

Litzkendorf

**mit
Graphic
Construction
Set**

**Das
große**

**GfA
BASIC**

Buch

EIN DATA BECKER BUCH

Litzkendorf

**mit
Graphic
Construction
Set**

**Das
große**

**GfA
BASIC**

Buch

EIN DATA BECKER BUCH

ISBN 3-89011-222-6

Copyright © 1986 DATA BECKER GmbH
Merowingerstraße 30
4000 Düsseldorf

Alle Rechte vorbehalten. Kein Teil dieses Programms darf in irgendeiner Form (Druck, Fotokopie oder einem anderen Verfahren) ohne schriftliche Genehmigung der DATA BECKER GmbH reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.*

Wichtiger Hinweis:

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

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

Vorwort

Ich kann mich noch sehr gut daran erinnern, als ich vor ca. 2 Jahren das erste Mal einen Computer mein Eigen nannte, wie sehr ich mit meinem Einzelgängerdasein zu kämpfen hatte. Selbstverständlich hatte ich Freunde und Bekannte. Ich bin ein kontaktfreudiger Mensch. Trotzdem wollte es mir nicht gelingen, eine Gruppe von Gleichgesinnten in meiner näheren Umgebung zu finden, mit denen ich Computerprobleme gemeinsam lösen konnte. Auf mich allein gestellt nahm ich alles an Büchern und Zeitschriften, was nur annähernd nach Computer roch oder aussah, auf, um diese Startschwierigkeiten zu überwinden. Nach diesen Erfahrungen habe ich mir es zur Aufgabe gemacht, anderen, die sich in ähnlicher Situation befinden, den Einstieg zu diesem Medium etwas zu erleichtern. Da nun nicht jeder die Möglichkeit hat, sich im Erfahrungsaustausch mit anderen fortzubilden, sind Bücher die beste Gelegenheit, seinen Wissensschatz zu erweitern. Dieses Buch soll Ihnen dabei helfen, einer neuen Programmiersprache näherzukommen, die in ihrer Komplexität und Vielseitigkeit alles Vergleichbare in die Schranken verweist.

Gerade als überzeugter BASIC-Fan bin ich immer wieder Computerfreaks begegnet, die es für notwendig hielten, mir klar machen zu müssen, daß ich als Nur-BASIC-Programmierer doch wohl nicht auf der richtigen Schiene laufe. Mit einem derartigen BASIC-Interpreter wie dem GfA-BASIC in Händen sehe ich heute nicht mehr die geringste Berechtigung zu dieser Aussage. Ein BASIC, das sich in Geschwindigkeitsvergleichen mit den üblichen Compilersprachen wie "C", "Pascal" oder "Modula" ohne weiteres messen kann und zudem die Möglichkeiten zu den meisten System- und Diskettenoperationen bietet, stellt eine Programmiersprache dar, die auf jeden Fall auch für ernsthafte Anwendungen wie Textverarbeitung, Grafikprogramme oder Kalkulations- und Verwaltungsprogramme in Betracht gezogen werden muß. Dank der überwiegenden Leistung eines Einzelnen fühle ich mich nun als BASIC-Programmierer mit meinesgleichen in allerbesten Gesellschaft. Dieser Einzelne ist der Programmierer dieses fantastischen BASIC, dem ich hier auch

namentlichen ein kleines Denkmal setzen möchte. Meine ehrliche Hochachtung und herzlichen Glückwunsch zu dieser echten Meisterleistung, Herr Ostrowski!

Es bleibt zu wünschen, daß sich die allgemeine Unmanier, auch qualitativ hochwertige Programme unerlaubterweise per Raubkopie zu verbreiten, in diesem Fall in erträglichen Maßen hält. Dieses BASIC gehört mit Sicherheit zur High Society unter den Software-Produkten, die für den Atari ST zu haben sind, und das zu einem erstaunlich niedrigen Preis.

Uwe Litzkendorf

Hildesheim, 21. August 1986

Inhaltsverzeichnis

Einleitung	11
1. Das GfA-BASIC	13
1.1 Ein- und Ausblicke.....	13
1.2 Der Editor - Komfort hoch drei	15
1.2.1 Cursor-Positionierung	15
1.2.2 Das Editor-Menü.....	18
1.2.3 Weitere Hilfsfunktionen.....	26
1.3 Der Direct-Modus - das schnelle Etwas	27
1.4 Der Run-only-Interpreter - der kleine Unterschied....	29
2. GfA-ABC	31
2.1 Strukturiertes Pogrammieren	31
2.2 Jetzt geht's los	34
Anhang A: GEM, GEMDOS, BIOS, XBIOS.....	423
Anhang B: Bedienungsanleitung zum "Graf Pro"	447
Anhang C: Fehlermeldungen	457
Anhang D: Übersicht der BASIC-Befehle.....	463
Anhang E: Programmänderung für den 260 ST.....	469

Einleitung

Sie sind im Besitz eines GfA-BASIC-Interpreters und nun fürchterlich gespannt darauf, was man denn nun alles damit anstellen kann. Die Ansprüche, die allgemein an ein BASIC gestellt werden, sind erfahrungsgemäß nicht sehr hoch. Um nun das Image dieser Sprache aufzupolieren, bleibt wohl nichts anderes übrig, als den Beweis anzutreten, daß man mit BASIC heutzutage Programmideen verwirklichen kann, die bisher nur den Compiler- oder Assemblersprachen zugemutet werden konnten. Dieses Buch soll Ihnen eine Vorstellung davon vermitteln. Mit einer kompletten Befehlsbeschreibung, einem ausbaufähigen Grafikprogramm, vielen Tricks und Beispielen soll versucht werden, Ihnen den Einstieg in diese neue Programmiersprache zu erleichtern und Ihnen so schnell wie möglich zu Erfolgserlebnissen zu verhelfen. Es gibt wohl kaum eine andere Programmiersprache, mit der auf so atemberaubend einfache Weise selbst schwierige Probleme lösbar sind, als das GfA-BASIC. Überzeugen Sie sich selbst!

Zum Konzept dieses Buches ist folgendes zu sagen: Sie finden in den ersten Abschnitten einige Anmerkungen zu verschiedenen Grundlagen, die als Basiswissen zum Umgang mit dieser Sprache unerläßlich sind. Daran schließt sich dann das komplette Listing eines Grafikprogramms an, das auch höheren Ansprüchen genügen wird. Dieses Programm wurde in nur drei Wochen in GfA-BASIC geschrieben. Über die Grundzeichenoperationen hinaus werden Funktionen angeboten, die z.B. das Verzerren oder Wölben von Bildausschnitten ermöglichen, sowie ein Trickfilmstudio, mit dem komplette Zeichentrickfilme erstellt, gespeichert, geladen und vorgeführt werden können. Innerhalb des Listings werden an jeweils relevanten Stellen die mächtigen Befehle dieses BASIC erläutert und gegebenenfalls anhand von kleinen Beispiel-Listings dokumentiert. Diese Listings sind zur Unterscheidung vom Hauptprogramm grau unterlegt.

Im Anschluß an das Listing finden Sie dann noch verschiedene GEM-, GEMDOS-, BIOS- und XBIOS-Routinen, die Ihnen Ihre Arbeit wesentlich erleichtern können und außerdem einen

zusätzlich Spielraum für Ihre individuellen Programmgestaltungswünsche eröffnen. Da in GfA-BASIC auch Pulldown-Menüs und Windowtechnik keine unerfüllbaren Träume mehr sind, gibt es natürlich auch hierzu ein Beispielprogramm. Zum schnellen Auffinden einzelner BASIC-Befehle können Sie den Index im Anhang D verwenden. Haben Sie "Appetit" bekommen, dann lassen Sie sich überraschen. Viel Spaß und viel Erfolg bei der Arbeit mit diesem Buch.

1. Das GfA-BASIC

Schon lange vor der Ausgabe der ersten Vorab-Versionen waren hochtrabende Töne über dieses Wunderding zu hören. Zwar hatte es noch niemand gesehen, aber wenn auch nur ein Teil der Gerüchte wahr werden sollte, dann... Wie sich heute herausstellt, ist nicht nur ein Teil der Gerüchte wahr geworden. Nachdem ich Ende Juni die lauffähige Endversion auf meinem ST geladen hatte, konnte ich mich vor Begeisterung kaum noch auf dem Stuhl halten. Gewohnt, mit allen möglichen und unmöglichen Tricks und Raffinessen meinem ST-BASIC auf die Sprünge zu helfen, fand ich nun Befehle, die wie selbstverständlich Möglichkeiten eröffneten, um die ich im ST-BASIC mit zerzausten Haaren und abgebißenen Fingernägeln einen aufreibenden Kampf führen mußte. Zudem eine Vielzahl an Befehlen, an die man bei anderen BASIC-Dialekten noch nicht mal zu denken gewagt hätte.

1.1 Ein- und Ausblicke

Es war nur eine Frage der Zeit, bis endlich die fantastischen Möglichkeiten, die sich durch die 16-Bit-Technik bieten, auch Auswirkungen in der Gestaltung der Computersprachen zeigen würden. Der MC 68000 von Motorola, der Hauptprozessor in den meisten neuen 16-Bitern wie z.B. dem Apple Macintosh, dem Amiga, dem QL aber eben auch in unserem heißgeliebten ATARI ST, bietet im Vergleich zu den herkömmlichen 8-Bit-Prozessoren, wie z.B. dem 6502 im C64, eine Vielzahl an Maschinenbefehlen, die erst durch die 16-Bit-Datenbreite verwirklicht werden konnten und auch gleichzeitig die Fähigkeiten und Geschwindigkeit eines 8-Megahertz-Takters voll zur Geltung brachten. Was heißt nun 8 Megahertz? Hertz ist eine aus der Physik bekannte Einheit für Schwingungen pro Sekunde. Unsere Fernseher z.B. arbeiten mit einer Bildwiederholungsfrequenz von 50 Hertz. D.h. jede Bildschirmzeile wird innerhalb einer Sekunde 50mal neu aufgebaut. Für unseren Computer bedeutet das, daß ein spezielles Schwing-Quarz mit einer Frequenz von 8 Millionen (Mega) Hertz die Stromspannung an

unseren Prozessor anlegt. Das heißt wiederum, daß pro Takt ein Schalter-Zustand bearbeitet werden kann. In Kombination mit 16 Daten- und 24 Adreßleitungen ergibt sich daraus eine kaum noch zu erfassende Variabilität.

Man stelle sich eine riesige Lagerhalle vor, in welcher Regale mit insgesamt ca. 16 (!) Millionen Schubladen untergebracht sind. In jeder dieser Schubladen läge eine Information, die bestimmte Auskünfte über die Arbeitssituation im Betrieb gibt. Nun soll jemand innerhalb kürzester Zeit erfassen, welche Information in welcher Schublade liegt und welche Auswirkungen der Inhalt dieser Schublade im Zusammenspiel mit vielen verschiedenen anderen Schubladeninhalten auf die Organisation des Gesamtbetriebes hat. Das ist, wenn man unser Gehirn als Vergleich nicht in Betracht zieht, unzweifelhaft mit menschlicher Kraft nicht machbar. Der Computer kann das. Innerhalb einer Sekunde wäre er sogar theoretisch in der Lage, die Hälfte aller Schubladen zu untersuchen. Da er jedoch auch analysieren, rechnen und einordnen muß, ergibt sich eine größere Zeitspanne.

Trotzdem reicht seine Geschwindigkeit aus, um z.B. mit dem GfA-Interpreter in weniger als einer zwanzigstel Sekunde eine FOR-NEXT-Schleife mit 1000 Schritten zu durchlaufen. Innerhalb eines Schrittes dieser Schleife muß er intern hunderte von Einzelschritten abarbeiten, die Richtigkeit des Programms in seiner Grammatik überprüfen und die verwendeten Befehle analysieren und zuordnen (interpretieren). Wahrhaft eine gewaltige Leistung. Es ist verständlich, daß auch die Programmierer der für derart vielfältige Möglichkeiten würdigen Programmiersprachen einige Zeit brauchten, um dieses Angebot auch in entsprechenden Interpretern und Compilern zu verwirklichen. War ein 8-Bitter in seinen Kombinationsmöglichkeiten noch einigermaßen überschaubar, ist ein 16-Bitter real kaum noch zu begreifen.

Glücklicherweise wächst die Fähigkeit mit den Möglichkeiten, die sich dem Interessierten bieten. So haben wir einen Interpreter, der das Angebot eines MC 68000 erstmals weitgehend in einer für viele nutzbaren Sprache zur Verfügung stellt. Er tritt

damit in eine ernstzunehmende Konkurrenz mit der bis heute favorisierten Compilersprache "C". Zugegeben, in manchen Beziehungen werden "C"- und Assemblerprogrammierung Vorrang behalten. Aber die Gruppe derer, die bereit sind, sich mit der komplizierten Compiler- und Assemblertechnik auseinanderzusetzen, wird sich mehr und mehr in Grenzen halten, da das GfA-BASIC den meisten Anforderungen mit Sicherheit Genüge tut.

1.2 Der Editor

Schauen wir uns unser Prachtstück einmal etwas aus der Nähe an. Neben einer gelungenen syntaktischen Anordnung der Sprache spielt der Editor die zweitgrößte Rolle bei einer Computersprache (Syntax, lat.: die logisch richtige Zusammensetzung einer Sprache. Im Gegensatz zur Grammatik, die die Art und Weise der Anwendung von Ausdrucksmitteln festlegt). Was soll ich mit einer noch so genialen Syntax, wenn die Funktionsweise des Editors mir Kabinettstückchen abverlangt, um meine Programmvorstellungen Realität werden zu lassen. Auch in diesem Punkt gibt es herzlich wenig am GfA-BASIC auszusetzen. Zumal, und das werden wir später noch genauer erkennen, dieser Editor für viele Zwecke erweiterbar ist. Nachfolgend werden nun die einzelnen Editorfunktionen erklärt.

1.2.1 Cursor-Positionierung

Cursor rechts:

Editor-Cursor in der aktuellen Programmzeile um eine Position nach rechts bewegen.

==> Pfeil-rechts-Cursor-Taste im Cursor-Tastenblock drücken.

Cursor links:

Editor-Cursor in der aktuellen Programmzeile um eine Position nach links bewegen.

==> Pfeil-links-Cursor-Taste im Cursor-Tastenblock drücken.

Cursor an Zeilenanfang:

Editor-Cursor in der aktuellen Programmzeile zum Zeilenanfang bewegen.

==> >Control<-Taste und Pfeil-links-Taste gleichzeitig drücken.

Cursor an Zeilenende:

Editor-Cursor in der aktuellen Programmzeile zum Zeilenende bewegen.

==> >Control<-Taste und Pfeil-rechts-Taste gleichzeitig drücken.

Anmerkung: Ist die Programmzeile länger als 79 Zeichen, wird die Zeile während der Edition jeweils um ein Zeichen nach links versetzt. Nach Abschluß der Zeile erscheint am rechten Bildrand in dieser Zeile ein Pfeil, der so andeutet, daß die Zeilenlänge mehr als 79 Zeichen beträgt.

Cursor-rechts-Tabulator:

Editor-Cursor in der aktuellen Programmzeile in 8er-Schritten nach rechts bewegen.

==> >Tab<-Taste drücken.

Cursor-links-Tabulator:

Editor-Cursor in der aktuellen Programmzeile in 8er-Schritten nach links bewegen.

==> >Control<-Taste und >Tab<-Taste gleichzeitig drücken.

Cursor eine Zeile Aufwärts:

Editor-Cursor um eine Zeile über die aktuelle Programmzeile bewegen + Syntaxkontrolle.

==> Pfeil-oben-Cursor-Taste im Cursor-Tastenblock drücken.

Cursor eine Zeile abwärts:

Editor-Cursor um eine Zeile unter die aktuelle Programmzeile bewegen + Syntaxkontrolle.

==> Pfeil-unten-Cursor-Taste oder >Return<-Taste drücken.

Cursor beliebig positionieren:

Editor-Cursor auf der aktuellen Seite an eine beliebige Stelle bewegen + Syntaxkontrolle bei Zeilenwechsel.

==> Anklicken der gewünschten Position mit dem Mauszeiger.

Anm.: Wenn der Cursor nicht an den Zeilenanfang bewegt werden soll, sondern direkt an die Mauszeigerposition, ist es ratsam, die Maustaste kurze Zeit gedrückt zu halten, da die Sprungorientierung sich zuerst auf den Zeilenanfang bezieht und erst in zweiter Instanz auf die Mauszeigerposition innerhalb der Zeile.

Cursor an Seitenanfang:

Editor-Cursor an den Anfang der aktuellen Editorseite bewegen + Syntaxkontrolle + Strukturierung.

==> >ClrHome<-Taste im Cursor-Tastenblock drücken.

Cursor eine Seite Rückwärts:

Editor-Cursor an den Anfang der vorhergehenden Editorseite bewegen + Syntaxkontrolle + Strukturierung.

==> >Control<-Taste und Pfeil-oben-Taste gleichzeitig oder >Shift<-Taste und >F7<-Taste gleichzeitig drücken oder "Pg up" im Editor-Menü anklicken.

Cursor eine Seite vorwärts:

Editor-Cursor an den Anfang der nächsten Editorseite bewegen + Syntaxkontrolle + Strukturierung.

==> >Control<-Taste und Pfeil-unten-Taste gleichzeitig oder >F7<-Taste drücken oder "Pg down" im Editor-Menü anklicken.

Cursor an Programmanfang:

Editor-Cursor an den Anfang des Programms bewegen + Syntaxkontrolle + Strukturierung.

==> >Control<-Taste und >ClrHome<-Taste gleichzeitig drücken.

Cursor an Programmende:

Editor Cursor an das Ende des Programms bewegen + Syntaxkontrolle + Strukturierung.

==> >Control<-Taste und >z<-Taste gleichzeitig drücken.

1.2.2. Das Editor-Menü

Programm speichern:

Programm wird mit dem Extender ".BAS" auf Diskette gespeichert. Dabei wird das Programm als gesamter Speicherblock direkt auf die Diskette verfrachtet, ohne die Interpreter-Kontrolle zu durchlaufen. Dadurch ergibt sich eine erheblich kürzere Speicherzeit.

==> >Shift<-Taste und >F1<-Taste gleichzeitig drücken oder "Save" im Editor-Menü anklicken.

Programm laden:

Programme mit dem Extender ".BAS" können von Diskette geladen werden. Hier wird ebenfalls die Interpreterkontrolle übersprungen. Die hiermit geladenen Dateien verdrängen die im Editor befindlichen Programme. Gegebenenfalls sollten Sie das aktuelle Programm im Editor zuerst auf Diskette sichern. Der "Load"-Vorgang läßt sich nach der Dateiauswahl in der File-Select-Box nur noch durch einen Reset stoppen. Dadurch wäre das Editorprogramm allerdings ebenfalls unwiderruflich verloren.

==> >F1<-Taste drücken oder "Load" im Editor-Menü anklicken.

Textdatei speichern:

Programm wird als Textfile mit dem Extender ".LST" auf Diskette gespeichert. Dabei durchläuft die geladene Datei die Interpreterkontrolle. Die benötigte Zeit für den Speichervorgang ist erheblich länger als mit "Save".

==> >Shift<-Taste und >F2<-Taste gleichzeitig drücken oder "Save,A" im Editor-Menü anklicken.

Textdatei laden:

Als Textdatei abgespeicherte Programme mit dem Extender ".LST" können von Diskette geladen werden. Auch diese Funktion benötigt mehr Zeit zum Laden, als die "Load"-Funktion. Sie hat dagegen allerdings den Vorteil, daß verschiedene Dateien miteinander verknüpft werden können.

==> >F2<-Taste drücken oder "Merge" im Editor-Menü anklicken.

Grundsätzliches zur File-Select-Box: ("File", engl.: Akte; "select", engl.: auswählen)

Wenn Lade- oder Speicherfunktionen aus dem Editor aufgerufen werden, erscheint eine sogenannte "File-Select-Box" (s. auch BASIC-Befehl: "Fileselect"). Diese Box listet in einem Fenster die unter dem angegebenen Pfadnamen (Zeile über dem Fenster) verfügbaren Dateien auf.

Die mit der inversen Raute gekennzeichneten Dateien stellen sogenannte "Sub-Directories" (engl.: Unter-Inhaltsverzeichnis) dar, in denen weitere Dateien unter dem für diese Sub-Directory typischen Pfadnamen abgespeichert sein können. Durch Anklicken des Sub-Dir-Namens wird sie geöffnet und die darin befindlichen Dateien können geladen werden. Da das TOS immer die Diskstation zur aktuellen erklärt, von welcher das Programm geladen wurde, kann es notwendig werden, im Pfadnamen die Stationsangabe zu ändern. Klicken Sie dazu die Pfadnamen-Zeile an, gehen Sie anhand der Cursor-Tasten an den Anfang der Zeile und ersetzen Sie den Stationsnamen. Das gleiche kann mit Extendern vorgenommen werden.

(z.B. "A:*.BAS_____ " ==> "C:*.*_____ ")

Anschließend klicken Sie den oberen Querbalken des Fensters in der linken oberen Fensterecke an. Es werden jetzt statt der ".BAS"-Files von Diskstation A alle (!) Files von Diskstation C aufgelistet. Ob man alle Dateien laden kann, ist eine andere Frage und abhängig vom Editor, aber so kann man ohne weiteres die gesamte Diskette nach bestimmten Dateien durchsuchen. Näheres wird unter dem Befehl "Fileselect" erklärt.

Interpreter-Arbeit beenden:

Es erscheint eine Kontroll-Abfrage. Antwort "JA" bewirkt Rückkehr zum Desktop. Antwort "NEIN" kehrt zum Interpreter zurück.

==> >Shift<-Taste und >F3<-Taste gleichzeitig drücken oder "Quit" im Editor-Menü anklicken.

Programm-Listing ausdrucken:

Es erscheint eine Kontroll-Abfrage. Antwort "JA" druckt formatiertes Listing auf angeschlossenem Drucker aus. (Vorsicht: sollte kein Drucker angeschlossen sein oder ist er nicht "On Line" kehrt das System erst nach ca. einer halben Minute zum Interpreter zurück. Also, nicht gleich den RESET-Knopf drücken!) Antwort "NEIN" kehrt zum Interpreter zurück.

==> >F3<-Taste drücken oder "Llist" im Editor-Menü anklicken.

Programm löschen:

Es erscheint eine Kontroll-Abfrage. Antwort "JA" löscht das im Editor befindliche Programm unwiderruflich. Antwort "NEIN" kehrt zum Editor zurück.

==> >Shift<-Taste und >F4<-Taste gleichzeitig drücken oder "New" im Editor-Menü anklicken.

Block-Operationen:

Wurde mit "Blk Sta" und "Blk End" ein Textblock definiert, so erscheint das Block-Menü. Fehlt eine der beiden Definitionen so erscheint "Block ???".

==> >F4<-Taste drücken oder "Block" im Editor-Menü anklicken.

Block Kopieren:

Ein Duplikat des definierten Blockes wird an die Stelle unmittelbar über der aktuellen Cursor-Position übertragen. Der Originalblock bleibt unverändert an seiner Position.

==> >c<-Taste drücken oder "Copy" im Block-Menü anklicken.

Block bewegen:

Der definierte Block wird an die Stelle unmittelbar über der aktuellen Cursor-Position bewegt. Dabei wird er an der Ursprungsposition gelöscht. Die Blockdefinition wird gleichzeitig aufgehoben.

==> >m<-Taste drücken oder "Move" im Block-Menü anklicken.

Anmerkung: Sollte sich bei Aufruf der beiden vorangegangenen Block-Operationen der Cursor innerhalb des Blocks befinden, wird die Fehlermeldung "Cursor in Block" ausgegeben.

Block speichern:

Der definierte Block wird (wieder mittels File-Select-Box) als ".LST"-Datei auf Diskette abgespeichert. Der Block bleibt im Editor unverändert erhalten.

==> >w<-Taste drücken oder "Write" im Block-Menü anklicken.

Block ausdrucken:

Der definierte Block wird ausgedruckt. Hinweis zu "PROGRAMMLISTING AUSDRUCKEN" gilt hier analog.

==> >l<-Taste drücken oder "Llist" im Block-Menü anklicken.

Blockanfang finden:

Der Cursor wird am Blockanfang positioniert.

==> >s<-Taste drücken oder "Start" im Block-Menü anklicken.

Blockende finden:

Der Cursor wird unmittelbar unter dem Blockende positioniert

==> >e<-Taste drücken oder "End" im Block-Menü anklicken.

Block löschen:

Der definierte Block wird ohne Kontrollabfrage sofort und unwiderruflich aus dem Editor gelöscht. (Also: VORSICHT !)

==> >Control<-Taste und >d<-Taste gleichzeitig drücken oder "Delete" im Block-Menü anklicken.

Anmerkung: Haben Sie sich nach Aufruf des Block-Menüs entschlossen, doch keine Funktion zu benutzen, drücken Sie entweder die "Pfeil oben" oder die "Pfeil unten" Cursor-Taste oder klicken Sie mit dem Mauszeiger unterhalb des Editor-Menüs in das Editions-Fenster. Das gleiche gilt bei "Block ???".

Blockanfang definieren:

Die Zeile, in welcher sich der Cursor befindet, wird zum Blockanfang erklärt.

==> >Shift<-Taste und >F5<-Taste gleichzeitig drücken oder "Blk Sta" im Editor-Menü anklicken.

Blockende definieren:

Die Zeile unmittelbar über dem Cursor wird zum Block-Ende erklärt.

==> >F5<-Taste drücken oder "Blk End" Im Editor-Menü anklicken.

Anmerkung: Wird das Block-Ende über oder in der Block-Anfangs-Zeile definiert, werden alle bis dahin getroffenen Block-Definitionen rückgängig gemacht. Das gleiche geschieht, wenn Sie nach einer Blockdefinition im "Block"-Menü zuerst die Taste "s" drücken, bzw. "S"tart anklicken und anschließend ebenfalls im "Block"-Menü die Taste "m" drücken bzw. "M"ove anklicken.

Ausdruck finden und ersetzen:

Es wird die Eingabe eines Zeichens oder einer Zeichenfolge erwartet, das/die im Programmtext unterhalb des Cursors gesucht werden soll. Anschließend wird die Eingabe eines weiteren Zeichens oder einer Zeichenfolge erwartet, das/die den gesuchten Ausdruck ersetzen soll.

==> >Shift<-Taste und >F6<-Taste gleichzeitig drücken oder "Replace" im Editor-Menü anklicken.

Anm.: Der Find- oder Replace-String kann eine Länge von 60 Zeichen nicht überschreiten. Die Edition des Find- oder Replace-Strings erfolgt wie üblich durch die Tasten >Delete<, >Backspace< und der Cursor-Horizontal-Pfeiltasten. Zudem kann der Cursor mit den Cursor-VertikalPfeiltasten jeweils an den Anfang oder das Ende des Strings bewegt werden. Soll die Stringeingabe abgebrochen werden, löschen Sie evtl. schon eingegebene Zeichen (>Delete< oder >Backspace<) und quittieren Sie mit >Return< oder drücken Sie die >Esc<-Taste und anschließend >Return<. Im Anschluß an die Eingabe kann dann durch Drücken der Tastenkombination >Control< und >f< die Suchfunktion aufgerufen werden. Es wird dabei immer von der Cursorposition aus in Richtung Programmende gesucht. D.h. also, daß über dem Cursor befindliche Textteile nicht durchsucht werden.

Um das gesamte Programm zu durchsuchen, müssen Sie gegebenenfalls den Cursor durch gleichzeitiges Drücken von >Control< und >ClrHome< am Programmanfang positionieren. Die Funktion zeigt nun der Reihe nach alle Positionen an, an denen sich mit dem Suchstring identische Textteile befinden. Dabei wird vorerst noch nicht gegen den Ersatzstring ausgetauscht. Soll der gefundene Ausdruck mit dem Ersatzausdruck ausgetauscht werden, drücken Sie die Tastenkombination >Control< und >r<. Anschließend kann mit >Control< und >f< oder auch mit >Control< und >r< weitergesucht werden. Allerdings müssen Sie, wenn Sie mit >Control<->r< weitersuchen, aufpassen, daß Sie nicht ungewollt zweimal hintereinander die "Replace"-Funktion aufrufen. Damit würde der nächste gefundene Ausdruck sofort ersetzt. Sollte das dennoch geschehen, können Sie den alten Ausdruck mit "Undo" wieder zurückholen, solange sich der Cursor noch in der betreffenden Zeile befindet.

Achtung: Da der Editor selbstständig alle Anfangsbuchstaben von Befehlen und Variablen in Großbuchstaben umwandelt, sind diese in der Suchfunktion ebenfalls mit großen Anfangsbuchstaben anzugeben.

Ausdruck finden:

Es wird die Eingabe eines zu suchenden Zeichens oder einer zu suchenden Zeichenfolge erwartet. Die Erläuterungen zu "AUSDRUCK FINDEN UND ERSETZEN" gelten hier analog. Gesucht wird durch gleichzeitiges Drücken von >Control< und >f< .

==> >F6<-Taste drücken oder "Find" im Editor-Menü anklicken.

Anmerkung: Werden nach Aufruf der REPLACE- oder FIND-Funktion keine Zeichen eingegeben und mit >Return< bestätigt, so wird die Funktion ohne Auftrag beendet und/oder der letzte REPLACE- oder FIND-Auftrag wird gelöscht.

Seite rückwärts:

==> >Shift<-Taste und >F7<-Taste gleichzeitig drücken oder "Pg up" im Editor-Menü anklicken. (s. auch bei Cursor-Funktionen).

Seite vorwärts:

==> >F7<-Taste drücken oder "Pg down" im Editor-Menü anklicken. (s. auch bei Cursor-Funktionen)

Editor-Zeilenhöhe ändern:

Der Editor-Text kann entweder mit einer Schrifthöhe von 8 Pixels (PICTure ELEmentS, engl.: Bild-Elemente) in max. 48 Zeilen und 79 Spalten oder mit einer Schrifthöhe von 16 Pixels in max. 23 Zeilen und 79 Spalten dargestellt werden.

==> >Shift<-Taste und >F8<-Taste gleichzeitig drücken oder "Text 16" bzw. "Text 8" im Editor-Menü anklicken

Text-Eingabemodus ändern:

Die eingegebenen Zeichen können verschiedene Auswirkungen auf den schon bestehenden Programmtext haben. Im "Insert"-

Modus ("insert", engl.: einfügen) wird das einzugebende Zeichen an der Cursorposition in den Text eingefügt. Der rechts davon liegende Zeilenrest wird um eine Stelle nach rechts verschoben. Im "Overwrite"-Modus ("overwrite", engl.: überschreiben) wird das Zeichen unter dem Cursor durch das neue Zeichen überschrieben.

==> >F8<-Taste drücken oder "Insert" bzw. "Overwrt" im Editor-Menü anklicken.

Direkt-Modus Einschalten:

Es wird in den sogenannten "Direct"- oder auch "Command"-Modus umgeschaltet (s. auch unter "Der DIREKT-Modus").

==> >Shift<-Taste und >F9<-Taste gleichzeitig drücken oder >Esc<-Taste drücken oder "Direct" im Editor-Menü anklicken.

Ausgabe-Fenster zeigen:

Das zweite Interpreterfenster, das zur Ausgabe von Grafik- oder Textoperationen dient, wird gezeigt. Dieser Vorgang bleibt ohne weitere Auswirkung auf die Editorarbeit. Nach Drücken einer beliebigen Taste oder einem Mausklick unterhalb des Editor-Menüs kehrt der Interpreter zum Editor zurück.

==> >F9<-Taste drücken oder "Flip" im Editor-Menü anklicken.

Programm starten:

Das im Editor befindliche Programm wird nach Strukturkontrolle gestartet.

==> >Shift<-Taste und >F10<-Taste gleichzeitig drücken oder "Run" im Editor-Menü anklicken.

Schleifenstruktur testen:

Der Interpreter prüft, ob alle Schleifen ordnungsgemäß abgeschlossen wurden und kehrt zum Editor zurück.

==> >F10<-Taste drücken oder "Test" im Editor-Menü anklicken.

Anmerkung: Diese Funktion hört sich etwas komfortabler an, als sie tatsächlich ist. Da verschiedene Schleifen-Endbefehle in ihrer Priorität nicht unterschieden

und zugeordnet werden können, können mißverständliche Fehlermeldungen erscheinen. Sehen Sie sich dazu folgendes kleine Beispiel an:

```
Procedure Test
For I = 1 To 100
Return
```

Es ist eindeutig, daß hier ein "Next I"-Befehl zum Abschluß des Schleifenstarts "For I = 1 To 100" gesetzt werden müßte. Es existiert ein Unterrouتينenstart "Procedure Test" und der dazugehörige Rückkehrbefehl "Return". Trotzdem wird statt der erwarteten "For ohne Next"-Meldung "Return ohne Procedure" ausgegeben und der Cursor springt zu der vermeintlichen Fehlerquelle. Das gleiche würde übrigens auch bei einem Programmstart geschehen. Ich empfehle Ihnen daher, um zeitraubendes Suchen der Leckstelle zu umgehen, jedem Schleifenstart unverzüglich den entsprechenden Schleifenendbefehl nachzustellen. Die in der Schleife auszuführenden Befehle können dann mit >Control< - >Insert< dazwischen eingefügt werden. Also: Kein "Procedure" ohne Return; kein If ohne Endif; Kein Loop ohne Do usw.

1.2.3. Weitere Hilfsfunktionen

Letzte Änderung löschen und Programmstruktur ordnen:

Wurde nach Änderung einer Programmzeile diese noch nicht mit >Return< oder einer Cursor-Vertikal-Pfeiltaste verlassen, kann diese Änderung wieder ungültig gemacht werden. Gleichzeitig wird das Programm komplett strukturiert. Wurden Programmteile verändert, ersetzt oder gelöscht, kann es sein, daß die sichtbare Programmstruktur nicht mehr korrekt geordnet ist. Mit dieser Funktion wird das Programm wieder in Ordnung gebracht. Beachten Sie jedoch die Anmerkung zur "Test"-Funktion.

==> >Undo<-Taste drücken

Leerzeile einfügen:

Die aktuelle Cursorzeile wird um eine Position nach unten bewegt. Dadurch entsteht unter dem Cursor eine Leerzeile. Diese Zeile wird dann wie jede andere neue Zeile behandelt. Nach Texteingabe und >Return< als Zeilenabschluß wird solange eine neue Leerzeile eingefügt bis diese dann leer mit >Return<, einer der Cursor-Vertikal-Pfeiltasten oder >Undo< geschlossen wird.

==> >Insert<-Taste drücken.

Aktuelle Cursorzeile löschen:

Es wird die Zeile, in welcher sich der Cursor befindet, komplett und unwiderruflich gelöscht.

==> >Control<-Taste und >Delete<-Taste gleichzeitig drücken.

Zeichen links vom Cursor löschen:

==> >Backspace<-Taste drücken.

Zeichen unter dem Cursor Löschen:

==> >Delete<-Taste drücken.

1.3. Der Direct-Modus - das schnelle Etwas

Da es nicht möglich ist, Befehle und Anweisungen, die auf der Editorebene erteilt werden, auch von dort aus einzeln mit "Return" zu starten, steht dem Programmierer hier eine Eingabemöglichkeit zur Verfügung, die nicht auf den Befehl "Run" wartet, sondern alle Eingaben sofort (direkt) nach deren Formulierung und anschließend Betätigen der Return-Taste ausführt. Es ist hier nicht möglich, Befehlsketten zu bilden, die dann der Reihe nach abgearbeitet werden. Es sei denn, man hat vorher im Programmeditor eine "Procedure" definiert, die dann mit einem "Gosub"-Befehl angesprungen werden kann. For-Next-Schleifen sind also im Direkt-Modus nicht machbar, da sie mindestens zwei getrennte Befehle benötigen. Welche Befehle und Anweisungen hier nicht möglich sind, wird Ihnen der Interpreter anschließend an die Fehleingabe in einer Alert-Box melden.

Das mag alles auf den ersten Blick etwas umständlich wirken, hat jedoch den unübersehbaren Vorteil, daß es dadurch möglich ist, sich eine eigene Bibliothek an Funktionen und Hilfsprogrammen zusammenzustellen, die dann einfach über einen Gosub-Sprungbefehl angesprochen werden können. So kann z.B. die im Editormenü fehlende File-Delete-Funktion spielend simuliert werden. Eine kleine Sammlung an Utilities finden Sie im Kapitel 6.2.

Allerdings sind hier auch zwei kleine Essigtropfen in der Suppe. Wer z.B. dazu neigt, viel mit RAM-Disks zu arbeiten und seinen Interpreter von dort aus gestartet hat, kann unter Umständen damit rechnen, daß der Ausgabebildschirm beim Umschalten in den Direkt-Modus einfach verschwunden ist und außerdem, außer an der erscheinenden ">OK"-Aufforderung, nicht zu erkennen ist, daß man sich nun auf der Direkteingabe-Ebene befindet. Der Editortext bleibt dann evtl. einfach stehen. Diesen Umstand werden viele nicht als besonders störend empfinden, zudem sich mit dem Befehl "Cls" der Bildschirm schnellstens löschen läßt. Es kann jedoch dann zu Schwierigkeiten führen, wenn man sich mühsam eine Grafik zusammengebastelt hat und diese nach Programmende mit "Alternate/Help" auf den Drucker übertragen möchte. Außer dem Editortext wird es dann nichts mehr zu drucken geben. Da in diesem Super-BASIC aber auch beinahe an alles gedacht wurde, braucht man nur an der Programmstelle, wo die Grafik gedruckt werden soll, ein "H" (Abk. für "Hardcopy") einzutippen und sich um den Verlust seiner Grafiken keine Sorgen mehr zu machen.

Der zweite Mangel ist, daß unter ungünstigsten Umständen ein Systemfehler, der zwar gemeldet wurde, aber nicht komplett abgefangen werden konnte (evtl. bei Spoke, Sdpoke, Slpoke und auch bei einfachen Pokes), dazu führen kann, daß der nicht mehr zu verhindernde Absturz erst bei der Umschaltung zum Direct-Modus stattfindet. D.h., daß man nach Programmende denkt, der Interpreter wäre noch intakt (da ja der Editor noch funktioniert) und es nicht für nötig hält, sein Programm abzuspeichern. Das böse Erwachen stellt sich ein, wenn man auf "Esc" drückt oder "Direct" im Editormenü anklickt und sich das System in bescheidener Stille verabschiedet. Um also Herzin-

farktgefährdeten das Leben zu erleichtern, sei hier empfohlen, Programmänderungen vorher abzuspeichern. Aber nochmal zur Beruhigung: Dieser Umstand tritt äußerst selten auf, da dieser Interpreter über ein "fast" perfektes Error-Handling verfügt. Es ist nicht, wie in einem anderen wohlbekannten BASIC-Dialekt, alle 15 bis 30 Minuten mit einem Absturz zu rechnen!

Des weiteren ist im Farbmodus zu beachten, daß die Schriftfarben für den Editor und den Direct-Modus aus verschiedenen Farbregistern bezogen werden. Wenn also über das Register 1 die Ausgabefarbe so eingestellt wurde, daß sie mit dem Hintergrund (Register 0) identisch ist, kann es sein, daß Sie den Editortext (Register 3) noch lesen können und im Direct-Modus keine Schrift mehr zu erkennen ist. Verändern Sie dann ganz einfach die Ausgabefarbe mit Setcolor 1,... , so daß sie sich vom Hintergrund unterscheidet.

Wollen Sie aus dem Direct-Modus wieder zum Editor wechseln, geben Sie einfach "Ed" ein oder drücken Sie auf die "Esc"-Taste und bestätigen Sie die Eingabe mit "Return". Außerdem erreichen Sie den Editor mit der Abbruchfunktion "Control/Shift/Alternate".

Wie Sie im Laufe ihrer wachsenden Routine an diesem Interpreter erkennen werden, kann man angesichts der erheblichen Vorteile dieses Modus die Nachteile fast vergessen.

1.4 Der Run-only-Interpreter - der "kleine" Unterschied

Was sicherlich nicht nur die Profis unter den Lesern interessieren wird, ist ein Programm, das "Run-only"-Interpreter (engl.: Nur-Lauf-Interpreter) genannt wird. Dies ist ein Interpreter, mit dem man Programme nur laden und starten kann. Eine Edition des geladenen BASIC-Programmes ist hiermit nicht möglich. Dieser Run-only-Interpreter ist frei kopierbar und kann also mit den GfA-BASIC-Programmen verbreitet werden. So ist es möglich, diese Programme lauffähig zu machen, ohne den eigentlichen Interpreter besitzen zu müssen.

Auch hier wurde wieder an die Kleinigkeiten gedacht. Es ist nämlich möglich, die mit dem Run-only-Interpreter zusammen auf der Diskette befindlichen ".BAS"-Dateien durch einfaches Anklicken zu starten. Auf diese Weise unterscheiden sich GfA-BASIC-Programme in der Bedienung für den Anwender überhaupt nicht mehr von compilierten oder assemblierten Programmen. Man klickt sie an und sie starten selbsttätig. Was man bei anderen BASIC-Dialekten immer schmerzlich vermissen mußte, ist hier ein Kinderspiel: der "Auto"-Start (er funktioniert übrigens auch bei dem Editor-Interpreter). Allerdings ist vom Programmierer daran zu denken, den Befehl "Quit" oder "System" mit aufzunehmen und eine Programmunterbrechung durch Drücken der Tastenkombination "Control/Shift/Alternate" mit einem "On Break Gosub" so abzufangen, daß das Programm direkt zum Desktop zurückkehrt. Sonst meldet sich nämlich eine File-Select-Box, die den Aufruf eines ".BAS"-Programmes erwartet. Wurde dann nicht daran gedacht, ein kleines ".BAS"-Programm mit dem Einzelbefehl "Quit" oder "System" auf der Diskette abzuspeichern, hat der Anwender keine Möglichkeit mehr, ohne ein "Reset" und erneutes System-Booten zum Desktop zurückzukehren. Hatte der Anwender nämlich eine nicht-residente RAM-Disk installiert und darauf Programme gespeichert, sind diese dann unwiderruflich verloren. Also: Vorsicht!

Ein regelrechter Leckerbissen ist in diesem Zusammenhang ein unscheinbarer BASIC-Befehl. Die einfache Abwandlung des "Save"-Befehls zu einem "Psave", der dann allerdings auf der Direkt-Ebene eingegeben werden muß, hat die fantastische Wirkung, daß das Programm-Listing nicht mehr lesbar ist. Es ist außerdem noch dermaßen verschlüsselt, daß wohl noch einige Zeit vergehen wird, bis ein sogenannter "Cracker" den Code knacken wird. Verblüffenderweise ist selbst über einen Diskettenmonitor außer dem reinen ASCII-Text, der sich z.B. in den Text-Strings befindet, von dem eigentlichen Programm nichts mehr zu erkennen. Der Versuch, ein derart geschütztes Programm zu editieren, hat die Folge, daß der Interpreter seinen Dienst verweigert und neu gebootet werden muß.

Herrliche Zeiten für BASIC-Programmierer!!

2. GfA-ABC

Wie bei jeder Sprache, muß man auch hier erstmal das ABC lernen, um fließend sprechen zu können. D.h., daß man die Grundstrukturen, an denen sich die Sprache orientiert, beherrschen muß, um vom Empfänger (in diesem Fall dem Interpreter) richtig verstanden zu werden. Glücklicherweise haben wir es hier mit BASIC zu tun, dem ja der Ruf anhängt, ein Tausend-sassa zu sein, was seine syntaktische Toleranz angeht. Dazu sollen in diesem Kapitel einige Erläuterungen folgen.

2.1 Strukturiertes Programmieren

Wer sich z.B. mit den Sprachen "C", "Modula" oder "Pascal" beschäftigt hat, dem werden die Eigenarten der strukturierten Programmierung nichts Neues sein. Wer sich allerdings an die herkömmliche Art der BASIC-Programmierung ("Spaghetti-Code") gewöhnt hat, wird hier einige Anregungen finden, die ihm den Umgang mit dem GfA-BASIC erleichtern können.

Als erstes ist zu erwähnen, daß all jene, die ihre BASIC-Gewohnheiten nicht ablegen möchten, ohne weiteres auch in Zukunft auf die alte Art und Weise programmieren können. Es darf dann allerdings kaum der typische Strukturbefehl "Do/Loop" verwendet werden. Dieser Befehl hat die Eigenschaft, keine Bedingungsabfrage implizit zu verwenden. Alle Programmteile, die sich innerhalb einer solchen Schleife befinden, werden entweder solange wiederholt, bis die in einer "Exit If"-Konstruktion angegebene Bedingung erfüllt ist, ein "End", "Ed"-oder "Stop"-Befehl auftaucht oder die Tasten "Control/Shift/Alternate" gedrückt werden. Das bedeutet, daß man sich von vornherein schon klare Vorstellungen von dieser Schleifenkonstruktion gemacht haben muß, um die Ausstiegsbedingungen klar formulieren zu können. Diese Schleifen-Art (Loop, engl.: Schleife/Schlinge) eignet sich besonders dazu, große Programmteile ständig wiederholen zu lassen. Ein Beispiel:

```
Do
  @ P_rozedur Ab
  @ P_rozedur Bc
  @ P_rozedur Cd Exit If Endbedingung erfüllt
Loop
Procedure Ab
  ' alle wichtigen Initialisierungen
Return
Procedure Bc
  ' Hauptprogramm mit End-Bedingungsabfrage
Return
Procedure Cd
  ' Nacharbeiten (z.B. Poke-Restoration oder Daten-Ausgabe)
Return
```

Da die Exit-If-Abfrage auch für andere Schleifen anwendbar ist (For/Next, Repeat/Until, While/Wend), muß die für die Do/Loop-Schleife gültige Abfrage auch unmißverständlich außerhalb einer anderen Schleifenart eingesetzt werden.

