere Bildschirmecke. Nun drücken Sie den Knopf und fahren dann den Cursor mit den Cursortasten um je einen Schritt nach unten und rechts. Jetzt sitzt der Cursor exakt in der linken oberen Ecke des rechten unteren Viertelbildes. Wir wollen jedoch das linke obere Viertelbild zum Verschieben markieren. Fahren Sie deshalb den Cursor mittels Joystick ein Stückchen nach oben und nach links. Dabei geht es nicht so genau zu: Zwischen ein und acht Punkten dürfen Sie in jede Richtung fahren, um noch genau das rechte untere 8*8-Punkte-Feld des ersten Viertelbildes zu treffen. Drücken Sie nun zum zweiten Mal den Knopf und es erscheint, wenn Sie alles richtig gemacht haben, die Markierung genau auf dem linken oberen Viertelbild. Wenn nicht, dann drücken Sie die Pfeil-nach-links-Taste und versuchen es noch einmal. Stimmt die Markierung, so setzen Sie den Cursor mit **HOME** wieder ins linke obere Bildschirmeck und fahren ihn dann mittels Cursortaste um einen Schritt, also um 160 Punkte,

nach rechts. Nun steht er in der linken oberen Ecke des rechten oberen Viertelbildes. Wenn Sie dort zum dritten Mal den Knopf drücken, wird das linke obere Viertelbild ins rechte obere Viertelbild kopiert. Und wenn Ihnen jetzt Tastatur und Monitor vor den Augen verschwimmen, dann machen Sie eine kleine Pause, lesen dann noch einmal ganz ruhig diesen Abschnitt durch und probieren es erneut. Es klappt bestimmt!

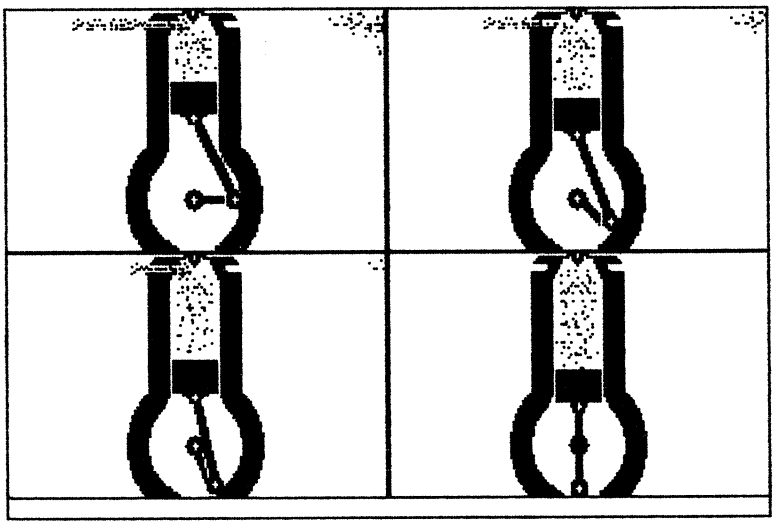


Bild 1.7: Der Ottomotor aus 64'er, Nummer 1/85

Nun geht es daran, die beweglichen Teile einzuzeichnen. Im Beispiel mit dem Motor fängt man am besten mit dem Kolben an, den man im Sprite-Editor erstellen kann und dann mittels **Append** in verschiedenen Höhen in die Zylinder einsetzt. Auch hierbei hilft wieder die programmierbare Schrittweite: Haben Sie den Kolben in ein Bild eingesetzt, so fahren Sie den Cursor mittels Cursortasten ins nächste Bild. Dort gehen Sie nun einige Pünktchen höher oder tiefer und setzen den Kolben wieder ein. Beachten Sie dabei auch die sinusförmige Bewegung des Kolbens: Er darf die Strecke vom oberen zum unteren Endpunkt nicht gleichmäßig schnell zurücklegen, sondern muß in der Mitte größere Schritte machen als am Rand.

Schwieriger wird es natürlich bei Bewegungsabläufen, bei denen alles *fließt* und nichts stillsteht. Zum Beispiel bei einem laufenden Pferd oder einem fliegenden Schwan. Hier kann man entweder ein Drahtgerüst erstellen, das man nachträglich genauer ausarbeitet, oder man arbeitet sich von Bild zu Bild vor, indem man das jeweils letzte Bild ins nächste kopiert und es entsprechend verändert.

Damit Sie ohne viel Arbeit sofort in den Genuß eines Trickfilmes kommen, hätte ich natürlich einige Bilder auf die Diskette aufnehmen können. Doch die ist bereits mit den vielen Construction-Sets und Beispielbildern gefüllt. Ich habe deshalb einen anderen Weg gewählt: Das Programm **3D-CLIPS** auf der Diskette berechnet und zeichnet die Bilder für sehr effektvolle, bewegte 3D-Grafiken. In Abb. 1.8 ist ein Ausschnitt aus einem solchen *Clip* zu sehen.

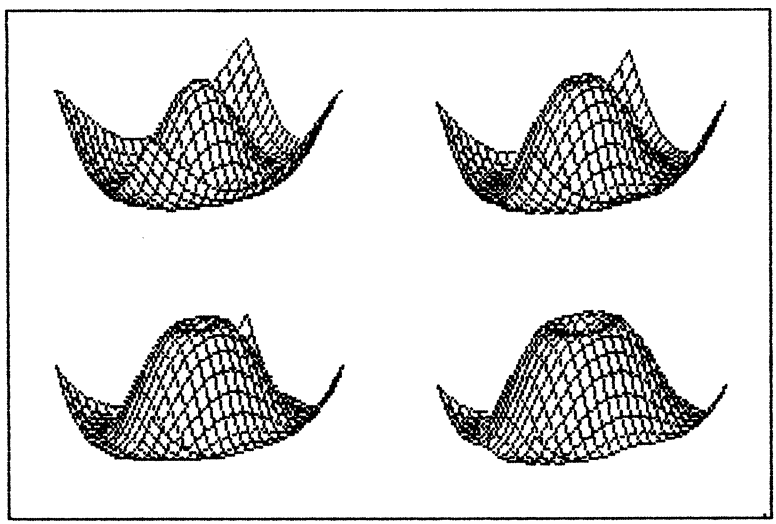


Bild 1.8: Ausschnitt aus einer bewegten 3D-Grafik

Laden Sie dazu das Programm **3D-CLIPS** und starten Sie es mit **RUN**. Es werden dann automatisch die Programme **BMC-$9000** und **NETZGRAFIK** (aus 64er 4/85) nachgeladen. Nehmen Sie dann die Programmdiskette heraus und legen Sie eine leere, formatierte Diskette ein.

Das Programm bietet Ihnen nun die Auswahl zwischen 15 *Clips*. Beginnen Sie vielleicht einmal mit dem Clip **BLUBB**, das auch in Abb. 1.8 dargestellt ist. Nach der Wahl der Grafik folgt die Frage **BMC?**. Beim BMC-Format (Bitmap-Compander) werden die Bitmaps in komprimierter Form auf Diskette aufgezeichnet, so daß Sie statt der üblichen 33 Blocks nur 10 bis 20 Blocks benötigen. Da Sie jedoch erst in Kapitel 4.3 erfahren, wie Bitmaps in diesem Format in den Hi-Eddi plus geladen werden, beantworten Sie die Frage zunächst einmal mit N.

Das Programm zeichnet nun die 4 Bitmaps für das Clip **BLUBB** und legt sie unter den Namen **BLUBB.1** bis **BLUBB.4** auf der Diskette ab. Am Ende wird noch der nötige Sequenzstring ausgegeben, den Sie sich bitte notieren.

Beantworten Sie die Frage **NOCH EIN CLIP** mit N, laden Sie wieder den Hi-Eddi plus und laden Sie dann die vier **BLUBB**-Bitmaps in die Bildschirmspeicher mit der entsprechenden Nummer, also **BLUBB.1** in Speicher 1 etc. Geben Sie dann mittels **SHIFT W** den zuvor notierten Sequenzstring ein, starten Sie dann mittels W den Trickfilm und staunen Sie!

Sie werden einige Zeit beschäftigt sein, wenn Sie alle 15 Clips, die das Programm **3D-CLIPS** erzeugen kann, ansehen wollen. Ich empfehle Ihnen, damit zu warten, bis Sie Kapitel 4.3 gelesen haben. Dort erfahren Sie nämlich, wie Sie die Bitmaps im komprimierten Format (BMC) laden können. Damit bringen Sie nicht nur mehrere Bitmaps auf eine Diskette als im normalen Format, sondern das Speichern und Laden geht auch viel schneller.

1.13 Erstellen einer Menütafel

Ich halte nicht allzu viel von einer menügesteuerten Befehlseingabe. Mir sind die direkt und schnell einzugebenden Tastaturbefehle lieber. Natürlich ist das meine subjektive Meinung. Dazu kommt noch, daß ich als *fortgeschrittener* Anwender auch meist recht schnell mit der Bedienung eines Programms zurechtkomme. Für einen Anfänger oder den "Otto-Normal-Verbraucher" sieht das sicherlich anders aus. Hier ist ein Menü, in dem alle verfügbaren Befehle zur Auswahl gestellt werden, sicherlich sehr nützlich. Doch dies war für mich noch kein ausreichender Grund, Hi-Eddi plus mit einer Menüeingabe zu versehen. Ich wollte mehr bieten und mit der folgenden Möglichkeit ist es mir vermutlich auch gelungen: Sie können damit eine eigene, selbst entworfene Menütafel erstellen. Wenn Ihnen das gelingt - Sie brauchen dazu keinerlei Programmierkenntnisse - dann besitzen Sie ein Programm, dessen Menü Sie selbst erstellt haben und das auch funktioniert!

Falls Sie daran kein Interesse haben sollten, können Sie diesen Abschnitt überspringen. Andernfalls wollen wir uns nun mit den Regeln beschäftigen, nach denen ein Menü aufzubauen ist.

Sehen Sie sich das *Originalmenü*, das in Abb. 1.9 dargestellt ist, genau an. Sicherlich fällt Ihnen dabei einiges auf:

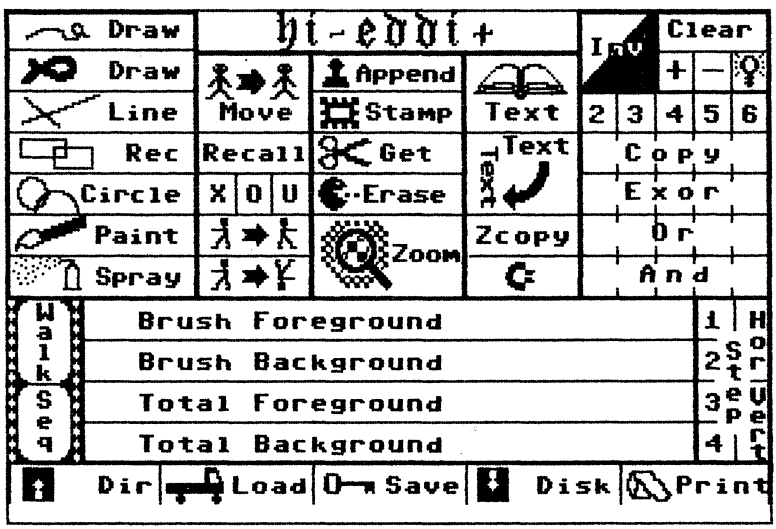


Bild 1.9: Das Menü des Hi-Eddi plus

Zunächst ist die ganze Tafel mit einem bestimmten Grundraster überzogen. Die Tafel ist nämlich in 20*12, also insgesamt 240 Quadrate eingeteilt. Ein solches 16*16 Punkte großes Quadrat ist gewissermaßen der Grundbaustein der Menütafel. Alle Befehlsfelder sind aus einem oder mehreren solcher Quadrate zusammengesetzt, es gibt kein Feld, das dieses Grundraster durchbricht. Wollen Sie also selbst eine Menütafel entwerfen, so müssen Sie sich ebenfalls an dieses Raster halten. Dagegen bleibt es Ihnen überlassen, wie viele solcher Quadrate Sie für einen Befehl zusammenhängen. Und es bleibt Ihnen auch freigestellt, welche Befehle Sie überhaupt ins Menü aufnehmen. Im Originalmenü sind alle Befehle enthalten, mehr können Sie also in Ihr eigenes Menü nicht aufnehmen. Was im Originalmenü noch fehlt, sind die Tabulatoren und die Cursortasten, die wohl kaum sinnvoll wären. Das gleiche gilt auch für die Pfeil-nach-links-Taste, die u.a. im Farb-Betrieb als Korrekturtaste dient und alle Farbveränderungen seit dem letzten Bildschirmwechsel rückgängig macht (siehe Kapitel 1.8). Da jedoch der Aufruf der Menütafel selbst ein Bildschirmwechsel ist, wäre dieser Befehl im Menü sinnlos.

Die zweite Regel lautet: Die unterste Zeile, entsprechend acht Punktreihen, muß frei bleiben! Der Grund dafür ist diesmal ausnahmsweise nicht die Farbauflösung, wie beim Walk-Befehl, sondern diese Zeile wird für Steuerinformationen benötigt. Denn irgendwoher muß Hi-Eddi plus wissen, was zu tun ist, wenn an einer bestimmten Stelle des Bildschirms im Menümodus der Knopf gedrückt wird. Doch um die Erstellung dieser Steuerinformationen brauchen Sie sich nicht zu kümmern, diese Arbeit nimmt Ihnen das Programm **MENUEMAKER** auf der beiliegenden Diskette ab.

Schließlich müssen Sie bei Ihrem Entwurf noch beachten, daß einige Befehle im Menü von den entsprechenden Tastaturbefehlen etwas abweichen: Das gilt für die Farb- und die Verknüpfungbefehle. Bei den Farbbefehlen wird im Menü die Farbe direkt angewählt, während sie bei der Tastatureingabe nur durchgeschaltet werden konnte. Bei den Verknüpfungsbefehlen schließlich erfolgt die Verknüpfung ganzer Bildschirme, für die bei der Tastatureingabe zwei Eingaben benötigt wurden, mit nur einem Befehlsfeld. Die drei Felder **O**, **X** und **U** im Block für die Verschiebebefehle im Originalmenü dienen nur zur Verknüpfung von Ausschnitten.

Soweit die Regeln, die Sie beim Entwurf Ihrer eigenen Menütafel beachten müssen. Sie malen also zunächst eine Menütafel nach Ihren eigenen Vorstellungen, aber den oben beschriebenen Regeln entsprechend. Diese Arbeit können Sie im Farb- oder Menübetrieb erledigen. Bei der Einteilung des Bildschirms in das 16-Punkte-Raster hilft Ihnen wieder die programmierbare Schrittweite: Sie können entweder die Schrittweite 16 für horizontal und vertikal auf einer Funktionstaste, zum Beispiel F5, programmieren oder Sie verwenden die bereits voreingestellte Schrittweite von 8 Punkten auf F3. Dabei müssen Sie natürlich jeweils 2 Schritte in jede Richtung gehen, um wieder auf das Raster zu kommen.

Sind Sie mit dem Malen fertig, dann fehlen noch die Steuerinformationen. Dazu speichern Sie Ihr Werk unter dem Namen **MENUE.PIC** als Farbbild auf Diskette ab. Dann starten Sie den Computer neu und laden das Programm **MENUEMAKER** von derbeiliegenden Diskette. Dieses Programm gibt Ihnen nun Hinweise zur weiteren Vorgehensweise, ich will es jedoch hier nochmals ausführlicher darstellen. Zunächst werden Sie aufgefordert, die Diskette mit Ihrem eben erstellten Menübild **MENUE.PIC** einzulegen. Dann folgt eine kurze Beschreibung dessen, was im folgenden zu tun ist: Es erscheint ein 16*16 Punkte großer Cursor - also genau den Maßen des Menü-Grundbausteines entsprechend, der auf jedem der insgesamt 240 Quadrate wissen will, welcher Befehl an dieser Stelle der Menütafel steht. Nehmen wir als Beispiel die Original-Menütafel: Das erste Quadrat, auf dem der Cursor erscheint, ist das ganz links oben. Das Quadrat dort gehört zu dem Befehlsfeld für **Draw** mit dünnem Pinsel, also tippen wir **D** ein. Dann rutscht der Cursor um ein Quadrat, also 16 Punkte, nach rechts und wartet wieder auf eine Eingabe. Da wir immer noch auf dem Befehlsfeld für **Draw** stehen, tippen wir nochmals **D**. Das geht so weiter, bis der Cursor auf dem ersten Quadrat neben dem Draw-Befehlsfeld steht. Dort steht jetzt eine Überschrift, es ist also gar kein Befehlsfeld. Deshalb tippen wir jetzt, wie in der Kurzanleitung des Menümakers angegeben, die RETURN-Taste. Damit erzeugen wir einen *Nichts-tun-Befehl*. Wenn wir also später im Menü den Cursor an diese Stelle fahren und den Knopf drücken, passiert gar nichts. Nach dieser Methode arbeiten Sie sich durch das ganze Menü und drücken überall die dem jeweiligen Befehl entsprechende(n) Taste(n). Wenn Sie zum Beispiel in der zweiten Zeile im Originalmenü angekommen sind, steht der Cursor auf dem Befehlsfeld für **Draw** mit dickem Pinsel. Entsprechend der Bedienungsanleitung tippen Sie hier **SHIFT D** ein. Eine Sonder-

stellung haben, wie oben schon angedeutet, die Farb- und Verknüpfungsbefehle. Bei den Verknüpfungsbefehlen, also **O**, **X** und **U**, fragt der Menümaker, mit welchem Bildschirm verknüpft werden soll. Falls mit gar keinem Bildschirm verknüpft werden soll, sondern mit einem Ausschnitt, so geben Sie **0** ein. Bei den Farbbefehlen fragt der Menümaker nach der Nummer der Farbe, die Sie folgender Tabelle entnehmen können:

Nummer	Farbe	Nummer	Farbe
0	schwarz	8	orange
1	weiß	9	braun
2	rot	10	hellrot
3	cyan	11	dunkelgrau
4	violett	12	mittelgrau
5	grün	13	hellgrün
6	blau	14	hellblau
7	gelb	15	hellgrau

Haben Sie sich bei einem Feld vertippt, so können Sie den Cursor mittels DEL-Taste auch zurückbewegen.

Wenn Sie das ganze Menü durchgearbeitet haben, dann legen Sie die Diskette ein, auf der Sie Ihre eigene Menütafel speichern wollen. Der Menümaker wird darauf dann die fertige Menütafel unter dem Namen MENUE ablegen.

Um diese eigene Menütafel zu verwenden, müssen Sie nach dem Start von Hi-Eddi plus, nachdem der ganze Vorspann abgelaufen ist und die Fragen zur Betriebsart erschienen sind, die Programmdiskette aus dem Laufwerk nehmen und dafür die Diskette mit Ihrem eigenen Menü einlegen. Dieses wird dann geladen und steht Ihnen nun zur Verfügung.

1.14 Hi-Eddi plus ausschalten

Folgerichtig als letztes Kapitel in der Bedienungsanleitung kommt nun der Befehl zum Ausschalten von Hi-Eddi plus. Damit Sie dazu nicht den Computer ausschalten müssen, geben Sie einfach den folgenden Befehl ein:

C= Q Quit - Hi-Eddi plus ausschalten

Der Computer wird damit wieder in den Einschaltzustand versetzt.

2 Anwendungen

In diesem Kapitel möchte ich Ihnen anhand mehrerer konkreter Beispiele schrittweise zeigen, wie Sie bei der Erstellung Ihrer Zeichnungen am besten vorgehen. Natürlich sind das keine festen Richtlinien, die eingehalten werden müssen. Sie wissen ja: Viele Wege führen nach Rom (hier ist ausnahmsweise die Hauptstadt des römischen Reiches gemeint und nicht der Nur-Lese-Speicher!). Jeder kann also bei der Erstellung von Zeichnungen nach seinem eigenen Geschmack vorgehen. Dennoch können Sie aus diesem Kapitel eine ganze Menge lernen. Zunächst werden Sie Anregungen erhalten, was man mit Hi-Eddi plus alles anstellen kann, und danach werde ich eine Reihe von Tips und Tricks verraten, auf die man selbst nicht ohne weiteres kommt. Sie werden an praktischen Beispielen erörtert, bei denen sie besonders nutzbringend einzusetzen sind. Worum es genau geht, werden Sie aus den Kapitelüberschriften ersehen, die neben der dargestellten Anwendung auch den Themenbereich, zu dem ich Tips gebe, enthalten.

An der Auswahl der Beispiele werden Sie unschwer erkennen, daß ich Elektrotechniker bin und mein Spezialgebiet innerhalb des weiten Feldes der Elektrotechnik die Datenverarbeitung ist. Deshalb werden Sie in den Beispielen nichts finden, was mit Drehstrom zu tun hat. Und eine saubere, den zahlreichen Normen entsprechende Konstruktionszeichnung von einem Hubraumdeckelrückholmechanismus dürfen Sie erst recht nicht von mir erwarten. Das soll natürlich nicht heißen, daß Hi-Eddi plus dafür nicht geeignet wäre. Sie können damit alles zeichnen, Sie müssen nur wissen, wie es aussehen soll!

2.1 Schaltpläne und das Arbeiten mit Construction Sets

Bei der Beschreibung der Diskbefehle haben Sie sicherlich schon das Construction Set für Schaltpläne von analogen Schaltungen und das daraus erstellte Beispielschaltbild - es handelt sich dabei um einen Telefon-Mithörverstärker - betrachtet. Daneben befindet sich auf der Diskette noch ein Construction Set für Digitalschaltungen (**DIGITAL.CS**) und auch dazu ein Beispiel (**FLIPFLOP.PIC**), nämlich die Nachbildung eines Master-Slave-JK-Flip-Flops mit RS-Flip-Flops (Abb. 2.1 und 2.2).

Diese beiden Construction-Sets sind sicherlich nicht für jeden der Weisheit letzter Schluß. Aber im Gegensatz zu Programmen, die fest auf einen bestimmten Verwendungszweck zugeschnitten sind, wie etwa auf das Zeichnen von Schaltungen, ist Hi-Eddi plus ungleich flexibler. Sie können die Formen der Bauteile ändern, können Elemente, die Sie nicht brauchen, aus dem Set entfernen und andere dafür einsetzen.

Wir wollen jedoch zunächst ein Problem aufgreifen, für das der Analog-Construction-Set ohne Änderungen verwendet werden kann.

Ich selbst pflege ein Construction-Set immer in einen der letzten Speicher, also Nummer 6 oder 7 zu laden. Gezeichnet wird dann auf Bildschirm 1. Die dahinter liegenden Speicher, also 2, 3 oder 4, dienen mir als Sicherheitskopien. Das gilt natürlich nur, wenn ein Bildschirm für die eigentliche Zeichnung reicht. Einer der ganz großen Vorteile von Hi-Eddi plus ist es, Pläne zu erstellen, die aus mehreren Bildern zusammengefügt werden. Doch mehr dazu in einem späteren Kapitel; zunächst soll uns eine einteilige Zeichnung genügen.

Laden Sie also den Analog-Construction-Set, wie schon gesagt, am besten in einen der hinteren Speicher.

Wir werden wieder ein Flip-Flop zeichnen, und zwar das in Abb. 2.3 dargestellte T-Flip-Flop. Abb. 2.3 ist absichtlich eine recht ungenau gezeichnete Freihandskizze, denn wir wollen den sauberen Plan ja erst erstellen. Eigentlich braucht man eine solche Skizze gar nicht, sondern kann gemäß seinen Vorstellungen die Zeichnung direkt eingeben. Da ich Ihnen jedoch schlecht beschreiben kann, was mir vorschwebt, bediene ich mich der Freihandskizze.

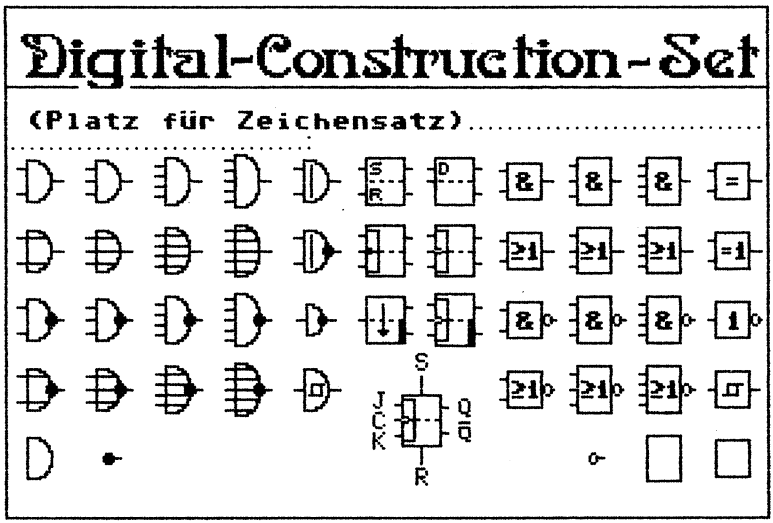


Bild 2.1: Digital Construction-Set auf der Diskette

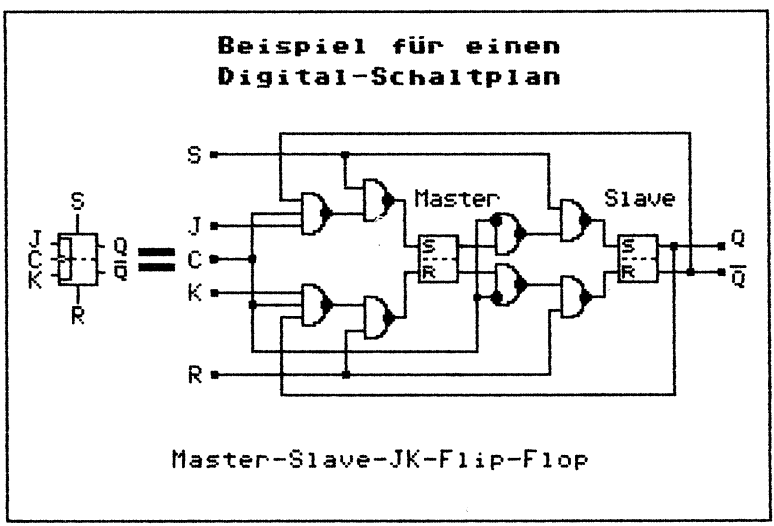


Bild 2.2: Ein weiteres Beispiel von der Diskette

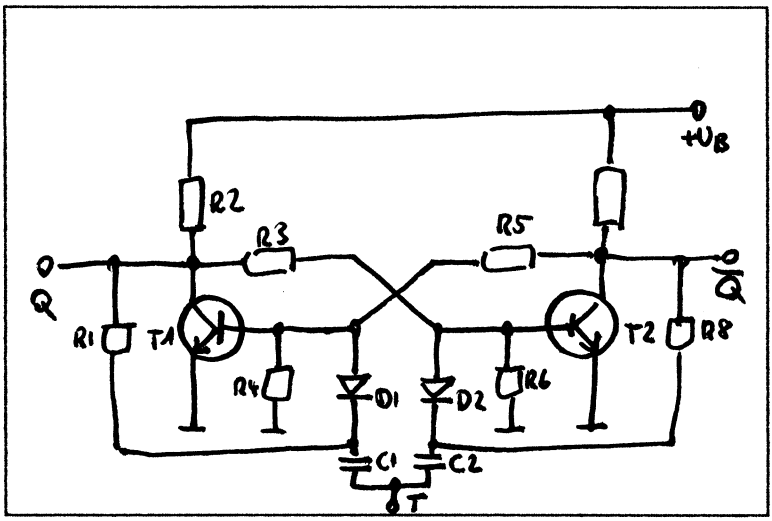


Bild 2.3: Eine Freihandskizze als Entwurf

Ich finde es sinnvoll, mit den größten Bestandteilen anzufangen. In diesem Fall wären das die beiden Transistoren. Also wählen wir die Bildschirmseite mit dem Construction-Set an und schalten mit G in den Get-Modus. Nun bewegen wir den Rahmen auf den Transistor (den ersten in der Reihe, nicht irgendeinen der FETs) und drücken den Knopf. Beim Positionieren des Rahmens geht es nicht so genau zu. Wir sollten nur darauf achten, daß der *Körper*, also der Kreis des Transistorsymbols, voll vom Rahmen erfaßt ist, wir jedoch kein benachbartes Symbol berühren. Wenn die Anschlußleitungen nicht ganz erfaßt werden, macht das gar nichts! Wie Sie sehen, haben wir den Transistor jetzt erfaßt und können ihn an verschiedenen Stellen positionieren.

Wählen Sie nun den Bildschirm, auf dem Sie zeichnen wollen, an. Bevor wir jedoch mit dem Übertragen des ersten Elementes beginnen, sollten wir uns über die Einteilung des Bildschirms klar werden. Schließlich soll das Bild einigermaßen in der Mitte sitzen! Falls wir uns tatsächlich einmal verschätzt haben, ist das allerdings halb so schlimm: Mit dem Move-Befehl können wir den ganzen Bildinhalt wieder verschieben und damit eine Fehlplanung korrigieren.

Im vorliegenden Fall ist die Einteilung nicht sehr schwierig, da uns ein symmetrisches Bild vorliegt. Wir werden den rechten Transistor in der Waagerechten bei etwa zwei Drittel bis drei Viertel der Bildschirmbreite und in der Senkrechten in der Mitte plazieren. Gehen Sie mit dem Cursor dorthin und drücken Sie den Knopf! Sollte Ihnen das Ergebnis nicht gefallen, so tippen sie **E**. Der Rahmen ist nun leer und wird bei Knopfdruck zum Radiergummi, mit dem Sie den Transistor weglöschen können. Danach schalten Sie wieder mittels **A** auf Append und können so den Transistor erneut positionieren.

Beim linken Transistor gibt es nun Schwierigkeiten: Sein Emitter zeigt nach links unten. Dies ist jedoch kein Problem für Hi-Eddi plus! Wählen Sie durch Drücken der Space-Taste (oder über das Menü, falls Sie im Menü-Betrieb arbeiten) den Sprite-Editor an. Sie erinnern sich: Aus den Zeichenmodi und einigen anderen Modi kommt man mittels Space in die Zoom-Funktion. Nicht nur aus den Sprite-Modi, sondern auch aus dem Append-Modus, in dem wir uns gerade befinden, kommt man in den Sprite-Editor. Dort sehen Sie jetzt den Transistor überdimensional vergrößert. Drücken Sie nun **M** und schon haben wir, was wir wollten! Sie können auch gleich noch versuchen, den Transistor mittels **T** und **M** an andere Positionen zu plazieren, zum Beispiel mit dem Emitter nach oben; und mit dem Befehl **R** bekommen Sie die Basis nach oben oder unten. Aber wahrscheinlich haben Sie jetzt den Transistor abgeschnitten! Wie Sie beim Rotate-Befehl nachlesen können, gehen die drei rechten Spalten des Sprites beim Rotieren verloren. Wenn Sie also einmal ein Element rotieren müssen, so achten Sie darauf, daß Sie es bereits beim Aufnehmen aus dem Menü ziemlich weit links in den Rahmen bekommen, so daß rechts drei Spalten frei bleiben!

Wenn es Ihnen gelungen ist, den Transistor wieder in der richtigen Lage im Sprite zu haben, dann wollen wir ihn ins Bild einsetzen. Und zwar bitte in derselben Höhe wie den rechten! Dazu fahren Sie den Transistor an den rechten heran und plazieren ihn so, daß sich die beiden Basisleitungen - oder die beiden Kreise - decken. Dann fahren Sie exakt horizontal d.h. nach links. Ich hoffe, Sie haben jetzt den Joystick in der Hand. Falls Sie das Pad benutzen, legen Sie es zur Seite und schließen den Joystick an Port 2 an, denn wie in Abschnitt 1.2 schon erwähnt, den Cursor exakt horizontal oder vertikal zu bewegen, geht mit dem Joystick viel besser als mit dem Pad. Steht der Cursor nun richtig - ist er etwa genauso weit vom linken Rand wie der rechte Transistor vom rechten Rand entfernt - dann drücken Sie den Knopf.

Als nächstes holen wir uns den Widerstand aus dem Construction-Set. Worauf muß man jetzt achten? Der Widerstand muß möglichst weit links im Rahmen liegen, denn wir müssen ihn rotieren! Dabei können Sie ruhig ein Stückchen des linken Anschlußdrahtes abzwicken, die Leitungen müssen wir ja sowieso neu zeichnen.

Wenn Sie wieder in Ihrem Arbeitsbildschirm sind, setzen Sie zunächst die beiden waagerechten Widerstände R3 und R5, über die Transistoren und ein bißchen zur Mitte hin, natürlich wieder in gleicher Höhe, wie wir das bereits bei den Transistoren getan haben. Dann müssen Sie den Widerstand im Sprite-Editor rotieren und können nun die restlichen Widerstände ins Bild einsetzen.

Genauso verfahren Sie mit den Dioden und Kondensatoren. Achten Sie dabei darauf, die Dioden etwa in dieselbe Höhe zu setzen wie die Widerstände R4 und R6 und zwischen den Dioden und den Kondensatoren genug Platz zu lassen, damit die Massepunkte und die Leitungen zu den Widerständen R1 und R8 noch Platz haben. Auch zwischen den Dioden und den benachbarten Widerständen sollte genug Platz für die Beschriftung sein.

Nun wäre es wohl an der Zeit, die erste Sicherheitskopie zu machen, da Sie nie wissen, welchen Fehler Ihnen unterlaufen werden. Wählen Sie dazu einen anderen Bildschirm an, zum Beispiel Nummer 2, wenn Sie in Nummer 1 zeichnen. Geben Sie dort **C= 1** ein und schon ist die Sicherheitskopie erstellt und Sie können wieder in Ihren Arbeitsbildschirm zurückkehren. Vergessen Sie nicht, auch weiterhin zwischendurch immer wieder eine Sicherheitskopie zu erstellen!

Nun könnten Sie gleich noch die Massepunkte setzen oder, was ich bevorzuge, erst einmal die Linien ziehen. Zum Linienziehen können Sie entweder den Draw- oder den Line-Befehl verwenden. Ich benutze den Line-Befehl (also **L** eintippen), da man dabei nicht ständig den Knopf gedrückt halten muß und eine falsche Steuerung des Cursors nicht gleich zu einem Fehler in der Zeichnung führt. Erst die endgültige Position des Cursors entscheidet über den Verlauf der Linie, nicht der Weg dorthin.

An einigen Stellen, zum Beispiel bei der langen Leitung von R1 zu C1, werden Sie nicht sicher sein, wie lang die Linie von R1 nach unten bzw. von C1 nach links gezeichnet werden muß, damit der Anschluß paßt. Geben Sie sich hier nicht allzuviel Mühe, denn es ist später mit der Zoom-Funktion schnell korrigiert, wenn eine Linie mal ein paar Pünktchen zu kurz oder zu lang geworden ist.

Die beiden schrägen Linien in der Mitte zeichnen Sie am besten im 45-Grad-Winkel, was besonders einfach geht: gleich viele Punkte in jede Richtung. Verwenden Sie dazu auch die Cursortasten, womit Sie immer gleich 8 Punkte in einem Schritt zurücklegen (vorausgesetzt, Sie haben noch F3 angewählt und diese Schrittweiten nicht verändert).

Sind die Linien alle aufgezeichnet, folgen die Massepunkte nach bewährter Manier - mit Get aus dem Construction-Set holen und in gleicher Höhe einsetzen - und anschließend die Knotenpunkte, also die Punkte, an denen sich Leitungen verzweigen. Dafür ist der Draw-Modus mit dem dicken Pinsel bestens geeignet. Geben Sie also **SHIFT D** ein, setzen Sie den Cursor auf die Knotenpunkte und drücken Sie den Knopf.

Nun können Sie noch die Stellen reparieren, an denen zuvor die Linien nicht exakt zusammenpaßten. Setzen Sie dazu den Cursor an die entsprechenden Stellen und drücken Sie die Space-Taste (oder wählen Sie die Zoom-Funktion im Menü an). Da wir uns im Augenblick in einem Zeichenmodus, nämlich dem Draw-Modus befinden, kommen wir jetzt nicht mehr in den Sprite-Editor, sondern in die Zoom-Funktion. Dort sehen Sie die fehlerhaften Stellen sehr deutlich und können auf Knopfdruck fehlende Punkte setzen oder mit SHIFT + Knopfdruck überflüssige Punkte beseitigen.

Zuletzt fehlt noch die Beschriftung. Wenn Ihnen die fette Commodore-Schrift nicht zusagt, laden Sie einen der Pica-Zeichensätze von der Diskette, am besten ins Construction-Set, da dort spezieller Platz dafür vorgesehen ist. Wie das geht und wie Sie die erforderliche Schrittweitenänderung vornehmen, können Sie in der Bedienungsanleitung bei der Erklärung der Diskbefehle und der Schrittweitenprogrammierung ausführlich nachlesen.

Zur Beschriftung selbst gibt es nicht mehr viel zu sagen. Wählen Sie mit **T** den Textmodus an, plazieren Sie den Cursor mittels Joystick genau und tippen Sie dann den gewünschten Text ein. Natürlich können Sie auch mittels **CTRL Z** die Schriftrichtung drehen und aufwärts oder abwärts schreiben, sofern dies sinnvoll ist.

Damit haben wir unsere erste technische Zeichnung erstellt! Ich hoffe, Sie hatten keine Schwierigkeiten dabei und sind mit dem Ergebnis zufrieden. Ich zeige Ihnen bewußt kein fertiges Bild, denn Sie sollten ja nicht ein fertiges Muster nachvollziehen, sondern nur aufgrund meiner Anleitung und einer ungenauen Skizze zu einem Bild kommen, von dem Sie nicht schon vorher wußten, wie es aussehen wird.

In den folgenden Kapiteln werde ich die einzelnen Schritte nicht mehr so ausführlich beschreiben, außer es handelt sich um besondere Fertigkeiten, die noch nicht besprochen wurden.

2.2 Die olympischen Ringe und die programmierbare Schrittweite

Als nächstes wollen wir die olympischen Ringe zeichnen, da sie sich hervorragend dazu eignen, die programmierbare Schrittweite zu erklären. Für dieses Feature ein eigenes Kapitel zu reservieren, halte ich durchaus für gerechtfertigt, denn aus den Anfragen nach der Veröffentlichung des alten Hi-Eddi habe ich erfahren, daß viele Anwender mit dieser leistungsfähigen Einrichtung nichts anzufangen wissen.

In Abb. 2.5 habe ich die Maße angegeben, nach denen wir die Ringe zeichnen wollen. Die Zahlen geben dabei den Abstand in Punkten an. Das Ergebnis soll dann Abb. 2.4 gleichen.

Dazu wählen wir zunächst die Funktionstaste F7 an, also **C= F7** eintippen. Ich benutze F7 immer für die größten Schrittweiten. Die horizontale Schrittweite (**H**) stellen wir auf 160 und die vertikale (**V**) auf 75 ein. Nun setzen wir den Cursor mit **HOME** ins linke obere Eck und fahren ihn mittels Cursortasten einen Schritt nach unten und nach rechts.

Jetzt stehen wir im Mittelpunkt des mittleren Ringes. Dieser Punkt soll uns als Ausgangspunkt für alle weiteren Aktionen dienen. Wir speichern ihn deshalb als Tabulatorposition auf einer Funktionstaste, zum Beispiel F5. Dazu tippen Sie **SHIFT F5**. Die Taste F7 nehmen wir nicht, da sie der Circle-Befehl bereits benutzt.

Nun wählen wir mittels **C= F5** die Funktionstaste 5 an und programmieren darauf die Schrittweiten 45 und 40 für horizontal bzw. vertikal.

Diese Schrittweiten sind so gewählt, daß wir mit den Cursortasten exakt die Mittelpunkte aller fünf Kreise ansteuern können. Allerdings müssen Sie aufpassen, daß Sie nicht an den Rand stoßen oder den Cursor mittels Joystick bewegen, denn dann wäre unser Cursor aus dem Takt geraten. Es macht aber auch nichts, wenn das doch einmal passiert: F5 drücken, und wir sind wieder am Ausgangspunkt und somit *synchronisiert*.

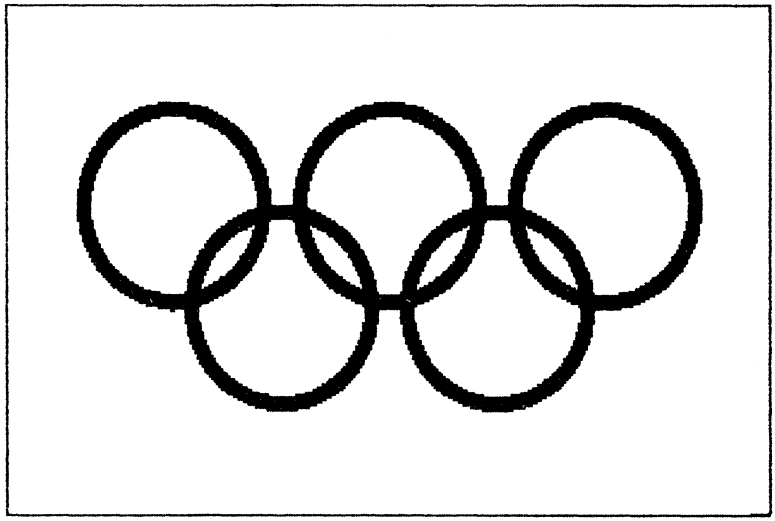


Bild 2.4: Die olympischen Ringe

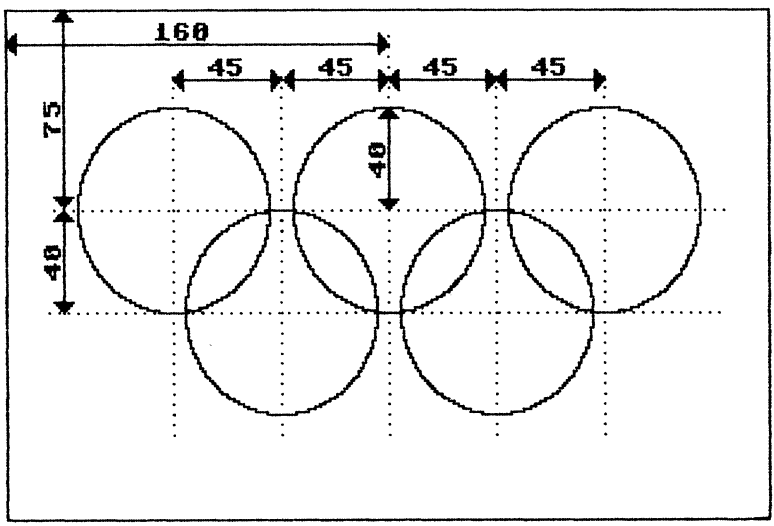


Bild 2.5: Die für die Ringe verwendeten Maße

Wählen Sie nun den Circle-Befehl an (C), plazieren Sie den Cursor im Mittelpunkt eines Kreises, drücken Sie den Knopf oder die RETURN-Taste, fahren Sie den Cursor um einen Schritt nach oben oder unten - denn unsere vertikale Schrittweite ist zugleich der Radius der Kreise - und drücken Sie den Knopf nochmals. Schon haben Sie den ersten Kreis! Das gleiche machen Sie jetzt für alle fünf Kreise, und danach erstellen Sie eine Sicherheitskopie!

Damit wir richtige Ringe bekommen, brauchen wir noch einen zweiten, inneren Kreis. Die Zwischenräume füllen wir dann mit dem Paint-Befehl aus. Für diesen Innenkreis habe ich einen Radius von 35 gewählt. Stellen Sie diesen Radius als vertikale Schrittweite statt der 40 von vorher ein! Setzen Sie mit F5 den Cursor wieder an den Ausgangspunkt und zeichnen Sie die Innenkreise für die oberen drei Ringe. Um zu den unteren Ringen zu kommen, müssen wir die vertikale Schrittweite kurzzeitig wieder auf 40 erhöhen. Dann verkleinern Sie sie wieder auf 35 und zeichnen die Innenkreise für die unteren beiden Ringe.

Bevor Sie nun ans Ausfüllen gehen, machen Sie sich unbedingt eine Sicherheitskopie! Das sollten Sie immer vor Anwendung des Paint-Befehls tun, denn dieser Befehl ist sehr gefährlich! Allzuleicht setzt man den Cursor falsch und im Nu ist etwas ausgefüllt, was man gar nicht ausfüllen wollte. Und bis man reagiert und die STOP-Taste drückt, ist meistens schon eine Menge kaputt.

Wenn Sie mit dem Ausfüllen fertig sind, haben Sie die exakte Nachbildung der olympischen Ringe vor Augen. Mit welchem anderen Programm hätten Sie das so schnell und genau geschafft?

Und weil wir gerade bei Kreisen und beim Vergleichen sind: Vergleichen Sie einmal die schönen, runden Kreise, die Hi-Eddi plus zeichnet mit dem, was andere Grafikprogramme produzieren. Die Kreise von Simon's BASIC zum Beispiel erinnern mehr an eine Freihandzeichnung. Aber auch die Kreise anderer Programme, die auf den ersten Blick recht rund wirken, erweisen sich bei näherem Hinsehen - insbesondere unter der Lupe, sprich Zoom-Funktion - als recht löchrig.

Dafür muß ich allerdings bekennen, daß Hi-Eddi plus keine Ellipsen darstellen kann. Das wäre eigentlich eine gute Idee für eine Erweiterung (siehe Kapitel 5).

2.3 PAPs und der Move-Befehl

Ein PAP, Programmablaufplan, Programmlaufplan, Flußdiagramm oder wie immer man es auch bezeichnet, ist eine Form, Programme so darzustellen, daß man sie möglichst leicht überblickt und versteht. Dabei werden die einzelnen Programmschritte in Kästen geschrieben, deren Form bereits einiges über die Funktion eines Elementes aussagt. So stehen zum Beispiel Ein- und Ausgaben in Parallelogrammen, Verzweigungen in Rauten, Programmstart und Ende in Ovalen und alles übrige in Rechtecken. Diese Kästen werden durch Pfeile, die den Ablauf des Programms darstellen, verbunden. Abb. 2.6 gibt ein Beispiel für einen Programmablaufplan.