Wem dieses als zu umständlich erscheint, der kann wie in anderen BASIC-Dialekten seine Programme ruhig weiter mit GOTO's durcheinanderbringen. Das einzige, worauf er absolut verzichten muß, ist das Aneinanderhängen von ganzen Befehlsfolgen, in denen der Doppelpunkt als Trennungszeichen gilt und das Vorkommen von Zeilennummern. Im GfA-BASIC wird in jeder Zeile jeweils nur ein Befehl akzeptiert. Außerdem werden die Zeilen vom Interpreter selbsttätig in die entsprechende optische Struktur eingeordnet. Es ist eine wahre Freude zu sehen, wie sauber und ordentlich ein derart durchstrukturiertes Programm hinterher aussieht.

Ein weiterer wichtiger Aspekt ist aber, daß bei der Fehlersuche dadurch eine immens große Zeitersparnis auftritt. Befehlszeilen sind ohne Zeichenscrolling auf einen Blick erfaßbar. Zusätzlich wird durch das Einrücken der Zeilen sofort erkennbar, z.B. welches "Endif" zu welchem "If" oder welches "Next" zu welchem "For" gehört. Wie Sie an dem in diesem Buch beschriebenen Grafik-Programm sehen werden, lassen sich sogar sehr komplexe

Programme dadurch überschauen. Anders wäre es auch kaum möglich gewesen, ein solches Programm in nicht ganz vier Wochen zu erstellen.

Zu den Zeilennummern ist zu sagen, daß sie hier absolut überflüssig geworden sind. Den einzigen Sinn, den die Zeilennummern hatten, ist der, Goto- und Gosub-Sprünge auf eine bestimmte Zeile richten zu können. Da wir uns aber in der BASIC-Neuzeit befinden, ist es heute möglich, sogenannte "Labels" einzusetzen. Das sind einfache wahlfreie Formulierungen, die an die Stelle im Programm gesetzt werden, zu welcher später ein Goto-Befehl springen soll. Dieses Label ist in jedem Fall mit einem Doppelpunkt abzuschließen, damit der Interpreter es auch als solches erkennen kann. Zu weiteren Operationen werden solche Labels (auch: Marken) nicht benötigt, da mit einem Gosub-Befehl nur noch zu einer definierten "Procedure" verzweigt werden kann.

Der Wegfall von Zeilennummern macht sich außerdem noch dadurch angenehm bemerkbar, daß man sich keine Gedanken mehr über die Abstände der Zeilennummern machen muß. Wenn also Zeilennummern eingefügt werden mußten und es war keine freie Zeilennummer vorhanden, mußte unter Umständen erst ein aufwendiges "Renumber"-Verfahren eingeleitet werden. Im GfA-BASIC wird die neue Zeile einfach zwischen die bestehenden Zeilen eingefügt. Dem Interpreter beigelegt ist übrigens ein kleines Programm mit dem Namen "ST_KILL.BAS", welches die Aufgabe hat, sämtliche Zeilennummern aus alten ST-BASIC-Programmen zu entfernen und ein ladbares Programm zu erstellen. Den Zeilen, die nicht der GfA-Syntax entsprechen, wird dann ein "==" vorangestellt.

Der vielleicht wichtigste Vorteil der Struktur-Programmierung ist aber der, daß beiläufig während der Programmerstellung immer wieder kleine Unter Routinen "abfallen", die in sich geschlossen sind und dadurch die Möglichkeit bieten, sich nach und nach eine umfangreiche Bibliothek an Hilfsprogrammen und allgemein verwendbaren Prozeduren zusammenzustellen. Das wäre prinzipiell in anderen Programmierarten genauso möglich, nur ergibt sich hier die Gelegenheit dazu erheblich seltener.

Komfortabel wird es dann noch, wenn man diesen Prozeduren (wie im GfA-BASIC) eine fast beliebig lange Parameterliste übergeben kann. Effektiver geht es fast nicht mehr.

2.2. Jetzt geht's los ...

Wie versprochen, hört hier der theoretische Teil auf. Ich habe zur Grundlage dieses Buches ein Grafikprogramm gewählt, da Programme dieser Art wohl am ehesten dazu geeignet sind, die verschiedenartigsten Anwendungsmöglichkeiten der einzelnen Befehle zu demonstrieren. Zum zweiten habe ich festgestellt, daß auch noch so raffinierte Programme anderer Art (z.B. Textverarbeitung oder Tabellenkalkulation) bei weitem nicht soviel Interesse finden wie ein effektvolles Grafikprogramm. Der Einsatz von verschiedenen Grafikeffekten in Programmen aller Art dient hauptsächlich dazu, den Spaß an der Arbeit mit diesen Programmen zu erhöhen. Es sollte nicht vergessen werden, daß eine gewisse "Zirkus"-Wirkung auch den sachlichsten Anwendungsprogrammen dazu dienen kann, sie für alle attraktiver zu machen.

Im folgenden werden Sie also ausschließlich über angewandte GfA-BASIC-Programmierung und den Einsatz verschiedener Grafikeffekte informiert. Dabei ist jedoch zu beachten, daß eine absolut erschöpfende Beschreibung des Programms in einem einzelnen Buch nicht zu verwirklichen ist. Es soll zwar versucht werden, auch den BASIC-Einsteigern soviel Grundlagenkenntnis wie möglich zu vermitteln, aber stellenweise muß doch ein gewisses Maß an Basiswissen vorausgesetzt werden. Wo dieses Wissen allein nicht mehr ausreicht, muß die Kombinationsfähigkeit des Einzelnen das Übrige tun. Die einzelnen BASIC-Befehle werden, soweit sie im Grafik-Programm Verwendung finden, in der Reihenfolge ihres Auftretens erläutert. Die Befehle, die im Programm nicht eingesetzt werden konnten, werden entweder im Zusammenhang mit verwandten Befehlen oder im Anschluß an das Listing erklärt.

Bei solchen Programmpunkten, die in der gleichen oder ähnlichen Form schon einmal aufgetreten sind und dort erklärt wur-

den, wird nicht noch einmal auf das Thema eingegangen. Es sei denn, die aufgetretene Variante bietet genügend Neuerungen, die es sich nochmal zu erklären lohnt.

Eine kurze Bedienungsanleitung zu diesem Programm finden Sie dann in Anhang B. Nun aber: die ersten Programmzeilen - es geht los!

@Pre.loading

@Predefs

@Verteiler

'Gosubs zu den einzelnen Programmteilen.'

GOSUB

Das Kürzel "@" hat exakt dieselbe Bedeutung wie der Befehl "Gosub". Wenn also dieses Zeichen oder "Gosub" vor einem Labelnamen steht, bedeutet das, daß hier zu der angegebenen Prozedur verzweigt wird.

Procedure Pre.loading

'Diese Routine lädt das Titelbild und die Menü-Icons

PROCEDURE / RETURN

Mit der Bezeichnung "Procedure" werden im GfA-BASIC Unterprogramme bezeichnet, in welchen Programmblöcke untergebracht werden, die von mehreren Stellen im Hauptprogramm aus aufgerufen werden sollen. Es ist nicht sehr rationell, bestimmte Programmläufe, die mehrfach im Hauptprogramm erforderlich werden, immer dort wieder neu zu formulieren, wo sie benötigt werden. Dazu werden möglichst allgemein gehaltene Unterprogramme zusammengestellt, die dann von den jeweiligen Stellen aus aufgerufen werden. Eine "Procedure" muß immer mit

dem Rückkehrbefehl "Return" abgeschlossen werden. Mit diesem Befehl kehrt der Interpreter zu der aufrufenden Programmzeile zurück und fährt dort mit der nächsten darauffolgenden Programmzeile fort.

On Break Gosub Schluss

ON BREAK GOSUB

Mit "On Break Gosub" wird eine Procedure bestimmt, zu welcher das Programm verzweigen soll, wenn die Abbruch-Tastenkombination "Control/Shift/Alternate" gedrückt wurde oder ein "Stop"-Befehl auftaucht.

If Exist("\pre_pro.pic")=0

IF / ELSE / ENDIF

Als erstes taucht hier eine Bedingungsabfrage auf. "If" dient dazu, eine Bedingung aufzustellen, von deren Gültigkeit bzw. Ungültigkeit dann weitere Operationen abhängig gemacht werden können. Wie die meisten es von anderen BASIC-Dialekten kennen werden, muß der Frage "If" (wenn) die Bestimmung "Then" (dann) folgen, wobei nach Then die Operationen angegeben werden, die je nach Wahrheitswert der Bedingungsabfrage abgearbeitet werden sollen. Das bedeutete, daß die Kette an Befehlen, die auf Then folgt, meistens durch die mögliche Zeilenlänge begrenzt wird. Im GfA-BASIC ist es möglich, auch umfangreiche Programmteile von der Erfüllung einer bestimmten Bedingung abhängig zu machen. Es braucht nach If kein Then mehr angegeben zu werden. Die Folge an Befehlen, die abhängig von If ausgeführt werden sollen, wird hier ausschließlich von dem nächsten auf die If-Abfrage folgenden Befehl eingegrenzt, bzw. ist es auch möglich, in die Abfrage den Befehl "Else" (sonst) einzusetzen, nach dessen Auftreten die Programmteile angegeben sind, die ausgeführt werden sollen, wenn sich die If-Abfrage als unwahr erweist.

```
z.B.  If (Bedingung)
      (Programmfolge, wenn Bedingung wahr)
      Else
      (Programmfolge, wenn Bedingung unwahr)
      Endif
```

Bedingungsabfragen können beliebig verschachtelt werden, wobei darauf zu achten ist, daß sich das jeweils auftretende Else oder Endif nur auf das direkt davor auftretende If bezieht.

EXIST

Der zweite Befehl, "Exist", erfragt, ob auf der angegebenen Diskette eine Datei mit dem in Klammern angegebenen Namen vorhanden ist. Dabei wird entweder eine 0 (nicht vorhanden) oder -1 (vorhanden) zurückgegeben. Dieser Wert kann in eine Variable überführt werden (A=Exist(Dateiname)) oder, wie hier in einer Abfrage direkt verwendet werden.

Die hierarchische Struktur der Pfadangabe sieht wie folgt aus:
Diskettenstation:\Ordner\Unterordner\Dateiname.Extender

```
z.B.:  B:\UTILITY\OUTPUTS\DRUCKE.WAS
```

Damit wird nur auf der Disk B innerhalb des Unterordners des Ordners nach der Datei "DRUCKE.WAS" gesucht.

Weitere Varianten:

B:\UTILITY*.*	==> sucht nur im Ordner nach der ersten auftretenden Datei
B:\UTILITY\OUTPUTS*.*	==> sucht nur im Unterordner nach der ersten Datei mit dem Extender ".WAS"
.	==> sucht auf der aktuellen Disk nach der ersten auftretenden Datei mit dem Extender ".WAS"

<code>\DR????</code>	==> sucht auf der aktuellen Diskette nach allen Dateien mit einem sechs Buchstaben langen Namen, der mit "DR" beginnt
<code>A:\DR?????.WAS</code>	==> sucht auf Disk A nach allen Dateien mit einem sechs Buchstaben langen Namen, der mit "DR" beginnt und den Extender ".WAS" trägt.
<code>\D*.*</code>	==> sucht auf der aktuellen Diskette nach allen Dateien, die mit dem Buchstaben "D" beginnen

Alle möglichen Kombinationen in der Pfadstruktur hier anzugeben, würde etwas zu weit führen. Probieren Sie es einfach aus.

Goto Enter.1

'Wenn also das Titelbild nicht gefunden wird, überspringe den Ladeprozess und gehe zu Label "Enter.1"

GOTO label

Dieses ist in alten BASIC-Dialekten einer der am häufigsten angewandten Befehle. Im GfA-BASIC wird sich seine Verwendung in Grenzen halten. Er bewirkt nichts weiter, als daß das Programm an der Stelle, wo es auf diesen Befehl trifft, zu der Stelle verzweigt, die mit dem angegebenen Labelnamen markiert ist. Dieses Label kann sich aus beliebigen Zeichen zusammensetzen. Genau wie bei Prozedur-Namen ist es auch möglich, eine Ziffer als erstes Zeichen zu verwenden. Der Labelname muß mit einem nachgestellten Doppelpunkt abgeschlossen werden.

z.B.: 1.label:

Sprünge in oder aus For/Next-Schleifen oder Prozeduren sind übrigens nicht erlaubt. Der Versuch wird vom Editor mit einer Fehlermeldung bestraft.

Endif

@Hidemouse

'Titelbild wurde gefunden.

'Schalte die Maus ab, da sonst von der Mausmaske ein häßliches Loch

'im Bild hinterlassen wird.

Bload "pre_pro.pic",Xbios(2)

'Lade das Titelbild in den Bildschirmspeicher.

BLOAD

Mit "Bload" werden Datenblöcke, die zuvor mit dem Befehl "Bsave" auf Diskette gespeichert wurden, in den Arbeitsspeicher zurückgeladen. Dabei wird ein Parameter angegeben, der bestimmt, an welche Stelle des Speichers geladen werden soll. Die Länge des Blockes wurde vorher mit Bsave definiert.

Es ist vor dem Laden sicherzustellen ,daß die zu ladende Datenmenge auch tatsächlich ab der angegebenen Speicherstelle untergeracht werden kann ohne in wichtige Bereiche hineinzuschreiben. Außerdem wird der Name der zu ladenden Datei übergeben. Die schon beschriebenen Pfad-Hierarchie ist auch hier gültig.

In diesem Fall soll ein Bild an die Anfangsadresse des Bildschirmspeichers geladen werden (der Xbios-Aufruf mit der Funktionsnummer 2 liefert diese Adresse).

@Showmouse

'Schalte die Maus wieder an.

Enter.1:

Dim M.icon\$(25)

'Hier wird ein eindimensionales Stringfeld (engl.: Array) mit dem Namen "M.icon\$" auf 25 Elemente festgelegt, welches die Menü-Icon-Daten aufnehmen soll.

DIM

"Dim" hat die Funktion, ein- oder mehrdimensionale Datenfelder anzulegen. Die Anzahl der Feldelemente wird bei mehrdimensionalen Feldern auf 65535 beschränkt. Bei eindimensionalen Feldern sind soviel Elemente erlaubt, wie freier Speicherplatz vorhanden ist. Zur Verwendung bei Dimensionierungen sind alle Arten von numerischen Variablen, sowie Bool'sche Variablen und Textvariablen erlaubt.

Aufbau eines mehrdimensionalen Feldes:

Stellen Sie sich bitte einen Schrank vor. Dieser Schrank wird nun in zwei Hälften geteilt. Eine rechte und eine linke Hälfte. Innerhalb der Hälften sind Schubladen untergebracht. Nehmen wir an, jede Hälfte besitzt zwei Schubladen. Diese Schubladen werden nun wiederum in einzelne Fächer unterteilt. Jede Schublade erhält drei Fächer. Wir haben nun also einen Schrank mit 12 Fächern eingerichtet. Die entsprechende Dimensionierung dazu:

```
Dim Schrank$(1,1,2)
```

Sie wundern sich evtl., warum hier nicht (2,2,3) steht. Das hat den Grund, daß diese Arrays immer mit der 0 zu zählen beginnen. So wird in diesem Fall vorausgesetzt, daß jeweils unter der niedrigsten positiven Zahl ein Null-Element vorhanden ist. Die angegebene Zahl bedeutet hier: dimensioniere bis Element X

```
Also:      (0+1,0+1,0+1+2) = (1,1,2)
```

In diesem Schrank sollen nun verschiedene Dinge untergebracht werden. Das können Werte, oder wie hier Textzeichen bzw. -strings sein. Gehen wir mal davon aus, daß die linke Schrank-

hälfte den Oberbegriff "Werkzeug" und die rechte den Oberbegriff "Bücher" erhält. Den Schubladen der Werkzeugseite geben wir jetzt die Aufschriften:

1. Schublade = Zangen
2. Schublade = Schraubenzieher

Die Schubladen der Bücherseite bekommen die Namen:

1. Schublade = Krimis
2. Schublade = Natur

Nun kann man den jeweils drei Fächern in den Schubladen wiederum Namen geben.

In der Schublade "Zangen":

Fach 1 = Kneifzange / Fach 2 = Rohrzange / Fach 3 = Spitzzange

In der Schublade "Schraubenzieher":

Fach 1 = kleiner S. / Fach 2 = mittlerer S. / Fach 3 = großer S.

In der Schublade "Krimis":

Fach 1 = englische K. / Fach 2 = deutsche K. / Fach 3 = amerik. K.

In der Schublade "Natur":

Fach 1 = Blumen / Fach 2 = Tiere / Fach 3 = Berge

Bis hierhin sieht das vielleicht etwas übertrieben aus. Der Sinn der Sache ist aber der, daß nun anhand von sogenannten "Indizes" (das sind eigentlich die Namen der Schubladen und Fächer) auf jedes einzelne Fach zugegriffen werden kann. Stellt man sich nun anstatt Blumen, Büchern u.s.w. Speicherplätze im Computer vor, so kann man sich nun durch die Angabe der Indizes jederzeit Informationen über den Inhalt der Plätze verschaffen. Dieses ist vor allem dann wichtig, wenn Informationen abgelegt werden, die bestimmten "Familien" zugeordnet werden sollen. Um nun z.B. den Inhalt des Faches "Blumen" zu ermitteln, müßte man durch "Print Schrank\$(1,1,0)" oder "A\$=Schrank\$(1,1,0)" dieses Fach ansprechen. "Bücher" ist die rechte Schrankseite, also Element 1 (Element 0 ist die linke Werkzeugseite) der ersten Dimension. "Natur" ist die zweite

(untere) Schublade dieser Seite, also Element 1 (Element 0 ist die obere Schublade dieser Seite) der zweiten Dimension. Und "Blumen" ist das erste Fach in dieser Schublade, also Element 0 (Element 1 ist das Fach "Tiere" ; Element 2 ist das Fach "Berge") in der dritten Dimension.

Dieses Spiel ließe sich nun mit weit mehr Dimensionen fortführen. Aus dem Beispiel ist hoffentlich das Prinzip der mehrdimensionalen Feld-Einteilung erkennbar geworden.

DIM ?

"Dim?(Feldvariable())" übergibt die Anzahl der Dimensionierungen, die für die angegebene Feldvariable eingerichtet wurden. Dieser Wert kann auf verschiedene Arten ermittelt werden:

z.B.:

```
Dim Feld%(2,3,4)
A%=Dim?(Feld%())
Print A%
If Dim?(Feld%())>0
    Print "Das Feld besitzt :";Dim?(Feld%());" Elemente"
Endif
Print Dim?(Feld%())
```

Auf dem Bildschirm erscheint:

```
60
Das Feld besitzt :60 Elemente
60
```

Da: $3 * 4 * 5 = 60$

Wird Dim?(Var()) mit einer Variablen aufgerufen, die vorher nicht dimensioniert wurde, wird der Wert 0 zurückgegeben.

ERASE

Hiermit kann ein durch "Dim" definiertes Variablenfeld mitsamt seiner Dimensionierung wieder gelöscht werden (Erase A_rrayname()). Seinen Sinn zeigt dieser Befehl hauptsächlich bei zwei Anwendungsmöglichkeiten. Die erste ist dann gegeben, wenn der Arbeitsspeicher weitgehend ausgelastet ist und es auf jedes freie Byte ankommt, wenn man die Fehlermeldung "Speicher voll" vermeiden will. Felder, die nach ihrer Verwendung nicht mehr benötigt werden, sollten sofort wieder mit diesem Befehl gelöscht werden, um den durch sie belegten Speicherplatz wieder dem Programm zur Verfügung zu stellen. Die zweite Möglichkeit ist, wenn in allgemein verwendbaren Prozeduren (Utilities) Felder verwendet werden. Da nicht erwartet werden kann, daß für jedes Utility die Dimensionierungen im Hauptprogramm vorgenommen werden, sollten diese also in der Prozedur selbst erfolgen. Nach Abschluß der Arbeiten in dieser Prozedur wird das Feld wieder gelöscht, um beim nächsten Aufruf wieder neu dimensioniert werden zu können.

Defmouse 2

DEFMOUSE

Defmouse legt die Darstellung des Mauszeigers fest

- 0 = Pfeil
- 1 = Doppelklammer (verzerktes X)
- 2 = die fleissige Biene
- 3 = Hand mit ausgestrecktem Zeigefinger
- 4 = offene Hand
- 5 = Fadenkreuz fein
- 6 = Fadenkreuz grob
- 7 = Fadenkreuz weiß mit Umrandung

Über einen Stringparameter kann eine freie Mausform definiert werden.

z.B. Defmouse M_ausdaten.1\$

Dabei ist zu beachten, daß alle Daten als 2Byte-Werte (Word= 0-65535) übergeben werden. Um Dezimalzahlen in Hexadezimalwerte zu übersetzen, wird hier am besten die Funktion "Mki\$" verwendet, da diese die Werte gleichzeitig in ein Stringformat mit zwei Byte verwandelt. Die Mausform besteht aus einem Raster mit 16*16 Pixels.

Der String muß wie folgt organisiert sein:

Das erste Word legt die X-Koordinate des Aktionspunktes fest.

Das zweite Word die Y-Koordinate.

Das dritte Word wird immer auf 1 gesetzt.

Das vierte Word legt die Maskenfarbe (Hintergrund des Mausbildes) fest (0 = weiß / 1 = schwarz).

Das fünfte Word die Cursorfarbe (Mausbild) 0 oder 1.

Die Words 6 bis 21 beinhalten das Bitmuster der Maske.

Die Words 22 bis 47 beinhalten das Bitmuster des Mausbildes.

Beispiel:

```

Dim Masken_zeile$(15),F_ormzeile$(15)
Restore Mausdatas.maske
For I%=0 To 15
  Read D.atm%
  Masken_zeile$(I%)=Mki$(D.atm%)
  Mausmaske$=Mausmaske$+Masken_zeile$(I%)
Next I%
Restore Mausdatas.form
For I%=0 To 15
  Read D.atf%
  F_ormzeile$(I%)=Mki$(D.atf%)
  Mausform$=Mausform$+F_ormzeile$(I%)
Next I%
M_ausdaten.1$=Mki$(1)+Mki$(65535)+Mki$(1)+Mki$(0)+Mki$(1)+
Mausmaske$+Mausform$
Defmouse M_ausdaten.1$
Deffill 1,2,8
Pbox 100,100,200,200
Repeat

```

```
Mouse X,Y,K
If Point(X,Y)=0
  Color 1
Else
  Color 0
Endif
Plot X,Y
Until K=1
Mausdatas:
Mausdatas.maske:
Data 112,248,508,1022,2047,4095,8191,8190,16380
Data 16376,32752,32736,65472,65280,64512,61440
Mausdatas.form:
Data 32,80,168,332,658,1317,2634,3220,5416
Data 4688,8608,8384,21248,19456,61440,49152
Edit
```

Das Beispiel erzeugt das Bild eines Stiftes, mit dem Sie nun Striche auf dem Bildschirm zeichnen können. Dieser Stift ist dabei auch auf schwarzem Untergrund erkennbar. Außerdem zeichnet er dort, wo er auf schwarzen Untergrund trifft, in weiß und dort, wo der Hintergrund weiß ist, in schwarz. Die Werte für die einzelnen Bildzeilen werden zuerst einem Feld mit 16 Elementen (=16 Words) zugeordnet und dann zu einem String zusammengefaßt. Anschließend wird der 37 Words lange Hauptstring gebildet. Hier hat der Aktionspunkt die Werte $X=1/Y=65535$, das heißt, daß er in der linken, unteren Ecke des Bildes liegt ($X=1/Y=1 \Rightarrow$ linke, obere Ecke ; $X=65535/Y=1 \Rightarrow$ rechte, obere Ecke ; $X=65535/Y=65535 \Rightarrow$ rechte, untere Ecke). Die Maskenfarbe hat den Wert 0 (=weiß) und die Formfarbe den Wert 1 (=schwarz). Im Grafikprogramm befindet sich ein Füllmuster-Editor, der die Werte des editierten Musters ausgibt. Dieser Editor läßt sich also auch zum Entwurf eines Mauszeigers verwenden, da das Füllmuster sich ebenfalls aus 16×16 Bildpunkten zusammensetzt. Es ist zu beachten, daß der Interpreter automatisch wieder den Pfeil als Mausform einsetzt, sobald eine File-Select- oder Alert-Box aufgerufen oder das Programm beendet wird. Die selbstdefinierte Mausform muß

also danach mit Defmouse immer wieder neu eingeschaltet werden. In diesem kleinen Beispiel finden Sie schon eine Fülle an Befehlen, die Ihnen später noch ausführlich erklärt werden.

SPRITE

Der Atari ST bietet die Möglichkeit, Sprites zu definieren und zu aktivieren. GfA-BASIC hat die Chance nicht ungenutzt gelassen, und bietet einen entsprechenden Befehl dazu an. Grundsätzlich sind diese Sprites nichts anderes als der Mauszeiger, der ja ebenfalls den Charakter eines Sprites besitzt, nur daß dieser Sprite von den Bewegungen der Maus gesteuert wird. Mit dem Befehl "Sprite" können Sie nun einen, oder auch mehrere Sprites definieren und nach Ihren Wünschen auf dem Bildschirm bewegen. Der Aufbau eines solchen Sprites ist dem eines Mauszeigers (s. "DEFMOUSE") sehr ähnlich. Die Spriteform besteht ebenfalls aus einem Raster mit 16*16 Pixels und es wird ebenfalls ein String übergeben, in dem die Sprite-Werte als Mki\$-Strings aneinandergefügt werden.

Die Organisation eines Spritestrings sieht folgendermaßen aus:

Das erste Word bestimmt die X-Koordinate des Aktionspunktes

Das zweite Word die Y-Koordinate

Das dritte Word kann entweder 0 (normal) oder 1 (XOR) beinhalten

Das vierte Word bestimmt die Maskenfarbe (0=weiß / 1=schwarz)

Das fünfte Word die Cursorfarbe (0=weiß / 1=schwarz)

Die Words 6 bis 47 werden nun abwechselnd mit den einzelnen Pixelzeilen von Maske und Bild gefüllt.

Dieser String, sowie die Koordinaten der Position, an welche der Sprite gesetzt werden soll, werden nun dem Befehl übergeben.

z.B.:

```
Dim B(32)
Aa$=Mki$(1)+Mki$(16)+Mki$(1)+Mki$(0)+Mki$(1)
Restore S.pritedatas.form
```



```
For I=2 To 32 Step 2
  Read B(I)
Next I
Restore S.pritedatas.maske
For I=1 To 31 Step 2
  Read B(I)
Next I
For I=1 To 32
  Aa$=Aa$+Mki$(B(I))
Next I
For I=50 To 60 Step 2
  For J=100 To 539
    Sprite Aa$,J,I
    Plot J-1,I
  Next J
Next I
U=Inp(2)
Sprite Aa$
Edit
S.pritedatas.maske:
Data 112,248,508,1022,2047,4095,8191,8190,16380
Data 16376,32752,32736,65472,65280,64512,61440
S.pritedatas.form:
Data 32,80,168,332,658,1317,2634,3220,5416
Data 4688,8608,8384,21248,19456,61440,49152
```

Der Einfachheit halber wurden hier dieselben Datas wie zur Definition des Mauszeigers verwendet. In diesem Beispiel wird das Sprite ständig bewegt. Es ist jedoch auch möglich, ein Sprite einfach auf den Bildschirm zu setzen, und es bei Bedarf wieder vom Bildschirm zu entfernen. Gesetzt wird der Sprite wie oben gezeigt. Gelöscht wird er einfach wieder, indem der "Sprite"-Befehl mit dem String, der die Daten des zu löschenden Sprites enthält, jedoch ohne Koordinatenangabe, neu aufgerufen.

EDIT

Dieser Befehl ist eigentlich völlig identisch mit dem Befehl "End", der ja ein Programm an der Stelle, wo er auftaucht, abbricht. Im GfA-BASIC wird aber bei End eine zusätzliche Sicherheitsabfrage durch den Interpreter ausgeführt, ob das Programm tatsächlich beendet werden soll. Erst nach Betätigen des entsprechenden Buttons kehrt der Interpreter dann zum Editor zurück. Mit Edit läßt sich nun diese Abfrage ausschalten, so daß bei Erscheinen dieses Befehls direkt zum Editor zurückgekehrt wird.

END

Siehe unter "EDIT".

Diese beiden Abbruchbefehle schließen den Programmlauf vollständig ab. Er kann danach nicht durch "Cont" fortgesetzt werden. Außerdem werden nach diesen Befehlen alle noch offenen #-Dateien wieder geschlossen.

STOP

Mit "Stop" kann der Programmlauf an jeder Stelle unterbrochen werden. Es werden keine Dateien geschlossen und das Programm kann durch die Eingabe von "Cont" im Direktmodus in der Zeile nach dem Stop-Befehl fortgesetzt werden.

CONT

Siehe unter "STOP".

"Cont" kann nicht eingesetzt werden, wenn nach Programmabbruch durch "Stop" entweder der "Clear"-Befehl verwendet, das Programm verändert, oder neue Variablen eingeführt wurden.

QUIT / SYSTEM

Diese beiden Befehle sind absolut identisch. Eine mittlerweile schon alte BASIC-Unart, beide Befehle für ein und dieselbe Funktion zur Auswahl zu stellen. "Wer die Wahl, hat die Qual!" Die Funktion dieser beiden Befehle, denen keinerlei Parameter übergeben werden, ist ganz einfach die, daß der Interpreter, sobald ihm einer davon begegnet, das Programm beendet, BASIC BASIC sein läßt und geradewegs zum Desktop zurückkehrt.

CLEAR

Wollen Sie an einer bestimmten Stelle des Programms alle Variableninhalte löschen, können Sie das mit dem Befehl "Clear" tun. Dieser Befehl wirkt sich auf sämtliche Variablen und Felder aus. Die Inhalte lassen sich hinterher nicht mehr restaurieren. Feld Dimensionierungen werden ebenfalls gelöscht. Am Programm-anfang ist dieser Befehl nicht nötig, da durch "Run" generell alle Variablen, sowie der Bildschirm gelöscht werden.

RUN

Der "Run"-Befehl startet das aktuelle BASIC-Programm von der ersten Zeile an. Dabei werden sämtliche Variablen und der Bildschirm gelöscht. "Run" kann im Programm enthalten sein oder im Direktmodus eingegeben werden.

NEW

Mit diesem Befehl können Sie an jeder Stelle des Programms dasselbe mitsamt seinen Variablen aus dem Arbeitsspeicher löschen. Er ist ebenfalls im Direktmodus möglich.

Restore *Il_text*

'Lese die nächsten Datas (Icon-Dateinamen) ab der Marke "*Il_text*"

RESTORE

"Restore" ohne Angabe einer Marke bewirkt, daß grundsätzlich an den den Anfang aller Data-Werte die im Programm enthalten sind, gesprungen wird. Zum anderen kann man mit dem Befehl "Restore Marke" erreichen, daß sich der Data-Lesezeiger auf den Anfang des Datablockes richtet, der auf die angegebene Sprungmarke folgt. Ab dort werden dann die gewünschten Data-Werte eingelesen. Es müssen dann bis zur letzten Zeile des Programms noch so viele Data-Werte vorhanden sein, wie abgefragt werden. Wenn nicht, wird eine Fehlermeldung ausgegeben.

```
E.xis=Exist("\micon_1.icn")
```

```
'Hier wird wieder mit "Exist" abgefragt, ob sich das erste der Menü-  
'Icons auf der Diskette im angegebenen Ordner befindet. Wie bereits  
'erwähnt, kann der zurückgegebene Wert (-1 oder 0) in einer Variablen  
'gespeichert werden.
```

```
If E.xis=0
```

```
'Wenn nicht, schalte Pfeilform wieder ein und überspringe den Lade-  
'prozess
```

```
Defmouse 0
```

```
Goto Enter.2
```

```
Endif
```

```
@D.size("\micon_1.icn")
```

```
'Icon-Dateien wurden gefunden. Gehe zur Unter-Routine "D.size"  
'und ermittle dort die Größe der Icon-Strings in Bytes.
```

GOSUB "Procedure"(Parameterliste)

PROCEDURE "Name"(lokale Variablenliste)

Hier wird eine Gosub-Variante angewandt, die sicher zu den "Highlights" dieses BASIC's zu zählen ist. GfA-BASIC bietet Ihnen die Möglichkeit, an eine Prozedur eine beliebig lange Parameterliste zu übergeben. Diese Parameter werden dort einer Liste von Variablen zugeordnet, die auf den Prozedurnamen (z.B. Procedure 1.Test(Var,Var\$)) folgt. Am Beispielfall ist zu erkennen, daß diese Variablenliste sowohl numerische, als auch Textvariablen beinhalten kann. Die Variablen "Var" und "Var\$" gelten innerhalb der dazugehörigen Procedure als lokale Variablen. D.h., daß die in ihnen enthaltenen Werte oder Strings nur innerhalb dieser Procedure eingesetzt oder abgefragt werden können. Benennen Sie außerhalb dieser Prozedur Variablen mit dem gleichen Namen, so ist der Interpreter in der Lage, diese von den prozedur-internen Variablen zu unterscheiden. Außerdem können Prozedurnamen allgemein, im Gegensatz zu Variablennamen auch eine Ziffer als erstes Zeichen erhalten.

Beispiel:

```

Var.1$="Global"
X%=444
A%=147
B$="Var-Text"
Print "Variableninhalt innerhalb der Procedure :"

```

```

Print "Variable Y% : ";Y%
Endif
Print X%'Y%'Aa%'Bb$'Cc%'Dd$'Var.1%'Var.1$'A%
Return

```

Bitte studieren Sie dieses Beispiel geduldig. Es ist schwierig, innerhalb eines kleinen Exempels die komplexe Struktur lokaler und globaler Variablen oder eines rekursiven Aufrufes zu erklären. Anhand des Beispiels läßt sich jedenfalls erkennen, daß der Interpreter zwischen globalen (im ganzen Programm gültigen) Variablen und lokalen (nur in der jeweiligen Prozedur gültigen) Variablen unterscheidet. Weiter läßt sich erkennen, daß sowohl Variablen, als auch Werte, bzw. Texte an die Prozedur übergeben werden können. Außerdem ist es möglich, direkt innerhalb der Prozedur Variablen als lokal zu deklarieren, die dann wie die Parameter-Variablen nur dort verwendet werden können. Sollen also innerhalb der Prozedur mit lokalen Variablen verschiedene arithmetische oder sprachliche Konstrukte gebildet werden, müssen diese dort vor Verlassen der Prozedur in globale Variablen übergeben werden, um ihre Inhalte außerhalb der Prozedur erfragen oder weiterverwenden zu können.

LOCAL

Wie eben erwähnt, können innerhalb von Prozeduren Variablen als lokale Variablen deklariert werden. Lesen Sie bitte dazu unter "GOSUB"/"PROCEDURE". Mit dem Befehl "Local Var1,Var2,Var3,Var4" läßt sich auch eine ganze Liste von Variablen als lokal erklären.

```
Icnflag%=1
```

```
'Merke Dir, daß Icons geladen wurden.
```

```
For I%=1 To 25
```

```
'Schleife zum Einlesen der Namen und Laden der Menü-Icon-Dateien.
```

FOR / TO / DOWNTO / STEP / NEXT

Mit der Konstruktion For/Next wird eine Wiederholungsschleife angelegt. Es muß eine numerische Variable angegeben werden, die zur Aufnahme des Zählerwertes dient. Als nächstes wird nach dem Gleichheitszeichen ein Startwert angegeben, mit dem der Zähler beginnt. Als nächstes folgt nach dem "To" der Endwert. Wenn der Zähler diesen Wert erreicht hat, wird die Schleife bei "Next" verlassen. Mit "Step" kann die Schrittweite des Zählers eingestellt werden. "Next" markiert den Wendepunkt der Schleife. Alle Programmzeilen, die zwischen der Zählweisung "For" und der dazugehörigen Wendemarke "Next" stehen, werden solange wiederholt, bis der Schleifen-Endwert erreicht ist.

Danach wird das Programm mit der auf die "Next"-Anweisung folgenden Programmzeile fortgesetzt.

Die For/Next- Schleife hat im GfA-BASIC eine besondere Variante. Es ist nämlich möglich, diese Schleife rückwärts zählen zu lassen, ohne das übliche "Step -(Schrittweite)" einsetzen zu müssen. Die Konstruktion sieht dann so aus:

```
For Startwert Downto Endwert
```

Die Schleife zählt dann in Einerschritten rückwärts. Diese Variante ermöglicht es allerdings nicht, mit "Step" die Schrittweite festzulegen. Diese ist hierbei immer 1. Sollen andere Schrittweiten eingesetzt werden, muß man wieder auf die altbewährte Konstruktion

```
For Startwert to Endwert Step -Schrittweite
```

zurückgreifen. Wenn es irgend möglich ist, sollten für die Zählvariable (hier: I%) Intervariablen (das Zeichen % deklariert die angegebene Variable als Typ Integer) verwendet werden, da sich die Durchlaufzeit dadurch bei kleinen und nicht so arbeitsaufwendigen Schleifen erheblich verkürzt. Außerdem benötigen Integer-Variablen grundsätzlich nur 2/3 des Speicherplatzes einer Fließkommavariablen. Eine Fließkommavariablen benötigt

zur Speicherung 6 Byte und eine Integer-Variable nur 4 Byte. Also auch bei allgemein verwendeten Variablen, ist der Typ Integer dem Typ Fließkomma, wenn möglich, vorzuziehen. Für die, die es nicht wissen: Eine Integer-Variable ist eine numerische Variable, die ausschließlich Integerwerte aufnehmen kann. D.h., daß selbst wenn dieser Variablen Realzahlenwerte zugewiesen werden, die Nachkommastellen dieses Wertes in die nächstkleinere Ganzzahl integriert werden (s. auch unter "INT").

```
z.B. Var.1 = 63.449
     Var.2% = Var.1
     Print Var.1''Var.2%
```

Das Ergebnis: 63.449 63

Hier ist schon wieder eine GfA-Besonderheit aufgetreten. Das bekannte Kürzel " ' " für "Rem" (Remark;engl.: Bemerkung) kann, solange es sich nicht am Zeilenanfang befindet, dazu benutzt werden, bei der Variablen- oder Textausgabe mit dem Print-Befehl zwischen oder nach den Variablen oder Texten als Leerzeichen zu dienen. Es werden dann an Stelle dieser Kürzel entsprechend viele Leerzeichen eingefügt. Bei der Ausgabeformatierung ist diese Möglichkeit von nicht zu unterschätzendem Wert.

```
M.icon$(I%)=Space$(Datsize%)
```

'Da mit Bload gearbeitet wird, fülle die Länge des Strings
'mit Leerzeichen auf, um Platz zu schaffen.

SPACES

Mit dem Befehl "Space\$(Anzahl)" ist es möglich, eine Textzeile oder eine Stringvariable mit der angegebenen Anzahl an Leerzeichen (Chr\$(32)) aufzufüllen oder die Leerzeichen einzufügen.

```
z.B.: Ausdruck$="ABC"+Space$(10)+"XYZ"
```


STRING\$

Derselbe Effekt kann mit dem Befehl " String\$(Anzahl," ") " erreicht werden, nur dieser Befehl bietet noch eine Möglichkeit mehr. Als zweiter Parameter hinter "Anzahl" wird ein Textausdruck übergeben, der so oft aneinandergeschaltet wird, wie als Wert in "Anzahl" angegeben wurde.

```
z.B.:  A$=String$(6,"OKAY ")
        Print A$
```

```
oder   Print String$(6,"OKAY ")
```

Es erscheint beide Male: OKAY OKAY OKAY OKAY OKAY OKAY

```
Oder   A$=="==> "+String$(10,"X")+" <=="
        Print A$
```

Es erscheint : ==> XXXXXXXXXXXX <==

UPPER\$

Dieser Stringfunktion wird ein Stringausdruck oder eine Stringvariable übergeben, deren Inhalt nach kleingeschriebenen Buchstaben durchsucht wird. Werden solche gefunden, wandelt sie diese in Großbuchstaben (engl.: upper case) um.

z.B.:

```
Print Upper$("Das große GfA-BASIC-Buch")
A$="Morgenstund hat Gold im Mund."
Print Upper$(A$)
B$=Upper$("Atari 520 ST+")
Print B$
```

Read Iconload\$

'Lese die einzelnen Dateinamen

READ / DATA

"Read" dient dazu, die relevanten Data-Einträge zu lesen und sie gleichzeitig der angegebenen Variablen zuzuordnen. Wie bei "Restore" beschrieben, kann ein Zeiger (engl.:Pointer) auf eine bestimmte Marke gerichtet werden. Read liest dann die Datas, die auf die angegebene Marke folgen. Wird kein Restore verwendet, werden der Reihe nach vom Programmanfang aus so viele Datas eingelesen, wie Read-Anweisungen vorhanden sind. Sind weniger Datas vorhanden als Read-Anweisungen, wird eine entsprechende Fehlermeldung ausgegeben. Eine weitere Besonderheit des GfA_BASIC ist, daß Text-Datas nicht zwischen An- bzw. Ausführungsstriche gesetzt werden müssen. So ist es dem Interpreter egal, wenn er bei einer alphanumerischen Read-Anweisung numerische Angaben findet. Außerdem werden Leerzeichen, die bei Text-Datas vor oder nach dem zu lesenden Text eingefügt werden, mitberücksichtigt.

z.B.:

```
Restore D_atas
For I%=1 To 9
  Read T_extdata$
  Print T_extdata$;" * ";
Next I%
D_atas:
Data 1,EE,I  0,  Okay ,Zwei ,2, 2,','',  ,
U=Inp(2)
Edit
```

Anders verhält es sich bei numerischen Read-Anweisungen. Diese sind darauf angewiesen, auch numerische Datas vorzufinden. Die Datas können dann allerdings auch in der binären, hexadezimalen oder oktalen Schreibweise angegeben sein. Variablen können in den Datas nicht übergeben werden. (s. auch unter "RESTORE") Kommen im Programm keine Read-Anweisungen vor, kann in Data-Zeilen beliebiger Text stehen. Dieser Text bleibt dann, wie bei "Rem" vom Programm unberücksichtigt.

REM

Möchten Sie Ihr Programm zur besseren Verständlichkeit mit Kommentar versehen, können Sie mit der Anweisung "Rem" an beliebiger Stelle im Programm einen beliebigen Hinweistext anbringen. Diese Anweisung und der nachgestellte Text wird vom Interpreter vernachlässigt und hat damit nicht die geringste Auswirkung auf das Programm. Es ist empfehlenswert, diese Möglichkeit ausgiebig zu nutzen. Bei größeren Programmen erleichtert es Ihnen das Auffinden von bestimmten Programmteilen. Außerdem machen kleine Bemerkungen die Logik Ihrer Programmführung und der Zusammenhänge auch für andere überschaubarer.

```
Bload Iconload$+".ICN",Varptr(M.icon$(I%))
```

'Lade die Dateien (Icon-Daten) an die vorbereitete Stelle.

'Hier ist zu sehen, daß auch einzelne oder zusammengesetzte

'Variablen zur Angabe des Dateinamens verwendet werden können.

VARPTR

Mit " Varptr(Var) / Varptr(Var\$) / Varptr(Feld(Indize)) " kann jeweils die Anfangsadresse der angegebenen Variablen ermittelt werden.

z.B.:

```
A$="BASIC"  
Garbage%=Fre(0)  
A_dresse%=Varptr(A$)  
Print Chr$(Peek(A_dresse%));Chr$(Peek(A_dresse%+1))
```

Ausgabe: BA

FRE

Bei den Funktionen Varptr und Arrptr (s.folgend) ist es ratsam, vor deren Einsatz eine sogenannte "Garbage-Collection" durchführen zu lassen. Der Interpreter sorgt sich nämlich ständig um den verbleibenden Speicherplatz. Dazu wird intern eine Funktion aufgerufen, die dann die Aufgabe hat, nicht mehr benötigte Speicherbereiche zu ermitteln, sie "rauszuwerfen" und die benachbarten Speicherbereiche zusammenzuschließen. Fürwahr eine sehr sinnvolle Einrichtung!

Nur kann vom BASIC-Programmierer nicht erwartet werden, ständig darüber im Bilde zu sein, wann zuletzt eine solche "Müllabfuhr" stattgefunden hat und wann die nächste stattfinden wird. Angenommen, sie taucht gerade zwischen der Ermittlung der Anfangsadresse einer Variablen durch "Varptr" und der Verwendung der für die Adresse vorgesehenen Variablen (im Beispiel zwischen "A_dresse%=Varptr(A\$)" und "Print") auf. Dann stimmt der Inhalt von "A_dresse%" nicht mehr mit der neuen Anfangsadresse der Variablen "A\$" überein. Dieser Umstand kann evtl. fatale Folgen nach sich ziehen. Um nun diese "Garbage-Collection" in solchen Situationen unter Kontrolle zu bekommen, wird einfach der Befehl "Fre(X)" aufgerufen. Der Wert "X" ist dabei ein beliebiger numerischer Ausdruck. "Fre(X)" hat eigentlich die Aufgabe, die Größe des zur Zeit verbleibenden freien Speicherplatzes in Byte zu ermitteln. Dazu wird intern generell ein solche Müllabfuhr durchgeführt. Somit ist sichergestellt, daß im obigen Beispiel der Inhalt der Variablen für die Anfangsadresse auch mit der aktuellen Anfangsadresse der gewünschten Variablen übereinstimmt. Um nun ganz sicher zu gehen, kann man natürlich auch den Varptr-Befehl direkt in die Ausgabe mit einbeziehen.

z.B.:

```
A$="BASIC"
```

```
Print Chr$(Peek(Varptr(A$)));Chr$(Peek(Varptr(A$)+1))
```

Ausgabe: BA

ARRPTR

Wieder, und nicht zum letzten Mal, eine GfA-Besonderheit. Es läßt sich über den Befehl "Arrptr" die Anfangsadresse eines sogenannten "Descriptors" ermitteln. Dieser Descriptor (describe;engl.:beschreiben) enthält verschiedene Angaben über Feld- und Stringvariablen. Die Möglichkeit, eine "Garbage-Collection" durchzuführen bzw. zu veranlassen, verlangt bei Variablen dieser Art (deren Länge ja variabel ist) eine Information darüber, wo die entsprechenden Einträge zu finden sind. Deshalb ist es nötig, eine solche Informationseinheit irgendwo im Speicher zu installieren. Dieses ist der Descriptor. Er gibt jederzeit Auskunft darüber, wo der Inhalt einer solchen Variablen zu finden ist, bzw. welchen Speicherplatz sie einnimmt. Strings und Felder werden hierbei noch unterschiedlich behandelt.

Der Arraypointer (Arrptr) eines Strings beinhaltet in seinen ersten vier Byte die Anfangsadresse des Strings (d.h., die Adresse seines ersten Textzeichens) und in den zwei darauffolgenden Byte seine Länge. Der String wiederum enthält im Anschluß an die durch ihn gespeicherten Textzeichen einen sogenannten "Backtrailer" (trailer;engl.: Schwanz/Anhänger), in welchem die gleiche Adresse steht, die über Arrptr erfahrbar ist. Der Descriptor und der Backtrailer weisen quasi durch ihren Inhalt aufeinander. Dies gewährleistet, daß der ständige Aufräumungsprozess der Garbage-Collection nicht mal "ausnahmsweise" ein Zeichen "verliert". Reale Müllmänner (-frauen?) würden vielleicht sagen: "Das kann ja mal passieren. Ist ja nicht so schlimm!". Hier darf es nicht passieren. Jedes Byte, daß verloren ginge, hätte unter Umständen katastrophale Folgen. Der Aufbau des Descriptors und des Strings mit seinem Backtrailer sieht schematisch so aus:

Der String soll "BASIC" heißen.

Der Arrptr weist auf das erste Byte des Descriptors.

	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6
Descriptor:						
	<= Stringadresse =>			<= Länge =>		
	= Adresse des			des		
	ersten Zeichens			Strings		
	(hier: B).			(hier: 5 Byte).		

Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8	Byte9	Byte10
B	A	S	I	C					
<= S T R I N G I N H A L T =>					<=?=>		<= Backtrailer =>		
beinhaltet Descriptor-Adresse									

^

Dieses Fragezeichen soll andeuten, daß immer dann, wenn der String eine ungerade Länge hat, ein Füllbyte eingesetzt wird, damit sichergestellt ist, daß der Backtrailer, der ja ein Longword (4Byte) beinhaltet, immer an einer geraden Bytegrenze beginnt. Words und Longwords können generell nur ab einer geraden Adresse gelesen werden.(s. auch Peek,Dpeek,Lpeek)

Die Inhalte werden nun ermittelt, indem mit Lpeek und Dpeek die entsprechenden Words und Longwords gelesen werden.

Da meine Bemühungen, die Richtigkeit dessen, was eben erläutert wurde, durch ein eigenes Listing zu beweisen, mit der mir vorliegenden Version nicht zum gewünschten Ergebnis geführt haben, füge ich hier mit freundlicher Erlaubnis des Herstellers die entsprechende Original-Dokumentation des GfA-BASIC-Handbuchs ein.

Nach Auskunft der Firma "GfA Systemtechnik" ist der Programmierer bereits dabei, die aufgetauchten Fehler zu beheben. Es ist, mit begründetem Glauben an die Fähigkeiten des Herrn Ostrowski, anzunehmen, daß zum Veröffentlichungstermin dieses Buches auch diese Mängel der Vergangenheit angehören werden und die dann erhältliche Version zu ungetrübter Programmiererfreude Anlaß geben wird.

```
A$="GfA"
Print "Adresse des Descriptors:"
Print Arrptr(A$)
Print "Anfangsadresse des Strings:"
Print Lpeek(Arrptr(A$))
Print "Zur Kontrolle:"
Print Varptr(A$)
Print "Stringlänge:"
Print Dpeek(Arrptr(A$)+4)
Print "String:"
Print Chr$(Peek(Varptr(A$)));
Print Chr$(Peek(Varptr(A$)+1));
Print Chr$(Peek(Varptr(A$)+2))
Print "Füllbyte:"
Print Peek(Varptr(A$)+3)
Print "Backtrailer:"
Print Lpeek(Varptr(A$)+4)
Edit
```

Nun zum Arraypointer für echte Arrays.

Sitzen Sie gut? Das, was jetzt folgt, sollte nicht spät abends genossen werden. Der gesunde Schlaf könnte darunter leiden. Spaß beiseite.

Als erstes untersuchen wir den Arrptr für Stringfelder.

Der Feld-Arraypointer gibt in seinen ersten vier Byte Auskunft über die Feldadresse. In den nächsten zwei Byte ist die Anzahl der Dimensionen enthalten.

z.B.:

```
Dim Feld$(2,3,1)
Print "Feldadresse : ";Lpeek(Addrptr(Feld$()))
Print "Anz.d.Dimens.: ";Dpeek(Addrptr(Feld$()))+4)
```

Der erste Print-Befehl liefert eine Adresse, die von Fall zu Fall natürlich variiert. Der zweite Print-Befehl gibt die Zahl 3 aus, weil ja 3 Dimensionen eingerichtet wurden (2,3,1).

Das hört sich noch recht einleuchtend an. Nur, unter Feldadresse ist hier nicht die Adresse des ersten Byte des ersten Elementes der ersten Dimension zu verstehen, sondern ein weiterer Descriptor, der Informationen über die einzelnen Dimensionen liefert. Die Größe dieses Descriptors ist abhängig von den Dimensionen, die mit "Dim" vorgesehen wurden. Am Anfang stehen in jeweils 4 Byte die Dimensionstiefen, und zwar rückwärts (!). Wenn also 3 Dimensionen angegeben wurden, ist der erste Sektor dieses Dim-Descriptors 12 Byte lang. Wurden 4 Dimensionen eingerichtet, ist er 16 Byte lang und wurden z.B. 2 Dimensionen angegeben, ist er nur acht Byte lang usw.. Nun steht nach der Formel "Dim-Descriptor-Adresse + (Anzahl der Dimensionen * 4) - (Dimensionsindex * 4)" die jeweils eingetragene Tiefe der entsprechenden Dimension in diesem Sektor.

Bei obigem Beispiel hieße das:

```
D_imdescript.%=Lpeek(Addrptr(Feld$()))
D_imanzahl%=Dpeek(Addrptr(Feld$()))+4)
For I%=1 To D_imanzahl%
  D_imdeep%=D_imdescript.%(4*D_imanzahl%)-(4*I%)
  Print "Tiefe der ";I%;" Dimension : ";Lpeek(D_imdeep%)
Next I%
```

```
Die Ausgabe: Tiefe der 1. Dimension : 3
              Tiefe der 2. Dimension : 4
              Tiefe der 3. Dimension : 2
```

Wie bereits erwähnt, ist die Anzahl der Dimensionsunterteilungen immer um 1 größer, als mit "Dim" angegeben, da die In-

dizierung immer mit Element 0 der jeweiligen Dimension beginnt. So, mit der obigen Routine dürfte dieses Problem zu bewältigen sein. Ab hier unterscheiden sich aber nochmal die Descriptoren für Stringfelder von denen für numerische Felder.

Für Stringfelder schließt sich nun an diesen ersten Sektor (der ja in seiner Länge von der Anzahl der Dimensionen abhängig ist) eine Liste von Descriptoren an, die mit dem normalen String-Varptr identisch sind. Diesmal glücklicherweise vorwärts. Der String-Varptr besteht bekanntlich aus 6 Byte, von denen die ersten vier die String-Adresse enthalten und die nächsten zwei Byte die Länge des Strings.

Hier gibt es aber nun nicht nur einen Descriptor, sondern so viele, wie Unterteilungen in der ersten Dimension vorhanden sind.

Bei der oben angeführten Dimensionierung wären das also drei. Diese Descriptoren lassen sich nach der Formel "Dim-Descriptor-Adresse + (Anzahl der Dimensionen * 4) + (Elementindize der ersten Dimension * 6)" diese begehrten 6 Byte herauslesen.

```
D_imdescript.%=Lpeek(Arrptr(Feld$()))
D_imanzahl%=Dpeek(Arrptr(Feld$))+4)
D_imdeep.0%=D_imdescript.%(4*D_imanzahl%)-4
S_stringptr%=D_imdescript.%(4*D_imanzahl%)
For I% = 0 to D_imdeep.0%
    Print "String-Adresse ";I%"/0/0 : ";Lpeek(S_stringptr%+(I%*6))
    Print "String-Länge   ";I%"/0/0 : ";Dpeek(S_stringptr%+4+(I%*6))
Next I%
```

Die Ausgabe würde hier, je nach Fall, anders aussehen.

Was aber wieder sehr, sagen wir mal, ungewöhnlich anmutet, ist die Tatsache, daß diese Feldstring-Pointer in ihren ersten vier Byte auf Adressen zeigen, die ca. 50 Kilobyte unterhalb der Bildschirmadresse liegen. Die Arrays sind nämlich (wieder was für Linkshänder) rückwärts angelegt. D.h., daß das erste Byte des jeweiligen Strings oben im Speicher liegt und die anderen sich dann der Reihe nach rückwärts daran anschließen.

Zu dieser Problematik finden Sie anschließend ein kleines Programm, das Ihnen dabei helfen soll, diese Vorgänge etwas leichter zu verstehen. Manchen mag es scheinen, daß bei diesem Programm etwas übertrieben wurde. Dazu gibt es ein Sprichwort:

"Übertreibungen machen deutlich!".

So läßt sich evtl. die Grundstruktur dieses Befehls in seinen Gesetzmäßigkeiten eher verstehen.

```
@Smalltext(1)
Dim A$(4,3,7)
A$(0,0,0)="A"
A$(1,0,0)="BC"
A$(2,0,0)="DEF"
A$(3,0,0)="GHIJ"
A$(4,0,0)="KLMNO"
D_escripadr%=Arrptr(A$())
A_rrayadr%=Lpeek(D_escripadr%)
Print String$(39,"-")
Print " Feld-Descriptor           : ";D_escripadr%
Print String$(39,"-")
Print " Feldadresse             : ";Lpeek(D_escripadr%)
Print String$(39,"-")
Print " Anz. d. Dimensionen      : ";Dpeek(D_escripadr%+4)
Print String$(39,"-")
Print " Anz.d.Elem.in 1.Dimension : ";Lpeek(A_rrayadr%+8)
Print " Anz.d.Elem.in 2.Dimension : ";Lpeek(A_rrayadr%+4)
Print " Anz.d.Elem.in 3.Dimension : ";Lpeek(A_rrayadr%)
Print String$(39,"-")
Print " 0/0/0 - Descriptoradresse  : ";A_rrayadr%+12
Print " 1/0/0 - Descriptoradresse  : ";A_rrayadr%+18
Print " 2/0/0 - Descriptoradresse  : ";A_rrayadr%+24
Print " 3/0/0 - Descriptoradresse  : ";A_rrayadr%+30
Print " 4/0/0 - Descriptoradresse  : ";A_rrayadr%+36
Print String$(39,"-")
Print " 0/0/0 - Adresse mit Varptr : ";Varptr(A$(0,0,0))
Print " 1/0/0 - Adresse mit Varptr : ";Varptr(A$(1,0,0))
Print " 2/0/0 - Adresse mit Varptr : ";Varptr(A$(2,0,0))
```

```

Print " 3/0/0 - Adresse mit Varptr      : ";Varptr(A$(3,0,0))
Print " 4/0/0 - Adresse mit Varptr      : ";Varptr(A$(4,0,0))
Print String$(39,"-")
Print " 0/0/0 Adr.im 0/0/0-Descriptor : ";Lpeek(A_rrayadr%+12)
Print " 1/0/0 Adr.im 1/0/0-Descriptor : ";Lpeek(A_rrayadr%+18)
Print " 2/0/0 Adr.im 2/0/0-Descriptor : ";Lpeek(A_rrayadr%+24)
Print " 3/0/0 Adr.im 3/0/0-Descriptor : ";Lpeek(A_rrayadr%+30)
Print " 4/0/0 Adr.im 4/0/0-Descriptor : ";Lpeek(A_rrayadr%+36)
Print String$(39,"-")
Print " Stringlänge 0/0/0                : ";Dpeek(A_rrayadr%+16)
Print " Stringlänge 1/0/0                : ";Dpeek(A_rrayadr%+22)
Print " Stringlänge 2/0/0                : ";Dpeek(A_rrayadr%+28)
Print " Stringlänge 3/0/0                : ";Dpeek(A_rrayadr%+34)
Print " Stringlänge 4/0/0                : ";Dpeek(A_rrayadr%+40)
Print String$(39,"-")
S_string.0$=Chr$(Peek(Lpeek(A_rrayadr%+12)))
Print " String 0/0/0                    : ";S_string.0$
For I%=0 To 1
    S_string.1$=S_string.1$+Chr$(Peek(Lpeek(A_rrayadr%+18)+I%))
Next I%
Print " String 1/0/0                    : ";S_string.1$
For I%=0 To 2
    S_string.2$=S_string.2$+Chr$(Peek(Lpeek(A_rrayadr%+24)+I%))
Next I%
Print " String 2/0/0                    : ";S_string.2$
For I%=0 To 3
    S_string.3$=S_string.3$+Chr$(Peek(Lpeek(A_rrayadr%+30)+I%))
Next I%
Print " String 3/0/0                    : ";S_string.3$
For I%=0 To 4
    S_string.4$=S_string.4$+Chr$(Peek(Lpeek(A_rrayadr%+36)+I%))
Next I%
Print " String 4/0/0                    : ";S_string.4$
Print String$(39,"-")
Print " 0/0/0 - Backtrailer zeigt auf : ";Lpeek(Lpeek(A_rrayadr%+12)+2)
Print " 1/0/0 - Backtrailer zeigt auf : ";Lpeek(Lpeek(A_rrayadr%+18)+2)
Print " 2/0/0 - Backtrailer zeigt auf : ";Lpeek(Lpeek(A_rrayadr%+24)+4)
Print " 3/0/0 - Backtrailer zeigt auf : ";Lpeek(Lpeek(A_rrayadr%+30)+4)
Print " 4/0/0 - Backtrailer zeigt auf : ";Lpeek(Lpeek(A_rrayadr%+36)+6)
Print String$(39,"-")

```

```
L_aenge.3_0%=Len(A$(3,0,0))
If L_aenge.3_0% Mod 2<>0
  Inc L_aenge.3_0%
Endif
Bt.3_0%=Varptr(A$(3,0,0))+L_aenge.3_0%
Print " 3/0/0- Backtrailer zeigt auf  : ";Lpeek(Bt.3_0%)
Print " (mit Len und Varptr ermittelt)"
Print String$(39,"-");Chr$(27);"H"
U=Inp(2)
@Smalltext(2)
Edit
```

Dieses ist kein Abdruck aus dem GfA-BASIC-Handbuch, da beim Testen dieses Arraypointers keinerlei Fehler aufgetreten sind.

Sie finden in diesem kleinen Listing zwei Prozedur-Aufrufe. Sie bewirken, daß der Systemfont zuerst auf die Color-Textgröße 8*8 und zum Ende wieder auf die normale Hires-Größe 8*16 umgestellt wird. Dieses hat, wie sie beim Testen dieses Programms sehen werden, den Vorteil, daß die gesamte Bildschirmausgabe auf einen Screen paßt. Sollten Sie dieses Listing isoliert verwenden, vergessen Sie bitte nicht, diese "Procedure Smalltext" mit einzubinden. Es ist übrigens dieselbe Umschaltfunktion, die im Editormenü bei "Text 16/Text 8" verwendet wird. Zum zweiten wird in der viertletzten Zeile ein TOS-Befehl verwendet (Chr\$(27);"H"), der den Textcursor nach Abschluß der letzten Print-Zeile nach "Home" (linke obere Bildschirmecke) bringt, da sonst durch das Zeilenscrolling die oberste Textzeile aus dem Bildschirm gedrängt würde. Näheres zu diesem Thema finden Sie unter "TOS - wie geht das?" Dieses Unterdrücken des Scrollings ist auch mit einem an den Print-Befehl angehängten Semikolon möglich (dazu mehr unter PRINT).

Die entsprechenden Informationen über numerische Variablenfelder sind, einschließlich der Dimensionstiefe, nach dem gleichen Muster zu ermitteln, wie bei Stringfeldern. Nach dem Sektor, der Auskunft über die Tiefen gibt, folgen jedoch hier die entsprechenden Elemente, bzw. deren Inhalte.

Um nun zu wissen, in welchen Byte-Abständen diese Werte zu finden sind, sei hier noch die Variablenorganisation des GfA-BASIC's erwähnt.

Es gibt vier Variablentypen:

Realzahl-Variablen	: keine Kennung	=> 6 Byte Speicherplatz
Integer-Variablen	: Kennung %	=> 4 Byte Speicherplatz
Bool'sche Variablen	: Kennung !	=> 2 Byte Speicherplatz
String-Variablen	: Kennung \$	=> 1 Byte Speicherplatz je Zeichen

Realzahlen (Fließkommazahlen) können hier mit einer Genauigkeit von bis zu 11 Nachkommastellen dargestellt werden. In der Exponential-Darstellung ist eine maximale Exponentengröße von 154 möglich (z.B. 2.345E+154).

Integerwerte (Nachkommastellen werden unterdrückt) können im Bereich von -2^{31} (-2147483648) bis $2^{31}-1$ (+2147483647) liegen.

Bool'sche Variablen können nur die beiden Werte 0 (falsch) und -1 (=> ungleich 0 = richtig) annehmen.

Für Stringvariablen ist eine maximale Länge von 32767 Zeichen vorgesehen, da der Teil des Descriptors, der die Länge beinhaltet zwei Byte (5 und 6) groß ist.

POKE / DPOKE / LPOKE

Syntax: Poke (bzw. Dpoke/Lpoke) Speicheradresse, Wert. Es gibt, wie wohl den meisten bekannt sein wird, in BASIC Möglichkeiten, direkt auf die Speicherstellen zuzugreifen und deren Inhalt entweder zu lesen (Peek,Dpeek,Lpeek) oder vom Programmierer gewählte Werte in diese Stellen hineinzuschreiben. Der "Poke"-Befehl, wie er aus anderen BASIC-Dialekten bekannt ist, hat im GfA-BASIC eine Erweiterung erfahren. So können nicht nur einzelne Byte-Werte in den Speicher geschrieben werden (was bei größeren Werten immer eine aufwendige Umfor-

mungsprozedur nach sich zog), sondern es können auch ganzzahlige Words und Longwords mit einem Befehl transportiert werden.

Byteformat = Poke => verändert den Wert eines einzelnen Bytes mit Werten im Bereich von -256 bis 255

Wordformat = Dpoke => verändert den Wert zweier hintereinanderliegender Bytes mit Werten im Bereich von -65536 bis 65535

Longformat = Lpoke => verändert den Wert von vier hintereinanderliegenden Bytes mit Werten im Bereich von -2147483648 bis 2147483647

Anders als bei früheren 8-Bitern verfügt der ST über die Fähigkeit, ohne Umformung Werte bis zu $2^{31}-1$ (32-Bit-Breite) verarbeiten zu können. Grundsätzlich arbeitet der ST mit Words, weshalb eine Ein-Byte-Manipulation durch Poke für ihn eigentlich mehr Arbeit bedeutet, als ein Word-Manipulation. Um nun systemintern sicher zu gehen, daß diese Zugriffsformate nicht durcheinanderkommen, können Words und Longs (Longwords) jeweils nur ab einer geraden Bytezahl geschrieben oder gelesen werden. Die Manipulation einzelner Bytewerte dagegen ist an jeder beliebigen Stelle möglich. Wobei allerdings zu beachten ist, daß numerische Variablen immer entweder in einem 4 Byte großen (Integervariablen) oder 6 Byte großen Speicher (Fließkommavariablen) abgelegt werden. Das folgende Beispiel liefert Ihnen nicht das vielleicht erwartete Ergebnis 22.

```
A%=111
Poke Varptr(A%),22
Print A%
```

sondern einen Wert von $22 \cdot 256 \cdot 256 \cdot 256 + 111 = 369098863$. Dieses hat seinen Grund eben darin, daß A% eine 4 Byte große Integerzahl ist. Mit "Poke Varptr(A%),22" verändern Sie also nicht die Zahl an sich, sondern nur das erste Byte davon. Die, der Variablen A% anfangs zugewiesene Zahl 111 benötigt nur den Platz eines Bytes, nämlich des letzten der vier vorhandenen.

Die davorliegenden 3 Byte dieses Longwords sind mit Nullen belegt. Wollen Sie also nur die Ein-Byte-Zahl verändern, müssen Sie den neuen Wert, solange er kleiner als 256 ist, in `Varptr(A%)+3` poken. Wenn der neue Wert größer als 255 und kleiner als 65536 ist, in `Varptr(A%)+2`. Dann allerdings mit `Dpoke`. Und nur wenn der neue Wert tatsächlich Long-Format hätte, also größer als 65535 ist, wäre dafür `Varptr(A%)` zuständig. Diese Zahl müßten Sie dann allerdings mit einem `Lpoke` übergeben. Mit dem Platz innerhalb definierter Stringvariablen lassen sich hiermit ebenfalls interessante Spielchen treiben. Hier können Sie dann nämlich sehr speicherplatzsparend Byte- oder Word-Werte ablegen, ohne jedesmal eine 4 Byte "verschwendende" numerische Variable einrichten zu müssen. Bei umfangreichen Feldoperationen kann dieser Effekt unter Umständen sehr wichtig werden. Mit Fließkommawerten ist dies jedoch nicht möglich, da diese besonders behandelt werden müssen. Diese Variablentypen benötigen deshalb auch einen 6 Byte großen Speicherplatz. Es ist sehr lohnenswert, sich mit dieser Materie genauer auseinanderzusetzen. Experimentieren Sie damit ruhig ein wenig herum. Stellen Sie aber vorher durch die Einrichtung von bestimmten Variablen (s.o.) sicher, daß Sie nicht in wichtige Bereiche des Systems oder des Interpreters hineinpoken. Lassen Sie bitte grundsätzlich bei allen Speichermanipulationen äußerste Vorsicht walten!

SPOKE / SDPOKE / SLPOKE

Diese Befehle dienen auch der Speichermanipulation. Man kann im GfA-BASIC diese Manipulationen in zwei Bereiche gliedern: die allgemein zugänglichen Speicherbereiche und die Speicherbereiche, die dem Supervisor zugeordnet werden. Der Supervisor-Modus hat die Aufgabe, bestimmte Zonen im Speicher vor unbefugtem Zugriff zu schützen, um wichtige Systemeinstellungen zu erhalten. Verändert man mit `Spoke`, `Sdpoke`, `Slpoke` (`S=Super`) diese Einstellungen und behält dabei nicht absolut den Überblick über die damit initialisierten Funktionen, wird man im Endeffekt wohl nur noch ein System-Neustart aus seiner mißlichen Lage befreit. Die Superpokes sind tatsächlich nur

dann anzuwenden, wenn man sich systemintern sehr genau auskennt und bereit ist, sich mit den Konsequenzen aus seinem Poken abzufinden.

PEEK / DPEEK / LPEEK

Syntax: Var%=Peek (bzw. Dpeek/Lpeek) (Speicheradresse)
oder: Print Peek (bzw. Dpeek/Lpeek) (Speicheradresse)

Die Umkehrfunktionen zu den Poke-Befehlen sind die "Peek"-Befehle. Wie auch bei den Pokes unterscheidet man hier zwischen Byte-Peeks, Word-Peeks und Long-Peeks. Dieser Befehl greift auf die in Klammern nachgestellte Speichersdresse zu und gibt ihren Inhalt zurück. Ebenfalls können hier Dpeek und Lpeek nur bei geraden Adressen verwendet werden.

```
Beispiel:      A$=" "  
               Poke Varptr(A$),34  
               Print Peek(Varptr(A$))  
               Poke Varptr(A$),Peek(Varptr(A$))+1  
               Print Peek(Varptr(A$))  
               Edit
```

Dieses Miniprogramm richtet eine Ein-Byte-Stringvariable ein, übergibt an diese Speicherstelle mit Poke den Wert 34 und liest anschließend gleichzeitig den Inhalt der Speicherstelle, erhöht den Wert um 1 und schreibt den neuen Wert wieder in die Speicherstelle hinein. Als Ausgabe erscheint zum Schluß der Wert 35

Übrigens arbeitet der Peek-Befehl ausschließlich im Supervisor-modus. Sie können dadurch also jederzeit Informationen über jede einzelne Speicherstelle Ihres Computers erhalten.

Next I%

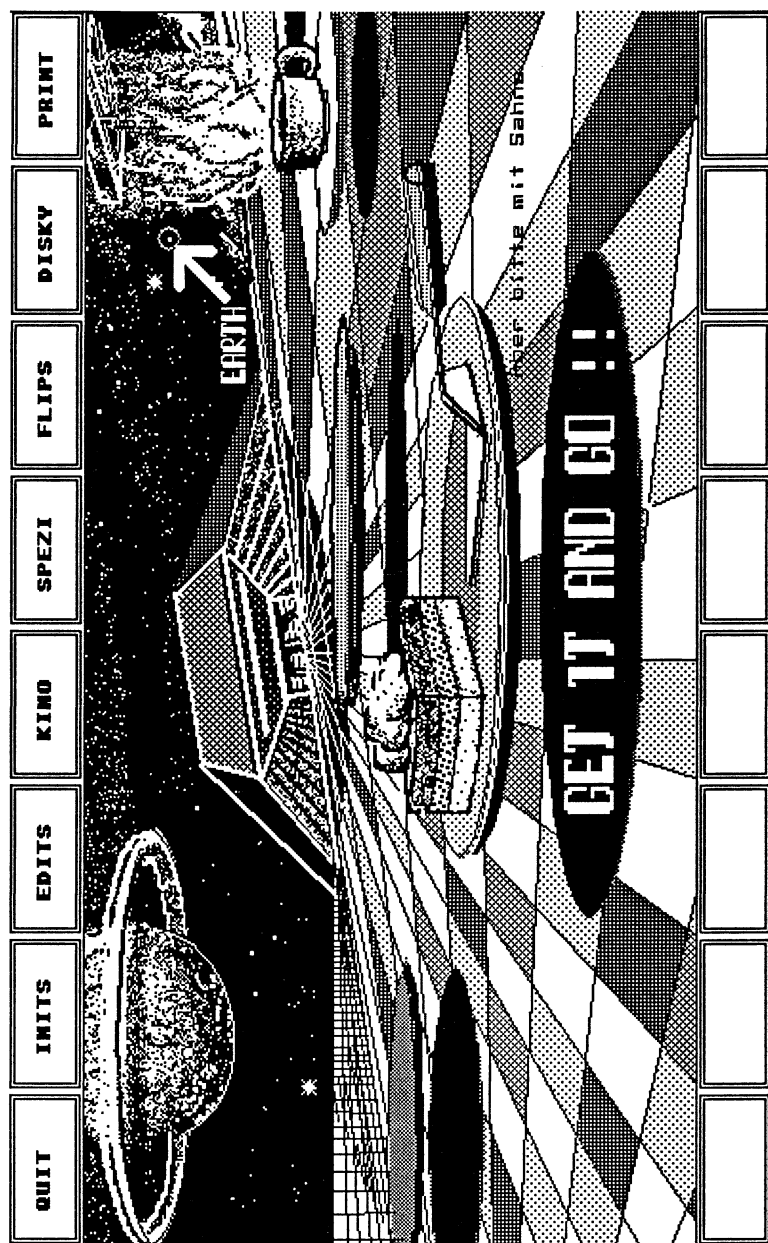
'Gehe solange zurück zur entsprechenden "For"-Anweisung, bis der dortige Endwert erreicht ist.

Defmouse 0

Enter.2:

Return

'Ende der Ladeprozedur. Gehe zurück zur aufrufenden Programmzeile
'und fahre mit der darauffolgenden Programmzeile fort.



Procedure Grafmenue

'Wenn Sie das Programm laufen lassen und dabei die "Esc"-Taste
'drücken, erscheint das Hauptmenü. Das Programmteil dazu beginnt hier.

aG.etflip

'Als erstes wird festgestellt, welche der beiden Grafik-Screens sich
'momentan auf dem Bildschirm befindet und diese in der aktuellen Form
'festgehalten (s.Procedure G.etflip).

Graphmode 1

'Da zwischenzeitlich die verschiedenen Einstellungen verändert wurden,
'müssen sie hier wieder auf den normalen Modus gesetzt werden.
'Als erstes wird hier der Grafikmodus "Replace" eingestellt.

GRAPHMODE

Für alle Grafikausgaben, außer der Raster-Operation "Put"
gelten vier verschiedene Grafik-Modi:

Graphmode 1 = Replace

Das verwendete Grafik-Element (Pbox, Line etc.) wird
vollflächig dargestellt. Alles, was sich darunter befindet,
wird davon überdeckt und ersetzt (replace).

==> neuer Punkt = farbmaske d. neuen Punktes AND
neuer Punkt

Graphmode 2 = Transparent

Es werden nur dort Bildpunkte gesetzt, wo dem neuen
Grafik-Element Farben zugeordnet wurden (hier: schwarz).
Wo keine Farben darzustellen sind (hier: weiß), bleibt der
alte Hintergrund erhalten. Die neue Fläche erscheint also
durchsichtig (transparent). Bei einer weißen Pbox mit
schwarzem Rand wäre in diesem Fall nur der Rand zu
sehen.

==> neuer Punkt = (farbmaske d. neuen Punktes AND
neuer Punkt)

OR (Farbe d. alten Punktes AND NOT neuer Punkt)

Graphmode 3 = Xor (==> e(X)klusives Oder)

Dieser Modus läßt sich nicht gerade einfach erklären. Man könnte ihn fast schon als intelligent bezeichnen. Er "merkt" sich nämlich die Punkte, die in der Grafik vorhanden waren, bevor sie mit einem neuen Grafik-Element in diesem Modus überdeckt werden und löscht sie anschließend. Sonst werden nur dort Punkte gesetzt, wo vorher noch keine waren. Setzt man zwei Grafik-Elemente der gleichen Art in diesem Modus auf exakt dieselbe Stelle, wird das erste vom zweiten wieder gelöscht, ohne den vorherigen Hintergrund zu zerstören. Damit lassen sich sehr wirkungsvolle Effekte erzielen.

==> neuer Punkt = neuer Punkt XOR Farbe d. alten Punktes

Graphmode 4 = Revers transparent

Dieser Modus ist identisch mit dem Modus 2, nur das die Farben revers (umgekehrt) dargestellt werden. Alles was im neuen Grafik-Element z.B. als weiß definiert wurde, wird schwarz dargestellt und alles was als schwarz definiert wurde, weiß. Wenn also zwei dieselben Grafik-Elemente einmal im Modus 2 und einmal im Modus 4 dargestellt würden, so wäre das zweite das exakte Negativ des ersten.

==> neuer Punkt = (Farbe d. alten Punktes AND neuer Punkt)

OR (Farbmaske d. neuen Punktes AND NOT neuer Punkt)

Anhand eines kleinen Beispielprogramms können Sie die Wirkungen der verschiedenen Modi testen. Drücken Sie während dieses Programms abwechselnd die linke, rechte oder beide Maustasten. Drücken Sie keine Maustaste, wird der Replace-Modus eingesetzt. Das Programm wird abgebrochen, indem Sie auf der Tastatur eine beliebige Taste drücken.

```
Deffill 1,2,1
Repeat
Mouse X,Y,K
Graphmode 1
If K=1
    Graphmode 2
Endif
If K=2
    Graphmode 3
Endif
If K=3
    Graphmode 4
Endif
Pbox X,Y,X+50,Y+50
Pbox X,Y,X+50,Y+50
Until Len(Inkey$)
Edit

Color 1

'setze Linienfarbe auf schwarz
```

COLOR

Mit "Color" wird das Farbregerister bestimmt, aus dem die Objekte Line, Plot, Draw, Circle und Box ihre Farben beziehen. Im Monochrom-Modus sind dies nur die Register 0 (weiß) und 1 (schwarz). In mittlerer Auflösung können 4 Farbregerister und in niedriger Auflösung 16 Farbregerister angesprochen werden, deren Inhalt mit dem Befehl für Farbmischung der Register "SetColor Reg.,Rotanteil,Grünanteil,Blauanteil" (= RGB-Mischung) bestimmt werden kann.

SETCOLOR

Wie eben erwähnt, kann mit diesem Befehl der Farbton (RGB-Mischung) der einzelnen Farbregerister (HIRES = 0-1; MIDRES = 0-3; LOWRES = 0-15) bestimmt werden. Der Versuch, im

HIRES-Modus die Register zu verändern, wird nicht mit verschiedenen Grautönen belohnt. Schwarz (Reg.1) bleibt schwarz und weiß (Reg.0) bleibt weiß. Es ergibt z.B. "Setcolor 1,7,7,0" einen Farbanteil von 7 für die Farbe Rot, ebenfalls 7 für die Farbe Grün und 0 für die Farbe Blau (=Gelb). So lassen sich 512 verschiedene Farben (3 Farben, je 8 Stufen = $8*8*8 = 512$ Farben) erzeugen. Bei verschiedenen Gelegenheiten ist es hier von Nutzen, die Registereinträge nicht mit drei unabhängigen Parametern vorzunehmen, sondern einen Gesamt-Mischungswert zu übergeben. Dies können Sie bei Setcolor auf zwei Arten erreichen. Der Interpreter unterscheidet bei Setcolor, ob ein oder mehrere Farbparameter übergeben wurde. Wird nur ein Parameter übergeben, setzt sich die Farbmischung nach folgendem Muster zusammen:

Soll z.B. ein Blauanteil von 3, ein Rotanteil von 2 und ein Grünanteil von 5 übergeben werden, errechnet sich der Wert, indem man den Rotanteil mit 256 multipliziert, den Grünanteil mit 16 und den Blauanteil mit 1. In diesem Fall müßte der Wert 595 übergeben werden, um die gewünschte Farbmischung zu erhalten.

Durch die Möglichkeit, mit Hexadezimalzahlen zu arbeiten, kann man diese Berechnung folgendermaßen vereinfachen:

```

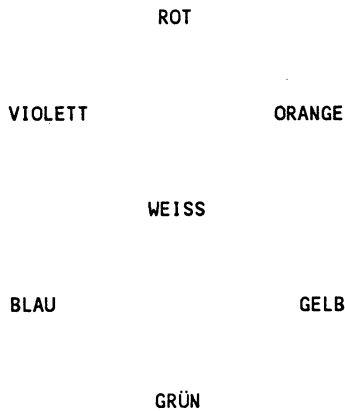
Setcolor 1,&H253
oder
A=&H253
Setcolor 1,A
oder
A$="&H"+"2"+"5"+"3"
A=Val(A$)
Setcolor 1,A
```

Das letzte Beispiel ist eine Phantasievariante, die jedoch evtl. notwendig werden könnte und zum anderen verdeutlicht, wie sich durch die Funktion "Val" mehrere Textausdrücke zu einer Zahl zusammenfassen lassen.

Drei Register haben eine besondere Bedeutung:

- Register 0 = Hintergrundfarbe
- Register 1 = Print-Ausgabefarbe
- Register 3 = Textfarbe des BASIC-Editors

Leider ist der Goethe'sche Farbkreis hier nicht anwendbar, da hier nicht die drei Elementarfarben Rot,Blau,Gelb gemischt werden. Um sich bei der Farbwahl zu orientieren, müßte aus dem Grünfilter die Farbe Blau subtrahiert werden. Trotzdem sei der Goethe'sche Farbkreis hier skizziert:



Es existieren drei Elementarfarben : ROT, BLAU, GELB Die 1:1-Mischung zweier dieser Farben ergibt die Komplementärfarbe (Gegenfarbe) der nicht beteiligten Farbe.

- 1:1-Mischung von ROT und BLAU ergibt VIOLETT (=Komplement zu GELB)
- 1:1-Mischung von ROT und GELB ergibt ORANGE (=Komplement zu BLAU)
- 1:1-Mischung von BLAU und GELB ergibt GRÜN (=Komplement zu ROT)
- 1:1:1-Mischung von ROT, BLAU und GELB ergibt WEISS
- Keine der Elementarfarben beteiligt, ergibt "Nichtfarbe" SCHWARZ

Werden die Anteilsgrößen verändert, entstehen weitere Farbabstufungen.

z.B. 2:1-Mischung von BLAU und GELB ergibt TÜRKIS
 2:1-Mischung von BLAU und ROT ergibt LILA
 2:1-Mischung von ROT und GELB ergibt ZINNOBER
 2:1-Mischung von ROT und BLAU ergibt PINK
 2:1-Mischung von GELB und BLAU ergibt LINDGRÜN
 2:1-Mischung von GELB und ROT ergibt DUNKELGELB
 u.s.w.

Die Farbpsychologie lehrt, daß Farben den seelischen Grundzustand des Menschen beeinflussen, bzw. daß Menschen mit ihren verschiedenen seelischen Stimmungen auch die Vorlieben und Abneigungen für bestimmte Farben wechseln.

ROT steht hier für Aktivität, Aggressivität, Liebe
 BLAU steht für Ruhe, Ausgeglichenheit, Ernsthaftigkeit
 GELB steht für Lebensfreude, Genuß, Optimismus, Phantasie
 VIOLETT für Individualismus, Kritik, Glaube, Sinnlichkeit
 GRÜN steht für Naturbezogenheit, Bodenständigkeit, Treue
 ORANGE steht für Sicherheit, Ausdauer, Ehrgeiz

Es ist sehr lehrreich, sich mit dem Thema Farbpsychologie zu beschäftigen. Es gibt einem oft unerwarteten Aufschluß über die momentane persönliche Situation.

Defline 1,1,0,0

'setze Linieneinstellungen auf "normal".

DEFLINE

"Defline" bestimmt in vier Parametern die Darstellungsart der Linien, die mit Box,Circle,Line,Rbox,Ellipse und Draw sichtbar gemacht werden können. Der erste Parameter ist dabei für den Linienstil, der zweite für die Liniendicke, der dritte für die Anfangsform und der vierte für die Endform der Linie zuständig.

Linienstil:

- 1 = durchgezogene Linie
- 2 = gestrichelte Linie mit kleinen Abständen
- 3 = gepunktete Linie
- 4 = Punkt-Strich-Linie (z.B. für Achsendarstellung)
- 5 = gestrichelte Linie mit großen Abständen
- 6 = Strich-Punkt-Punkt-Linie
- 1 bis -65535 = selbstdefinierter Linienstil

Der selbstdefinierte Linienstil setzt sich aus einem Word (16 Bit) zusammen, wobei jedes gesetzte Bit von rechts aus gezählt einen Punkt in der Linie darstellt und ein nicht gesetztes Bit eine freie Stelle.

z.B. die Bitkombination 1110110111101101 ergibt hier den Wert 60909.

Von rechts aus gesehen heißt das:

Bit 0	gesetzt	=	2^0	=	1
Bit 1	nicht gesetzt	=			0
Bit 2	gesetzt	=	2^2	=	4
Bit 3	gesetzt	=	2^3	=	8
Bit 4	nicht gesetzt	=			0
Bit 5	gesetzt	=	2^5	=	32
Bit 6	gesetzt	=	2^6	=	64
Bit 7	gesetzt	=	2^7	=	128
Bit 8	gesetzt	=	2^8	=	256
Bit 9	nicht gesetzt	=			0
Bit 10	gesetzt	=	2^{10}	=	1024
Bit 11	gesetzt	=	2^{11}	=	2048
Bit 12	nicht gesetzt	=			0
Bit 13	gesetzt	=	2^{13}	=	8192
Bit 14	gesetzt	=	2^{14}	=	16384
Bit 15	gesetzt	=	2^{15}	=	32768

Die Summe aller gesetzten Bits = 60909

Diese Zahl muß als Minuswert im ersten Deflineparameter eingesetzt werden, um einen Liniestil mit dem angegebenen Bitmuster zu erzeugen. Die Linie setzt sich dann aus dem Vielfachen dieser 16 Bits zusammen.

Der zweite Parameter legt die Liniendicke fest. Es ist eine maximale Liniendicke von 40 Pixels möglich. Dabei ist zu beachten, daß sich die Dicke immer in Zweireschritten erhöht. Der Wert 2 erzeugt also die gleiche Dicke wie Wert 1 und Wert 16 die gleiche wie Wert 15 usw..

Der dritte und vierte Parameter legen die Formen des Linienanfangs und des Linienendes fest.

Es stehen drei Formen zur Verfügung:

```
Wert 1 = eckig  
Wert 2 = pfeilförmig  
Wert 3 = rund
```

@Clipprec(2)

'Falls zwischenzeitlich ein "clipping Retangle", also ein bestimmter
'Bildausschnitt definiert wurde, an dessen Rand die Grafikausgaben
'(außer der Rasterbefehl "Put") abgeschnitten werden sollen, so muß
'dieser Ausschnitt wieder auf die volle Bildschirmgröße erweitert
'werden, da sonst das Menü nur zum Teil erscheint. Wie sich ein
'clipping Rectangle auswirkt, können Sie am besten erkennen, wenn Sie
'im Hauptmenü den Menüpunkt "SPEZIS" und anschließend die Untermenü-
'punkte "Set Clipping" und "Clipping An" klicken. Sie werden dann auf-
'gefordert, einen Bildausschnitt festzulegen. Nachdem Sie das getan
'haben, wählen Sie im Einstellmenü ("Tab"-Taste drücken) einen be-
'liebigen Grafikelement (außer "Self") und zeichnen damit. Sie werden
'sehen, daß Sie außerhalb des gewählten Ausschnittes nicht zeichnen
'können. Dieser Effekt wird mit einer GEM-Funktion erzeugt, die später
'noch erläutert werden soll.

,

If Menstartflag%=1

'Hier wird erfragt, ob das Hauptmenü schon einmal aufgerufen wurde.

For I%=0 To 39 Step 4

'Wenn ja, werden die in Up- und Dwnmenue\$ festgehalten Bilder
'davon mit dieser Schleife aus dem oberen bzw. unteren Bildrand
'heraus-"geschoben".

Put 0,-37+I%,Upmenue\$,3

Put 0,396-I%,Dwnmenue\$,3

'Dazu werden die beiden Teilmenü-Ausschnitte, die beim ersten
'Hauptmenü-Aufruf gespeichert wurden, in Vierer-Steps vom oberen
'Bildrand bzw. vom unteren Bildrand aus auf den Bildschirm ge-
'bracht. Dabei wird beim ersten Step nur 1/10 der Menüleistenhöhe
'angezeigt, beim zweiten Step 2/10, beim dritten Step 3/10 u.s.w.,
'bis nach 10 Steps endlich das ganze Bild der jeweiligen Menü-
'leiste erscheint.

'Bei jedem Step wird das vorherige Bild vom nächsten überlagert.
'Da sich die Put-Koordinaten immer auf die linke, obere Ecke des
'gespeicherten Bildes beziehen, muß bei der oberen Leiste also im
'Minusbereich angefangen werden.

GET / PUT

Das interne Speichern der Bilder wird mit einem Befehl namens "Get" bewirkt. Der Befehl "Put" ist die entsprechende Umkehrung dazu. Der Get-Befehl (nicht zu verwechseln mit dem Befehl "Ge:#", der zum Lesen von Disketten-Datensätzen verwendet wird) bietet die Möglichkeit, beliebige Bildschirmausschnitte zu speichern, um Sie hinterher mit dem Put-Befehl (ebenfalls nicht zu verwechseln mit dem Diskettenbefehl "Put#", mit dem Datensätze auf Diskette geschrieben werden) weiterzuverwenden. Dabei wird das Bitmuster des gewählten Ausschnittes in einem Text-String abgelegt, dessen Inhalt allerdings nicht für Textoperationen verwendet werden kann.

Die Syntax der beiden Befehle:

"XLO" bedeutet hier : X-Koordinate links oben

"YLO" : Y-Koordinate links oben

"XRU" : X-Koordinate rechts unten

"YRU" : Y-Koordinate rechts unten

Die Koordinaten-Angaben sind immer relativ zum linken oberen Eckpunkt des Bildschirms.

Get XLO,YLO,XRU,YRU,Speicher\$

Put XLO,YLO,Speicher\$,Modus (Modus ist optional)

Die Eck-Koordinaten des Put-Befehls. können dabei beliebig gewählt werden. Die Koordinaten des rechten unteren Eckpunktes ergeben aus dem mit Get gespeicherten Ausschnittes. Dabei ist es möglich, die Put-Koordinaten aus dem Bildschirmbereich hinaus zu verlagern, sodaß entweder nur Teile des Ausschnittes oder gar nichts davon zu sehen ist. Diesem Effekt werden Sie bei Anwendung des Grafikprogrammes häufiger begegnen. Beim Get-Befehl müssen die angegebenen Eck-Koordinaten innerhalb des Bildschirmbereichs liegen, da sonst der Interpreter den Befehl nicht ausführt. Würde er das tun, könnte das Auswirkungen auf die benachbarten Speicherbereiche unter oder über dem Bildschirmspeicher haben, die das System zum Absturz bringen.

Zum Modus beim Put-Befehl ist zu sagen, daß diese den Darstellungs-Modi beim Graphmode-Befehl ähnlich sind, nur ist die Auswahl bei Rasteroperationen diesbezüglich wesentlich größer.