Uns interessiert natürlich wieder, wie man so etwas mit Hi-Eddi plus zeichnet. Das ist gar nicht so schwierig, wie Sie vielleicht vermuten. Wir wollen dabei auch wieder einen Weg finden, das Ganze unter Ausnutzung der Möglichkeiten, die uns Hi-Eddi plus bietet, so einfach wie möglich zu machen.

Die Rechtecke und Pfeile sind sicherlich kein Problem. Die Rechtecke macht der Rectangle-Befehl, die Linien der Line-Befehl und die Pfeilspitzen erstellen wir als Sprite und stempeln sie mittels Append ins Bild. Auch der Text bereitet keine Probleme, wobei wir bei beengten Platzverhältnissen wieder auf die engere Pica-Schrift ausweichen können.

Das Problem sind das Parallelogramm, die Raute und das Oval. Schauen wir uns also an, wie man diese Gebilde erstellt.

Das Parallelogramm erstellen wir mit Hilfe einer alten Bekannten: der programmierbaren Schrittweite. Wir wählen zum Beispiel 20 Punkte als Höhe und 80 als Länge einer Waagerechten, wobei die obere Waagerechte gegenüber der unteren um 10 verschoben sein soll. Also programmieren wir als horizontale Schrittweite 10 und als vertikale 20 Punkte (oder auch 10, damit es einheitlich ist). Die Waagerechte ergibt sich damit durch 8 Schritte. Alles weitere geschieht jetzt mittels Line.

Ebenso geht es bei der Raute. Für eine Höhe von 40 und eine Breite von 60 beispielsweise brauchen wir nicht einmal unsere zuvor eingestellten Schrittweiten zu ändern.

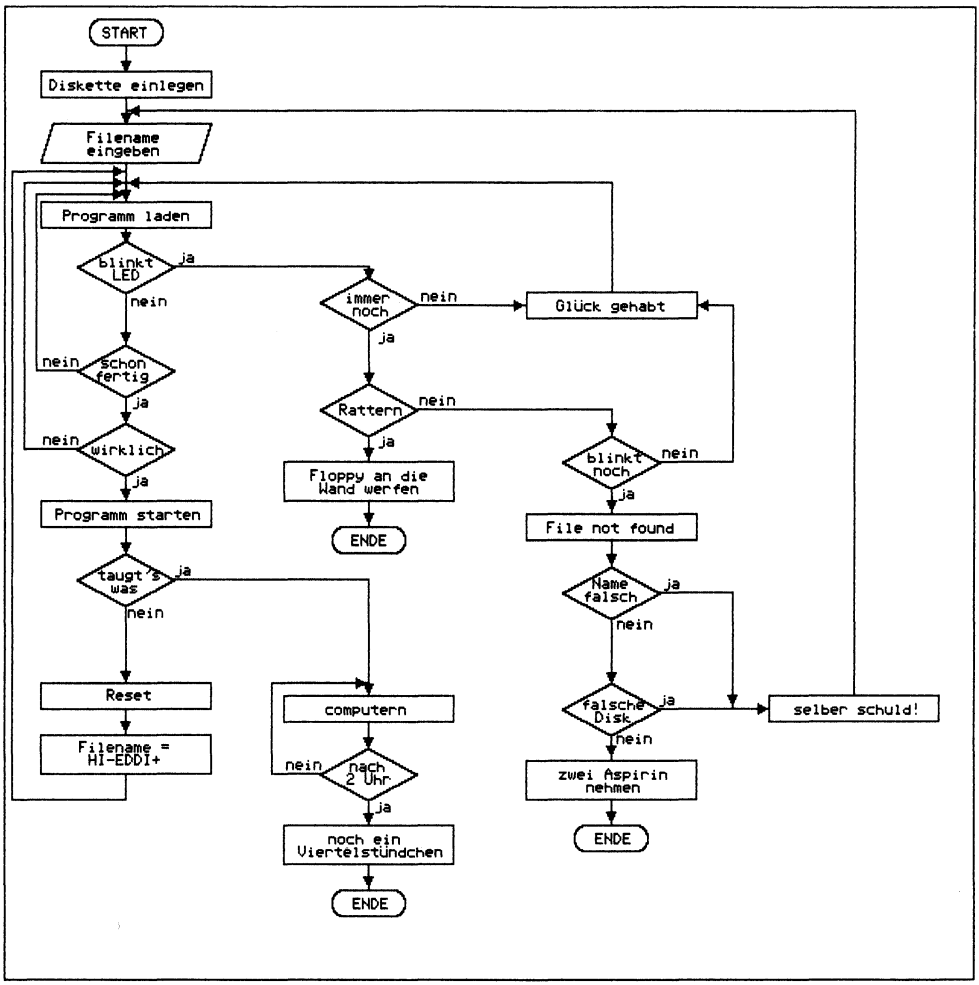


Bild 2.6: PAP für: Programm von Disk laden

Auch das Oval gestalten wir noch mit 10 als vertikaler und horizontaler Schrittweite. Zeichnen Sie zwei Kreise mit Radius 10 im Abstand von 40 Punkten von Mittelpunkt zu Mittelpunkt, natürlich in gleicher Höhe. Schalten Sie dann auf Erase (E) und radieren Sie die inneren Hälften der Kreise weg. Die beiden übrigen Halbkreise werden durch zwei Linien (L) verbunden.

So weit, so gut. Nun wollen wir diese Prozedur aber nicht jedesmal vornehmen. Vielleicht haben wir sogar einige Blöcke, deren Inhalte gleich oder fast gleich sind; dann wäre es schön, wenn man diese Blöcke nur einmal zu zeichnen brauchte und sie anschließend kopieren könnte. Dieses Problem stellt sich auch bei vielen anderen Anwendungen, wie den Hydraulikplänen, Blockschaltbildern oder auch elektrischen Schaltplänen. Sehen Sie sich dazu auch die ALU in Abb. 2.7 an, die Ihnen vielleicht schon aus 64'er bekannt ist: Die vier Gatter-Ansammlungen auf der linken Seite sind identisch.

Für den Sprite-Rahmen sind solche Gebilde zu groß. Wir müssen also zum Move-Befehl greifen. Doch der arbeitet nur im 8-Punkte-Raster, was scheinbar ein Nachteil ist. Scheinbar! Denn könnte man punktgenau verschieben, würde man nur äußerst schwer "treffen". Mit großer Wahrscheinlichkeit läge man ein oder zwei Punkte daneben. Sich dagegen gleich um 8 Punkte zu verschätzen ist schon unwahrscheinlicher.

Am besten wäre es natürlich, wenn man sehen könnte, was man verschiebt, wie bei Append. Aber für Objekte, die über die Maße eines Sprites hinausgehen, ist das sehr schwierig. Man könnte dann nicht mehr mit Sprites arbeiten, sondern müßte viel im Grafikbild herumarbeiten. Ich wage zu behaupten, daß das auf einem C64 nur unter extremem Aufwand möglich wäre, der dann die anderen Fähigkeiten von Hi-Eddi plus sehr einschränken würde. Meines Wissens gibt es auch kein Programm, das eine derartige Möglichkeit bietet. Einige Programme stellen den verschobenen Bereich als Rahmen dar, der jedoch nicht viel bringt, da er auch nicht zeigt, wo zum Beispiel die Anschlüsse eines Blocks liegen.

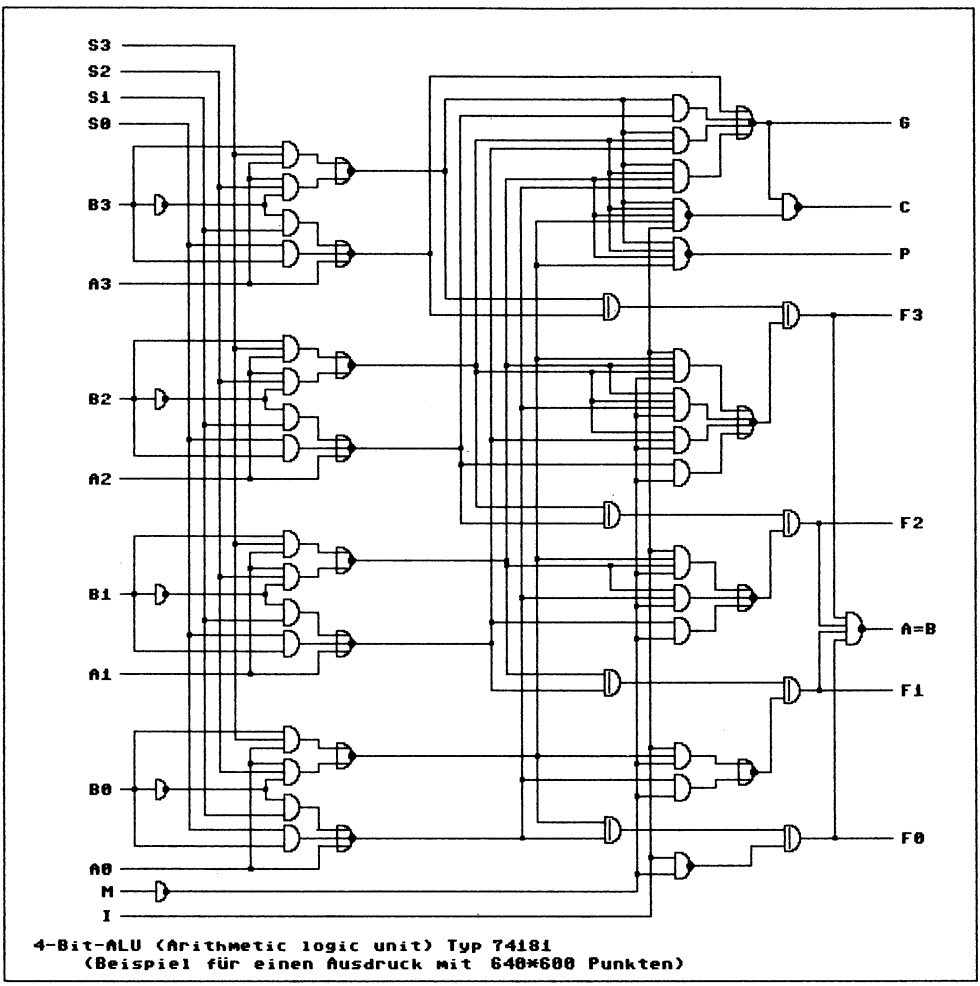


Bild 2.7: Ein Beispiel aus der Zeitschrift 64'er

Doch kehren wir zurück zum Move-Befehl, der, wie Sie wissen, im 8-Punkte-Raster arbeitet. Dabei könnte vor allem folgendes Problem auftreten: Wir verschieben einen Block und stellen fest, daß er um 4 Punkte zu weit links liegt. Veschieben wir ihn mit Move nach rechts, dann liegt er 4 Punkte zu weit rechts. Dazu darf es gar nicht erst kommen! Wir müssen bereits bei der Erstellung der einzelnen Blöcke darauf achten, daß sie alle im 8-Punkte-Raster zusammenpassen. Dabei hift uns, wie so oft, die programmierbare Schrittweite!

Wählen Sie die Funktionstaste F3 an, auf der die Schrittweite 8 für horizontal und vertikal gespeichert ist (falls Sie sie zwischendurch verändert haben, stellen Sie sie wieder auf 8 ein). Wenn wir nun den Cursor in die HOME-Position fahren und dann nur mittels Cursortasten auf dem Bildschirm umherbewegen, wissen wir, daß der Cursor immer in der linken oberen Ecke eines 8*8-Punkte-Feldes steht. Allerdings darf er nicht an den rechten oder unteren Rand stoßen, denn dann käme er aus dem Tritt und wir müßten ihn wieder von der HOME-Position aus starten.

Als konkretes Übungsbeispiel dient uns der Mini-PAP in Abb. 2.8. Wir wollen ihn aber nicht direkt zeichnen, sondern die Elemente einzeln erstellen, wie in Abb. 2.9 zu sehen ist und ihn daraus mittels Move-Befehl zusammenfügen. Ob das in diesem Fall die schnellste Lösung ist, darf bezweifelt werden. Aber es dient ja nur der Übung; in größeren Zeichnungen ist es bestimmt die bessere Lösung.

Nun geht es also darum, die drei Bauteile *rastergerecht* zu zeichnen. Dazu vereinbaren wir, daß alle senkrechten Verbindungslinien in der ersten, also der linken Spalte eines 8*8-Punkte-Feldes und alle waagerechten Verbindungslinien in der obersten Zeile eines 8*8-Punkte-Feldes verlaufen. Dies ist in Abb. 2.10 dargestellt, in der ich die 8-Punkte-Rasterung stark vergrößert und die Lage der Linien eingezeichnet habe. Sie sehen dort auch, daß die Linien immer an den Feldgrenzen beginnen und enden. Für den PAP wäre das nicht nötig, da man die Blöcke wohl sowieso nicht direkt aneinandersetzt, sondern die Verbindungslinien mittels Line-Befehl zeichnet. Für andere Pläne jedoch kann es durchaus wichtig sein, daß der Anschluß nicht nur seitlich, sondern auch der Länge nach paßt.

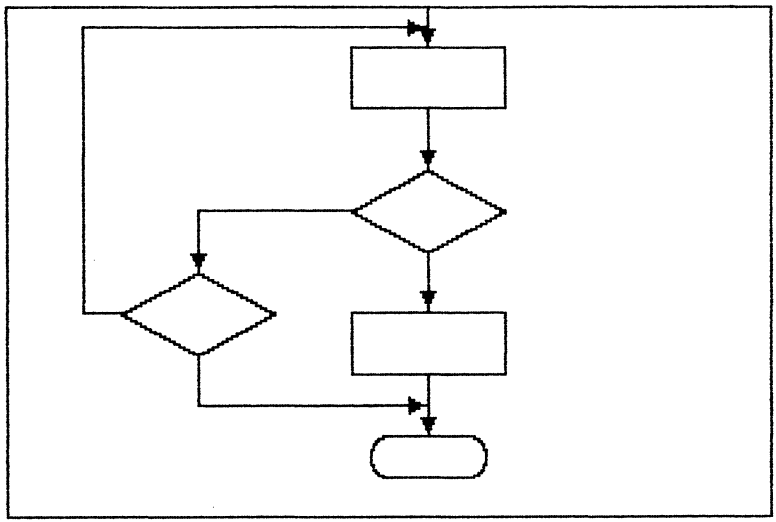


Bild 2.8: Ein Mini-PAP als Lernobjekt

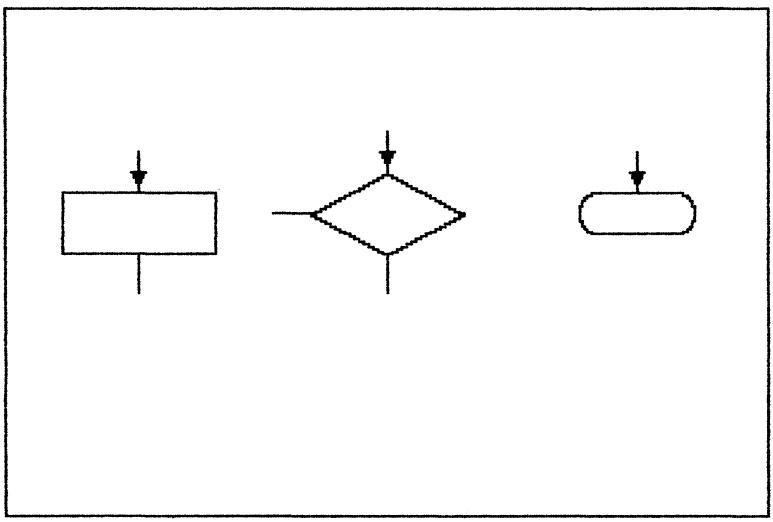


Bild 2.9: Die Elemente des Mini-PAP

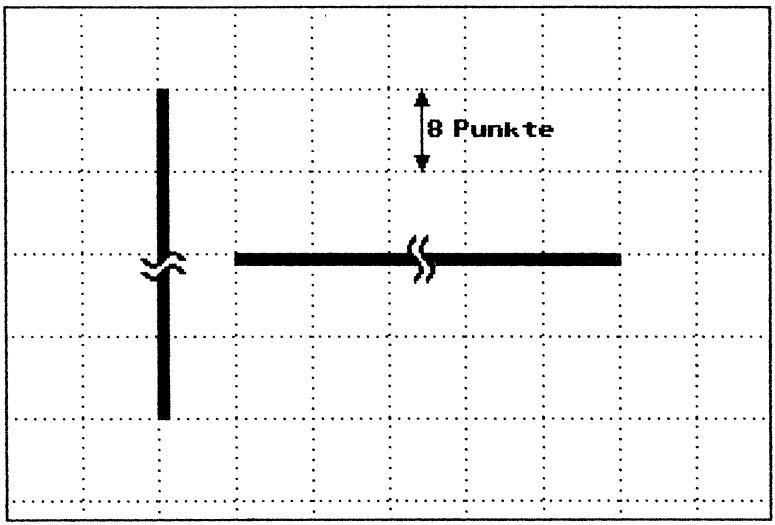


Bild 2.10 So liegen die Linien im 8-Punkte-Raster

Nun fahren wir den Cursor von der HOME-Position mittels Cursortasten etwa 5 Schritte nach rechts und nach unten. Genau dort beginnen wir mit dem Pfeil zum rechteckigen Kasten. Auch das Ende der Ausgangslinie fahren wir von der HOME-Position aus an, wobei wir korrekterweise, wie oben angedeutet, mit dem Joystick einen Punkt nach oben gehen müssen. Dann steht der Cursor in der linken unteren Ecke eines 8*8-Punkte-Feldes, wo gemäß Abb. 2.10 eine senkrechte Linie enden muß. Wir könnten es auch anders machen, nämlich diesen Punkt von der unteren linken Bildschirmecke aus anfahren, dann ersparen wir uns die Korrektur mittels Joystick. Für den PAP ist dies jedoch nicht nötig.

Nach dem gleichen Muster zeichnen wir nun die anderen beiden Elemente - wie man die Raute und das Oval zeichnet, wurde weiter oben bereits erläutert - und fahren auch hierbei jeden Anschlußpunkt aus der HOME-Position oder einer anderen definierten Position an. (Wir können uns auch aus Zeitersparnis eine Position mit dem Tabulator definieren.) Vergessen Sie bei der Raute den seitlichen Anschluß nicht, auch er muß in der linken, oberen Ecke eines Feldes liegen!

Jetzt setzen Sie den PAP aus diesen Elementen zusammen. Holen Sie sich als erstes mit Move das Rechteck und setzen Sie es auf einem anderen Bildschirm oben in die Mitte. Daran soll nun die Raute anschließen. Wie wir wissen, können wir uns nur noch um 8 Punkte oder ein Vielfaches vertun, kleinere Fehler können uns gar nicht mehr passieren. Damit uns das mit den 8 Punkten nicht passiert, wenden wir nun einen Trick an: Setzen Sie den Cursor genau auf den oberen Anschlußpunkt der Raute und fahren Sie dann mittels Cursortasten so weit nach links, daß der Move-Befehl die Raute voll erfaßt. Dort drücken Sie den Knopf, und vor allem merken Sie sich die Anzahl der Schritte, die Sie nach links gefahren sind. Nachdem sie mit dem zweiten Knopfdruck den Quellbereich vollständig markiert und auf den Zielbildschirm umgeschaltet haben, setzen Sie den Cursor dort hin, wo der Anschlußpunkt zu liegen kommen soll - also unter den Anschluß des Rechtecks - und fahren dann genauso viele Schritte wie zuvor nach links. Damit ist sichergestellt, daß der Anschluß exakt paßt!

Bevor Sie nun zum dritten Mal den Knopf drücken, geben Sie noch **O** ein für Oder-Verknüpfung. Damit vermeiden Sie, daß durch einen eventuell etwas zu groß gewählten Bereich Teile der bereits bestehenden Zeichnung gelöscht werden.

Ich hoffe, Sie haben das Prinzip verstanden und sind in der Lage, den PAP zu vervollständigen. Entsprechend gehen Sie bei allen Plänen vor, in denen Module, die für den Sprite-Rahmen zu groß sind, zusammengesetzt werden sollen.

2.4 Struktogramme und mehrteilige Zeichnungen

Nun kommen wir zu einer Eigenschaft, die Hi-Eddi plus von allen anderen mir bekannten Grafikprogrammen für den C64 abhebt: die Möglichkeit, Bilder aus mehreren Bildschirmen zusammenzusetzen und damit Zeichnungen zu erstellen und auszudrucken, die die Auflösung des Computers mehrfach übertreffen. Dabei sind Zeichnungen mit 640*400, 640*600 oder sogar noch mehr Punkten möglich.

Die einzelnen Bildschirme einer solchen Zeichnung müssen jedoch so erstellt werden, daß sie auch wirklich ohne Stoßstellen zusammenpassen. Um dies zu erreichen, gibt es drei Möglichkeiten. In diesem Kapitel möchte ich die erste, die *Anstückelmethode*, beschreiben.

Als Beispiel dient das in Abb. 2.11 dargestellte Struktogramm. Ein Struktogramm oder Nassi-Schneiderman-Diagramm ist im Prinzip dasselbe wie ein PAP, also ein Werkzeug, um Programmentwicklung und Verständis zu erleichtern. Der wesentliche Unterschied zum PAP ist jedoch, daß ein Struktogramm, wie der Name schon sagt, speziell für den Einsatz in Verbindung mit den strukturierten Hochsprachen, zum Beispiel Pascal, Modula II, PL/M, Ada oder auch Logo gedacht ist. Struktogramme sind, wenn man sie erst einmal begriffen hat, leichter lesbar als PAPs. Sie sind übersichtlicher, und lassen sich ohne großen Aufwand, praktisch 1:1 in ein Programm in einer der oben genannten Sprachen umsetzen. Durch die Regeln zum Aufbau von Struktogrammen - an die man sich natürlich halten muß - wird von vornherein ein klar gegliederter, strukturierter Programmaufbau gefördert und *Spaghettiprogramme* werden grundsätzlich ausgeschlossen.

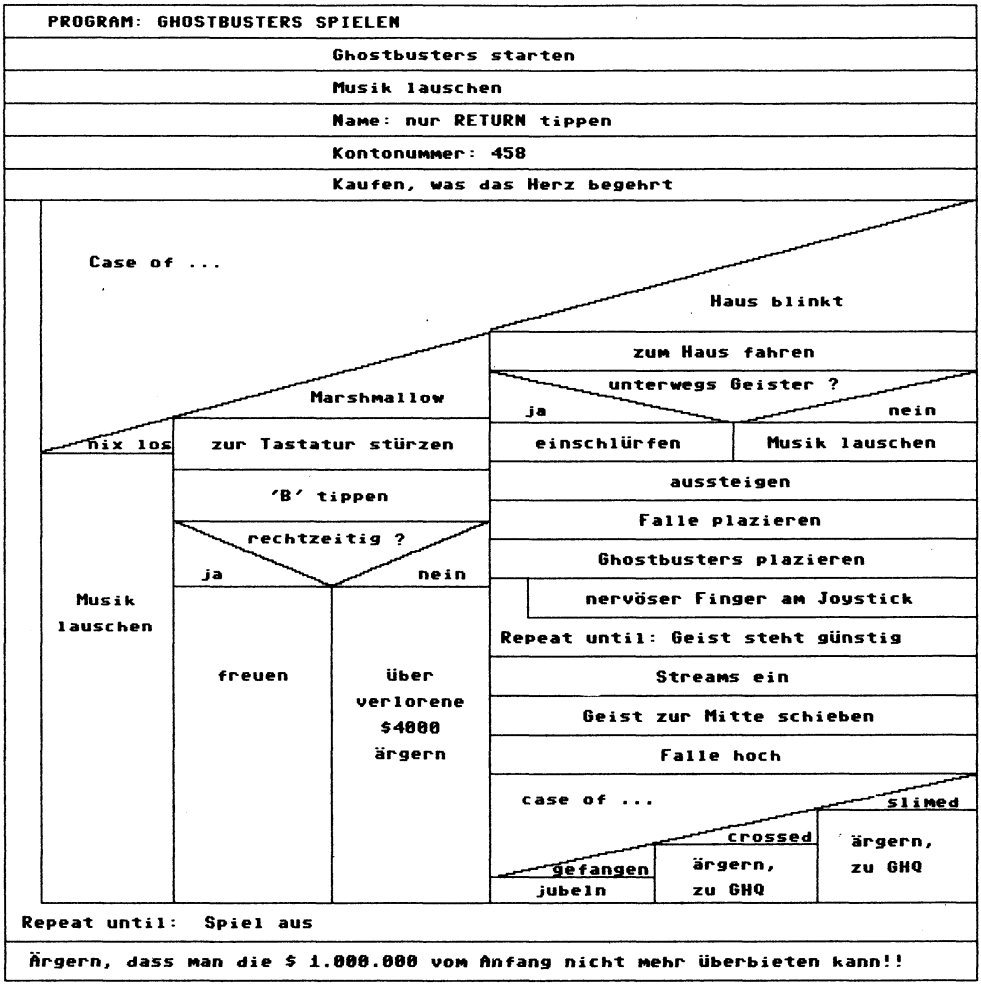


Bild 2.11: Das Spiel "GHOSTBUSTERS" als Struktogramm

Nun ist dies jedoch kein Buch über strukturierte Programmierung und ich möchte deshalb nicht näher auf dieses interessante Thema eingehen. Uns interessiert der rein grafische Aspekt dabei.

Um das Diagramm auf mehrere Bildschirme aufzuteilen, ist es zunächst sinnvoll, die Grenzen so zu legen, daß sie möglichst wenig zerschneiden. Wir werden also die senkrechte Grenze auf die Trennlinie zwischen die Felder *Marshmallow* und *Haus blinkt* legen. Die beiden waagerechten Grenzen legen wir an die Oberkante der Felder *zum Haus fahren* und *Streams ein*. Die Linien, auf die wir die Grenzen gelegt haben, brauchen dann nur jeweils am Rand eines Bildschirms vorhanden sein.

Die waagerechten und senkrechten Linien so zu zeichnen, daß sie zusammenpassen, ist nicht weiter schwierig: Haben wir zum Beispiel am linken oberen Bildschirm eine Waagerechte gezeichnet, so lassen wir den Cursor in dieser Höhe stehen, schalten auf den daneben liegenden Bildschirm um und zeichnen dort in derselben Höhe ebenfalls eine Waagerechte. Entsprechendes gilt für eine Senkrechte. Daß die so erzeugten Linien zusammenpassen, ist wohl einsichtig.

Etwas schwieriger ist es bei der schrägen Linie. Hier hilft uns wieder einmal die programmierbare Schrittweite. Mit ihr können wir nämlich jederzeit beliebige Steigungen reproduzieren. Wenn die Linie zum Beispiel auf 20 Punkten um 9 Punkte steigt, dann programmieren wir einfach diese beiden Werte als Schrittweiten und können so beliebig oft eine Gerade mit dieser Steigung zeichnen. Überschreitet nun eine solche Gerade an einer beliebigen Stelle die Grenze zwischen zwei Bildschirmen, so merken wir uns wieder die Höhe dieser Stelle, indem wir den Cursor dort stehen lassen und können dann im anderen Bildschirm in derselben Höhe die Linie fortsetzen.

In unserem Struktogramm ist die Sache jedoch noch einfacher: Die schräge Linie geht genau durch den Kreuzungspunkt zweier Grenzen. Das hat allerdings den Nachteil, daß sich eine kleine Stoßstelle nicht verhindern läßt. Ferner habe ich ein einfaches Steigungsverhältnis gewählt, nämlich 4:1. Auf der Bildschirmbreite von 320 Punkten steigt die Gerade um 80 Punkte. Eine solche Steigung läßt sich auch ohne große Schrittweitenprogrammierungen erzielen. Zum Beispiel gilt mit der eingestellten Schrittweite von acht horizontal und vertikal: vier Schritte nach rechts und einer nach oben, oder zehn Schritte nach oben auf die ganze Bildschirmbreite.

In Abb. 2.12 habe ich den Berührungspunkt der vier oberen Bildschirme in auseinandergezogener Darstellung, gewissermaßen als Explosionszeichnung, dargestellt.

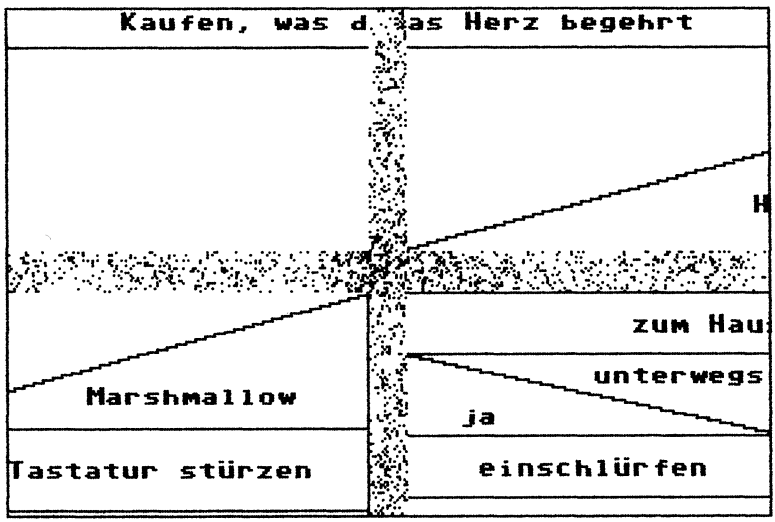


Bild 2.12: Ausschnitt aus dem Struktogramm

Natürlich gibt es für schräge Linien auch noch eine ganz andere Lösung: Man läßt sie weg und zeichnet sie im fertigen Ausdruck mit Bleistift und Lineal ein. Bei Verwendung des entsprechenden Bleistifts merkt man das nicht einmal bei genauerem Hinsehen (ein Freund hat mich mit dieser Methode schon getäuscht!). Einem Profi kann so etwas natürlich nicht passieren. Ein Bleistift! Der hat ja nicht einmal Druckwegoptimierung! Ich muß Ihnen jedoch gestehen, daß ich kein absoluter *Freak* bin. Meine Schallplatten- und Kassettendatei steht auf ganz normalen Zetteln aus Papier, denn wenn ich Bachs Orchestersuite Nr.2 hören will, dann will ich nicht erst den Computer anschalten, Programme laden, mich durch Menüs durchkämpfen und dem Programm verraten, den wievielten wir heute haben. Dann will ich vielmehr wissen, wo ich die Suite finde! Ebenso halte ich es für unsinnig, ein CAD-Programm in BASIC zu schreiben, das mit solch atemberaubender Geschwindigkeit arbeitet, daß ich dieselbe Zeichnung in einem Bruchteil der Zeit auch mit Tusche zustande gebracht hätte. Mit anderen Worten: Der Einsatz eines Computers für ernsthafte Anwendungen muß auch Vorteile bringen, sonst verzichte ich lieber darauf.

Doch nach diesem Exkurs wieder zurück zu unserem Struktogramm und der schrägen Linie, für die es ja eine echte Alternative zum Bleistift gibt. Sie ist vielleicht etwas komplizierter, aber durchaus erlernbar.

Ein kleines Problem stellt noch die Schrift dar. Hier gilt ganz einfach: keine Buchstaben zerschneiden! Um das zu erreichen, fahren wir den Cursor, wenn er in der richtigen Höhe steht, an den rechten Rand und von dort mittels Cursortasten wieder zurück. Beim Eintippen des Textes erreichen wir nun, unabhängig von einer eventuell breiteren oder schmaleren Schrift, punktgenau die letzte Position am Bildschirmrand und können stoßfrei - und natürlich in derselben Höhe - auf dem Nachbarbildschirm weiterschreiben.

Mit diesen Regeln dürfte es kein Problem sein, Zeichnungen wie die in Abb. 2.11 gezeigte zu erstellen. Aber es gibt auch Anwendungsfälle, bei denen sich die Zeichnung nicht so leicht zerschneiden läßt. Wie man in diesem Fall vorgeht, erfahren Sie im nächsten Kapitel.

2.5 Platinenlayouts und mehrteilige Zeichnungen

Leider kann Ihnen Hi-Eddi plus keine Platinenlayouts entwerfen, aber das Programm kann Ihnen beim Entwurf helfen. Wenn Sie nämlich feststellen, daß eine Leitung falsch liegt, dann ist sie schnell wegradiert und kann an anderer Stelle neu gezeichnet werden. Und wenn Sie in der glücklichen Lage sind, einen Drucker zu besitzen, der einen Plot-Modus hat (zum Beispiel einen Epson FX-80), dann können Sie mit Hi-Eddi plus sogar reprofähige Vorlagen erzeugen. Der Plot-Modus bewirkt, daß der Ausdruck in Höhe und Breite exakt gleich ist. Dies trifft bei den CRT-Grafik-Modi, die zum Ausdruck von Grafikbildschirmen gedacht sind, meist nicht zu. Da jedoch die absolute Größe nicht stimmen wird, müssen Sie sich das Ganze noch in einem Kopierladen auf das endgültige Maß vergrößern oder verkleinern lassen. Am besten nehmen Sie dafür Transparentpapier, da es billiger als Folie ist und genauso gut funktioniert, wie ich mir von einem kompetenten Bastler sagen ließ. Ich selbst habe keine Erfahrung mit dem Ätzen von Platinen.

Da das Anpassen des Druckers nicht ganz einfach ist - noch dazu wenn man alle Features, wie etwa den Plot-Modus ausnutzen möchte - habe ich dafür ein eigenes Kapitel reserviert. In diesem Abschnitt wollen wir uns auf das Zeichnen eines Layouts, wie es in Abb. 2.13 zu sehen ist, beschränken. Wie Sie bereits sehen, ist dieses Layout aus zwei Bildschirmen zusammengesetzt. Wo ist die Trennlinie? Antwort: Es gibt keine!

Ich habe den gesamten Mittelteil dieses Layouts auf einem einzigen Bildschirm erstellt. Mittels Move-Befehl habe ich dann die beiden Hälften dieses Bildschirms in zwei verschiedene Bildschirme kopiert und dort das Layout nach außen hin vervollständigt. Abb. 2.14 verdeutlicht diese Vorgehensweise.

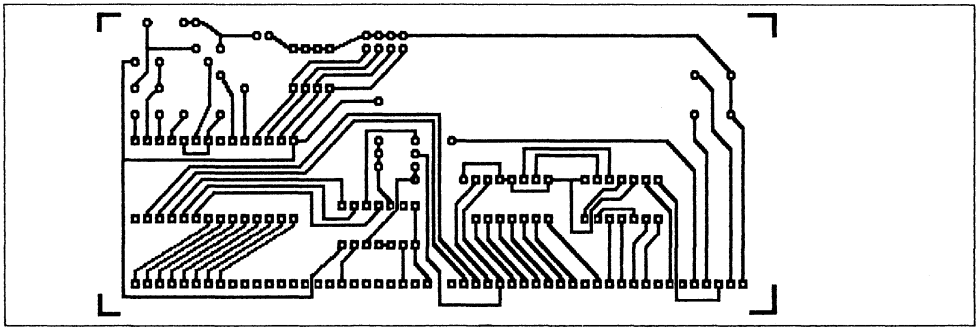


Bild 2.13: Platinen-Layout, aus 2 Bildschirmen zusammengesetzt

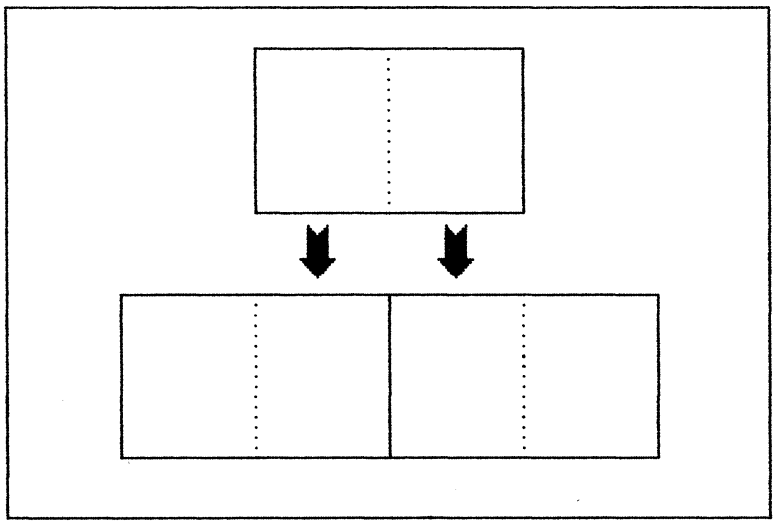


Bild 2.14: Darstellung der "Aus-1-mach-2-Methode"

Und das war auch schon das ganze Geheimnis der zweiten Methode zum Zeichnen mehrteiliger Bilder. Man könnte Sie die *Aus-eins-mach-zwei-Methode* nennen. Die dritte Methode, die ich Ihnen versprochen habe, kommt erst in Kapitel 4. Sie ist die komfortabelste!

Ich möchte Ihnen nun noch einige Tips zum Zeichnen von Layouts geben:

Zunächst empfehle ich Ihnen, wenn Sie oft Layouts zeichnen, ein Construction-Set dafür anzulegen. Das wichtigste, was da hineingehört, sind Lötstützpunkte in verschiedenen Größen. Dann können Sie fertige Anordnungen, zum Beispiel für verschiedene Transistoren, eckige und runde ICs und was Sie sonst noch benötigen, vorsehen. Alles, was es auf den im Handel erhältlichen Karten mit Abreibesymbol gibt, können Sie in ihr Construction-Set aufnehmen, natürlich nur, sofern dies sinnvoll ist. Eine gerade Linie zum Beispiel werden Sie nicht als Symbol im Construction-Set brauchen, die können Sie direkt mit dem Line-Befehl zeichnen, auch wenn Sie einige Linien nebeneinander setzen müssen, um auf die erforderliche Dicke zu kommen. Oder Sie verwenden den Draw-Modus mit dem dicken Pinsel.

Auch die programmierbare Schrittweite leistet hierbei wieder gute Dienste: Programmieren Sie das Rastermaß Ihrer Platine als Schrittweite! Wollen Sie eine Anschlußleiste zeichnen, zum Beispiel mit 22 Kontakten, wie sie für den Expansion-Port des C64 gebraucht wird, dann erstellen Sie einen Kontakt als Sprite (im Construction-Set ablegen!) und stempeln ihn 22 mal nebeneinander, wobei Sie den Abstand mittels Schrittweite exakt einhalten können. Wenn Ihnen das noch zu langsam geht, dann speichern Sie die nötige Schrittweite auf der Funktionstaste F1 ab, von wo sich die Joysticksteuerung die Schrittweite holt. Dann brauchen Sie nur noch den Knopf zu drücken, den Joystick in die gewünschte Richtung zu halten und haben im Nu Ihre Anschlußleiste - wahrscheinlich ist sie sogar viel länger, als Sie sie haben wollten!

2.6 Verknüpfungbefehle

Ähnlich wie die programmierbare Schrittweite, deren Vielseitigkeit Sie ja inzwischen kennengelernt haben, sind auch die Verknüpfungsbefehle - ein Leistungsmerkmal von Hi-Eddi plus, mit dem der Laie meist nicht viel anzufangen weiß. Ich will Ihnen deshalb einige Anwendungsmöglichkeiten der Verknüpfungsbefehle, also der Befehle **O**, **X** und **U**, zeigen.

Nehmen wir zum Beispiel in Abb. 2.15, das Bild mit dem *Torwand-Design*. Es erinnert an die Torwand des aktuellen Sportstudios im ZDF!

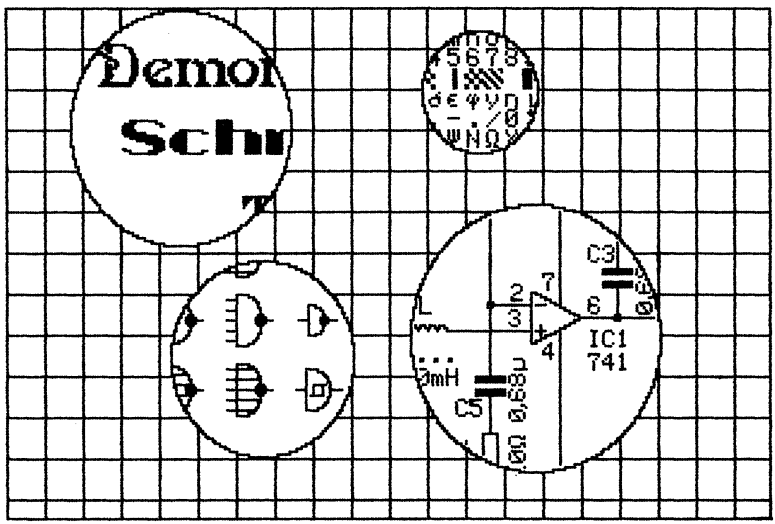


Bild 2.15: Demobild im "Torwand-Design"

Die vier kreisrunden Ausschnitte werden Sie sicherlich wiedererkennen: Sie stammen aus den Bildern auf der Diskette. Mittels der Befehle **U** und **O** läßt sich dieses Bild in Minutenschnelle erstellen. Die Vorgehensweise ist in Abb.2.16 skizziert. Sie wird hier, wie in Abb.2.16, nur für einen Kreisausschnitt dargestellt. Natürlich läßt sie sich entsprechend auch für mehrere Kreise anwenden.

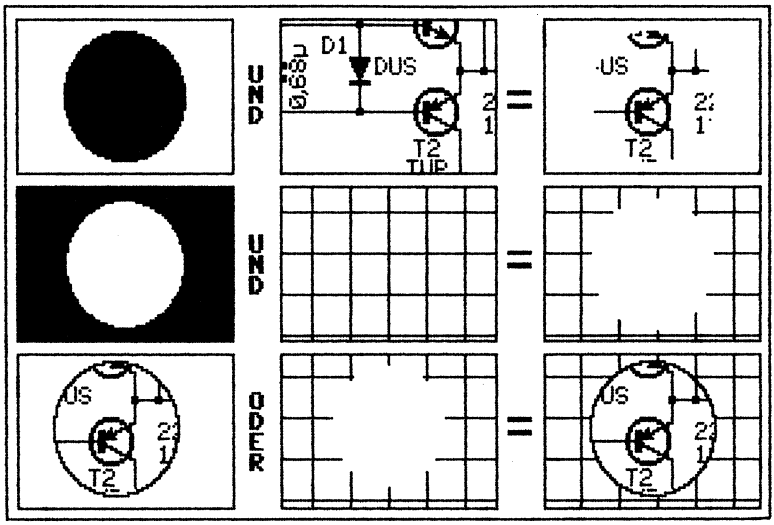


Bild 2.16: Verknüpfungs-Anleitung

Zunächst brauchen wir eine kreisrunde, ausgefüllte Scheibe. Die zeichnen wir mittels Circle- und Paint-Befehl auf einen leeren Bildschirm. Wenn Sie den Paint-Befehl nicht mehr brauchen, schalten Sie ihn wieder ab, indem Sie beispielsweise mit **C** wieder auf Circle zurückschalten. Es kann leicht passieren, daß man versehentlich einmal den Knopf drückt. Wenn dann der Paint-Befehl noch eingeschaltet ist, wird das sehr unangenehm!

Positionieren Sie diese Kreisscheibe bereits beim Zeichnen so, daß sie in derselben Gegend liegt wie der Bildausschnitt, den Sie in den Kreis hineinbekommen möchten. Um dies möglichst genau zu erreichen, gibt es einen Trick. Wollen Sie zum Beispiel den Operationsverstärker aus der Beispiel-Schaltung (**SCHALTUNG.PIC** auf der Diskette) in einen Kreis bekommen, wie dies in Abb. 2.15 zu sehen ist, gehen Sie folgendermaßen vor: Wählen Sie den Circle-Befehl an, setzen Sie den Cursor mitten auf den OP und drücken Sie den Knopf. Fahren Sie dann den Cursor nach außen, so daß der komplette OP, so weit Sie ihn im Bild haben wollen, erfaßt wird. Schalten Sie erst jetzt, unmittelbar vor dem zweiten Knopfdruck, auf den leeren Bildschirm um. Dort drücken Sie nun zum zweiten Mal den Knopf und Sie können sicher sein, daß der so entstandene Kreis den gewünschten Ausschnitt erfaßt. Nachdem Sie den Kreis mittels Paint ausgefüllt haben, machen Sie sich unbedingt eine Kopie dieses Bildschirmes (mit dem Copy-Befehl), denn wir brauchen die Scheibe, wie Sie in Abb. 2.16 schon sehen können, noch ein zweites Mal.

Verknüpfen Sie nun den Bildschirm, auf dem Sie die Scheibe gezeichnet haben, durch den Und-Befehl mit der Schaltung, oder was Sie gerade als Beispiel verwenden. Dazu tippen Sie U und die Nummer des Bildschirms, in dem Ihr Beispiel-Bild liegt. Als Ergebnis müssen Sie einen kreisrunden Ausschnitt Ihres Beispielbildes, wie er in Abb. 2.16 rechts oben zu sehen ist, erhalten. Vergleichen Sie dieses Ergebnis auch mit der Verknüpfung zweier einfacher Symbole in Abb. 1.1 in Kapitel 1. Sie werden das bestätigt finden, was ich in Kapitel 1 über die Und-Verknüpfung gesagt habe: Nur dort, wo in beiden zu verknüpfenden Bildern ein Punkt gesetzt ist, ist auch im Ergebnis ein Punkt gesetzt.