- Modus 0 : alten Hintergrund löschen
- Modus 1 : neues Bild AND Hintergrund
- Modus 2 : neues Bild AND NOT Hintergrund
- Modus 3 : neues Bild (ersetzt den Hintergrund)
- Modus 4 : (NOT neues Bild) AND Hintergrund
- Modus 5 : Hintergrund (es geschieht nichts)
- Modus 6 : neues Bild XOR Hintergrund
- Modus 7 : neues Bild OR Hintergrund
- Modus 8 : NOT (neues Bild OR Hintergrund)

Modus 9 :	NOT (neues Bild XOR Hintergrund)
Modus 10:	NOT Hintergrund (Hintergrund revers)
Modus 11:	neues Bild OR (NOT Hintergrund)
Modus 12:	NOT neues Bild (revers transparent)
Modus 13:	(NOT neues Bild) OR Hintergrund
Modus 14:	NOT (neues Bild AND Hintergrund)
Modus 15:	1 (=schwarz)

Der Rastermodus 3	entspricht dem Graphmode 1
Rastermodus 7	dem Graphmode 2
Rastermodus 6	dem Graphmode 3
Rastermodus 12	dem Graphmode 4

Wie sich die einzelnen Modi auswirken, können Sie anhand des kleinen Beispielprogramms testen. Drücken Sie dabei keine Taste, wird das neue Bild pro Durchlauf nur einmal an der Mausposition gezeichnet. Drücken Sie die linke Maustaste, wird es zweimal hintereinander an die gleiche Mausposition gezeichnet. Drücken Sie die rechte Maustaste, wird der nächste Modus eingeschaltet.

```
Deffill ,3,2
Pbox 0,0,639,399
Get 10,10,50,50,Ff$
For I=0 To 15
  Print At(20,6);"Rastermodus : ";I
  Repeat
    Mouse X,Y,K
    Put X,Y,Ff$,I
    If K=1
      Put X,Y,Ff$,I
    Endif
  Until K=2
  Pbox 0,0,639,399
Next I
Edit
```

Sie sollten bei der Angabe der Rastermodi vermeiden, daß evtl. ein Minuswert dafür eingesetzt wird. Dies schmeckt dem Interpreter nicht und er bedankt sich mit unangenehmen Fehlfunktionen.

Algemeines zum Bildschirmspeicher:

Der Monochrom-Bildschirm wird genaugenommen in ein Feld von 400 Zeilen zu je 80 Byte aufgeteilt. Diese Zeilen hängen wie eine lange Kette aneinander. Das erste Byte des Bildschirms (das ist übrigens die Adresse, die mit "Xbios(2)" erfragt werden kann), sind die ersten acht Pixels in der ersten Zeile von oben. Das zweite Byte folgt rechts davon usw. bis zum Byte 80, das in der ersten Zeile ganz rechts außen liegt. Das Byte 81 liegt dann wieder auf der linken Bildschirmseite und zwar in der zweiten Zeile von oben. Byte 82 folgt dann wieder rechts davon usw.. Auf diese Art setzt sich der Bildschirm mit der Auflösung von 640 Pixels (80 Byte zu je 8 Bit = 640) in der Breite und 400 Pixels (400 Zeilen) in der Höhe zusammen, bis die Bytekette in der rechten unteren Bildschirmecke angekommen ist. Beim mitelauflösenden Bildschirm und beim niedrigauflösenden Bildschirm gilt das gleiche Verfahren, nur daß hier für die Farbbestimmung noch zusätzlich Bytes benötigt werden, die die Informationen dazu beinhalten. Deshalb kann ein hochauflösender ST-Bildschirm mit 640*400 Punkten auch nicht mehr als 2 Farben darstellen. Der für den Bildschirm vorgesehene Speicherplatz (400 Zeilen zu je 80 Byte = 32000 Byte) ist hier schon allein durch die Farbbestimmung (schwarz = Bit ist 1 / weiß = Bit ist 0) voll ausgenutzt. Das Problem wäre gelöst, wenn man wesentlich mehr Speicherplatz zu Verfügung stellen könnte. Bei Groß-Computern ist z.B. eine Farbauswahl mit weit über 100.000 Farben bei einer Auflösung von weit über 1000*1000 Bildpunkte ohne weiteres möglich (s. Computergrafik-Effekte im Fernsehen).

```
Next I%  
Endif
```

If Menstartflag%=0

'Hauptmenü wird zum erstenmal aufgerufen.

Color 1

Defline 0,1,0,0

Deffill ,0,0

'Zuerst müssen die nötigen Einstellungen vorgenommen werden.

DEFFILL

Hier ist der Befehl "Deffill" neu. Mit diesem Befehl können drei Parameter übergeben werden, welche das aktuelle Füllmuster bestimmen.

Erster Parameter : Füllmusterfarbe

Zweiter Parameter : Füllmusterstil

0 = mit Hintergrundfarbe ausgefüllt

1 = mit Objektfarbe ausgefüllt

2 = mit Objektfarbe punktiert

3 = mit Objektfarbe schraffiert

4 = Aufruf des selbstdefinierten Füllmusters

Solange kein eigenes Füllmuster definiert wurde,
erscheint hier das ATARI-Zeichen.

Dritter Parameter : Füllmustersauswahl

Es stehen insgesamt 36 Füllmuster intern zur Verfügung, wobei die beiden ausgefüllten Füllstile nicht berücksichtigt sind.

Im Füllstil 2 sind es die Muster 1 bis 24

Im Füllstil 3 die Muster 1 bis 11

Im Füllstil 4 das ATARI-Zeichen oder ein eigenes

Über die Variation:

Deffill (Füllfarbe), F.muster\$

läßt sich ein eigenes Füllmuster einrichten. Ähnlich wie bei der Mausformdefinition wird hier ein 32 Bytes langer String einge-

richtet. Das Füllmuster besteht hier ebenfalls aus 16*16 Pixels. Da hier keine Maske oder Aktionspunkt definiert werden müssen, brauchen nur 16 Words übergeben werden, die der Reihe nach die Bitmuster der einzelnen Füllmusterzeilen beinhalten. Dies geschieht wieder am besten über die Funktion "Mki\$"

z.B.:

```
Dim Fm.zeile%(16)
Fm.zeile%(1) = &X1111111111111111
Fm.zeile%(2) = &X1000000000000000
Fm.zeile%(3) = &X1000000110000000
Fm.zeile%(4) = &X1000001001000000
Fm.zeile%(5) = &X1000010000100000
Fm.zeile%(6) = &X1000100000010000
Fm.zeile%(7) = &X1001000110001000
Fm.zeile%(8) = &X1010001111000100
Fm.zeile%(9) = &X1001000110001000
Fm.zeile%(10) = &X1000100000010000
Fm.zeile%(11) = &X1000010000100000
Fm.zeile%(12) = &X1000001001000000
Fm.zeile%(13) = &X1000000110000000
Fm.zeile%(14) = &X1011111111111110
Fm.zeile%(15) = &X1001111111111000
Fm.zeile%(16) = &X1000000000000000
For I% = 1 To 16
    F.muster$ = F.muster$ + Mki$(Fm.zeile%(I%))
Next I%
Deffill 1, F.muster$
Pbox 100, 100, 200, 200
U=Inp(2)
Edit
```

Sie können das Füllmuster auch mit dem, im Programm integrierten Pattern-Editor entwerfen und sich die Werte ausgeben lassen.

BIN\$ / HEX\$ / OCT\$ / &H / &X / &O

Die einzelnen Zeilen wurden hier nach einem anderen Verfahren aufgeschlüsselt, als Sie es bei der Definition des Linienstils gesehen haben. GfA-BASIC bietet nämlich "schon wieder" eine kleine Raffinesse an, die tatsächlich große Wirkung hat. Aber dazu erstmal eine kleine Einführung in die verschiedenen Zahlensysteme.

Es gibt im Computerbereich vier Arten von Zahlensystemen:

- das dezimale System
- das hexadezimale System
- das binäre System
- das oktale System

Das allgemein übliche Dezimalsystem hat die Zahl 10 zur Basis (dezi = 10 - z.B. 1 Dezimeter=10 Zentimeter). Diese Basis wird jeweils zur Ermittlung der Wertigkeit einer Zahl herangezogen. So werden die Einerstellen aus der Null-Potenz der Zahl 10 ermittelt ($10 \text{ hoch } 0 = 1$), die Zehnerstellen aus der Einer-Potenz ($10 \text{ hoch } 1 = 10$), die Hunderterstellen aus der Zweier-Potenz ($10 \text{ hoch } 2 = 100$) usw..

In den anderen Zahlensystemen ist dieser Potenzierungsvorgang exakt derselbe, nur werden hier andere Zahlen als Basis verwendet. So wird im Hexadezimalsystem (Hexa (6) + Dezi (10) ==> Hexadezi (16)) die Zahl 16 als Basis verwendet, im Binärsystem (Bi = 2) die Zahl 2 und im Oktalsystem (Okta = 8) die Zahl 8. So erklären sich die geringen Wertigkeiten der Binärzahlen in ihren Stellen und die hohen Wertigkeiten der Hexadezimalzahlen.

usw. <==	Stelle 3:		Stelle 2:		Stelle 1:
Dezimal	: $10 \text{ hoch } 0 = 1$	;	$10 \text{ hoch } 1 = 10$	;	$10 \text{ hoch } 2 = 100$
Binär	: $2 \text{ hoch } 0 = 1$	;	$2 \text{ hoch } 1 = 2$	;	$2 \text{ hoch } 2 = 4$
Hexadezimal:	$16 \text{ hoch } 0 = 1$	;	$16 \text{ hoch } 1 = 16$	;	$16 \text{ hoch } 2 = 256$
Oktal	: $8 \text{ hoch } 0 = 1$	;	$8 \text{ hoch } 1 = 8$	;	$8 \text{ hoch } 2 = 64$

Bei den Hexadezimalzahlen gibt es eine Besonderheit zu beachten. Da sich die Zahl 16 nicht mit arabischen Zahlen einstellig darstellen läßt, hat man zu einem Trick gegriffen. Die Hex-Zahlen 0 - 9 werden genauso dargestellt wie im Dezimalsystem. Die Zahlen von 10 bis 15 werden dagegen durch einen Buchstaben repräsentiert (A=10; B=11; C=12; D=13; E=14; F=15). Daher also die Buchstaben in Hexadezimalzahlen.

Die Dezimalzahl 1037 setzt sich also wie folgt zusammen:

$$\begin{array}{rcl}
 (10 \text{ hoch } 0) * 7 & = & 7 \\
 + (10 \text{ hoch } 1) * 3 & = & 30 \\
 (die \text{ dritte Stelle ist nicht besetzt, also: } 0) & & \\
 + (10 \text{ hoch } 3) * 1 & = & 1000 \\
 & & \text{-----} \\
 \text{Summe :} & & 1037
 \end{array}$$

Die Binärzahl 100011 ergibt dagegen nach der Wertigkeit ihrer Stellen: (immer von rechts gesehen, wie im Dezimalsystem auch)

$$\begin{array}{rcl}
 (2 \text{ hoch } 0) * 1 & = & 1 \\
 + (2 \text{ hoch } 1) * 1 & = & 2 \\
 (die \text{ dritte Stelle ist nicht besetzt, also: } 0) & & \\
 (die \text{ vierte Stelle ist nicht besetzt, also: } 0) & & \\
 (die \text{ fünfte Stelle ist nicht besetzt, also: } 0) & & \\
 + (2 \text{ hoch } 5) * 1 & = & 32 \\
 & & \text{-----} \\
 \text{Summe :} & & 35
 \end{array}$$

Das gleiche für die Hexadezimalzahl F3B0: (die erste Stelle ist nicht besetzt, also: 0)

$$\begin{array}{rcl}
 (16 \text{ hoch } 1) * 11 & = & 176 \\
 + (16 \text{ hoch } 2) * 3 & = & 2816 \\
 + (16 \text{ hoch } 3) * 15 & = & 61440 \\
 & & \text{-----} \\
 \text{Summe :} & & 64432
 \end{array}$$

```

(8 hoch 0) * 7   =    7
+ (8 hoch 1) * 3   =   24
+ (8 hoch 2) * 5   =  320
  (die vierte Stelle ist nicht besetzt, also: 0)
+ (8 hoch 4) * 1   = 4096
      -----
Summe :           4447

```

Die Dezimalzahl 1352 soll in das binäre Format verwandelt werden.

1352 : 2	=	676	Rest 0
676 : 2	=	338	Rest 0
338 : 2	=	169	Rest 0
169 : 2	=	84	Rest 1
84 : 2	=	42	Rest 0
42 : 2	=	21	Rest 0
21 : 2	=	10	Rest 1
10 : 2	=	5	Rest 0
5 : 2	=	2	Rest 1
2 : 2	=	1	Rest 0
1 : 2	=	0	Rest 1
0 : 2	=	0	Rest 0

Die Binärzahl lautet:

Nach dem gleichen Schema lassen sich auch Dezimalzahlen in die übrigen beiden Zahlensysteme konvertieren.

Dazu noch das Beispiel für Hexadezimalzahlen:

Die Dezimalzahl 35117 soll in das Hexadezimalformat gewandelt werden.

35117 : 16 = 2194 =>	2194 * 16 = 35104 =>	Diff. = 13 (D)	_____
2194 : 16 = 137 =>	137 * 16 = 2192 =>	Diff. = 2 (2)	_____
137 : 16 = 8 =>	8 * 16 = 128 =>	Diff. = 9 (9)	_____
8 : 16 = 0 =>	0 * 16 = 0 =>	Diff. = 8 (8)	_____
0 : 16 = 0 =>	0 * 16 = 0 =>	Diff. = 0	_____
Die Hexadezimalzahl lautet :			8 9 2 D

Soviel zur Grundlagenvermittlung. Bei derart komfortablen Sprachen wie dem GfA-BASIC hat man es natürlich nicht mehr nötig, diese Zahlen auf diese Art und Weise zu ermitteln. Trotzdem kann es unter Umständen von Vorteil sein, sich damit einigermaßen auszukennen.

Das Hexadezimale Zahlensystem wurde aus einem bestimmten Grund entwickelt. Es lassen sich nämlich "zufälligerweise" genau die Inhalte von vier Bits (Tetrade) mit einer Hex-Zahl darstellen.

Die Binärzahl 1111 kann maximal den Wert 15 ($2^0+2^1+2^2+2^3$) annehmen. Und genau dieser Wert läßt sich auch maximal mit einer Hex-Zahl darstellen (F). Ein Byte (8 Bit) wird demnach immer zu einer Zweier-Tetrade ($2*4=8$), ein Word (16 Bit) zu einer Vierer-Tetrade ($4*4=16$) und ein Longword (32 Bit) immer zu einer Achter-Tetrade ($8*4=32$) zusammengefaßt.

Nun zu der Besonderheit. Wie vielleicht bekannt, können in den meisten neueren BASIC-Dialekten Hexadezimalzahlen mit dem vorangestellten Kürzel "&H" direkt übergeben werden.

Im GfA-BASIC kann man zusätzlich noch Oktalzahlen (Kürzel: "&O") und Binärzahlen (Kürzel: "&X") direkt angeben. Genau das ist in dem oben aufgeführten Beispiel geschehen. Weiterhin gibt es die Möglichkeit, sich die verschiedenen Zahlenarten in ein gewünschtes anderes Zahlensystem umwandeln zu lassen. Das geschieht über verschiedene Stringfunktionen.

"Bin\$" wandelt die angegebene Zahl in das Binärformat

"Hex\$" wandelt die angegebene Zahl in das Hexadezimalformat

"Oct\$" wandelt die angegebene Zahl in das Octalformat

Die entstehenden Ausdrücke können z.B. mit einem Print-Befehl direkt ausgegeben, einer Stringvariablen übergeben oder in einen Stringausdruck eingebunden werden.

Studieren Sie folgendes Beispiel ausführlich und Sie werden in Zukunft wesentlich weniger Probleme mit komplizierten Zahlenumformungen haben.

```
A.a$="11011010"
B.b$="F81A"
C.c$="5430"
Aaa%=Val("&X"+A.a$)
Bbb%=Val("&H"+B.b$)
Ccc%=Val("&O"+C.c$)
Aa$=Bin$(Aaa%)
Bb$=Hex$(Bbb%)
Cc$=Oct$(Ccc%)
Print A.a$'B.b$'C.c$
Print Aaa%'Bbb%'Ccc%
Print Aa$'Bb$'Cc$
Print Bin$(6223)
Print Hex$(6223)
Print Oct$(6223)
Print &X11100101
Print &HF51C
Print &O4437
U=Inp(2)
Edit
```

MKIS / MKLS / MKSS / MKFS / MKDS

Mit diesen Stringfunktionen können beliebige numerische Werte in einen, der Wertgröße und dem gewünschten Format entsprechenden, Stringausdruck umgewandelt werden. Der Variablenaufbau der verschiedenen Systeme, Interpreter und Compiler

kann sich voneinander stark unterscheiden. Unter "ARRPTR" wurde die Variablenorganisation des GfA-BASIC-Interpreters erläutert. Daraus ist erkennbar, daß selbst hier schon mit verschiedenen Zahlenformaten gearbeitet werden muß. Um nicht zeitaufwendige Rechenoperationen ausführen zu müssen, die einzelnen Bytewerte in das benötigte Format zu übertragen, können diese Komfortfunktionen verwendet werden.

Mki\$ - wandelt einen 16-Bit(2Byte)-Integerwert in einen 2 Byte langen Stringausdruck.
Mkl\$ - wandelt einen 32-Bit(4Byte)-Integerwert in einen 4 Byte langen Stringausdruck.
Mks\$ - wandelt einen beliebigen Wert in einen Atari-BASIC kompatiblen 4 Byte Stringausdruck.
Mkf\$ - wandelt einen beliebigen Wert in das GfA-BASIC eigene 6 Byte lange Realzahlenformat.
Mkd\$ - wandelt einen beliebigen Wert in einen MBASIC kompatiblen 8 Byte Stringausdruck.

Die so gewandelten Werte können nun Byteweise ausgewertet werden.

z.B.:

```
A$=Mkl$(346769123)
For I%=0 To 3
  Print Peek(Varptr(A$)+I%);
Next I%
```

Ein weiteres Beispiel hierzu folgt unter "CVI, CVL, CVS, CVF, CVD".

CVI / CVL / CVS / CVF / CVD

Dieses sind die entsprechenden Umkehrfunktionen zu den eben beschriebenen Umwandlungsfunktionen.

Es werden die den einzelnen Funktionen entsprechenden Längen der angegebenen Stringausdrücke unter Berücksichtigung der einzelnen Formatvorgaben in numerische Werte umgewandelt.

z.B.:

```
A=126422.6237
```

```
A1$=Mki$(A)
```

```
A2$=Mkl$(A)
```

```
A3$=Mks$(A)
```

```
A4$=Mkf$(A)
```

```
A5$=Mkd$(A)
```

```
A.1=Cvi(A1$)
```

```
A.2=Cvl(A2$)
```

```
A.3=Cvs(A3$)
```

```
A.4=Cvf(A4$)
```

```
A.5=Cvd(A5$)
```

```
Print "Ursprungswert : ";A
```

```
Print "Stringformate : ";A1$'A2$'A3$'A4$'A5$
```

```
Print "Rückwandlung : ";A.1'A.2'A.3'A.4'A.5
```

```
Print Mkf$(Random(1000000)+Rnd)
```

```
Print Cvf("BASIC.")
```

INP

Diesem Befehl sind Sie bisher in den Beispielprogrammen schon öfter begegnet. Er eignet sich in diesen Fällen besonders dazu, den Programmlauf zu unterbrechen und auf einen Tastendruck zu warten. Seine eigentliche Funktion ist jedoch, jeweils ein einzelnes Byte aufzunehmen, das entweder von einer Disketten-datei oder von den Ports gesendet wird. Hier entwickelt er erst seine wahren Qualitäten.

In einer nachgestellten Klammer wird ihm entweder die Nummer des Datenkanals oder eine Identifikationsziffer für die einzelnen Ports übergeben. Dabei ist Vorsicht geboten. Dieser Befehl läßt sich in seinem Eifer, auf ein Byte zu warten auch nicht von der Abbruch-Funktion "Control/Shift/Alternate" beirren.

Wurde er erstmal aufgerufen und ist auf dem angegebenen Port kein Byte zu finden, hilft nur noch der Resetknopf.

Die Ports haben folgende Identifikatoren:

- 0 = LST: (Druckerport)
- 1 = AUX: (Auxilliary-Port)
- 2 = CON: (Consolport = Tastatur)
- 3 = MID: (MIDI-Port)
- 4 = IKB: (Keyboard-Prozessor)
- 5 = VID: (Videoport = Monitor)

z.B.:

- X=Inp(2) - wartet auf einen Tastendruck und speichert deren ASCII-Wert in "X".
- X=Inp(3) - wartet darauf, daß ein Byte am MIDI-Port anliegt und speichert dessen Wert in "X". (MIDI = Music Instruments Digital Interchange)
- Print Inp(#1) - gibt den ASCII-Wert des Zeichens einer offenen Datei mit der Kanalnummer #1 aus, auf dem sich der aktuelle Filepointer befindet.

OUT

Das Gegenstück zu "Inp" ist der Befehl "Out". Hiermit lassen sich Bytewerte an einen bestimmten Datenkanal oder Port senden.

Die Syntax: Out Ziel,Bytewert (0-255)

z.B.:

Out 5,122 - gibt das Zeichen "z" (ASCII-Wert=122) an der aktuellen Cursorposition (!) auf dem Monitor aus (somit lassen sich auch die ersten 32 Steuerzeichen auf dem Bildschirm beliebig placieren)

Out 4,18 - schaltet Mauskontakt ab (Maus ist inaktiv)

Out 4,8 - schaltet Mauskontakt ein (Maus ist wieder aktiv) Diese beiden auf den Keyboard-Prozessor gerichteten Befehle geben nur ein kleines Spektrum der Möglichkeiten wieder, diesen Prozessor zu steuern. Dies ist jedoch ein sehr kompliziertes und umfangreiches Thema, da meistens gleich mehrere "Out"-Befehle hintereinander zu einem Prozessorbefehl kombiniert werden müssen. Es soll hier nicht weiter darauf eingegangen werden. Außerdem bedankt sich IKB: bei falscher Bedienung mit unvorhersehbaren Reaktionen (also vorsichtig sein).

Out #1,65 - schreibt den ASCII-Wert des Zeichens "A" (65) an die aktuelle Filepointerposition in der Datei mit der angegebenen Kanalnummer.

STR\$

Ebenfalls eine Funktion, die Ihnen in den Beispielen schon begegnet ist, ist die Textfunktion "Str\$". Mit dieser Funktion lassen sich numerische Werte in einen Textausdruck umwandeln. Der entstehende String hat anschließend die Länge, die der Anzahl der Ziffern des übergebenen Wertes im Dezimalformat entspricht. Als Hexadezimal-, Binär- oder Octalzahl angegebene Werte werden also vorher in das Dezimalformat umgewandelt.

z.B.:

```
A=1577.69271
```

```
Print Str$(A)
```

```
Ausgabe : 1577.69271
```

Die hier ausgegebene Ziffernfolge ist keine Zahl mehr, die einen Wert darstellt, sondern lediglich ein Textstring, der die einzelnen Ziffern des Wertes als Textzeichen enthält.

```
Print Str$(Val("&HE5AF"))
```

```
Ausgabe : 58799
```

Diese Zahl ist ebenfalls kein Wert mehr, sondern eine Folge von 5 Textzeichen.

VAL

Dieses ist nun die Umkehrfunktion zu "Str\$". Sie wandelt in Strings enthaltene Ziffernzeichen in numerische Werte um. Allerdings nur, soweit diese Ziffernzeichen am Anfang des Strings stehen. Eine Ausnahme bilden die Zeichen &H, &X, &O. Sie gelten als Teil der Zahlen. Bei so angegebenen Hexadezimalzahlen werden auch die nächsten Zeichen, die sich als Hexadezimalzahl verwerten lassen (A-F) mit aufgenommen.

```
z.B.      Print Val("&HEEZeichen")

           Ausgabe : 238 (&HEE=238)

           A$="&X110123E"
           Print Val(A$)

           Ausgabe : 13 (&X1101=13)

           Print Val("&2.37E+07")

           Ausgabe : 23700000
```

VAL?

Mit dieser Funktion kann die Anzahl der Zeichen eines Strings vom Anfang des Strings aus ermittelt werden.

z.B.:

```
Print Val?("&HEEZeichen")
```

Ausgabe : 4 (die ersten vier Zeichen)

```
Print Val?("237E07")
```

Ausgabe : 6 (Exponentialformat)

```
A$="&X110123E"
```

```
Print Val?(A$)
```

Ausgabe : 6 (incl. Identifikator)

Pbox 0,0,639,40

Pbox 0,359,639,399

PBOX

Der "Pbox"-Befehl dient dazu, mit dem eingestellten Füllmuster ausgefüllte Rechtecke zu zeichnen. Diese Rechtecke können mit beliebigen Koordinatenpunkten definiert werden (auch im Minusbereich). Mit dem schon angedeuteten "Clipping Rectangle" werden diese vom Interpreter allerdings an den Bildschirmrändern abgeschnitten. Das heißt nicht, daß sie sonst außerhalb des Bildschirms darstellbar wären. Nur kann der Befehl dann Systemintern kein Unheil anrichten, da auch bei diesem Befehl mit Rasteroperationen gearbeitet wird. Sinnvoll ist es meistens sowieso nur, die Koordinaten innerhalb des Bildschirms zu legen. Der erste Parameter dieses Befehls beschreibt die auf die linke obere Bildschirmecke (0/0) bezogene X-Koordinate der linken oberen Ecke der Box. Der zweite ist die Y-Koordinate desselben Punktes. Der dritte Parameter bestimmt die, ebenfalls auf die linke obere Bildschirmecke bezogene X-Koordinate des rechten unteren Eckpunktes der Box, sowie der vierte die Y-Koordinate desselben Punktes darstellt. Somit sind alle vier Seiten der rechteckigen Box definiert.

X steht hier übrigens stellvertretend für die horizontale (waagerechte) und Y für die vertikale (senkrechte) Ausrichtung. Diese Box wird mit einer Umrandung von der Stärke eins Pixels gezeichnet. Diese Umrandung läßt sich von Interpreter-Seite nicht ein- oder ausschalten. Dies kann mit einer GEM-Funktion bewerkstelligt werden, die Sie unter dem Namen "Procedure Rahmen(Visibel%)" im Programmlisting finden.

```
For I%=0 To 600 Step 80
```

```
'In dieser Schleife werden die einzelnen Menüfelder gezeichnet.
```

```
Box I%+2,2,I%+78,38
```

```
Box I%+4,4,I%+76,36
```

```
Box I%+2,361,I%+78,397
```

```
Box I%+4,363,I%+76,395
```

BOX

Der "Box"-Befehl ist prinzipiell gleich zu behandeln, wie der eben beschriebene Pbox-Befehl. Er unterscheidet sich durch zwei Eigenarten von diesem. Erstens muß kein Füllmuster vorher definiert werden, da die "Box" nur als Rahmen ohne Inhalt gezeichnet wird und zweitens wirken sich die Linien-Einstellungen auf diese Box aus. Es kann also ebenfalls mit "Defline" der Stil, die Dicke und die Endformen eingestellt werden. Dabei ist das linke Ende der oberen Seite als Anfang und das obere Ende der linken Seite als Ende anzusehen.

```
Next I%
```

```
Graphmode 2
```

```
'Da beim Einsetzen der folgenden Textzeilen im Replace-Modus der  
'Rand der Menüfelder evtl. überlagert würde, muß hier der Trans-  
'parent-Modus verwendet werden
```

```
Deftext 1,1,0,6
```

DEFTEXT

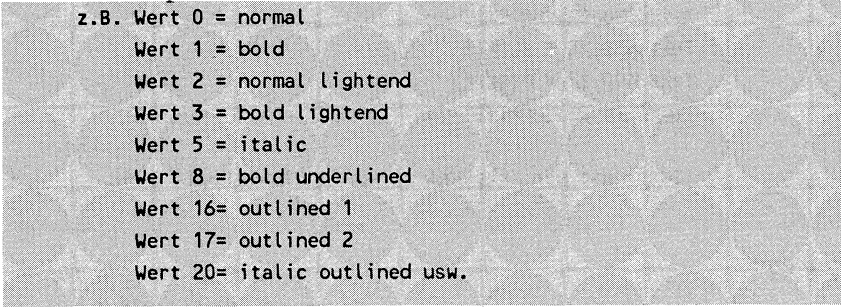
Mit "Deftext" lassen sich die verschiedenen Darstellungsweisen der mit dem Befehl "Text" ausgegebenen Zeichen bestimmen. Auf die Textausgabe mit dem "Print"-Befehl haben diese Einstellungen keinen Einfluß. Dieser läßt sich nur über verschiedene TOS-Befehle modifizieren. Allerdings können z.B. die in Alert- und Fileselect-Boxen verwendeten Texte über einen GEM-Befehl verändert werden. Ebenso die darin verwendeten Linien. Diese lassen sich also bis zu einem gewissen Grad individuell verändern. Die GEM-Prozeduren dazu finden Sie im Anhang B.

Die Parameter haben folgende Bedeutung:

Parameter 1 : Bestimmung des Registers, aus welchem der Text seine Farbe beziehen soll

Parameter 2 : Einstellung der Textart

Werte von 0 bis 31



z.B. Wert 0 = normal
Wert 1 = bold
Wert 2 = normal lightend
Wert 3 = bold lightend
Wert 5 = italic
Wert 8 = bold underlined
Wert 16= outlined 1
Wert 17= outlined 2
Wert 20= italic outlined usw.

Setzt man übrigens den Text einmal mit Wert 16 und einmal mit Wert 17 auf dieselbe Position, erscheint der Text schattiert. Dazu muß allerdings der Graphmode 2 eingeschaltet werden. Mit diesem Graphmode lassen sich alle Textarten miteinander kombinieren.

Parameter 3 : Einstellung des Textrotationswinkels

Wert 0 = normale rechtsausgerichtete Lage.
 Wert 900 = Text wird senkrecht von unten nach oben ausgegeben.
 Wert 1800 = Text wird auf dem Kopf stehend waagrecht von rechts nach links ausgegeben.
 Wert 2700 = Text erscheint senkrecht von oben nach unten.

Parameter 4 : Einstellung der Texthöhe

Werte von 0 bis 26

z.B. Wert 0 = nichts
 Wert 4 = Desktop-Icon-Schrift
 Wert 6 = Farbschrift (8*8)
 Wert 13 = Normalschrift (8*16)
 Wert 26 = Mammutschrift

Text 10,22," QUIT"
 Text 90,22," INITS"
 Text 170,22," EDITS"
 Text 253,22," KINO"
 Text 330,22," SPEZI"
 Text 410,22," FLIPS"
 Text 490,22," DISKY"
 Text 570,22," PRINT"

'Setze die Menütitel an die gewünschten Positionen.

TEXT

Durch die Textausgabefunktion " Text X,Y,L,"String" " kann an einer beliebigen Bildschirmposition in der durch "Deftext" eingestellten Darstellungsform Text ausgegeben werden. Diese Textausgabe ist allerdings zu 100 Prozent bildschirmorientiert. D.h., daß z.B. hierdurch kein Text auf die Diskette oder den Drucker gebracht werden kann. Die ersten beiden Parameter geben die X- und Y-Koordinaten an, an denen der Text erscheinen soll.

ausschlaggebende Punkt ist immer der, der bei normaler Lage des Textes (waagrecht von links nach rechts) in der Zeichenbox des ersten Zeichens (auch bei Leerzeichen) links unten liegt. Mit diesem Punkt wird der Text generell an die angegebene X/Y-Position angelegt. Der dritte Parameter dient der Formatierung des Textes. Hier kann auf drei verschiedene Arten festgelegt werden, ob der Text

- a) in normaler Länge (Zeichenboxbreite * Anz.d.Zeichen) Parameter 3 = 0
- b) in vorgegebener Länge mit Zeichendehnung Parameter 3 = positive Textlänge in Pixels
- c) in vorgegebener Länge mit Wortdehnung Parameter 3 = negative Textlänge in Pixels ausgegeben wird.

Das folgende Beispiel verdeutlicht, was gemeint ist:

```

Deffill ,2,1
Pbox 0,0,639,399
Deftext 1,16,0,16
For I%=1 To 4
  Graphmode I%
  Text 10,20+I%*30,0,"TEXT (normal) GRAPHMODE "+Str$(I%)
  Text 10,140+I%*30,600,"TEXT (Zeichendehng.) GRAPHMODE
"+Str$(I%)
  Text 10,260+I%*30,-600,"TEXT (Wortdehnung) GRAPHMODE "+Str$(I%)
Next I%
U=Inp(2)
Edit

```

```
Get 0,0,639,40,Upmenue$
```

```
Get 0,359,639,399,Dwnmenue$
```

'halte die beiden Menü-Leisten als Bild fest.

'Hier werden die beiden Menüleisten-Bilder nach dem ersten Aufruf

'des Hauptmenüs, wo sie ja zum ersten Mal gezeichnet werden, als

'Bild abgespeichert. Dies sind die beiden Bilder, die später bei

'jedem erneuten Menüaufruf einfach mit der am Procedur-Anfang be-

'schriebenen Put-Schleife auf den Bildschirm "geschoben" werden. Sie

'brauchen also jetzt nicht mehr neu gezeichnet werden.

@Set.paras

'In der Prozedur "Setparas" werden die vom Anwender vorgenommenen
'Grafik-Einstellungen wieder eingesetzt, so daß sich durch den
'Aufruf des Menüs daran nichts verändert.

Menstartflag%=1

'Merke Dir, daß das Hauptmenü schon einmal gezeichnet wurde und nun
'als Bild jederzeit abrufbar ist.

Endif

Repeat

Mouse X,Y,K

Until Len(Inkey\$) Or Mousek

'Wiederhole diese Repeat/Until-Schleife solange, bis entweder eine
'Maustaste oder eine Taste der Tastatur gedrückt wurde und merke
'Dir dabei die Mausposition und den Zustand der Mausknöpfe.

REPEAT / UNTIL

Eine für BASIC-Dialekte ungewöhnliche Schleifenart ist die "Repeat/Until"-Schleife. Diese Schleife kann immer dann angewendet werden, wenn die Anzahl der Schleifendurchläufe nicht durch das Erreichen eines Endwertes (For/Next) festgelegt, und die Schleife mindestens einmal durchlaufen werden soll.

Die Bedingung zum Verlassen der Schleife wird hier am Schleifenende geprüft. D.h., daß die Schleife mindestens einmal bis zu der in "Until" vereinbarten Bedingung durchlaufen wird. Es sei denn, daß innerhalb der Schleife eine "Exit If"-Bedingungsabfrage mit "wahr" beantwortet wird (Exit/If s. unten).

Diese Konstruktion läßt sich bestens dazu verwenden, um mehrfach kombinierte Abbruchbedingungen zu stellen. So können z.B. gleichzeitig ein bestimmter Status der Tastatur, der

Maus und der Mausknöpfe, der Inhalt von Variablen und das Erreichen von Werten als Bedingung angegeben werden.

Angenommen, für den Abbruch einer Repeat/Until-Schleife sollen die folgenden Bedingungen ausschlaggebend sein:

- rechte Maustaste gedrückt und Variablenwert größer 100
- oder Esc-Taste gedrückt
- oder die Differenz zweier Timer-Aufrufe soll größer 2000 sein.

Die Schleifenkonstruktion sieht dann so aus:

```
T%=Timer
Repeat
  Inc A%
  Mouse X,Y,K
  K_ey$=Inkey$
  Key%=Asc(Left$(K_ey$))+Asc(Right$(K_ey$))
Until (K=2 And A%>100) Or Key%=54 Or Timer-T%>2000
Edit
```

Mit den Bool'schen Operatoren And/Or/Not/Xor/Imp/Eqv lassen sich diese Bedingungen auf's abenteuerlichste miteinander verknüpfen. Ich werde an anderer Stelle in diesem Buch noch einmal ausführlicher auf diese logischen Verknüpfungen eingehen. Es ist jedoch unmöglich, eine umfassende Lektion zusätzlich in dieses Buch mit aufzunehmen. Wer sich intensiver mit dieser Thematik auseinandersetzen möchte, sollte sich ein Buch beschaffen, das sich ausschließlich damit befaßt. Der Inhalt der als Beispiel gezeigten Schleifenkonstruktion wird im folgenden genauer erläutert.

ASC / CHR\$

Diese beiden Textfunktionen sind jeweils die Umkehrfunktionen voneinander. Mit "Asc" kann der "ASCII"-Code (American Stan-

dard Code for Information Interchange) eines Zeichens ermittelt werden.

```
z.B.:  
  
A$="D"  
Print Asc(A$)  
  
Ausgabe: 68  
  
A%=Asc("E")  
Print A%  
  
Ausgabe: 69  
  
Print Asc("BASIC")  
  
Ausgabe: 66
```

Im letzten Fall wird nur der ASCII-Wert des ersten Zeichens eines Strings zurückgegeben. Ist im String kein Zeichen enthalten, gibt die Funktion den Wert 0 zurück.

Die "Chr\$" -Funktion ermittelt ihrerseits das zu einem bestimmten ASCII-Wert gehörige Textzeichen.

```
z.B.:  
  
A$=Chr$(68)  
Print A$  
  
Ausgabe: D  
  
Print Chr$(66)  
  
Ausgabe: B
```

Die ersten 32 ASCII-Werte (0-31) sind durch Steuerzeichen belegt und können durch Chr\$ nicht dargestellt werden. Es sind insgesamt (incl. der Steuerzeichen) 256 ASCII-Zeichen (0-255)

vorhanden. Am Anhang finden Sie eine Auflistung aller ASCII-Zeichen und ihre Codes.

INC / DEC

Durch diese beiden arithmetischen Funktionen lassen sich die Inhalte numerischer Variablen entweder durch "Inc" um 1 erhöhen (erhöhen = incrementieren) oder durch "Dec" um 1 vermindern (vermindern = decrementieren). Andere Schrittweiten sind hier nicht einstellbar.

z.B.:

A%=10

Inc A%

Print A%

Dec A%

Dec A%

Print A%

Ausgabe : 11
 9

ADD / SUB / MUL / DIV

Wollen Sie die Schrittweite beim Erhöhen oder Vermindern variieren können, haben Sie die Möglichkeit durch "Add Var,Wert" eine entsprechende Incrementierung oder durch "Sub Var,Wert" eine Decrementierung des Variableninhalts vorzunehmen. Als "Wert" ist hier auch die Angabe von Fließkommazahlen (positiv und negativ) möglich.

z.B.:

A=33

Add A,12.55

Print A

Sub A,-27.21

```

Print A
Sub A,36
Print A

Ausgabe : 45.55
          72.76
          36.76

```

"Mul Var,Wert" hat die Aufgabe, einen bestimmten Variableninhalt mit einem bestimmten Wert malzunehmen (multiplizieren). "Div Var,Wert" ist die entsprechende Teilungsfunktion (Division).

Der wesentliche Vorteil an diesen arithmetischen Funktionen ist der, daß durch ihre Maschinennähe Geschwindigkeitsvorteile von 30% (bei Mul und Div) bis 100% (bei Add und Sub) in Konkurrenz zu den vergleichbaren Funktionen (Var=Var +/-*/Wert) erzielt werden können.

WHILE / WEND

Eine weitere Schleifenkonstruktion ist die "While/Wend"-Schleife. Diese Schleifenart hat die hervorstechende Eigenart, daß die Bedingungsabfrage zum Verlassen der Schleife bereits am Anfang der Schleife stattfindet. So kann es sein, daß die Schleife zwar in der ersten Zeile betreten wird, jedoch sofort in der Zeile nach dem dazugehörigen "Wend" fortgefahren wird. Dann nämlich, wenn die Abbruchbedingung bei Betreten der Schleife in "While Bedingung" bereits erfüllt ist.

```

z.B.:

Enter:
A%=Random(100)
B%=32*12.5+int(A%/.66)
While B%<440
    Inc B%
    Print B%
Wend

```

```
Print "Schleife verlassen !"  
Pause 20  
Goto Enter
```

DO LOOP

Dies ist eine Schleifenkonstruktion, die keine Abbruchbedingung vorsieht. D.h., daß sie solange durchlaufen wird, bis sie auf andere Abbruch-Anweisungen trifft (End, Edit, Stop, Exit If) oder durch die Tastenkombination "Control/Shift/Alternate" das Programm abgebrochen wird. Tritt keiner dieser Fälle auf, wird sie endlos durchlaufen.

```
z.B.:  
  
Do  
    Print "***";  
Loop
```

EXIT IF

Eine sehr nützliche Sonderfunktion, um in For/Next-, Do/Loop-, Repeat/Until- und While/Wend-Schleifen an bestimmten Stellen in dem zwischen Einstiegs- und Wendepunkt eingeschlossenen Programmblock zusätzliche Abbruchbedingungen zu stellen, ist der Befehl "Exit If Bedingung". Mit diesem Befehl kann unabhängig vom Zustand der Schleife diese jederzeit verlassen werden, wenn die durch "Exit If" gestellte Bedingung mit "wahr" beantwortet wird. Geschieht dies, wird das Programm mit der nach dem Schleifenwendepunkt stehenden Programmzeile fortgesetzt.

```
z.B.:  
  
Enter:  
Repeat  
    A%=Random(100)  
    Print A%
```

```
Mouse X,Y,Button%
Exit If Button%=1
Until A%=50
Print "Schleife verlassen"
If A%=50
    Print "Endwert ";A%;" erreicht !"
Else
    Print "A% <> 50"
Endif
Pause 40
Goto Enter
```

Drücken Sie beim Lauf dieses Programmes die linke Maustaste, wird die Schleife unabhängig von der in "Until" gestellten Bedingung verlassen.

TIMER

Der Atari ST verfügt über einen Counter (Zähler), der im Abstand einer 200stel Sekunde inkrementiert wird. Dieser Zähler beginnt zum Zeitpunkt des Einschaltens des Computers bei Null und erhöht sich also in jeder Sekunde um den Wert 200. Dabei ist es unerheblich, ob zwischenzeitlich irgendwelche Anwendungen ausgeführt werden. Der Zähler orientiert sich an dem konstant bleibenden Takt des Prozessors. D.h. also, daß Sie anhand dieses Zählers feststellen können, wieviel Zeit seit dem Systemstart vergangen ist. Außerdem läßt sich dieser "Timer" hervorragend dazu verwenden, von einer bestimmten Zeitdauer abhängige Arbeiten ausführen zu lassen oder die Zeitdauer bestimmter Prozesse zu messen (Benchmark-Tests).

z.B.:

```
Time=Timer
Do
    X%=(Timer-Time)/20
    T_time$=Str$(X%)
    T_time$=String$(6-Len(T_time$),"0")+T_time$
    Print At(32,11);T_time$
```

```
Exit If Inkey$<>" or Mousek
Loop
Print Timer/200/60
U=Inp(2)
Edit
```

Dieses kleine Programm gibt in 1/10 Sekunden die abgelaufene Zeit seit Programmstart an und zeigt Ihnen nach Betätigung einer beliebigen Taste die seit Systemstart verstrichene Zeit in Minuten.

TIMES / DATES

Es gibt im GfA-BASIC noch zwei weitere Zeitfunktionen. Wie Sie sicher wissen, kann im Kontrollfeld des Desktops das aktuelle Datum und die aktuelle Tageszeit eingegeben werden. Diese beiden Angaben werden systemintern ständig auf den neuesten Stand gebracht. Haben Sie selber keine Veränderungen an diesen Einstellungen vorgenommen, erscheint hier immer das jeweilige TOS-Versionsdatum und die damit verbundene Tageszeit.

Sie können nun diese beiden Informationen jederzeit aus einem BASIC-Programm abrufen.

Mit "Time\$" wird im HH:MM:SS-Format die aktuelle Uhrzeit ausgegeben.

Mit "Date\$" erhalten Sie das aktuelle Datum im Format DD.MM.YYYY.

Die Sekunden der Zeitangabe werden übrigens nur in Zweierschritten erhöht.

SETTIME

Was nützt Ihnen das schönste Datum und die schönste Uhrzeit, wenn Sie nicht auch vom Programm aus darauf zugreifen und diese verändern können. Dazu gibt es den Befehl " Settime

"HH:MM:SS","DD.MM.YYYY" ". Dieser Befehl erwartet als Parameter zuerst die Tageszeit im genannten Format und, ebenfalls im entsprechenden Format, das Datum. Getrennt werden diese beiden Angaben durch ein Komma. Bei der Angabe der Uhrzeit können die Sekunden vernachlässigt werden. Ebenso verhält es sich mit den ersten beiden Ziffern der Jahreszahl, sofern das Datum zwischen 1980 und 2079 liegt. Werden die Parameter nicht ordnungsgemäß übergeben, werden vom Interpreter keine Veränderungen durchgeführt. Beim Datumstring können sogar die Anführungsstriche entfallen, da diese vom Interpreter vollständig gesetzt werden.

z.B.:

```
Settime "15:37","15.08.86" => verändert beide Einträge
Settime "",4.6.83           => verändert nur das Datum
Settime "17:45:22",         => verändert nur die Uhrzeit
```

Zu den letzten drei Funktionen hier nun eine kleine Routine, die Sie evtl., nach Bedarf abgewandelt natürlich, in eigenen Programmen verwenden können.

```
@T_imeset
Cls
Deffill ,2,4
Pbox 200,100,440,300
Deffill ,0,0
Pbox 220,120,420,280
Print At(36,12);Date$
Print At(36,13);D.ay$
Repeat
  Print At(36,14);Time$
Until Len(Inkey$)
Edit
Procedure T_imeset
  Print At(7,11),"UHRZEIT, DATUM UND WOCHENTAG EINGEBEN"
  Print At(7,12)," ( z.B.: 15:36:22,18.7.86,FREITAG )"
  Print At(7,13)," ( 3 * >RETURN< = DEFAULT )"
  Print At(7,14)," ";
  Input T.ime$,D.ate$,D.ay$
```



```
If D.day$=""  
    D.day$="A N Y D A Y"  
Endif  
Settime T.time$,D.ate$  
Return
```

LEN

"Len(Var\$)" ist eine Textfunktion, die Ihnen die Länge der in Klammern angegebenen Stringvariablen zurückgibt. Dieser Wert kann direkt weiterverwendet werden (Print Len(Var\$)) oder einer numerischen Variablen (sinnvollerweise einer Integervariablen, da "Len" kein gebrochener Wert sein wird) übergeben werden (A%=Len(Var\$)). Eine Variante der Len-Verwendung finden Sie auch im Beispielprogramm unter "TIMER".

INKEY\$

Ein BASIC ohne Inkey\$ (oder vergleichbarer Funktion)? Das soll es schon mal gegeben haben. Wer vorher mit dem ST-BASIC gearbeitet hat, wird wissen, wovon die Rede ist. Dieses Problem haben wir hier nicht. Mit Inkey\$ können nicht nur die "normalen" Tastenbelegungen erfragt werden, sondern auch alle Sondertasten (z.B. F1-F10/Cursorblock). Zusätzlich werden fast alle Kombinationen von Control-, Shift- oder Alternate-Tasten mit anderen Tasten registriert. So lassen sich die Funktionstasten in Kombination mit der Shift-Taste doppelt belegen. Die Funktion schaltet intern auf einen anderen Modus um, sobald Sondertasten bzw. ihnen entsprechende Tastenkombinationen gedrückt werden. Im Normalfall wird ein Ein-Zeichen-String zurückgegeben, der das der gedrückten Taste entsprechende ASCII-Zeichen beinhaltet. Bei Sonder-Tasten wird dagegen ein Zwei-Byte-String zurückgegeben, der im ersten Byte eine Null enthält und im zweiten Byte dann einen Code, der zwar ebenfalls ASCII-Zeichen produziert, aber eigentlich nichts mehr mit diesen zu tun hat. Dieser Code leitet sich aus einer systeminternen Tasten-Identifikations-Matrix ab, die Scan-Code genannt wird. Damit lassen sich dann sämtliche Tasten, die auf der

Tastatur zu finden sind und sämtliche Tastenkombinationen gezielt identifizieren. Sie können z.B. auch, genau wie auf dem Desktop ohne Maus den Mauszeiger in Achtpixel-Schritten vertikal und horizontal bewegen, indem Sie bei einer Inkey\$-Abfrage die Tastenkombinationen der Alternate-Taste mit den vier Cursor-Pfeiltasten wählen. Sie werden sehen, daß sich der Mauszeiger entsprechend der Cursortasten über den Bildschirm bewegt. Hier nun ein kleines Programm, mit dem Sie die Tastenbelegungen erforschen können:

```

Do
  Repeat
    Key$=Inkey$
  Until Key$>""
  If Len(Key$)=1
    Print Key$""Asc(Key$)
  Else
    Print "Sondercode : ";Asc(Right$(Key$))
  Endif
Loop

```

Da gibt es allerdings ein Problem. Der Tastaturpuffer hat die, meist unangenehme Eigenart, sich die Tasten, die gedrückt wurden, zu merken. Und zwar alle! Das führt dazu, daß wenn man ganz kontrolliert nur einen Tastendruck zulassen will, über Inkey\$ alle Zeichen ausgegeben werden, die im Tastaturspeicher evtl. durch einen Dauertastendruck gespeichert wurden. Dieses Problem läßt sich dadurch beseitigen, indem man nach der Inkey\$-Abfrage eine kleine Schleife einbaut, die den Tastaturpuffer leert.

Diese Schleife sieht so aus:

```

Repeat
Until Inkey$=""

```

Ein weiterer Trick aus der GfA-Kiste ist eine Kombination von "Len" und "Inkey\$". Jeder, der sich etwas intensiver mit der BASIC-Programmierung auseinandergesetzt hat, kennt die folgende Konstruktion einer Tastatur-Warteschleife:

```
100 A$=Inkey$:If A$="" Then Goto 100
```

Diese Konstruktion wird in GfA-BASIC ersetzt durch:

```
Repeat  
Until Len(Inkey$)
```

oder

```
Repeat  
Until Inkey$<>""
```

Sie hat die gleiche Aufgabe, nämlich darauf zu warten, daß irgendeine Taste auf dem Keyboard gedrückt wird. Um eine bestimmte Taste zu erfragen, gibt es in anderen BASIC-Dialekten die altbekannte Möglichkeit:

```
100 A$=Inkey$:If A$<>"b" Then Goto 100
```

Das gleiche in GfA-BASIC:

```
Repeat  
  A$=Inkey$  
Until A$="b"
```

Damit kann also kontrolliert werden, ob eine bestimmte Taste (hier: b) gedrückt wurde. Wird eine andere Taste gedrückt, bleibt das Programm in der Warteschleife.

MOUSE X.var%,Y.var%,K.var%

Mit dieser Mausfunktion können die zum Zeitpunkt des Auftretens dieses Befehls aktuellen Mauszustände an frei wählbare numerische Variablen übergeben werden. X.var% beinhaltet die aktuelle X-Position des Mauszeigers, Y.var% die aktuelle Y-Position und K.var% den aktuellen Mausknopf-Status. Der Mausknopf-Status kann vier verschiedene Zustände annehmen:

```

0 = keine Maustaste wurde gedrückt
1 = die linke Maustaste wurde gedrückt
2 = die rechte Maustaste wurde gedrückt
3 = beide Maustasten wurden gedrückt

```

z.B.:

```

Repeat
  Mouse X%,Y%,K%
  Print at(37,12);X%'Y%'K%'
Until X%=0 And Y%=0

```

MOUSEX

Dieses ist die Solo-Funktion für die Übergabe der aktuellen X-Position des Mauszeigers (X%=MouseX).

MOUSEY

Das gleiche für die aktuelle Y-Position des Mauszeigers (Y%=MouseY).

MOUSEK

Und das gleiche für den aktuellen Mausknopf-Status (K%=MouseK). Dieser Funktion kommt allerdings noch eine andere Bedeutung zu. Genau wie bei der Sonderbehandlung des Inkey\$ in Kombination mit Len (z.B. Exit If Len(Inkey\$)), die ja ausschließlich darauf reagiert, daß der Inhalt des Inkey\$-Speichers größer als Null ist, gibt es hier die Möglichkeit, mit MouseK (z.B. Exit If MouseK) ausschließlich darauf zu reagieren, daß eine der beiden Maustasten bzw. beide gedrückt wurden. Wird also keine Maustaste gedrückt, bliebe das Programm in dem nachfolgenden Beispiel in der Schleife.

```

Repeat
Until MouseK

```

```
Ypos%=0
```

'Um sicherzustellen, daß keine ungewünschten Menüfunktionen ausgelöst werden, muß der Index auf Null gestellt werden.

```
If Y>2 And Y<38
```

```
    Ypos%=1
```

```
Endif
```

'Falls sich der Mauszeiger bei Mausknopfbetätigung zwischen der Pixelzeile 2 und der Pixelzeile 38 (obere Menüleiste) befindet, setze den Vertikal-Index auf 1.

```
If Ypos%=1
```

```
    Xpos%=Int(X/80)+1
```

'Anschließend berechne den Horizontal-Index des Menüfeldes der oberen Menüleiste, auf der sich der Mauszeiger befindet.

INT

"Int" hat die Funktion, den in Klammern angegebenen Wert zu der nächstkleineren Ganzzahl zu verkürzen. D.h., daß alle Nachkommastellen integriert werden.

z.B.:

```
Print Int(2.521)
```

Ausgabe : 2

```
Print Int(-13.759)
```

Ausgabe : -14

Sollen mathematisch exakte Rundungen vorgenommen werden, müssen Sie dem Konstrukt, das sie zuweisen wollen den Wert 0.5 hinzufügen. Das bedeutet, daß die entstehende Integerzahl bei

einem gebrochenen Anteil von < 0.5 abgerundet und bei einem gebrochenen Anteil von > 0.5 aufgerundet wird.

TRUNC

Die Funktion "Trunc" ist im positiven Wertebereich identisch mit "Int", da sie die Nachkommastellen abschneidet und nur den ganzzahligen Anteil des in Klammern übergebenen Wertes zurückgibt.

Im negativen Wertebereich verhält sich "Trunc" jedoch anders als "Int". Während "Int" die Nachkommastellen in die nächstkleinere Ganzzahl integriert (s. Beispiel 2 unter "INT"), werden bei "Trunc" die Nachkommastellen ignoriert.

z.B.:

```
Print Trunc(2.521)
```

Ausgabe : 2

```
Print Trunc(-13.759)
```

Ausgabe : -13

FIX

"Fix" ist in absolut jeder Beziehung völlig identisch mit "Trunc". Erklärung s. dort.

FRAC

Dieses ist die Umkehrfunktion zu "Fix", bzw. "Trunc". Sie liefert nur den Nachkommaanteil des in Klammern übergebenen Wertes zurück. Diesen Teil nennt man auch den "gebrochenen" Anteil (fraction; engl.: Bruchteil). Der ganzzahlige Teil des Wertes wird ignoriert.

```
If Xpos%=1
    @Schluss
Endif
```

```
'Befindet er sich auf "Quit", beende das Programm in Procedur
'Schluss
```

```
If Xpos%=2
    Pause 10
    @Set.inits
Endif
```

```
'Befindet er sich auf "Inits", gehe zum entsprechenden Untermenü.
'Da in Set.inits das Menü bei Mausbetätigung wieder geschlossen
'wird, muß hier eine kleine Pause eingelegt werden, um sicherzu-
'gehen, daß die Maustaste bei Erreichen von Set.inits wieder losge-
'lassen wurde.
```

```
If Xpos%=3
    @P.utflip
    @Starter
    @P.utflip
Endif
```

```
'Wurde "Edits" angeklickt, aktualisiere die Screen, gehe zum Editor,
'überlagere diesen nach Abschluß wieder mit der aktuellen Screen.
```

```
If Xpos%=4
    @Cinema
    Goto M.enuout
Endif
```

```
'Wurde "Kino" gewählt, gehe zum Cinema-Unterprogramm. Hier muß ein
'Goto zum Menü-Schleifenende eingesetzt werden, da in Cinema eben-
'falls mit dem Variablennamen Xpos% gearbeitet wird und dessen Wert
'einen ungewünschten Menüpunkt aufrufen könnten.
```

```
If Xpos%=5
  Pause 10
  @Set.specials
Endif
```

'Wenn "Spezis" angeklickt wurde, lege eine kleine Pause ein (wie bei
'Menüpunkt "Inits" und gehe dann zum entsprechenden Untermenü.

PAUSE

Mit diesem Befehl können Sie an beliebigen Stellen im Programm genau definierbare Zeitpausen bestimmen. Das Problem der 16-Bit-Computer in dieser Beziehung ist nicht mehr, daß sie so langsam sind und dadurch zwangsläufig Pausen schaffen (wie die 8-Bitter zum Beispiel), sondern daß sie so schnell sind, daß man ihnen manchmal eine kleine Verschnaufpause aufzwingen muß, um bestimmte Programmläufe überhaupt noch kontrollieren zu können. Dem Befehl "Pause" wird ein maximaler Integerwert von 2147483647 übergeben. Es sind zwar auch Minuswerte möglich, aber ob dadurch auch Minuspausen möglich sind, ist sehr zu bezweifeln. Der Interpreter ignoriert diese Werte ganz einfach. Es wird eine Nullpause ausgeführt.

Schneller wird der Computer dadurch jedenfalls nicht. Versuchen Sie übrigens doch mal, den Maximalwert einzugeben. Wenn Sie nach Start dieser Pause nicht das Programm durch Reset oder Control/Shift/Alternate unterbrechen, würde sich der Computer nach sagenhaften 15 Monaten wieder bei Ihnen melden. Aber den Gefallen tun wir ihm nicht.

```
If Xpos%=6
  @F.lipservice
Endif
```

'Hauptmenüpunkt "Flips". Gehe zu der Routine, die das Hin- und Her-
'schalten der beiden Screens verwaltet.
,

```
If Xpos%=7
  @Disk.ops
```


Endif

'Hauptmenüpunkt "Disky". Gehe zum Untermenü für Disketten-
'Operationen.

If Xpos%=8

'Hauptmenüpunkt "Print" wurde gewählt.

Alert 2,"Welche SCREEN soll gedruckt|werden?|(Evtl. Druckeranpassu
ng mit|Desktop-Accessory vornehmen)",3,"SCREEN 1|SCREEN 2|ABBRUCH",Dddd

ALERT

Diese GEM-Funktion erlaubt es, mit dem Anwender unkompliziert zu kommunizieren. Prinzipiell lassen sich damit sogar umfangreiche Menüs verwalten. Da "Alert" aber nur maximal drei Auswahlmöglichkeiten zur Verfügung stellt, ist er eher für kleine Abfragen geeignet. Außerdem kann man ihn zur Darstellung einer Hinweis- oder Alarmbox (alert;engl.:wachsam, alarmbereit) verwenden.

Da die Parameter dieses Befehls recht zahlreich sind, soll hier die Syntax und die einzelnen Auswirkungen der Parameter näher erklärt werden.

Syntax : Alert,Icon,Boxtext\$,Default-Button,Buttontext\$,Backvar%

Der Befehl "Alert" besteht also aus fünf Komponenten:

1) icon : Es können drei verschiedene Bildsymbole dargestellt werden:

- 0 = kein Symbol
- 1 = Ausrufungszeichen
- 2 = Fragezeichen
- 3 = STOP-Schild

2) **boxtext** : Hier wird der eigentliche Text (Mitteilung/Frage) an die Funktion übergeben. Das Zeichen "|" gilt darin als Trennzeichen zwischen den einzelnen Zeilen. Es können insgesamt 4 Zeilen zu maximal je 30 Zeichen dargestellt werden. Der Text kann direkt in den Befehl geschrieben, oder auch als Stringvariable übergeben werden.

3) **button** : Es wird die Nummer (1,2,3) des Buttons übergeben, welcher außer durch Mausklick auch durch die 'RETURN'-Taste (default) bestätigt werden kann. Dieser Button wird in der Box stark umrandet gezeichnet. Eine 0 bewirkt, daß kein Default-Button vorhanden ist. Der, oder einer der Buttons muß dann durch Mausklick bestätigt werden.

4) **buttontext**: Hiedurch erfolgt die Beschriftung der Button. Jeder Button kann mit jeweils maximal 8 Zeichen beschriftet werden. Auch hier gilt das "|" -Zeichen als Trennstrich zwischen den einzelnen Button.

5) **backvar** : An letzter Stelle wird eine frei benennbare numerische Variable angegeben, in welcher die Nummer des bestätigten Button (1,2,3) von der Funktion abgelegt wird.

z.B.

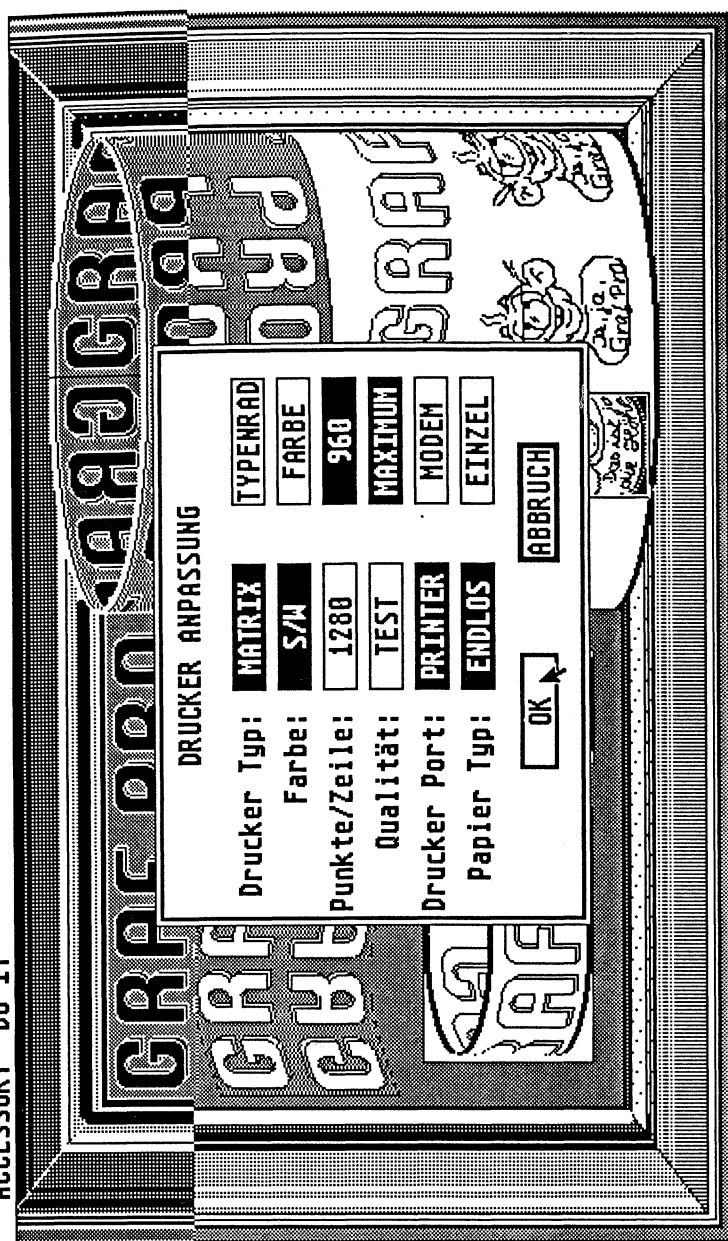
```
B_oxtext$="Klicken Sie bitte eines|der unteren Kästchen an!"
B_oxtext$=B_oxtext$+"|Ich sage Ihnen dann, welches|es war!"
Buttontext$="XX 1 XX|XX 2 XX|XX 3 XX"
Alert 1,B_oxtext$,0,Buttontext$,Backvar%
B_oxtext$="Es war :      | |das Kästchen "+Str$(Backvar%)
Buttontext$=" OKAY | OKAY | OKAY "
Alert 1,B_oxtext$,2,Buttontext$,Backvar%
Edit
```

Diese Form der Alert-Box ist aus Gründen der Sicherheit nicht besonders variabel. Vom Interpreter werden sämtliche Eingaben auf ihre Zulässigkeit überprüft. Werden z.B. mehr als 30 Zeichen Boxtext angegeben, werden sie vom Interpreter nach dem 30sten Zeichen abgeschnitten. Wer die eigentliche GEM-Alertbox kennt, der wird diese Sicherheitsmaßnahmen verstehen. Trotzdem finden Sie im Listing eine Routine "Procedure Formalert", die frei definierbar ist. Allerdings wird von Systemseite aus keine Rücksicht auf falsche Eingaben oder zu lange

Texte genommen. Ist der Text bei dieser "echten" GEM-Alert-Box zu lang, macht das System schlicht und ergreifend eine Bauchlandung, vor der Sie auch der Interpreter mit seinem Absturzverhinderer kaum noch retten kann.

Von Digital Research vorgegeben ist hier eine Zeilenlänge von 40 Zeichen und eine Buttontextlänge von 10 Zeichen je Button. Wie eine solche Alert-Box trotzdem aussehen kann, sehen Sie, wenn Sie im Grafikprogramm die "Help"-Taste drücken.

ACCESSORY DO IT



```
If Ddddd<3
```

```
Alert 2,"Accessories aufrufen ?      ",1," OKAY | NEIN ",B.alert  
%
```

```
'Wird diese Alertbox mit OKAY beantwortet, gehe zur Routine zum  
'Aufruf der Desktop-Accessories
```

```
If B.alert%=1
```

```
    Pause 20
```

```
    @P.utflip
```

```
    @Accessory
```

```
Endif
```

```
If Q_uitter%(0)=1
```

```
'Wenn bei der "Accessory"-Bedienung "N O W" angeklickt wurde,  
'dann drucke die SCREEN.
```

```
If Ddddd=1
```

```
    Flipp.flag%=0
```

```
Endif
```

```
'Wurde "Screen 1" angeklickt, setze das Verwaltungsflag für  
'Screenflipping auf den entsprechenden Wert 0.
```

```
If Ddddd=2
```

```
    Flipp.flag%=1
```

```
Endif
```

```
'Das gleiche für "Screen 2".
```

```
@P.utflip
```

```
'bringe die gewählte Screen auf den Bildschirm
```

```
@Hidemouse
```

```
'schalte Maus ab, damit sie nicht mitgedruckt wird
```

```
Hardcopy
```

```
'drucke das Bild
```

HARDCOPY

Durch "Hardcopy" wird dieselbe Systemfunktion wie durch das gleichzeitige Drücken von Alternate/Help ausgelöst. Der aktuelle Bildschirminhalt wird als Screen-Dump auf den Drucker übertragen. Ist der Drucker nicht "On Line", nicht angeschaltet oder überhaupt nicht angeschlossen, wird das Programm nach ca. 1/2 Minute fortgesetzt.

```
@Showmouse
```

```
'Fertig. Schalte Maus wieder an
```

```
Endif
```

```
Endif
```

```
Endif
```

```
Endif
```

```
M.enuout:
```

```
@Screenset
```

```
If Flip.done%=1 And Mc.done%=0 And Areas%=1
```

```
'Wenn MOVE oder COPY an ist (Areas%=1), das Flipping-Aktivflag  
'gesetzt ist (also Flips angeklickt => Flip.done%=1), aber der  
'gewählte Block noch nicht fixiert wurde, ist diese Bedingung wahr.
```

```
If M.ove%=1
```

```
'wenn MOVE an ist..
```

```
Swap M.ove%,C.opy%
```

```
'.. vertausche das Move- mit dem Copyflag, damit auf der neuen  
'Screen nicht der gewählte Ausschnitt gelöscht wird. Ab hier wird  
'im COPY-Modus weitergearbeitet.
```

SWAP

Hiermit kann man Variableninhalte gleicher Variablentypen miteinander vertauschen. Es sind sowohl numerische (aller Typen) als auch alphanumerische (Strings) Tauschoperationen möglich. Bei Austausch von Feldinhalten werden auch gleichzeitig die Dimensionierungen vertauscht.

z.B.:

```
Dim A%(3),A$(1),Aa(4),Bb(6)
Arrayfill Aa(),16
Arrayfill Bb(),11.7
A%=10
B%=32
A!=-1
B!=0
A=37.66
B=123.5
A%(2)=144
A%(3)=316
A$="ABC"
B$="DEF"
A$(1)="GHI"
A$(0)="JKL"
Swap Aa(),Bb()
Swap A%,B%
Swap A!,B!
Swap A,B
Swap A%(2),A%(3)
Swap A$,B$
Swap A$(1),A$(0)
Print Aa(3),Bb(0)
Print A%,B%
Print A!,B!
Print A,B
Print A%(2),A%(3)
Print A$,B$
Print A$(1)
Print A$(0)
```

S.wapper%=1

'merke Dir, daß die Flags vertauscht wurden

Endif

@Setareas

'Gehe mit Copy On wieder zur Routine für Blockoperationen

Endif

If Const.flag%=1

Const.flag%=0

@Set.const

'Wenn "Fadenkreuz Fix" angeklickt wurde, setze das Setting-Flag

'dafür wieder auf Null und gehe zu der Routine, mit der die Schnitt-

'punktcoordinate gesetzt wird

Endif

If Clipp.set%=1

Clipp.set%=0

@Set.clipping

'Wenn "Clipping setzen" angeklickt wurde, setze das Setting-Flag

'dafür wieder auf Null und gehe zu der Routine, mit der die

'Clipping-Box gesetzt wird

Endif

@Getfkeys

'gehe zur Hauptabfrage-Routine für Maus und Tastatur

Return

Procedure Screenset

'Hier nun eine Trickroutine. Wenn Sie das Programm laufen lassen, wird

'im unregelmäßigen Wechsel mit einer Full-Screen nach jedem Hauptmenü-

'Aufruf das aktuelle Bild aus vier Teilen wieder zu einem ganzen Bild
'zusammengesetzt. Dies ist die Routine dazu.

CLS

'Erstmal wird der Bildschirm gelöscht, damit der Gag auch seine volle
'Wirkung erzielt.

CLS

Zu diesem Befehl ist nicht viel zu sagen. Er kann überall im Programm eingesetzt werden. Seine Funktion ist das Löschen des Bildschirms (CLearScreen) und das Setzen des TOS-Cursors in die linke obere Bildschirmecke (Home). Er kann auch im Direktmodus verwendet werden.

Zufall%=Random(2)

'erzeugt eine Zufallszahl zwischen -1 und 2.

RANDOM

"Random" produziert eine Zufallszahl zwischen -1 (also minimal 0) und dem in Klammern angegebenen Integerwert, wenn der Wert positiv ist und zwischen +1 (max. 0) und einem evtl. angegebenen Negativwert. Die gelieferte Zahl ist in jedem Fall ein ganzzahliger Wert.

RND

Mit "Rnd" lassen sich Zufallswerte im Bereich von 0 bis 0.99999999999 (11-stellig) erzeugen.

z.B.:

Print Int(Rnd*10)+3

gibt eine Ganzzahl im Bereich von 3 bis 10 aus

Do

Print Rnd'''Int(Rnd*1000)

Exit If Len(Inkey\$)

Loop

If Flip.done%=1

Zufall%=1

'Wurde das "FLIPS"-Menüfeld im Hauptmenü angeklickt, soll immer eine
'Full-Screen gesetzt werden.

Endif

For I%=400 To -1 Step -100

'Mit dieser For/Next-Schleife werden die beiden Screen-Settings ge-
'steuert. Setzen Sie hier z.B. als Step-Wert die Zahl -50 ein, benö-
'tigen die Settings die doppelte Zeit, bei der Zahl -20 die fünffache
'Zeit usw. Sie sollten damit etwas experimentieren, da hier das
'Verhalten der mit "Put" aktivierten Ausschnitte beim "gestotterten"
'Setzen sehr unterschiedlich auf den Einzelnen wirken kann. Wird der
'Step-Wert z.B. auf -1 gesetzt, dauert der Vorgang zwar recht lange,
'dafür sieht es aber sehr realistisch aus, wie sich das Bild langsam
'in den Bildschirm "schleicht".

If Zufall%=0

'Es wurde die Zufallszahl 0 "gewürfelt".

Put -200+(200-I%/2),0,F.screen\$(0)

Put 640-(320-I%/2),200,F.screen\$(3)

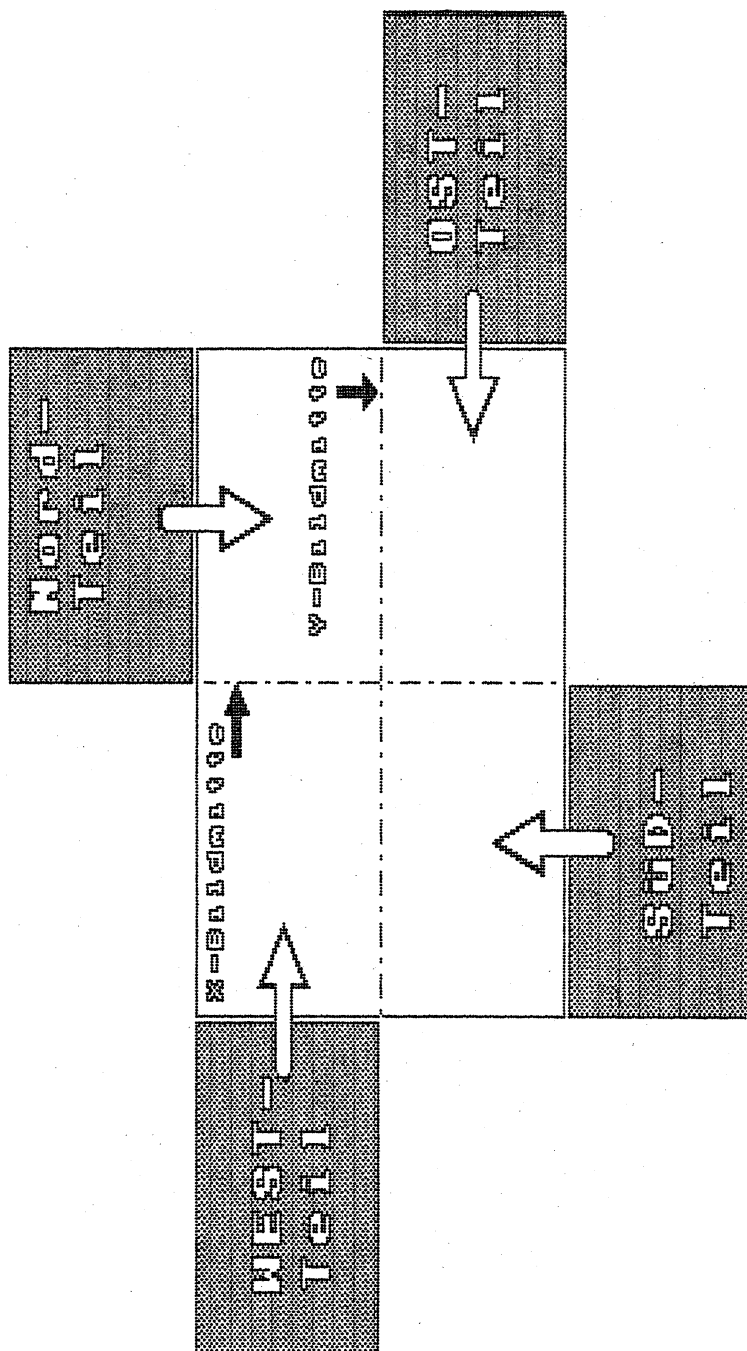
Put 320,-200+(200-I%/2),F.screen\$(1)

Put 0,400-(200-I%/2),F.screen\$(2)

'In diesem Fall soll sich die aktuelle SCREEN aus vier Einzelteilen
'wieder zusammensetzen. Und zwar soll jedes Screen-Viertel aus einer
'anderen Himmelsrichtung "angeflogen" kommen.

'In der Vorbereitungsprozedur "Procedure Predefs" wurden zum Beginn

'des Programms diese vier Teile zu erstmal mit dem "Get"-Befehl
'fixiert.
'Bei jedem Hauptmenü-Aufruf werden diese Screen-Teile, sowie die
'Screen im Ganzen einmal aktualisiert. Das geschieht am Anfang dieser
'"Procedure Screenset" zuerst (@G.etflip). Es soll ja nach der Menü-
'bedienung (außer bei FLIPS) dasselbe Bild wieder erscheinen, daß
'von dem Menü überlagert wurde.
'Hier wird nun der Algorithmus (Verfahrensweise) dazu zum Einsatz ge-
'bracht.
'Dabei müssen Sie sich die beiden Mittelachsen des Bildschirms und
'die vier Außenseiten als Schienen vorstellen, in denen nun diese
'vier kleinen Jalousien geführt werden. Und zwar wird für jedes Teil
'eine andere Himmelsrichtung vorgesehen.
'Eine kleine Grafik soll das näher verdeutlichen.



'Da die Koordinaten, die mit Put übergeben werden, sich immer auf
'die linke obere Ecke des Ausschnittes beziehen, müssen für das
'West- und das Nordteil Minuswerte als Startkoordinaten angegeben
'werden, da die linke obere Ecke dieser Teile um die ganze Breite
'bzw. die ganze Höhe außerhalb des Bildschirms liegen.
'Durch die Schleife wird nun bei dem linken Ausschnitt die X-Koordi-
'nate jeweils um die in Step angegebene Schrittweite erhöht, während
'die Y-Koordinate am oberen Bildschirmrand konstant bleibt. Beim
'oberen Ausschnitt wird die Y-Koordinate erhöht und die X-Koordinate
'in Bildschirmitte bleibt gleich. Für das rechte Teilbild muß die
'X-Koordinate vermindert werden und die Y-Koordinate in Bildmitte
'bleibt unverändert. Bei dem Süd-Teil wird die Y-Koordinate vermin-
'dert und die X-Koordinate auf der vertikalen Mittelachse bleibt
'unverändert.
'So entsteht dieser dynamische Bewegungseindruck.

Endif

If Zufall%=1

'Wenn der Zufall eine 1 zurückgegeben hat...

If Flipp.flag%=0

Put 0,1%,Screen.eins\$,3

Else

Put 0,1%,Screen.zwei\$,3

Endif

'Bringe mit den gleichen Steps die zuletzt aktive Screen, bzw. bei
'Auswahl von FLIPS, die andere Screen (das Flipp.flag wurde dann
'zwischenzeitlich verändert) auf den Bildschirm. Und zwar diesmal
'als Full-Screen von unten nach oben.

Endif

Next I%

'Bild steht wieder

Deftext 1,16,0,25

Get 100,180,540,220,Infoback\$

'Merke dir den Bereich des Bildes, der gleich von der Infozeile über-
'deckt wird

```
Graphmode 1
Deffill ,0,0
Pbox 100,180,540,220
```

'Zeichne als Rahmen für die Schrift eine weiße Box

```
Graphmode 2
```

'Stelle Transparent-Modus ein, da im Replace-Modus evtl. der Rahmen
'von der Textmaske zerstört wird...

```
If Flipp.flag%=0
  Text 150,210,340,"S C R E E N   E I N S"
Else
  Text 150,210,340,"S C R E E N   Z W E I"
Endif
```

'...und zeige die der Screen entsprechenden Infozeile

```
Deffill ,2,6
Fill 110,182
```

""Umhülle" die Schrift mit einem Füllmuster

FILL

Mit diesem Befehl lassen sich beliebige Bildbereiche mit dem eingestellten Füllmuster ausfüllen. Es wird einfach eine Bildschirmkoordinate (X/Y) übergeben, von der aus der Füllvorgang sich ausbreitet, bis er an geschlossene Grenzen stößt. "Fill" ist manchmal nur mit Vorsicht zu genießen. Dies hat zwei Gründe. Der eine ist, daß das ausbreitende Füllmuster die kleinsten (1 Pixel) Schlupfwinkel entdeckt, durch die es sich weiter ausbreiten kann. Das führt einerseits dazu, daß Bereiche, die nicht gefüllt werden sollen durch den Fill-Befehl zunichte gemacht werden und es dann meistens fast unmöglich ist, diesen Bereich

wieder von dem unerwünschten Füllmuster zu säubern. Bei diesem Grafikprogramm ist das allerdings kein großes Problem, da vor dem Füllvorgang entweder das Bild mit der Insert-Taste fixiert werden kann, oder Sie einfach zuerst ein Duplikat davon auf der zweiten Grafik-Screen anlegen. Sollten Sie die erste Möglichkeit wählen und der Füllvorgang verläuft nicht nach Ihren Wünschen, können Sie einfach mit der Undo-Taste das zuletzt mit Insert fixierte Bild wieder aufrufen und es noch einmal versuchen. Die zweite, manchmal recht unangenehme Eigenschaft des Fill-Befehls ist, daß er bei größeren Füllflächen, die schon mit einem anderen, nicht geschlossenen Füllmuster ausgefüllt sind, je nach Größe der Füllfläche sehr, sehr lange braucht, um seine Arbeit zu vollenden. In dieser Zeit tut sich außer dem Füllen nichts. Dieser Befehl wird ausschließlich vom GEM gesteuert und ist dann auch nicht mehr durch irgendwelche Abbruchfunktionen zu unterbrechen. Sollte Ihnen das Malheur passieren, ein offenes Füllmuster mit sich selbst auf großen Flächen zu füllen, kann der Eindruck entstehen, daß Ihnen der Computer Adieu gesagt hat. Lassen Sie ihm Zeit, er wird sich dann schon wieder melden. Also, nicht gleich zum Reset-Knopf greifen. Der Grund für dieses Schneckentempo ist darin zu suchen, daß im Color-Modus der Computer aufgrund der vielen verschiedenen Farben, die dann gleichzeitig darstellbar sind, eine gewaltige Rechenleistung zu vollbringen hat. Im Monochrom-Modus macht er da keine Ausnahme, obwohl ja nur zwei Farben vorhanden sein können.

Put 100,180,Infoback\$

'Die Zeit, die der Fill-Vorgang benötigt, reicht aus, um die Zeile
'lesen zu können. Danach setze den überdeckten Bildausschnitt wieder
'in das Bild ein. Die Grafik-Screen ist jetzt wieder komplett.

Return

Procedure Alterscreen

'Diese Routine wird dann aufgerufen, wenn Sie unter SPEZIS den Border-
'Modus eingeschaltet haben, anschließend bei der Arbeit auf der Screen
'mit dem Mauszeiger den linken Bildschirmrand im mittleren Bereich

'berühren und danach zweimal kurz hintereinander (Doppelklick) eine
'Maustaste drücken.
'Sie können jetzt die jeweils andere Screen mit der Mausbewegung in
'das aktuelle Bild zum Anschauen hineinziehen. Drücken Sie erneut eine
'Taste, verschwindet diese Screen wieder.

@G.etflip

'Zuerst merke Dir die aktuelle Screen

Repeat

'Hier beginnt die Screen-Bewegungsschleife

Alterx=X

Altery=Y

'Merke Dir die letzte Mauskoordinate

Mouse X,Y,Key

'Merke Dir die momentane Mauskoordinate

If Flipp.flag%=0

Put X-639,Y-399,Screen.zwei\$,3

If Alterx<>X Or Altery<>Y

Put 0,0,Screen.eins\$

Endif

Endif

If Flipp.flag%=1

Put X-639,Y-399,Screen.eins\$,3

If Alterx<>X Or Altery<>Y

Put 0,0,Screen.zwei\$

Endif

Endif

'In diesen beiden If-Abfragen wird die letzte Mauskoordinate bei jedem
'Schleifendurchgang mit der aktuellen verglichen. Sobald sie sich von-
'einander unterscheiden, wird die aktuelle Screen komplett neu gesetzt
'und anschließend die Alternativ-Screen an der neuen Position darüber

'gelegt.

'Das hat den Vorteil, daß das Bild nicht flimmert, sondern ruhig lie-

'bleibt, solange die Maus nicht bewegt wird.

Until Mousek

'Beende den Vorgang, sobald eine Maustaste gedrückt wird und gehe zum

'Grafikprogramm zurück.

Pause 10

'Hier muß dem Computer wieder eine kleine Pause verordnet werden, sonst

'benutzt das Programm diesen letzten Mausklick auch gleichzeitig dazu,

'das zuletzt eingestellte Grafik-Element unerwünschterweise zu

'aktivieren.

@P.utflip

'Zum Schluss zeichne die aktuelle Screen nochmal neu

Return

Procedure G.etflip

'Diese Routine merkt sich bei Bedarf die gerade sichtbare

'Grafik-Screen.

If Flipp.flag%=0

Get 0,0,639,399,Screen.eins\$

Endif

If Flipp.flag%=1

Get 0,0,639,399,Screen.zwei\$

Endif

Return

Procedure P.utflip

'Hier wird gegebenenfalls die aktuelle Grafik-Screen neu gezeichnet.

If Flipp.flag%=0

Put 0,0,Screen.eins\$,3

Endif

```
If Flipp.flag%=1
    Put 0,0,Screen.zwei$,3
Endif
Return
Procedure F.lipservice
```

'Wenn FLIPS im Hauptmenü aktiviert wurde, muß kontrolliert werden,
'welche Screen denn gerade aktiv war, um die jeweils andere
'anzuzeigen. Das geschieht hier.

```
Pause 10
Flip.done%=1
```

'Der Umstand, daß "geflippt" wurde, ist an anderer Stelle noch von
'Interesse. Deswegen wird ein entsprechendes Flag auf 1 (aktiv)
'gesetzt.

```
Inc Flippcount%
```

'Um die Nummer der Screen zu ermitteln, wird ein Zähler eingesetzt.
'Dieser Zähler steht bei Programmstart auf Null.