Um diesen Ausschnitt besser hervorzuheben, empfiehlt es sich, einen weiteren Kreis darum zu zeichnen. Sie erinnern sich wahrscheinlich: Der Mittelpunkt des Kreises ist noch auf der Funktionstaste F7 gespeichert (vorausgesetzt, Sie haben zwischenzeitlich noch nichts anderes gezeichnet oder die Zoom-Funktion aufgerufen). Setzen Sie also den Cursor mittels F7 auf den Kreismittelpunkt, drücken Sie den Knopf, fahren Sie den Cursor an den Rand des Kreises und drücken Sie den Knopf nochmals. Das Ergebnis sollte etwa so wie in Abb. 2.16 links unten aussehen.

Eigentlich sieht das jetzt schon recht ansprechend aus und man könnte es dabei bewenden lassen. Da ich Ihnen jedoch die Verknüpfungsbefehle demonstrieren möchte, machen wir jetzt mit dem Torwand-Hintergrund weiter.

Dazu brauchen wir zunächt einen Bildschirm mit einem Gitterraster. Ein solches läßt sich - Sie ahnen es sicher schon - mit Hilfe der programmierbaren Schrittweite sehr schnell erstellen. Ich habe zum Beispiel eines mit dem Rastermaß 16 verwendet. Programmieren Sie dazu zunächst die 16 als horizontale (H) und die 200 als vertikale (V) Schrittweite. Mit diesen Schrittweiten und dem Line-Befehl lassen sich die senkrechten Linien zeichnen. Beginnend von der HOME-Position gehen Sie dazu folgendermaßen vor: ein Schritt nach rechts - RETURN - ein Schritt nach unten - RETURN - ein Schritt nach rechts - RETURN - ein Schritt nach oben - RETURN - und so weiter. Für die waagerechten Linien muß die vertikale Schrittweite 16 sein und für die horizontale verwenden Sie 160. Größer als 255 kann die Schrittweite nicht sein, 320 geht also nicht. Sie müssen nun von einem Rand zum anderen immer zwei Schritte gehen, sonst funktioniert es entsprechend. Wenn Sie sich beim Eingeben nicht ständig vertun - wie es mir passiert ist - dann brauchen Sie für das Gitterraster weniger als eine Minute! Und wie es noch schneller und vor allem ohne diese mühsame Tipperei geht, erfahren Sie in Kapitel 4.

Nun wählen Sie den Bildschirm an, auf dem Sie sich vorher eine Kopie der Kreisscheibe angelegt hatten, und invertieren ihn mittels **I**. Was nun zu tun ist, zeigt die mittlere Zeile in Abb.2.16: Führen Sie eine UND-Verknüpfung der invertierten Kreisscheibe und des Gitterrasters durch. Das Ergebnis ist ein Gitterraster mit einem Loch.

Nun folgt der letzte, in Abb. 2.16 angedeutete Schritt. Dabei kommt eine zweite Verknüpfung, nämlich die Oder-Verknüpfung, zum Einsatz.

Bei dieser Verknüpfung wird im Ergebnis schon ein Punkt gesetzt, wenn in nur einem der zu verknüpfenden Bilder an der entsprechenden Stelle ein Punkt gesetzt ist. Die Oder-Verknüpfung ist also eine Überlagerung, wie etwa das Übereinanderlegen zweier Folien. Wenn Sie nun die *Folien* mit dem durchlöcherten Gitter und die zuvor erstellte mit dem Kreisausschnitt übereinanderlegen - also eine Oder-Verknüpfung dieser beiden Bildschirme durchführen - haben Sie das fertige Ergebnis im *Torwand-Design*!

Nach dem gleichen Prinzip lassen sich auch Schraffuren erstellen, die ja für Konstruktionszeichnungen häufig gebraucht werden. In Abb. 2.17 sehen Sie die einzelnen Etappen beim Schraffieren beliebiger Flächen in einer Zeichnung. Gehen wir sie einmal der Reihe nach durch.

Zunächst brauchen Sie einen ganzen Bildschirm mit der gewünschten Schraffur. Der geneigte Leser weiß bereits: Mit der programmierbaren Schrittweite geht es besonders einfach. Außerdem empfehle ich Ihnen Schrittweiten im 8-Punkte-Raster zu wählen. Warum? Weil man sich so sehr viel Arbeit sparen kann. Sie brauchen dann nämlich nicht den ganzen Bildschirm zu schraffieren, sondern zum Beispiel nur ein Feld von etwa 100*100 Punkten, also ein Sechstel des ganzen Bildschirmes. Mittels Move-Befehl vervielfältigen Sie nun dieses Feld und füllen den ganzen Bildschirm damit aus. Diese Schraffur-Seite sollten Sie sich auch auf Diskette abspeichern, denn Sie werden immer wieder etwas schraffieren müssen.

An dieser Stelle noch ein Tip, der nicht nur die Schraffur betrifft, sondern allgemein für schräge Linien gilt, bei denen es auf die genaue Lage ankommt: Zeichnen Sie solche Linien immer in die gleiche Richtung, also zum Beispiel in der Schraffur alle von links unten nach rechts oben. Bedingt durch den Line-Algorithmus liegen nämlich schräge Linien, die zwar dieselben Endpunkte haben, aber in verschiedene Richtungen gezeichnet wurden, nicht immer exakt aufeinander!

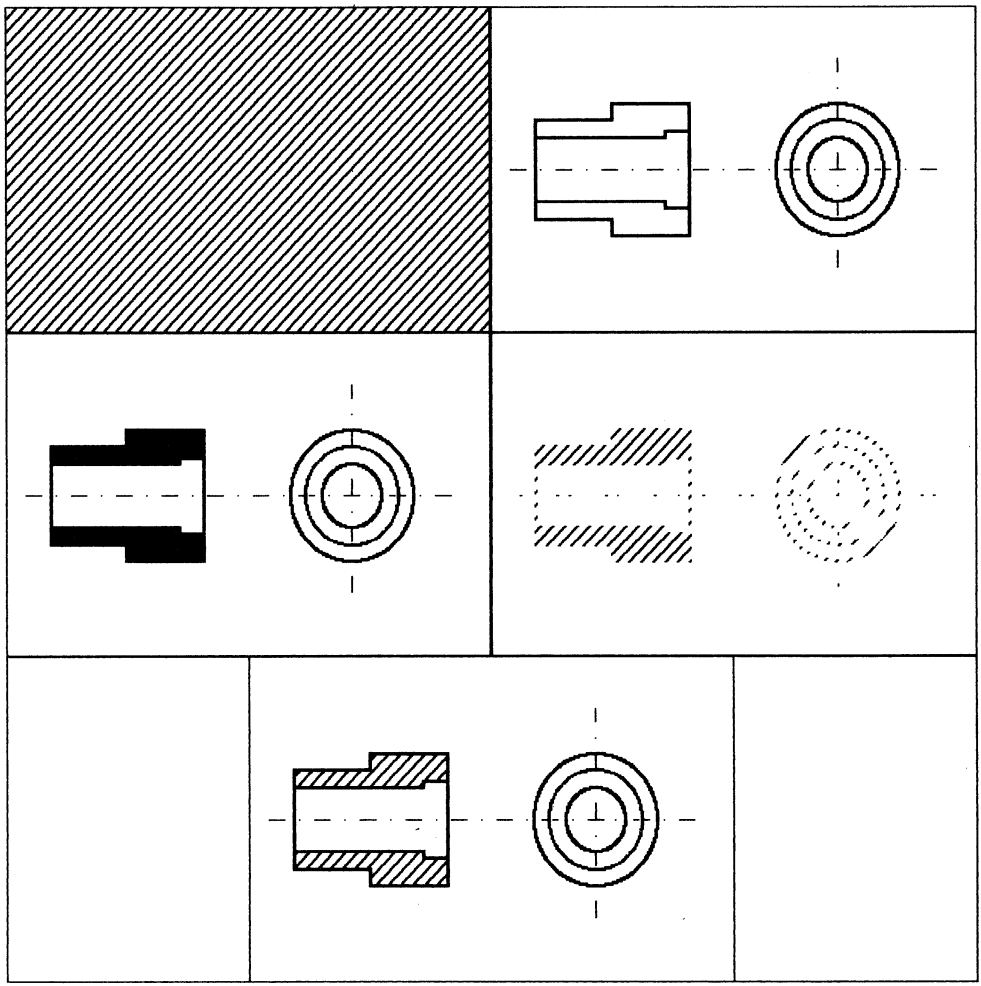


Bild 2.17: Die Schritte zur Erzeugung einer Schraffur

Als nächstes brauchen wir natürlich die Konstruktionszeichnung selbst, die Sie in Abb. 2.17 rechts oben sehen. Wie es die Norm vorschreibt, sind die Kantenlinien dicker; man zeichnet dazu immer zwei Linien nebeneinander. Die Mittellinien sind mittels Append erzeugt. Ich habe dazu im Sprite-Editor einen Strich und einen Punkt ins Sprite gezeichnet und anschließend das Sprite in Schrittweiten entsprechender Länge stempelnderweise über den Bildschirm laufen lassen. Damit die Zeichnung maßstabsgerecht wird, kann man sich wieder der programmierbaren Schrittweite bedienen. Schön wäre natürlich auch eine Koordinatenanzeige, die, in den Bildschirm eingeblendet, immer die aktuelle Cursorposition anzeigt. Das kann Hi-Eddi plus auch! Allerdings muß ich Sie damit auf Kapitel 4 vertrösten.

Erstellen Sie nun eine Kopie der Zeichnung und füllen Sie darin die zu schraffierenden Flächen mit dem Paint-Befehl voll aus. Nun kommen die Verknüpfungsbefehle zum Einsatz: Führen Sie eine Und-Verknüpfung dieses soeben erstellten Bildes mit dem Schraffur-Bildschirm durch. Das Ergebnis sollte etwa so aussehen wie in Abb. 2.17 rechts in der Mitte. Deutlich ist darin bereits die Schraffur in der passenden Form zu erkennen. Stören Sie sich nicht an den *Rückständen*, die bei dieser Verknüpfung entstanden sind; im nächsten Schritt verschwinden sie wieder. Dieser nächste Schritt ist die Oder-Verknüpfung des soeben erzeugten Bildes mit der noch nicht schraffierten und nicht ausgefüllten Konstruktionszeichnung. Das fertige Ergebnis sehen Sie in Abb. 2.17 unten.

Natürlich können in einer Zeichnung auch verschiedene Schraffuren vorkommen. Aus der zuvor erstellten Schraffur-Seite läßt sich mittels Mirror eine zweite, entgegengesetzt gerichtete Schraffur erstellen. Aus beiden zusammen können Sie mittels Oder-Verknüpfung eine Kreuzschraffur erstellen.

Um bei einem Bild verschiedene Schraffuren zu ereichen, müssen Sie lediglich die zuvor beschriebenen Schritte für jede Schraffur wiederholen. Sie dürfen also nicht alle zu schraffierenden Flächen auf einmal mit Paint auffüllen, sondern nur immer die Flächen, die die gleiche Schraffur erhalten sollen.

Kommen wir nun zur dritten Verknüpfung, der Exclusiv-Oder-Verknüpfung, im Englischen auch oft mit Exor abgekürzt. Was ist der Unterschied zwischen dieser Verknüpfung und der *normalen* Oder-Verknüpfung? Was ist an ihr so exklusiv? Ich will versuchen, das an zwei nicht ganz ernst zu nehmenden Beispielen darzulegen.

Wenn sich der Computerfreak überlegt: "Soll ich mir einen Drucker oder einen Farbmonitor kaufen?", dann ist das in der Regel ein exklusives *oder*. Der Freak kann auswählen, ob er entweder das eine oder das andere kauft. Beides zusammen läßt der Geldbeutel nicht zu. Die beiden Möglichkeiten schließen sich also gegenseitig aus, d.h. sie sind exklusiv. Auch ein exklusiver Club schließt den Normalbürger aus, weil ihm Geld oder eine andere erforderliche Vorraussetzung fehlt.

Wenn der Freak dann überlegt, wie er trotzdem zu einem Drucker und einem Monitor kommen könnte, dann stößt er womöglich auf folgenden Text: "Wer Banknoten nachmacht oder verfälscht oder nachgemachte oder verfälschte Banknoten sich verschafft und in Verkehr bringt, wird mit Freiheitsstrafe nicht unter zwei Jahren bestraft" (Eine entsprechende Formulierung gibt es auch über bestimmte Software, doch davon soll hier nicht die Rede sein.) Diese vielen *oder* sind alle definitiv, also nicht exklusiv, denn Sie kommen ins Gefängnis, egal, ob Sie nur eine der hier dargestellten Möglichkeiten wahrnehmen oder aber mehrere.

Übertragen wir die so gewonnenen Erkenntnisse auf Grafikbilder: Bei der Oder-Verknüpfung sind Punkte gesetzt, wenn in nur einem oder aber auch in beiden zu verknüpfenden Bildern Punkte an den entsprechenden Stellen gesetzt sind. Bei der Exor-Verknüpfung dagegen sind Punkte nur dann gesetzt, wenn sie entweder im ersten oder im zweiten zu verknüpfenden Bild gesetzt sind, nicht dagegen, wenn sie in beiden gesetzt sind. Sehen Sie sich dazu auch Abb. 1.1 an.

Was bringt das? Überlegen Sie sich, was passiert, wenn man ein aus einer Exor-Verknüpfung entstandenes Bild nochmals mit einem der zwei Bilder, aus denen es entstanden ist, verknüpft. Das Ergebnis zeigt Abb. 1.2: Sie bekommen wieder das andere der beiden Ausgangsbilder zurück.

Was man mit dieser Methode anfangen kann, zeigt Abb. 2.18: Sie können ein Gitterraster, wie es in der unteren Hälfte der Abbildung zu sehen ist, in eine Zeichnung einblenden, um eine bessere Orientierung zu erhalten. Wie das dann aussieht, zeigt die obere Hälfte der Abbildung. Zwar ist das exakte Positionieren in einem bestimmten Raster auch mit der programmierbaren Schrittweite möglich, aber das Gitter verschafft einem doch einen besseren und schnelleren Überblick zur Grobeinteilung eines Bildes.

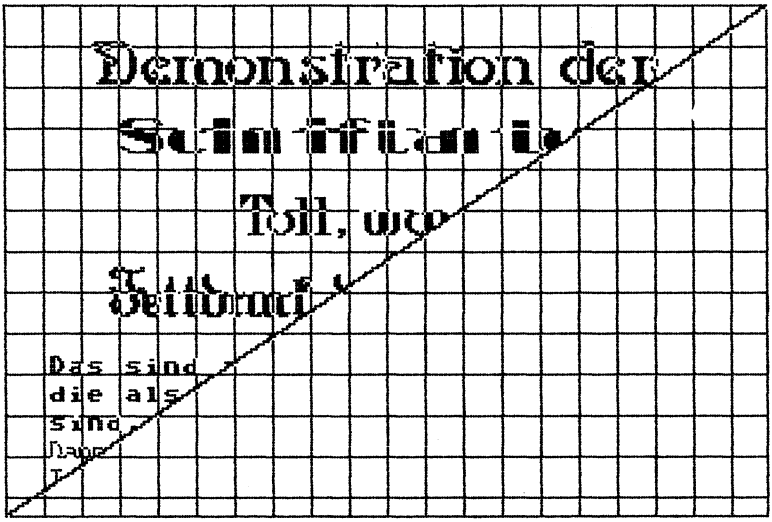


Bild 2.18: Ein Gitterraster, mittels EXOR eingeblendet

Alles was Sie dazu brauchen ist ein Gitterraster, wie wir es bereits für das *Torwand*-Bild erstellt haben. Dieses Gitter können Sie bei Bedarf in ihre Zeichnung einblenden, indem Sie eine Exor-Verknüpfung mit dem Bildschirm, der dieses Gitter beinhaltet, durchführen. Bei nochmaliger Verknüpfung verschwindet das Gitter wieder vollständig. Allerdings sollten Sie das Gitter sofort, nachdem Sie den Cursor entsprechend plaziert oder sich einen Überblick verschafft haben, wieder ausblenden. Wenn Sie das Gitter nämlich übermalen, kann es passieren, daß es doch nicht vollständig herausgeholt wird und Spuren davon zurückbleiben.

Natürlich ist diese Methode des Ein- und wieder Ausblendens nicht auf Gitter beschränkt. Sie können waagerechte oder senkrechte Linien oder ein Punktraster verwenden, wie in Abb. 2.19 dargestellt. Dort sehen Sie auch, wie man mittels Und-Verknüpfung das Punktraster aus den senkrechten und waagerechten Linien gewinnen kann. Eine Oder-Verknüpfung der Linien ergäbe wieder das Gitterraster.

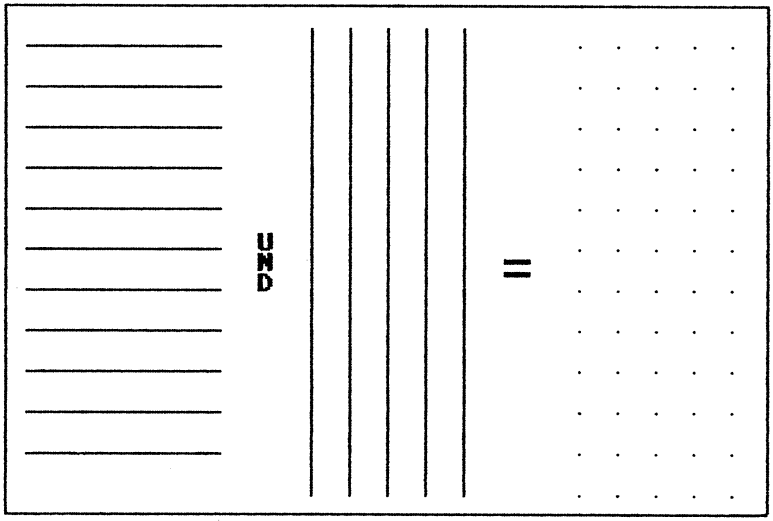


Bild 2.19: Verschiedene Gitter und die Verknüpfung daraus

Die Ein- und Ausblendmethode eignet sich auch als *Probeverknüpfung* vor einer Und- oder Oder-Verknüpfung. Wenn Sie zwei Bildschirme verknüpfen möchten, aber nicht genau wissen, wie sie zusammenpassen, dann machen Sie erst eine Exor-Verknüpfung, da Sie diese wieder rückgängig machen können. Eine Und- oder Oder-Verknüpfung dagegen ist endgültig, sie läßt sich nicht rückgängig machen!

Zwei weitere Anwendungsmöglichkeiten der Exor-Verknüpfung habe ich bereits in Kapitel 1 angedeutet: das bereichsweise Löschen und Invertieren. Beide Möglichkeiten machen davon Gebrauch, daß sich die Verknüpfungsbefehle nicht nur für ganze Bildschirme, sondern in Verbindung mit dem Move-Befehl auch auf Bereiche anwenden lassen. Das gilt natürlich auch für alle bisher beschriebenen Anwendungen: Wenn es sinnvoll ist, die Verknüpfungen nicht mit ganzen Bildschirmen, sondern mit Bereichen durchzuführen, so ist das zusammen mit dem Move-Befehl möglich. Die Verknüpfungsregeln sind dabei dieselben wie für ganze Bildschirme.

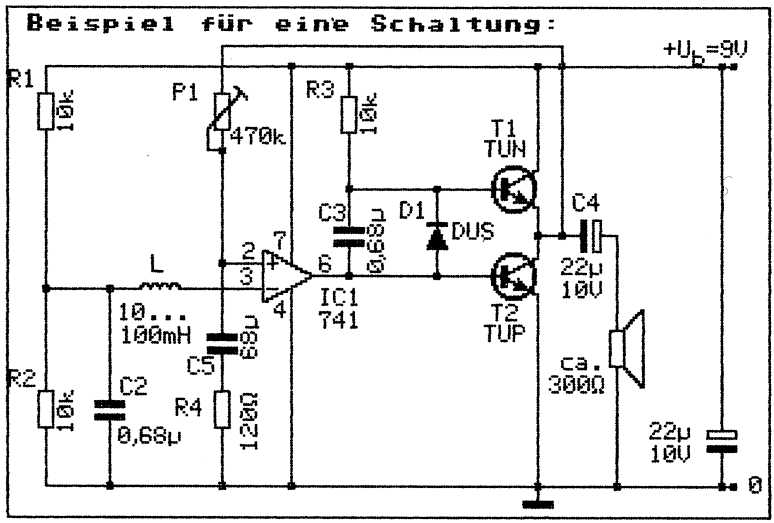
Sehen wir uns zuerst das Löschen an. Es soll beispielsweise die ganze rechte Bildschirmhälfte gelöscht werden. Wählen Sie dazu den Move-Modus (M) und markieren Sie die rechte Bildschirmhälfte durch zweimaliges Drücken des Knopfes in diagonalen Ecken. Am sinnvollsten ist es, als erstes die rechte untere Ecke zu wählen und als zweites die linke obere, denn dort muß der Cursor auch beim dritten Knopfdruck stehen. Doch vor dem dritten Knopfdruck geben Sie X für die Exclusiv-Oder-Verknüpfung ein. Das Ergebnis dieser Exor-Verknüpfung eines Bildschirmausschnittes mit sich selbst ist, daß alles gelöscht wird, denn es gibt keine Punkte, die in einem Ausschnitt gesetzt sind und im anderen nicht. Nach dem gleichen Prinzip funktioniert übrigens der Befehl C= CLR zum Löschen des Bildschirms. Da die Verknüpfungsbefehle vorhanden sind, habe ich mir die Porgrammierung eines eigenen Löschbefehls erspart und erledige das durch eine Exor-Verknüpfung des sichtbaren Bildschirms mit sich selbst.

Eine weitere Anwendung ist das bereichsweise invertieren des Bildschirms. Der Befehl I kann nur den ganzen Bildschirm invertieren; wollen Sie nun ein Rechteck in einem Bild invertieren, dann legen Sie sich zunächst einen *vollen* Bildschirm an, indem Sie einen gelöschten Bildschirm mit I invertieren. Wählen Sie dann den Move-Modus an, sowie das Bild, in dem invertiert werden soll. Setzen Sie mit dem ersten Knopfdruck am besten wieder die rechte untere Ecke und plazieren Sie dann den Cursor auf die linke obere Ecke des zu invertierenden Bereichs. Bevor Sie aber den Knopf drücken, schalten Sie auf den vollen Bildschirm um. Nun haben Sie ein Stück des vollen Bildschirms markiert und verknüpfen es mit dem entsprechenden Ausschnitt Ihres Bildes. Schalten Sie dazu auf Ihr Bild zurück, geben Sie X ein und drücken Sie den Knopf!

Der Nachteil bei dieser Methode ist, daß Sie auf das 8*8-Punkte-Raster des Move-Befehls beschränkt sind. Es gibt jedoch einen anderen Weg. Wollen Sie ein beliebig großes oder beliebig geformtes Stück Ihres Bildes invertieren, zeichnen Sie diese Form als ausgefüllte Fläche auf einen leeren Bildschirm und führen eine Exor-Verknüpfung dieses ganzen Bildschirms mit dem Bild durch. In Abb. 2.20 ist das Ergebnis dieser Operation zu sehen. Hier wurde die Schrift-Demo mit einer sternförmigen Fläche invertiert.

Bild 2.20: Die Schrift-Demo, hier invertiert mit einem Stern

Zum Abschluß dieses Kapitels noch ein kleines Spielchen. Sicherlich kennen Sie aus Zeitungen die Bilder, in denen man Unterschiede finden muß. Hi-Eddi plus kann bzw. könnte Ihnen bei der Suche gewaltig helfen, wenn die Bilder aus der Zeitung in den Computer geladen werden könnten. Ersatzweise habe ich Ihnen in Abb. 2.21 ein kleines Suchbild dieser Art gezeichnet, indem ich ins Beispielbild 'SCHALTUNG.PIC' von der Diskette einige Fehler eingebaut habe. Finden Sie sie?



Bilde 2.21: Die Schaltung von der Diskette, aber leicht geändert

Wie wir schon gesehen haben, ergibt eine Verknüpfung zweier identischer Bilder gar nichts, also einen gelöschten Bildschirm. Was geschieht aber, wenn doch kleine Unterschiede vorhanden sind? Die werden dann groß und deutlich herausgehoben und sind nicht mehr zu übersehen. So auch in Abb. 2.22, in der die Fehler, die ich in Abb. 2.21 eingebaut habe, durch die Exor-Verknüpfung entdeckt wurden!

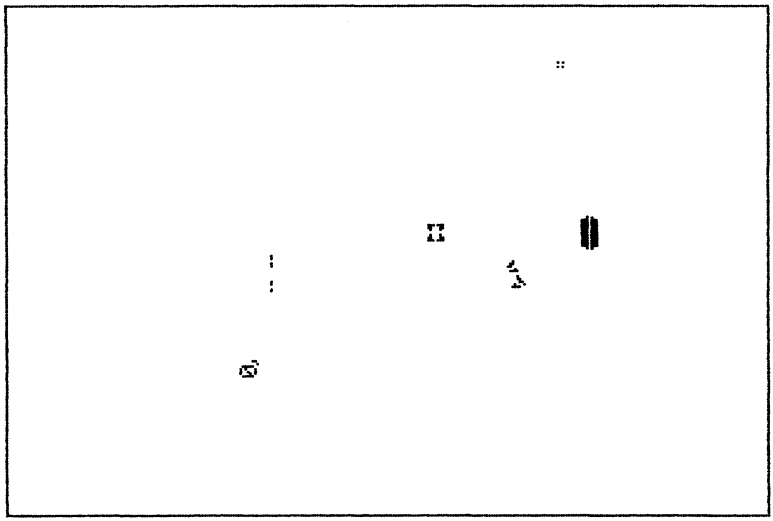


Bild 2.22: Die "Fehler" aus Bild 2.21

2.7 Der Paint-Befehl

Sie werden sich vielleicht fragen, was es zum Paint-Befehl überhaupt zu sagen gibt, da seine Anwendung doch recht eindeutig ist: Er dient zum Ausfüllen von Flächen, ausgefüllt oder mit Schachbrettmuster, wie in Kapitel 1 schon zu lesen war. Zusammen mit 'I' kann man ihn außerdem zum Löschen benutzen.

Tatsächlich gibt es dazu auch kaum mehr zu sagen. Daß ich diesem Befehl dennoch ein eigenes Kapitel widme, liegt darin begründet, daß ich Ihnen nach der Darlegung so vieler Anwendungen, Tips, Tricks und insbesondere auch der Verknüpfungsbefehle im letzten Abschnitt nun die Gelegenheit geben möchte, ein wenig zu entspannen.

Ein weiterer Grund ist der: Ein Befehl zum Ausfüllen unregelmäßiger Flächen ist programmiertechnisch gesehen so ziemlich das Anspruchsvollste, was man in ein Grafikprogramm aufnehmen kann. Deshalb gibt es auch Grafikprogramme oder -erweiterungen, die zwar im großen und ganzen alles bieten, was man braucht, aber gerade dieser wichtige Befehl fehlt! Da es mir noch dazu gelungen ist, die Paint-Routine in kaum mehr als 200 Bytes unterzubringen - die Routine zum Zeichnen einer Linie ist schon fast genauso lang - ist es wohl verständlich, daß mir gerade dieser Befehl besonders am Herzen liegt.

Doch nun zu dem versprochenen Entspannungsspiel, das Ihnen zwar für die praktische Arbeit mit Hi-Eddi plus nichts bringen wird, Ihnen aber einen Blick hinter die Kulissen gewährt und vor Augen führt, daß der Paint-Befehl doch nicht ganz so perfekt ist, wie man nach den obigen Zeilen vielleicht vermuten könnte.

Zeichnen Sie zunächst die *Treppe* aus Abb. 2.23 ab. Die Stufen dieser Treppe erstellen Sie am schnellsten im Append-Modus: Wählen Sie den Sprite-Editor an und zeichnen Sie einen waagerechten Strich quer durch das ganze Sprite. Kehren Sie in den Grafik-Editor zurück und setzen Sie den Rahmencursor in die linke untere Ecke. Wählen Sie nun mittels Commodore-Taste und F1 die Schrittweite für die Joysticksteuerung an und vergrößern Sie die vertikale und horizontale Schrittweite auf 4. Nun brauchen Sie nur noch den Knopf zu drücken und den Joystick nach rechts oben zu halten. Schon läuft der Cursor über den Bildschirm und hinterläßt dabei als Fußabdrücke im 4-Punkte-Abstand die Stufen. Sollten sich am Anfang oder Ende ein paar Fehler eingeschlichen haben, lassen sich diese leicht korrigieren, zum Beispiel mittels Erase oder Draw mit dickem Pinsel. Die Ge-

nauigkeit spielt in diesem Fall auch gar keine so große Rolle, es sollen nur etwa 40 bis 50 sich überlappende Stufen quer über den ganzen Bildschirm sein. Schließen Sie nun die Treppe noch mit zwei Linien ein, die bis zum Rand gehen sollen, die Stufen jedoch nicht berühren dürfen. Erstellen Sie sich von diesem Werk unbedingt eine Kopie, denn Sie werden es öfter brauchen.

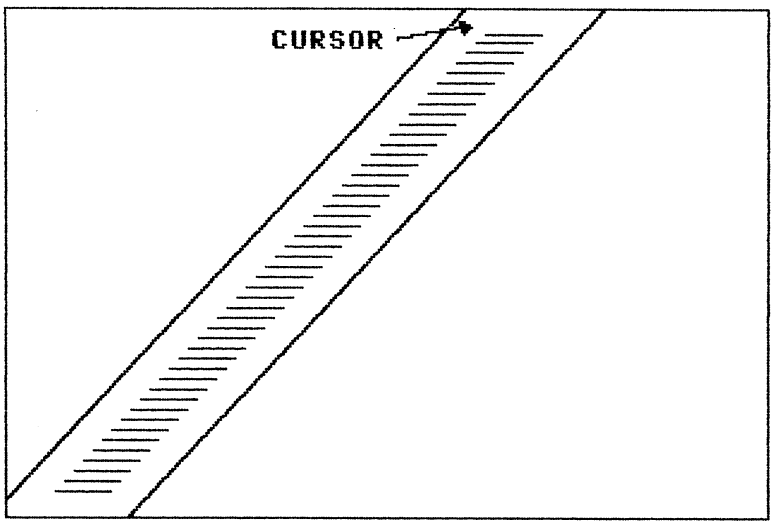


Bild 2.23: Die Paint-Treppe

Wählen Sie nun den Paint-Befehl an, setzen Sie den Cursor an die in Abb. 2.23 markierte Stelle und drücken Sie den Knopf. Schauen Sie jetzt genau zu, wie der *Wurm*, der die Fläche ausfüllt, läuft! Da der Vorgang recht schnell abläuft, sollten Sie sich den Vorgang ruhig ein paarmal anschauen. Dazu holen Sie sich jedes Mal die zuvor erstellte Kopie mit dem Copy-Befehl zurück.

Wie Sie sehen werden, läuft der Wurm zunächst an der rechten Seite der Treppe herunter, schlüpft dann unterwegs plötzlich ohne ersichtlichen Grund durch die Treppe hindurch und läuft am linken Rand wieder hinauf, wobei er die Zwischenräume zwischen den Stufen ausläßt. Die kommen erst dran, nachdem die Treppe nach unten Stufe für Stufe ausgemalt wurde. Ganz zum Schluß folgen dann noch ein paar übriggebliebene Ecken.

Warum schlüpft der Wurm unterwegs durch die Treppe hindurch, wo doch gar keine Unregelmäßigkeit vorliegt? Um das zu verstehen, müssen wir uns gewissermaßen in die Lage unseres Wurmes versetzen. Er hat die Aufgabe, die ganze Fläche, die er erreichen kann, aufzufüllen. Erreicht er unterwegs eine Abzweigung - und jede Treppenstufe ist eine solche Abzweigung - dann muß er sich für einen Weg entscheiden. Natürlich gibt es auch Paint-Algorithmen, die sich in alle Richtungen gleichzeitig *vorkämpfen*, wie das zum Beispiel beim Malprogramm Paint Magic der Fall ist. Unser Wurm jedoch ist unteilbar. Er entscheidet sich daher an einer Abzweigung für einen der möglichen Wege, schreibt sich jedoch in einem Notizblock die Stelle auf, um später dorthin zurückzukehren und den anderen Weg zu gehen. Nach einiger Zeit jedoch ist sein Notizblock voll und er kann sich nichts mehr aufschreiben. Es bleibt ihm dann nichts anderes übrig, als den Weg, den er sich sonst aufgeschrieben hätte, sofort zu erkunden. Das passiert genau dann, wenn er durch die Treppe hindurchschlüpft. Von dort aus wählt er nun einen beliebigen Weg, in unserem Falle nach oben, bis ans obere Ende der Treppe. Er trifft dabei auf viele weitere Abzweigungen, kann sich aber nichts mehr merken! Zum Glück jedoch wird er die übersehenen Flächen, nämlich die Zwischenräume zwischen den Stufen, von der anderen Seite her auffüllen können, denn dort hat er sich die Abzweigungen ja aufgeschrieben. Es kann aber durchaus passieren, daß der Wurm Flächen übersieht, die er von keiner anderen Seite mehr auffüllen kann. Diese Flächenstücke bleiben leer und müssen gesondert aufgefüllt werden. Dies ist auch die oben erwähnte *Unvollkommenheit* des Paint-Befehls. Warum ich sie nicht beseitigt habe? Weil der Notizblock in Wirklichkeit nichts anderes ist als RAM-Speicherplatz, und der ist im Hi-Eddi plus sehr knapp. Den Grund dafür erfahren Sie in Kapitel 4. Eine weitere Schwäche ist die Tatsache, daß der Wurm, wenn er ein Schachbrettmuster ausfüllen soll, nicht mehr durch einen engen Flaschenhals kommt. Dieses Problem dürfte allerdings sehr selten auftreten.

Betrachten wir nun noch den restlichen Weg unseres Wurmes über die Treppe. Mit vollem Notizblock arbeitet er sich nach unten vor, wobei er jede Abzweigung sofort inspizieren muß und dabei immer Gefahr läuft, eine Abzweigung zu übersehen. Da sich aber in der Treppe alle Wege wieder treffen, passiert das nicht. Versuchen Sie nun einmal eine Fläche zu zeichnen, mit der Sie den Wurm überlisten können! Es wird Ihnen wahrscheinlich gelingen, aber ich bin mir sicher, daß solche Fälle in der Praxis nur äußerst selten auftreten werden.

Soweit nun dieses kleine Wurmspielchen. Ich werde in Kapitel 4 noch darauf zurückkommen und Ihnen dann zeigen, wie man dem Wurm in seinen Notizblock schauen kann! Speichern Sie also die Treppe auf Diskette ab.

2.8 Zierschrift

Nachdem wir bisher fast ausschließlich technische Anwendungen betrachtet haben, wird es nun wesentlich weniger technisch. Wie in der Einleitung schon erwähnt, können Sie mit Hi-Eddi nicht nur technische Pläne und Skizzen, sondern auch Glückwunschkarten, Einladungen, Briefköpfe, QSL-Karten etc. erstellen.

Wichtigste Vorraussetzung dafür sind natürlich verschiedene möglichst ansprechende Schriftarten, die größer als die normale im Textmodus verwendete Schrift sein sollen. Eine simple Vergrößerung der Commodore-Schrift oder eines selbst erstellten Zeichensatzes im 8*8-Punkte-Raster wäre hier sicher nicht geeignet. Durch eine Vergrößerung würde dann zum Beispiel aus einem Punkt ein Feld von 2*2 oder 3*3 Punkten, was eine eckige und unansehnliche Schrift ergäbe. Sehen Sie sich deshalb einmal die kleine Schrift mit der Zoom-Funktion an!

Um eine schöne Schrift zu erhalten, muß man einen anderen Weg gehen: Die Zeichen müssen in einer viel größeren Matrix als der 8*8-Matrix des Textmodus erstellt werden. Dafür bietet sich die Sprite-Matrix mit ihren 24*21 Punkten geradezu an. Sie ist groß genug, um bereits wohlgeformte Buchstaben zu erstellen, die eine passende Größe für die meisten Anwendungsfälle haben. Solche Buchstaben kann man sich selbst malen und ein Construction-Set daraus erstellen.

Wem das Selbermalen zu aufwendig ist, der findet auf der Diskette bereits vier fertige Zierschriftsätze als Construction- Sets: **ANTIC.CS, BOCKLIN.CS, FRAKTUR.CS** und **MODERN.CS**. Die Abb. 2.24 bis 2.27 zeigen diese Schriften.

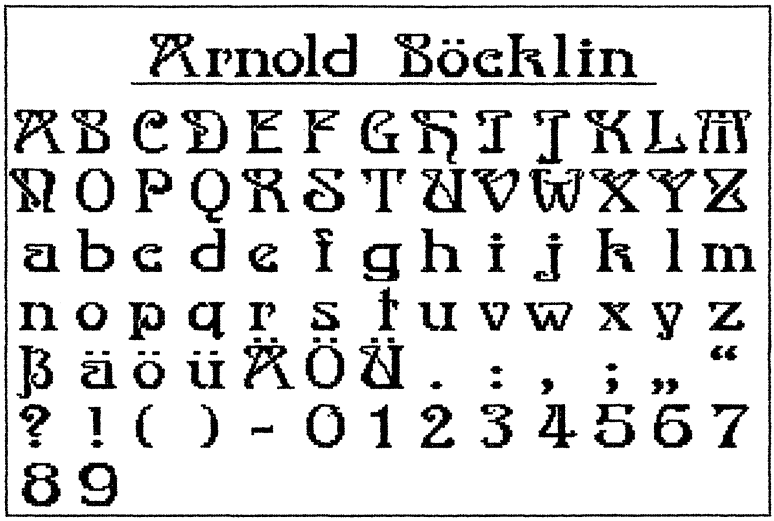


Bild 2.24: Die Zierschrift "Arnold Böcklin"

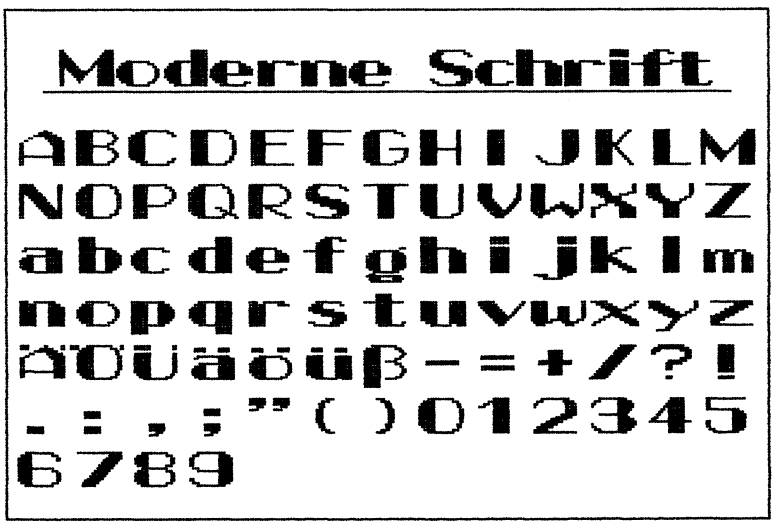


Bild 2.25: Eine moderne Schrift als Construction-Set

Bild 2.26: Ebenfalls auf der Diskette: Eine Antiqua-ähnliche Schrift

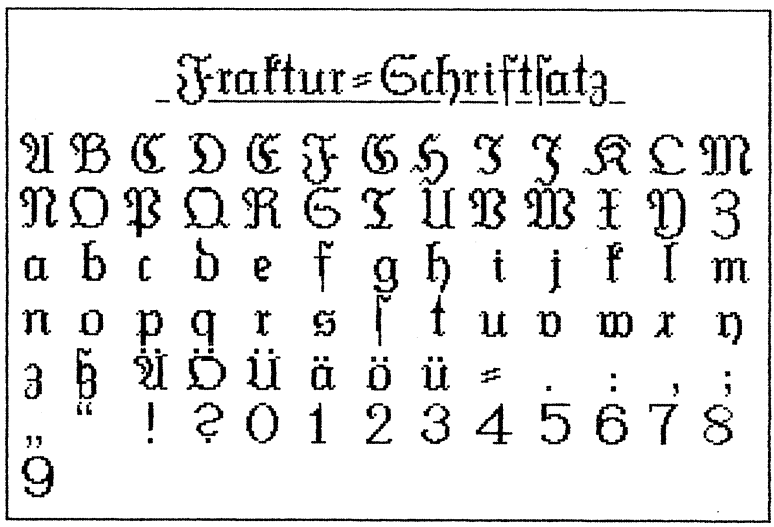


Bild 2.27: Auch Fraktur ist auf der Diskette

Wie die Bezeichnung Construction-Set schon andeutet, muß man zum *Schreiben* mit diesen Schriften wie mit den Schaltungs-Construction-Sets vorgehen. Sie holen mit Get einen Buchstaben aus dem Set und fügen ihn in das zu erstellende Bild ein. Das ist natürlich nicht so komfortabel wie das Eintippen der Buchstaben im Textmodus; andererseits wird man bei der Größe der Buchstaben aber auch nicht allzu viel schreiben.

Außerdem hat diese Methode den Vorteil, daß Sie in Proportionalschrift schreiben können. Sie brauchen also nicht für jedes Zeichen einen bestimmten, festen Platz, sondern können bei schmalen Buchstaben die einzelnen Zeichen enger zusammenrücken.

Wie geht man nun beim Schreiben im einzelnen vor? Wie schon gesagt, müssen die Buchstaben mittels Get (G) aus dem Set geholt werden. Bereits beim *Stempeln* des ersten Zeichens empfiehlt es sich nun, darauf zu achten, daß die geschriebene Zeile bei einem anschließenden Move-Vorgang ins 8*8-Raster des Move-Befehls paßt. Sie werden die Zeile kaum auf Anhieb in die richtige Position bekommen, vielmehr werden Sie sie anschließend zentrieren, etwas nach links, rechts, oben oder unten verschieben wollen. Dazu brauchen Sie den Move-Befehl.

Wie man den Cursor so plaziert, daß ein anschließendes Move gelingt, haben Sie bereits im Kapitel über den Programmablaufplan erfahren: Sie müssen dazu lediglich den Cursor von der HOME-Position aus in 8-Punkte-Schritten bewegen. Diese 8-Punkte-Schritte erreichen Sie durch die programmierbare Schrittweite. Im Einschaltzustand ist bereits eine Schrittweite von 8 eingestellt. Sie brauchen den Cursor jetzt nur noch mit den Cursortasten zu bewegen, um die 8-Punkte-Schritte zu erreichen.

Die Beachtung des Move-Rasters ist freilich nur beim ersten Zeichen einer Zeile nötig; die folgenden liegen dann automatisch in derselben Höhe. Allerdings sollten Sie beim Holen des ersten Zeichens aus dem Construction-Set darauf achten, den Rahmen so zu plazieren, daß auch eventuelle Unterlängen noch Platz darin hätten. Ansonsten könnte es passieren, daß bei einem Move-Vorgang gerade diese Unterlängen abgeschnitten werden.

Nach links und rechts brauchen Sie keine Rücksicht auf das Move-Raster zu nehmen, denn hier ist in der Regel so viel Platz, daß der zu verschiebende Bereich ruhig etwas überstehen kann. Falls Sie zwischen den Zeilen viel Platz lassen wollen, brauchen Sie sich eigentlich überhaupt nicht um das Move-Raster zu kümmern. Ich habe mir das Ausrichten der einzelnen Zeilen jedoch angewöhnt, da man dadurch immer die Möglichkeit hat, die Zeilen mittels Move exakt zu erfassen, d.h. ohne Abschneiden, aber

auch ohne überstehende Ränder nach oben und unten. Das hält einem die Möglichkeit offen, die Zeilen bei Bedarf doch enger zusammenzurücken. Außerdem wird auf diese Weise ein gleichmäßiger Zeilenabstand erreicht.

Und gleich noch ein Tip im Zusammenhang mit Move: Wenn Ihnen das ständige Umschalten zwischen dem Bildschirm mit dem Construction-Set und dem mit Ihrer Zeichnung zu umständlich ist, dann schreiben Sie die Zeile einfach an den unteren Rand des Sets; von dort holen Sie sie wieder mittels Move-Befehl. Oder löschen Sie die Überschrift des Sets, so daß Sie mehr Platz haben. Dieses Löschen geschieht ebenfalls am besten mit dem Move-Befehl und der Exor-Verknüpfung, die im Kapitel über Verknüpfungsbefehle beschrieben ist. Falls Sie am unteren Rand schreiben, müssen Sie wieder darauf achten, daß auch ein Buchstabe mit Unterlänge noch Platz hat!

Wenn Sie das erste Zeichen gesetzt haben und mit dem zweiten bereitstehen, taucht die Frage auf, wie man es so plaziert, daß es exakt in derselben Höhe wie das erste steht. Das ist jedoch kein Problem! Wir gehen dabei genauso vor, wie bei unserem ersten Beispiel mit den beiden Transistoren: Wir fahren den Cursorrahmen mit dem zweiten Zeichen über das erste und wenn die beiden Zeichen auf gleicher Höhe liegen, schieben wir das zweite nach rechts und stempeln es neben das erste. Wichtig ist dabei, daß Sie für den Bildschirmrahmen eine Farbe wählen, die sowohl zum Hintergrund als auch zur Zeichenfarbe einen guten Kontrast bildet, denn das Zeichen im Rahmencursor hat dieselbe Farbe wie der Bildschirmrahmen.

Der Abstand zwischen den einzelnen Zeichen bleibt Ihnen überlassen, je nachdem, ob Sie eng oder breit schreiben wollen. Wählen Sie die Zwischenräume in jedem Fall immer gleich, die Proportionalschrift ergibt sich dann ganz von selbst. Auch das Aufnehmen der Buchstaben aus dem Set ist unkritisch. Es kommt dabei nicht darauf an, ob Sie den Rahmen genau positionieren, Sie müssen nur darauf achten, den gewünschten Buchstaben vollständig erfaßt zu haben, ohne noch ein Stück von einem anderen mitzunehmen. Das ergäbe unerwünschte Flecken im Bild.