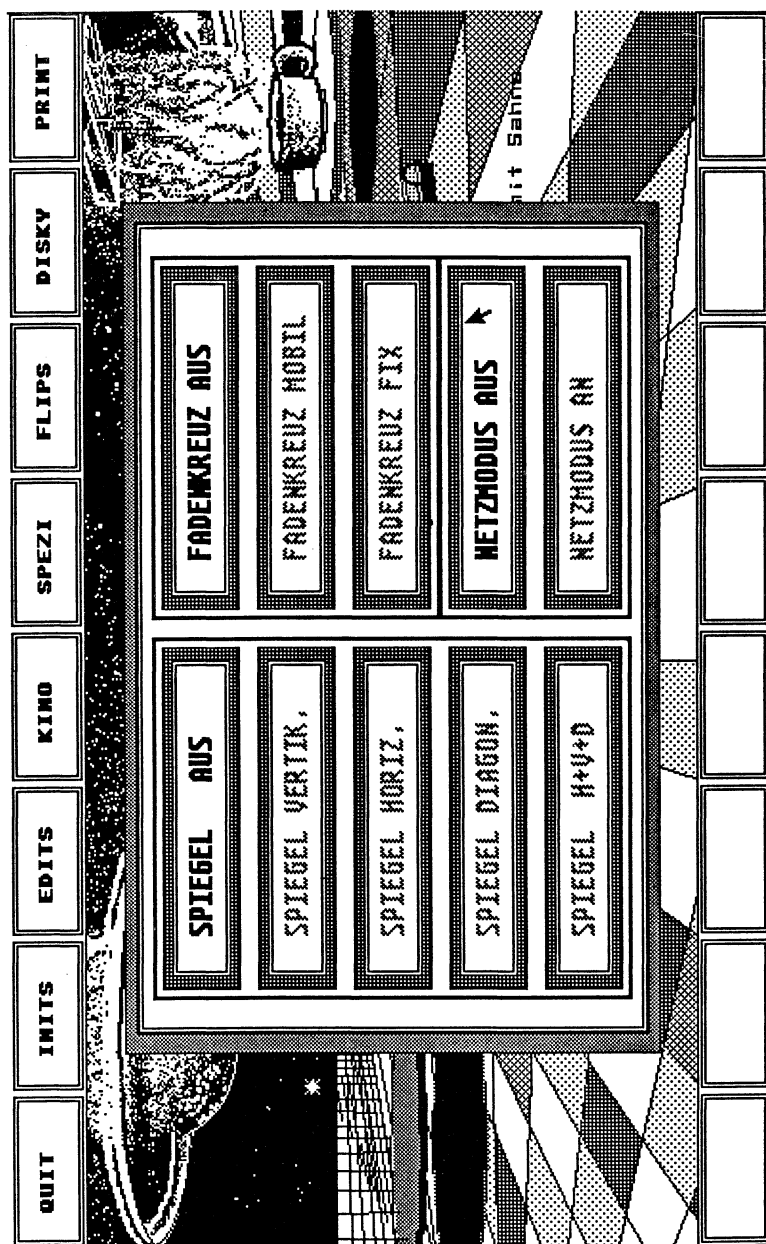
```
If Flippcount%=1
    Flipp.flag%=1
```

'Wenn der Zähler auf 1 steht, setze das Flag für den Screen-Index
'ebenfalls auf 1 (= Screen 2)

```
Endif
If Flippcount%=2
    Flippcount%=0
    Flipp.flag%=0
```

'Steht der Zähler auf 2, setze ihn wieder auf 0, damit er wieder von
'vorne anfangen kann (es gibt ja nur 2 Screens) und setze das Index-
'flag ebenfalls auf 0 (= Screen 1).

```
Endif
Return
```



Procedure Set.inits

'Hier wird das INITS-Untermenü aufgebaut

```
Graphmode 1
Color 1
Defline 1,1,0,0
If Initstartflag%=1
```

'Genau wie beim Aufbau des Hauptmenüs (Procedure Grafmenue), wird hier
'unterschieden, ob das Menü zum ersten Mal gezeichnet wird. Wurde es
'schon einmal gezeichnet, ist das Initstartflag=1. Also wird in diesen
'Block verzweigt.

```
For I=0 To 140 Step 2
  Put 100,200-I,Initmenu$(70-I/2),3
  Put 100,200+I,Initmenu$(70+I/2),3
Next I
```

'Hier werden nun nach einem ähnlichen Schema wie beim Neuzeichnen der
'Screens in einer Schleife einzelne waagerechte Pixelstreifen so an-
'einandergereiht, daß der Eindruck entsteht, das Menü wird aus der
'Bildmitte nach oben und nach unten "ausgerollt". Diese Bildstreifen
'werden weiter unten nach jedem Menüaufruf neu gespeichert, damit
'die gewählten Funktionen auch beim nächsten Aufruf sichtbar sind.

```
Endif
If Initstartflag%=0
```

'Hier wird das INITS-Menü zum erstenmal gezeichnet.

```
Deffill ,2,4
Pbox 100,60,540,340
Deffill ,0,0
Pbox 110,66,530,334
Box 111,67,529,333
Box 113,69,527,331
For Ii%=0 To 4
  Deffill ,2,6
  Pbox 133,80+Ii%*50,310,120+Ii%*50
```

```

Pbox 330,80+Ii%*50,507,120+Ii%*50
Deffill ,0,0
Pbox 140,85+Ii%*50,303,115+Ii%*50
Pbox 337,85+Ii%*50,500,115+Ii%*50
Box 142,87+Ii%*50,301,113+Ii%*50
Box 339,87+Ii%*50,498,113+Ii%*50
Next Ii%
@M.enumtext.1
@M.enumtext.2
@M.enumtext.21
@M.enumtext.22
@M.enumtext.23
Box 128,75,315,325
Box 129,76,314,324
Box 325,75,512,225
Box 326,76,511,224
Box 325,225,512,325
Box 326,226,511,324
Dim Initmenu$(142)
Initstartflag%=1

```

'Dimensioniere danach das Feld für die Bildstreifen
'und setze das Initstartflag auf 1, damit das Menü nicht noch einmal
'gezeichnet wird.

```

Endif
Repeat

```

'Hier beginnt die Schleife zur Abfrage der Mausposition über oder
'außerhalb des Menüs und zur Abfrage des Maustasten-Status.

```

Mouse X,Y,Button%
X.1%=Int((X-142)/180)+1
Y.1%=Int((Y-75)/50)+1

```

'Entsprechend der Lage der Menüfelder wird die Mausposition hier in
'den dem jeweiligen Feld zugehörigen Index umgewandelt. In X-Richtung
'beginnt das erste Menüfeld bei 142. Die Menüfelder sind 180 Pixels
'breit. In Y-Richtung beginnt das erste Feld bei 75 und die Felder
'sind 50 Pixels hoch.

```
If X.1%=1 And Button%=1
```

'Befindet sich der Mauszeiger auf der linken Menüseite (X.1 = 1)
'und wurde die linke Maustaste gedrückt, wird in den nächsten fünf
'If-Abfragen entsprechend des Y-Indexes das Flag "Mirror" für die
'gewählte Spiegelfunktion gesetzt.
'Dabei wird durch abwechselndes Anzeigen des angeklickten Menüfeld-
'textes in verschiedenen Textarten ein dynamischer Effekt eingesetzt.

```
    If Y.1%=1
        Mirror%=0
        @M.enumtext.1
        Text 160,105,"SPIEGEL  AUS"
    Endif
    If Y.1%=2
        Mirror%=1
        @M.enumtext.1
        Text 160,155,"SPIEGEL VERTIK."
    Endif
    If Y.1%=3
        Mirror%=2
        @M.enumtext.1
        Text 160,205,"SPIEGEL HORIZ."
    Endif
    If Y.1%=4
        Mirror%=3
        @M.enumtext.1
        Text 160,255,"SPIEGEL DIAGON."
    Endif
    If Y.1%=5
        Mirror%=4
        @M.enumtext.1
        Text 160,305,"SPIEGEL  H+V+D"
    Endif
Endif
If X.1%=2 And Button%=1
```

'Befindet sich der Mauszeiger auf der linken Menüseite (X.1 = 1)
'und wurde die linke Maustaste gedrückt, werden in den nächsten fünf

'If-Abfragen entsprechend des Y-Indexes die dazugehörigen Flags
'gesetzt. Der Text-Effekt ist der gleiche wie oben.

```
If Y.1%=1
    @M.enutext.21
    @M.enutext.22
    Const.lines%=0
    Showlines%=0
    Text 355,105,"FADENKREUZ AUS"
Endif
If Y.1%=2
    @M.enutext.2
    Showlines%=1
    Text 355,155,"FADENKREUZ MOBIL"
Endif
If Y.1%=3
    @M.enutext.2
    Const.lines%=1
    Const.flag%=1
    Text 355,205,"FADENKREUZ FIX"
Endif
If Y.1%=4
    @M.enutext.23
    Netzmod%=0
    Text 355,255,"NETZMODUS AUS"
Endif
If Y.1%=5
    @M.enutext.23
    Netzmod%=1
    Text 355,305,"NETZMODUS AN"
    @Set.grid
```

'Wurde "NETZMODUS AN" gewählt, gehe zu der Routine zur Bestimmung
'der Maschenweite.

```
Endif
Endif
Until (Button%>0 And (X.1%<1 Or X.1%>2 Or Y.1%<1 Or Y.1%>5)) Or
Len(Inkey$)
```

'Verlasse das Menü nur, wenn a) ein Mausbutton gedrückt wird UND der
'Mauszeiger sich außerhalb der Menüfelder befindet ODER b) eine belie-
'bige Taste auf dem Keyboard betätigt wurde.

```
For I=0 To 140 Step 2
  Get 100,I+60,540,I+62,Initmenu$(I/2)
  Get 100,340-I,540,342-I,Initmenu$(140-I/2)
Next I
```

'Mit dieser Schleife wird nach Menüschluß der aktuelle Menüstand, in
'die oben beschriebenen Bildstreifen zerlegt, als Bild festgehalten.

```
@Set.paras
```

'Restauriere alle vom Anwender gewünschten Grafik-Einstellungen

```
Return
```

'Hier schließen sich die Routinen für die einzelnen Texteffekte des
'Init-Menüs an

```
Procedure M.enutext.2
  Deftext 1,2,0,13
  Text 355,105,"FADENKREUZ AUS"
  Deftext 1,1,0,13
Return
Procedure M.enutext.21
  Deftext ,2,,13
  Text 355,155,"FADENKREUZ MOBIL"
  Deftext 1,1,0,13
Return
Procedure M.enutext.22
  Deftext 1,2,0,13
  Text 355,205,"FADENKREUZ FIX"
  Deftext 1,1,0,13
Return
Procedure M.enutext.23
  Deftext 1,2,0,13
  Text 355,255,"NETZMODUS AUS"
  Text 355,305,"NETZMODUS AN"
```



```
Deftext 1,1,0,13
```

```
Return
```

```
Procedure M.enutext.1
```

```
Deftext 1,2,0,13
```

```
Text 160,105,"SPIEGEL AUS"
```

```
Text 160,155,"SPIEGEL VERTIK."
```

```
Text 160,205,"SPIEGEL HORIZ."
```

```
Text 160,255,"SPIEGEL DIAGON."
```

```
Text 160,305,"SPIEGEL H+V+D"
```

```
Deftext 1,1,0,13
```

```
Return
```

```
Procedure Set.const
```

'Hier wird der Zentrumsunkt für das konstante Fadenkreuz fixiert.

```
@Clipprec(2)
```

```
Deftext 1,1,0,6
```

```
Color 1
```

```
Defline 1,1,0,0
```

```
Graphmode 3
```

```
Repeat
```

```
Mouse H.line%,V.line%,Button%
```

```
Co.txt$="X:"+Str$(H.line%)+ " Y:"+Str$(V.line%)
```

```
Hor.ko%=H.line%-Int(H.line%/6)
```

```
Ver.ko%=V.line%-Int((V.line%-200)/15)
```

```
Text Hor.ko%,Ver.ko%+8,Co.txt$
```

```
Text Hor.ko%,Ver.ko%+8,Co.txt$
```

```
Text Hor.ko%,Ver.ko%-1,"SET FIXPOINT"
```

```
Text Hor.ko%,Ver.ko%-1,"SET FIXPOINT"
```

'Wieder ein kleiner Trick. Die Koordinaten zur Darstellung des Hin-
 'weistextes, der dem Mauszeiger folgt, werden aus den Mauskoordinaten
 'so ermittelt, das dieser Text immer innerhalb des Bildschirms liegt.
 'Wenn nämlich die Mauskoordinate als Textkoordinate verwendet würde,
 'würde der Text immer dann aus dem Bildschirm verschwinden, wenn der
 'Mauszeiger sich dem rechten bzw. oberen Bildrand nähert.
 'Außerdem wird dadurch daß die Linien und der Text zweimal hinter-
 'einander im Graphmode 3 an derselben Position gezeichnet werden, der
 'Eindruck erweckt, daß diese über den Schirm schweben, ohne den
 'Hintergrund zu zerstören (s. auch unter GRAPHMODE).

```

Line 0,V.line%,639,V.line%
Line H.line%,0,H.line%,399
Line 0,V.line%,639,V.line%
Line H.line%,0,H.line%,399

```

LINE

Der "Line"-Befehl wird dazu verwendet, um Linien auf den Bildschirm zu zeichnen. Dazu werden ihm in den ersten beiden Parametern die X/Y-Koordinaten des Startpunktes und in den letzten beiden Parametern die X/YKoordinaten des Endpunktes übergeben. Die Einstellungen der Linienart mit dem Defline-Befehl wirken sich auf diesen Befehl aus.

```

Until Button%>0
Pause 10
@aConst.lines
@aSet.paras
Return
Procedure Set.grid

```

'In dieser Routine wird der Hintergrund für die "Dragbox" zur
'Bestimmung der Netzweite gezeichnet.

```

Get 200,120,440,260,Gridback$
Graphmode 1
Deffill ,0,0
Pbox 200,120,440,260
Box 201,121,439,259
Box 205,125,435,255
Deftext 1,16,0,13
Text 215,150,"Netzweite einstellen : "
Deffill ,2,4
Pbox 220,160,420,200
Deffill ,0,0
Pbox 225,165,415,195
Box 227,167,413,193

```

Box 302,220,340,245

Box 303,221,339,244

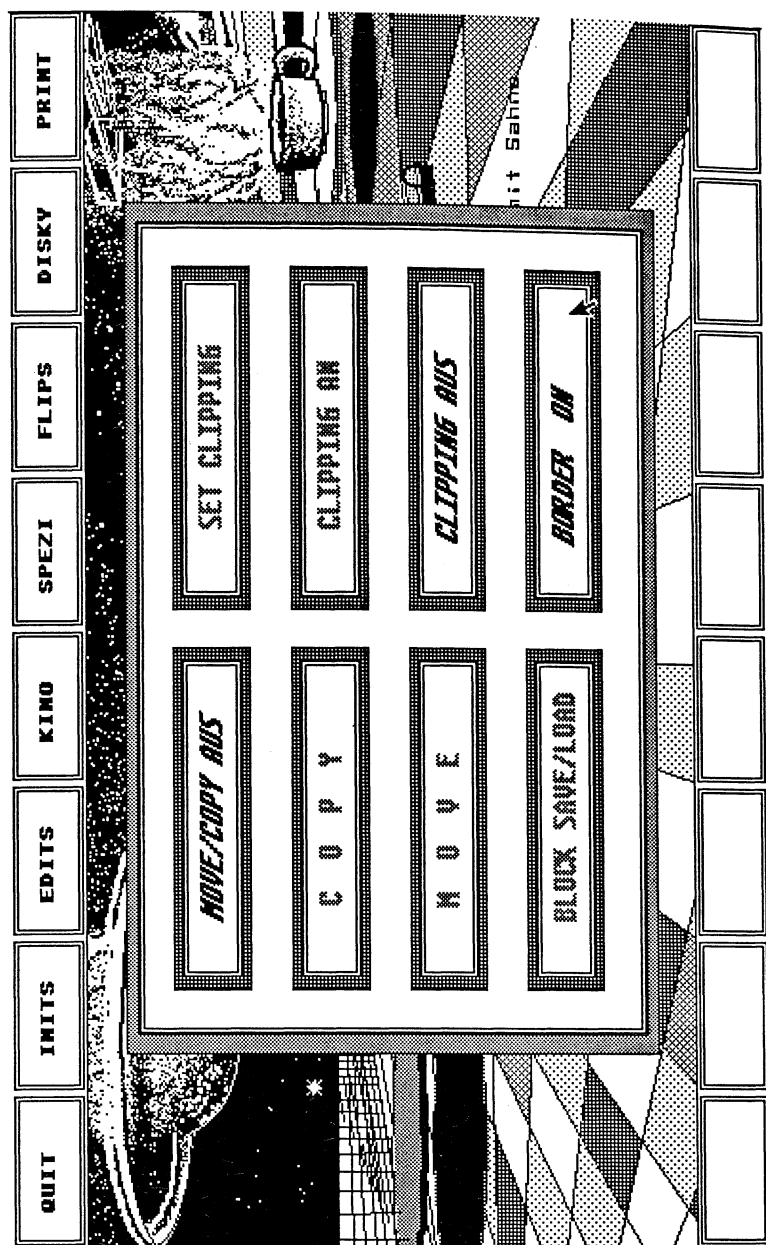
@Dragbox(1,229,169,6,22,229,169,182,22)

'Zur Erläuterung der Dragbox s. in "Procedure Dragbox"

Put 200,120,Gridback\$,3

@Set.paras

Return



Procedure Set.specials

'Zum Verständnis dieser Prozedur s. "Procedure Set.inits". Diese
'beiden Prozeduren sind bis auf wenige Ausnahmen identisch. Nur das
'hier das SPEZI-Untermenü gezeichnet und bedient wird.

```
@Clipprec(2)
Graphmode 1
Color 1
Defline 1,1,0,0
If Spec.startflag%=1
  Put 100,60,Spec.menu$,3
Endif
If Spec.startflag%=0
  Deffill ,2,4
  Pbox 100,60,540,340
  Deffill ,0,0
  Pbox 110,66,530,334
  Box 111,67,529,333
  Box 113,69,527,331
  For Ii%=0 To 3
    Deffill ,2,6
    Pbox 133,85+Ii%*62,310,125+Ii%*62
    Pbox 330,85+Ii%*62,507,125+Ii%*62
    Deffill ,0,0
    Pbox 140,90+Ii%*62,303,120+Ii%*62
    Pbox 337,90+Ii%*62,500,120+Ii%*62
    Box 142,92+Ii%*62,301,118+Ii%*62
    Box 339,92+Ii%*62,498,118+Ii%*62
  Next Ii%
  @M.enutext.3
  @M.enutext.31
  @M.enutext.4
  Spec.startflag%=1
Endif
Repeat
  Mouse X,Y,Button%
  X.1%=Int((X-142)/180)+1
  Y.1%=Int((Y-78)/62)+1
  If X.1%=1 And Button%=1
```

```

If Y.1%=1
  Areas%=0
  C.opy%=0
  M.ove%=0
  @M.enutext.3
  Text 160,110," MOVE/COPY AUS "
Endif
If Y.1%=2
  Areas%=1
  C.opy%=1
  M.ove%=0
  @M.enutext.3
  Text 160,172," C O P Y "
Endif
If Y.1%=3
  Areas%=1
  M.ove%=1
  C.opy%=0
  @M.enutext.3
  Text 160,234," M O V E "
Endif
If Y.1%=4
  @M.enutext.3
  Text 160,296," BLOCK SAVE/LOAD "
  Alert 2,"          B L O C K :          ",0," SAVE | LOAD |
CANCEL",Block.flag%
  If Block.flag%=1
    Areas%=1
    C.opy%=1
    @M.enutext.3
    Alert 3,"Bitte den Mauszeiger auf linke,|obere Ecke des zu
speichernden|Blockes setzen, linke Taste|drücken und Block markieren
!",1," OKAY ",Backalert%
    Sp.out.flag%=1
    Goto Specloop.out
  Endif
  If Block.flag%=2
    @Blockloader
    Block.op%=1

```

```
Areas%=1
C.opy%=1
Bl.load%=1
```

'Falls ein Bildschirmblock geladen wurde, schalte grundsätzlich
'auf den COPY-Modus um. Dieser wird später in "Procedure Areaprocs"
'wieder ausgeschaltet.

```
Endif
If Selected$<>"\" And Selected$<>".BLO" And Selected$>""
```

'Zum Verständnis dieser If-Abfrage s. unter FILESELECT

```
@M.enutext.3
Sp.out.flag%=1
Goto Specloop.out
Endif
@M.enutext.3
If Block.flag%=3
Block.flag%=0
Endif
Endif
Endif
If X.1%=2 And Button%=1
If Y.1%=1
Clipp.set%=1
@M.enutext.31
Text 355,110," SET CLIPPING "
If C.lipp%=1
Text 355,172," CLIPPING AN "
Endif
Endif
If Y.1%=2
C.lipp%=1
Clipp.set%=0
@Clipprec(1)
@M.enutext.31
Text 355,172," CLIPPING AN "
Endif
If Y.1%=3
```

```

    C.lipp%=2
    Clipp.set%=0
    @Clipprec(2)
    @M.enutext.31
    Text 355,234," CLIPPING AUS "
Endif
If Y.1%=4
    Inc B.order
    If B.order=1
        Border%=1
        @M.enutext.4
        Pause 5
        Text 355,296," BORDER ON "
    Endif
    If B.order=2
        B.order=0
        Border%=0
        @M.enutext.4
        Pause 5
        Text 355,296," BORDER OFF "
    Endif
Endif
Endif
Specloop.out:
    Until (Button%>0 And (X.1%<1 Or X.1%>2 Or Y.1%<1 Or Y.1%>4)) Or
Len(Inkey$) Or Sp.out.flag%=1
    Sp.out.flag%=0
    Get 100,60,540,340,Spec.menu$
    @Set.paras
Return
Procedure M.enutext.3
    @Clipprec(2)
    Deftext 1,3,0,13
    Text 160,110," MOVE/COPY AUS "
    Text 160,172," C O P Y "
    Text 160,234," M O V E "
    Text 160,296," BLOCK SAVE/LOAD "
    Deftext 1,5,0,13
Return

```


Procedure M.enutext.31

@Clipprec(2)

Deftext 1,3,0,13

Text 355,110," SET CLIPPING "

Text 355,172," CLIPPING AN "

Text 355,234," CLIPPING AUS "

Deftext 1,5,0,13

Return

Procedure M.enutext.4

Deftext 1,3,0,13

Text 355,296," BORDER "

Deftext 1,5,0,13

Return

Procedure Disk.ops

'Hier beginnt die Unteroutine für Diskettenoperationen

@Clipprec(2)

Get 250,150,390,280,B.ackmenu\$

Deftext 1,1,0,9

Deffill ,0,0

Graphmode 1

Pbox 250,150,390,280

Box 251,151,389,279

Box 255,155,385,275

Deffill ,2,3

Pbox 258,158,382,272

Deffill ,0,0

Pbox 263,161,376,179

Pbox 263,191,376,209

Pbox 263,221,376,239

Pbox 263,251,376,269

Text 266,175,"SAVE SOME"

Text 266,205,"LOAD SOME"

Text 266,235,"KILL SOME"

Text 266,265,"NAME SOME"

Pause 10

Repeat

Mouse X,Y,K

Until Mousek Or Len(Inkey\$)

```

If X>263 And X<376
  If Y>161 And Y<179
    F.ops%=1
    @Sl.some
  Endif
  If Y>191 And Y<209
    F.ops%=2
    @Sl.some
  Endif
  If Y>221 And Y<239
    Alert 2,"Soll tatsächlich eine|Datei gelöscht werden ?","0," OKAY |
NEIN ",Backalert%
    If Backalert%=1
      Fileselect "\*.*", "", Selected$

```

FILESELECT

Dieser Befehl ist einer der wichtigsten Befehle zur Bewältigung von Diskettenoperationen. Die Menge an Funktionen, die dieser so einfach scheinende Befehl in sich vereinigt, kann man nur ermessen, wenn man schon einmal versucht hat, auf die herkömmliche Weise Dateien zu speichern, zu laden, zu löschen oder zu verändern. Es werden drei Parameter übergeben. Der erste ist ein String oder eine Stringvariable, in welcher der Pfad zu der gesuchten Datei angegeben wird (Struktur s. unter EXIST). Im zweiten, der ebenfalls ein String oder eine Stringvariable ist, kann gegebenenfalls ein Dateiname angegeben werden, der bei Aufruf der Box in die Eintragszeile rechts unter "Auswahl" eingesetzt wird. Hier ist zu beachten, daß tatsächlich nur maximal 12 Zeichen (incl. Punkt) verwendet werden (8 für Dateinamen / 1 für Trennpunkt / 3 für Extender). Es kann auch, wie im obigen Fall "" eingesetzt werden oder nur der Extender (".Ext") der Datei. Der dritte Parameter ist eine beliebig zu benennende Stringvariable, die den kompletten Namen der vom Anwender ausgewählten Datei (mit Pfadbezeichnung) aufnehmen soll. Bei Rückkehr zum Programm sind fünf verschiedene Eintragsmöglichkeiten in diese Variable gegeben. Wurde tatsächlich vom Anwender eine Datei entweder durch Doppelklick auf den entsprechenden Namen im Directory-Fen-

ster oder einfachen Klick darauf und anschließendem Klick in die Ok-Box gewählt, steht ihr Name auch anschließend in der Rückgabeveriable. Wurde jedoch ohne Auswahl auf die OK-Box geklickt, gibt es drei Varianten. Entweder Sie hatten vorher einen Dateinamen übergeben, der in "Auswahl" steht und die Ok-Box wurde bedient, dann steht dieser Name auch in der Rückgabeveriable. Oder Sie haben nur einen Extender übergeben, dann steht "\.EXT" (bzw. der von Ihnen gewählte) in der Variablen. Oder aber es wurde keine Auswahl getroffen, ein Name bzw. Extender wurde nicht übergeben oder wurde in der Box vom Anwender gelöscht und die Ok-Box wird angeklickt. Dann erhalten Sie als Rückgabe in der Variablen den "Backslash" (\). Wird dagegen "Abbruch" angeklickt, ist die Variable absolut leer. Der Inhalt der Fileselect-Box kann übrigens während des Aufrufs verändert werden. Sie können in der Box mit dem Mauszeiger in die Zeile unter "Index" klicken. Der Strichcursor begibt sich dann in die zweite Zeile und Sie können nun durch die Tasten "Esc", "Backspace", "Delete" oder die Cursortasten die Zeile oder einzelne Zeichen löschen oder den Cursor an eine bestimmte Stelle bewegen. Haben Sie das getan, können Sie nun die Pfadbezeichnung ändern und sich so z.B. den Inhalt einer anderen Diskette anzeigen lassen. Nach der Änderung brauchen Sie jetzt nur noch den grauen Balken über dem Directory-Fenster anzuklicken und die neue Directory wird angezeigt.

```
If Selected$>"" And Selected$<>"\"
Kill Selected$
```

KILL

Möchten Sie eine Datei auf der Diskette löschen, können Sie das mit diesem Befehl tun. Es wird dabei der Name der zu löschenden Datei (incl. Pfad) angegeben. Eine Prozedur, die die Dateiauswahl dazu erleichtert und die Sie auch im Directmodus ansprechen können, finden Sie unter dem Namen "Procedure Kill" im Listing.

```

    Endif
  Endif
Endif
If Y>251 And Y<269
  Alert 2,"Soll tatsächlich eine|Datei umbenannt werden ?",0," OKAY | N
EIN ",Backalert%
  If Backalert%=1
    Nameagain:
    Fileselect "\*.*", "", Selected$
    If Selected$="" And Selected$<>"\"
      Deffill ,0,0
      Pbox 120,150,520,280
      Box 122,152,518,278
      Box 123,153,517,277
      Print At(18,11);">> DATEI-NAMEN ÄNDERN << ( Nullstring=ABBRUCH )"
      Print At(18,12);"Alter Dateiname:"
      Print At(18,13);Selected$
      Print At(18,14);"=====
      Print At(18,15);"Bitte neuen Dateinamen eingeben:"
      Print At(18,17);"_____
      Print At(18,17);

```

PRINT

Mit "Print" läßt sich fast alles auf den Bildschirm schreiben. Er ist einer der Tausendsassa-Befehle, die man in BASIC kennt. Um seinen Variantenreichtum ausloten zu können, wird es Ihnen nicht erspart bleiben, seinen Einsatz im Programm zu verfolgen.

Verwendungsbeispiel:

```

A%=1
A$="-Demo"
Print At(23,12);Chr$(27);"p";"PRINT"'A$,"String",A;Chr $(27);"q"!!!!
U=Inp(2)
Edit

```

In diesem bunt zusammengesetzten Print-Befehl haben die einzelnen Komponenten folgende Bedeutung:

At(XP,YP):

Mit diesem wahlfreien Zusatz können Sie bestimmen, an welcher Cursor-Position die Ausgabe erfolgen soll. Der TOS-Bildschirm besteht, von der "Home"Position aus gezählt, aus 80 Spalten (1-80) und 25 Zeilen (1-25).

; (Semikolon):

Dieses Zeichen dient dazu, um verschiedene Ausdrücke miteinander zu verbinden. Der nach diesem Zeichen stehende Ausdruck wird dann direkt an den vorhergehenden angeschlossen. Wenn das Semikolon als Schlußzeichen eingesetzt wird, wird auch der erste Ausdruck des nächsten Print-Befehls an den zuletzt ausgegebenen Print-Ausdruck angehängt, da der "Carriage Return" (Wagenrücklauf/neue Zeile) in diesem Fall dadurch unterdrückt wird.

, (Komma):

Wird das Komma eingesetzt, wird der folgendem Ausdruck an den nächsten von 5 Tabulatorpunkten nach dem letzten Ausdruck gesetzt. Diese Punkte haben die X-Positionen 1,17,33,49,65.

' (Apostroph):

Ein zwischen den Ausdrücken verwendetes Apostroph steht als Leerzeichen. Es werden also in der Bildschirmausgabe überall dort Leerzeichen gesetzt, wo sie durch diese Zeichen gekennzeichnet sind.

Chr\$(27);"Z":

Das "Z" steht hier zur Kenntlichmachung dafür, daß ein entsprechender TOS-Befehl eingesetzt werden muß. Das ASCII-Zeichen 27 (Esc) ist eines der TOS-Steuerzeichen, die man beim Print-Befehl einsetzen kann. In diesem Fall wird durch Esc"p" die folgende Schrift invertiert und durch Esc"q" wieder auf Normaldarstellung umgeschaltet.

Ein einzelner Print-Befehl ohne nachgestellte Ausdrücke bewirkt den Ausdruck einer Leerzeile. Wollen Sie innerhalb eines Ausdrucks Sonderzeichen verwenden, können Sie das auf dieselbe Art erreichen, wie es unter "INPUT" beschrieben wurde.

Ein kleiner Laufschrift-Trick:

```
A$="Wer nicht sucht, der nicht findet!"
B$=Space$(55)
Repeat
  A$=Right$(A$,Len(A$)-1)+Left$(A$)
  Rset
  B$=A$
  Print At(1,12);B$
Until Len(Inkey$)
Edit
```

Eine Variante des Print-Befehls ist "Print#". In diesem Fall folgt auf das Nummernzeichen eine Zahl, die den anzusprechenden Datenkanal angibt. Wenn Sie also in eine geöffnete Disketten-datei etwas hineinschreiben möchten, ist das hiermit möglich (z.B. Print#1;"BASIC"). Die Formatierungszeichen können hier eingesetzt werden.

Liste der TOS-"Escape"-Funktionen:

Chr\$(27);"A"	Cursor eine Zeile nach oben
Chr\$(27);"B"	Cursor eine Zeile nach unten
Chr\$(27);"C"	Cursor ein Zeichen nach rechts
Chr\$(27);"D"	Cursor ein Zeichen nach links
Chr\$(27);"E"	Clear/Home
Chr\$(27);"H"	Home
Chr\$(27);"I"	Cursor eine Zeile nach oben+scrollen
Chr\$(27);"J"	Bildschirm ab Cursor löschen
Chr\$(27);"K"	Zeile ab Cursor löschen
Chr\$(27);"L"	Zeile einfügen
Chr\$(27);"M"	ganze Zeile löschen

Chr\$(27);"b";F ¹	bestimmt Cursorfarbe
Chr\$(27);"c";F ¹	bestimmt Hintergrundfarbe
Chr\$(27);"d"	Bildschirm bis Cursor löschen
Chr\$(27);"e"	Cursor an
Chr\$(27);"f"	Cursor aus
Chr\$(27);"j"	Position des Cursors speichern
Chr\$(27);"k"	Cursor zu gespeicherter Position
Chr\$(27);"l"	Zeile löschen
Chr\$(27);"o"	Zeile bis Cursor löschen
Chr\$(27);"p"	Revers ein
Chr\$(27);"q"	Revers aus
Chr\$(27);"v"	Cursor soll sich nach Erreichen des Zeilenendes automatisch an den Anfang der nächsten Zeile setzen
Chr\$(27);"q"	Cursor soll am Zeilenende stehenbleiben

SPC

Ein Befehl, der nur im Zusammenhang mit "Print" verwendet werden kann, ist "Spc(Lz)". Der in Klammern angegebene Ausdruck steht für "Leerzeichen", d.h., daß hier ein Wert zwischen 0 und 255 übergeben werden kann, durch den bestimmt wird, wieviele Leerzeichen (SPaCe) ausgegeben werden sollen.

z.B.:

```
Print "==>";Spc(20);"Ende"
Ausgabe: ==> Ende
```

nicht möglich:

```
A$="==>"+Spc(20)+"Ende"
Es wird ein Syntax Fehler angezeigt.
Für solche Konstrukte eignet sich "Space$(20)".
```

¹ Farbwert

POS

"Pos" liefert Ihnen die aktuelle Cursorspalte. Es muß ein, in Klammern nachgestelltes, numerisches Scheinargument angegeben werden. Diese Zahl kann beliebig gewählt werden und hat keinen weiteren Einfluß auf die Funktion. Da Stringkonstrukte mit mehr als 80 Zeichen ausgegeben werden können und eine TOS-Bildschirmzeile 256 Zeichen aufnehmen kann, kann sich der zurückgegebene Wert im Bereich von 0 - 255 bewegen. Die durch "Pos" ermittelte Spaltenzahl ist nicht identisch mit der bei "Print At(XP,YP)" angegebenen Spaltenzahl. Sie differiert um den Wert -1.

z.B.:

```
Print At(1,1);Pos(0);Spc(20);Pos(0)
```

Ausgabe : 0 21

```
Print Space$(4000);Pos(0)
```

Ausgabe : (4000 Leerzeichen)160

Der Wert 160 im Beispiel errechnet sich aus:

$4000 \text{ Mod } 256 = 160$

TAB

Es kann eine Tabulatorposition bestimmt werden, an der die nachfolgende Printanweisung ausgeführt wird. Diese Position kann im Bereich zwischen 0 und 256 liegen. Größere Werte werden mit Mod 256 auf diesen Bereich zurückgerechnet. Befindet sich der Cursor in einer Zeile hinter der zuletzt angegebenen Tabulatorposition und liegt der Wert einer sich anschließenden "Tab"-Anweisung zwischen 256 und der aktuellen Cursorposition, wird dieser Tabulator in derselben Zeile ausgeführt. Ist ein "Tab"-Wert kleiner als die aktuelle Cursorposition, wird diese Anweisung in der nächsten Zeile ausgeführt. Die "Tab"-Anweisung ist nur in Verbindung mit dem "Print"-Befehl ausführbar. Es ist nicht möglich, sie in einen Stringausdruck einzubinden.

z.B.:

```
Print At(22,1);Tab(25);"Die"
Print Tab(16);"Vier";Tab(35);"Jahreszeiten : "
Print
For J%=1 To 5
Restore
T_ext
For I%=1 To 4
Read A$
Print Tab(16+J%*4);A$;
Next I%
Next J%
T_ext:
Data FRÜHLING,SOMMER,HERBST,WINTER
Edit
```

PRINT USING

Durch diesen Befehl kann ein Ausgabeformat für Werte und Strings bestimmt werden. Ihm folgt als erstes ein String oder eine Stringvariable, in der das gewünschte Format angegeben wird. Als nächstes werden nach einem Komma die Ausdrücke oder Variablen übergeben, die diesem Format entsprechen sollen. Werden mehrere Ausdrücke angegeben, müssen diese durch weitere Kommas getrennt werden. Werden bei numerischen Werten im Format weniger Nachkommastellen angegeben, als vorhanden sind, werden die übrigen Nachkommastellen integriert.

z.B.:

```
A=12.4672
Print Using "###.##",A
      Ausgabe: 12.46
```

```
A$="BASIC"
Print Using "GfA-!asic",A$
Ausgabe: GfA-Basic
```

Als Formatierungszeichen sind vorgesehen:

- # = Platzhalter für eine Ziffer
- . = Position des Dezimalpunktes
- + = Ausgabe auch des positiven Vorzeichens
- = Platzhalter für negatives Vorzeichen
- * = führende Nullen werden durch * ersetzt, sonst wie #
- \$\$ = Voranstellung eines
- \$, = Einfügen eines Kommas (Tausendertrennung)
- ^^^ = Ausgabe im Exponentialformat
- ! = Das erste Zeichen eines Strings wird ausgegeben
- & = Gesamtstring wird ausgegeben
- \..\ = Ausgabe von sovielen Zeichen des Strings, wie Länge
 von \..\ (incl. Backslashes \)
- = Druckt das nachfolgende Zeichen.

Diese Zeichenliste ist wieder aus dem GfA-BASIC-Handbuch entliehen, da verschiedene Testläufe zu diesem Befehl mit der mir vorliegenden BASIC-Version nicht das erwartete Ergebnis lieferten. Wie auch beim "Arrptr" ist anzunehmen, daß die bei Auslieferung des Buches erhältliche Version von diesem Mangel befreit sein wird.

WRITE

Ein Output-Befehl, der in erster Linie zur Datenspeicherung in sequentiellen (sequent;engl.: aufeinanderfolgend) Dateien dient, ist "Write". Dieser Befehl kann aber auch zur Text-, bzw. Datenausgabe auf dem Bildschirm verwendet werden. Er ist dem Print-Befehl sehr ähnlich, hat jedoch einen anderen syntaktischen Aufbau. Außerdem werden hier die Anführungszeichen eines übergebenen Strings (Ausdruck oder Variable) sowie die Kommas, mit denen hier die einzelnen Ausdrücke voneinander

getrennt werden, ebenfalls ausgegeben. Seinen eigentlichen Sinn zeigt dieser Befehl jedoch erst in Verbindung mit "Input #", wenn mit einem "Input #-Befehl mehrere Werte oder Strings gleichzeitig eingelesen werden sollen.

z.B.:

```
Open "0",#1,"Friends"
Restore N_amen
For I%=1 To 4
Read N.ame$,B.eruf$,T.elefon$
Write #1,N.ame$,B.eruf$,T.elefon$
Next I%
Close #1
Clear N_amen:
Data "Elizabeth","Königin      ","London/112233"
Data "Kashogghi","Milliardär  ","Riad/1.000.000.000"
Data "Boris     ","The Winner  ","Leimen/666666"
Data "Yeti      ","Schneemensch","Himalaya/XY-ungelöst"
Open "U",#1,"Friends"
Print "Meine besten Freunde:"
Print For I%=1 To 4
Input #1,N.ame$,B.eruf$,T.elefon$
Write N.ame$,B.eruf$,T.elefon$
Next I%
Close #1
U=Inp(2)
Edit
```

Wie Ihnen vielleicht aufgefallen sein wird, sind die Telefonnummern frei erfunden. Also bitte, nicht anrufen! Die richtigen Telefonnummern behalte ich natürlich für mich. Meine Freunde haben schon genug zu tun.

```
Input """,N.ewname$
```

INPUT

Dieses ist der auf TOS-Ebene noch am meisten verwendete Befehl zur Eingabe von Werten oder Strings an das Programm. Im Anschluß an diesen Befehl kann ein Textausdruck angegeben werden, der vor der ersten Eingabestelle erscheinen soll. Nach einem Komma oder Semikolon wird dann die Werte- oder Textvariable angegeben, die die eingegebenen Daten aufzunehmen hat. Der Unterschied zwischen der Angabe eines Kommas oder eines Semikolons ist der, daß bei einem Komma nach dem Befehl, bzw. nach dem eingefügten Text ein Leerzeichen und ein Fragezeichen erscheint, während beim Semikolon direkt nach dem Befehl oder Text der Cursor erscheint und dort die Eingabe beginnen kann. Sollen mehrer Werte oder Strings über einen Input-Befehl eingegeben werden, können Sie durch Kommas getrennt eine Liste der dafür vorgesehenen Variablen anfügen. Der Befehl ist dann erst abgeschlossen, wenn für jede angegebene Variable die entsprechende Daten eingegeben wurden. Dabei ist es sogar möglich, verschiedene Variablentypen zu verwenden. Sie können somit also mit einem Input-Befehl gleichzeitig Werte und Text erfragen. Jede einzelne Eingabe kann hier entweder durch "Return" abgeschlossen werden oder Sie können die Daten in einer Zeile durch ein Komma voneinander getrennt eingeben. Weisen Sie jedoch dem angegebenen Variablentyp nicht entsprechenden Daten zu (bei numerischen Variablen Text oder umgekehrt), wird ein akustisches Signal ausgegeben (sofern Sie die Lautstärke Ihres Monitors nicht auf 0 gestellt haben) und die Eingabe kann wiederholt werden. Bei Stringeingaben sind Strings mit einer Länge von bis zu 255 Zeichen möglich. Innerhalb der Eingabezeile können Sie die eingegebenen Daten auf verschiedene Weise verändern.

- | | |
|-------------------|--|
| Backspace-Taste | = löscht das links vom Cursor stehende Zeichen |
| Delete-Taste | = löscht das unter dem Cursor stehende Zeichen |
| Pfeil-links-Taste | = bewegt den Cursor um eine Stelle nach links |

- Pfeil-rechts-Taste = bewegt den Cursor um eine Stelle nach rechts (sofern er nicht am Zeilenende steht)
- Pfeil-hoch-Taste = bewegt den Cursor zum Zeilenanfang
- Pfeil-runter-Taste = bewegt den Cursor zum Zeilenende

Wollen Sie bei der Eingabe Sonderzeichen verwenden, gibt es dazu drei Möglichkeiten:

1. Alternate-Taste und andere Taste
2. gleichzeitig Control-Taste und "S" drücken, anschließend andere Taste
3. gleichzeitig Control-Taste und "A" drücken, anschließend den ASCII-Wert des gewünschten Zeichens eingeben Hiermit lassen sich allerdings die ersten 32 ASCII-Zeichen nicht darstellen (Steuerzeichen).

Wie beim Print-Befehl gibt es auch hier die "Kanal"-Variante. Mit Input # und der daran anschließenden Nummer des gewählten Datenkanals können auch Daten aus Diskettendateien bezogen werden.

z.B.

Input #1,A\$,B\$,C\$

INPUT\$

Wollen Sie, daß Strings "verdeckt" eingegeben werden, können Sie das mit diesem Befehl erreichen. Anders als bei allen anderen Input-Arten wird hier die aufnehmende Variable nicht dem Befehl nachgestellt, sondern der String muß einer Variablen zugewiesen werden (A\$=Input\$(10)) oder kann direkt nach Abschluß der Eingabe ausgegeben werden (Print Input\$(10)). Dem Befehl folgt in Klammern eine Zahl, die angibt, wieviele Zeichen maximal eingegeben werden können. Wird diese Zeichenanzahl erreicht, wird das Programm, ohne auf die Betätigung der "Return"-Taste zu warten, fortgesetzt. Der eingegebene Text ist während der Eingabe nicht zu sehen. Auch hier ist

die "Kanal"-Variante möglich (`A$=Input$(10,#1)`). Dieser Befehl eignet sich besonders dazu, um eine bestimmte Datenmenge aus einer Diskettendatei zu lesen. Um zum Beispiel eine ganze Datei in einen String einzulesen (vorausgesetzt, sie ist nicht länger als die maximale Stringlänge = 32767 Byte), können Sie folgendes Mini-Programm verwenden:

```
Open "I",#1,"Dateiname"  
A$=Input$(LOF(#1),#1)  
Close #1  
Print A$  
U=Inp(2)
```

Sie brauchen hierbei die Länge der Datei nicht zu kennen, da sie durch `"LOF(#1)"` ermittelt wird. Sollte der gesamte Text der Datei nicht auf den Bildschirm passen, müssen Sie durch die verschiedenen Stringfunktionen einzelne Blöcke aus dem String herausnehmen und sich anzeigen lassen (z.B. `Print Left$(A$,1000)`). So kann zum Beispiel auch der Source-Code beliebiger Programme eingelesen und modifiziert werden.

LINE INPUT

Der "Line Input"-Befehl ist ein direkter Nachkomme des normalen Input-Befehls. Sie unterscheiden sich nur dadurch, daß "Line Input" ausschließlich für Texteingabe zu verwenden ist, und daß bei Angabe einer Variablenliste die Eingaben alle einzeln mit "Return" abgeschlossen werden müssen. Während beim Input-Befehl durch die Eingabe eines Kommas kenntlich gemacht wird, daß nun die nächste Variable in der Liste bedient wird, wird hier das Komma als Textzeichen innerhalb des eingegebenen Strings anerkannt. Genauso wie beim Input-Befehl kann auch hier ein Textstring übergeben werden, der dann vor dem ersten einzugebenden Zeichen erscheint, z.B.:

```
Line Input "Bitte Text eingeben : ",A$,Bb$,Cc$
```

Durch Verwendung eines Kommas oder Semikolons zwischen dem Befehl (bzw. Text) und der Variablen (bzw. Variablenliste)

kann auch hier bestimmt werden, ob ein Leerzeichen und ein Fragezeichen vor der ersten Eingabestelle erscheinen soll oder nicht.

FORM INPUT

Eine Kombination aus Input- und Input\$-Befehl ist der "FormInput". Sie können damit eine bestimmte Länge des einzugebenden Textes vorherbestimmen. Es ist eine maximale Länge von 255 Zeichen möglich. Wird vom Anwender die vorgesehene Stringlänge erreicht, bleibt der Cursor an der letzten Stelle stehen und es ertönt ein Alarmton (falls die Lautstärke an ist). Veränderungen am Text sind auf die gleiche Art und Weise möglich wie beim Input-Befehl, ebenso die Eingabe von Sonderzeichen. Die Angabe einer Variablenliste oder eines vorangestellten Textes ist hier allerdings nicht möglich. Es wird generell der Cursor auf die erste Stelle des Eingabebereichs gesetzt.

z.B.:

Form Input 32,T_ext\$

Positionieren können Sie den Eingabebereich übrigens, wie auch bei Input und Line Input, indem Sie mit "Print At(XP,YP);" den Cursor an die Stelle bringen, wo das erste Zeichen der Eingabe, bzw. das erste Zeichen des Input-Textes erscheinen soll. Durch das dem Print-Befehl nachgestellte Semikolon wird ja der Wagenrücklauf (CR = Carriage Return) und der Zeilenvorschub (LF = Line Feed) unterdrückt. Das nächste Zeichen einer TOS-Ausgabe wird dann an die Position direkt hinter dem zuletzt ausgegebenen Zeichen gesetzt.

On Error Gosub N.ame_error

ON ERROR GOSUB

Es kann eine Procedure definiert werden, zu welcher das Programm verzweigt, wenn ein Fehler auftritt. In diesem Fall wird

keine Fehlermeldung ausgegeben, sondern das Error-Handling wird dem Programmierer überlassen. Dazu gibt es eine Informationseinheit, die Auskunft darüber gibt, welcher Fehler gerade aufgetreten ist. Und zwar wird in der reservierten Variablen "Err" beim Auftreten eines Errors die entsprechenden Fehlernummer eingetragen. So ist es möglich, anhand dieser Fehlernummer eigene Fehlermeldungen auszugeben oder das Programm entsprechend Ihrer Vorstellung, wie am besten auf einen bestimmten Fehler reagiert werden soll, zu verzweigen.

```
z.B.:  
  
On Error GOSUB E_errors  
A%=34*0  
Enter:  
B=Int(Sqr(23.7^8)/A%)  
On Error  
Print B  
Edit  
Procedure E_errors  
  If Err=0  
    Add A%,.0000000001  
  Endif  
  Resume Enter  
Return
```

In diesem Beispiel wird versucht, eine Zahl durch Null zu teilen. In diesem Fall würde der Interpreter eine Alert-Box mit der Fehlermeldung "Division durch Null" (Err=0) auf den Bildschirm bringen. Dies kann unterdrückt werden, indem durch "On Error GOSUB" zu einer Procedure verzweigt wird, in welcher hier einfach der Null-Wert um .0000000001 erhöht wird und dann dorthin zurück gesprungen wird, wo der Fehler aufgetaucht ist. Mit dem abschließenden Befehl "On Error" wird dem Interpreter wieder das Error-Handling übergeben. Eine Liste der möglichen Fehlermeldungen und der dazugehörigen Fehlernummern finden Sie im Anhang. Außerdem wurde in diesem Programm an mehreren Stellen das Error-Handling selbst verwaltet, so daß Sie die Anwendung dieses Befehls dort nachschlagen können.

ERR

Dies ist eigentlich kein Befehl, sondern wie oben bereits erwähnt, eine reservierte Variable. Neben "Fatal" die einzige reservierte Variable, die das GfA-BASIC kennt. Err enthält bei Auftreten eines System- oder Interpreter-Errors die Fehlernummer dieses Errors (s.Fehlerliste).

RESUME

Mit "Resume" (engl.: wieder aufnehmen) kann ein Labelname übergeben werden, von dem ab das Programm nach dem Auftreten eines selbst verwalteten Fehlers fortgesetzt werden soll. Wird Resume allein ohne Labelnamen benutzt, springt das Programm von der programmeigenen Error-Routine zu der Zeile zurück in der der Fehler aufgetaucht ist und wiederholt diese. Die dritte Möglichkeit ist, daß mit "Resume Next" die auf die fehlerhafte Zeile folgende Programmzeile als Sprungziel bestimmt und das Programm dort fortgesetzt wird. Befindet sich das angegebene Label außerhalb der Error-Routine, wird das Register, in dem die Gosub-Sprünge festgehalten werden, gelöscht und alle globalen Variablen wieder restauriert. Bei evtl. auftretenden Bomben-Errors (Err = 102 - 109) können "Resume" und "Resume Next" nicht eingesetzt werden. In diesem muß "Resume Label" verwendet werden.

FATAL

"Fatal" übergibt Ihnen einen Wert von -1 oder 0. Es handelt sich hier um eine reservierte Variable (s.Err), die zwar mit "Let" ebenfalls benannt werden kann, jedoch den zugewiesenen Wert nicht annimmt. Die beiden Werte, welche durch "Fatal" geliefert werde, haben folgende Bedeutung:

- 0 = es ist ein allgemeiner Fehler aufgetreten. Die Variablen- und Labeladressen, sowie der Gosubadressenstapel sind intakt.
- 1 = es ist ein Systemfehler aufgetreten, durch den die Adressenlage zerstört wird (Absturz). Im GfA-BASIC werden diese Fehler ebenfalls abgefangen, da das System bei den "normalen" BombenErrors die Registerinhalte in einen besonderen Bereich rettet, der, falls kein Total-Breakdown (80 Bomben) stattgefunden hat, zur Reinitialisierung wieder ausgelesen werden kann.

z.B.:

```
On Error Goto E_rrors
Error 102
Print "Fatal-Error!"
Goto Label.2
Label: Print "Normal-Error!"
Label.2:
Print "Programm geht ab hier weiter!"
U=Inp(2)
Edit
Procedure E_rrors
  Print Err;"Fatal"
  If Fatal=0
    Resume Label
  Else
    Resume Next
  Endif
Return
```

Der Rettungsspeicher hat folgende Aufteilung:

Adresse \$0380 (896)	- sind die geretteten Daten gültig, wird hier der Wert 305419896 eingetragen
" \$0384 (900)	- die alten Datenregister (D0 - D7)
" \$03A4 (932)	- die alten Adressregister (A0 - A6)
" \$03C0 (960)	- alter Super-Stackpointer
" \$03C4 (964)	- Error-Index
" \$03C8 (968)	- altes Reg. A7 (User-Stackpointer)
" \$03CC (972)	- 16 alte Super-Stack-Words

ERROR

Durch "Error x" kann ein Fehler absichtlich in das Programm eingebaut werden. "x" steht hier für die Fehlernummer, die den zu simulierenden Fehler kennzeichnet (s. Fehlerliste im Anhang). Sie brauchen bei Einsatz dieses Befehles keine Sorge zu haben. Es wird nur so getan, als ob ein Fehler aufgetreten ist. Sonst verhalten sich so produzierte Errors exakt wie ihre großen Brüder, die echten Errors. Es wird eine Fehlermeldung ausgegeben, oder, falls "On Error Gosub" verwendet wurde, zu der Fehler-Prozedur verzweigt. Nach Ausgabe einer Fehlermeldung durch "Error" wird auch das Programm abgebrochen.

LET

Der GfA-BASIC-Interpreter reagiert auf Variablennamen, die mit einem BASIC-Befehl identisch sind, wie jeder andere BASIC-Interpreter auch, wenn diese Variable ohne den Zuweisungsbefehl "Let" definiert wurde. Er gibt im Editor eine entsprechenden Fehlermeldung aus. Wurde allerdings die Variable durch "Let" definiert, behandelt er diese, auch wenn ihr Name mit dem eines BASIC-Befehls identisch ist, wie jede andere Variable auch. Die einzige Ausnahme bilden hier die oben erläuterten Variablen "Err" und "Fatal". Ihr Name wird in keinem Fall als ein allgemeiner Variablenname akzeptiert, bzw. gewertet.

z.B.:

```
Let Gosub$="BASIC"
Print Gosub$
      Ausgabe:  BASIC

If N.ewname$>""
If Left$(N.ewname$)<>" "
```

Wir kommen nun zu einigen hochinteressanten Stringfunktionen. Mit diesen Funktionen können Textausdrücke sehr wirkungsvoll analysiert und modifiziert werden.

LEFT\$

Es können hiermit beliebig viele Zeichen eines Strings vom Anfang aus gelesen werden. Dabei wird in Klammern nach dem Befehl der String-Variablenname oder ein Textausdruck angegeben und gegebenenfalls nach einem Komma die Anzahl der Zeichen, die gelesen werden sollen oder eine Variable, die diesen Wert beinhaltet. Wird in der Klammer die Zeichenanzahl weggelassen, wird nur der erste Buchstabe des angegebenen Strings gelesen. Wird als Anzahl der zu lesenden Zeichen ein höherer Wert angegeben, als der Ausdruck lang ist, wird der untersuchte String im Ganzen zurückgegeben. Ist der angegebene String ohne Inhalt, wird ein Nullstring geliefert. Ein Beispiel finden Sie im Anschluß an "Rset".

RIGHT\$

"Right\$" hat die gleiche Funktion wie "Left\$". In diesem Fall wird die angegebene Zeichenanzahl vom Ende des Strings aus gelesen. Die Syntax der beiden Befehle ist identisch. Ein Beispiel finden Sie im Anschluß an "Rset".

MID\$

Eine Kombination der beiden vorangegangenen Funktionen ist der Befehl "Mid\$". Diesem werden in Klammern nach dem Textausdruck oder einer Textvariablen, die den zu untersuchenden Ausdruck enthält, zwei, durch Komma getrennte numerische Werte oder Variablen übergeben. Der erste Wert gibt die Stelle im String an, von der aus gelesen werden soll. Der zweite bestimmt die Anzahl der Zeichen, die ab dieser Stelle gelesen werden sollen. Ist der zweite Wert größer, als die nach der Anfangsposition verbleibenden Textstellen, wird maximal der Reststring bis zum Stringende zurückgegeben. Wird als Anzahl eine Null übergeben, wird auch ein Nullstring geliefert. Ein Beispiel finden Sie im Anschluß an "Rset".

INSTR

Diese Funktion übergibt einen numerischen Wert, der angibt, an welcher Position eines Strings ein anderer Ausdruck zu finden ist. Diese beiden Strings werden in Klammern der Funktion als Parameter übergeben. Zusätzlich ist es möglich, eine Position innerhalb des zu untersuchenden Strings anzugeben, von der ab mit der Suche begonnen werden soll. Die Angabe dieser Stelle ist Ihnen allerdings freigestellt. Erfolgt sie nicht, wird ab der ersten Stelle des Strings mit der Suche begonnen. Kann der zu suchende String in dem zu untersuchenden String nicht gefunden werden, wird eine 0 zurückgegeben. Sind beides Nullstrings, wird eine 1 geliefert. Bei der Angabe des Suchstrings ist darauf zu achten, daß hier zwischen Groß- und Kleinschreibung unterschieden wird.

z.B.:

A\$="ATARI ST"

B\$="I S"

C\$="Ata"

A%=Instr(2,A\$,B\$)

'untersuche ab der 2.Stelle, an welcher Position B\$ in A\$ steht.

B%=Instr(7,A\$,B\$)

```
'untersuche ab der 7.Stelle
C%=Instr(A$,C$)
'untersuche ab der 1. Stelle
D%=Instr(A$,Left$(C$,1))
'untersuche ab der 1.Stelle, ob das erste Zeichen von 'C$ in A$
enthalten ist und gebe die Position zurück.
Print A$'B$'C$'D$
U=Inp(2)
Edit
```

Ausgabe: 5 0 0 1

Ein weiteres Beispiel finden Sie im Anschluß an "Rset".

LSET

Durch "Lset" können Textausdrücke in einen bereits bestehenden String eingesetzt werden. Der neue Ausdruck wird von links beginnend in den alten Ausdruck eingefügt. Wenn der neue Ausdruck länger ist als der alte, werden die Stellen des neuen Ausdrucks, die über die Länge des alten Ausdrucks hinausreichen, abgeschnitten. Ist dagegen der neue Ausdruck kürzer als der alte, werden die restlichen Stellen des alten Ausdrucks mit Leerzeichen aufgefüllt. Generell ist es so, daß anschließend von dem alten Ausdruck in dem neuen String nichts mehr zu finden ist. Ein Beispiel finden Sie im Anschluß an "Rset".

RSET

"Rset" hat die gleiche Funktion wie "Lset", nur daß hier der neue Textausdruck von rechts beginnend in den alten eingesetzt wird. Alles andere ist mit "Lset" identisch.

Beispiel zu den Textfunktionen:

Mit diesem Beispiel soll verdeutlicht werden, daß durch die verschiedenen Textfunktionen mit einem relativ geringen Programmieraufwand sogar komfortable Textverarbeitung möglich wird. Es soll Ihnen zur Anregung dienen, sich mit diesem sehr reizvollen Thema näher auseinander zu setzen. Nehmen Sie sich die Zeit, diese Mini-Textverarbeitung zu verstehen. Das Wissen um diese reinen Stringfunktionen wird Ihnen in vielen Beziehungen zu mehr Verständnis für fremde Programmstrukturen verhelfen.

```
A$="Ihnen stehen nachfolgend 5 Zeilen"
A$=A$+" zu je 40 Zeichen zur Texteingabe"
B$="zur Verfügung. Geben Sie bitte einen"
B$=B$+" beliebigen (sinnvollen) Text ein!"
Print At(6,3);A$
Print At(6,4);B$
Print
Dim T_ext$(10)
For I%=1 To 5
    Print At(1,I%+6);" ==> ";
    Form Input 40,T_ext$(I%)
    T_ext$(I%+5)=T_ext$(I%)
Next I%
Print
Print Chr$(27);"j"
Print "    Suchstring   : ";
Line Input "",Search$
Print Chr$(27);"k"
Print Chr$(27);"A"
Print Space$(79)
Print Chr$(27);"A";Chr$(27);"A"
Print "    Ersatzstring : ";
Line Input "",Replace$
Print Chr$(27);"A";Space$(79)
For I%=1 To 5
    P_os%=0
    Enter1:
```

```

A%=Instr(P_os%,T_ext$(I%),Search$)
If A%>0
    P_os%=A%+1 Print At(A%+5,I%+6);Chr$(27);"p";Search$
    Pause 10
    Print At(A%+5,I%+6);Chr$(27);"q";Search$
    S_len%=Len(Search$)
    T_len%=Len(T_ext$(I%))
    R_len%=Len(T_ext$(I%))-S_len%-A%+1
    L_text$=Left$(T_ext$(I%),A%-1)
    R_text$=Right$(T_ext$(I%),R_len%)
    T_ext$(I%)=L_text$+Replace$+R_text$
    Print At(6,I%+6);T_ext$(I%)
    Goto Enter1
Endif
Next I%
For I%=1 To 5
    P_os%=0
    Enter2:
    A%=Instr(P_os%,T_ext$(I%)," ")
    If A%>0
        T_extdummy$=Mid$(T_ext$(I%),P_os%,A-P_os%)+""+T_extdummy$
        P_os%=A%+1
        Goto Enter2
    Endif
    W_orddummy$=Mid$(T_ext$(I%),P_os%,Len(T_ext$(I%))-P_os%+1)
    T_extdummy$=W_orddummy$+" "+T_extdummy$
    Print At(6,I%+6);T_extdummy$
    T_extdummy$=""
Next I%
Deffill ,2,8
Pbox 0,0,639,399
Print Chr$(27);"p"
For I%=1 To 5
    E_mpty$=String$(75,"X")
    Rset E_mpty$=T_ext$(I%)
    Print At(6,I%+6);E_mpty$
    Pause 150
    Print At(6,I%+6);Space$(75)
    For J%=0 To Len(T_ext$(I%+5))
        E_mpty$=Space$(J%)

```



```
Lset E_empty$=T_ext$(I%+5)
Pause 2
Sound 1,12,11,5
Pause 2
Print At(6,I%+6);E_empty$
Sound 1,0
Next J%
Next I%
For I%=1 To 5
  Print At(30,22);Chr$(27);"p";"Bitte Taste drücken"
  Sound 1,12,6,4
  Pause 10
  Print At(30,22);Chr$(27);"q";"Bitte Taste drücken"
  Sound 1,12,6,5
  Pause 10
Next I%
Sound 1,0
U=Inp(2)
Edit
```

Es wurden hier einige Befehle verwendet, deren Bedeutung später noch beschrieben werden soll.

Name Selected\$ As N.ewname\$

NAME AS

Die Syntax dieses Befehls ist denkbar einfach. Wie oben zu sehen ist, werden ihm zwei Parameter übergeben. Der erste davon ist der Dateiname, dessen Name verändert werden soll. nach einem angehängten "As" wird ein weiterer Name übergeben, der nun an die Stelle des alten Namens tritt. Beide Namen können entweder als Stringausdruck, als Stringvariable oder als Kombination von beidem übergeben werden. Unter "EXIST" finden Sie die Pfadstruktur, die auch hier verwendet werden kann. Zu beachten ist bei diesem Befehl, daß die Laufwerks-Spezifikation (z.B. A:) bei beiden Namen gleich ist. Ist beim alten Namen eine Diskettenstation angegeben und ist diese Station auch die aktu-

elle, dann kann beim neuen Namen die Spezifikation entfallen. Innerhalb einer Station kann auch ohne weiteres eine im Hauptdirectory verzeichnete Datei durch eine entsprechende Angabe im neuen Namen in einen Ordner verlegt werden, falls der angegebene Ordner vorhanden ist. Ist er nicht vorhanden, muß er natürlich vorher eingerichtet werden.

z.B.

```
Mkdir "Akte"
Fileselect
"*.*", "", Aa$
If Aa$>" " And Aa$<>"\"
    Name Aa$ As "\Akte\"+Aa$
Endif
Edit
```

MKDIR

In obigem Beispiel tritt ein weiterer Diskettenbefehl auf, der deshalb zum Anlaß genommen werden soll, sich näher mit Befehlen dieser Art zu beschäftigen. Mit dem Befehl "Mkdir" können in einer beliebigen Diskettenstation entweder im Haupt-Directory oder einem beliebigen Ordner weitere Ordner angelegt werden. Dabei wird wieder die unter "EXIST" beschriebene Pfadstruktur verwendet. Als Überblick folgt eine Auflistung verschiedener Varianten dieses Befehls.

Mkdir "akte" = erzeugt einen neuen Ordner auf der aktuellen Diskette entweder im Haupt-Directory oder in dem Ordner, der gerade geöffnet ist.

Mkdir "b:\fach_a\akte" = erzeugt einen neuen Ordner auf Disk B im Ordner "FACH_A"

A\$="\fach_a\
Mkdir A\$+"akte" = erzeugt einen neuen Ordner auf der
aktuellen Disk im Ordner
"FACH_A"

Ist bereits ein Ordner mit derselben Pfadbezeichnung vorhanden, wird eine Fehlermeldung ausgegeben. Die Pfadbezeichnung und der Ordnername wurden hier mit Absicht klein geschrieben, um deutlich zu machen, daß das System diese Angaben selbstständig in Großbuchstaben umwandelt. Dies ist grundsätzlich bei allen Diskettenoperationen gegeben. Zurückgegeben werden die Dateinamen vom System allerdings nur in Großbuchstaben (z.B. der Rückgabestring im Befehl "Fileselect" oder "Dir\$").

CHDIR

Hiermit kann auf der aktuellen Diskette ein Ordner geöffnet werden, bzw. falls schon einer geöffnet ist, wird dieser vorher geschlossen und der angegebene Ordner geöffnet. Dem Befehl wird einfach der Name des Ordners übergeben, der geöffnet werden soll (Pfadstruktur s. "EXIST"). Diese Bezeichnung kann wieder entweder in einer Variablen enthalten sein, oder als Textausdruck übergeben werden (z.B. Chdir "Fach_a").

RMDIR

Dieser Befehl ist genauso zu behandeln, wie "Chdir", nur daß der angegebene Ordner nicht geöffnet, sondern gelöscht wird. Befinden sich allerdings noch Dateien in dem Ordner, kann dieser Befehl nicht ausgeführt werden und es erscheint eine Fehlermeldung. Dateien können mit dem Befehl "Kill" gelöscht werden.

DIR\$

Mit "Print Dir\$(x)" oder "A\$=Dir\$(x)" kann der Name des zur Zeit geöffneten Ordners ermittelt werden. "x" steht hier für die Diskstation, auf der nachgesehen werden soll.

- 0 = aktuelles Laufwerk
- 1 = Laufwerk A
- 2 = Laufwerk B

Ist zur Zeit kein Ordner geöffnet, wird ein Leerstring zurückgegeben.

DIR

Durch "Dir" kann man sich auf verschiedene Weise Inhaltsverzeichnisse entweder einer beliebigen Diskettenstation oder eines bestimmten Ordners ausgeben lassen. Dabei ist es auch möglich, diese Directory in eine bestimmte Diskettendatei schreiben oder auf dem Drucker ausdrucken zu lassen (im Listing finden Sie eine Routine "Procedure Accessory", die die aktiven Accessories aufruft. Falls Sie das "CONTROL.ACC" aktiv im Desktop haben, können Sie vom BASIC aus damit die DruckerAnpassung aufrufen. Diese Prozedur ist als Utility programmunabhängig. Aufruf im Direct-Modus: @Accessory). Zur Pfadangabe können sämtliche unter "EXIST" beschriebenen Spezifikationen verwendet werden.

z.B.:

Dir "*.*)" =
oder Dir

listet alle Dateien außerhalb von
Ordern der aktuellen Disk auf
dem Bildschirm auf

```
A$="A:\Fach_a\  
B$="D????"  
Dir A$+B$+".EXT"=
```

listet alle Dateien der Disk A im Ordner "Fach_a", deren Name mit dem Buchstaben "D" beginnt, gleichzeitig mindestens fünf Buchstaben hat und den Extender ".EXT" trägt.

```
Dir "\*.EXT" To "B:\DIRECTOR\SUBDIR_1.SUB"
```

= schreibt alle Dateien mit dem Extender ".EXT", die sich auf der aktuellen Diskette außerhalb von Ordnern befinden in eine neue Datei mit dem Namen "Subdir_1.sub", die in dem bestehenden Ordner "Director" abgelegt wird.

Wird im letzten Beispiel statt der Pfadbezeichnung hinter "To" der Ausdruck "LST:" angefügt, würden die Dateinamen auf dem Drucker ausgegeben.

FILES

Genügen Ihnen die reinen Dateinamen nicht, die mit "Dir" ausgegeben werden, können Sie mittels dieses Befehls auch erweiterte Attribute der einzelnen Dateien erfahren. So haben Sie die Möglichkeit, sich zusätzlich die Größe der Dateien, das Datum und die Uhrzeit ihrer Erstellung ermitteln zu lassen. Im übrigen wird dieser Befehl exakt genauso angewendet wie "Dir" (z.B. "Files" listet alle Dateien der aktuellen Disk außerhalb von Ordnern, sowie alle Ordner auf). Um zum Beispiel nur die Größe einer bestimmten Datei zu ermitteln, gibt es verschiedene Möglichkeiten. Eine davon finden Sie in der Prozedur "D.size". Eine andere könnte evtl. so aussehen:

```
@Get_size
Print Datsize%
U=inp(2)
Edit
Procedure Get_size
  Fileselect "*.*", "", Selected$
  If Exist(Selected$)<>0
    Files Selected$ To "buffdat"
    Buffer$=Space$(45)
    Bload "buffdat",Varptr(Buffer$)
    Datsize%=Val(Mid$(Buffer$,16,7))
  Endif
Return
```

Hier wird ein kleiner Umweg gegangen, da leider der von "Files" in diesem Fall gelieferte String nicht direkt in eine Variable übergeben werden kann. So wird die Infozeile der gewählten Datei erstmal in einer kleinen Datei zwischengespeichert, um dann mit "Bload" in einen vorbereiteten Puffer geschrieben zu werden. Jetzt ist der String unter "Buffer\$" in einer Variablen verfügbar. Nun braucht man nur noch die Stellen der Zeile, in der die Dateigröße steht (16-22) mit "Val" in einen numerischen Wert zu verwandeln, und schon hat man die Bytegröße als numerische Variable. So einfach ist das!

CHDRIVE

Es kann ein Laufwerk zum aktuellen Laufwerk bestimmt werden. "Chdrive x" wechselt zu dem durch "x" spezifizierten Laufwerk (A: = 1, B: = 2). Wird bei der Pfadbezeichnung, die den meisten Diskettenbefehlen zu übergeben ist, kein Laufwerk bestimmt (z.B. Mkdir "\Ordner"), wird von nun ab das hiermit aktualisierte Laufwerk angesprochen.

LOAD

Mit "Load" kann ein beliebiges BASIC-Programm in den Arbeitsspeicher geladen werden. Prinzipiell verhält sich dieser Befehl so, als wenn Sie den Menüpunkt "Load" im Editormenü anklicken oder die Taste "F1" drücken. Auch hier ist die in "EXIST" beschriebene Pfadstruktur zu verwenden. Wird dem Programmnamen kein Extender beigefügt, wird vom Interpreter selbstständig der Extender ".BAS" angehängt.

z.B.:

```
Load "\Utility\Drei_d"
```

SAVE

Ebenfalls identisch mit der Menüfunktion "Save" ist dieser Befehl. Er speichert das im Programmspeicher befindliche Programm unter dem angegebenen Namen auf Diskette. Die Syntax ist identisch mit "Load". Sollte sich bereits ein Programm mit demselben Namen auf der angesprochenen Diskette befinden, wird dieses durch das neu zu speichernde Programm gelöscht, bzw. überschrieben.

PSAVE

Für diesen Befehl gilt die gleiche Ausführung wie zu "Save". Er bietet nur eine kleine Besonderheit, die aber in ihrer Wirkung sehr beeindruckend ist. Das "P" in diesem Befehlsnamen steht für "protected" (engl.: geschützt). Das heißt, daß Programme, die hiermit abgespeichert wurden:

1. nicht mehr gelistet werden können
2. sofort nach dem Laden gestartet werden (Autostart)
3. der Versuch, mit dem Interpreter das (nicht sichtbare) Listing zu bearbeiten, durch einen "ZwangsResetknopf-Druck" bestraft wird. Der Interpreter wird Ihnen danach also jegliche weitere Dienstleistung verweigern.

CHAIN

Dieser Befehl ist eigentlich identisch mit "Load". Allerdings wird das hiermit geladene Programm nach dem Laden automatisch gestartet. Da bei diesem Programmstart, wie sonst auch, alle Variablen und Felder gelöscht werden, können zwischen den "gechainten" Programmen diese nicht ausgetauscht werden. Hier hilft nur eine Möglichkeit. Alle wichtigen Werte des ersten Programms, die im zweiten Programm weiterverwendet werden sollen, müssen vor dem Chain-Befehl in eine Datei geschrieben werden (evtl. mit "Print #1"), aus dem sie mit dem "gechainten" Programm wieder herausgelesen werden. Die Syntax dieses Befehls ist identisch mit "Load" oder "Save".

z.B.

PROGRAMM 1

```
A=1025
B=399.22
C$="BASIC"
Open "O",#1,"Vars.Dat"
Write #1,A,B,C$
Close #1
Chain "Prog_2.Bas"
```

PROGRAMM "Prog_2.":

```
Open "I",#1,"Vars.Dat"
Input #1,A,B,C$
Close #1
Print "Variablen aus Programm 1 : ";A'B'C$
U=Inp(2)
Edit
```

LIST

Möchten Sie das im Arbeitsspeicher befindliche Programm auf dem Monitor auflisten oder als ASCII-File auf Diskette abspeichern, hilft Ihnen dieser Befehl. Geben Sie im Direktmodus oder

als Programmzeile nur den Befehlsnamen ohne Angabe eines Dateinamens ein, wird das gesamte Programm auf dem Ausgabebildschirm ausgegeben. Bei der mir vorliegenden Version des GfA-BASIC's läßt sich dieses Listing dann jedoch nicht mehr abbrechen. Oder, besser gesagt, es läßt sich mit der Tastenkombination "Control/C" zwar abbrechen, dann jedoch mit dem unangenehmen Effekt, daß Sie wieder Ihren Resetknopf bemühen müssen, denn der Interpreter ist zwischenzeitlich "baden" gegangen (muß auch mal sein!). Meine Erfahrungen mit dieser "Abbruchfunktion" habe ich schon der Firma "GfA-Systemtechnik" mitgeteilt und es ist anzunehmen, das sich der Programmierer dieses BASIC's dazu etwas einfallen lassen wird. Wenn dieses Buch erscheint, werden die dann erhältlichen Versionen wohl schon mit einer komfortableren Möglichkeit ausgestattet sein, den Listingausdruck ab- oder unterbrechen zu können. Eine Unterbrechungsfunktion existiert allerdings. Drücken Sie während des Auflistens auf dem Bildschirm die Tastenkombination "Control/S", wird das Listing unterbrochen. Mit der Kombination "Control/Q" kann es anschließend wieder fortgesetzt werden. Wollen Sie das Listing als sogenanntes "ASCII"-File auf Diskette abspeichern, übergeben Sie dem Befehl (in Anführungsstrichen) den Namen der Datei, in welcher das Listing abgelegt werden soll. Dieser Befehl ist absolut identisch mit der Editor-Funktion "Save,A". Sie können also hiermit abgespeicherte Programme jederzeit mit der Editor-Funktion "Merge" in ein anderes, im Arbeitsspeicher befindliches Programm einbinden. Die Anführungsstriche zum Dateinamen können gänzlich vernachlässigt werden, da sie vom Interpreter (falls nicht vorhanden) selbsttätig gesetzt werden. Der Extender zum Dateinamen kann ebenfalls vernachlässigt werden. Er wird auch vom Interpreter gesetzt (".LST"). Ein ASCII-File ist eine Datei, in welcher jedes einzelne Text- und Steuerzeichen des zu speichernden Textes in der Reihenfolge seines Auftretens mit dem ihm zustehenden ASCII-Wert (0-255) nacheinander in die Datei geschrieben wird. Die meisten textverarbeitenden Programme besitzen die Möglichkeit, ASCII-Files zu speichern und zu laden. D.h., daß Texte dieser Art unter den verschiedenen Programmen, Systemen und Computern austauschbar sind, soweit diese sich an den "American Standard Code for Information Interchange" halten (engl.: amerikanischer Standard-Code zur Infor-

mationsübertragung). Dateien mit anderen Formaten, wie z.B. die ".BAS"-Dateien des GfA-BASIC's werden nach einem anderen Schema codiert. Dieser Code kann vom jeweiligen Programmierer selbst entwickelt werden, um zum Beispiel eine höhere Lade- oder Speichergeschwindigkeit, einen speziellen Effekt oder eine erschwerte Lesbarkeit zu erreichen (s."Psave"). Diese Dateien sind dann allerdings nicht mehr mit anderen Programmen (Interpretern etc.) austauschbar. Sie sind nicht mehr "kompatibel" (compatibility, engl.: Vereinbarkeit/ Verträglichkeit).

DEFLIST

Sie haben die Möglichkeit zwischen zwei Arten der Listingdarstellung zu wählen. Auch der im Editor befindliche Programmtext wird hierdurch angesprochen.

Deflist 0 = es werden alle BASIC-Befehls- und Funktionsnamen in großen Buchstaben dargestellt. Alle Variablenamen werden klein geschrieben.

z.B.:

```

DEFTXT ,16,,25
TEXT 10,200,500,"Mini-Demo !"
GET 10,175,510,210,aaa$
CLS
DO
    MOUSE x,y,k
    PUT x,y,aaa$,6
    PUT x,y,aaa$,6
    IF k=1
        PUT x,y,aaa$,7
    ENDIF
    IF k=2
        CLS
    ENDIF
LOOP

```

Deflist 1 = Es wird von allen Befehls- Funktions- und
bzw. >1 Variablennamen nur der erste Buchstabe groß
geschrieben.

z.B.:

Auch ein Beispielprogramm kann nützlich sein! Besonders bei umfangreichen Programmen kann es manchmal wichtig sein, sich eine Liste der verwendeten Variablen und Prozeduren erstellen, bzw. ausdrucken zu lassen. Die GfA-".BAS"-Files bieten die Möglichkeit, sich diese Liste aus dem Datei-Anfang herauszulesen. Das folgende Utility-Programm hat den Zweck,

1. aus einer beliebigen ".BAS"-Datei die einzelnen Variablen- und Prozedurnamen auszulesen und anschließend in eine Datei "Vars.Map" zu übertragen. Dabei wird die Liste auch auf dem Bildschirm ausgedruckt.
2. Die abgelegten Variablen- und Prozedurnamen aus "Vars.Map" wieder herauszulesen und entweder auf dem Drucker oder dem Monitor auszugeben.

```
Dim A$(1000)
Again: Fileselect "\*.Bas","",Selected$
If Selected$=""
    Goto Again
Endif
If Selected$=""
    Edit
Endif
Open "U",#1,Selected$
Open "O",#2,"vars.map"
Alert 1,"Taste drücken = ABBRUCH",1,"OKAY",Uu
Repeat
    Zeichen%=Inp(#1)
    If Zeichen%>31
        Aa=0
        B$=Chr$(Zeichen%)
```

```

    A$(X)=A$(X)+B$
Else
    If Aa=0
        Aa=1
        Print #2;A$(X);" ";
        Print A$(X),
        Inc X
    Endif
    Inc Aa
Endif
Until Mousek Or Len(Inkey$)
Close
Open "U",#1,"vars.map"
For I=1 To Lof(#1)
    Zeichen%=Inp(#1)
    If Chr$(Zeichen%)<>" "
        Aa=0
        A$(Xx)=A$(Xx)+Chr$(Zeichen%)
    Else
        If Aa=0
            Inc Xx
        Endif
        Aa=1
        Inc Xx
    Endif
    Inc Aa
Next I
Close #1
Alert 2,"Drucker oder Monitor?",0,"Drucker|Monitor",U
For I=0 To Xx
    If U=1
        Lprint A$(I),
    Else
        Print A$(I),
    Endif
Next I
Alert 1,"READY !!",1,"OKAY",U
Edit

```

OPEN

Dieser Befehl vereinigt sehr viele positive Eigenschaften in sich. Seine Syntax ist auf den ersten Blick etwas kompliziert, aber wenn man sich nicht in's Bockshorn jagen läßt, gewinnt man ihn schnell lieb. Die Möglichkeiten, die dem Programmierer mit den Befehlen zur Dateiverwaltung an die Hand gegeben werden, sind schier unüberschaubar. Syntax:

```
Open "Modus",#Kanalnummer,"Dateiname"
```

Modus:

- "O" öffnet die Datei zum Herauslesen von Daten, oder, falls die angegebene Datei noch nicht vorhanden ist, installiert diese im angegebenen Kanal.
- "I" öffnet eine vorhandene Datei zum Hineinschreiben von Daten
- "A" öffnet eine vorhandene Datei und setzt den Filepointer auf das Dateiende. Alle an diese Datei auszugebenden Daten werden an das Dateiende angehängt (append;engl.:anhängen/hinzufügen). Soll die Datei anders beschrieben werden, muß der angegebene Kanal mit "Close #Kanalnummer" vorher wieder geschlossen werden, um dann mit Open "O" z.B. wieder geöffnet zu werden.
- "U" In diesem Modus kann in eine vorhandene Datei geschrieben und aus ihr gelesen werden.
- "R" es kann eine Random-Access-Datei eingerichtet werden (mehr dazu unten).

Kanalnummer:

Es können gleichzeitig 100 Dateien eröffnet sein. Dabei wird zur Identifikation jeder Datei eine Zahl zugewiesen. Unter Angabe dieser Zahl kann nun z.B. mit den Befehlen "Print #Kanalnummer" oder "Input #Kanalnummer" (Sie wissen schon: die Kanalvariante) auf jede dieser Dateien zugegriffen werden. Die angegebene Nummer muß im Bereich von 0 bis 99 liegen.

Dateiname:

Hier kann wieder die schon so oft erwähnte Pfadstruktur verwendet werden (s.EXIST). Allerdings gibt es hier noch eine Ausnahme: Sie können mit fast allen Diskettenbefehlen auch die verschiedenen Schnittstellen Ihres Atari ST ansprechen. Dazu müssen Sie ebenfalls eine Quasi-Datei eröffnen. Mit Datei ist in diesem Fall jedoch nicht eine Diskettendatei gemeint, sondern der jeweilige Port. Es stehen Ihnen sieben solcher Ports zur Auswahl:

"CON:" = Consolport
"LST:" = Druckerport
"PRN:" = ebenfalls Druckerport
"AUX:" = Auxilliary-Port (serieller Port)
"MID:" = zwei MIDI-Ports
"VID:" = Monitor (CON: transparent)
"IKB:" = Tastaturprozessor (die Herz/Lungenmaschine)

Der komplette Ausdruck vor dem Gleichheitszeichen wird in diesem Fall als Dateiname an "Open" übergeben. Die Angabe des Modus' kann bei Verwendung dieser Möglichkeit entfallen.

z.B.

```
Open "" ,#1,"VID:"  
For I%=0 To 31  
    Print #1;Chr$(I%);  
Next I%  
Close #1  
U=Inp(2)  
Edit
```

Dieses ist eine weitere Möglichkeit (s. unter "OUT"), ohne Tastaturbedienung die Sonderzeichen mit den ASCII-Werten von 0 bis 31 zu Gesicht zu bekommen.

Random-Access-Dateien:(random access;engl.:wahlfreier Zugriff)

Alle anderen Zugriffsmöglichkeiten auf Diskettendateien arbeiten grundsätzlich "seriell", d.h. der Reihe nach. Es wird also ein-dimensional Zeichen für Zeichen nacheinander gelesen, bzw. geschrieben. Diese Reihenfolge ist nur durch "Seek" und "Relseek" veränderbar. Mit Random-Access-Dateien ist es möglich, sich eine Reihe von Datensätzen innerhalb einer Datei anzulegen, die dann durch Identifikatoren angesprochen werden können. Der "Open"-Befehl erweitert sich gegebenenfalls hierfür um einen Parameter:

```
Open "R",#Kanalnr., "Dateiname", Satzlänge
```

Unter Satzlänge ist ein Wert zu verstehen, den Sie als Dateiverwalter selbst bestimmen können. Und zwar gibt er die Anzahl der Bytes an, die sie den Datensätzen einer Datei zukommen lassen wollen. Wird diese Angabe bei "Open" unterlassen, wird vom Interpreter eine Datensatzlänge von 128 Byte vorgesehen. Genauer zur Verwaltung dieser Art von Dateien erfahren Sie unter "FIELD".

CLOSE

Alle mit "Open" eröffneten Kanäle müssen mit "Close #Kanalnr." geschlossen werden, um für eine andere Datei verwendet werden zu können. Wird nur "Close" angegeben, werden sämtliche Kanäle zugleich mit diesem Befehl geschlossen. Ein Kanal der noch nicht mit "Close #Kanalnr." geschlossen wurde und mit "Open" angesprochen wird, verursacht eine Fehlermeldung.

LOF

Bei reinen Diskettendateien ist es möglich, falls Sie überhaupt einen Inhalt haben, sich ihre Länge ausgeben zu lassen, bzw. generell ihre Länge festzustellen. "LOF" (Length Of File; engl.: Länge der Datei) wird, in Klammern nachgestellt, die Kanalnummer der Datei übergeben, deren Länge ermittelt werden soll. Dazu muß dieser Datei natürlich vorher durch den Befehl "Open" eine Kanalnummer zugewiesen worden sein. Eine

Anwendungsmöglichkeit habe Sie bereits im Beispiel zu "INPUT\$" kennengelernt. Weitere Möglichkeiten:

```
Open "U",#1,"Dateiname"

Print LOF(#1)

oder A%=LOF(#1)

oder If LOF(#1)>32767
      A$=Input$(32767,#1)
Else
      A$=Input$(LOF(#1),#1)
Endif
Print A$
```

EOF

Durch "EOF(#Kanalnr.)" kann, wieder nur bei Diskettendateien, das letzte Byte der Datei mit der angegebenen Kanalnummer ermittelt werden (falls vorhanden). Die wohl wichtigste Anwendung dieses Befehls ist die, die Fehlermeldung "Fileende erreicht" zu vermeiden.

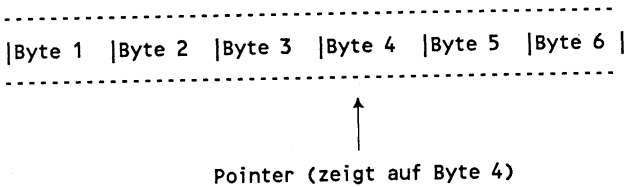
z.B.

```
Open "U",#1,"Dateinummer"
While Eof(#1)=0
  Print Input$(1,#1);
Wend
U=Inp(2)
Edit
```

In diesem Beispiel werden alle Bytes der Datei in ihrer Reihenfolge ausgegeben. Die Schleife wird bei While erst verlassen, wenn der Wahrheitswert von Eof = -1 ist (Eof ist wahr). Solange Eof = 0 ist (Eof ist unwahr, also falsch), wird jeweils ein Byte gelesen und gleichzeitig ausgegeben.

LOC

Durch "Loc (#Kanalnr.)" kann der Abstand des Filepointers zum Dateianfang ermittelt werden. D.h., der Schreib- und Lesezeiger der Datei mit der angegebenen Kanalnummer befindet sich auf dem Byte mit der Nummer, die durch "Loc" zurückgegeben wird.



```
Print Loc(#1) ==>   Ausgabe: 4
```

z.B.:

```

Open "U",#1,"Stand_6"
Seek #1,Int(Lof(#1)/2)
A%=Loc(#1)
Print "Byte: ";A%;" Zeichen: ";Input$(1,#1)';" ASCII: ";
Seek #1,A%
Print Inp(#1)
U=Inp(2)
Edit

```

SEEK

Bei Bedarf können Sie mit diesem Befehl den Filepointer auf ein beliebiges Byte der Datei mit der angeführten Kanalnummer setzen. Dazu wird als erster Parameter die Kanalnummer und als zweiter Parameter, nach einem Komma die Stelle angegeben, auf die der Pointer gesetzt werden soll (s. Beispiel zu "LOC").

RELSEEK

"Relseek" bedeutet "relativ Seek". D.h., daß der Filepointer entweder in Relation zum Fileanfang oder in Relation zum Fileende um eine bestimmte Anzahl von Bytes verschoben wird.

z.B.

```
Open "U",#Kanalnr., "Dateiname
Seek #1,Int(Lof(#1)/2)
'Der Zeiger befindet sich nun in der Mitte
'der angegebenen Datei
Relseek #1,3
'Der Zeiger wird in Richtung Fileende um 3
'Byte weitergesetzt
Relseek #1,-6
'Der Zeiger wird in Richtung Fileanfang um 6
'Byte zurückgesetzt. Er befindet sich jetzt
'3 Byte vor der durch "Seek" bestimmten Posi-
'tion. Print Inp(#1)
'ASCII-Wert des an dieser Stelle befindlichen
'Bytes wird ausgegeben.
Close #1
```

Bei den Befehlen "Seek" und "Relseek" ist darauf zu achten, daß nicht versucht wird, den Zeiger auf eine Position zu setzen, die in der angegebenen Datei gar nicht vorhanden ist, bzw. sein kann (z.B. im Minusbereich oder hinter dem Fileende). Wird dies versucht, wird die Fehlermeldung "Seek falsch?" ausgegeben.

Werden sämtliche Kanalnummern im Programm nur einmal verwendet, müssen die Dateien nicht bei Programmende abgeschlossen werden, durch die Befehle "End" oder "Edit" wird dies vom Interpreter erledigt. Anders bei dem Befehl "Stop". Dieser schließt keine offenen Dateien. Sollte mit "Cont" danach das Programm nicht fortgesetzt werden, ist es ratsam, durch "Close" (ohne Angabe einer Kanalnummer) im Directmodus die Dateien zu schließen.

Vorbemerkung zu "GET#","PUT#" und "FIELD#":

Wie die Behandlung von Variablenfeldern eine Sonderstellung einnimmt, so sind auch die Befehle zur Behandlung von RandomAccess-Dateien eine besondere Erklärung wert. Eine solche Datei kann wie ein Variablenfeld (allerdings nur eindimensional) verschiedene Angaben, die unter einem Oberbegriff (die Feldnummer) zusammengefaßt werden können, auf Diskette abgespeichert werden. Dazu muß zuerst eine Datei eröffnet werden, die durch ihren Arbeitsmodus kenntlich macht, daß es sich hier nicht um eine normal zu behandelnde Datei geht.

```
Open "R",#1,Dateiname,Satzlänge
```

Nachdem dieses getan wurde, kann mit dem Befehl "Field" bestimmt werden, in wieviele Felder der gesamte Satz unterteilt, und wieviele Bytes jedem einzelnen dieser Felder zugeteilt werden soll. Da die Länge des gesamten Datensatzes durch die Angabe von "Satzlänge" im Open-Befehl bestimmt wird, ist es einleuchtend, daß die Summe der Bytes aller angegebenen Felder zusammen nicht größer sein darf, als die Satzlänge. Sie darf aber kleiner sein. Die einmal so bestimmte Größe einer Datei und ihre Einteilung darf von nun an nicht mehr verändert werden. Natürlich können Sie neue Datenfelder in einen neu definierten Datensatz mit demselben Dateinamen einsetzen, nur dann sind die vorher darin enthaltenen Daten nicht mehr zu erreichen. Zum anderen dürfen die durch "Put#" in die Feldeinteilung geschriebenen Datenfelder in ihrer Größe nicht mehr variiert werden, da sonst Daten des einen Feldes in das nächste Feld mit hineingeschrieben werden. Um nun diese festgelegten Feldgrößen auch immer einzuhalten, werden hier die beiden Textfunktionen "Lset" und "Rset" eingesetzt, die dabei Variablen verwenden, die vorher in ihrer Länge auf die Länge der Felder abgestimmt wurden. Diese Stringvariablen werden, wie schon unter diesen beiden Befehlen beschrieben, durch das Einsetzen eines anderen Strings in ihrer Länge nicht verändert. Sie sollten darauf achten, daß die so vorbereiteten Variablen im weiteren nicht auf andere Weise verwendet werden. Es sei denn, daß Sie nicht mehr vorhaben, die Datei mit "Put#" noch einmal anzu-

sprechen. Außerdem ist es nicht möglich, eine einmal so installierte Random-Access-Datei zu erweitern. Erweiterungen sind nur möglich, indem eine neue Datei mit anderen Spezifikationen eingerichtet wird und die Datenfelder der alten Datei in die neue integriert werden. Dazu müßten nach dem Eimerketten-Prinzip die alten Felder mit dem "Get#" - Befehl aus der alten Datei herausgelesen werden, um die dazu verwendeten Variablen anschließend zur Neueinrichtung der anderen Datei an diese weiterzugeben. Achten Sie außerdem darauf, daß bei jedem "Open"-Befehl für eine bestimmte, schon eingerichtete Datei dieser Art immer exakt dieselbe Datensatzlänge angegeben wird, bzw. die neu angegebene Satzlänge größer als die vorhandene ist. Wird nämlich keine Satzlänge genannt, wird diese vom Interpreter selbsttätig auf 128 Bytes festgelegt. Falls Sie dann die angesprochene Datei vorher größer dimensioniert hatten, wird beim Versuch, ein Feld anzusprechen, das mit seiner Länge über die angegebene Satzlänge hinausreicht, die Fehlermeldung "Fileende erreicht" ausgegeben.

Eine Grafik soll verdeutlichen, was gemeint ist.

```
|<=      Ein Datensatz      =>|<= Satzlänge = 45 Byte =>| | | | |
|---|---|---|---|---|---|
|Feld 1 |Feld 2 |Feld 3 |Feld 1 |Feld 2 |Feld 3 |
|=10 Byte|=20 Byte|=15 Byte|=10 Byte|=20 Byte|=15 Byte|
|-----|
```

Diese Grafik zeigt vereinfacht das Prinzip dessen, was zu erklären versucht wurde. Die Anzahl der Felder, die Größe der einzelnen Felder, sowie die Satzlänge können von Ihnen frei bestimmt werden. Begrenzt werden sie nur durch eine maximale Datensatzlänge von 65535 Byte, durch die maximale Eingabemöglichkeit für eine Field-Anweisung (Programmzeilenlänge = max. 255 Zeichen), durch die Notwendigkeit, daß die Summe aller Einträge bzw. Felddängen die angegebene Satzlänge nicht überschreiten darf und durch den Speicherplatz auf Ihrer Diskette.

Zum tieferen Verständnis folgt nun ein Beispielprogramm, daß Ihnen als Anregung zu eigenen Experimenten dienen soll.

Dieses Programm besteht aus drei voneinander unabhängigen Blöcken. Es werden nach den ersten beiden Blöcken alle Variablen und der Bildschirm gelöscht. Im ersten Block wird eine Random-Access-Datei mit einer Satzlänge von 90 Byte installiert. Diese 90 Byte werden mit der Field-Anweisung in drei Datenfelder zu je 30 Byte unterteilt. Gleichzeitig werden die für die Aufnahme des Feldinhaltes vorgesehenen Stringvariablen mit soviel Leerzeichen aufgefüllt, wie für die einzelnen Felder vorgesehen wurde (hier je 30). Im Anschluß daran werden in der For/Next-Schleife sechs Datensätze eingelesen (falls Sie mehr als sechs Freunde haben, können Sie den Schleifenendwert natürlich erhöhen). Nach der jeweilige Eingabe der Feldinhalte, die ja nicht alle die gleiche Anzahl an Zeichen besitzen, werden die Variableninhalte der Input-Variablen durch "Lset" linksbündig in den vorgesehenen, 30 Byte langen Feldstring eingesetzt. Zu guter Letzt wird dann dieser Datensatz mit "Put#" unter Angabe der Datensatznummer (hier der Schleifenindex I%) in die Datei geschrieben.

Die folgenden beiden Programmblöcke demonstrieren zwei Möglichkeiten, diese Daten wieder aus der Datei herauszulesen und auf den Bildschirm zu bringen.

```
Open "R",#1,"Friends",90
Field #1,30 As D_feld.0$,30 As D_feld.1$,30 As D_feld.2$
For I%=1 To 6
    Print At(20,11);"Name      :";
    Form Input 30,N$
    Print At(20,12);"Vorname  :";
    Form Input 30,V$
    Print At(20,13);"Telefon  :";
    Form Input 30,T$
    Lset D_feld.0$=N$
    Lset D_feld.1$=V$
    Lset D_feld.2$=T$
    Put #1,I% Cls Next I%
Close #1
Clear
Open "R",#1,"Friends",90
Field #1,30 As A$,30 As B$,30 As C$
```

```
For I%=1 To 6
  Get #1,I%
  Print "Name      : ";A$
  Print "Vorname   : ";B$
  Print "Telefon   : ";C$
  Print
Next I%
Close #1
Pause 200
Clear
Cls
Open "R",#1,"Friends",90
Field #1,30 As Feld1$,30 As Feld2$,30 As Feld3$
Repeat
  Print At(20,11);"Bitte Feldnummer eingeben (1-6): ";
  Input F_eld%
  Get #1,F_eld%
  Print At(20,10);"Name      : ";Feld1$
  Print At(20,11);"Vorname   : ";Feld2$
  Print At(20,12);"Telefon   : ";Feld3$
Repeat
Until Len(Inkey$)
Cls
Until Asc(Left$(Inkey$))=27
Edit
```

GET

Das obige Beispiel dürfte genügend Aufschluß über die Syntax und Verwendung des "Get#" - Befehls geben. Ganz allgemein wird seine Funktion mit "Datensatz lesen" am besten beschrieben.

PUT

Funktion: Datensatz schreiben. Sonst siehe "Get#".

FIELD

Auch die Funktion dieses Befehls und seine Syntax sind hoffentlich im obigen Beispiel deutlich geworden. Allgemein: Field weist einer beliebigen Anzahl von Datenfeldern einzelne Byte-längen zu und füllt die für die Aufnahme der Felder vorgesehenen Stringvariablen mit dementsprechend vielen Leerzeichen. Zu beachten ist, daß für jeden geöffneten Datenkanal im "R"Modus nur eine Field-Anweisung zugelassen ist. Wird versucht mehrere Field-Anweisungen an denselben Datenkanal zu geben, erscheint die Fehlermeldung "Nur ein Field zu einem Open "R" möglich". Es ist also bei größeren Datenfeldern ratsam, die Namen der aufnehmenden Stringvariablen so kurz wie möglich zu wählen, da die Field-Anweisung in ihrer Länge durch die maximale Eingabezeilenlänge von 256 Zeichen begrenzt wird.