In Abb. 2.28, der Schrift-Demo von der Diskette (**SCHRIFT.PIC**), sehen Sie eine weitere Möglichkeit, wie man die Schrift manipulieren kann: Durch zweimaliges Stempeln jedes Zeichens im Abstand von einem Punkt erreicht man Fettdruck. Natürlich können Sie auch zweimal untereinander oder um je einen Punkt zur Seite und nach unten verschoben drucken. Dadurch ergeben sich verschiedene Schriftbreiten und -bilder. Allerdings be-

steht auch die Gefahr, die Buchstaben zu unansehnlichen, breiten Zeichen zu verwischen, in denen die hübschen Schnörkel gar nicht mehr zu erkennen sind.

Bild 2.28: Ein Beispiel für die verschiedenen Schriftarten

Falls Ihnen ein Fehler unterlaufen ist, löschen Sie ihn am besten mit Erase. Sollte jedoch aus irgend einem Grund der Erase-Rahmen zu groß sein, so gibt es, wie bereits in früheren Kapiteln angedeutet, noch weitere Möglichkeiten: Löschen mit dem dicken Draw-Pinsel oder mit einem beliebig geformten Pinsel im Append-Modus oder mit der Zoom-Funktion für Feinkorrekturen *unter der Lupe*. Für die Zoom-Funktion müssen Sie zunächst in einen der Zeichenmodi schalten, zum Beispiel Draw, da Sie sonst mit der Space-Taste oder dem entsprechenden Feld im Menü nicht in die Zoom-Funktion, sondern in den Sprite-Editor kämen.

Außerdem können Sie Buchstaben auch ganz genau löschen. Wollen Sie aus einem Wort einen Buchstaben entfernen, gehen Sie folgendermaßen vor: Holen Sie diesen Buchstaben aus dem Set, plazieren Sie ihn genau auf dem zu löschenden, invertieren Sie den Bildschirm mit I und stempeln Sie den Buchstaben nun in das so entstandene *Loch*, das genau der Form dieses Zeichens entspricht. Nach nochmaligem Invertieren des Bildschirms ist das Zeichen verschwunden.

2.9 Glückwunschkarten

Nachdem wir nun gelernt haben, wie man Zierschrift auf dem Bildschirm gestaltet, wollen wir uns am Beispiel einer Glückwunschkarte zum Führerschein einmal ansehen, was man damit anfangen kann (Abb. 2.29). Auf der Diskette sind das die Files **GLUECK1.PIC** bis **GLUECK3.PIC**. Auf dem letzten der Bilder ist auch eine kleine Anleitung zum Ausdrucken dieser Karte. Vorraussetzung zum Ausdruck ist aber ein grafikfähiger Drucker mit mindestens 640 Punkten pro Zeile. Ein Commodore VC1525/MPS801 ist dafür also nicht geeignet, da damit keine zwei Bilder nebeneinander ausgedruckt werden können.

Zunächst müssen die ersten beiden Bilder mittels **SHIFT T** gedreht werden. Dann muß Bild 2 ausgedruckt werden, und zwar an linken Rand des Blattes. Da ein kleines, einzelnes Bild normalerweise in die Mitte des Blattes gedruckt wird, wenden wir dazu einen Trick an: Als zweites, rechtes Bild lassen wir einen leeren Bildschirm ausdrucken. Genauso geht es bei Bild Nummer 1, das direkt anschließend an das vorherige ausgedruckt wird. Vor dem nächsten Bild geben wir ein paar Zeilenvorschübe mittels Line-Feed-Taste am Drucker. Dann wird das Bild Nummer 3 ausgedruckt, das nun jedoch am rechten Rand stehen muß. Dazu wird diesmal als erstes Bild der leere Bildschirm ausgedruckt. Natürlich müssen Sie vor dem Ausdruck die kleine Anleitung auf Bild 3 löschen; Sie gehört nicht zum Glückwunsch.

Bild 2.29: Die Vorderseite der Glückwunschkarte von der Diskette

Wie Sie die *Karte* falten, zeigt Abb. 2.30. Und schon ist die Glückwunschkarte fertig!

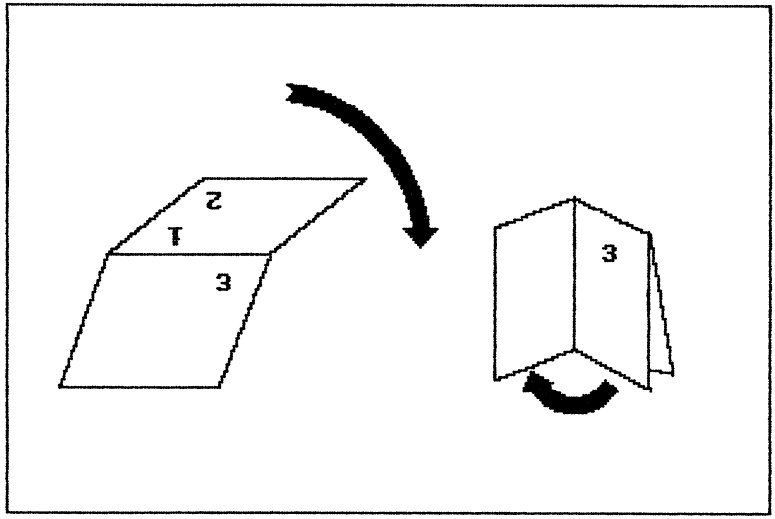


Bild 2.30: So wird die Glückwunschkarte gefaltet

Ein Wort wäre vielleicht noch zum Rahmen zu sagen, der aus dem ebenfalls auf der Diskette befindlichen Rahmen-Construction Set (**RAHMEN.CS**, Abb. 2.31) entnommen wurde. Natürlich geht auch das wieder mittels Get und Append. Damit die einzelnen Rahmenelemente in genau gleichen Abständen plaziert werden, benutzen wir die programmierbare Schrittweite. Die Länge der Schritte richtet sich dabei nicht nach den Spritemaßen, sondern nach der Größe der Rahmenelemete. Für ein Element der Größe 20*20 brauchen wir logischerweise die 20 als horizontale und vertikale Schrittweite.

Wenn sich der Rahmen über mehrere Bildschirme erstreckt, wie es im Beispiel der Fall ist, sollten wir darauf achten, daß an der Trennstelle kein Element zerschnitten wird. Dazu fängt man mit dem Rahmen ganz an der Trennstelle an und arbeitet sich von dort aus um den ganzen Bildschirm herum vorwärts. Das Element muß dann allerdings so im Sprite liegen, daß Sie damit wirklich an den Rand herankommen. Wollen Sie zum Beispiel an den unteren Bildschirmrand herankommen, muß das Element auch am unteren Rand des Sprites anstehen. Sie haben jedoch bereits eine Möglichkeit kennengelernt, dieses **Randproblem** bei mehrteiligen Bildern zu umgehen: die **Aus-eins-mach-zwei-Methode**. Eine noch bessere Methode habe ich Ihnen für Kapitel 4 versprochen.

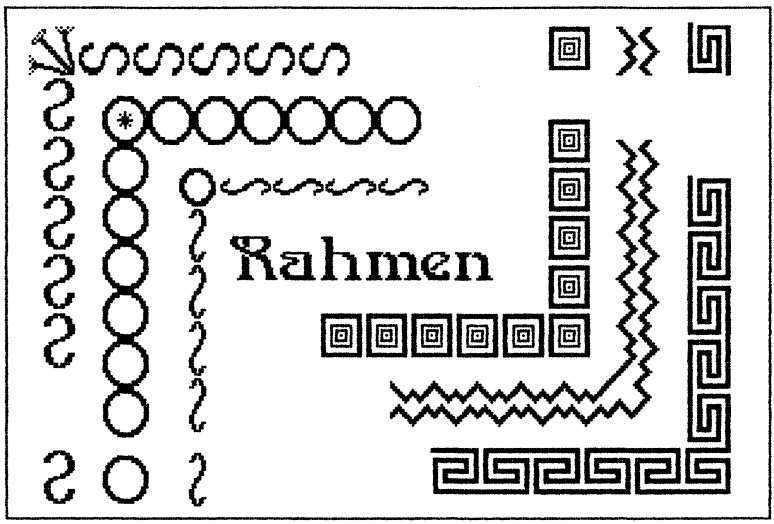


Bild 2.31: Verschiedene Rahmen-Elemente als Construction-Set

Für die verschiedenen Seiten des Rahmens muß das Element entsprechend gedreht werden, was wir im Sprite-Editor mit den Befehlen **M**, **T** und **R** erreichen. Bei dem im Beispiel verwendeten Rahmen ergibt sich noch eine weitere Schwierigkeit: Selbst in einer Seite des Rahmens sind abwechselnd verschiedene Elemente. Wir können hier im ersten Durchgang nur jedes zweite Element drucken, müssen anschließend das Element spiegeln und können daraufhin erst die Lücken auffüllen.

Durch die recht großen Rahmenelemente ist es nicht immer möglich, den Bildschirm optimal auszunutzen. Sie sehen das auch im Beispiel an den unterschiedlich großen freien Rändern. Es empfiehlt sich daher, mit dem Rahmen zu beginnen und erst dann den Inhalt zu zeichnen. Sie könnten dabei sogar den Inhalt auf einen anderen Bildschirm zeichnen und ihn erst wenn er fertig ist mit dem Rahmen verknüpfen, denn ohne den Rahmen läßt es sich besser arbeiten! Zum Beipiel braucht man bei Move nicht darauf zu achten, daß man den Bereich nicht zu groß wählt. Vor der eigentlichen Verknüpfung ist jedoch eine *Probeverknüpfung* mittels Exor angebracht, wie ich Sie im Kapitel über die Verknüpfungsbefehle beschrieben habe.

2.10 Zeichensätze für den Text-Modus

Nachdem wir uns in den letzten beiden Abschnitten mit den großen Buchstaben, den Zierschriften, befaßt haben, kehren wir nun wieder zu den *kleinen* Buchstaben zurück. Gemeint sind damit die Zeichensätze, die man im Text-Modus verwendet.

Wie man sie verwendet, von Diskette lädt und die Schrittweite einstellt, mit der sich der Cursor bei der Eingabe der Buchstaben vorwärtsbewegt, habe ich bereits in früheren Kapiteln beschrieben. In diesem Kapitel geht es darum, wie man die Zeichensätze erstellt und was dabei zu beachten ist.

Dabei stoßen wir - wieder einmal - auf die 8*8-Punkte-Rasterung. Sie ist nun einmal die Grundeinheit der Grafik des C64; der ganze Grafikbildschirm ist aus Päckchen zu je 8*8 Punkten aufgebaut. Deshalb bestehen auch die Zeichen, die normalerweise im sogenannten Character-ROM abgelegt sind, aus einer Matrix von 8*8 Punkten. Wenn wir den Zeichensatz mit **SHIFT Z** ins RAM kopieren, können wir das ganz genau sehen. Fahren Sie dazu den Cursor in die HOME-Position und wählen Sie die Zoom-Funktion an. Zum besseren Abzählen der Punkte können Sie nun noch die Grid-Funktion zuschalten (G), die die einzelnen Punkte deutlich markiert.

Das erste Feld von 8*8 Punkten, ganz links oben, ergibt die Punktmatrix für den Klammeraffen, die 8*8 Punkte unmittelbar daneben ergeben das A und so weiter. Wollen wir einen eigenen Zeichensatz erstellen, dann müssen wir uns an diese Einteilung halten. Auch wenn wir schmalere Buchstaben haben, dürfen wir sie nicht näher zusammenrücken. Laden Sie den Zeichensatz **PICA.ZS** von der Diskette - dort können Sie diesen Sachverhalt genau erkennen. Obwohl keines der Zeichen breiter als 5 Punkte ist, muß jedes in einem eigenen 8*8-Punkte-Feld stehen, und dies sollte es immer in gleicher Weise tun. So habe ich alle 5 Punkte breiten Zeichen derart angeordnet, daß zum linken Rand des 8*8-Punkte-Feldes 1 Punkt frei bleibt und zum rechten 2 Punkte. Würde man diese Regel nicht einhalten und ein Zeichen an dem linken, das nächste an dem rechten Rand anbringen, dann würden sich einige Zeichen beim Schreiben mit verringerter Schrittweite gegenseitig überdecken, während zwischen anderen eine Lücke bliebe.

Außerdem habe ich die Zeichen näher zum linken als zum rechten Rand hin orientiert, damit es beim Löschen mit der DEL-Taste keine Fehler unterlaufen. Bei Betätigung dieser Taste wird nämlich immer das volle, vom Cursor erfaßte 8*8-Punkte-Feld gelöscht. Würde man die Zeichen zum rechten Rand hin setzen, dann würde mit DEL nicht nur das letzte Zeichen gelöscht, sondern auch das vorhergehende noch beschädigt.

Ein ähnliches Problem gibt es bei schmalen Zeichen in inverser Darstellung. Sicher ist Ihnen am Pica-Zeichensatz sofort aufgefallen, daß der Teil, in dem die inversen Zeichen stehen, nicht vollständig invertiert ist. Die letzte Spalte jedes 8*8-Punkte-Feldes ist frei geblieben. Wäre sie das nicht, dann würden sich auch hier die Buchstaben gegenseitig überdecken. Die letzte Spalte eines Zeichens würde dann die erste von den 5 benutzten Spalten des nächsten Zeichens überschreiben.

Falls Sie jedoch lieber hohe breite Buchstaben haben, ist das auch möglich. Sie können die 8*8-Matrix dabei voll ausnutzen! Allerdings müssen Sie dann die Schrittweiten vergrößern, da sonst die Buchstaben aneinanderhängen. Probleme gibt es jedoch dann mit der inversen Schrift: Es bleiben immer Lücken zwischen den Buchstaben, die Sie mit Line oder Draw ausfüllen müssen.

Soweit nun die Regeln, die Sie bei der Erstellung eigener Zeichensätze beachten müssen. Zum Erstellen selbst gibt es nicht viel zu sagen. Das wohl geeignetste Hilfsmittel dabei ist die Zoom-Funktion, in der sich die einzelnen Punkte exakt setzen oder löschen lassen.

Wollen Sie einen bestehenden Zeichensatz modifizieren, so ist auch der Textmodus selbst ein hilfreiches Instrument: Einzelne Zeichen lassen sich mit DEL oder INST herauslöschen, wobei jedoch immer darauf zu achten ist, daß sich der Cursor in 8-Punkte-Schritten bewegt und synchronisiert ist, d.h. von der HOME-Position aus gestartet wird. Sollen Zeichen nur ihren Platz wechseln, so löschen Sie die alte Eingabe und drucken statt dessen die neuen Zeichen an deren Stelle. Auch wenn ein Zeichen erstellt werden soll, das einem anderen sehr ähnlich ist, kann man letzteres als Ausgangsbasis benutzen. So wird man zum Beispiel für die deutschen Umlaute die ensprechenden Vokale an die gewünschte Stelle setzen und dann die Pünktchen hinzufügen.

Will man die inversen Zeichen erstellen, verschiebt man die nicht inversen mittels Move-Befehl in den Bereich für die inversen oder tippt sie mittels Text-Befehl dorthin - wenn es nicht viele sind - und wendet dann die bereichsweise Invertierung an, die im Abschnitt über die Verknüpfungsbefehle beschrieben wurde. Da sowohl diese Invertierung als auch der Zeichensatz an das 8*8-Punkte-Raster gebunden ist, paßt das sehr gut zusammen.

Soll ein komplett neuer Zeichensatz erstellt werden, empfiehlt es sich, als *Unterlage* den Commodore-Groß-Kleinschrift-Zeichensatz zu verwenden, den Sie dann Zeichen für Zeichen mit den neugestalteten Buchstaben überschreiben. Ohne Unterlage wäre Ihnen nicht bekannt, welches Zeichen wo zu stehen hat! Natürlich können Sie auch - wo es sinnvoll ist - Zeichen durch andere ersetzen und beispielsweise die eckigen Klammern durch Umlaute überschreiben. Wenig sinnvoll, aber dafür um so lustiger ist es, wenn man etwa an die Stelle, an der das **A** stehen sollte, ein **B** tippt, oder den ganzen Zeichensatz mittels Move-Befehl um ein Zeichen nach rechts verschiebt. Probieren Sie das doch einmal aus! Sie werden staunen, was Sie im Text-Modus plötzlich für einen Unsinn schreiben!

2.11 Mirror und Turn

Wie Sie bereits aus der Bedienungsanleitung erfahren haben, können Sie mit den den Befehlen Mirror (**SHIFT M**) und Turn (**SHIFT T**) den ganzen Bildschirm oder beliebige Ausschnitte daraus drehen oder spiegeln. Bei Mirror wird dabei zur Senkrechten gespiegelt, bei Turn um 180 Grad gedreht. Das entspricht auch den gleichnamigen Befehlen im Sprite-Editor. Falls Sie übrigens den Befehl Rotate zum Drehen um 90 Grad, den es ja im Sprite-Editor gibt, im Grafik-Editor vermissen: Er steckt in der Erweiterung und wird somit erst in Kapitel 4 besprochen.

Nachfolgend möchte ich Ihnen nun ein eindrucksvolles Beispiel dafür geben, wie sich die Befehle Mirror und Turn anwenden lassen. Sehen Sie sich dazu Abb. 2.32 an.

Bild 2.32: Eindrucksvolles Kaleidoskop, mit wenig Aufwand realisiert

Grundlage für dieses filigrane Kaleidoskop ist das recht einfache Bild in Abb. 2.33. Dieses Bild wurde, wie Sie leicht erkennen können, mittels Circle und Paint erstellt. Zeichnen Sie einige Kreise - nicht zu viele! - und malen Sie diese mit Paint so aus, daß nie zwei unmittelbar benachbarte Felder ausgefüllt sind. Nun kopieren Sie dieses Bild dreimal, da wir insgesamt viermal dasselbe Bild benötigen.

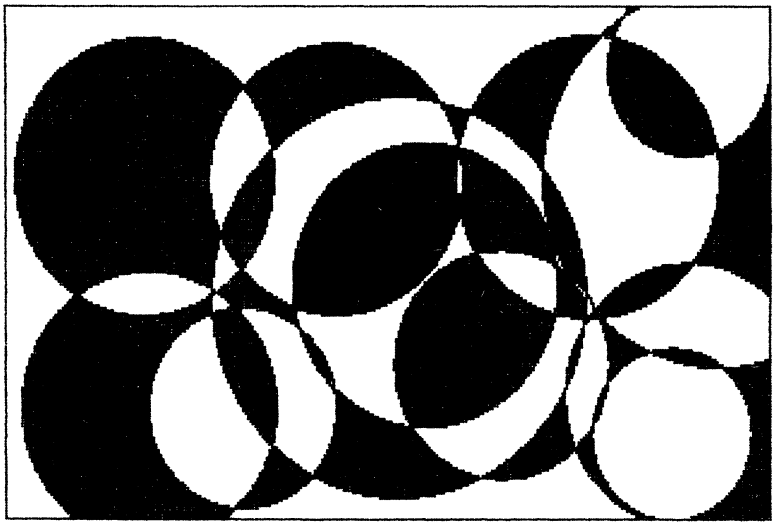


Bild 2.33: Die Grundlage zum Kaleidoskop in Bild 2.32

Das erste Bild belassen Sie so, wie es ist. Das zweite spiegeln Sie (Mirror), das dritte drehen Sie (Turn) und auf das vierte Bild wenden Sie beide Operationen an. Nun verknüpfen Sie alle vier Bilder durch Exor, und schon haben Sie das Kaleidoskop!

Natürlich hätten wir als Ausgangsbasis nicht das Bild mit den Kreisen gebraucht - jedes beliebige Bild kann so behandelt werden. Es sollte allerdings aus wenigen, größeren Flächen bestehen, sonst wird das Ergebnis zu unübersichtlich und zu fein gerastert. Wenn Sie nun noch die Schrift betrachten, die ich in das Kaleidoskop eingefügt habe, werden Sie feststellen, daß die Buchstaben ein schmaler Rand umschließt, in dem alle Punkte gelöscht sind. Ohne diesen Rand könnte man die Buchstaben kaum erkennen. Wie man diesen Rand einfach und schnell erzeugt, erfahren Sie in Kapitel 4 bei den Makrobefehlen.

Mit solchen Kaleidoskopen können Sie Ihre Glückwunschkarten, Einladungen oder Sonstiges verzieren. Sie sehen, Hi-Eddi plus ist keineswegs nur auf technische Anwendungen beschränkt!

2.12 Farbige Bilder

Die Farbe ist bei den bisherigen Erörterungen weitgehend außer Acht gelassen worden. Das liegt wohl daran, daß ich selbst fast ausschließlich Schwarzweißes, weil Technisches, mit Hi-Eddi plus erstelle. Für Bilder zur Dokumentation, für Schaltungen etc. ist dieses Programm für mich zu einem selbstverständlichen Werkzeug geworden, genauso wie das Textverarbeitungsprogramm, mit dem ich meine Korrespondenzen, Programmbeschreibungen und schließlich auch dieses Buch erstelle.

Aber Hi-Eddi plus kann auch farbig zeichnen! Zum Beispiel können Sie damit die Bilder zu einem selbst erstellten Grafikadventure malen. Die Erweiterung bietet sogar eine Möglichkeit, Farbbilder in einem Format auf Diskette abzuspeichern, das ich extra für Grafik-Adventures entwickelt habe. Mehr dazu erfahren Sie in Kapitel 4.

Ein Problem bei farbigen Bildern sind natürlich die eingeschränkten Farbfähigkeiten des High-Resolution-Modus. Sie wissen ja schon aus Kapitel 1: Auf einem 8*8-Punkte-Feld dürfen nur zwei verschiedene Farben vorkommen. Beim Multicolor-Modus dagegen stehen auf einem Feld gleicher Größe - es umfaßt wegen der halben Auslösung nur 4*8 Punkte - vier Farben gleichzeitig zur Verfügung, von denen allerdings eine für alle Felder des Bildschirms dieselbe ist.

Wir können dieses Problem jedoch umgehen, indem wir bereits beim Zeichnen die Einteilung des Bildes in 8*8-Punkte-Felder berücksichtigen. Wie das möglich ist, wissen Sie schon. Wir haben diese Möglichkeit bereits beim Move-Befehl angewandt, da dieser auf dieselbe Rasterung wie die Farben beschränkt ist.

Als Beispiel dient das bunte Bild mit der Burg, das Sie auf der Titelseite sehen. In Abb. 2.34 ist das Bild ohne Farben dargestellt.

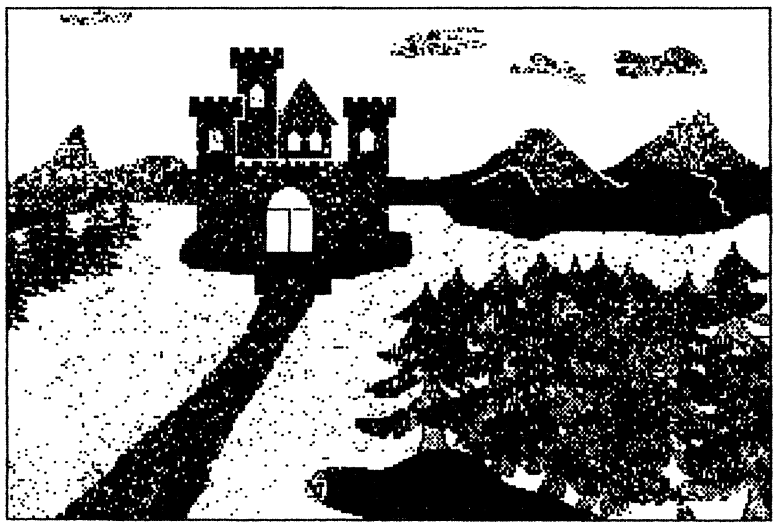


Bild 2.34: Die Burg in Schwarzweiß-Darstellung

Sehen wir uns ein Detail daraus, nämlich das Burgtor mit der Zugbrücke davor, genauer an. Hier treffen mehrere Farben auf engstem Raum aufeinander: die graue Mauer, das orange Tor, die braune Zugbrücke, der blaue Wassergraben und das grüne Gras. Solche Ansammlungen von Farben sind das größte Problem beim Malen im High-Resolution-Modus.

Vergleichen Sie nun das bunte Bild mit der nicht kolorierten Zeichnung. Wie Sie sehen, ist zwischen Mauer, Graben und Zugbrücke keine Trennung zu erkennen, doch dort verläuft die Trennlinie zwischen benachbarten 8*8-Punkte-Feldern! Anders ausgedrückt: Die Mauer geht bis an die Unterkante eines Feldes, die Zugbrücke geht von der linken Kante eines Feldes bis zur rechten Kante. Bereits beim Zeichnen wurden Zugbrücke, Burg, Wassergraben und Tor entsprechend plaziert! Wie Sie dabei vorgehen, haben Sie beim Move-Befehl erfahren: Mittels programmierbarer Schrittweite (F3) bewegen wir den Cursor aus einer Eckposition heraus und wissen dann immer, in welchem Eck eines 8*8-Punkte-Feldes er steht.

Berücksichtigt man diese Einteilung, kann man Bilder erzeugen, denen man die eingeschränkten Farbmöglichkeiten des High-Resolution-Modus überhaupt nicht ansieht. Dennoch unterscheiden sie sich deutlich von Multicolor-Bildern, allerdings im positiven Sinne wegen ihrer doppelt so hohen Auflösung. Die runden Ufer des Wassergrabens und des Sees zum Beispiel würden in einem Multicolor-Bild eckiger wirken!

Neben der Farbe kommt nun endlich auch einmal das Koala-Pad zum Einsatz. Für die runden Ufer, die Wolken und die Bäume ist das Pad geradezu ideal. Man zeichnet diese Gebilde fast so, wie man mit Bleistift und Papier umgeht. Mit dem Joystick ist das Erstellen dieser unregelmäßigen Gebilde eine äußerst mühsame und zeitraubende Angelegenheit.

Übrigens habe ich für das Bild nur einen einzigen Baum gezeichnet. Mit Move, kombiniert mit der Oder-Verknüpfung, habe ich ihn dann mehrmals ins Bild gesetzt. Damit mehr Abwechslung ins Bild kommt, habe ich sogar mit Mirror einen zweiten Baum aus dem ersten erzeugt!

Eine häufig verwendete Funktion beim Erstellen dieses Bildes war auch die Spray-Funktion. Ein paar Blumen auf der Ritterwiese, ein paar Löcher in der alten Mauer und wenn der Wald irgendwo zu dicht wird, dann hilft dagegen ebenfalls die Spray-Funktion.

3 Druckeranpassung

Nach der Veröffentlichung des Hi-Eddi in der Zeitschrift 64'er mußte ich feststellen, daß die Druckeranpassung vor allem für Laien ein großes Problem darstellt. Erschwerend kommt hinzu, daß es nicht nur eine unübersehbare Vielfalt von Druckern, sondern auch eine Menge verschiedener Interfaces gibt. Wenn ein Interface das Prädikat *grafikfähig* trägt, dann heißt das noch lange nicht, daß es auch mit jedem Grafikprogramm einwandfrei zusammenarbeitet.

Deshalb habe ich bereits beim Hi-Eddi im 64'er die Druckerroutine so geschrieben, daß jegliche Grafikfunktionen des Interfaces umgangen werden und somit die Verwirrung bezüglich der Interfaces eigentlich ausgeschlossen sein sollte. Doch es war nicht an dem! Obwohl ich in der Ausgabe 3/85 von 64'er ausdrücklich darauf hingewiesen habe, erhielt ich viele Anrufe: "Mein Interface ist grafikfähig und trotzdem funktioniert es nicht!"

Beim Hi-Eddi plus ist nun jedoch einiges anders, denn die neue Druckerroutine kann auch die Commodore-Drucker VC1525, MPS801 und MPS803 ansteuern. Die Druckerroutine ist bereits für diese Drucker voreingestellt. Wenn Sie einen dieser Drucker besitzen, brauchen Sie keine Anpassung mehr vorzunehmen, sondern können sofort drucken.

Da die meisten grafikfähigen Hardware-Interfaces diese Drucker emulieren - das heißt sie übersetzen die vom Computer kommenden Grafikdaten in ein für den angeschlossenen Drucker verständliches Format - laufen auch die meisten damit betriebenen Drucker problemlos mit Hi-Eddi plus.

Wie Sie schon in Kapitel 1 erfahren haben, können Sie die Druck-Fähigkeiten des Hi-Eddi plus mit einem der oben genannten Commodore-Drukker bei weitem nicht ausnutzen. Das gilt auch, wenn Sie ein grafikfähiges Interface benutzen!

Deshalb sollten Sie einen Epson- oder einen Epson-kompatiblen Drucker auch als solchen ansteuern. An dieser Stelle betone ich noch einmal: Die Grafikfähigkeiten eines Interfaces sind dann nicht mehr gefragt! Sie müssen das Interface so einstellen, als ob es ein einfaches Kabel wäre, das die Daten ohne irgendeine Übersetzung oder Veränderung an den Drucker übergibt! Wie Sie das erreichen, dazu kommen wir gleich.

Sie können auch ein Kabel zum Anschluß am Userport benutzen, denn die Druckerroutine des Hi-Eddi plus beeinhaltet ein Software-Interface, das einen am Userport angeschlossenen Drucker automatisch bedient.

Kommen wir nun zur Anpassung selbst, die, wie bereits erwähnt, für Epson- und Epson-kompatible Drucker funktioniert. Laden Sie dazu das Programm **DRUCKER** von der beiliegenden Diskette und starten Sie es mit **RUN**.

Wenn Sie keinen Epson-Drucker besitzen, werden Sie sicherlich fragen, ob denn Ihr Drucker wenigstens Epson-kompatibel ist. Doch zunächst, was ist überhaupt mit **kompatibel** gemeint?

Jeder Drucker muß über bestimmte Steuerzeichen oder Sequenzen programmiert werden. Man muß ja dem Drucker mitteilen, in welcher Schriftart er die Zeichen drucken soll oder ob die gesendeten Bytes vielleicht gar keine Schriftzeichen, sondern Grafikbytes sind. Die Steuerzeichen der Epson-Drucker sind dabei ein gewisser Standard geworden, an den sich die meisten Hersteller von Druckern halten. Zwar gibt es geringe Abweichungen, die vor allem durch die Fähigkeiten des Druckers bedingt sind, aber die sind für die vorliegende Druckerroutine und das Anpaßprogramm **DRUCKER** kein Problem.

Wenn Ihr Drucker in der Liste, die Ihnen das Anpaßprogramm nach dem Start anbietet, enthalten ist, dann ist er Epson-kompatibel. Sie brauchen dann die Steuersequenzen gar nicht mehr einzugeben, da diese bereits im Programm gespeichert sind.

Es werden dann nur noch drei weitere Fragen gestellt, auf die ich nun eingehen möchte.

Zunächst werden Sie nach der Primär-(Geräte-)Adresse und der Sekundäradresse gefragt. Falls Sie den Drucker am Userport betreiben, dann können Sie hier zweimal RETURN eintippen, weil diese Adressen dann unwichtig sind.

Wenn Sie jedoch den Drucker mittels Hardware-Interface am seriellen Bus betreiben, müssen Sie mit diesen Adressen das Interface auf Direktmodus einstellen. Damit sind wir wieder bei der Grafikfähigkeit und müssen deshalb noch einmal sehr ausführlich werden:

Ein Hardware-Interface besitzt verschiedene Betriebsarten. Es gibt zum Beispiel eine Betriebsart, in der sich der Drucker wie ein Commodore-Drucker verhält; oder es sind verschiedene Grafik-, List-, oder Text-Betriebsarten möglich. Was wir jedoch brauchen, ist der sogenannte Direktmodus oder Linearmodus. Dabei werden sämtliche Funktionen des Interfaces abgeschaltet und die Daten werden unverändert vom Computer an den Drucker übergeben.

Eingestellt wird dieser Direktmodus, den alle Hardware-Interfaces besitzen, über die oben erwähnten Adressen; genauer gesagt ist es meist nur die Sekundäradresse. Die Primäradresse ist in der Regel 4. Die Sekundäradresse jedoch ist die, auf die es ankommt. Den richtigen Wert dafür müssen Sie der Bedienungsanleitung zu Ihrem Interface entnehmen. Einige Beispiele für die gängigsten Interfaces zeigt Ihnen das Druckeranpaßprogramm bereits. Wie Sie dort sehen, dient meist die Sekundäradresse 1 zur Ansteuerung des Direktmodus. Leider halten sich jedoch nicht alle Interface-Hersteller daran, so daß eine feste Einstellung dieses Wertes nicht möglich ist.

Die dritte Frage bezieht sich darauf, ob Ihr Drucker nach einem Carriage-Return (ASCII-Code 13) einen Linefeed (ASCII-Code 10) braucht oder nicht - man spricht dann von Auto-Linefeed. Die meisten Drucker brauchen keinen Linefeed, deshalb ist das **n** bereits vorgegeben, so daß Sie nur noch RETURN zu drücken brauchen. Sollte jedoch Ihr Drucker dann immer nur auf dieselbe Zeile drucken, bis sich das Papier auflöst, braucht er einen Linefeed. In diesem Falle müssen Sie also die Druckeranpassung wiederholen und bei der entsprechenden Frage mit **j** antworten.

Soviel dazu, wenn Ihr Drucker in der Liste des Anpaßprogramms aufgeführt ist. Was geschieht aber, wenn dies nicht der Fall ist?

In diese Fall gibt es zwei Möglichkeiten: Entweder er ist doch Epson-kompatibel, aber nicht in der Liste enthalten. Das ist durchaus möglich, denn es gibt eine große Anzahl von Epson-kompatiblen Druckern, da sich sehr viele Hersteller nach diesem Quasi-Standard richten. Deshalb kann es möglich sein, daß ich einige der weniger bekannten Drucker nicht in die Liste aufgenommen habe. Außerdem kommen laufend neue Drucker auf den Markt, die natürlich auch noch nicht berücksichtigt sein können.

Die zweite Möglichkeit: Sie besitzen einen der sogenannten *Exoten*, die absolut nicht Epson-kompatibel sind. Daß es solche Drucker gibt, mußte ich nach der Veröffentlichung des Hi-Eddi im 64'er erfahren. Dabei ist die Epson-Kompatibilität der Druckerroutine des Hi-Eddi plus wirklich sehr großzügig. Immerhin besteht die Möglichkeit, sämtliche Steuersequenzen selbst einzugeben, wobei auch bezüglich der Länge ein Spielraum vorhanden ist: Fünf Zeichen darf eine Steuersequenz lang sein (bei Epson ist sie zwei bis drei Zeichen lang). Die Druckerroutine bietet sogar die Möglichkeit, die Grafikbytes auf den Kopf zu stellen, was einige Drucker benötigen. Woran es jedoch scheitern kann, ist die Übergabe der Grafik-Bytes. Bei Epson wird vor einer Zeile mit Grafikbytes die Länge dieser Zeile im üblichen Low/High-Byte-Format übergeben. Einige exotische Druckerhersteller scheinen jedoch eine rege Phantasie zu haben, wenn es darum geht, eigene (und unhandliche!) Prozeduren zur Übergabe von Grafikbytes zu entwickeln.

Doch der langen Rede kurzer Sinn: Falls Sie im Besitz eines dieser Exoten sind, kann ich Ihnen leider nicht helfen.

Doch kehren wir zum ersten Fall zurück, nämlich zu den Epson-kompatiblen Druckern. Möglicherweise ist dieser Drucker dann auch zu einem in der Liste aufgeführten voll kompatibel, wodurch Sie sich natürlich die Eingabe der Sequenzen ersparen könnten. Am besten erfahren Sie das von Ihrem Händler oder aus den Informationen oder der Bedienungsanleitung zum Drucker. Außerdem darf man mit einer gewissen Kontinuität innerhalb der Produktpalette eines Herstellers rechnen: Oft sind neue Drucker kompatibel zu ihren Vorgängern. Meist können jedoch die neueren Typen mehr als ihre Vorgänger, weshalb sich eine neue Anpassung lohnt.

Sehen wir uns nun an, wie Sie dabei vorgehen. Eine Kurzanweisung gibt bereits das Anpaßprogramm selbst und dem einigermaßen erfahrenen Anwender wird das genügen. Damit jedoch keine Unklarheiten entstehen, werde ich hier noch einmal ausführlicher auf die Anpassung eingehen.

Am wichtigsten ist dabei das Handbuch zum Drucker (nicht die Bedienungsanleitung zum Interface!)

Die erste Sequenz, die wir eingeben müssen, ist diejenige zum Einschalten des CRT-Grafik-Modus für 80 Punkte/Inch, was 640 Punkte/Zeile ergibt. CRT steht für Cathode Ray Tube, deutsch: Bildschirm. Dieser Grafikmodus ist speziell für den Ausdruck von Bildschirm-Grafiken gedacht, was auch aus der Anzahl von 640 Punkte/Zeile ersichtlich ist. Da der Grafikbildschirm des C64 320 Punkte breit ist, kann man ihn gerade zweimal neben-

einander oder einmal in doppelter Größe auf das Blatt drucken. Nach der Sequenz, für die das Anpaßprogramm ein Beispiel gibt, käme noch die Anzahl der auszugebenden Bytes im Low/High-Format. Diese brauchen Sie jedoch nicht einzugeben, denn darum kümmert sich die Druckerroutine selbst.

Zur Eingabe der Sequenzen ist nicht viel zu sagen. Die erlaubten Eingaben gibt das Anpaßprogramm selbst an. Wenn Sie eine Eingabe tätigen, die das Programm nicht versteht oder die ihm nicht plausibel erscheint, gibt es eine entsprechende Meldung aus und fordert Sie auf, die Eingabe zu wiederholen. In der Regel beginnen alle Steuersequenzen mit ESC, deshalb wird das Programm mißtrauisch, wenn Sie dieses Zeichen nicht als erstes eingeben. Bei ESC spielt die Groß/Klein-Schreibung keine Rolle, Sie können auch Esc oder esc eingeben. Anders ist es dagegen bei ASCII-Zeichen in Hochkommas. Hier müssen Sie natürlich einen Buchstaben so eingeben, wie es das Druckerhandbuch verlangt.

Falls Ihr Drucker über einen Plot-Modus verfügt, geben Sie als nächstes die Sequenz dafür ein. Was ist der Unterschied zum CRT-Modus?

Beim CRT-Modus ist die Punktdichte für die Ausgabe von Bildschirmgrafiken optimiert; daher die 640 Punkte/Zeile. Das Ergebnis ist dabei jedoch meist, daß das Verhältnis von Höhe zu Breite nicht exakt 1:1 ist. Ein Kreis wird beispielsweise zu einem Ei. Deshalb bieten einige Drucker den Plot-Modus, dessen Punktdichte so gewählt ist, daß ein Kreis auch wirklich rund wird. Dies ist insbesondere beim Ausdrucken von Platinenvorlagen wichtig, wie ich in Kapitel 2 bereits angedeutet habe. Bei den mir bekannten Druckern, die einen Plot-Modus besitzen, ergibt sich damit eine Punktdichte von 72 Punkten/Inch entsprechend 576 Punkten/Zeile. Das paßt natürlich mit den 320 Punkten des Bildschirms überhaupt nicht mehr zusammen. Hi-Eddi plus druckt deshalb im Plot-Modus nur so viel aus, wie auf das Papier paßt: Bei zwei kleinen Bildern nebeneinander gehen die rechten 64 Punkte (8 Spalten) des rechten Bildes verloren, beim Ausdruck eines großen Bildes die rechten 32 Punkte (4 Spalten). Ein kleines Bild wird natürlich vollständig ausgedruckt. Es ist also bereits beim Erstellen eines Bildes - insbesondere natürlich eines Platinenlayouts - zu berücksichtigen, daß es nicht ganz ausgedruckt wird.

Da die Punktdichte im Plot-Modus davon abhängt, wie dicht die Nadeln im Druckkopf beieinander liegen, wäre es theoretisch möglich, daß es Drucker gibt, die im Plot-Modus eine andere Anzahl von Punkten in eine Zeile drucken. Das gilt natürlich auch für breite Drucker. Es könnte auch Drukker geben, die einen Plot-Modus gar nicht brauchen, weil sie bereits im

CRT-Modus maßstabgetreu drucken. Mir sind solche Drucker allerdings nicht bekannt.

Falls Sie die Anzahl der auszudruckenden Bytes ändern wollen, müssen Sie in Kapitel 5 bei der Dokumentation aufschlagen und sich mit einem Monitor ans Werk machen. Im Rahmen der hier besprochenen Druckeranpassung sind solche Eingriffe nicht vorgesehen.

Nun wieder zurück zu unseren Sequenzen. Als nächstes erfragt das Anpaßprogramm die Sequenz für den Zeilenvorschub im Grafik-Betrieb. Damit soll erreicht werden, daß die einzelnen Grafikzeilen nahtlos aneinander ausgedruckt werden. Ich empfehle Ihnen, diesen Abstand lieber ein bißchen zu knapp zu wählen, denn weiße Nähte in der Hardcopy sind unangenehmer als ein leichtes, gegenseitiges Überdrucken der Zeilen. Das Beispiel, das das Anpaßprogramm gibt, nämlich 24/216 Inch, gibt den korrekten Wert an. Bei den Parametern, die im Programm fest gespeichert sind, also für die Drucker in der Liste, habe ich dagegen immer etwas kleinere Werte genommen. Möglicherweise ist dies für einige Drucker nicht optimal. In diesem Fall müßten Sie die Sequenzen selbst eingeben, statt den Drucker aus der Liste auszuwählen.

Nun kommen zwei Sequenzen, zu deren Erklärung ich etwas weiter ausholen muß. Möglicherweise besitzt ihr Drucker auch eine Sequenz, mit der der sogenannte Double-Strike-Modus angewählt wird. Diese Sequenz benötigen wir jedoch nicht, da dieser Modus in der Regel nur bei Textausgabe funktioniert. Dabei wird jede Zeile zweimal gedruckt, das zweite Mal jedoch etwas versetzt, d.h. das Papier wird vor dem zweiten Druck ein winziges Stück weitergeschoben. Somit werden beim zweiten Druck die Zwischenräume zwischen den einzelnen Punkten aufgefüllt, wodurch das Schriftbild verbessert wird.

Hi-Eddi plus bietet nun dasselbe für den Grafik-Ausdruck. Dabei braucht Ihr Drucker den Double-Strike-Modus gar nicht zu beherrschen, denn die Druckerroutine macht das selbst. Sie druckt jede Zeile zweimal und schiebt zwischen diesen beiden Ausdrucken das Papier ein kleines Stück weiter.

Diese Fähigkeit bringt nicht nur ein besseres Druckbild mit sich, sondern ist besonders beim Ausdrucken von Platinenlayouts (in Zusammenhang mit dem Plot-Mode) sehr wichtig. Außerdem erhält man damit auch bei einem alten Farbband noch einen lesbaren Ausdruck.

Insbesondere im Zusammenhang mit dem Plot-Mode ist es jedoch wichtig, daß der Zeilenvorschub exakt stimmt, sonst ist der Vorteil des Plot-Modus, nämlich das Verhältnis von 1:1 von Höhe zu Breite, aufgehoben. Deshalb können die beiden im Double-Strike-Modus benützten Zeilenvorschübe - als erstes der zwischen zwei Zeilen und als zweites der Mini-Vorschub zwischen dem ersten und zweiten Druck derselben Zeile - getrennt vom normalen Zeilenvorschub definiert werden. Die Summe der beiden Zeilenvorschübe sollte den exakten Wert für einen vollständigen Zeilenvorschub ergeben. Für den Epson FX 80 zum Beispiel ist der exakte Wert 24/216 Inch, den man in 23/216 Inch und 1/216 Inch zerlegt. Wie Sie sehen, kann somit der Wert, den man für einen Zeilenvorschub mit und ohne Double-Strike verwendet, sogar gleich sein - ich habe Ihnen bereits weiter oben, im Zusammenhang mit dem normalen Zeilenvorschub geraten, den Wert etwas kleiner zu wählen. Da dies jedoch stark vom jeweiligen Drucker abhängt, sollten Sie hier am besten selbst experimentieren.

Die nächste einzugebende Sequenz schließlich stellt den Drucker nach einer Hardcopy wieder auf den für Textausgabe verwendeten Zeilenvorschub, üblicherweise auf 1/6 Inch/Zeile. Eine eventuelle Textausgabe nach der Grafik-Hardcopy funktioniert somit wieder normal.

Nach der Eingabe dieser Sequenzen folgt zunächst die Frage, wie viele Blanks die Druckerroutine vor einem kleinen Bild im CRT-Modus ausdrukken soll. Da man das Bild normalerweise in der Mitte des Blattes haben will, ergibt sich der Wert von 20 Leerzeichen. Hier ist natürlich darauf zu achten, daß der Drucker vor einer Hardcopy auf Normalschrift (80 Zeichen/Zeile) steht. Bei einer engeren oder breiteren Schrift würde nämlich das Bild nach links oder rechts verschoben.

Falls der Drucker einen Plot-Modus besitzt, können Sie auch dafür die Anzahl der Blanks vor einem kleinen Bild angeben. Durch den Plot-Modus ist das Bild nämlich breiter, und man braucht weniger Blanks, um es in die Mitte zu rücken.

4 Erweiterungen zu Hi-Eddi plus

Sicherlich sind Sie gespannt, was ich Ihnen in diesem Kapitel, auf das ich bereits mehrfach hingewiesen habe, eröffnen werde. Deshalb eine kurze Übersicht, was Sie erwartet:

Ich werde Ihnen in diesem Kapitel eine Befehlserweiterung zu Hi-Eddi plus vorstellen, die einiges zu bieten hat: einen Scroller, der es Ihnen ermöglicht, den sichtbaren Bildschirm wie ein Fenster kreuz und quer über ein 640*600 Punkte großes Bild zu verschieben, dann eine in den Bildschirm einblendbare Koordinatenanzeige, einen Befehl zum Rotieren beliebiger Bildschirmausschnitte, erweiterte Load- und Save-Befehle sowie die Möglichkeit, eine Sequenz von Befehlen zu einem Makrobefehl zusammenzuhängen. Für den Profi ist diese Erweiterung auch als Beispiel und Anregung gedacht, denn wer Assemblerprogrammierung beherrscht, kann auch eigene Erweiterungen zu Hi-Eddi plus schreiben! Dazu folgt im nächsten Kapitel eine umfassende Dokumentation des Programms, die alle Arbeitsspeicherzellen und Routinen des Hi-Eddi plus erklärt. Damit kann der Profi Routinen aus dem Hi-Eddi plus in eigene Programme übernehmen oder, was noch interessanter ist, das Programm individuell erweitern.

Als Maschinensprache-Programmierer, kenne ich sehr gut den Wunsch, auch gute Programme noch weiter zu verbessern. Hat man dazu keine Dokumentation, ist dies äußerst schwierig. Man ist dann auf das Disassemblerlisting, das man sich selbst erstellen kann, angewiesen, und das ist ein sehr mühsames Vorgehen. Hi-Eddi plus ist dagegen sehr erweiterungsfreundlich: Es ist stark modular aufgebaut. Das Hauptprogramm besteht aus wenigen Zeilen, alles Wesentliche geschieht in Unterprogrammen. Außerdem sind in der Tabelle, in der alle Befehle zusammengefaßt sind, speziell für Erweiterungen einige Plätze frei. Die unter dem Namen **EXT** auf der Diskette befindliche Erweiterung ist ein Beispiel für die Erweiterbarkeit von Hi-Eddi plus. Sehen wir uns dieses Programm nun einmal näher an.

4.1 Die Erweiterung auf der Diskette

Vielleicht fragen Sie sich, warum es so eine Erweiterung überhaupt gibt. Warum habe ich die zusätzlichen Möglichkeiten nicht fest in den Hi-Eddi plus eingebaut? Um das zu erklären, muß ich etwas ausholen:

Wenn Sie den alten Hi-Eddi aus dem 64'er kennen, wissen Sie schon einiges über die Enstehungsgeschichte dieses Programms. Angeregt wurde ich dadurch, daß mein chinesischer Freund seinen neu erworbenen C64 dazu nutzte, seinen Namen auf dem Bildschirm darzustellen. In Simon's BASIC brauchte er dafür ein viele Kilobyte langes Programm, um die komplizierten chinesischen Schriftzeichen zu definieren. Zufällig entdeckte ich zur gleichen Zeit ein Grafikprogramm in der amerikanischen Zeitschrift COMPUTE, das es ermöglichte, die in einem integrierten Sprite-Editor erstellten Sprites in das Grafikbild einzufügen. Das war für das Erstellen und Drukken chinesischer Zeichen geradezu ideal! Zu einer richtigen chinesischen Textverarbeitung ist ein Programm dieser Art natürlich nicht geeignet, aber für Adreßaufkleber, Namensschilder oder Glückwunschkarten reicht es.

Nachdem ich das Programm abgetippt und etwas genauer untersucht hatte, mußte ich jedoch feststellen, daß der hier angewandte Programmierstil an Effektivität und Übersichtlichkeit doch sehr zu wünschen übrig ließ. Denn, obwohl das Programm recht umfangreich war, waren Fähigkeiten und Geschwindigkeit doch sehr bescheiden. Die oben bereits angedeutete Möglichkeit, bestehende Programme zu verbessern, war hier nicht anzuwenden.

Die Anregungen, die ich durch das Programm erhielt, waren allerdings gut. Und so begann ich, einen eigenen und viel leistungsfähigeren Grafikeditor zu schreiben, wobei ich mir selbst hohe Auflagen machte.

Eine dieser Auflagen, die unter keinen Umständen angetastet werden durfte, war: Das Programm sollte die maximal mögliche Anzahl von Grafikbildschirmen zur Verfügung stellen. Wie viele sind das? Der C64 besitzt 64 kByte RAM und eine Bitmap, d.h. der Speicherbereich für ein Grafikbild, ist 8 kByte lang. Demgemäß haben acht Grafikbildschirme im Rechner Platz. Theoretisch ist das zwar richtig, aber in der Praxis stimmt es nicht, denn zur Grafikdarstellung benötigt der VIC noch das sogenannte Video-RAM, in dem die Farbinformationen liegen und das beansprucht 1 kByte. Dazu kommt noch etwa ein kByte für reservierte Speicherbereiche wie Zeropage, Stack und Systemvariable. Außerdem braucht das Programm selbst auch noch Platz!

Somit ist die in der Praxis erreichbare Zahl von Bitmaps auf sieben beschränkt. Diese Forderung ist immer noch hart genug, denn wenn man den Speicherbedarf für die Bitmaps und die 2 kByte für Video-RAM und System abzieht, bleiben für das Programm selbst gerade 6 kByte. Selbst einfachere Grafikerweiterungen, die nicht so viel können wie Hi-Eddi oder Hi-Eddi plus, brauchen meist schon 8 kByte! Ich habe es dennoch versucht, und es hat geklappt! Der Hi-Eddi aus dem 64'er nutzt diesen Bereich noch nicht einmal ganz aus. Den fraglichen Befehl **.BY $2C**, den sogar Commodore in seinem Betriebssystem verwendet, um 2-Byte-Befehle zu verstekken, werden Sie beispielsweise im Hi-Eddi plus nicht finden. Allein durch effektive Algorithmen und einen strukturierten Programmaufbau, in dem ähnliche Programmteile als universell verwendbare Unterprogramme geschrieben sind, erreiche ich diese Kompaktheit.

Doch nach der Veröffentlichung des Hi-Eddi im 64'er erreichte mich bald eine Flut von Verbesserungsvorschlägen und ich begann, die besten in die Tat umzusetzen. Vor allem durch die Umsetzung des BASIC-Teils des alten Hi-Eddi in Maschinensprache konnte ich viel Platz für neue Befehle gewinnen. Aber irgendwann war eben dann doch Schluß - mit dem Speicherplatz - mit den Ideen allerdings noch lange nicht!

Da ich jedoch unter keinen Umständen den siebten Bildschirm antasten wollte, entschied ich mich für eine andere Lösungsmöglichkeit, die es dem Benutzer erlaubt, selbst zwischen maximaler Anzahl der Grafikbildschirme und maximalen Fähigkeiten zu wählen. So hatte ich es auch bereits bei der Wahl zwischen Farbe und Schwarzweiß realisiert.

Ich entschied mich dafür, die zusätzlichen Fähigkeiten in eine Erweiterung zu packen, die man bei Bedarf nachladen kann. Damit geht aber leider ein Grafikbildschirm verloren, denn die Erweiterung braucht auch Platz.

Man erhält damit auf Anhieb 8 kByte zusätzlichen Programm-Speicherplatz, was mehr ist als der ganze Hi-Eddi plus braucht! Da die Erweiterung EXT noch nicht einmal 4 kByte braucht, gibt es hier noch ein reiches Betätigungsfeld für Maschinensprachetüftler.

Nach diesem kleinen Exkurs werden wir nun endlich die Erweiterung auch anwenden!

4.2 Laden der Erweiterung

Wie bereits gesagt, wird die Erweiterung in den Speicherbereich für ein Grafikbild geladen. Der Ladevorgang ist der gleiche wie bei einem Grafikbild, also **C= L** und **G**. Hi-Eddi plus erkennt von selbst, daß kein Bild, sondern eine Erweiterung geladen wurde und startet sie daraufhin, damit sie sich initialisieren kann.

Zu dieser Initialisierung gehört auch, daß der Bildschirmspeicher, in den Sie die Erweiterung geladen haben, blockiert wird. Er ist dann nicht mehr zugänglich!

Doch in welchen Speicher sollen Sie die Erweiterung laden? Durch die Auswahl der Nummer, Sie benutzen, beeinflussen Sie auch die Fähigkeiten der Erweiterung und sogar die des Hi-Eddi plus.

Beginnen wir mit Speicher Nummer 1. Dieser Speicher beherbergt in der Menübetriebsart die Menütafel. Wenn Sie die Erweiterung dennoch in der Menübetriebsart in Speicher 1 laden, wird diese Betriebsart automatisch abgeschaltet, und Hi-Eddi plus läuft in der Farb-Betriebsart.

Eine weitere Sonderstellung nimmt der Bildschirm 6 bei Farb- und Menü-Betriebsart bzw. 7 bei Schwarzweiß ein: Er dient als Leinwand für den Walk-Befehl. Lädt man die Erweiterung in diesen Speicher, wird der Walk-Befehl blockiert. Ich denke jedoch, daß man bei Anwendung des Walk-Befehls sowieso alle Bildschirme zur Verfügung haben möchte und deshalb auf die Erweiterung verzichtet. Dies ist auch der Sinn des Konzeptes: Man soll bei Bedarf über die maximal mögliche Anzahl von Bildschirmen verfügen können.

Sie können die Wahl zwischen Erweiterung und maximaler Bildschirmanzahl auch während des Programmlaufs treffen. Ein einsichtiges Beispiel dazu ist der Bitmap-Compander, mit dem Sie Bilder in komprimierter Form auf Diskette speichern können. Dieses Feature bietet sich für die vielen Bitmaps, die für einen Bewegungsablauf mittels Walk nötig sein können, geradezu an. Wenn Sie alle sieben Bildschirme (sechs plus Leinwand) für einen Trickfilm benötigen, brauchen Sie dennoch nicht auf die Vorteile des Bitmap-Companders zu verzichten. Haben Sie die sechs Bilder erstellt, laden Sie die Erweiterung in den Leinwand-Bildschirmspeicher und speichern anschließend die Bitmaps mittels Compander ab. Um sie erneut zu laden, muß sich die Erweiterung ebenfalls wieder im Leinwand-Bildschirm befinden. Vor dem Start des Trickfilms entfernen Sie dann die Erweiterung wieder.

Ich finde es deshalb durchaus sinnvoll, die Erweiterung in den letzten Bildschirm zu laden, also Nummer 6 im Farb- und Menü-Betrieb und Nummer 7 bei Schwarzweiß. Dadurch teilt der fehlende Bildschirm nicht die Reihe der noch verfügbaren, sondern man hat die fünf bzw. sechs noch zur Verfügung stehenden Bildschirme in einem Stück.

Apropos Menü: Wenn Sie im vorliegenden Buch bis an diese Stelle vorgedrungen sind, kennen Sie sich wahrscheinlich mit der Bedienung von Hi-Eddi plus schon so gut aus, daß Sie auf das Menü verzichten können. Außerdem ist es sowieso nicht vorgesehen, die Befehle der Erweiterung vom Menü aus aufzurufen.

Es gibt noch einen anderen Grund, die Erweiterung im Schwarzweiß-Betrieb in Speicher 7 zu laden: Der bereits erwähnte Scroller verwaltet die Bildschirme 1 bis 6 wie ein großes, zusammenhängendes Bild mit 640*600 Punkten. Damit er überhaupt aktiviert wird, muß deshalb die Erweiterung in Speicher 7 geladen werden. Daraus folgt auch, daß der Scroller nur im Schwarzweiß-Betrieb funktioniert, da es im Farb-Betrieb keinen Speicher 7 gibt. Da der Scroller jedoch nur bei Zeichnungen, die ausgedruckt werden sollen, sinnvoll einzusetzen ist, entsteht daraus bestimmt kein Nachteil.

Um alle Möglichkeiten der Erweiterung, insbesondere natürlich den Scroller, testen zu können, empfehle ich Ihnen, Hi-Eddi plus im Schwarzweiß-Betrieb zu starten und die Erweiterung EXT in den Speicher 7 zu laden.

4.3 Das Extended Disk System

Die erweiterten Load- und Save-Befehle stellen neben den drei bzw. vier Auswahlmöglichkeiten der Speicherung auf Diskette - nämlich Grafikbild, Farbbild, Zeichensatz und Sprite - sechs weitere Formate zur Verfügung.

Gehen wir diese zusätzlichen Formate, die bei Aufruf des Load- oder Save-Befehls zusammen mit den bisherigen Formaten zur Auswahl gestellt werden, der Reihe nach durch:

Bitmap-Compander:

In den meisten Bildern, insbesondere in den technischen Plänen, Diagrammen, PAPs oder Struktogrammen, gibt es große, freie Flächen. Im Speicher werden diese Flächen als lange Folge von Nullen repräsentiert - und auch als solche auf Diskette abgespeichert. Das ist jedoch reine Platzverschwendung! Es ist durchaus möglich, wesentlich mehr Bilder auf eine Diskette zu bekommen, wenn man diese Redundanz - so das Fachwort - verändert.

Der Bitmap-Compander tut dies. Er komprimiert die Bilder bei der Speicherung auf Diskette und expandiert sie beim Laden wieder. Daher auch der Name Compander, in dem die Worte Compressor und Expander enthalten sind.

Je nachdem, wie viele freie Flächen in einem Bild vorhanden sind, ergibt sich dadurch eine Reduzierung des Platzbedarfs auf der Diskette um bis zu 75%. Die Abbildungen in diesem Buch brauchen durchschnittlich etwa 10 bis 14 Blocks statt der normalerweise nötigen 33. Man bekommt also zwei- bis dreimal so viele Bilder auf eine Diskette, außerdem verringert sich die Zeit zum Laden und Speichern um diesen Faktor!

Ebenso wie beim Grafikbild-Format wird mittels Bitmap-Compander nur die Bitmap selbst ohne Farbinformation auf Diskette abgespeichert.

Genaueres über den Bitmap-Compander, über die angewandte Methode und darüber, wie Sie Bilder in diesem Format auch in eigene Programme laden können, lesen Sie im 64'er, Ausgabe 8/85.

Paint Magic:

Bei der Beschreibung der Load- und Save-Befehle habe ich Ihnen bereits erklärt, welche Probleme beim Austausch von Bildern mit Programmen, die im Multicolor-Modus arbeiten, auftreten. Dennoch kann dieser Austausch sinnvoll sein. Die Bilder des Malprogramms Paint Magic zum Beispiel lassen sich mit Hi-Eddi plus ausdrucken. Allerdings ist das Laden dieser Bilder wegen des anderen Formats auf Diskette etwas umständlich. Mit den erweiterten Load-Load- und Save-Routinen ist das anders!

Sie können damit Paint Magic-Bilder sofort einladen und sogar Bilder in diesem Format wieder abspeichern, die dann mit Paint Magic geladen werden können. Die Problematik mit dem Multicolor-Modus bleibt allerdings bestehen, d.h. beim Laden der Paint Magic-Bilder geht deren Farbinformation verloren. Sie müssen daher das Bild nach dem Laden erst in passablen Farben darstellen. Vor einem Ausdruck müssen Sie es eventuell auch noch mit **I** invertieren, damit Sie kein Negativ erhalten.

Wenn Sie Bilder vom Hi-Eddi aus im Paint-Magic-Format abspeichern, bleibt zwar die Farbe voll erhalten, die Auflösung geht jedoch verloren. In Paint Magic geladen, werden die Farben also stimmen, die Details werden jedoch eckiger, Kreise werden wackliger und kleine Schrift wird fast unleserlich.

Ebenso wie die original Paint-Magic-Bilder sind natürlich auch die vom Extendend Disk System erzeugten Bilder dieses Formats mit einem BASIC-Lader versehen. Sie können diese Bilder ohne Hilfe eines anderen Programms anschauen, indem Sie sie wie ein BASIC-Programm mit **LOAD "Name",8** laden und mit **RUN** starten. Mit **STOP/RESTORE** kommen Sie wieder aus dem Bild heraus.

Koala Painter:

Mit dem Extended Disk System ist es auch möglich, Bilder des Koala Painters zu laden und sogar in dessen Format wieder zu speichern. Dazu müssen allerdings einige Hindernisse überwunden werden, die die Programmierer des Koala Painters einem Austausch der Bilder in den Weg gelegt haben. Das fängt schon an mit dem Filenamen, der mit einem Steuerzeichen beginnt, das man ohne besondere Tricks gar nicht eingeben kann. Der nachfolgende Teil des Namens muß exakt der Form **PIC X NAME** entsprechen.

Das X steht dabei für einen beliebigen Buchstaben und NAME für den eigentlichen Namen des Bildes, der jedoch nur noch acht Zeichen lang sein darf. Ist er kürzer, so muß er mit Blanks auf die erforderliche Länge aufgefüllt werden.

Ich hätte natürlich das Extended Disk System so auslegen können, daß es auf die Einhaltung dieser Form automatisch achtet. Das wollte ich jedoch nicht, denn es könnte ja sein, daß ein anderes Programm dasselbe Format zur Abspeicherung seiner Bilder benutzt. Ein Laden oder Speichern dieser Bilder soll dann nicht an einem *File not found-Error* scheitern, weil dessen Filename nicht den strengen Vorschriften des Koala Painters entspricht. Ich habe deshalb einen Kompromiß gewählt: Beim Laden eines Bildes im Koala Painter-Format wird der eingegebene Filename nicht geändert. Um ein solches Bild zu laden, gibt man den Namen deshalb in der Form

?PIC X NAME*

ein. Das Fragezeichen dient dabei als Joker, um das Steuerzeichen zu umgehen und der Stern am Ende steht für die fehlenden Blanks. Beim Speichern wird der Filename nur dann vom Extended Disk System geändert, wenn er mit einem Fragezeichen beginnt. In diesem Fall wird dann das Fragezeichen durch das oben genannte Steuerzeichen ersetzt und der Name wird auf die erforderliche Länge mit Blanks aufgefüllt. Sie brauchen den Namen also ebenfalls nur in der obigen Form einzugeben, allerdings ohne das Sternchen am Ende. Dieses ist nämlich in einem Filenamen beim Speichern nicht erlaubt.

Das nächste Problem am Koala-Painter-Format ist die schon in Kapitel 1 angesprochene ungünstige Bildschirmorganisation. Das Extended Disk System wandelt deshalb das Bild beim Laden in ein brauchbares Format um. Ich bin dabei, wie schon bei Paint Magic, davon ausgegangen, daß man Bilder dieser Programme wohl nur laden wird, um sie mit Hi-Eddi plus auszudrucken. Deshalb bleibt auch hier die Farbinformation auf der Strekke. Da jedoch zur Umwandlung auch das Video- und Farb-RAM von der Diskette gelesen werden muß, steht das Koala-Painter-Format nur im Farb-Betrieb zur Verfügung. Damit die Umwandlung auch einwandfrei funktioniert, muß man bereits beim Einfärben des Bildes im Koala Painter darauf achten, möglichst kontraststarke Farben zu verwenden. Am besten können Sie dies testen, indem Sie an Ihrem Sichtgerät die Farbe wegdrehen. Erst wenn dann noch ein deutlicher Kontrast zwischen den einzelnen Flächen des Bildes besteht, wird die Umwandlung ein befriedigendes Ergebnis liefern.

Auch das Speichern in diesem Format ist wieder möglich, wobei das bereits bei Paint Magic Gesagte gilt.

Blazing Paddles:

Obwohl Blazing Paddles fast dieselbe Bildorganisation wie der Koala Painter benutzt und Bitmap, Video- und Farb-RAM auch in derselben Reihenfolge abspeichert, sind die beiden Formate dennoch nicht kompatibel. Denn während Blazing Paddles - ebenso wie Paint Magic, Doodle und Hi-Eddi plus - jeweils in vollen Kilobytes abspeichert, spart sich der Koala Painter ein paar Bytes, wodurch die Farbinformationen verschoben sind. Für dieses Format gilt weitgehend dasselbe wie beim Koala Painter, insbesondere die Tatsache, daß es nur im Farb-Betrieb zur Verfügung steht und daß man bereits vor dem Laden auf ausreichenden Kontrast der Farben achten sollte. Keine Komplikationen gibt es mit dem Filenamen; er muß lediglich mit **PI.** beginnen. Bezüglich des Speicherns in diesem Format gilt dasselbe wie bei Paint Magic.

Doodle:

Neben Hi-Eddi plus ist Doodle das einzige mir bekannte Zeichenprogramm (abgesehen von einigen weniger brauchbaren Exemplaren dieser Gattung), das im High-Resolution-Modus arbeitet. Das Problem mit dem Multicolor-Modus gibt es deshalb nicht, es müßte also ein problemloser Austausch von Bildern möglich sein. Jedoch besitzen Doodle-Bilder ein anderes Format: Doodle speichert zuerst die Farbinformation, danach die Bitmap ab. Hi-Eddi plus macht es umgekehrt (logischer, wie ich meine). Das Extended Disk System ermöglicht es nun, Bilder auch in diesem *verdrehten* Format abzuspeichern und zu laden.

Wenn Sie ein Doodle-Bild im Schwarzweiß-Betrieb laden, so wird nur die Bitmap geladen. Im Farb-Betrieb wird dagegen auch die Farbinformation mit von der Diskette geholt.

Natürlich können Sie auch hier Bilder im Doodle-Format abspeichern, die dann ohne Komplikationen von diesem Programm geladen werden können. Im Gegensatz zu den Multicolorbildern der vorigen drei Programme ist hier, zumindest im Farb-Betrieb, ein wirklich vollständiger Austausch der Bilder möglich. Es geht weder Farbe noch Auflösung verloren. Sie müssen lediglich darauf achten, daß der Filename mit **DD** beginnt, sonst akzeptiert ihn Doodle nicht.

Adventure Grafik System:

Dieses letzte Format steht nur in der Farb-Betriebsart zur Verfügung.

Nach der Veröffentlichung des Hi-Eddi im 64'er fragten viele Leser, wie man mit Hi-Eddi erstellte Bilder in selbstgeschriebene Grafik-Adventures laden könne. Der im 64'er, Ausgabe 3/85 vorgestellte PIC-LOADER war dazu ungeeignet, da er keine Farbbilder laden konnte. Der PIC-LOADER auf der Diskette, die diesem Buch beiliegt, ist da schon besser geeignet, da er auch Farbbilder laden kann. Aber ideal ist er auch nicht, da jedesmal die ganze Bitmap geladen wird, was nicht nur zeitaufwendig sondern auch unnötig ist. Fast alle Grafik-Adventures nützen nämlich nur die obere Bildschirmhälfte für die Grafik aus, die untere wird für Text benötigt.

Aus diesem Grunde entwickelte ich das Grafik-Adventure-System, das sich unter dem Namen **AGS** auf der Diskette befindet. Es zeichnet sich dadurch aus, daß nur die halbe Bitmap auf Diskette geschrieben wird, allerdings in Farbe. Außerdem übernimmt das AGS die Aufteilung des Bildschirms in einen Grafik- und einen Textteil. Als Besonderheit können maximal fünf Bilder gleichzeitig unter den ROMs und den I/O-Bausteinen versteckt werden, zwischen denen blitzschnell umgeschaltet werden kann. Dadurch muß nicht bei jeder Bewegung des Spielers im Adventure ein Bild nachgeladen werden, sondern es läßt sich ein Teil des Adventures auf einmal laden, in dem dann ohne lästige Nachladepausen gespielt werden kann. Erst wenn dieser Teil verlassen wird, muß wieder nachgeladen werden.

Zur komfortablen Erstellung der Bilder für ein eigenes Adventure bietet sich natürlich der Hi-Eddi plus an. Das Extended Disk System erlaubt es deshalb, Bilder in diesem Format zu speichern, die dann vom Adventure-Grafik-System geladen werden können.

Es wird dazu die obere Hälfte des aktuellen Bildschirms - genauer gesagt: die oberen zwölf Zeilen entsprechend 96 Punkten, was nicht ganz der Hälfte entspricht- auf Diskette geschrieben. Die Begrenzung auf zwölf Zeilen war nötig, damit im AGS ohne BASIC-Speicher-Platzverlust fünf Bilder gleichzeitig im Rechner Platz haben. Außerdem wird die halbe Bitmap noch durch den Bitmap-Compander geschickt. Bei Bildern mit freien Flächen ergibt sich dadurch eine Speicherplatzeinsparung auf der Diskette.

Um AGS-Bilder in eigene Grafikadventures zu laden, muß sich zusammen mit dem eigentlichen Adventure auch das AGS im Rechner befinden. Dies erreicht man am besten mit folgender Zeile am Anfang des Adventures:

```
1 IF A=0 THEN A=1: LOAD"AGS",8,1
```

Natürlich müssen Sie dazu das AGS auf Ihre Adventure-Diskette kopieren. Programmiert wird das AGS durch die folgenden SYS-Befehle:

Bild laden: SYS 49251, Nr,8,2,"Name"
Bild anwählen: SYS 49373,Nr
Grafik einschalten: SYS 49493
Grafik ausschalten: SYS 49529

Nr ist die Nummer des Bildes (1 bis 5), **Name** kann auch eine Stringvariable sein. Beim Laden wird die Grafik vorübergehend ausgeschaltet, da die zeitkritischen Busroutinen mit dem Rasterzeilen-Interrupt kollidieren würden. Der Teil des Textbildschirmes unter der Grafik bleibt unangetastet, so daß er für zusätzliche Texte, wie z.B. beim *Hobbit*, verwendet werden kann.

Makro:

Dies ist kein Format zum Speichern von Bildern, sondern damit können Sie die selbst definierbaren Makrobefehle, die in einem späteren Kapitel beschrieben werden, auf Diskette ablegen.

Wie schon bei den normalen Load- und Save-Befehlen gilt natürlich auch beim Extended Disk System, daß Sie Bilder nur mit dem Format, mit dem Sie sie gespeichert haben, auch wieder laden dürfen. Dazu empfiehlt es sich, die Filenamen auf der Diskette entsprechend zu markieren, zum Beispiel durch die Anhängsel **.BMC**, **.PM**, **.AGS** und **.MAC**. Die Doodle-Bilder werden bereits automatisch mit einem vorangestellten **DD** markiert.

Natürlich können Sie, wenn es Ihnen Spaß macht, auch probieren, was passiert, wenn Sie Bilder mit falschen Formaten laden. Meistens wird dabei Chaos auf dem Bildschirm entstehen. Im schlimmsten Fall, wie in Kapitel 1 bereits angedeutet, kann es zum Absturz kommen. In Kapitel 1 haben Sie jedoch auch erfahren, daß sinnvolle Ergebnisse auftreten können. So könnte es zum Beispiel noch weitere Grafikprogramme geben, für die eines der vielen Formate paßt.

Auf alle Fälle sollten Sie vor dem Experimentieren mit fremden Files die sich eventuell im Speicher befindlichen Bilder auf Diskette speichern, denn Sie wissen vorher nie, was passiert.

4.4 Der Rotate-Befehl

Dieser Befehl hätte eigentlich schon in den Hi-Eddi plus selbst gehört. Dort hatte er jedoch keinen Platz mehr.

Wie bereits besprochen, gibt es im Sprite-Editor Befehle zum Spiegeln (Mirror), zum Drehen um 180 Grad (Turn) und zum Drehen um 90 Grad (Rotate). Im Grafik-Editor standen bisher nur die Befehle Mirror und Turn zur Verfügung.

Die Erweiterung behebt diesen Mangel nun und stellt den Rotate-Befehl auch für den Grafik-Editor zur Verfügung. Aufgerufen wird dieser Befehl analog zu Mirror (**SHIFT M**) und Turn (**SHIFT T**) mit **SHIFT R**.

Und was bewirkt der Befehl? Er dreht einen quadratischen Bereich um 90 Grad im Uhrzeigersinn. Quadratisch deshalb, da sonst ein Teil des Bereichs verlorengehen würde, wie das ja im Sprite-Editor der Fall ist.

Wird der Befehl ohne vorheriges Move eingegeben, so werden die linken 25 Spalten des Bildschirms rotiert. Gibt man den Befehl dagegen unmittelbar nach einem Move-Vorgang ein, so bezieht er sich, ebenso wie Mirror und Turn, auf den gerade verschobenen Bereich. Da dieser Bereich jedoch in der Regel rechteckig sein wird, beschränkt sich der Rotate-Befehl auf den linken quadratischen Teil in einem liegenden bzw. auf den oberen quadratischen Teil in einem stehenden Rechteck.

Dieser Befehl ist insbesondere auch in Verbindung mit dem Ausdruck von Bildern sinnvoll, so zum Beispiel, wenn Sie ein Bild im Querformat ausdrucken wollen. Um einen ganzen Bildschirm zu rotieren, drehen Sie zunächst die linken 25 Spalten. Die verbleibenden 15 Spalten verschieben Sie mittels Move an den linken Rand des darunterliegenden Bildschirms. Dort drehen Sie nun wieder die 25 linken Spalten und erreichen bei einem aufeinanderfolgenden Ausdruck dieser beiden Bildschirme den Ausdruck im Querformat.

4.5 Die Koordinatenanzeige

Mit der programmierbaren Schrittweite ist es leicht, Bilder in einem bestimmten Raster anzulegen oder sich wiederholende Objekte - Punkte im Draw-Modus ebenso wie komplexe Gebilde im Append-Modus - in gleichmäßigen Abständen zu zeichnen. Das koordinatengenaue Positionieren des Cursors ist damit zwar ebenfalls möglich, aber nicht gerade sehr komfortabel. Insbesondere für Konstruktionszeichnungen, die man maßstabsgenau zeichnen möchte, bietet deshalb die Erweiterung eine bessere Möglichkeit an: eine in den Bildschirm einblendbare Koordinatenanzeige, die ständig die aktuelle Cursorposition numerisch anzeigt.

Eingeschaltet wird diese Anzeige durch Eingabe von **K**. Nochmaliges Eintippen dieses Buchstabens schaltet die Anzeige wieder ab.

Die Anzeige befindet sich normalerweise in der rechten unteren Bildschirmecke. Kommt man ihr jedoch mit dem Cursor zu nahe, dann weicht sie nach rechts oben aus. Es passiert also nie, daß sie den gerade bearbeiteten Teil des Bildschirms überdeckt. Erst wenn der Cursor weit genug von der unteren Bildschirmecke entfernt ist, kehrt die Anzeige wieder an ihren angestammten Platz zurück.

Es werden zwei Zahlen angezeigt. Die obere der beiden Zahlen ist die X-Koordinate, die sich im Bereich von 0 bis 319 bewegt. Die untere ist die Y-Koordinate, sie kann Werte von 0 bis 199 annehmen. Der Nullpunkt liegt dabei links oben in der HOME-Position, was dem bei den meisten Grafikerweiterungen üblichen Koordinatensystem entspricht.

Beim kreuzförmigen Cursor ist die angezeigte Position natürlich der Mittelpunkt des Kreuzes, also der Punkt, an dem gezeichnet wird. Für den kleinen und großen Rahmen (Text bzw. Append, Stamp, Get und Erase) habe ich die Punkte so gewählt, daß beim Umschalten der Modi und somit der verschiedenen Cursor die Koordinatenanzeige unverändert bleibt. Es ergeben sich somit Punkte, die in der Mitte der Rahmen liegen. Wenn Sie den Cursor in die HOME-Position fahren, werden diese Koordinatenwerte angezeigt. Für den großen Rahmen sind dies die Werte 10,10 und für den kleinen die Werte 3,3.

In Zusammenhang mit dem Scroller, der im nächsten Kapitel beschrieben wird, ist es auch möglich, eine Koordinatenanzeige zu erreichen, deren Nullpunkt in der HOME-Position des Bildschirms Nummer 1 liegt. Einge-

schaltet wird diese Form der Anzeige, die nur bei aktivem Scroller funktioniert, d.h. wenn die Erweiterung in Bildschirm 7 liegt, durch Eingabe von **SHIFT K** anstatt nur **K**.

In Bildschirm Nummer 6, dem rechten unteren des Gesamtbildes mit 640*600 Punkten, nehmen die Koordinaten dann Werte von 320 bis 639 bzw. 400 bis 599 an. Selbstverständlich werden auch in allen Zwischenpositionen, die der Scroller ansteuern kann, die Koordinatenwerte absolut zur HOME-Position des Bildschirms 1 angezeigt. Damit sind auch größere Konstruktionszeichnungen leicht zu realisieren.

Die Anzeige selbst wird natürlich nicht tatsächlich in den Bildschirm eingebaut, sondern ist als Sprite realisiert. Sie wird also bei Verknüpfungen nicht berücksichtigt und, ebenso wie der Cursor, nicht mit ausgedruckt.

Ihre Farbe entspricht der des Bildschirmrahmens. Man sollte daher für den Rahmen eine Farbe wählen, die zum Hintergrund und zum Vordergrund einen guten Kontrast gibt.

Sie werden jedoch feststellen, daß sich die Farbe der Anzeige nicht zugleich mit der des Rahmens ändert. Wenn Sie die Hintergrund- oder Vordergrundfarbe ändern, kann es auch passieren, daß in der Anzeige Unsinn erscheint. Und beim setzen eines Merk-Cursors bei Line, Rec, Circle und Move verschwindet die Anzeige sogar. Der Grund dafür ist, daß die Anzeige - sowohl ihr Wert als auch ihre Farbe - nur bei einer Cursorbewegung oder einem Moduswechsel aktualisiert wird. Da Sie jedoch die Anzeige nur benötigen, wenn Sie den Cursor plazieren, also bewegen, ist sie garantiert zur Stelle, wenn sie gebraucht wird.

Und bevor wir nun zum interessantesten Teil der Erweiterung, dem Scroller kommen, möchte ich Ihnen noch ein kleines Spiel präsentieren. In Kapitel 2 haben wir uns den Paint-Befehl näher angesehen und ich habe Ihnen versprochen, daß wir in Kapitel 4 dem *Wurm* in sein Notizbuch sehen werden. Das werden wir jetzt tun. Laden Sie dazu wieder das Bild mit der Treppe und starten Sie das Spiel, diesmal aber bei eingeschalteter Koordinatenanzeige. Sie werden staunen, was in der Anzeige alles geschieht! Dies ist das Notizbuch des Wurms und zugleich auch das Notizbuch des Prozessors im C64. In der Fachsprache heißt es *Stack*, zu deutsch Stapel oder Kellerspeicher. Der Profi wird sich vielleicht fragen, warum ich ein Sprite in den Stack lege. Nun, der VIC kann nur 16 kByte adressieren, und die sind bereits voll. Mir blieb also nichts anderes übrig, als in den Stack auszuweichen.

4.6 Der Scroller

In Kapitel 2 habe ich Ihnen zwei Methoden gezeigt, wie man mehrteilige Bilder anfertigt: die *Anstückel-Methode*, bei der geschickte Cursorplazierungen das Zusammenpassen der Bildschirme gewährleisten sollen, sowie die *Aus-eins-mach-zwei-Methode*, die darin besteht, die Nahtstelle zwischen zwei Bildern zunächst zusammenhängend in einem Bildschirm zu entwerfen. Beide Methoden sind jedoch recht umständlich und für größere Zeichnungen mit viel Inhalt schlecht zu gebrauchen.

Der Scroller schafft hier Abhilfe. Er verwaltet die Bildschirme 1 bis 6 in der in Abb. 4.1 gezeigten Anordnung wie ein großes, 640*600 Punkte umfassendes Bild. Der sichtbare Bildschirm kann nun wie ein Fenster beliebig in 8-Punkte-Schritten über das ganze Riesenbild gescrollt werden.

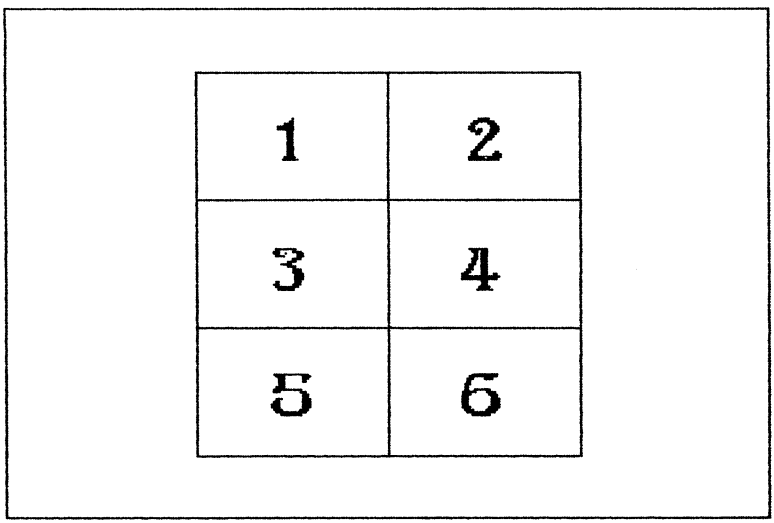


Bild 4.1: Die Anordnung der einzelnen Bildschirme beim Scroller

Eingeschaltet wird der Scroller durch Y. (Eine sinnvollere Taste ist nicht mehr frei, da bereits fast alle belegt sind.) Danach ist kein Cursor mehr zu sehen, denn jetzt ist der Bildschirmrahmen gewissermaßen der *Cursor*. Mittels Joystick können Sie nun diesen *Cursor* wie ein Fenster über das ganze

Bild fahren. Um zum Beispiel den oberen Rand zu sehen, müssen Sie den Joystick nach oben drücken. Der Bildschirminhalt wandert dann natürlich nach unten. Ich finde jedoch die Lösung, den Bildschirmrahmen zu steuern, logischer.

Falls Sie das Pad benutzen, müssen Sie den Stift sehr langsam über die Zeichenfläche bewegen, sonst kommt der Scroller nicht mit. Da der Scroller jedoch vor allem für technische Anwendungen interessant ist, wird man hier sowieso meist mit dem Joystick arbeiten.

Abgeschaltet wird der Scroll-Modus durch Knopfdruck oder eine beliebige Tastatureingabe. Der Scroll-Befehl ist somit kein echter Modus-Befehl, denn der aktuelle Modus wird nicht verändert. Man tippt lediglich ein Y, um den Bildschirm schnell zu verschieben, und kann dann sofort, ohne neue Befehlseingabe, im aktuellen Modus weiterzeichnen.

Wie bereits erwähnt, funktioniert der Scroller nur, wenn die Erweiterung in Speicher 7 liegt. Damit ist auch klar, daß der Scroller nur im Schwarzweiß-Betrieb arbeitet. Dies ist jedoch keine Einschränkung: Da man sich ein mehrteiliges Bild nicht vollständig auf dem Bildschirm ansehen kann, ist das Erstellen eines solchen Bildes wohl nur sinnvoll, wenn es ausgedruckt werden soll. Der Ausdruck wiederum ist nur in Schwarzweiß möglich.

Es gilt nach wie vor der Grundsatz, daß nur der sichtbare Bildausschnitt manipuliert werden kann. So ist es zum Beispiel nicht möglich, eine Diagonale quer über das ganze Riesenbild in einem Stück zu zeichnen.

Um dies zu ermöglichen, hätte die Speicherorganisation des ganzen Programms grundlegend geändert werden müssen. Der Scroller bringt die Speicherverwaltung des Hi-Eddi plus sowieso schon gehörig durcheinander. Sie merken dies, wenn Sie einen Bildschirm durch Eingabe seiner Nummer aufrufen, was nach wie vor möglich ist. Dann muß nämlich erst wieder Ordnung in das Durcheinander des Speichers gebracht werden, was dadurch geschieht, daß nach der Eingabe einer Nummer der Bildschirm von selbst in die nächstliegende Normposition scrollt. Erst jetzt kann Hi-Eddi plus auf den angewählten Bildschirm umschalten. Eine Normposition liegt vor, wenn sich der sichtbare Bildschirm mit einem der sechs Einzelbildschirme deckt. Wenn dagegen Ausschnitte von zwei oder vier Bildschirmen sichtbar sind, bezeichne ich dies als Zwischenposition.

Die komplizierte Speicherverwaltung macht sich aber auch bei allen Befehlen bemerkbar, die nicht nur auf den sichtbaren, sondern auch auf die nicht sichtbaren Speicher zugreifen. Dazu gehören in erster Linie die Be-

fehle zum Kopieren von Bildschirmen (Commodore-Taste und Nummer), zum Verknüpfen ganzer Bildschirme (**U**, **O**, **X** gefolgt von einer Nummer) und der Befehl Print zum Ausdrucken von Bildschirmen. Alle diese Befehle funktionieren nur, wenn der Scroller in einer Normposition steht, also einer der sechs Bildschirme voll sichtbar ist. Steht der Scroller dagegen in einer Zwischenposition, so wird die Eingabe der oben genannten Befehle ignoriert.

Auch der Move-Befehl ist davon betroffen. Innerhalb eines Bildschirms kann beliebig verschoben und verknüpft werden, ganz egal, ob sich der Scroller in einer Normposition befindet oder nicht. Soll dagegen von einem Bildschirm in einen anderen verschoben werden, so müssen Quell- und Zielbereich in einem der sechs per Nummer anwählbaren Bildschirme liegen. Sie müssen diese dann auch über ihre Nummer anwählen, denn der Scroller löscht, wenn er eingeschaltet wird, eine eventuell bestehende Move-Markierung. Damit soll das Verschieben von einer zur anderen Zwischenposition verhindert werden, da dabei Chaos entstehen würde.

Eine durch den Scroller gelöschte Markierung kann dagegen durch den Befehl Move Restore (Pfeil-Nach-oben-Taste) wieder geholt werden. Auch dabei ist jedoch Vorsicht geboten. Denn durch diesen Befehl wird nicht ein markierter Bereich, sondern nur die Markierung selbst zurückgeholt. Dieser Befehl bemerkt nicht, wenn der Bildschirm mittels Scroller verschoben wurde. Ein Beispiel: Sie markieren das linke untere Viertel des Bildschirms. Nun verschieben Sie mittels Scroller den Bildschirmrahmen um einige Zeilen nach unten, so daß der markierte Bereich eigentlich nach oben wandern müßte. Wenn Sie nun aber den Befehl Move Restore eingeben, so wird doch wieder das linke untere Viertel markiert!

Und noch einen Befehl gibt es, der auf unsichtbare Bildschirmspeicher zugreift: der Text-Modus, wenn ein Zeichensatz im RAM verwendet wird. Ich empfehle Ihnen, diesen Zeichensatz dann in einen der beiden oberen Speicher, also 1 oder 2 zu legen, da er dort nur einen schmalen Rand beansprucht und somit nicht zuviel Raum vom Gesamtbild in Anspruch nimmt. Genauer gesagt, die nutzbare Fläche mißt noch 640*544 Punkte. Vor dem Ausdruck oder dem Abspeichern löschen Sie den Zeichensatz (Umschalten auf den Commodore-Zeichensatz mittels **C= Z** nicht vergessen).

Damit nun der Zugriff auf diesen im RAM befindlichen Zeichensatz trotz der chaotischen Speicherverhältnisse einwandfrei funktioniert, müssen Sie folgendes beachten: Wenn Sie Text eingeben, darf der Scroller nicht in einer Zwischenposition stehen, in der die Bildschirme 1 und/oder 2 teilweise zu sehen sind. Sie können also nur dann Text eintippen, wenn entweder nur die Bildschirme 3 bis 6 zu sehen sind, wobei es egal ist, ob diese ganz oder teilweise sichtbar sind, oder aber wenn Bildschirm 1 oder 2 komplett sichtbar ist, also in Normposition liegt.

Das war vielleicht etwas kompliziert, aber keine Angst, wenn Ihnen dabei ein Fehler unterläuft, kann nichts kaputtgehen. Sie werden dann lediglich statt der gewünschten Buchstaben oder Zeichen irgendwelche Hieroglyphen drucken.

Nach so vielen Einschränkungen möchte ich Ihnen nun auch noch sagen, welche Befehle in Verbindung mit dem Scroller funktionieren: alle, außer den oben genannten! Denn all diese Befehle beziehen sich nur auf den sichtbaren Bildschirm, und dabei ist es egal, ob dieser sichtbare Bildschirm in Normlage liegt oder nicht. So können Sie zum Beispiel sogar einen in Zwischenlage befindlichen Bildschirm mittels Save auf Diskette speichern und wieder laden, sofern Sie dabei nicht selbst den Überblick verlieren.

4.7 Makros

Nun zum letzten Befehl der Erweiterung - genauer gesagt sind es eigentlich drei Befehle. Sie sind zwar etwas kompliziert, wenn Sie sie jedoch verstanden haben, können Sie sich damit eine Menge Tipparbeit ersparen und sogar kleine Grafikprogramme schreiben. Im Grunde sind Makros nämlich eine Mini-Programmiersprache. Was sich damit anfangen läßt, zeigen Ihnen die Abb. 4.4 bis 4.8.

Zunächst wenden wir uns den Befehlen zu, die Sie kennen müssen, um Makros anwenden zu können:

=	Makro aufrufen
C= =	Makro definieren
SHIFT =	Makro auswählen und anzeigen

Nach den Funktionstasten und der Taste **D** ist jetzt auch das Gleichheitszeichen dreifach belegt.

Doch bevor wir fortfahren, wollen wir die Frage klären: Was ist ein Makro? Ein Makro ist ein selbstdefinierter Befehl, der seinerseits aus einer Reihe von Befehlen besteht. Dazu ein Beispiel:

In Kapitel 2 habe ich Ihnen gezeigt, wie man mit Hilfe der programmierbaren Schrittweite schnell ein Gitterraster zeichnen kann. Wir werden das hier noch einmal wiederholen, allerdings einen anderen Lösungsweg dabei verwenden.

Programmieren Sie zunächst auf der Funktionstaste F5 die horizontale Schrittweite 16 und die vertikale Schrittweite 200. Nun fahren Sie den Cursor in die HOME-Position und tippen folgende Sequenz ein:

'L' - 'C= F5' - right (Cursortasten!) - RETURN - down - RETURN - F7 - right - RETURN - down - RETURN - F7 - right - RETURN - down - RETURN - F7 - .....

Wenn Sie auf diese Weise den ganzen Bildschirm mit senkrechten Linien überzogen haben - und sich dabei wahrscheinlich einige Male vertippt haben - werden Sie mir wohl zustimmen, daß das ein mühsames Vorgehen ist.

Nun wollen wir das ganze per Makro erledigen. Löschen Sie dazu den Bildschirm, fahren Sie den Cursor wieder in die HOME-Position und geben Sie folgendes ein:

'C= =' (also Commodore-Taste und Gleichheitszeichen) - 'L' - 'C= F5' - right - RETURN - down - RETURN - F7 - '='

Nun tippen Sie noch einige Male = und Sie werden sehen, daß bei jedem Tastendruck eine Linie entsteht. So haben Sie ohne viel Aufwand sehr schnell den ganzen Bildschirm mit Linien gefüllt.

Nun schauen wir uns an, wie wir zu diesem Ergebnis gekommen sind. Der wichtigste Befehl war der erste, also das **C= =**. Damit wird die Definition eines Makros eingeleitet. Das Programm befindet sich nach Eingabe dieses Befehls im Speicherbetrieb, in dem es sich alle Eingaben, die wir tätigen, merkt. Angezeigt wird dieser Speicherbetrieb durch die veränderten Blinkfarben des Cursors (blau-gelb statt schwarzweiß). Außerdem sind im Speicherbetrieb nicht alle Befehle möglich, sondern nur ein Teil der Hi-Eddi plus-Befehle. So wäre es zum Beispiel recht sinnlos, einen Load-Befehl im Makro zu definieren. Alle Befehle, die weitere Eingaben erfordern, wie Disk- und Druckerbefehle, **H**, **V**, Zoom und Text sind in Makros nicht möglich. Auch die Cursorsteuerung ist nur mittels Cursortasten möglich, die Joystick- oder Pad-Steuerung ist im Speicherbetrieb ebenso wirkungslos wie der Feuerknopf. Nur die folgenden Befehle sind innerhalb eines Makros zulässig:

1. Cursor- und Funktionstasten, HOME, RETURN
2. Die Zeichenmodi D,L,R,C,J,P
3. Move, Restore Move, U,O,X, Invert, Mirror, Rotate
4. Die Sprite-Befehle A,S,G,E
5. F,B

Abgeschlossen wird die Makrodefiniton, also der Speicherbetrieb, durch Eingabe von =.

Nochmalige Eingabe von = ruft dann das eben definierte Makro auf, worauf alle zuvor in Speicherbetrieb eingegebenen Befehle ausgeführt werden.

Natürlich hätten wir die Befehle **L** und **C= F5** nicht mit ins Makro eintippen müssen, wenn sichergestellt wäre, daß beim Aufrufen des Makros, also bei Eingabe von =, der Line-Modus aktiv und die Funktionstaste F5 angewählt ist. Da man diese Voreinstellungen aber leicht vergißt, empfehle ich, grundsätzlich auch die Modus- und Funktionstastenanwahl mit ins Makro

einzugeben. Ein Makro darf aus bis zu 255 Befehlseingaben bestehen. Wenn man nicht gerade in Einer-Schritten quer über den Bildschirm laufen möchte, dann müßte dies für alle Fälle ausreichen, auch dazu, um einige anscheinend überflüssige Befehle mit abzuspeichern.

Falls Sie einmal vergessen haben, welche Befehle Sie in einem Makro eingetippt haben, können Sie sich das Makro auflisten lassen. Geben Sie dazu **SHIFT =** ein. Danach erscheint zunächst die Frage nach der Nummer des Makros. Die Extension stellt uns nämlich vier Makros gleichzeitig zur Verfügung, damit wir nicht bei jeder neuen Makro-Definiton das alte verlieren. Wir müssen nun also eine Nummer von 1 bis 4 eingeben. Falls Sie jedoch nicht wissen, welches Makro gerade angewählt ist - so wie jetzt beispielsweise - dann drücken Sie eine beliebige andere Taste. Die Extension zeigt Ihnen daraufhin das gerade aktuelle Makro an.

In unserem Falle ist Makro Nummer 1 aktiv, wie Sie sehen werden. Das ist logischerweise im Einschaltzustand immer so.

Darunter finden Sie das komplette Makro in der folgenden, abgekürzten Schreibweise:

```
L cF5 r ! d ! F7
```

Diese Schreibweise werde ich auch weiterhin für die Beispiele benutzen. Wie Sie sehen, steht für RETURN ein **!**, das vorangestellte **c** bedeutet Commodore-Taste und ein vorangestelltes **s** steht für SHIFT. Die Richtungen up, down, left, right werden durch ihre Anfangsbuchstaben **u**, **d**, **l**, **r** abgekürzt. Durch einen beliebigen Tastendruck gelangen Sie wieder in den Grafik-Editor zurück.

Nun wollen wir auch für die waagerechten Linien, die uns zum Gitterraster noch fehlen, ein Makro erstellen. Damit wir jedoch das bereits bestehende Makro nicht verlieren - denn bei einer Neudefinition eines Makros wird sein alter Inhalt gelöscht - wählen wir nun das Makro Nummer 2 an. Dazu geben Sie wieder **SHIFT =**, dann auf die Frage der Nummer jedoch eine 2 ein. Die Extension zeigt uns an, daß dieses Makro noch nicht belegt ist.

Um es zu belegen, benötigen wir allerdings andere Schrittweiten, wie Sie aus Kapitel 2 vielleicht noch wissen. Da zudem die Schrittweiten nicht größer als 255 sein dürfen, brauchen wir als horizontale Schrittweite die 160; die vertikale muß 16 sein. Speichern Sie diese Schrittweiten wieder auf F5!

Danach stellen Sie den Cursor in die Home-Position, wählen mit C= = die Makro-Definition an und zeichnen die erste Linie - denn nichts anderes ist die Makro-Eingabe!

Sie müssen demgemäß folgendes eingeben:

L cF5 d ! r r ! F7

Beendet wird der Vorgang durch =; durch dieselbe Eingabe wird das Makro auch aufgerufen und überzieht dabei unseren Bildschirm mit waagerechten Linien.

Das F7 am Ende des Makros, mit dem der Cursor an den Anfangspunkt der Linie zurückgesetzt wird, wäre für das Zeichnen der ersten Linie nicht nötig gewesen. Jedoch müssen wir daran denken, den Cursor für den nächsten Makro-Aufruf richtig zu plazieren. Da wir die Linie von links nach rechts zeichnen, muß der Cursor für den nächsten Makro-Aufruf wieder an den linken Bildschirmrand gefahren werden.

Nun wollen wir auf unser erstes Makro zurückschalten (SHIFT = und 1). Sie werden vielleicht vermuten, daß Sie, um dieses Makro verwenden zu können, wieder die Schrittweiten 16 und 200 auf F5 einprogrammieren müssen. Doch sehen Sie sich die Schrittweiten auf F5 an. Sie stimmen bereits!

Der Grund: Bei einer Makro-Definition (C= =) wird nicht nur das Makro selbst, sondern auch der komplette Satz der vier Schrittweitenpaare mit abgespeichert. Bei der Anwahl eines Makros (SHIFT =) werden diese Schrittweiten dann wieder zurückgeholt. Damit wird der Tatsache Rechnung getragen, daß die Schrittweiten gewissermaßen zum Makro gehören; denn ein Makro ist nur zusammen mit den Schrittweiten, für die es gedacht ist, sinnvoll anzuwenden. Ich werde deshalb in den folgenden Beispielen immer auch die notwendigen Schrittweiten für F5 und F7 angeben. Für F1 und F3 setze ich die voreingestellten Schrittweiten von 1,1 und 8,8 voraus.

Den Grund, warum die Schrittweiten nur bei der Makro-Anwahl, nicht jedoch beim Makro-Aufruf (=) geholt werden, werde ich Ihnen später nennen.

Bevor wir nämlich zu den komplizierteren Anwendungen kommen, (Abb. 4.4 bis 4.8) wollen wir noch ein paar *einfachere*, dafür aber ebenso nützliche Beispiele betrachten.

Strichpunktierte Linien, wie sie in Konstruktionszeichnungen als Mittellinien verwendet werden (siehe Abb. 2.17.), können wir besonders einfach mittels Makro zeichnen. Für waagerechte Linien brauchen wir folgendes Makro:

```
cF3 L ! r ! r D ! r
```

und für senkrechte Linien:

```
cF3 L ! d ! d D ! d
```

Geben Sie diese Makros ein und überzeugen Sie sich von der Funktion!

Natürlich sind Sie nicht auf waagerechte und senkrechte Linien beschränkt. Durch entsprechende Schrittweiten können Sie jede Steigung erzielen. Hier zum Beispiel die Makros für die einfachsten Fälle schräger Linien, nämlich für gestrichelte Linien im 45-Grad-Winkel. Das erste Makro ist für eine Linie von links oben nach rechts unten, das zweite für links unten nach rechts oben:

```
L cF3 ! r d ! r d
L cF3 ! r u ! r u
```

Auch das Erstellen von Schraffuren wird mit Makros wesentlich erleichtert. Wie Sie aus Kapitel 2 wissen, braucht man dazu zunächst einen vollen Bildschirm mit der gewünschten Schraffur. Programmieren Sie dazu auf F5 die Schrittweiten 100,200, setzen Sie den Cursor in die linke untere Bildschirmecke und tippen Sie folgendes Makro ein:

```
L cF5 ! u r ! F7 cF3 r
```

Mit diesem Makro können Sie einen Bildschirm wie den in Abb. 4.2 gezeigten erzeugen. Dabei ist nur etwa das mittlere Drittel, das ich in dieser Abbildung gestrichelt markiert habe, zu verwenden. Wir verschieben also dieses Drittel nach links und rechts, um einen vollständig mit der Schraffur ausgefüllten Bildschirm zu erhalten. Das Verschieben mit Move ist problemlos möglich, da die Linien einen waagerechten Abstand von 8 Punkten haben!

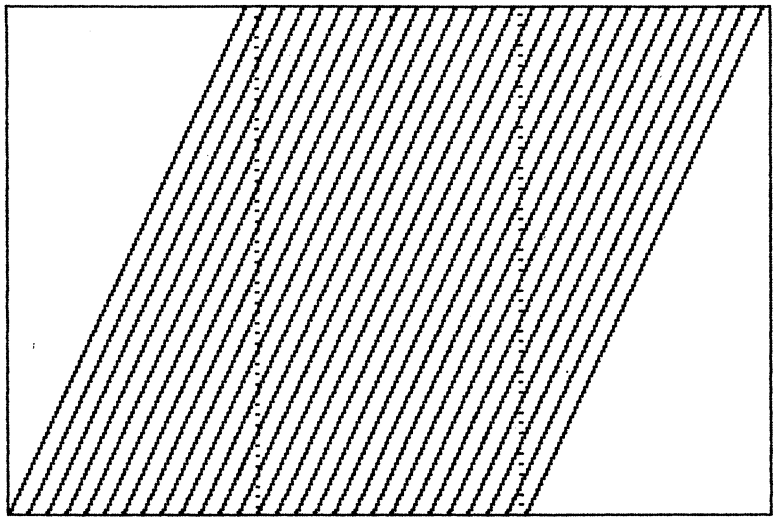


Bild 4.2: Eine Schraffur, mittels Makro erzeugt

Und was wäre passiert, wenn Sie das Makro noch aufgerufen hätten, während das obere Ende der Linie bereits am rechten Bildschirmrand anstößt? Probieren Sie es aus!

Für flachere Schraffuren ist das eben gezeigte Makro nicht so gut geeignet, da dann der nutzbare Bereich schmaler werden würde. Um dieses Problem zu umgehen, fallen Ihnen bestimmt mehrere Lösungen ein. Ich möchte Ihnen eine besonders komplizierte Lösung vorführen, um dadurch einen neuen Aspekt einzubringen.

Bisher haben wir nur immer die Schrittweiten als Vorraussetzung für das Funktionieren der Makros benutzt. Wir können aber auch eine Tabulatorposition verwenden, um das Makro daran *anzuschließen.*

Dies wollen wir für die in Abb. 4.3 dargestellte Schraffur tun. Das dazugehörige Makro lautet:

```
L cF3 d ! F5 ! r sF5 F7
```

Bevor Sie jedoch dieses Makro eingeben, stellen Sie den Cursor in die HOME-Position, fahren ihn von dort einen Schritt (= 8 Punkte, wir benutzen die Schrittweiten auf F3) nach rechts und setzen dort einen Tabulator auf F5 (**SHIFT F5**). Dann fahren Sie den Cursor in die HOME-Position zurück und geben das Makro ein.

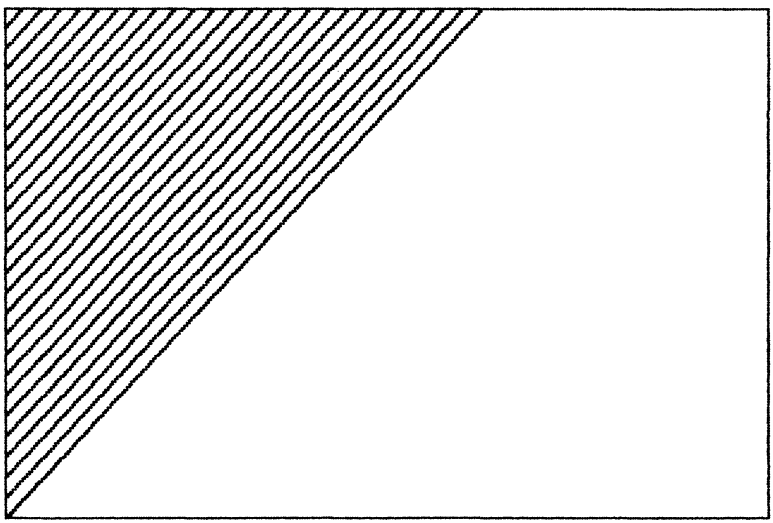


Bild 4.3: Eine flachere Schraffur, ebenfalls mittels Makro erstellt

Um das so entstandene Schraffur-Dreieck über den ganzen Bildschirm auszubreiten, kopieren Sie zunächst die obere Bildschirmhälfte nach unten und vervielfältigen dann den brauchbaren Streifen am linken Rand nach rechts.

Sie können ruhig einmal ausprobieren, was passiert, wenn die Tabulatorposition vor Aufruf des Makros nicht stimmt, sondern zum Beispiel einen Schritt zu weit rechts liegt. Das Ergebnis mag zwar künstlerisch ansprechend sein, als Schraffur in einer Konstruktionszeichnung ist es aber nicht geeignet!

Eine weitere Anwendung dieser Art, nämlich das Zugreifen auf Tabulatorpositionen, ist ein Befehl zum Zeichnen dicker Linien. Dieser Befehl entspricht gewissermaßen dem **SHIFT D**, gilt aber für Linien. Das Makro dazu lautet:

```
L cF1 F1 u ! F3 u ! F1 d ! F3 d ! F1 l ! F3 l ! F1 r ! F3 r ! F3 sF1
```

Vor Aufruf dieses Makros müssen Sie auf F1 und F3 die Endpunkte dieser Linie speichern. Das Makro sieht ein bißchen länglich aus, aber um für alle möglichen Steigungen eine gleich dicke Linie zu bekommen, sind vier einzelne Linien nötig. Das **F3 sF1** am Ende wäre für die Funktion des Makros nicht nötig. Es erleichtert jedoch das Zeichnen eines zusammenhängenden Linienzuges, da man nur immer den neuen Zielpunkt auf F3 speichern muß. Die Belegung von F1 mit dem Startpunkt nimmt das Makro selbst vor.

Ich hoffe, Sie haben nun den Umgang mit Makros so weit im Griff, daß wir uns an die kompliziertesten Anwendungen wagen können! Falls nicht, üben Sie erst noch ein bißchen!

Wie soll man zum Beispiel die nette Grafik in Abb. 4.4 herstellen? Sie können gar nicht von selbst darauf kommen, da ich Ihnen bisher zwei Befehle, die in einem Makro vorkommen dürfen, noch vorenthalten habe. Es handelt sich dabei um **H** und **V**, allerdings in einer etwas abgeänderten Form.

Am besten sehen wir uns zuerst einmal das Makro an; die Erklärung dazu folgt später. Vor dem Eintippen müssen Sie allerdings noch die Schrittweiten 6,6 auf F5 speichern und den Cursor in die Nähe der HOME-Position setzen.

```
R cF5 ! d r ! F7 H+0 V+6 cF1 d d r r
```

Die Befehle **H** und **V** geben Sie so ein, wie sie hier abgebildet sind, also nacheinander **H**, **+** und **0** bzw. **V**, **+** und **6**. Diese Eingabe erfolgt blind, d.h., ohne daß Sie sehen, was Sie tippen!

Und nun rufen Sie das Makro ein paarmal auf. Wenn Sie alles richtig gemacht haben, dann muß ein ähnliche Grafik wie in Abb. 4.4 entstehen.

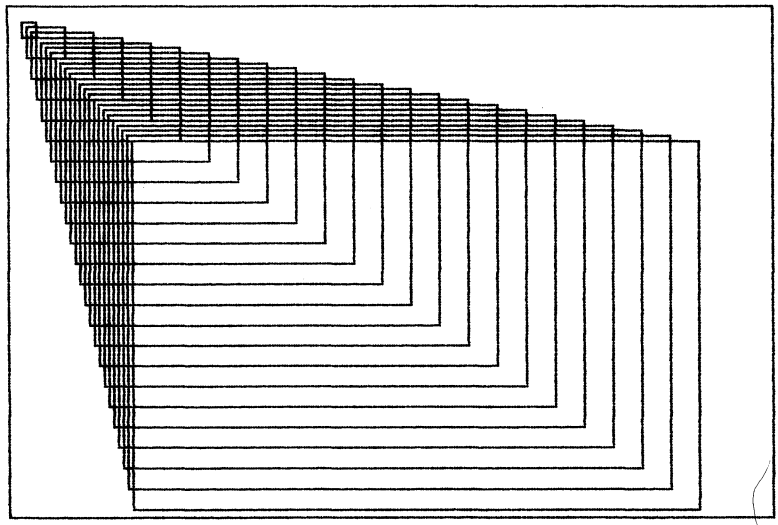


Bild 4.4: Rechtecke, ein etwas komplizierteres Makro

Und nun zur Erklärung: Wie Sie vielleicht schon vermuten, verändern die Befehle **H** und **V**, die in dieser Form nur in Makros verfügbar sind, die aktuelle Schrittweite. Mit **V+6** wird die aktuelle vertikale Schrittweite um 6 vergrößert. Analog würde Sie mit **V-6** um 6 verkleinert. Das Programm achtet dabei natürlich selbst darauf, daß die Grenzen (1 bis 255) nicht überschritten werden. Da nur eine Ziffer eingegeben werden kann, sind Schrittweitenänderungen von 1 bis 10 möglich. Die **0** steht dabei für 10.

Da die Eingabe blind erfolgt, ist das Programm auch nicht so kleinlich mit der Syntax. Statt **H+0** können Sie auch **+H0**, **H0**, **+0** oder nur **0** eingeben. Das Pluszeichen kann ebenso wie das H weggelassen werden, nicht dagegen - und V. Bei der Reihenfolge müssen Sie lediglich darauf achten, daß die Ziffer als letztes eingegeben wird.

Jetzt verstehen Sie auch, warum die Schrittweiten, die zusammen mit dem Makro gespeichert werden, nicht bei jedem Makro-Aufruf geholt werden. Dann wären nämlich die Befehle **H** und **V** in dieser Form sinnlos. Wenn Sie

jetzt die Schrittweiten auf F5 anschauen, werden Sie viel größere Werte als am Anfang finden. Um die Anfangswerte zurückzuholen und das Makro noch einmal von vorn starten zu können, wählen Sie es mittels **SHIFT =** wieder an.

Falls Sie die gespeicherten Schrittweiten ändern wollen, ohne das Makro neu einzugeben, gehen Sie folgendermaßen vor: Schalten Sie mit **C=** = auf Makro-Eingabe, tippen Sie jedoch als erstes Zeichen den Pfeil nach links. Das Makro wird dann nicht geändert, es werden jedoch die momentan eingestellten Schrittweiten zum Makro gespeichert.

Noch etwas anspruchsvoller ist das Makro, mit dem Abb. 4.5, das Alphorn, erzeugt wurde. Es lautet:

```
C cF7 ! d ! u r H+1 V+2 cF5 d
```

Als Schrittweiten geben Sie vor dem Eintippen des Makros 6,6 auf F5 und 3,3 auf F7 ein.

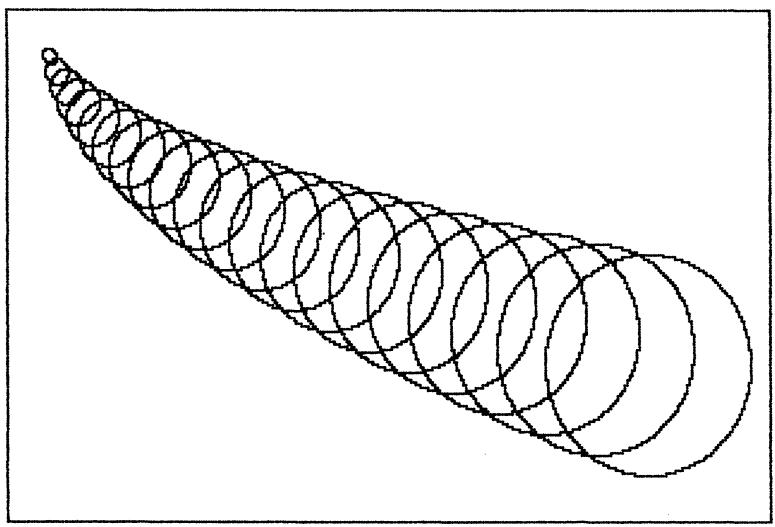


Bild 4.5: Mittels Makro erzeugtes "Alphorn"

Das Besondere an diesem Makro ist, daß nicht nur die Größe des Kreises, wie auch zuvor beim Rechteck, sondern auch noch der horizontale Abstand der Kreise zueinander geändert wird.

Ein weiteres Makro zum Staunen ist das folgende:

```
M cF3 h ! F1 1 ! h r X !
```

Das **h** steht für HOME. Bevor Sie dieses Makro jedoch eintippen, laden Sie die Olympischen Ringe, die Sie in Kapitel 2 (hoffentlich) gezeichnet haben und speichern sie auf der Funktionstaste F1 als Tabulator die rechte untere Bildschirmecke (ist im Einschaltzustand bereits der Fall). Wenn Sie die Ringe nicht mehr haben, zeichnen Sie sie jetzt - natürlich unter Zuhilfenahme der Makros. Das ist eine gute Übung!

Haben Sie dann das obige Makro eingetippt und einige Male aufgerufen, wird Ihr Bild zunächst sehr *zerzaust* aussehen. Am Schlimmsten wird es nach dem 63. Makroaufruf, dann sieht das Bild aus wie Abb. 4.6. Doch beim 64. Aufruf haben Sie nichts desto trotz wieder das Originalbild vor sich, vollkommen unbeschädigt!

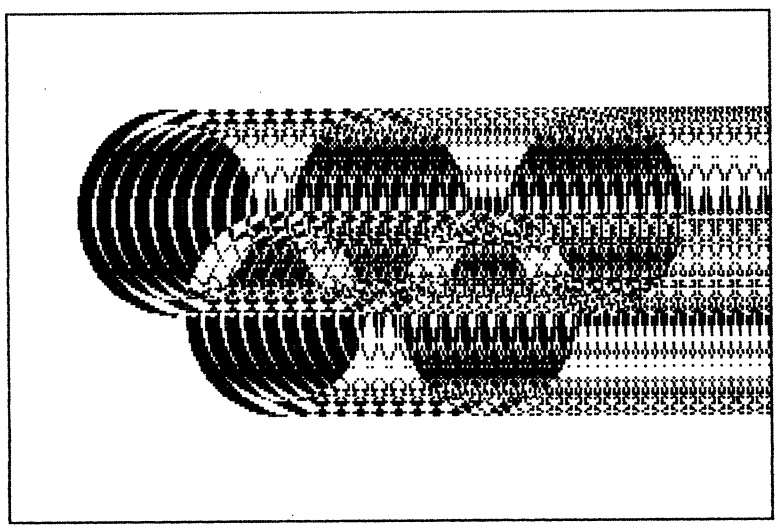


Bild 4.6: Tolle Effekte durch EXOR-Makro

Ein weiteres *Trickmakro* diente zur Erzeugung von Abb. 4.7. Vor dem Eintippen brauchen Sie auf F5 die Schrittweiten 6,6 und auf derselben Taste muß als Tabulator die Position auf halber Höhe des Bildschirms nahe des rechten Randes gespeichert sein. Der Cursor selbst soll sich beim Eintippen nahe der HOME-Position befinden. Hier nun das Makro:

```
L cF5 ! F5 ! 1 d sF5 F7 r H+1
```

Sie ahnen sicher schon, daß mit etwas Phantasie die unmöglichsten Gebilde möglich sind. Mit nur einer Zeile (allerdings nicht gerade in einer gut leserlichen Form) sind somit Grafiken möglich, für die Sie in BASIC (natürlich mit Erweiterung) oder einer anderen grafikfähigen Programmiersprache immerhin ein Programm brauchten, an dem Sie wohl mindestens genauso lang herumtüfteln müßten.

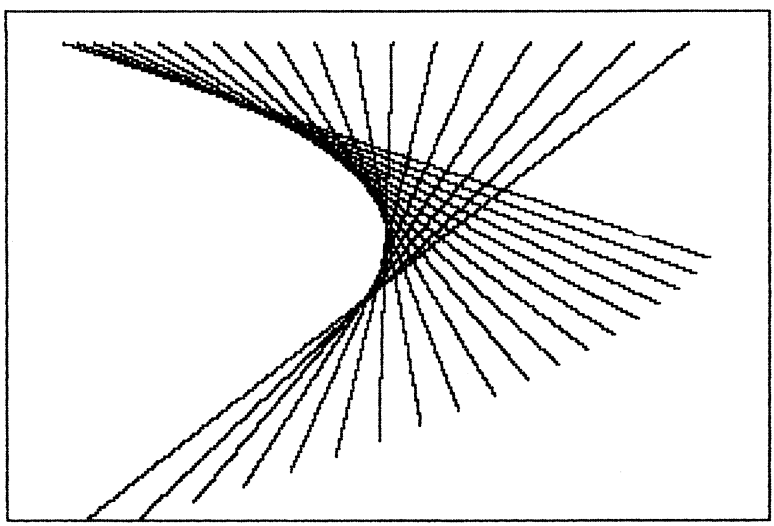


Bild 4.7: Ein weiteres "ausgefuchstes" Makro

Daß die mittels Makro erzeugten Grafiken erst das Grundgerüst zu anspruchsvollen Kunstwerken sind, zeigt Abb. 4.8. Dort habe ich lediglich mit Paint einiges aufgefüllt und mit Hilfe der Zoom-Funktion einige Linien, die den 3D-Eindruck störten, beseitigt.

Beinahe hätte ich vergessen Ihnen zu zeigen, wie die Schrift in Kapitel 2 ins Kaleidoskop gekommen ist. Sie erinnern sich, dort hatten die Buchstaben aus dem Construction-Set einen Rand, der Sie von der Umgebung abhob.

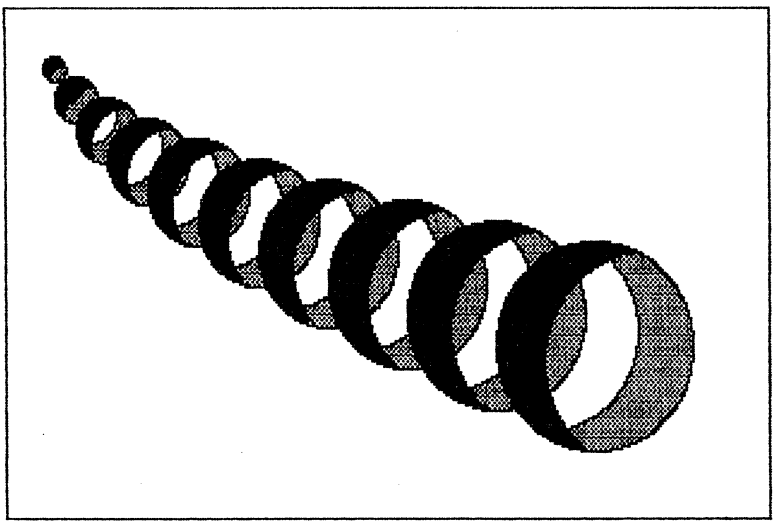


Bild 4.8: Nachbearbeitete Makro-Grafik

Und wie geht das? In Kapitel 2 habe ich Ihnen gezeigt, wie man Buchstaben pixelgenau löschen kann, indem man nämlich den Bildschirm invertiert, dann das Zeichen druckt (Append) und anschließend den Bildschirm zurückinvertiert. Wenn wir nun den Buchstaben mehrmals löschen - jedesmal gegenüber der eigentlichen Position in eine andere Richtung verschoben - erhalten wir einen dicken gelöschten Buchstaben. Dorthinein drucken wir dann den eigentlichen Buchstaben.

Das Löschen aller acht Nachbarpositionen ist allerdings mit vielen Eingaben verbunden; deshalb werden wir es jetzt per Makro machen:

```
A cF1 I l u ! r ! r ! d ! d ! l ! l ! u ! r I ! cF3
```

Die Handhabung des Makros ist sehr einfach: Sie holen den gewünschten Buchstaben aus dem Construction-Set und plazieren ihn wie gewohnt. Statt den Knopf zu drücken, rufen Sie das Makro auf (=). Natürlich sind die Cursorbewegungen innerhalb des Makros so gewählt, daß der Cursor nach dessen Aufruf wieder an seiner Ausgangsposition steht. Dazu darf der Cursor beim Aufruf aber nicht am Bildschirmrand anliegen! Außerdem schaltet das Makro wieder auf die Schrittweite F3 zurück, die Sie wahrscheinlich wegen der Ausrichtung der Schrift für den Move-Modus benötigen.

Wenn Sie auf diese Art jedoch die Buchstaben direkt ins Kaleidoskop oder in ein anderes Bild eintippen, hat das einen wesentlichen Nachteil: Sie können die Schrift nicht mehr verschieben, da Sie dabei den Hintergrund mit verschieben würden! Um das zu umgehen, empfehle ich Ihnen, die Schrift zunächst in einen *vollen* Bildschirm (also löschen und invertieren) zu schreiben. Das Ergebnis würde dabei so aussehen, wie in Abb. 4.9, wo Sie die Schrift noch beliebig verschieben können.

Um die Schrift dann ins Kaleidoskop zu bekommen, bedienen wir uns wieder der Verknüpfungsbefehle. Machen Sie sich zunächst von beiden Bildschirmen zur Sicherheit Kopien, wählen Sie dann den Bildschirm mit dem Kaleidoskop an und verknüpfen Sie ihn per Und mit der Schrift. Dadurch entsteht bereits der Rand. Um die Schrift selbst ins Bild zu bekommen, müssen Sie alles Überflüssige im Schrift-Bildschirm weglöschen. Wie das geht, wissen Sie ja schon: Bildschirm invertieren und die entstandenen Löcher, zum Beispiel in den Os, sowie den ganzen Bildschirm um die Schrift herum mit Paint auffüllen. Nach dem Zurückinvertieren darf nur noch die Schrift zu sehen sein, ohne Rand oder sonstige Zeichen. Diese Schrift blenden Sie dann mittels Oder-Verknüpfung in das Kaleidoskop ein. Fertig!

Bild 4.9: Ebenfalls mittels Makro: "Rand-Schrift"

4.8 Abschalten der Erweiterung

Ist die Erweiterung geladen, so dient der Befehl Quit (**C= Q**) nicht mehr zum Abschalten des Hi-Eddi plus, sondern zum Abschalten der Erweiterung. Alle neuen Befehle und Möglichkeiten sind dann wieder vergessen. Auch der durch die Erweiterung belegte Bildschirmspeicher ist dann wieder frei und der Walk-Befehl wieder entriegelt. Außerdem schaltet der Quit-Befehl wieder den Hi-Eddi plus ab.

Die Erweiterung können Sie aus dem Bildschirm, in den Sie sie geladen hatten, löschen.

5 Programmdokumentation

Falls Sie von Assemblersprache nichts verstehen, dann werden Sie wohl mit dem, was nun folgt, nicht viel anfangen können. Wenn Sie eines Tages jedoch selbst in Assembler programmieren können, wird Ihnen dieses letzte Kapitel sicherlich auch von Nutzen sein.

Ich muß zugeben, daß Assembler weder eine leichte noch eine übersichtliche Programmiersprache ist. Für jedes kleine Problem, das Sie in den meisten höheren Programmiersprachen mit einem oder wenigen Befehlen lösen können, müssen Sie in Assembler ein ganzes, oft sehr langes Programm schreiben. Allein das Multiplizieren zweier Zahlen, das Sie zum Beispiel in BASIC *einfach hinschreiben* können, erfordert in Assembler schon ein nicht unbedingt leicht zu durchschauendes Programm. Dennoch ist Assembler die universellste und leistungsfähigste aller Sprachen! Mit ihr können Sie alle Probleme lösen, auch die, für die die höheren Sprachen nicht mehr geeignet sind. Und wenn Sie es richtig anstellen, sind in Assembler geschriebene Programme im Ablauf schneller als alle, die Sie mit anderen Programmiersprachen erstellen können.

Das muß allerdings nicht so sein! Angaben wie *50 kByte reiner Maschinencode*, mit denen gern und viel geworben wird, zeugen nicht immer von besonders guter Qualität. Oft umschreiben sie vielmehr die Unfähigkeit des Programmierers, effektive Programme zu schreiben!

Ich will Sie mit diesen etwas kritischen Worten keineswegs abschrecken, sondern Ihnen nur aufzeigen, daß Assembler nicht so leicht zu erlernen ist wie zum Beipiel BASIC, und daß auch berufliche Programmierer nicht immer Profis im Umgang mit Assembler sind. Dennoch laufen deren Programme - von Ausnahmen einmal abgesehen. Von Ihnen als Hobbyprogrammierer verlangt erst recht niemand, daß Ihre Programme bis aufs letzte Byte optimiert sein sollen.

Falls Sie Interesse haben, sich in die faszinierende Welt der Assemblerprogrammierung vorzuwagen, kann ich Ihnen die Kurse über Assemblerprogrammierung und Grafik von Heimo Ponnath, erschienen im 64er, wärmstens empfehlen. Es gibt zwar zu diesen Themen auch eine Menge Literatur; ich habe bisher jedoch noch kein Buch endeckt, in dem die Assemblerprogrammierung so verständlich und ausführlich dargelegt ist - insbesondere für den Anfänger - wie in den erwähnten Kursen.

Doch nun zu denjenigen unter Ihnen, die bereits in der Assemblerprogrammierung geübt sind und erfahren wollen, wie man mit dem Hi-Eddi plus noch arbeiten kann.

Ich hätte hier nun das Quellcodelisting, das von mir recht brauchbar kommentiert ist (über 60 kByte Sourcecode für 6 kByte Objektcode), abdrucken können. Das möchte ich jedoch aus zwei Gründen nicht tun:

Erstens ist es für dieses Buch zu lang und zweitens bringt es meiner Meinung nach nicht viel.

Auch im Quellcode sucht man einige Zeit herum, bis man herausgefunden hat, welche Parameter eine Routine braucht, welche Sie zurückgibt und welche Speicherzellen Sie beeinflußt. Außerdem können Sie sich ein Disassemblerlisting mittels Disassembler jederzeit selbst erstellen.

Deshalb habe ich mich dafür entschieden, Ihnen eine ausführliche Beschreibung aller Zeropage- und sonstigen Speicherstellen sowie aller Routinen mit Einsprungadresse, Parameter und Funktionsbeschreibung zu erstellen. Je nach Wichtigkeit werde ich dabei auf die verschiedenen Speicherzellen und Routinen unterschiedlich genau eingehen. Nur in den seltensten Fällen werde ich erläutern, wie eine Routine genau arbeitet. Für die Benutzung dieser Routine ist das auch gar nicht wichtig. Wichtig ist nur, was sie tut und welche Parameter sie braucht und zurückgibt. Den Rest der Rou tine kann man als Black Box betrachten.

Nach der Dokumentation des Hi-Eddi plus selbst folgt dann diejenige für die Druckerroutine, wobei ich besonders auf die notwendigen Änderungen, die die Anzahl der auszudruckenden Bytes betreffen (wichtig zum Beispiel für breite Drucker), eingehen werde.

Eine ausführliche Dokumentation der Erweiterung erspare ich mir, da sie aus wenigen, dafür aber um so komplexeren Routinen besteht, die man kaum in anderem Zusammenhang benutzen wird. Die einzigen interessanten Teile, die man auch anderweitig einsetzen kann, sind das Adventure-Grafik-System und vor allem der Bitmap-Compander. Für letzteren möchte ich Sie auf den entsprechenden Artikel in der 64er verweisen (siehe Kapitel 4).

In einem gesonderten Abschnitt wird auf das Einbinden einer Erweiterung in Hi-Eddi plus eingegangen. Dabei werde ich dann natürlich die Extension, genauer gesagt ihren Initialisierungsteil, als Beispiel heranziehen.

5.1 Arbeitsspeicherzellen des Hi-Eddi plus

Ich werde im folgenden bei allen Adressen sowie bei den Routinen die Namen angeben, die ich im Quellentext benutzt habe, und werde auch auch bei der weiteren Beschreibung nur diese Namen verwenden. Die Namen sagen nämlich mehr aus als die Adressen und sind außerdem einprägsamer. Wenn Sie natürlich das Programm mittels Disassembler durchforsten, haben Sie nur die Adressen vor sich. Da ich jedoch Speicherzellen und Routinen nach steigender Adresse anordne, finden Sie schnell zu einer Adresse auch den Namen.

```
$02-$03  PTR        $04-$05  PTR2
```

Pointer für verschiedene Zwecke, vor allem für Bitmaps und Video-RAM.

```
$06      PY
$07      PXL        $08      PXH
```

Pixelkoordinaten gemäß der üblichen Bildschirmeinteilung: PY von 0 bis 199 und PX von 0 bis 319, Nullpunkt links oben.

```
$09      BITPOS
```

Bitmaske, die die Position eines Punktes innerhalb eines Bytes im Grafikbildschirm angibt.

```
$0A      SPRBIT     $0B      ZWS
$0C      FLAG       $0D      BITS
$0E      ZEILE      $0F      SPALTE
$10      TMP        $11      SPHPTR
$12      NUMBER     $13      WORK
$14      COUNT      $15      BYTES
```

Diverse Zwischenspeicher, Zähler und Flags zur temporären Benützung.

```
$16      MTZ1       $17      MTS1
$18      MTZ2       $19      MTS2
```

Arbeitsbereich für die Befehle Mirror und Turn.

$1A MCNT $1B MTSP
$1C CNT

Zählvariable, temporäre Benutzung.

$1D COLFL

Betriebsarten und Colorflag, entspricht dem Betriebsartenbyte, das beim alten Hi-Eddi bei Programmstart eingegeben werden mußte. Die Bedeutung der Bits:

```
Bit7: 0: Schwarzweiß
      1: Farbe
Bit6: 0: Tastatureingabe
      1: Menüeingabe
Bit0: Beim Programmstart:
      0: Alle Bildschirme löschen
      1: Nichts löschen
      Während des Programmlaufs:
      0: Background-Colormode
      1: Foreground-Colormode
```

Die Bits 1 bis 5 werden von Hi-Eddi plus nicht benutzt und bei Programmstart alle auf 0 gesetzt. Die Erweiterung nützt diese Bits.

$1E MEMSHF

Letzter Shift-Code, entsprechend der Systemzelle $28D.

$1F JPORT

Letzter Joystick-Port-Wert, entsprechend dem Register $DC00, aber invertiert. Auch die Pad-Eingaben und die RETURN-Taste werden in diesem Byte entsprechend den Joystick-Eingaben berücksichtigt.

$20 TIME

Momentane Verzögerungszeit für die Beschleunigung des Cursors bei Joysticksteuerung. Nimmt Werte zwischen 42 (Anfangsgeschwindigkeit) bis herunter zu 6 (schnelle Endgeschwindigkeit) bzw. 16 (langsame Endgeschwindigkeit) an. Bei Pad-Steuerung ist es immer 0.

$21 SPRNR

Nummer der Sprite-Form des aktuellen Cursor-Sprites. Die Nummern sind:

```
0:  Kreuz
1:  Kleiner Rahmen (Text)
2:  großer Rahmen (Sprite-Befehle)
3:  Kreuz für Draw mit dickem Pinsel und Paint
4:  Fläche als Sprite-Hintergrund im Sprite-Editor
```

$22 MODUS

Enthält den aktuellen Modus. Die Nummern sind:

```
0:  Draw
1:  Draw dick
2:  Line
3:  Rec
4:  Circle
5:  Paint
6:  Jots (Spray)
7:  Move
8:  Fore/Back
9:  Text
10: Menü
11: Stamp
12: Append
13: Get
14: Erase
```

Zur Unterscheidung zwischen Fore und Back dient Bit 0 in COLFL.

$23 SY
$24 SXL $25 SXH

Momentane Sprite-Koordinaten, entsprechend den VIC-Registern.

$26 COLOUR

Farben des Grafik-Editors im Schwarzweiß-Betrieb sowie des Sprite-Editors im Farb-Betrieb. High Nibble ist Vordergrund, low Nibble Hintergrund.

$27 STPCNT

Nummer der gerade angewählten Schrittweite, 0 bis 3 entsprechend Funktionstaste F1 bis F7.

$28-$2B HSTTAB $2C-$2F VSTTAB

Tabelle der vier horizontalen und vier vertikalen Schrittweiten.

$30 GRIDFL

Grid-Flag für Sprite-Editor. 1 = Grid aus, 3 = Grid ein.

$31 ZFLAG

Zeichensatzflag. Ist 0, wenn ROM-Zeichensatz. Bei RAM-Zeichensatz enthält es die Nummer des Bildschirms, in dem der Zeichensatz liegt.

$32 SRFLAG

Schreibrichtungsflag, 0 = normal, 1 = abwärts, 2 = auf dem Kopf stehen, 3 = aufwärts.

$33 SPEED

Maximale Cursorgeschwindigkeit bei Joysticksteuerung, entweder 6 (schnell) oder 16 (langsam). Siehe auch TIME.

$34 INVFLAG

Inversflag für Textmodus. 0 = normal, $80 = invers.

$35-$3B ORGTAB

Tabelle zur Bildschirmorganisation. $35 enthält die Nummer der Bitmap ab $2000 (sichtbar), $36 die ab $4000, $37 die ab $6000 usw. Im Farb-Betrieb steht in $36 eine 0, da von $4000 bis $6000 die Farbinformationen liegen.

$3C-$47 MEMTAB

Tabelle der Tabulatoren, in der Anordnung 4 mal Y-Koord., 4 mal X-Low, 4 mal X-High (jeweils Sprite-Koord., also entsprechend SY, SXL, SXH).

$48 MKSCR

Nummer des Quellbildschirms für Move.

$49 CRSR

Cursor-Blink-Flag. 0 = Cursor blinkt, $80 = Cursor einfarbig.

$4A PADFL

Pad-Flag. Ist 0, wenn Koala Pad angeschlossen, sonst 42. Dies ist zugleich die Verzögerung der Cursorbewegung bei Joysticksteuerung (TIME).

$4B DISMIN

Koala Pad-Empfindlichkeit. Ist 2 für empfindlich (+), 4 für unempfindlich (-) und 8 im Sprite-Editor. Diese Werte sind die minimale Distanz, die der Stift auf dem Pad bewegt werden muß, um eine Bewegung des Cursors zu bewirken.

$4C EXTFLAG

Extension-Flag. Wird bei Programmstart mit 0 belegt und war ursprünglich als Flag für eine geladene Extension gedacht. Wird aber bisher nicht benutzt.

$4D-$4E MEMAD

Für Koala Pad-Steuerung. Speichert die letzten AD-Wandler-Werte, um Bewegungen des Stiftes zu erkennen.

$4F UOXFL

Flag für Und, Oder, Exor. Normalerweise 0, nach der Eingabe von 'U', 'O', 'X' sind folgende Werte enthalten:

```
Or:    $11
Und:   $31
Exor:  $51
```

Das sind die Opcodes dieser Befehle für nachindizierte Adressierung. Vorsicht: Hier ist Programmodifikation im Spiel!

$50 LRCFL

Flag für Line, Rec, Circle: Normalerweise 0, 1 nach dem ersten Knopfdruck, nach dem zweiten wieder 0.

$51 SLIDE $52 WNDFLAG

Bildschirmverschiebung und Sprite-Fenster-Position im Sprite-Editor/Zoom-Funktion.

$53-$56 COLTAB

Farbentabelle für Grid-Funktion im Sprite-Editor/Zoom-Funktion

$57 MPY
$58 MPXL $59 MPXH

Zwischenspeicher für die Pixelkoordinaten PY, PXL, PXH. Dient bei Line, Rec, Circle zur Speicherung des ersten Knopfdruckes.

$5A MEMSCR

Dient bei Menüanwahl zur Speicherung der aktuellen Bildschirmnummer.

$5B-$5E MEMBUF

Zur Speicherung des aktuellen Modus und der Sprite-Koordinaten bei Menüanwahl.

$5F-$66 MKBUF

Markierungsbuffer, speichert Quell- und Ziel-Bereich bei Move: Die ersten beiden Bytes geben die linke obere Ecke des Quellbereichs (jeweils Zeile und Spalte entsprechend der Lowres-Einteilung des Bildschirms), die nächsten beiden die rechte untere Ecke des Quellbereichs und die nächsten vier dasselbe für den Zielbereich. Die Quellbereichsmarkierung stimmt freilich erst nach dem zweiten Knopfdruck, die Zielbereichsmarkierung nach dem dritten.

$67 MKZEIL $68 MKSPAL

Anzahl der Zeilen - 1 bzw. Spalten - 1 des Bereichs bei Move.

$69-$6A MKPTR

Pointer auf den Quellbereichsanfang (Video-RAM-Adresse, nicht Bitmap!)

$6B MKFLAG

Markierungs-Flag bei Move: 0 nach Move-Anwahl oder Korrektur (Pfeil nach links), 1 nach dem ersten Knopfdruck, 2 nach dem zweiten oder Move Restore und 3 nach dem dritten. Nach dem vierten dann wieder 1, egal, ob vorher 3 oder 0 war.

$6C NAMLEN

Filenamenslänge

$6D LSFLAG

Load-Save-Flag: 0 = Load relativ, 1 = Save, $FF = Load absolut.

$6E-$6F STADR $70-$71 ENDADR

Start- und Endadresse für Load und Save.

$100 SPH

Die unteren 64 Bytes des Stack werden von den Sprite-Editor-Befehlen Mirror, Turn und Rotate als "Hilfssprite", also Zwischenspeicher, benutzt.

$200 BUF

80 Bytes als Eingabebuffer wie bei BASIC sowie als universeller Arbeits-Buffer, zum Beispiel für Line und Circle. Bei Zoom wird hier das aktuelle Sprite zwischengespeichert.

$2A7 SEQSTR

80 Bytes für den Sequenzstring.

$340 SPRA

Aktuelles Cursor-Sprite. Mögliche Cursorsprites siehe SPRNR. Aus Speicherplatzgründen existiert nur jeweils die Form des aktuellen Cursorsprites.

$380 SP0

Aktuelles Sprite für Get, Stamp, Append, Sprite-Editor. Wird während Zoom in BUF zwischengespeichert, da dann hier der vergrößerte Bildschirmausschnitt liegt.

$3C0 SPX

Alternatives Sprite, dessen Inhalt durch 'C' mit SP0 ausgetauscht wird.

An Betriebssystem-Zellen werden folgende benutzt:

$90	STATUS	Floppy-Status
$9D	BITMSG	Systemmeldungen unterdrücken
$A2	CLOCK	Interrupt-Uhr
$C6	ANZTAS	Anzahl Tasten im Tastaturbuffer
$CB	TASTE	Tastaturcode
$286	ZCOL	Schriftfarbe
$28D	SHIFT	Shift-Code
$318	NMIVEK	NMI-Vektor

Das Programm selbst belegt den Bereich von $0801 bis $1FE2. Ab $2000 wird das gesamte RAM von Bitmaps belegt, zumindest im Schwarzweiß-Betrieb. Bei Farbe dient der Bereich von $4000 bis $6000 als Speicher für die Farbinformationen, also die Video-RAMs.

Dabei ist $4000 bis $4400 ein Zwischenspeicher für das Video-RAM (VZWS), wenn der Lowres-Screen eingeschaltet wird. Daher kann bei Load und Save der Bereich $2000 bis $4400 komplett gespeichert oder geladen werden. Von $4400 bis $5C00 liegen die Video-RAMs der sechs Bildschirme, und zwar in der Reihenfolge der Bildschirmnummern: ab $4400 liegt Nummer 1, ab $4800 Nummer 2 usw. Dies entspricht also nicht der Organisation der Bitmaps, die völlig ungeordnet im Speicher liegen und deshalb das Anlegen einer Tabelle (siehe ORGTAB) nötig machen.

Der Bereich von $5C00 bis $6000 ist frei! Man könnte ihn für eine Extension nützen, aber eben nur bei Farb-Betrieb. Bei Programmstart wird dieser Bereich mit Nullen überschrieben (außer man antwortet auf die Frage 'LOESCHEN' mit 'N').

5.2 Beschreibung der Routinen des Hi-Eddi plus

Nun zu den Routinen. Wie bereits gesagt, werde ich die genaue Beschreibung einer Routine nur dann vornehmen, wenn es insbesondere für Erweiterungen wichtig ist. Am offensichtlichsten ist dies der Fall bei der Befehlseingabe und dem anschließenden Verteiler.

In den meisten Fällen werde ich lediglich die Funktion einer Routine sowie unter der Rubrik *Eingaben* die nötigen Eingabeparameter angeben. Daneben gibt es noch die Rubriken *Temporär* und *Ausgaben*. Beide Rubriken geben die Speicherstellen an, die von der Routine beeinflußt werden. Während jedoch die temporären nur als lokale Zwischenspeicher dienen und nach Verlassen der Routine keine wichtigen Werte beinhalten, werden die Ausgabeparameter von einer anderen oder derselben Routine wieder gelesen. Die Ausgabeparameter sind meist globale Parameter. Möchte man sie - eventuell in einer Erweiterung - ändern, muß man sich genau überlegen, welche Routinen auf diese Parameter zugreifen und welche Folgen eine Änderung haben kann. Die meisten Routinen prüfen nämlich beim Zugriff nicht, ob diese Parameter auch sinnvolle Werte enthalten. Solch eine Prüfung ist nur bei Eingaben vom Benutzer her sinnvoll, innerhalb des Programms werden gültige Werte vorausgesetzt.

So müssen zum Beipiel die Werte für die Pixelkoordinaten PY, PXL, PXH innerhalb der erlaubten Grenzen liegen, sonst gibt es Probleme.

Zusätzlich gebe ich bei den meisten Routinen die verwendeten (Unter) Routinen an. Deren Parameter kommen dann natürlich noch zu den angegebenen dazu.

Manche Routinen besitzen auch mehrere Einsprungadressen. Ich werde diese jedoch nur angeben, wenn es für Erweiterungen interessant sein kann.

```
$080F  GRAFED        Hauptprogramm
$0815  MAIN          Hauptschleife
```

Zunächst wird der Stack initialisiert und die Kaltstart-Routine aufgerufen. Ab MAIN läuft die Hauptschleife, in der nacheinander die Routinen zur Tastaturabfrage, Joystick-Richtungsabfrage, Knopfabfrage und Cursorfarbe aufgerufen werden. Das ganze Hauptprogramm ist 31 Bytes lang. Da es sich hierbei nicht um eine Routine handelt, gibt es auch nichts über Parameter zu sagen.

`$082E TAST` Tastaturabfrage

Eingaben: TASTE, SHIFT (beides Systemzellen), MODUS
Ausgaben: XR (Tastaturcode), YR (ASCII), MEMSHF

Wurde eine Taste gedrückt, so wird geprüft, ob der Textmodus aktiv ist, die CTRL-Taste gedrückt ist oder ein Steuerzeichen vorliegt. Entsprechend wird dann nach TEXT oder BEFEHL verzweigt.

`$084B BEFEHL` Auswertung einer Befehlseingabe

Eingaben: XR (Tastaturcode)
Ausgaben: XR (Befehlsnummer)

Wenn der in XR übergebene Tastaturcode einem gültigen Befehl entspricht, so wird zu VERTEIL verzweigt, andernfalls RTS.

`$0857 VERTEIL` Verteiler

Eingaben: XR (Befehlsnummer), MEMSHF, MODUS, MKFLAG
Temporär: TMP
Ausgaben: AC (Befehlsnummer), Sign (Shift),
Carry (Commodore-Taste)
Routinen: RESETC, RESTA

Da diese Routine zusammen mit den beiden nachfolgenden Tabellen die wichtigste in bezug auf Erweiterungen ist, wollen wir sie etwas genauer durchleuchten.

Zunächst wird geprüft, ob eine Move-Markierung existiert (MKFLAG = 2). Ist dies der Fall, wird sie bei einer Befehlsnummer kleiner 33 gelöscht (RESETC). Es gibt nämlich Befehle, bei denen das Löschen der Markierung nötig ist (z.B. Bildschirmwahl) und solche, bei denen es unerwünscht ist (z.B. Cursortasten). Durch einen Fehler in dieser Abfrage beim alten Hi-Eddi war es möglich, die Markierung durch Fehlbedienung zu *verewigen.*

Anschließend folgt eine Abfrage auf Menü-Modus. Ist er aktiv, wird vom Menü auf den aktuellen Bildschirm zurückgeschaltet (RESTA).

Nun wird die der Befehlsnummer entsprechende Adresse aus ADRTAB geholt und auf den Stack gebracht, AC, Sign und Carry entsprechend obiger Angabe belegt und mittels RTS zur Befehlsausführung verzweigt.

$0885 BEFTAB Tastaturcodes der Befehle

Diese Tabelle enthält in der Reihenfolge der Befehlsnummern die Tastaturcodes der entsprechenden Befehle. Sie ist gemäß der 49 Befehle (Nummer 0 bis 48) 49 Bytes lang. Bei den drei noch unbenutzten Befehlen (siehe ADRTAB) steht als Tastaturcode 99, ein Wert, den es nicht gibt (Tastaturcodes gehen nur bis 64).

$08B6 ADRTAB Adreßtabelle

Diese Tabelle ist das Wichtigste am Hi-Eddi plus, deshalb drucke ich sie hier komplett ab:

```
ADRTAB ;Adressen, Befehlsnr, Taste, Befehl
       .SE CLEAR-1     ; 0 CLR  Loeschen
       .SE TOGGLE-1    ; 1 1    Bildschirmwahl
       .SE TOGGLE-1    ; 2 2    Bildschirmwahl
       .SE TOGGLE-1    ; 3 3    Bildschirmwahl
       .SE TOGGLE-1    ; 4 4    Bildschirmwahl
       .SE TOGGLE-1    ; 5 5    Bildschirmwahl
       .SE TOGGLE-1    ; 6 6    Bildschirmwahl
       .SE TOGGLE-1    ; 7 7    Bildschirmwahl
       .SE HORVER-1    ; 8 H    Hor. Step
       .SE HORVER-1    ; 9 V    Vert. Step
       .SE FBCOL-1     ;10 B    Background
       .SE FBCOL-1     ;11 F    Foreground
       .SE INVERT-1    ;12 I    Invert
       .SE INIGRF-1    ;13 _    Korrektur
       .SE MOVRST-1    ;14 ^    Move Restore
       .SE BLINK-1     ;15 *    Cursor-Blinken
       .SE DRI-1       ;16 D    Draw/Directory
       .SE ZEICH-1     ;17 Z    Zeichensatz
       .SE LDSV-1      ;18 L    Line/Load
       .SE MODI-1      ;19 R    Rectangle
       .SE CMD-1       ;20 C    Circle/Command
       .SE PICPR-1     ;21 P    Paint/Print
       .SE MODI-1      ;22 J    Spray
       .SE SHFMOD-1    ;23 M    Move/Mirror
```

```
.SE WALK-1      ;24 W    Walk
.SE SHFMOD-1    ;25 T    Text/Turn
.SE SPACE-1     ;26 Sp   Menue/Sprite-Editor
.SE LDSV-1      ;27 S    Stamp/Save
.SE MODI-1      ;28 A    Append
.SE MODI-1      ;29 G    Get
.SE MODI-1      ;30 E    Erase
.SE RETAS-1     ;31      Frei
.SE RETAS-1     ;32      Frei
.SE RETAS-1     ;33      Frei
.SE QUIT-1      ;34 Q    Ausschalten
.SE INSDEL-1    ;35 INS/DEL
.SE MEM-1       ;36 F1   Tabulator/Step
.SE MEM-1       ;37 F3   Tabulator/Step
.SE MEM-1       ;38 F5   Tabulator/Step
.SE MEM-1       ;39 F7   Tabulator/Step
.SE RIGLEF-1    ;40 CRSR Hor.
.SE UPDN-1      ;41 CRSR Ver.
.SE SLOW-1      ;42 -
.SE FAST-1      ;43 +
.SE OXU-1       ;44 O    Oder
.SE OXU-1       ;45 X    Exor
.SE OXU-1       ;46 U    Und
.SE RVSON-1     ;47 9
.SE RVSOFF-1    ;48 0
```

Für Erweiterungen lassen sich nun ohne weiteres diese Adreßvektoren verbiegen, wie dies bei den Befehlen 'L', 'S' oder 'R' in der Extension gemacht wurde. Für neue Befehle, wie 'Y', '=' und 'K' in der Extension, sind extra noch drei Plätze frei: die Nummern 31 bis 33. Sie liegen mitten in der Adreßtabelle, um die Grenze, ab der eine Move-Markierung gelöscht werden soll, für diese Befehle verschieben zu können (Speicherstelle $085A). In der jetzigen Stellung wird bei Befehl 33 eine Markierung nicht gelöscht (bei 'K' sinnvoll), bei 31 und 32 dagegen wird sie gelöscht (bei 'Y' und '=' nötig!).

Ferner ist noch darauf zu achten, daß die Adressen in der ADRTAB um eins kleiner sein müssen als die tatsächlichen Zieladressen (daher das '-1'), da sie mit RTS angesprungen werden. Außerdem müssen die Tastaturcodes für neue Befehle in BEFTAB eingetragen werden.

$0918 SHFMOD Move/Mirror, Text/Turn
$091D CBMMOD Disk- und Druckerbefehle/Modusbefehle

Eingaben: AC, Sign, Carry
Routinen: MIRTRN, INISCR, MODI

Diese Routinen dienen dazu, bei einigen mehrfach belegten Tasten zu verschiedenen Befehlen zu verzweigen.

CBMMOD wird von den Disk- und Druckerbefehlen aufgerufen. Wird dabei festgestellt, daß die Commodore-Taste nicht gedrückt war, dann wird die Rückkehradresse zu der aufrufenden Routine vom Stack entfernt (das ist eigentlich nicht in Ordnung!) und zu MODI verzweigt. Andernfalls wird der Lowres-Screen initialisiert und zum aufrufenden Programm zurückgekehrt.

$0929 MODI Modus setzen

Eingaben: AC (Befehlsnummer)
Ausgaben: MODUS, SPRNR ,LRCFL, UOXFL, MKFLAG
Routinen: SPFORM, CHKCOR

Diese Routine wird grundsätzlich bei Eingabe eines Modusbefehls aufgerufen (bei Fore/Back und Menü über Umwege). Dabei wird entsprechend der Befehlsnummer der MODUS gesetzt (die Modusbefehle sind in ADRTAB so plaziert, daß die unteren vier Bits der Befehlsnummer den MODUS ergeben), die drei oben angegebenen Flags werden gelöscht und das Cursor-Sprite wird erstellt (SPFORM) und plaziert (CHKCOR). Die Spritenummer SPRNR, die die Form des Cursor-Sprites bestimmt, sowie die Belegung des Spriteschalter-Registers $D015 werden dabei gemäß dem MODUS aus zwei Tabellen geholt, die dieser Routine unmittelbar folgen.

$0967 INVERT Bildschirm invertieren

Temporär: PTR

Der sichtbare Bildschirm wird invertiert

```
$097E  CLEAR          Bildschirm löschen, HOME
```

Eingaben: Carry (Commodore-Taste)
Ausgaben: SY, SXL, SXH (nur bei HOME)
Routinen: CHKCOR (bei HOME), PRGTOG2 (bei Clear)

Das Löschen des Bildschirms geschieht durch eine Exor-Verknüpfung mit sich selbst.

```
$0991  PRGTOG         Bildschirmwahl vom Programm
$0996  TOGGLE         Bildschirmwahl, Verknüpfung, Copy
```

Eingaben: AC (Bildschirmnummer), Carry, UOXFL (nur TOGGLE), COLFL
Temporär: PTR, PTR2, CNT
Ausgaben: ORGTAB (nicht bei Copy und Verkn.), UOXFL
Routinen: SCORG, SCCORG, SAVCOL, LDCOL, RESCUC

PRGTOG wird beim Ein- und Ausschalten des Menüs, bei Move Restore und am Beginn von Walk aufgerufen. Es handelt sich dabei immer um eine Bildschirmanwahl, ein eventuell gesetztes UOXFL wird gelöscht, um eine Verknüpfung zu verhindern.

TOGGLE wird bei Eingabe einer Nummer aufgerufen. Je nach Carry und UOXFL wird eine Bildschirmwahl, Verknüpfung oder Copy (Carry = 1) ausgeführt. UOXFL wird nach erfolgter Verknüpfung zurückgesetzt, um eine mehrmalige Verknüpfung zu verhindern.

```
$0A0F  OXU            Oder, Exor, Und
```

Eingaben: AC (Befehlsnummer)
Ausgaben: UOXFL

Entsprechend der Befehlsnummer - auch hier wurde die Position in ORGTAB in geeigneter Weise gewählt - wird das UOXFL gesetzt.

$0A1B SCORG Bildschirmorganisation

Eingaben: AC (Bildschirmnummer)
Ausgaben: AC, XR, YR, PTR, PTR2, CNT

PTR2 wird mit der Anfangsadresse der Bitmap geladen, deren Nummer in AC übergeben wurde. Die Position dieser Bitmap (0 = $2000 bis 6 = $E000) wird in AC zurückgegeben. Die übrigen Ausgabeparameter werden für Bitmaptausch, Verknüpfung oder Copy vorbelegt: PTR = $2000, XR = CNT = 32 (Anzahl der Blocks) und YR = 0.

$0A3E FBCOL Farbbefehle

Eingaben: AC (Befehlsnummer), Sign, COLFL (Bit 7)
Temporär: TMP, PTR
Ausgaben: COLFL (Bit 0), CRSR
Routinen: MODI, COLSET

Je nach Befehlsnummer, Sign und Betriebsart wird die Rahmenfarbe weitergeschaltet, der Fore- oder Back-Mode angewählt oder Vorder- oder Hintergrund eingefärbt. Bei Letzterem wird der Cursor auf blinkend geschaltet, da man ihn sonst nicht mehr sehen würde.

$0A75 SETCOL Video-RAM für Schwarzweiß setzen

Eingaben: COLOUR
Temporär: PTR
Routinen: SETSPB

Der gesamte Bildschirm wird gemäß COLOUR eingefärbt. Wird von RESETC bei Schwarzweiß-Betrieb sowie vom Sprite-Editor aufgerufen.

$0A8B RESETC Video-RAM wiederherstellen.

Eingaben: COLFL
Routinen: SETCOL, LDCOL

Beim Einschalten des Hires-Screen (zum Beispiel nach Load, Save) und bei Korrektur oder Löschen einer Move-Markierung dient diese Routine dazu, das Video-RAM ($400) wieder zu belegen. Im Schwarzweiß-Betrieb macht das SETCOL, bei Farbe wird der Video-RAM-Zwischenspeicher (VZWS, $4000) mittels LDCOL ins Video-RAM verschoben.

$0A91 LDCOL Farbinfo ins Video-RAM laden.

Eingaben: AC (High-Byte der Quelladresse)
Temporär: PTR, PTR2
Routinen: MOVUNI

Lädt eine Farbinfo aus dem Speicher ($4000 - $6000) ins Video-RAM ($400).

$0A99 RESCUC Video-RAM retten

Eingaben: COLFL
Routinen: SAVCOL

Vor dem Einschalten des Lowres-Screens oder dem Markieren muß im Farb-Betrieb das Video-RAM in den Video-RAM-Zwischenspeicher gerettet werden. Dies geschieht mittels SAVCOL.

$0A9F SAVCOL Video-RAM in Speicher

Eingaben: AC (High-Byte der Zieladresse)
Temporär: PTR, PTR2
Routinen: MOVUNI

Speichert das Video-RAM in den Speicher für Farbinformationen ($4000 - $6000).

$0AA8 SETSPB Sprite-Block-Pointer setzen.

Ausgaben: COLOUR

Die Sprite-Block-Pointer, die bei Manipulationen am Video-RAM überschrieben werden, werden wieder neu gesetzt. Außerdem wird COLOUR gemäß der Farbe im linken oberen Bildschirmeck gesetzt (für SETCOL und Sprite-Editor).

$0ABB SCCORG Farbinfo-Bildschirmorganisation

Eingaben: AC (Bildschirmnummer)
Ausgaben: AC (High-Byte der Farbinfo-Adresse)

Für Bildschirmwahl und Copy bei Farb-Betrieb. Mit der Ausgabe kann dann LDCOL oder SAVCOL versorgt werden.

$0AC5 MEM Funktionstasten

Eingaben: AC (Befehlsnummer), Sign, Carry
Ausgaben: SY, SXL, SXH, MEMTAB, STPCNT

Es wird entsprechend den unteren beiden Bits der Befehlsnummer entweder die momentane Cursorposition gespeichert (Sign = 1), eine Schrittweite angewählt (Carry = 1) oder der Cursor an eine gespeicherte Stelle gefahren.

$0ADD STORE Cursorposition speichern

Eingaben: XR (F-Tasten-Nummer 0 bis 3)
Ausgaben: MEMTAB

Dies ist eine Einsprungadresse in MEM, die bei Line, Rec, Circle, Move und Zoom angesprungen wird, um die aktuelle Cursorposition zu speichern.

```
$0AED  FAST        Schnelle Cursorgeschwindigkeit (+)
$0AF3  SLOW        Langsame Cursorgeschwindigkeit (-)
```

Ausgaben: SPEED, DISMIN

Hiermit wird die Cursor-Endgeschwindigkeit und die Koala-Pad-Empfindlichkeit eingestellt. Wem der Cursor zu langsam ist, der kann den Wert in $0AF0 (6) auf 4 verkleinern.

```
$0AFC  RVSON       Revers ein
$0B00  RVSOFF      Revers aus
```

Eingaben: MEMSHF (CTRL-Bit)
Ausgaben: INVFL

```
$0B0B  TEXT        Zeichen in Grafikbildschirm drucken
```

Eingaben: YR (ASCII-Code), INVFL, ZFLAG, SRFLAG
Temporär: PTR, BUF, FLAG
Routinen: SCORG, WRITE, TEXTCRSR

Die dem ASCII-Zeichen entsprechende Matrix wird aus ROM oder RAM in BUF geholt und bei Bedarf rotiert. Mittels WRITE wird das Zeichen in den Bildschirm geschrieben und mittels TEXTCRSR der Cursor weiterbewegt.

```
$0B7F  WRITE       Zeichen schreiben oder löschen.
```

Eingaben: BUF, FLAG, SY, SXL, SXH
Temporär: SPRBIT, BITPOS, PTR
Routinen: SPRPIX, CALC, BITCNT

Bei positivem FLAG wird das Zeichen in BUF an der momentanen Cursorposition geschrieben, bei negativem werden dort 8*8 Punkte gelöscht.

$0BBE ZEICH Zeichensatz und Schreibrichtung

Eingaben: Sign, Carry, ORGTAB
Temporär: PTR, PTR2
Routinen: MOVUNI (ist Teil von ZEICH)

Es wird entweder der ROM-Zeichensatz ins RAM kopiert und ZFLAG gesetzt (Sign = 1) oder ZFLAG gelöscht (Carry = 1) oder SRFLAG erhöht.

$0BDF MOVUNI Universelle Speicher-Verschieberoutine

Eingaben: XR, PTR+1, PTR2+1

Diese Routine verschiebt XR Blocks von PTR nach PTR2. Es müssen nur die High-Bytes der Pointer belegt sein, die Low-Bytes setzt MOVUNI auf 0. Außerdem wird am Ende das ROM eingeschaltet.

$0BFE WALK Walk-Befehl, Sequenzstring eingeben

Da man diese Routine für eine Erweiterung wohl kaum aufrufen wird und da sie sehr komplex und lang ist, wird auf eine detaillierte Erklärung hier verzichtet.

$0D56 SPACE Menü-Modus/Sprite-Editor

Eingaben: Sign, COLFL

Es wird entweder zu MENUE oder zu SPEDIT verzweigt.

$$0D61 MENUE Menü-Modus anwählen

Eingaben: ORGTAB, MODUS, SY, SXL, SXH
Ausgaben: MEMSCR, MEMBUF
Routinen: PRGTOG, MENMOD

Momentane Bildschirmnummer, Sprite-Position und MODUS werden gerettet, der Menübildschirm und der Menü-Modus angewählt.

$0D78 RESTA Zustand vor Menüaufruf restaurieren.

Eingaben: MEMSCR, MEMBUF
Ausgaben: MODUS, SY, SXL, SXH
Routinen: PRGTOG, MENMOD

Nach einer Befehlseingabe im Menü wird wieder der Zustand vor Menüaufruf hergestellt. MENMOD ist eine Einsprungadresse in MODI, bei der die Flags UOXFL, LRCFL, MKFLAG nicht beeinflußt werden ($0931).

$0D8B MIRTRN Mirror und Turn

Eingaben: AC (Befehlsnummer), MKFLAG, MKBUF+4 bis MKSPAL
Temporär: PTR, PTR2, FLAG, COUNT bis CNT
Routinen: POSADR, MAL8. DIV8, DREH, RESCUC

Wenn MKFLAG = 3, dann wird der Move-Zielbereich, ansonsten der ganze Bildschirm gedreht oder gespiegelt (AC = 23)

$0E79 DREH Byte drehen

Eingaben: AC
Temporär: WORK, TMP
Ausgaben: AC

Diese kleine Routine zum Umdrehen eines Bytes (Bit 0 nach Bit 7 und umgekehrt) wird von den Mirror- und Turn-Befehlen des Grafik- und des Sprite-Editors gebraucht.

$0E88 HORVER Schrittweiten eingeben

Eingaben: AC (Befehlsnummer), STPCNT, HSTTAB, VSTTAB
Temporär: TMP, NUMBER
Ausgaben: HSTTAB, VSTTAB
Routinen: INISCR, PRINT, PRNUM, INPUTB, INIGRF

Die eingegebene Zahl darf maximal 255 sein. Bei Nichtbeachtung passiert nichts, außer daß Sie einen unsinnigen Wert erhalten. Bei Eingabe einer 0 oder eines nicht-numerischen Ausdrucks wird die alte Schrittweite nicht verändert.

`$0EE0 BLINK` Cursorblinken ein/aus

Eingaben: CRSR
Ausgaben: CRSR

`$0EE7 BLICOL` Cursorfarbe holen

Eingaben: CRSR, CLOCK
Ausgaben: AC (Cursorfarbe)

Blinkt der Cursor (CRSR = 0), so wird die Farbe Schwarz oder Weiß zurückgegeben, andernfalls die Farbe des Rahmens.

`$0EF7 LDSV` Load und Save

Eingaben: AC (Befehlsnummer), COLFL, ORGTAB (für ZFLAG)
Temporär: LSFLAG, STADR, ENDADR, NAMLEN
Ausgaben: ZFLAG
Routinen: CBMMOD, PRINT, INPUTB, DISK2, CCMD2

In der folgenden Tabelle ($0F4B-$0F4E) stehen die vier möglichen Eingaben 'G', 'F', 'S' und 'Z'. Die Tabelle danach ($0F4F-$0F5E) enthält Start- und Endadressen dieser Objekte.

`$0F5F COMMAND` Kommando an Floppy senden

Temporär: FLAG, BUF, ANZTAS
Routinen: CBMMOD, PRINT, INPUTB, PRINT2

Innerhalb dieser Routine werden noch zwei Einsprungadressen benutzt: CCMD2 ($0F6B, AC muß 0 sein) liest den Fehlerkanal und wartet, wenn ein Fehler aufgetreten ist (wird von LDSV benutzt) sowie CCMD3 ($0F6F, AC muß 0 sein), bei der das Programm auch ohne Fehler wartet, wenn FLAG ungleich 0 ist. CCMD3 wird von DIR angesprungen.

Am Ende dieser Routine wird getestet, ob eine Erweiterung geladen wurde (Kennung CBM80 ab $2000). Wenn ja, dann wird sie angesprungen, andernfalls wird mit INIGRF wieder in den Grafik-Editor gesprungen.

`$0FD0 DIR` Directory anzeigen

Temporär: TMP, FLAG
Routinen: CBMMOD, PRNUM, CCMD3

Am Anfang dieser Routine folgt erst eine Abfrage auf SHIFT (Sign) wegen Draw mit dickem Pinsel. 'D' ist, ebenso wie die Funktionstasten, dreifach belegt!

`$1035 DISK2` Load/Save-Unterprogramm

Eingaben: LSFLAG, STADR, ENDADR
Temporär: PTR

Diese Routine wird nicht nur für Load und Save, sondern auch zum Nachladen der Menütafel und für die Overlaytechnik benutzt.

`$1062 PICPR` Druckerroutine nachladen

Temporär: LSFLAG
Routinen: DISK2, DROUT, INIGRF

Nachladen und Aufrufen der Druckerroutine (OVER2), anschließend wieder Laden des Sprite-Editors (OVER1).

`$1085 PRNUM` Zahl drucken

Eingaben: XR (Low), YR (High)
Temporär: MPXL, MPXH, BUF

Die übergebene Zwei-Byte-Zahl wird dezimal, ohne Blanks und führende Nullen, gedruckt. Wird für DIR und HORVER benutzt.

`$10BE INPUTB` Eingabe einer Zeichenfolge

Ausgabe: BUF, NAMLEN, YR

Entspricht etwa dem BASIC-Input-Befehl, jedoch werden Kommas und Doppelpunkte angenommen. Die eingegebene Zeichenfolge wird in BUF abgelegt, mit 0 abgeschlossen und die Anzahl der eingegebenen Zeichen in NAMLEN und YR übergeben. Führende Blanks werden unterdrückt.

`$10E7 JN` `Ja/Nein-Eingabe`

Ausgabe: AC

Bei 'J' wird $FF, bei 'N' 0 zurückgegeben. YR wird zwischengespeichert.

`$110C PRINT` `Texte ausgeben`

Eingabe: YR

Gibt Texte mit der Startadresse MSGS + YR aus, bis eine 0 angetroffen wird (die wird auch ausgegeben!).

`$1116 PRINT2` `Texte aus dem Buffer ausgeben`

Eingabe: YR

Gibt Texte mit der Startadresse BUF + YR, also aus dem Buffer oder dem Sequenzstring aus, bis 0 angetroffen wird. Wird für Ausgabe der Fehlermeldung und des Sequenzstrings benutzt.

`$1120 MSGS` `Texte (Messages)`

`$11B9 JOY` `Joystick-(Pad-, Cursortasten-) Richtungsabfrage`

Eingaben: PADFL, TIME, SPEED
Ausgaben: TIME, MEMSHF
Routinen: RDPORT, VT, HT, DELAY

Dies ist gewissermaßen das Hauptprogramm für die Cursorsteuerung. Es werden die entsprechenden Unterprogramme aufgerufen und durch Herunterzählen von TIME die Beschleunigung des Cursors bei Joysticksteuerung realisiert.

$11F1 RDPORT Joy-Port lesen

Eingaben: DISMIN, MEMAD, TASTE
Ausgaben: PADFL, MEMAD, JPORT, AC
Routinen: DELAY1

Diese Routine stellt fest, ob ein Joystick oder Pad angeschlossen ist, belegt PADFL entsprechend und gibt in AC und JPORT die Bits der einzelnen Bewegungsrichtungen und des Knopfes (einschließlich RETURN-Taste) zurück. Die Bits bedeuten: Bit 0 = Up, Bit 1 = Down, Bit 2 = Left, Bit 3 = Right, Bit 4 = Button. Die Bits sind aktiv high, Bits 5 bis 7 sind 0.

$1262 WAITJY Warten auf Loslassen des Knopfes

Routinen: RDPORT, DELAY

$126F INSDEL Insert/Delete

Eingaben: MODUS, MEMSHF
Routinen: DELCRSR, WRITE, TEXTCRSR

Je nach MEMSHF wird zuerst der Cursor zurückgesetzt (DELCRSR) und dann 8*8 Punkte gelöscht (WRITE) oder nach dem Löschen der Cursor vorwärts bewegt (TEXTCRSR).

$1287 TEXTCRSR Cursor vorwärts
$1289 DELCRSR Cursor rückwärts

Eingaben: SRFLAG, AC (letzteres nur bei DELCRSR)
Ausgaben: MEMSHF

Bei TEXTCRSR wird der Cursor in Schreibrichtung vorwärtsbewegt, bei DELCRSR und AC = 2 zurück.

```
$1290  RIGLEF      Horizontale Cursorbewegung (angewählte Schrittweite)
$1292  HT          Horizontale Cursorbewegung (F1)
```

Eingaben: STPCNT (nur RIGLEF), MEMSHF, SXL, SXH, HSTTAB
Ausgaben: SXL, SXH
Routinen: CHKCOR

Gemäß Bit 0 in MEMSHF fährt der Cursor nach links (Bit 0 = 1) oder rechts.

```
$12BD  UPDN        Vertikale Cursorbewegung (angewählte Schrittweite)
$12BF  VT          Vertikale Cursorbewegung (F1)
```

Eingaben: STPCNT (nur UPDN), MEMSHF, SY, VSTTAB
Ausgaben: SY
Routinen: CHKCOR

Gemäß Bit 0 in MEMSHF fährt der Cursor nach oben (Bit 0 = 1) oder unten.

```
$12D5  CHKCOR      Sprite-Koordinaten testen
```

Eingaben: SY, SXL, SXH
Ausgaben: SY, SXL, SXH
Routinen: PTRLIM, CORSET

Die Sprite-Koordinaten werden, in Abhängigkeit von der Form des Cursorsprites, auf Bereichüberschreitung geprüft und bei Bedarf korrigiert. Anschließend wird das Sprite in CORSET plaziert.

```
$1302  CORSET      Cursorsprite plazieren
```

Eingaben: SY, SXL, SXH
Temporär: TMP

Die Sprite-Koordinaten-Register des VIC werden gemäß SY, SXL und SXH belegt.

`$1324 PTRLIM` Setzen des Pointers auf die Grenzwert-Tabelle

Eingabe: SPRNR
Ausgabe: XR

Entsprechend SPRNR wird der Pointer auf die für dieses Sprite gültigen Werte in der Tabelle der Grenzwerte (LIMTAB) gesetzt.

`$132A LIMTAB` Tabelle der Grenzwerte

4 mal 4 Bytes für die vier möglichen Cursorsprites, jeweils untere und obere SY- sowie untere und obere SXL-Grenze.

`$133A BUTTON` Knopfabfrage und Verteiler

Eingaben: JPORT, MODUS
Routinen: SPRPIX

Wurde der Feuerknopf oder die RETURN-Taste gedrückt (Bit 4 in JPORT = 1), dann werden die Sprite-Koordinaten SY, SXL, SXH in Pixelkoordinaten PY, PXL und PXH umgerechnet; danach wird entsprechend MODUS zur Befehlsausführung verzweigt. Ebenso wie bei der Tastaturabfrage und dem dazugehörigen Verteiler (VERTEIL), gibt es auch hier einen Verteiler, der die Adresse aus einer Adressentabelle (ADRTB3) liest und per RTS anspringt. Nachfolgend die Tabelle, die bei $1350 beginnt und natürlich ebenfalls einen Ansatzpunkt für Erweiterungen darstellt:

```
ADRTB3   ;Adresse, Modusnummer, Modus
       .SE PLOT-1      ; 0   Draw
       .SE DDRAW-1     ; 1   Draw dick
       .SE LNBUT-1     ; 2   Line
       .SE RCBUT-1     ; 3   Rec
       .SE CIBUT-1     ; 4   Circle
       .SE PABUT-1     ; 5   Paint
       .SE JOTBUT-1    ; 6   Spray
       .SE MOVBUT-1    ; 7   Move
       .SE COLBUT-1    ; 8   Fore/Back
       .SE TEXTBUT-1   ; 9   Menü/Sprite-Editor
       .SE MENBUT-1    ;10   Menü
       .SE STMBUT-1    ;11   Stamp
       .SE APPBUT-1    ;12   Append
       .SE GETBUT-1    ;13   Get
       .SE ERSBUT-1    ;14   Erase
```

Zu TEXTBUT: Im Textmodus bewirkt ein Knopfdruck die Anwahl von Menü oder Sprite-Editor. TEXTBUT ist deshalb eine Einsprungadresse in SPACE, nämlich $0D58.

$136E PLOT Punkt setzen oder löschen

Eingaben: SHIFT, PX (steht für PXL und PXH), PY
Temporär: PTR
Routinen: CALC

Je nach SHIFT wird unter Verwendung von CALC der Punkt PY, PXL, PXH gelöscht oder gesetzt (Bit 0 von SHIFT = 0).

$1383 DDRAW 3*3 Punkte setzen oder löschen

Eingaben: PY, PX
Temporär: MCNT
Routinen: PLOT

Es werden 3*3 Punkte gesetzt oder gelöscht (je nach SHIFT), wobei die Koordinaten PY, PXL, PXH den linken oberen Punkt kennzeichnen. Somit darf PY nur Werte von 0 bis 197, PX Werte von 0 bis 317 annehmen (wird von CHKCOR überwacht). PY und PX werden von DDRAW verändert!

$13A2 STMBUT Stamp
$13A5 APPBUT Append
$13A9 ERSBUT Erase

Eingaben: SP0, PY, PX
Temporär: PTR, FLAG, SPRBIT, BUF
Routinen: CALC, BITCNT

Je nach Einsprungadresse wird ein spritegroßes Feld gelöscht und/oder das Sprite an der Cursorposition gedruckt. Wie bei DDRAW gibt PY, PX auch hier die linke obere Ecke an und wird verändert.

$13F8 GETBUT Get

Eingaben: PY, PX
Temporär: PTR, SPRBIT, BITPOS
Ausgaben: SP0, MODUS
Routinen: CALC, BITCNT, WAITJY

Das spritegroße Feld, ausgehend von PY, PX, wird in SP0 kopiert, anschließend der Append-Modus eingeschaltet. PX, PY werden verändert.

$1430 CALC Pixelkoordinaten in Adresse umrechnen

Eingaben: PY, PX
Temporär: ZWS
Ausgaben: PTR, BITPOS, AC

Die wohl am häufigsten benutzte Routine in jedem Grafikprogramm. Zu den Pixelkoordinaten PY, PXL, PXH (bleiben unverändert) wird die Adresse in der sichtbaren Bitmap ($2000) berechnet und in PTR übergeben, die Bitmaske, die die Position des Pixels innerhalb dieses Bytes angibt, steht in AC und BITPOS. Das Register XR bleibt unberührt.

Man könnte diese Routine vielleicht etwas schneller machen, aber wohl kaum kürzer. Immerhin braucht sie keine Tabellen, wie viele andere Ausführungen dieser Routine.

$1478 BITCNT Ein Pixel nach rechts

Eingaben: BITPOS, YR
Ausgaben: BITPOS, YR

Die Bitmaske BITPOS wird um eine Position nach rechts verschoben. Geschieht dies von Bit 0 nach Bit 7, dann wird YR, das in den aufrufenden Routinen als Bitmap-Pointer dient, um 8 erhöht (ergo begrenzter Bereich!).

$1483 SPRPIX Sprite- in Pixelkoordinaten umrechnen

Eingaben: SY, SXL, SXH
Ausgaben: PY, PXL, PXH
Routinen: PTRLIM

Unter Berücksichtigung der Form des Cursorsprites (PTRLIM) werden die Pixelkoordinaten anhand der Spritekoordinaten berechnet.

$149D MEMPIX 1. Knopfdruck bei 'L', 'R', 'C' merken
$14A8 MEMMOV 1. Knopfdruck bei Move merken

Eingaben: LRCFL (nur MEMPIX), PY, PXL, PXH
Ausgaben: LRCFL (nur MEMPIX), MPY, MPXL, MPXH
Routinen: WAITJY, STORE

Beim ersten Knopfdruck wird bei MEMPIX das LRCFL gesetzt, beim zweiten wird es zurückgesetzt. Außerdem wird bei beiden Einsprungadressen beim ersten Knopfdruck ein Merkcursor eingeschaltet, die momentane Cursorposition gespeichert (STORE), die Pixelkoordinaten PY, PX nach MPY, MPX übernommen und die Rückkehradresse zum aufrufenden Programm vom Stack geholt (!). Beim zweiten Knopfdruck wird lediglich der Merkcursor wieder ausgeschaltet.

$14D7 LNBUT Linie zeichnen
$14DA LINE Linie zeichnen

Eingaben: PY, PX, MPY, MPX
Temporär: CNT, BUF
Routinen: MEMPIX (nur LNBUT), PLOT, XCOUN2, YCOUN2

LNBUT muß zweimal aufgerufen werden, wobei die Koordinaten jeweils in PY, PX zu übergeben sind (siehe auch MEMPIX). Bei LINE wird dagegen bei jedem Aufruf eine Linie zwischen PY, PX und MPY, MPX gezeichnet (von RCBUT verwendet). Alle Koordinaten bleiben unverändert.

`$1579 RCBUT` Rechteck zeichnen

Eingaben: PY, PX, MPY, MPX
Routinen: MEMPIX, LINE

Muß wie LNBUT zweimal aufgerufen werden, die Koordinaten jeweils in PY, PX (werden verändert).

`$15B9 CIBUT` Kreis zeichnen

Eingaben: PY, PX, MPY, MPX
Temporär: WORK, FLAG, PTR, PTR2, CNT, TMP, BUF
Routinen: MEMPIX, PLOT, XCOUNT, XCOUN2, YCOUNT, YCOUN2

Zweimaliger Aufruf wie LNBUT. Die Mittelpunktskoordinaten (MPY, MPX) bleiben unverändert, die Randkoordinaten (PY, PX) nur, wenn der Kreis nicht am Rand anstößt. Neben den oben angegebenen Routinen benutzt CIBUT noch drei weitere ($1630, $165F, $16AB), die ich jedoch nicht aufführe, da sie wohl kaum anderweitig verwendbar sind. Ihre temporären Parameter sind oben bereits mit abgegeben.

`$16CF YCOUNT` PY inkrementieren/dekrementieren
`$16D1 YCOUN2` wie oben

Eingaben: FLAG (bei YCOUNT), AC (bei YCOUN2) ,PY
Ausgaben: PY, Carry

Bit 1 von FLAG bzw. AC gibt an, ob PY dekremetiert (Bit 1 = 1) oder inkrementiert wird. Beim Anstoßen an den Bildschirmrand wird der Befehl nicht ausgeführt, sondern das Carry gelöscht, das bei odnungsgemäßer Ausführung gesetzt ist.

```
$16E7  XCOUNT      PX inkrementieren/dekrementieren
$16E9  XCOUN2      wie oben
```

Eingaben: FLAG (XCOUNT), AC (XCOUN2) ,PX
Ausgaben: PX, Carry

Bit 0 von FLAG bzw. AC gibt an, ob PY dekrementiert (Bit 1 = 1) oder inkrementiert wird. Beim Anstoßen an den Bildschirmrand wird der Befehl nicht ausgeführt, sondern das Carry gelöscht, das bei odnungsgemäßer Ausführung gesetzt ist.

```
$1710  JOTBUT      Spray
```

Eingaben: PY, PX, CLOCK
Temporär: MPY, MPX, COUNT, TMP
Routinen: PLOT

Pro Aufruf werden 10 Punkte zufällig (CLOCK und $D012 = Rasterzeile dienen als Zufallsgeneratoren) gesetzt oder gelöscht. PY, PX bleiben unverändert.

```
$1770  PABUT       Paint
```

Eingaben: PY, PX, SHIFT
Temporär: MEMSHF, PTR, PTR2, MPY, MPX, TMP, FLAG, BITPOS
Routinen: CALC, XCOUNT, WAIT

Die durch PY+1 und PX+1 (!) markierte Fläche wird ausgefüllt oder gerastert (SHIFT Bit 0 = 1). Bezüglich der Grenzen von PY, PX gilt das gleiche wie bei DDRAW. PY, PX werden verändert.

```
$1852  MOVBUT      Move, Bereiche verknüpfen
```

Eingaben: MKBUF bis MKFLAG, PY, PX, MKSCR, UOXFL, ORGTAB, COLOUR, COLFL
Temporär: PTR, PTR2, FLAG, ZEILE, SPALTE
Ausgaben: MKBUF bis MKFLAG, MKSCR, UOXFL
Routinen: WAITJY, MEMMOV, RESCUC, RESETC, SCCORG

Wie die vielen Eingabeparameter schon erahnen lassen, handelt es sich hier um eine äußerst komplexe Routine. Dennoch ist die Benutzung dieser Routine recht einfach: Der wichtigste Parameter, der alles organisiert, ist MKFLAG. Vor dem ersten Aufruf der Routine muß er 0 oder 3 sein, vor dem zweiten 1 und vor dem dritten 2. Da MOVBUT diesen Parameter selbst weiterzählt, muß man ihn nur einmal initialisieren (was normalerweise in MODI getan wird). Außerdem müssen natürlich bei jedem Aufruf in PY, PX die gewünschten Koordinaten stehen, vor dem dritten Aufruf muß in UOXFL die gewünschte Verknüpfung stehen (0 = keine Verknüpfung, siehe Zeropagebelegung). COLOUR ist lediglich für die Markierungsfarben zuständig und COLFL wird natürlich wegen Farbe oder Schwarzweiß abgefragt. Alle anderen Parameter, so der gesamte MKBUF-Bereich oder MKSCR, belegt sich die Routine bei den einzelnen Aufrufen selbst, so daß man sich nicht darum kümmern muß.

Die genaue Aufschlüsselung des MKBUF-Bereichs, die zum Beispiel für Mirror und Turn interessant ist, finden Sie bei der Zeropagebelegung.

Ebenso wie CIBUT (und übrigens auch JOTBUT und PABUT) benutzt MOVBUT neben den oben angegebenen noch einige "interne" Subroutinen, von denen ich POSADR, MAL8 und DIV8 gesondert erklären werde, da sie auch von anderen Routinen aufgerufen werden.

$18A6 MOVRST Move Restore (Recall)

Eingaben: MKSCR
Ausgaben: MKFLAG
Routinen: PRGTOG, MODI, MARK

Sofern bereits ein Quellbereich markiert wurde (andernfalls ist MKSCR negativ), wird MKFLAG auf 2 gesetzt sowie der Move-Modus und der Quellbildschirm (MKSCR) angewählt. MARK ($18B6) ist eine Einsprungadresse in MOVBUT, dort wird der Quellbereich markiert.

$19E9 POSADR Position in Video-RAM-Adresse umrechnen

Eingaben: AC (Zeile), YR (Spalte)
Temporär: ZEILE, SPALTE
Ausgaben: PTR

Eine Position entsprechend dem Lowres-Bildschirm - also Zeile von 0 bis 24 und Spalte von 0 bis 39 - wird in die entsprechende Video-RAM-Adresse (PTR) umgerechnet. Die eingegebenen Werte stehen unverändert in ZEILE und SPALTE.

$1A30 MAL8 Multiplikation mit 8
$1A3E DIV8 Division durch 8

Eingaben: PTR, PTR2
Ausgaben: PTR, PTR2

$1A4C COLBUT Fore/Back

Eingaben: PY, PX, SHIFT, COLFL
Temporär: PTR, PTR2, TMP
Routinen: CALC, DIV8

Je nach Bit 0 von COLFL wird der Vorder- oder Hintergrund des durch PY, PX markierten Feldes eingefärbt. SHIFT gibt an, ob dazu die Rahmenfarbe (SHIFT Bit 0 = 0) oder der Video-RAM-Zwischenspeicher ($4000) benutzt wird.

$1A62 COLSET Farbe in einem Feld setzen

Eingaben: PTR, Carry, COLFL, (PTR2)
Temporär: TMP

Dies ist eine Einsprungadresse in COLBUT, die von den Farbbefehlen total Fore und total Back (FBCOL) angewandt wird. Wie bei COLBUT wird anhand von Bit 0 in COLFL entweder der Vorder- oder Hintergrund eines Feldes eingefärbt, wobei die Adresse dieses Feldes in PTR übergeben werden muß. Bei Carry = 0 wird dazu die Rahmenfarbe herangezogen, bei Carry = 1 (wird im Programm nicht benutzt) müßte die Quelle in PTR2 stehen.

$1A97 MENBUT Knopfdruck im Menü auswerten

Eingaben: PY, PX
Temporär: TMP, PTR
Ausgaben: XR, MEMSHF
Routinen: WAITJY, BEFEHL

Wird im Menü der Knopf gedrückt, so holt diese Routine anhand von PY, PX den Tastaturcode und den Shift-Code aus der Steuerzeile des Menüs (ab $3E00) und übergibt sie in XR und MEMSHF an die Routine zur Befehlsausführung (BEFEHL).

$1B0A COLD2 Kaltstart

Temporär: TMP, PTR
Ausgaben: COLFL, MEMSCR, ORGTAB
Routinen: CPICPR, INISCR, JN, PRINT, DISK2, MENINIT, INIGRF

Zunächst wird der Sprite-Editor von Diskette nachgeladen, dann werden die Betriebsart-Fragen gestellt und die Bitmaps und Arbeitsspeicherbereiche initialisiert und eventuell die Menütafel nachgeladen.

$1B9E INIGRF Grafik-Editor initialisieren

Eingaben: MODUS
Routinen: RESETC, MODI, WAIT

Diese Routine wird sowohl zum Einschalten des Grafik-Editors nach einer Eingabe auf dem Lowres-Bildschirm als auch zur Korrektur (Pfeil nach links) verwendet. Sie schaltet den Hires-Modus ein, stellt das Video-RAM her und löscht damit alle Farbveränderungen oder Markierungen seit dem letzten Aufruf von RESCUC. Auch Merkcursor werden abgeschaltet und die Flags LRCFL, MKFLAG, UOXFL (in MODI) gelöscht.

$1BBF INISCR Lowres-Screen initialisieren

Routinen: RESCUC, WAIT

Dies ist das Gegenstück zur vorherigen Routine. Das Video-RAM wird gerettet, alle Sprites ausgeschaltet und auf einen gelöschten Lowres-Screen umgeschaltet.

```
$1BE6  WAIT        Warten auf Loslassen einer Taste
```

Eingaben: TASTE
Ausgaben: ANZTAS (wird gelöscht)

```
$1BF1  CLSP0       Sprite SP0 wird gelöscht
```

Ausgaben: SP0

```
$1BFC  DELAY1      Feste Verzögerung
$1BFE  DELAY       Frei wählbare Verzögerung
```

Eingaben: AC (nur DELAY)

```
$1C0C  SPFORM      Formen der Cursorsprites erstellen
```

Eingaben: AC
Temporär: PTR
Ausgaben: SPRA

Gemäß der in AC übergebenen Spritenummer (0 bis 4, siehe SPRNR) wird das Cursorsprite erstellt.

```
$1C7B
```

Bei dieser Adresse beginnt der Overlaybereich. Im Einschaltzustand befindet sich hier die Routine zum Zeichnen des Titelbildes, danach wird hierher der Sprite-Editor geladen (COLD2) und beim Print-Befehl beginnt hier die Druckerroutine.

Beginnen wir mit dem Sprite-Editor:

```
$1C7B  JMP INIGRF
$1C7E  SPEDIT      Sprite-Editor / Zoom-Funktion
```

Der Sprite-Editor beginnt erst bei $1C7E, der JMP-Befehl davor dient dazu, bei einem mißlungenen Versuch, die Druckerroutine zu laden (weil zum Beispiel die falsche Diskette im Laufwerk war), wieder in die Hauptschleife zurückzukehren. Eine entsprechende Sicherung ist auch in der Druckerroutine, falls bei Verlassen der Druckerroutine der Sprite-Editor nicht nachgeladen werden konnte. Das würde jedoch heißen, daß jemand

während eines Druckvorgangs die Programmdiskette aus dem Laufwerk nehmen müßte. Merke: Besonders Laien kommen auf die unmöglichsten Ideen, wenn es darum geht, ein Programm falsch zu bedienen!

Kommen wir nun zu der aufgeschlüsselten Beschreibung des Sprite-Editors.

```
$1C7E  SPEDIT        Sprite-Editor, Zoom-Funktion: Initialisierung
```

Eingaben: MODUS, COLOUR, SP0, SY, SX
Temporär: BUF, PTR, PTR2, COLTAB,
Ausgaben: PY, PXL, DISMIN, SLIDE, WNDFLAG
Routinen: STORE, CHKCOR, SPRPIX, GETBUT, RESCUC, SETCOL, LNIB, SPFORM, CHKCOR, SPRWND, WAIT

Was bei der Initialisierung alles geschieht, erkennen Sie schon, wenn Sie die Liste der Parameter und Routinen durchsehen. Deshalb nur noch einige zusätzliche Hinweise: Je nach Cursorstellung (SY) wird das Sprite-Fenster links oder rechts plaziert, damit der Grafikbildschirm nicht so weit verschoben werden muß. Dies wird er in jedem Fall, jedoch nur in der Zoom-Funktion bleibt das Übersichtsfenster neben dem Sprite-Fenster frei. Dort ist das Sprite selbst auf einem zweiten, voll ausgefüllten "Hintergrund-Sprite" zu sehen, das den Grafikbildschirm abdecken soll. Denn erst bei Verlassen der Zoom-Funktion wird der Sprite-Inhalt in den Grafikbildschirm eingesetzt.

In der Zoom-Funktion wird der Inhalt von SP0 in BUF, genauer gesagt ab BUF+3, zwischengespeichert. STMBUT, das vor Verlassen der Zoom-Funktion aufgerufen wird, benutzt nur die ersten drei Bytes von BUF.

DISMIN wird für den Sprite-Editor neu belegt, der aktuelle Wert auf dem Stack zwischengespeichert.

Die kunstvollen Operationen mit COLTAB sind für die Grid-Funktion nötig, weil der Hires-Modus eingeschaltet bleibt. Der Bildschirm wird vom Sprite-Fenster nur wegen der gleichen Farben für Vorder- und Hintergrund scheinbar überdeckt.

$1D48 SPMAIN Sprite-Editor Hauptprogramm

Eingaben: TASTE
Routinen: SPTAST, SPJOY, SPBUT, SPBLINK

Wie schon im Grafik-Editor ist auch hier das Hauptprogramm sehr kurz: 18 Bytes. Diese Schleife wird verlassen, wenn die Space-Taste gedrückt ist.

$1D5A Rückkehr zum Grafik-Editor

Eingaben: BUF, SLIDE
Temporär: PTR, PTR2
Ausgaben: MODUS, DISMON, SP0, SPX
Routinen: SPRPIX, STMBUT, LPUNI, SETTAB, INIGRF

Hier wird alles für die Rückkehr in den Grafik-Editor vorbereitet: Bitmap zurückverschieben, Sprites plazieren, bei Zoom Sprite-Inhalt auf Bildschirm drucken und SP0 aus BUF zurückholen, MODUS und DISMIN wieder wie vorher belegen.

$1D8F SPTAST Tastaturabfrage und Befehlsauswertung

Eingaben: TASTE
Ausgaben: AC

Da es hier im Vergleich zum Grafik-Editor nur sehr wenig Befehle gibt, hätte sich ein Verteiler nicht gelohnt. Deshalb wird ganz einfach zu Beginn eines jeden Befehls verglichen, ob der in AC gelieferte Tastaturcode mit dem für diesen Befehl zutreffenden Code übereinstimmt. Wenn nicht, wird zum nächsten Befehl weiterverzweigt.

$1D95 MIRROR Mirror

Eingaben: SP0
Temporär: SPH
Ausgaben: SP0
Routinen: DREH, MOVE

`$1DC0 TURN` Turn

Parameter und Routinen wie MIRROR

`$1DD8 ROTATE` Rotate

Eingaben: SP0
Temporär: SPH, MCNT, CNT, SPHPTR, BITS
Routinen: MOVE

`$1E20 GRID` Grid

Eingaben: GRIDFL
Ausgaben: GRIDFL
Routinen: ZOOM

`$1E2D BOCOL` Rahmenfarbe weiterschalten

Routinen: SPRWND

`$1E37 TSBLINK` Cursorblinken ein/aus

Routinen: BLINK

`$1E3E EXCHG` Sprite wechseln

Eingaben: SP0, SPX
Ausgaben: SPX, SP0
Routinen: ZOOM

`$1E57 SPUOX` Sprites verknüpfen

Eingaben: SP0, SPX
Ausgaben: SP0
Routinen: ZOOM

`$1E7D SPINV` Sprite invertieren

Eingaben: SP0
Ausgaben: SP0
Routinen: ZOOM

`$1E91 SPECLR` Sprite löschen

Eingaben: SHIFT
Ausgaben: SP0
Routinen: CLSP0, ZOOM

`$1EA2 SPCRSR` Cursortasten

Eingaben: SHIFT
Ausgaben: AC (Richtungsinformation gemäß JPORT, für JOYTAS)
Routinen: JOYTAS

`$1ED2 SPJOY` Joystick-Richtungsabfrage
`$1ED9 JOYTAS` Einsprungadresse für Cursortastensteuerung

Eingaben: AC (nur JOYTAS), PY, PXL, PADFL
Ausgaben: PY, PXL
Routinen: RDPORT (nur SPJOY), DELAY, ZOOM

PY und PXL dienen als Koordinaten im Sprite-Editor. Sie werden gemäß AC (aus RDPORT oder SPCRSR) verändert, jedoch gleichzeitig auf Bereichsüberschreitung hin geprüft.

Am Ende erfolgt eine Verzögerung je nachdem, ob ein Joystick oder ein Pad angeschlossen ist. Nebeneffekt dieser "Sparschaltung": Bei angeschlossenem Joystick ist die Cursortasten-Eingabe etwas zäh.

$1F05 SPBUT Knopfabfrage

Eingaben: JPORT, SHIFT, PXL, PY
Temporär: BITPOS
Ausgaben: SP0
Routinen: MASKE

MASKE ($1469) ist eine Einsprungadresse in CALC, bei der nur die Bitmaske berechnet wird.

$1F33 SPRWND Bildschirm für Sprite-Editor initialisieren

Eingaben: WNDFLAG, MODUS
Temporär: PTR, SPALTE
Routinen: LNIB, ZOOM

Gemäß WNDFLAG wird der linke oder rechte Teil des Grafikbildschirms für das Sprite-Fenster abgedeckt. Beim Sprite-Editor (also nicht Zoom) wird der ganze Bildschirm abgedeckt.

$1F68 ZOOM Sprite zoomen

Eingaben: WNDFLAG, SP0, GRIDFL, COLTAB, PY, PXL
Temporär: CNT, PTR, SPH
Routinen: POSADR, BLICOL, LNIB

Das Sprite SP0 wird ins Sprite-Fenster gezeichnet und der Cursor an der Stelle PY, PXL plaziert.

$1FB4 SPBLINK Cursorfarbe setzen

Eingaben: PTR, WNDFLAG
Routinen: BLICOL, LNIB

Das ist eine Einsprungadresse in ZOOM, die vom Hauptprogramm benutzt wird, um die Cursorfarbe zu aktualisieren.

$1FC2 LNIB Low Nibble duplizieren

Eingaben: AC
Temporär: TMP
Ausgaben: AC

$1FCD COLCONV Farbtabelle

Dies ist eine 16 Byte lange Tabelle kontrastschwacher Farben für die Grid-Funktion. Zum Beispiel: Ein gelber Hintergrund wird mit Hellgrau durchsetzt.

$1FDD QUIT Hi-Eddi plus beenden

Eingaben: Carry (C=-Taste)
Ausgaben: Ein jungfräulicher C64

5.3 Dokumentation der Druckerroutine

Die Druckerroutine liegt unter der Bezeichnung **OVER2** auf der Diskette. Sie wird vom Programm **DRUCKER** erzeugt, indem das File **EPSON** oder **VC1525** in das File **OVER2** kopiert wird. Beim File **EPSON** hängt das Drucker-Anpaßprogramm noch einen Satz Daten an.

Da vor allem die Anpassung an Epson-ähnliche Drucker interessant ist, beschränke ich mich bei der Dokumentation auf die Epson-Routine.

Die Epson-Routine benutzt folgende Hi-Eddi Speicherzellen:

PTR, PTR2, FLAG, ZEILE, SPALTE, WORK, COUNT, CNT, ORGTAB, BUF

und folgende Routinen:

CPICPR, INPUTB, JN.

Dazu einige Erklärungen:

FLAG gibt den Druckmodus an:

```
BIT7 = 0:  normal groß
     = 1:  doppelt groß
BIT6 = 0:  ein Bild
     = 1:  zwei Bilder nebeneinander
BIT5 = 0:  CRT-Grafik
     = 1:  PLOT-Modus
BIT4 = 0:  Single Strike
     = 1:  Double Strike
```

COUNT dient beim Ausdruck von großen Bildern als Flag für die erste (COUNT=4) bzw. zweite (COUNT=8) Halbzeile.

WORK ist ein Flag für den aktuellen Bus: $00 = seriell, $80 = Userport.

CPICPR ($1075) ist die Rücksprungadresse ins Hauptprogramm; dort wird wieder der Sprite-Editor (OVER1) geladen.

Nun zur Gliederung der Druckerroutine:

```
$1C7B  DROUT       Druckerroutine Hauptprogramm
$1C7F              Eingaben, FLAG belegen
$1CDF              Bus initialisieren
$1CE4              PTRs auf Bitmaps berechnen
$1D02              Byte-Orientierung festlegen
$1D11              Zeilenvorschub für Single Strike
$1D20  LPPRB       Ausdruck-Hauptschleife
$1D50              PTRs auf nächste Bitmapzeile
$1D63              Abfrage auf STOP-Taste
$1D6E              Ende der Hauptschleife
$1D76              Eingabe der Bildunterschrift
$1D84              Bildunterschrift drucken
$1DB7              Druckerfile schließen
$1DBF              Frage "NOCHMAL"
$1DCC              Ende des Hauptprogramms
```

```
$1DCF  PRLINE      Ausdrucken einer Druckzeile
```

Eingaben: FLAG, COUNT, PTR, PTR2
Temporär: BUF, SPALTE, CNT

Es wird eine Druckzeile ausgegeben, einschließlich der führenden ESC-Grafik-Sequenz und evtl. führender Blanks, jedoch ohne abschließendes CR. Für Double Strike muß diese Routine zweimal aufgerufen werden. Alle Eingabevariablen bleiben unverändert.

```
$1E88  BUSINIT     Bus initialisieren
```

Ausgaben: Carry, WORK

Durch Ausgabe einer 0 wird getestet, ob am Userport ein Drucker reagiert. Ist dies nicht der Fall, so wird ein File auf den seriellen Bus eröffnet. Reagiert dort auch kein Gerät, wird das Carry-Flag als Fehlermeldung gesetzt.

`$1EE7 BYTOUT` Byte auf Bus ausgeben

Eingaben: AC, WORK

Je nach WORK wird das in AC übergebene Byte auf den seriellen Bus oder den Userport ausgegeben.

`$1F08 SEQUENZ` CR oder ESC-Sequenz ausgeben

Eingaben: YR

Der Offset der auszugebenden Sequenz in der Tabelle SEQ muß in YR übergeben werden.

`$1F16 PRINT` Meldung ausgeben

Eingaben: YR

Ausgabe einer Meldung aus der Tabelle MSGS.

`$1F20 OPTAB` ROL und ROR

Je nachdem, ob der Drucker das LSB auf die unterste oder oberste Nadel legt, wird vom Hauptprogramm aus der Befehl ROL oder ROR in PRLINE geschrieben (dies ist wieder die bereits angeführte Selbstmodifizierung!).

`$1F22 MSGS` Texte (Messages)
`$1FA4 BYTETAB` Anzahl der Bytes/Druckzeile

Diese Tabelle enthält die Anzahl der auszudruckenden Bytes (Low/High-Darstellung) für die einzelnen Druckmodi in der folgenden Reihenfolge:

Ein Bild, klein, CRT
Ein Bild, klein, PLOT
Zwei Bilder, klein, CRT
Zwei Bilder, klein, PLOT
Ein Bild, groß, CRT
Ein Bild, groß, PLOT

Um einen breiten Drucker, z. B. EPSON FX-100, im PLOT-Modus voll auszunutzen, kann man die Zahlen entsprechend ändern (die ersten beiden 320, sonst 640).

```
$1FB0  SPALTAB      Anzahl Bildschirmspalten
```

Analog zu BYTETAB steht hier die Anzahl der auszudruckenden Bildschirmspalten (bei zwei Bildern für den zweiten Bildschirm). Für einen breiten Drucker müssen alle 6 Werte 40 sein.

```
$1FB6  Ende des Files "EPSON"
```

Die Daten ab dieser Adresse fügt das Druckeranpaßprogramm an. Die Sequenzen ab der Adresse $1FBC müssen mit $FF auf die Länge von jeweils 6 Bytes aufgefüllt sein.

```
$1FB6  Primäradresse
$1FB7  Sekundäradresse
$1FB8  Plotflag (=0: kein Plotmodus)
$1FB9  0 = LSB unten, 1 = LSB obere Nadel
$1FBA  Anzahl führender Blanks bei CRT
$1FBB  dito bei PLOT
$1FBC  Carriage-Return Line-Feed
$1FC2  CRT-Grafik (ohne Byte-Anzahlen!)
$1FC8  PLOT-Grafik
$1FCE  Single-Strike-Linespacing
$1FD4  Double-Strike-Linespacing
$1FDA  Dot-Linespacing
$1FE0  Text-Linespacing normal
```

5.4 Erweiterungen einbinden

Nachdem Sie nun den Aufbau und die Routinen des Hi-Eddi plus kennen, sind Sie in der Lage, selbst Erweiterungen zu schreiben. Was noch fehlt, ist eine Erläuterung der Technik, wie man diese Erweiterung einbindet. Diese Erläuterung werde ich Ihnen nun anhand der Erweiterung EXT geben. Auf die Erweiterung EXT selbst werde ich nicht näher eingehen, da es sich dabei um wenige, dafür aber recht komplexe Routinen handelt, die man für eigene Erweiterungen kaum verwenden kann.

Nach dem Laden eines Bildes testet die Load-Routine des Hi-Eddi plus, ob in den ersten fünf Bytes dieses *Bildes* die Kennung **CBM80** liegt. Ist dies der Fall, so weiß Hi-Eddi plus, daß eine Erweiterung vorliegt und springt nach $2005, also unmittelbar hinter die Kennung.

In EXT wird nun als erstes der Hires-Bildschirm wieder eingeschaltet (INIGRF); man könnte hier aber auch eine Einschaltmeldung der Erweiterung ausgeben. Zuvor müßte dann aber geprüft werden, ob nicht schon eine Erweiterung im Speicher vorhanden ist. Ist dies der Fall, so ist Bit 7 in ORGTAB+2 gesetzt. In EXT wird dann sofort per RTS ins Hauptprogramm zurückgesprungen.

Als nächstes wird geprüft, ob in Menübetriebsart die Erweiterung in Bildschirm 1 geladen wurde (dann wird die Menübetriebsart ausgeschaltet, indem Bit 6 in COLFL gelöscht wird) und ob die Erweiterung in die *Leinwand* geladen wurde, (dann wird der Walk-Befehl blockiert, indem der Tastaturcode für Walk in BEFTAB mit 99 überschrieben wird).

Im nächsten Abschnitt geht es um den kompliziertesten Vorgang bei der Initialisierung der Erweiterung: Die Erweiterung wird aus dem sichtbaren Bildschirm ($2000) in den für sie vorgesehenen Speicherbereich ($6000) verschoben. Zugleich muß das ab $6000 liegende Bild in den sichtbaren Speicher geschoben werden, damit es nicht durch die Erweiterung überschrieben wird. Dies alles übernimmt die Routine PRGTOG, die ja zum Austausch von Bildschirmen vorgesehen ist. Es wird also der Bildschirm im $6000er-Bereich angewählt (seine Nummer liegt in ORGTAB+2), womit automatisch die Erweiterung an die richtige Stelle gelangt. Allerdings kann PRGTOG nicht einfach mit JSR aufgerufen werden, da ja das aufrufende Programm *wandert* und der Rücksprung damit ins Leere gehen würde. Es wird deshalb zuerst die neue Rückkehradresse auf den Stack gebracht und dann PRGTOG mittels JMP aufgerufen.

Nun liegt die Erweiterung an ihrem Platz und die Initialisierung kann abgeschlossen werden. Dazu wird zunächst das Bit 7 in ORGTAB+2 gesetzt, wodurch verhindert wird, daß eine zweite Erweiterung nachgeladen werden kann. Außerdem ist der Bildschirmspeicher, in dem jetzt die Erweiterung liegt, damit blockiert.

Danach werden die Vektoren in der ADRTAB verbogen, was in EXT gemeinsam für alle Befehle der Erweiterung in tabellarischer Form von einem Unterprogramm ($60C2) erledigt wird.

Ab $6040 werden in EXT die Initialisierungsblöcke für die einzelnen Routinen der Erweiterung durchlaufen, in denen die Tastaturcodes der neuen Befehle in BEFTAB geschrieben, die Arbeitsspeicherbereiche der Erweiterung initialisiert und einige weitere Abzweigungen in Hi-Eddi plus eingefügt werden.

Bei $6087 ist die Initialisierung beendet und es wird mittels RTS zum Hauptprogramm zurückgekehrt. Nach diesem RTS folgen zwei NOPs, damit man die Möglichkeit hat, hier einen JMP in den noch freien Bereich ab $6F7C einzubauen, wenn man EXT noch erweitern will.

Genauso wichtig wie die Initialisierung ist aber auch das Abschalten der Erweiterung! Dazu wird der Vektor auf den Befehl Quit in der ADRTAB verbogen und zeigt in EXT auf $608A. Das Abschalten besteht darin, daß das Bit 7 in ORGTAB+2 wieder gelöscht wird, (damit ist der Bildschirmspeicher, in dem sich die Erweiterung befindet, wieder zugänglich) und daß alle Veränderungen, die an Hi-Eddi plus vorgenommen wurden, wieder rückgängig gemacht werden. Es müssen also alle Vektoren in ADRTAB repariert, die neuen Befehle in BEFTAB gelöscht, der Walk-Befehl, der evtl. blockiert wurde, wieder freigegeben und alle sonstigen Abzweigungen etc. im Hi-Eddi plus wieder beseitigt werden.

Auch am Ende dieser Abschalt-Routine ($60BF) befinden sich wieder zwei NOPs, damit man sie in eine eigene Erweiterung verlängern kann.

Für Ihre eigene Erweiterung gibt es zwei Möglichkeiten: Entweder Sie erweitern EXT oder Sie schreiben eine ganz neue Erweiterung. Für den ersten Fall brauchen Sie noch die Speicherbelegung von EXT: Das Programm belegt den Bereich von $6000 bis $6F7C. Von $79E0 bis $7E00 liegen die Makros und der Bereich von $7E00 bis $8000 dient als temporärer Buffer, z. B. beim Scrollen. Diesen Bereich können Sie temporär ebenfalls benutzen. Uneingeschränkt benutzbar ist der Bereich von $6F7C bis $79E0,

also knapp 4 kByte. Ferner benutzt EXT neben den Hi-Eddi plus-Speicherzellen noch die Zeropagezellen $72 bis $77.

Nun bleibt mir nur noch, Ihnen viel Erfolg bei der Realisierung Ihrer Projekte zu wünschen!

Befehlsübersicht

Zeichenmodi (Kapitel 1.3):

D	Draw; Freihändig zeichnen
SHIFT D	Zeichnen mit dickem Pinsel
L	Line; Linien ziehen
R	Rectangle; Rechtecke zeichnen
C	Circle; Kreise zeichnen
P	Paint; Begrenzte Flächen ausfüllen
J	Jots; Spray-Funktion

Bildschirmverwaltung (Kapitel 1.4):

1-7	Bildschirm anwählen
C=1 - C=7	Bildschirm kopieren
	(C= bedeutet Commodore-Taste)
C= CLR	Clear; Bildschirm löschen
I	Invert; Bildschirm invertieren

Verschieben, Verknüpfen, Spiegeln (Kapitel 1.5):

M	Mover; Bereiche verschieben kopieren
Pfeil nach oben	Recall; Move-Quellbereich zurückholen
O	Oder-Verknüpfung von Bildschirmen
	oder Bereichen
X	Exor-(Exclusiv-Oder-)Verknüpfung
U	Und-Verknüpfung
SHIFT M	Mirror; Spiegelung zur Senkrechten
SHIFT T	Turm; Drehung um 180 Grad

Sprite-Befehle (Kapitel 1.6):

G	Get; Sprite aus Bildschirm herauskopieren
A	Append; Sprite in den Bildschirm einfügen
S	Stamp; Dito mit Löschem des Hintergrundes
E	Erase; Mit Spriterahmen löschen

Zoom-Funktion und Sprite-Editor (Kapitel 1.7):

SPACE	Aufruf; Aus Sprite-Modus in Sprite-Editor Aus den übrigen Modi in Zoom-Funktion

Befehle innerhalb der Zoom-Funktion/Sprite-Editor:

M	Mirror; Sprite zur Senkrechten spiegeln
T	Turn; Sprite um 180 Grad drehen
R	Rotate; Sprite um 90 Grad drehen
G	Grid; Punktraster ein- und ausblenden
I	Invert; Sprite invertieren
SHIFT CLR	Sprite löschen
B	Border; Rahmenfarbe weiterschalten
*	Cursorblinken ein/aus
C	Change; Sprites vertauschen
O,X,U	Sprites logisch verknüpfen
Space	Zurück in den Grafik-Editor

Farbbefehle (Kapitel 1.8):

F	Fore-Mode; Vordergrund-Modus anwählen (nur in Farb-Betriebsart) und Rahmenfarbe weiterschalten
B	Back-Mode; Hintergrund-Modus anwählen (nur in Farb-Betriebsart) und Rahmenfarbe weiterschalten
SHIFT F	Total Fore; Gesamten Vordergrund mit Rahmenfarbe einfärben
SHIFT B	Total Back; Gesamten Hintergrund mit Rahmenfarbe einfärben

Textbefehle (Kapitel 1.9):

T	Textmodus anwählen
Z	Schriftrichtung um 90 Grad weiterschalten
SHIFT Z	Commodore-Zeichensatz ins RAM kopieren
C= Z	Commodore-Zeichensatz aktivieren

Befehle zur Cursorsteuerung (Kapitel 1.10):

F1 - F7	Tabulatorposition anfahren
SHIFT F1 - SHIFT F7	Cursorposition als Tabulator speichern
H	Horizontale Schrittweite programmieren
V	Vertikale Schrittweite programmieren
C= F1 - C= F7	Schrittweiten speichern
*	Cursorblinken ein /aus
+	Schnelle Cursorgeschwindigkeit (Kapitel 1.2)
-	Langsame Cursorgeschwindigkeit (Kapitel 1.2)

Disk und Druckerbefehle (Kapitel 1.11)

C= D	Directory anzeigen
C= L	Load; Bilder, Sprites oder Zeichensätze laden
C= S	Save; Bilder, Sprites oder Zeichensätze speichern
C= C	Command; Floppy-Kommando senden
C= P	Print; Hardcopies drucken

Sonstige Befehle:

W	Walk; Bildfolge ablaufen lassen (Kapitel 1.12)
SHIFT W	Bildfolge progrmmieren
C= Q	Hi-Eddi plus ausschalten

Befehle der Erweiterung 'EXT'(Kapitel 4):

C= L, C= S	Load und Save erweitert (Kapitel 4.3)
SHIFT R	Rotate; Bereiche um 90 Grad drehen (Kapitel 4.4)
K	Koordinatenanzeige ein/aus (Kapitel 4.5)
SHIFT K	Koordinatenanzeige absolut (mit Scroller)
Y	Scroller; Bildschirmfenster über gesamtes Bild bewegen (Kapitel 4.6)
C= =	Makro-Definition (Kapitel 4.7)
SHIFT =	Makro anwählen und listen
=	Makro aufrufen
C= Q	Erweiterung abschalten

Index

Spitzen-Software für Commodore 128/128 D

WordStar 3.0 mit MailMerge

Der Bestseller unter den Textverarbeitungsprogrammen für PCs bietet Ihnen bildschirmorientierte Formatierung, deutschen Zeichensatz und DIN-Tastatur sowie integrierte Hilfstexte. Mit MailMerge können Sie Serienbriefe mit persönlicher Anrede an eine beliebige Anzahl von Adressen schreiben und auch die Adreßaufkleber drucken.

WordStar/MailMerge für den Commodore 128 PC
Bestell-Nr. MS 103 (5¼"-Diskette)

Hardware-Anforderungen: Commodore 128 PC, Diskettenlaufwerk, 80-Zeichen-Monitor, beliebiger Commodore-Drucker oder ein Drucker mit Centronics-Schnittstelle

Für nur DM 199,–* **(sFr. 178,–/öS 1890,–*)**

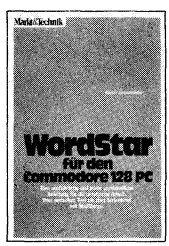
***inkl. MwSt. Unverbindliche Preisempfehlung**

Markt&Technik
128er-Software

MicroPro
WordStar 3.0
mit MailMerge für den Commodore 128 PC

5¼"-Diskette
im Floppy 1541-Format

Und dazu die weiterführende Literatur:



Mit diesem Buch haben Sie eine wertvolle Ergänzung zum WordStar-Handbuch: Anhand vieler Beispiele steigen Sie mühelos in die Praxis der Textverarbeitung mit **WordStar** ein. Angefangen beim einfachen Brief bis hin zur umfangreichen Manuskripterstellung zeigt Ihnen dieses Buch auch, wie Sie mit Hilfe von MailMerge Serienbriefe an eine beliebige Anzahl von Adressen mit persönlicher Anrede senden können.

Best.-Nr. MT 780
ISBN 3-89090-181-6
DM 49,– (sFr. 45,10/öS 382,20)

Erhältlich bei Ihrem Buchhändler.

Sie erhalten jedes **WordStar-Programm** für Ihren Commodore 128 fertig angepaßt (Bildschirmsteuerung). **Jeweils Originalprodukte!** Jedes Programmpaket enthält außerdem ein ausführliches Handbuch mit kompakter Befehlsübersicht.

Diese Markt&Technik-Softwareprodukte erhalten Sie in den Computer-Abteilungen der Kaufhäuser oder bei Ihrem Computerhändler.

610321

Markt&Technik
UNTERNEHMENSBEREICH
BUCHVERLAG
Hans-Pinsel-Straße 2, 8013 Haar bei München

Spitzen-Software für Commodore 128/128 D

dBASE II, Version 2.41

dBASE II, das meistverkaufte Programm unter den Datenbanksystemen, eröffnet Ihnen optimale Möglichkeiten der Daten- u. Dateihandhabung. Einfach u. schnell können Datenstrukturen definiert, benutzt und geändert werden. Der Datenzugriff erfolgt sequentiell oder nach frei wählbaren Kriterien, die integrierte Kommandosprache ermöglicht den Aufbau kompletter Anwendungen wie Finanzbuchhaltung, Lagerverwaltung, Betriebsabrechnung usw.

dBASE II für den Commodore 128 PC
Bestell-Nr. MS 303 (5¼"-Diskette)

Hardware-Anforderungen: Commodore 128 PC, Diskettenlaufwerk, 80-Zeichen-Monitor, beliebiger Commodore-Drucker oder ein Drucker mit Centronics-Schnittstelle

Für nur DM 199,–* (sFr. 178,–/öS 1890,–*)

***inkl. MwSt. Unverbindliche Preisempfehlung**

Markt&Technik
128er-Software

dBASE™ II
ASHTON-TATE
für den Commodore 128 PC

5¼"-Diskette
im Floppy 1541-Format

Und dazu die weiterführende Literatur:

Zu einem Weltbestseller unter den Datenbanksystemen gehört auch ein klassisches Einführungs- und Nachschlagewerk! Dieses Buch von dem deutschen Erfolgsautor Dr. Peter Albrecht begleitet Sie mit nützlichen Hinweisen bei Ihrer täglichen Arbeit mit **dBASE II.** Schon nach Beherrschung weniger Befehle ist der Einsteiger in der Lage, Dateien zu erstellen, mit Informationen zu laden und auszuwerten.

Best.-Nr. MT 838
ISBN 3-89090-189-1
DM 49,– (sFr. 45,10/öS 382,20)

Erhältlich bei Ihrem Buchhändler.

Sie erhalten jedes **dBASE II-Programm** für Ihren Commodore 128 PC fertig angepaßt (Bildschirmsteuerung). **Jeweils Originalprodukte!** Jedes Programmpaket enthält außerdem ein ausführliches Handbuch mit kompakter Befehlsübersicht.

Diese Markt&Technik-Softwareprodukte erhalten Sie in den Computer-Abteilungen der Kaufhäuser oder bei Ihrem Computerhändler

610322

Markt&Technik
UNTERNEHMENSBEREICH
BUCHVERLAG
Hans-Pinsel-Straße 2, 8013 Haar bei München

Spitzen-Software für Commodore 128/128 D

MULTIPLAN, Version 1.06

Wenn Sie die zeitraubende manuelle Verwaltung tabellarischer Aufstellungen mit Bleistift, Radiergummi und Rechenmaschine satt haben, dann ist MULTIPLAN, das System zur Bearbeitung »elektronischer Datenblätter«, genau das richtige für Sie! Das benutzerfreundliche und leistungsfähige Tabellenkalkulationsprogramm kann bei allen Analyse- und Planungsberechnungen eingesetzt werden wie z. B. Budgetplanungen, Produktkalkulationen, Personalkosten usw. Spezielle Formatierungs-, Aufbereitungs- und Druckanweisungen ermöglichen außerdem optimal aufbereitete Präsentationsunterlagen!

MULTIPLAN für den Commodore 128 PC
Bestell-Nr. MS 203 (5¼"-Diskette)

Hardware-Anforderungen: Commodore 128 PC, Diskettenlaufwerk, 80-Zeichen-Monitor, beliebiger Commodore-Drucker oder ein Drucker mit Centronics-Schnittstelle

Für nur DM 199,–* **(sFr. 178,–/öS 1890,–*)**

***inkl. MwSt. Unverbindliche Preisempfehlung**

Markt&Technik
128er-Software

MICROSOFT
MULTIPLAN
für den
Commodore 128 PC

5¼"-Diskette
im Floppy 1541-Format

Und dazu die weiterführende Literatur:

Dank seiner Menütechnik ist **MULTIPLAN** sehr schnell erlernbar. Mit diesem Buch von Dr. Peter Albrecht werden Sie Ihre Tabellenkalulation ohne Probleme in den Griff bekommen. Als Nachschlagewerk leistet es auch dem Profi nützliche Dienste.

Best.-Nr. MT 836
ISBN 3-89090-187-5
DM 49,– (sFr. 45,10/öS 382,20)

Erhältlich bei Ihrem Buchhändler.

Sie erhalten jedes **MULTIPLAN-Programm** für Ihren Commodore 128 PC fertig angepaßt (Bildschirmsteuerung). **Jeweils Originalprodukte!** Jedes Programmpaket enthält außerdem ein ausführliches Handbuch mit kompakter Befehlsübersicht.

Diese Markt & Technik-Softwareprodukte erhalten Sie in den Computer-Abteilungen der Kaufhäuser oder bei Ihrem Computerhändler.

610323

Markt&Technik
UNTERNEHMENSBEREICH
BUCHVERLAG
Hans-Pinsel-Straße 2, 8013 Haar bei München

Bücher zum Commodore 128/128 D

H. Ponnath

Grafik-Programmierung C128

1986, 196 Seiten

inkl. Beispieldiskette

Ein mächtiges Werkzeug hat der Anwender von Computergrafik mit dem Basic 7.0 des Commodore 128 PC in den Händen! Was man damit alles anfangen kann, soll Ihnen dieses Buch zeigen: hochauflösende Grafik, Multicolorbilder, Sprites und Shapes werden anhand von vielen Beispielprogrammen besprochen. Die Videochips und ihre Möglichkeiten sind ebenso Thema wie einige nützliche Assemblerroutinen, die Speicherorganisation, der 80-Zeichen-Bildschirm und vieles andere mehr. Außerdem enthält das Buch eine Diskette mit allen Programmen.

Best.-Nr. MT 90202
ISBN 3-89090-202-2
DM 52,–/sFr. 47,80/öS 405,60

P. Rosenbeck

Das Commodore 128-Handbuch

1985, 383 Seiten

Dieses Buch sagt Ihnen alles, was Sie über Ihren C128 wissen müssen: die Hardware, die drei Betriebssystem-Modi und was die CP/M-Fähigkeit für Ihren Computer bedeutet. Aber Sie werden irgendwann Lust verspüren, tiefer in Ihren C128 einzusteigen. Auch dafür ist gesorgt: an einen Assemblerkurs, der Ihnen zugleich die Funktionsweise des eingebauten Monitors nahebringt, schließen sich Kapitel an, die mit Ihnen auf Entdeckungsreise ins Innere der Maschine gehen. Daß die Reise spannend wird, dafür sorgen die Beispiele, aus denen Sie viel über die Interna des Systems lernen können – bis hin zur Grafik-Programmierung.

Best.-Nr. MT 90195
ISBN 3-89090-195-6
DM 52,–/sFr. 47,80/öS 405,60

J. Hückstädt

BASIC 7.0 auf dem Commodore 128

1985, 239 Seiten

Das neue BASIC 7.0 des C128 eröffnet mit seinen ca. 150 Befehlen ganz neue Dimensionen der BASIC-Programmierung. Es ermöglicht dem Anfänger den einfachen und effektiven Zugriff auf die erstaunlichen Grafik- und Tonmöglichkeiten des C128; der Fortgeschrittene findet die nötigen Informationen für (auch systemnahe) Profi-Programmierung mit strukturierten Sprachmitteln.

An praxisnahen Beispielen (wie z.B. der Dateiverwaltung) zeigt der Autor auf, wie man die für den 128er typischen Merkmale und Eigenschaften (Sprites, Shapes, hochauflösende Grafik, Musikprogrammierung und Geräusche) optimal nutzt!.

Best.-Nr. MT 90149
ISBN 3-89090-170-0
DM 52,–/sFr. 47,80/öS405,60

610324

Markt & Technik-Fachbücher erhalten Sie bei Ihrem Buchhändler

Markt&Technik
UNTERNEHMENSBEREICH
BUCHVERLAG
Hans-Pinsel-Straße 2, 8013 Haar bei München

TOP-ASS

Der ASE-Macro-Assembler für den Commodore 128 PC mit integriertem Editor, Monitor und Linker.

Dieser 6502-Macroassembler setzt neue Maßstäbe. Seine Leistungsfähigkeit wird jeden verwöhnten Maschinenprogrammierer überzeugen:

- integrierter Editor, der schon bei der Eingabe des Quelltextes eine Syntaxüberprüfung vornimmt;
- integrierter Linker, mit dem quellgesteuertes Linken von relokatiblen Modulen möglich ist;
- assemblereigene schnelle und gleichzeitig sehr leistungsfähige Integerarithmetik;
- über 2000 Labels können gleichzeitig verwaltet werden, das heißt Maschinenprogramme bis zu einer Länge von ca. 25 KByte Objektcode können bei Bedarf in einem Rutsch assembliert werden;
- Macros mit beliebig vielen Parametern, Macrobibliotheken, Minimacs, bedingte Assemblierung, Labeleingabe im Dialog, Ausgabe formatierter Assemblerlistings, Ausgabe sortierter Symboltabellen und vieles andere mehr.

Außerdem wird der ASE-Macroassembler von einem sehr guten Monitor und einem Relativlader unterstützt, der relokatible Module an beliebige Speicheradressen laden kann und endlich Schluß macht mit den Dutzenden Maschinenprogrammen auf Diskette, die sich nur durch ihre Startadresse unterscheiden!

Lernen Sie es kennen, das TOP-ASS Assembler-Entwicklungssystem! Es lohnt sich!

Best.-Nr. MD 253A **DM 89,-**
inkl. MwSt. Unverbindliche Preisempfehlung.

PROTEXT

Die Profi-Textverarbeitung mit vollautomatischer Silbentrennung, integrierter Tabellenkalkulation und Zusatzprogramm zum Überprüfen der Rechtschreibung für den Commodore 128 PC.

PROTEXT ist ein leicht bedienbares Textprogramm mit hoher Leistungsfähigkeit. Eingebaute Hilfefunktionen ermöglichen eine schnelle Einarbeitung. Mit **PROTEXT** sind daher auch Anfänger in der Lage, alle Vorteile eines professionellen Textprogramms zu nutzen. Überzeugen Sie sich selbst!

Was **PROTEXT** alles kann:

- Farbkombination für Hintergrund und Schrift (Vordergrund) frei wählbar;
- formatierte Ausgabe auf Bildschirm und Drucker mit programmierbaren Haltepunkten über serielle, V24- oder zwei Software-Centronics-Schnittstellen;
- vielfältige Formatanweisungen: linker/rechter Rand, vollautomatische Silbentrennung, Kopf-/Fußzeilen, Fußnoten, Zentrieren usw.;
- schnelle selbstlernende Textkorrektur mit deutschem (ca. 25000 Worte) Grundwortschatz sowie neun Kundenbibliotheken, die in Text umgewandelt, bearbeitet, ergänzt, sortiert und ausdruckbar sind;
- Textübertragung per DFÜ mit Space-Optimierung und automatischer Fehlerkorrektur;
- leistungsfähige Rechenmöglichkeiten mit Zeilenmarkierung (Rechentabulator), Kolonnenverarbeitung, progr. Tabellenkalkulation und Taschenrechner.

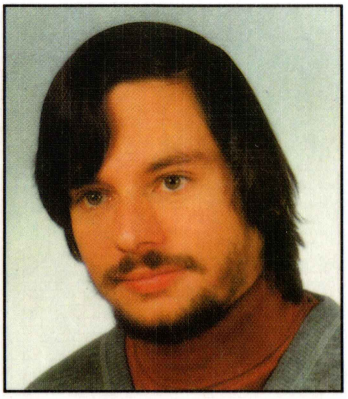
Hardwareanforderungen: C128 oder C128D, 80-Zeichen-Monitor, Commodore-Drucker oder Drucker mit Centronics-Schnittstelle.

Best.-Nr. MD 254A **DM 89,-** inkl. MwSt. Unverbindliche Preisempfehlung.

610325

TOP-ASS und PROTEXT erhalten Sie in den Computer-Abteilungen der Kaufhäuser und in Computershops.

Markt&Technik
UNTERNEHMENSBEREICH
BUCHVERLAG
Hans-Pinsel-Straße 2, 8013 Haar bei München



HANS HABERL
Jahrgang 1960, absolvierte 1980 das Abitur. Sein Interesse gilt allem Technischen, was sich neben den Hobbys Elektronikbasteln und Computerei auch in der Studienwahl äußert: Er steht kurz vor dem Abschluß seines Elektronikstudiums mit Schwerpunkt Datenverarbeitung an der Technischen Universität München. Neben seinen technischen Neigungen ist er ein Fan klassischer Musik.

Mini-CAD mit Hi-Eddi plus auf dem C 64 / C 128

Zusammen mit diesem Buch erhalten Sie auf Diskette ein CAD-Programm, welches das komfortable Erstellen von technischen Zeichnungen, Plänen oder Diagrammen ebenso erlaubt wie das Malen von Bildern in Farbe, das Entwerfen und Ausdrucken von Glückwunschkarten, Schildern, Briefköpfen und sogar das Erstellen von kleinen Trickfilmen oder von Schaufensterwerbung.
Bei dem Programm handelt es sich um eine stark verbesserte Version des Zeichenprogrammes »Hi-Eddi« aus der Zeitschrift 64'er.
Das Buch beschreibt ausführlich und auch für den Laien verständlich die Bedienung des Programmes. Es zeigt Ihnen die Anwendungsmöglichkeiten an einer Reihe von Beispielen wie das Erstellen von Schaltplänen, Platinenlayouts und Struktogrammen. Dazu bietet es durch eine ausführliche Dokumentation dem maschinenspracheerfahrenen Programmierer die Möglichkeit, Grafikroutinen in eigene Programme zu übernehmen oder den Hi-Eddi plus zu erweitern.

Hardwareanforderungen:

- Commodore 64 oder Commodore 128 PC (C 128 D).
  Bei Benutzung eines Commodore 128: Die mitgelieferte Diskette mit dem Hi-Eddi-plus-Programm ist nur im 64er-Modus lauffähig!
- 40-Zeichen-Monitor (schwarzweiß/farbig) oder ein Fernsehgerät
- Floppylaufwerk 1541 oder 1571

Für folgende Druckermodelle besitzt Hi-Eddi plus bereits einen fertig programmierten Druckertreiber: VC 1525/MPS 801/MPS 802/MPS 803, STAR-Drucker und Epson-kompatible Drucker.
Als Eingabegerät kann ein Joystick oder ein Koala-Pad verwendet werden.

ISB N 3-89090-136-0

4 001057 901360

Markt&Technik

DM 48,-
sFr. 44,20
öS 374,40