DFREE

Bei Verwendung dieses Befehls wird der momentan freie Speicherplatz der angegebenen Diskettenstation ermittelt und zurückgegeben.

z.B.:

```
A%=Dfree(0)
oder Print Dfree(1)
```

Wie schon erwähnt, werden den Laufwerken Nummern zugewiesen, über die sie bei Befehlen wie diesem zu erreichen sind. Es können maximal 15 Laufwerke angesprochen werden (aktuelles Laufwerk = 0 ; A = 1 ; B = 2 u.s.w)

```
Endif
Endif
Endif
Endif
Endif
```

```

Endif
On Error
Put 250,150,B.ackmenu$,3
@Set.paras
Return
Procedure N.ame_error

```

'Falls bei der Pfadbezeichnung zur Änderung eines Dateinamens ein Fehler
'aufgetreten ist, wird zu dieser Prozedur verzweigt.

```

If Err=-34
    Alert 3,"Falsche Pfadbezeichnung!|Versuchen Sie's nochmal!",1," OKAY
",Kkkk
    Resume N.ameagain
Endif
Resume Next
Return
Procedure Sl.some

```

'Hier wird das Load- und Savemenü gezeichnet und bedient

```

Deffill ,0,0
Pbox 258,158,382,272
Deftext 1,9,0,6
If F.ops%=1
    Text 278,168,"SAVING :"
Endif
If F.ops%=2
    Text 278,168,"LOADING :"
Endif
Graphmode 2
For I%=0 To 1
    Deftext 1,16+I%,0,6
    Text 278,184,"SCREEN 1"
    Text 278,196,"SCREEN 2"
    Text 278,208,"ICON"
    Text 278,220,"PATTERN"
    Text 278,232,"      "
Next I%

```



```

Pause 10
Graphmode 3
Deffill ,2,8
Repeat
    Mouse X,Y,K
    Xpos%=0
    Ypos%=0

```

'Die Menüpunkt-Indizierung wurde weiter oben bereits erwähnt

```

If X>258 And X<382
    Ypos%=Int((Y-163)/12)
    If Ypos%>0 And Ypos%<6
        Pbox 259,163+(Ypos%*12),381,175+(Ypos%*12)
        Pause 6
        Pbox 259,163+(Ypos%*12),381,175+(Ypos%*12)

```

'Um die unter dem Mauszeiger befindlichen Menüpunkte erkennbar
'zu machen, wird im Graphmode 3 zweimal hintereinander eine
'schwarze Pbox auf die jeweilige Zeile gelegt.

```

    Endif
Endif
Until Mousek Or Len(Inkey$)
Graphmode 1
If X>258 And X<382 And Ypos%>0 And Ypos%<6
    If F.ops%=2 And (Ypos%=1 Or Ypos%=2)
        Alert 1,"Das aktuelle Bild in|der gewählten Screen|geht durch den
|Ladevorgang verloren !",0," OKAY | NEIN ",Backalert%
        If Backalert%=2
            Goto No.load
        Endif
        Alert 2," Welches Bildformat :|PRO-PATCH|DEGAS|MAPS&LEGENDS",1,"
PRO |DEGAS| MAPS",Back.alert%
        If Back.alert%=1
            Sc.offset%=0
        Else
            Sc.offset%=46
        Endif
    Endif
Endif

```

```

Start.select:
Fileselect "\*,*","",Selected$
If Selected$<>"\" And Selected$<>".BLO" And Selected$>""

```

'Die Notwendigkeit dieser If-Abfrage wird unter "FILESELECT" erklärt

```

If F.ops%=1
  If Ypos%=1
    Defmouse 2
    E.xtender$="PRO"
    @Extend

```

'E.xtender\$ legt den Extender für die entsprechende Dateiengruppe
'fest. Diese Angabe wird hinterher in der "Procedure Extend"
'verwendet, um falsche oder fehlende Extender-Angaben zu ersetzen,
'bzw. anzufügen (.PRO = Extender für "Graf Pro"-Fullscreens)

```

Put 0,0,Screen.eins$
@Hidemouse
Bsave Selected$,Xbios(2),32000

```

BSAVE

Ein Komfort-Befehl, auf den man sicher immer wieder zurück-
kommen wird. Er bietet die Möglichkeit, eine vorgegebene An-
zahl von RAM-Bytes als ganzen Block auf Diskette zu spei-
chern. Es muß der Name der Datei angegeben werden, in die
der Block abgespeichert werden soll. Nach einem Komma folgt
die RAM-Adresse, in welcher das erste Byte des Blocks steht
und abschließend wieder nach einem Komma die Anzahl der
Bytes, die von dieser Adresse ab gelesen und gespeichert werden
soll. In diesem Fall wird der gesamte sichtbare Bildschirmspei-
cher (Anfangsadresse = Xbios(2)) auf Diskette verfrachtet.

```

@Showmouse
Defmouse 0
Endif
If Ypos%=2
  Defmouse 2

```

```
E.xtender$="PRO"
@Extend
Put 0,0,Screen.zwei$
@Hidemouse
Bsave Selected$,Xbios(2),32000
@showmouse
Defmouse 0
Endif
If Ypos%=3
  Defmouse 2
  E.xtender$="ICN"
  @Extend
  Bsave Selected$,Varptr(Mc.part$),Len(Mc.part$)
  Defmouse 0
Endif
If Ypos%=4
  Defmouse 2
  E.xtender$="PAT"
  @Extend
  Bsave Selected$,Varptr(Patword%(0)),64
  Defmouse 0
Endif
If Ypos%=5
  Endif
Endif
If F.ops%=2
  If Ypos%=1
```

'Hier wird ein PRO bzw. DEGAS/MAPS-Bild für SCREEN EINS geladen

```
@Hidemouse
Put 640,400,Screen.eins$
Cls
Bload Selected$,Xbios(2)+Sc.offset%
Get 0,0,639,399,Screenfix.eins$
Get 0,0,319,199,F.screen$(0)
Get 320,0,639,199,F.screen$(1)
Get 0,200,319,399,F.screen$(2)
Get 320,200,639,399,F.screen$(3)
Screen.eins$=Screenfix.eins$
```

```

@Showmouse
Sc.offset%=0
Endif
If Ypos%=2

```

'Das gleiche für SCREEN ZWEI

```

@Hidemouse
Put 640,400,Screen.zwei$
Cls
Bload Selected$,Xbios(2)+Sc.offset%
Get 0,0,319,199,F.screen$(0)
Get 320,0,639,199,F.screen$(1)
Get 0,200,319,399,F.screen$(2)
Get 320,200,639,399,F.screen$(3)
Get 0,0,639,399,Screenfix.zwei$
Screen.zwei$=Screenfix.zwei$
@Showmouse
Sc.offset%=0
Endif
If Ypos%=3
If Right$(Selected$,4)=".ICN"

```

'In diesem Block werden die mit dem Extender ".ICN" gespeicherten Icons geladen

```

Defmouse 2
@d.size(Selected$)

```

'Dazu wird in "Procedure D.size" die Größe des Icons ermittelt

```

Mc.part$=Space$(Datsize%)

```

'Anschließend wird der String, der das Icon aufnehmen soll, auf diese Größe gebracht...

```

Bload Selected$,Varptr(Mc.part$)

```

'...und das Icon an die Anfangsadresse dieses Strings geladen

```

        Defmouse 0
    Else
        Alert 1,"Mit dieser Funktion können|nur '.ICN'-Dateien gelad
en|werden !",1,"Nochmal|Weiter ",Backalert%
        If Backalert%=1
            Goto Start.select
        Endif
    Endif
Endif
If Ypos%=4
    If Right$(Selected$,4)=".PAT"

```

'Sollen Füllmuster von Diskette geladen werden, müssen erst
'einige Vorbereitungen getroffen werden.

```

        Defmouse 2
        Erase Patword%()
        Dim Patword%(15)

```

'Feld zur Aufnahme der einzelnen Zeilenwerte des Musters dimen-
'sionieren...

```

        Arrayfill Patword%(),1

```

'...die einzelnen Feldelemente mit Einsen auffüllen...

ARRAYFILL

Mit diesem Befehl können sämtliche Elemente eines bestimmten Feldes (Array) mit dem angegebenen Wert gefüllt werden (fill). Es wird der Name des Arrays mit einer Leerklammer angegeben und nach einem Komma der Wert mit dem das Array gefüllt werden soll.

```

Alte Methode :   Dim A_rray%(20,32,16)
                  For A%=0 To 20
                    For B%=0 To 32
                      For C%=0 To 16
                        A_rray%(A%,B%,C%)=237

```

```

                Next C%
            Next B%
        Next A%

GfA _ Methode :   Dim A_rray%(20,32,16)
                  Arrayfill A_rray%(),237

```

Es können selbstverständlich auch Bool'sche- bzw. Realzahlenfelder verwendet werden.

```
Bload Selected$,Varptr(Patword%(0))
```

'...und den Zeilenblock an die Anfangsadresse laden

```

Dpoke Contrl+2,0
Dpoke Contrl+6,16
Dpoke Contrl+12,2
For J=0 To 15
    Dpoke Intin+J*2,Patword%(J)
Next J
Vdisys 112

```

'Zum Verständnis dieser GEM-Funktion s. unter "Procedure
'Patedit" und "GEM"

```

Defmouse 0
Erase Patword%()
Else
    Alert 1,"Mit dieser Funktion können|nur '.PAT'-Dateien gelad
en|werden !",1,"Nochmal|Weiter ",Backalert%
    If Backalert%=1
        Goto Start.select
    Endif
Endif
Endif
If Ypos%=5
Endif
Endif
Endif
Endif

```

No.load:

Return

Procedure Predefs

'Prozedur zur Programmvorbereitung

```
Dim T.extvar(4),T_grow%(1),T_sign%(1)
Dim Pl.nxx(6),Pl.nyy(6),Pl.nx(6),Pl.ny(6),F.screen$(3),D.mode$(25,3),Xma
rk(1),Ymark(1) 'Achtung: Fehlende Zeilen siehe Anhang E
@Clipprec(2)
Get 0,0,639,399,Screenfix.eins$
Get 0,0,639,399,Screenfix.zwei$
Get 0,0,639,399,Screen.eins$
Get 0,0,639,399,Screen.zwei$
Restore Modetext
For I=1 To 25
  For J=1 To 3
    Read D.mode$(I,J)
  Next J
Next I
T.extvar(1)=1
T.extvar(2)=0
T.extvar(3)=0
T.extvar(4)=13
Schrittweite%=1
Rastermod%=3
Lncol%=1
D.rawmode%=1
Color 1
Defline 0,1,0,0
Fmuster%=0
Fstil%=0
Lndick%=1
Lnstil%=1
C.lipp%=2
Grafmod%=1
Lnflag%=1
Mc.done%=1
Return
```

'Ab hier folgen die 25 Prozeduren, mit denen die einzelnen Grafik-
 'Elemente
 '(Quadrat,Kreis,Rechtecke,Linien,Filling,Spray etc.) ausgegeben werden.
 'Alle diese Prozeduren werden von der "Procedure Verteiler" aus aufge-
 'rufen.

Procedure F.srecp

'F.srecp = Filled sharp Rectangle Patchmode
 ' = gefülltes scharfeckiges Rechteck im Patchmodus

Graphmode 3

@D.efpatch

'In der Prozedur "D.efpatch" werden entsprechend der Mauskoordinate
 'im ersten Grafik-Element die Polyline-Koordinaten für eine saubere
 'Umrandung der Pbox (auch bei dickeren Linien) bestimmt.

Defline Lnstil%,Lndick%,0,0
 Pbox X-Halbe%,Y-Halbe%,X+Halbe%,Y+Halbe%

'Die Variable "Halbe%" wird eingesetzt, um den Mauszeiger immer in der
 'Mitte der Box zu halten. Sonst wäre der Mauszeiger in der linken oberen
 'Ecke der Box, was bei Benutzung der Spiegelfunktion die Symmetrie des
 'Bildes zerstören würde.

Polyline 6,Plnx(),Plny()
 Pbox X-Halbe%,Y-Halbe%,X+Halbe%,Y+Halbe%
 Polyline 6,Plnx(),Plny()
 Graphmode Grafmod%
 If K=1
 Pbox X-Halbe%,Y-Halbe%,X+Halbe%,Y+Halbe%
 Polyline 6,Plnx(),Plny()
 Endif
 Return

POLYLINE, POLYFILL, POLYMARK

Wer sich für 3D-Programmierung interessiert, wird von diesen Befehlen begeistert sein. Als Vorbereitung zu diesen Befehlen sind zwei eindimensionale numerische Variablenfelder zu dimensionieren und die Elemente zu bestimmen. Die Anzahl der Feldelemente richtet sich nach der Menge an Punkten, die hiermit angesprochen werden sollen. In das eine Feld werden die X-Werte und in das andere Feld die Y-Werte der Punktkoordinaten eingetragen.

Die Befehle haben folgende Funktionen:

- | | | |
|----------|---|--|
| Polyline | = | Es werden die vordefinierten Punktkoordinaten mit Linien in der eingestellten Linienart verbunden. |
| Polyfill | = | Es wird Polyline ausgeführt und die zwischen den Linien eingegrenzten Bereiche mit dem eingestellten Muster gefüllt. |
| Polymark | = | Die definierten Punkte werden mit dem eingestellten Symbol markiert. |

Alle drei Befehle verwenden die Einstellungen, die mit "Defline", "Deffill" oder "Defmark" vorgenommen wurden. Der erste zu definierende Punkt ist im Null-Element der Felder abzulegen, da die beiden Inhalte dieser Elemente als Koordinatenwerte des ersten Punktes verwendet werden. Maximal dürfen 128 Punkte angegeben werden. Soll ein geschlossener Linienzug gezeichnet werden, muß der letzte Punkt mit dem ersten identisch sein. Um sich nicht jedesmal die Arbeit machen zu müssen, die Koordinatenpaare neu zu bestimmen, wenn dieselbe Figur an einer anderen Stelle gezeichnet werden soll, kann mit dem Zusatzbefehl "Offset", der dann diesen Befehlen angehängt wird, in X- und in Y-Richtung ein Versatz der Figur um einen bestimmten Betrag (negativ oder positiv) bewirkt werden. Zur Verdeutlichung der Funktion dieser Befehle, ihrer Syntax und Wirkung folgt mal wieder ein kleines Beispielprogramm.

```

Dim X(5),Y(5)
Graphmode 3
Do
  For I=0 To 4
    Aa=Int(Rnd*5)+1
    Bb=Int(Rnd*24)
    Cc=Random(5)+1
    Dd=Random(2)
    Ee=Random(2)
    X(I)=Random(400)+120
    Y(I)=Random(260)+70
  Next I
  X(I)=X(0)
  Y(I)=Y(0)
  Defmark ,Aa,80
  Deffill 1,2,Bb
  Defline Cc,1,Dd,Ee
  Polymark 5,X(),Y()
  Polyfill 5,X(),Y()
  For J%=1 To 2
    For I%=3 To 90 Step 3
      Polyline 5,X(),Y() Offset I%,I%
    Next I%
    For I%=90 To 3 Step -3
      Polyline 5,X(),Y() Offset I%,I%
    Next I%
  Next J%
  Polymark 5,X(),Y()
  Polyfill 5,X(),Y()
  Exit If Mousek
Loop
Graphmode 1
Edit

```

Procedure L.srecp

'L.srecp = Line sharp Rectangle Patchmode
 = Nur-Linie scharfeckiges Rechteck im Patchmodus

```

Graphmode 3
@D.efpatch
Defline Lnstil%,Lndick%,0,0
Polyline 6,Plnx(),Plny()
Polyline 6,Plnx(),Plny()
Graphmode Grafmod%
If K=1
    Polyline 6,Plnx(),Plny()
Endif
Return
Procedure A.srecp

```

'A.srecp = Area sharp Rectangle Patchmode
 = Nur-Fläche scharfkantiges Rechteck ohne Rahmen im Patchmodus

```

Graphmode 3
Pbox X-Halbe%,Y-Halbe%,X+Halbe%,Y+Halbe%
Pbox X-Halbe%,Y-Halbe%,X+Halbe%,Y+Halbe%
Graphmode Grafmod%
If K=1
    @Rahmen(0)

```

'In der "Procedure Rahmen" kann der Rahmen der Funktionen "Pbox",
 "'Prbox", "Pcircle" und "Pellipse" über eine GEM-Funktion ein- und
 'ausgeschaltet werden (0 = aus ; 1 = an).

```

    Pbox X-Halbe%,Y-Halbe%,X+Halbe%,Y+Halbe%
    @Rahmen(1)
Endif
Return
Procedure F.rrecp

```

'F.rrecp = Filled round Rectangle Patchmode
 = gefülltes rundkantiges Rechteck im Patchmodus

```

Graphmode 3
Prbox X-Halbe%,Y-Halbe%,X+Halbe%,Y+Halbe%
Prbox X-Halbe%,Y-Halbe%,X+Halbe%,Y+Halbe%

```

PRBOX

Dieser Befehl ist exakt genauso zu behandeln wie "Pbox", nur das die hiermit dargestellten Rechtecke abgerundete Ecken haben.

```

Graphmode Grafmod%
If K=1
  Prbox X-Halbe%,Y-Halbe%,X+Halbe%,Y+Halbe%
Endif
Return
Procedure L.rrecp

```

'Line round Rectangle Patchmode
'Nur-Linie rundkantiges Rechteck im Patchmodus

```

Graphmode 3
Define Lnstil%,Lndick%,0,0
Rbox X-Halbe%,Y-Halbe%,X+Halbe%,Y+Halbe%
Rbox X-Halbe%,Y-Halbe%,X+Halbe%,Y+Halbe%
Graphmode Grafmod%
If K=1
  Rbox X-Halbe%,Y-Halbe%,X+Halbe%,Y+Halbe%

```

RBOX

Syntax und Funktion ist identisch mit "Pbox","Box" oder "Prbox". Es wird ein rundeckiger Rahmen erzeugt.

```

Endif
Return
Procedure A.rrecp

```

'A.rrecp = Area round Rectangle Patchmode
' = Nur-Fläche rundkantiges Rechteck ohne Rahmen im Patchmodus

```

Graphmode 3
Prbox X-Halbe%,Y-Halbe%,X+Halbe%,Y+Halbe%
Prbox X-Halbe%,Y-Halbe%,X+Halbe%,Y+Halbe%

```

```

Graphmode Grafmod%
If K=1
  @Rahmen(0)
  Prbox X-Halbe%,Y-Halbe%,X+Halbe%,Y+Halbe%
  @Rahmen(1)
Endif
Return
Procedure F.cp

'F.cp = Filled Circle Patchmodus
'      = gefüllter Kreis im Patchmodus

Graphmode 3
Pcircle X,Y,Halbe%
Pcircle X,Y,Halbe%

```

PCIRCLE

Mit diesem Befehl kann ein mit dem eingestellten Füllmuster ausgefüllter Kreis erzeugt werden.

Die drei Parameter haben folgende Bedeutung:

1. Parameter = X-Koordinate des Kreismittelpunktes
2. Parameter = Y-Koordinate des Kreismittelpunktes
3. Parameter = Radius des Kreises

```

Graphmode Grafmod%
If K=1
  Pcircle X,Y,Halbe%
Endif
Return
Procedure L.cp

'L.cp = Line Circle Patchmode
'      = Nur-Linie Kreis im Patchmodus

Graphmode 3
Defline Lnstil%,Lndick%,0,0

```

```
Circle X,Y,Halbe%
```

```
Circle X,Y,Halbe%
```

CIRCLE

"Circle" ist identisch mit "Pcircle". Der Kreis besteht jedoch nur aus der umfassenden Linie ohne Musterfüllung.

```
Graphmode Grafmod%
```

```
If K=1
```

```
    Circle X,Y,Halbe%
```

```
Endif
```

```
Return
```

```
Procedure A.cp
```

```
'A.cp = Area Circle Patchmode
```

```
= Nur-Fläche Kreis ohne Rahmen im Patchmodus
```

```
Graphmode 3
```

```
Pcircle X,Y,Halbe%
```

```
Pcircle X,Y,Halbe%
```

```
Graphmode Grafmod%
```

```
If K=1
```

```
    @Rahmen(0)
```

```
    Pcircle X,Y,Halbe%
```

```
    @Rahmen(1)
```

```
Endif
```

```
Return
```

```
Procedure Iconp
```

'In dieser Prozedur werden die selbstdefinierten Icons zum Einsatz gebracht. Diese werden grundsätzlich in der Stringvariablen "Mc.Part\$" gespeichert. Auch die mit MOVE/COPY markierten Bildausschnitte werden in dieser Variablen abgelegt. Das führt dazu, daß mit diesen Ausschnitten im "SELF"-Modus auch gezeichnet werden kann. Es wird mit einer der beiden Blockfunktionen einfach ein Bildausschnitt markiert, anschließend die gewählte Blockfunktion wieder ausgeschaltet und im Grafik-Menü "SELF" angewählt. Dieser zuletzt definierte Block erscheint jetzt blinkend auf dem Bildschirm. Nun kann er mittels der linken Maustaste

'gesetzt, bzw. mittels der rechten Maustaste bewegt werden, wobei die
'Relation des Mauszeigers zum Icon während des Tastendrucks beibehalten
'wird. Sollte es passieren, daß Sie das Icon aus dem Bildschirmbereich
'hinausbewegt haben und nicht mehr wiederfinden, brauchen Sie nur erneut
'"SELF" im Grafikmenü anwählen und es erscheint anschließend wieder in
'der linken oberen Bildschirmecke.

'Die im Icon-Editor entworfenen Icons, sowie auch die mit der "BLOCK-
'LOAD"-Funktion im "Spezi"-Menü geladenen Ausschnitte werden ebenfalls
'in "Mc.part\$" abgelegt. Somit dürfte es kein Problem sein, sich ganze
'Bibliotheken an Bildausschnitten und Icons anzulegen. Damit wäre sogar
'ein Zeichensatz in beliebiger Größe realisierbar. Die einzelnen Zeichen
'müßten dann mit dem Icon-Editor entworfen, und mit der "Sv"-Funktion
'des Editors auf Diskette abgespeichert werden. Anschließend lassen
'sich dann diese Zeichen mit der "BLOCK-LOAD"-Funktion laden und mit
'der "SELF"-Grafikfunktion auf den Bildschirm bringen. Und das dann zu-
'sätzlich noch in allen 4 Graphmodes oder durch die Shift-Pfeiltasten-
'Kombinationen bzw. den Pfeiltasten-Funktionen unter "COPY" oder "MOVE"
'vergrößert, verkleinert, in beide Dimensionen gespiegelt, oder als
'"Tonne" bzw. "Welle" dargestellt.

'Ihrer Phantasie sind hier "fast" keine Grenzen gesetzt. Lassen sie ihr
'den nötigen Auslauf!

Mouse In.x%,In.y%,K

'Um die Icon-Koordinaten unabhängig von der Mausposition zu
'machen und zur
'Kenntlichmachung innerhalb der Prozedur, werden hier für
'die Mauskoordinaten
'spezielle Variablen definiert. Diese Variablen beinhalten immer
'die aktuelle Mausposition.

If K=0

'Dieser Teil wird nur durchlaufen, wenn keine Maustaste gedrückt wird.

Put Icnx%,Icny%,Mc.part\$,6

Put Icnx%,Icny%,Mc.part\$,6

'Die Variablen "Icnx%" und "Icny%" stehen beim Einschalten der "SELF"-
'Funktion grundsätzlich auf Null (s. "Procedure Setgrafmode"). An die

'darin enthaltene Position wird das Icon mit der linken oberen Ecke
'angelegt, wenn eine der beiden Maustasten gedrückt wird

```
Mouse X.fix%,Y.fix%,Button%
```

'Um zu wissen, wo sich die Maus zur Zeit eines Tastendrucks befunden
'hat, werden hier weitere Koordinatenvariablen eingesetzt. Da diese
'nur verändert werden, wenn keine Maustaste gedrückt wird, bleiben
'sie also konstant, während eine Maustaste gedrückt wird. Diese fixier-
'te Position wird benötigt, um das Offset (Versatz) der Maus zum Icon
'feststellen zu können.

```
If Netzmod%=1
```

'Ist der Netzmodus aktiv, werden hier die Rasterpunkte entsprechend
'der eingestellten Netzweite berechnet und die Koordinatenvariablen
'entsprechend angepaßt.

```
X.fix%=Int(X.fix%/Netzweite%)*Netzweite%
```

```
Y.fix%=Int(Y.fix%/Netzweite%)*Netzweite%
```

```
Endif
```

```
X.diff%=X.fix%-Icnx%
```

```
Y.diff%=Y.fix%-Icny%
```

'Hier werden die Differenzen der Icon- zu den Mauskoordinaten für beide
'Dimensionen in speziellen Variablen festgehalten.

```
Endif
```

```
If K=1
```

'Hier kommt das Programm nur hin, wenn die linke Maustaste gedrückt,
'also gezeichnet wird.

```
Put X-X.diff%,Y-Y.diff%,Mc.part$,Rastermod%
```

Setze das Icon im eingestellten Graphmode (der für Rasteroperationen
'in "Procedure Changemod" zu "Rastermod%" umgewandelt wird) an die
'Position, die sich aus der aktuellen Mausposition minus der vorher
'berechneten Differenz ergibt. Die Koordinatenvariablen "X" und "Y"
'werden hier schon in der "Procedure Verteiler" gesetzt.

Icnx%=In.x%-X.diff%

Icny%=In.y%-Y.diff%

'Anschließend merke Dir in "Icnx%" und "Icny%" die zuletzt gesetzte
'Icon-Position. An dieser Position verharrt das Icon flimmernd, falls
'keine Maustaste gedrückt wird. Diese Koordinate wird zur Berechnung
'der Differenz von Iconposition zu der aktuellen Mausposition verwen-
'det, wenn keine Taste gedrückt wird (s.o.).

Endif

If K=2

'Dieser Block findet nur Verwendung, wenn die rechte Maustaste ge-
'drückt wird, also das Icon ohne zu zeichnen über den Bildschirm be-
'wegt werden soll.

Put In.x%-X.diff%,In.y%-Y.diff%,Mc.part\$,6

Put In.x%-X.diff%,In.y%-Y.diff%,Mc.part\$,6

Icnx%=In.x%-X.diff%

Icny%=In.y%-Y.diff%

'Hier findet das gleiche Spiel statt, als wenn die linke Maustaste
'gedrückt wird. Nur daß hier das Icon im XOR-Modus zweimal auf die-
'selbe Stelle gesetzt wird (Schwebe-Effekt).

Endif

Return

Procedure Rubbgoing

'In dieser kleinen Prozedur wird für die Rubbmode-Grafikelemente fest-
'gestellt, ob ein spezielles Spiegelflag ("Mirflag%") gesetzt ist.
'Wurde es nicht gesetzt, bedeutet es, daß bereits eine Rubb-Figur im
'Spiegelmodus definiert wurde. Ist dieses Flag dagegen gesetzt, heißt
'das, daß die Spiegeloperationen für diese einzelne Figur abgeschlossen
'sind. Diese Information wird benötigt, weil die Spiegel-Funktion im
'"Verteiler" gesteuert wird. Sonst würde das Programm bei jedem zu
'zeichnenden Spiegelbild der definierten Rubb-Figur wieder die Eingabe
'einer neuen Figur erwarten.

```
If Mirflag%=1
```

```
  @Rubbself
```

'Wurde also dieses Flag gesetzt (Spiegelungen sind vollbracht, falls
'der Spiegelmodus überhaupt eingeschaltet war), gehe zur "Procedure
'Rubbself", wo die Rubb-Figuren definiert werden.

```
Endif
```

```
Return
```

```
Procedure F.srecre
```

```
'F.srecre = Filled sharp Rectangle Rubbmode
```

```
'      = gefülltes scharfkantiges Rechteck im Rubbmodus
```

```
  @Rubbgoing
```

'Diesen Prozedur-Aufruf finden Sie in allen "Rubb"-Grafik-Routinen.
'Seine Bedeutung wurde unter "Procedure Rubbgoing" beschrieben.

```
  @D.ef.patch.2
```

'Dieser Prozedur-Aufruf erfüllt denselben Zweck für Rubb-Figuren wie
'"@D.efpatch" in "Procedure F.srecre". Beschreibung s. dort.

```
If Rubbflag%=1 And Norubb%=0
```

'Die Abfrage dieser beiden Variablen finden Sie ebenfalls in allen
'"Rubb"-Grafik-Routinen.

'Wenn eines der Rubb-Elemente aufgerufen wurde und anschließend
'"MOVE" oder "COPY" aktiviert wird, wird hier analysiert, ob nun
'gerade eine Rubb-Figur in "Procedure Rubbself" definiert wurde, oder
'ein MOVE/COPY-Ausschnitt. Beide Funktionen werden nämlich durch diese
'Prozedur bedient. "Rubbflag%" wird in der "Procedure Rubbself" gesetzt,
'bzw. nicht gesetzt, um dies auseinanderhalten zu können.
'Ist "Rubbflag%" = 0, heißt das, daß ein Block-Ausschnitt in "Rubbself"
'markiert wurde und deshalb keine Rubb-Figur auf diesen Ausschnitt
'gezeichnet werden darf.

'In der Variablen "Norubb%" wird abgelegt, ob während der Definition
'in "Rubbself" zusätzlich die rechte Maustaste gedrückt wurde. Dies
'bedeutet, daß die soweit schon geöffnete Markierungsbox wieder ungül-

'tig gemacht, also gelöscht werden soll. Wurde zusätzlich die rechte
'Maustaste in "Rubbself" nicht gedrückt, ist "Norubb%" = 0 und die
'Rubb-Figur kann nun hiermit gezeichnet werden.

```

    Defline ,,0,0
    Rubbflag%=0
    Pbox Xl%,Yl%,Xx%,Yy%
    Polyline 6,Pl.nxx(),Pl.nyy()
    Defline ,,Lnstart%,Lnende%
Endif
Return
Procedure L.srecr

'L.srecr = Line sharp Rectangle Rubbmode
'          = Nur-Linie scharfkantiges Rechteck im Rubbmodus

@Rubbgoing
@D.ef.patch.2
If Rubbflag%=1 And Norubb%=0
    Defline ,,0,0
    Rubbflag%=0
    Box Xl%,Yl%,Xx%,Yy%
    Polyline 6,Pl.nxx(),Pl.nyy()
    Defline ,,Lnstart%,Lnende%
Endif
Return
Procedure A.srecr

'A.srecr = Area sharp Rectangle Rubbmode
'          = Nur-Fläche scharfkantiges Rechteck ohne Rahmen im Rubbmodus

@Rubbgoing
If Rubbflag%=1 And Norubb%=0
    Rubbflag%=0
    @Rahmen(0)
    Pbox Xl%,Yl%,Xx%,Yy%
    @Rahmen(1)
Endif
Return
Procedure F.rrecr

```

'F.rrecr = Filled round Retangle Rubbmode

' = gefülltes rundkantiges Rechteck im Rubbmodus

@Rubbgoing

If Rubbflag%=1 And Norubb%=0

Rubbflag%=0

Prbox Xl%,Yl%,Xx%,Yy%

Endif

Return

Procedure L.rrecr

'L.rrecr = Line round Rectangle Rubbmode

' = Nur-Linie rundkantiges Rechteck im Rubbmodus

@Rubbgoing

If Rubbflag%=1 And Norubb%=0

Rubbflag%=0

Rbox Xl%,Yl%,Xx%,Yy%

Endif

Return

Procedure A.rrecr

'A.rrecr = Area round Rectangle Rubbmode

' = Nur-Fläche rundkantiges Rechteck ohne Rahmen im Rubbmodus

@Rubbgoing

If Rubbflag%=1 And Norubb%=0

Rubbflag%=0

@Rahmen(0)

Prbox Xl%,Yl%,Xx%,Yy%

@Rahmen(1)

Endif

Return

Procedure F.cr

'F.cr = Filled Circle Rubbmode

' = gefüllter Kreis (bzw. Ellipse) im Rubbmodus

```

@Rubbing
If Rubbflag%=1 And Norubb%=0
    Rubbflag%=0
    Pellipse Xx%,Yy%,Xrad%,Yrad%

```

PELLIPSE

Dieser Befehl erzeugt eine mit dem eingestellten Füllmuster ausgefüllte Ellipse.

Die vier Parameter haben folgende Bedeutung:

1. Parameter = X-Koordinate des Ellipsen-Mittelpunktes
2. Parameter = Y-Koordinate des Ellipsen-Mittelpunktes
3. Parameter = Ellipsen-Durchmesser in X-Richtung
4. Parameter = Ellipsen-Durchmesser in Y-Richtung

```

Endif
Return
Procedure L.cr

```

```

'L.cr = Line Circle Rubbmode
'      = Nur-Linie Kreis (bzw. Ellipse) im Rubbmodus

```

```

@Rubbing
If Rubbflag%=1 And Norubb%=0
    Rubbflag%=0
    Ellipse Xx%,Yy%,Xrad%,Yrad%

```

ELLIPSE

"Ellipse" hat die gleiche Funktion wie "Pellipse", nur daß die hiermit erzeugte Ellipse nicht mit dem Füllmuster gefüllt wird.

```

Endif
Return
Procedure A.cr

```

```

'A.cr = Area Circle Rubbmode
'      = Nur-Fläche Kreis (bzw. Ellipse) ohne Rahmen im Rubbmodus

```

```

@Rubbgoing
If Rubbflag%=1 And Norubb%=0
    Rubbflag%=0
    @Rahmen(0)
    Pellipse Xx%,Yy%,Xrad%,Yrad%
    @Rahmen(1)
Endif
Return
Procedure S.lin

'S.lin = Straight Line
'      = gerade Linie
'Diese Bezeichnung mag etwas irreführend klingen. Trotzdem sind auch
' einzelne Punkte in diesem Fall für den Computer gerade Linien. Bei
' einzelnen Punkten sind die Start- und Endkoordinate einer Linie iden-
' tisch. Selbst eine schräg verlaufende Linie setzt sich aus vielen
' kleinen waagerecht verlaufenden Linien zusammen.

Mouse X,Y,K
If Netzmod%=1
    X=Int(X/Netzweite%)*Netzweite%
    Y=Int(Y/Netzweite%)*Netzweite%
Endif
If K=1

'Falls die linke Maustaste gedrückt wird...

    Graphmode Grafmod%
    If Mirror%=1

'...und Vertikal-Spiegel aktiv...

        Xm.1%=639-X

'...setze eine Hilfsvariable mit der vertikal gegenüberliegenden
'X-Position

        Endif
        If Mirror%=2

```

'...und Horizontal-Spiegel aktiv

Ym.1%=399-Y

'...setze eine Hilfsvariable mit der horizontal gegenüberliegenden
'Y-Position

Endif

If Mirror%=3 Or Mirror%=4

'...und Diagonal-Spiegel oder alle drei Spiegelachsen aktiv...

Xm.1%=639-X

Ym.1%=399-Y

'...setze beide Hilfsvariablen mit den jeweils gegenüberliegenden
'Positionen

Endif

Do

'Bleibe solange in dieser Freihandlinien-Schleife bis die linke
'Mautaste wieder freigegeben wird.

Mouse Xx.2%,Yy.2%,Lbutt%

If Netzmod%=1

Xx.2%=Int(Xx.2%/Netzweite%)*Netzweite%

Yy.2%=Int(Yy.2%/Netzweite%)*Netzweite%

Endif

Line X,Y,Xx.2%,Yy.2%

'Zeichne eine Linie von der letzten Mausposition zur neuen Maus-
'position.

If Mirror%<3

'Um das Linienzeichnen nicht unnötig zu verzögern, wurde hier die
'Abfrage, ob gespiegelt werden soll, in zwei Blöcke geteilt. Sonst
'müßten bei jedem Durchlauf alle vier Abfragen beantwortet werden.

'Außerdem wird ein "Goto" zum Schleifenende angebracht, damit nach
'vollbrachter Spiegelung nicht der Rest der Schleife auch noch
'absolviert werden muß. Beides hat nur den Sinn, Zeit zu sparen.

```

      If Mirror%=1
        @Mir.1.line.1
        Goto N.extpt
      Endif
      If Mirror%=2
        @Mir.2.line.1
        Goto N.extpt
      Endif
    Else
      If Mirror%=3
        @Mir.3.line.1
        Goto N.extpt
      Endif
      If Mirror%=4
        @Mir.4.line.1
      Endif
    Endif
    N.extpt:
    X=Xx.2%
    Y=Yy.2%

```

'Solange sich das Programm in dieser Schleife befindet, wird die
'letzte Mausposition hier festgehalten, um der nächsten Linie als
'Startpunkt zu dienen (s.Schleifenanfang).

```

      Exit If Lbutt%=0
    Loop
  Endif
  If K=2

```

'Falls die rechte Maustaste gedrückt wurde...

```

      Mouse Xx.x%,Yy.y%,Button%

```

'...fixiere die Startkoordinate


```
If Netzmod%=1

'Falls Netzmodus eingeschaltet...

    Xx.x%=Int(Xx.x%/Netzweite%)*Netzweite%
    Yy.y%=Int(Yy.y%/Netzweite%)*Netzweite%

'...ermittle die Rasterkoordinaten...

    If Mirror%=1
        Xm.1%=Xx.x%
    Endif
    If Mirror%=2
        Ym.1%=Yy.y%
    Endif
    If Mirror%=3 Or Mirror%=4
        Xm.1%=Xx.x%
        Ym.1%=Yy.y%
    Endif

'...und passe die Spiegelkoordinaten, falls notwendig, an.

    Endif
    Deftext 1,0,0,6
    Graphmode 3
    Do

'In dieser Schleife bleibt das Programm solange, bis nicht mehr
'nur die rechte Maustaste gedrückt wird oder eine Taste auf der
'Tastatur (evtl.Menüaufruf) betätigt wird.

        Mouse Xx.1%,Yy.1%,Button%

'Merke Dir die aktuelle Mauskoordinate (als Linienendpunkt)...

    If Netzmod%=1

'Falls Netzmodus an...

        Xx.1%=Int(Xx.1%/Netzweite%)*Netzweite%
```

```
Yy.1%=Int(Yy.1%/Netzweite%)*Netzweite%
```

```
'...passe auch diese Koordinaten an
```

```
Endif
```

```
Line Xx.x%,Yy.y%,Xx.1%,Yy.1%
```

```
Line Xx.x%,Yy.y%,Xx.1%,Yy.1%
```

```
'Zeichne im Graphmode 3 zweimal hintereinander die Linie (schweben)
```

```
If Showkoordi=1
```

```
'Falls im "Border On"-Modus die rechte untere Bildschirmecke
```

```
'berührt, und anschließend durch zweimaliges Klicken angezeigt
```

```
'wurde, daß ab nun die horizontalen, vertikalen und diagonalen
```

```
'Punkteabstände angezeigt werden sollen...
```

```
L.o.h%=Abs(Xx.1%-Xx.x%)
```

```
L.o.v%=Abs(Yy.1%-Yy.y%)
```

```
L.o.d%=Sqr(L.o.h%^2+L.o.v%^2)
```

```
'Wurzel aus (L.o.h (Length of horizontal) zum Quadrat + L.o.v
```

```
'(Length of vertikal) zum Quadrat) ergibt L.o.d (Length of diago-
```

```
'nal) ==> "Satz des Phytagoras"
```

SQR

Diese arithmetische Funktion gibt die Quadratwurzel (Square Radical) des in Klammern nachgestellten mathematischen Ausdrucks zurück. Der Ausdruck kann als Variable oder als komplexer Ausdruck angegeben werden.

ABS

Eine weitere arithmetische Funktion soll in diesem Zusammenhang erläutert werden. "Abs" liefert den Absolutwert des in Klammern als Variable oder mathematischen Ausdruck enthalte-

nen Betrages (Betragsfunktion). D.h., daß das Vorzeichen des angegebenen Wertes ignoriert wird. Es wird der reine Betrag des Wertes ermittelt. Der ist in jedem Fall gleich oder größer Null.

z.B.:

A=-123

B=334

Print Abs(A)''Abs(B)''Abs(65*(-0.32))

Ausgabe: 123 334 20.8

SGN

Mit "Sgn" läßt sich in Erfahrung bringen, ob eine bestimmte Zahl ein negatives oder positives Vorzeichen besitzt, bzw. ob die sie null ist. Der Funktion wird in Klammern entweder eine numerische Variable, ein mathematischer Ausdruck oder eine Zahl übergeben, deren Vorzeichen (SiGNum) ermittelt werden soll. Bei negativem Vorzeichen wird der Wert -1, bei positivem Vorzeichen der Wert 1 und bei Nullwerten der Wert 0 zurückgegeben.

z.B.:

A=-328.77

Print Sgn(A)''Sgn(A*(-1))''Sgn(A/A-1)

Ausgabe: -1 1 0

EXP

Dieser Funktion wird ein beliebiger numerischer Wert oder Ausdruck im Bereich von -354 bis +354 übergeben, der dann als Exponent für die Euler'sche Zahl (2.718281828) verwendet wird.

z.B.:

```
Print Exp(Sqr(22^3.5722*(317.66*22.7))+Sin(125.44))'';
Print Exp(23.744)'';
A=6^3/3.55
Print Exp(A)
```

Ausgabe: 90910245797E+101 20506341156 26587550967E+26

L.o.l\$="X:"+Str\$(L.o.h%)+ " Y:"+Str\$(L.o.v%)+ " D:"+Str\$(L.o.d%)

'...errechne die Abstände, faße sie zu einem String zusammen...

```
Text Xx.1%-Int(Xx.1%/4.5),Yy.1%-Int((Yy.1%-200)/15),L.o.l$
Text Xx.1%-Int(Xx.1%/4.5),Yy.1%-Int((Yy.1%-200)/15),L.o.l$
```

'...und gebe sie an der Mausposition aus. Und zwar so, daß der
'String auf jeden Fall komplett innerhalb des Bildschirmes liegt.

```
Endif
If Mirror%=1
  @Mir.1.line.2
  Goto N.extre
Endif
If Mirror%=2
  @Mir.2.line.2
  Goto N.extre
Endif
If Mirror%=3
  @Mir.3.line.2
  Goto N.extre
Endif
If Mirror%=4
  @Mir.1.line.2
  @Mir.2.line.2
  @Mir.3.line.2
Endif
```

'Falls notwendig, gehe zu den einzelnen Spiegelroutinen

@Getfkeys

'Frage in "Getfkeys" nach, ob eine Taste gedrückt wurde

```

Graphmode 3
Exit If Button%<>2 Or Fkeys%>0
Loop
Graphmode Grafmod%
If Button%=3

```

'Falls zusätzlich die rechte Maustaste gedrückt wurde (also beide
'Tasten gleichzeitig), mache die Liniendarstellung wieder rückgängig

```

Line Xx.x%,Yy.y%,Xx.1%,Yy.1%
If Mirror%=1
    Line Xm.1%,Yy.y%,Xm.2%,Yy.1%
Endif
If Mirror%=2
    Line Xx.x%,Ym.1%,Xx.1%,Ym.2%
Endif
If Mirror%=3
    Line Xm.1%,Ym.1%,Xm.2%,Ym.2%
Endif
If Mirror%=4
    Line Xm.1%,Yy.y%,Xm.2%,Yy.1%
    Line Xx.x%,Ym.1%,Xx.1%,Ym.2%
    Line Xm.1%,Ym.1%,Xm.2%,Ym.2%
Endif
Endif
Endif
Return

```

'Die nächsten acht Routinen dienen allein dazu, die gespiegelte Linie
'entsprechend der eingestellten Spiegelachse(n) zu zeichnen

```

Procedure Mir.1.line.1
Xm.2%=639-Xx.2%
Line Xm.1%,Y,Xm.2%,Yy.2%
Xm.1%=Xm.2%

```

Return

Procedure Mir.2.line.1

$Ym.2\% = 399 - Yy.2\%$

Line X, Ym.1%, Xx.2%, Ym.2%

$Ym.1\% = Ym.2\%$

Return

Procedure Mir.3.line.1

$Xm.2\% = 639 - Xx.2\%$

$Ym.2\% = 399 - Yy.2\%$

Line Xm.1%, Ym.1%, Xm.2%, Ym.2%

$Xm.1\% = Xm.2\%$

$Ym.1\% = Ym.2\%$

Return

Procedure Mir.4.line.1

$Xm.2\% = 639 - Xx.2\%$

$Ym.2\% = 399 - Yy.2\%$

Line Xm.1%, Y, Xm.2%, Yy.2%

Line X, Ym.1%, Xx.2%, Ym.2%

Line Xm.1%, Ym.1%, Xm.2%, Ym.2%

$Xm.1\% = Xm.2\%$

$Ym.1\% = Ym.2\%$

Return

Procedure Mir.1.line.2

$Xm.1\% = 639 - Xx.x\%$

$Xm.2\% = 639 - Xx.1\%$

Line Xm.1%, Yy.y%, Xm.2%, Yy.1%

Line Xm.1%, Yy.y%, Xm.2%, Yy.1%

Return

Procedure Mir.2.line.2

$Ym.1\% = 399 - Yy.y\%$

$Ym.2\% = 399 - Yy.1\%$

Line Xx.x%, Ym.1%, Xx.1%, Ym.2%

Line Xx.x%, Ym.1%, Xx.1%, Ym.2%

Return

Procedure Mir.3.line.2

$Xm.1\% = 639 - Xx.x\%$

$Xm.2\% = 639 - Xx.1\%$

$Ym.1\% = 399 - Yy.y\%$

$Ym.2\% = 399 - Yy.1\%$

Line Xm.1%, Ym.1%, Xm.2%, Ym.2%

Line Xm.1%,Ym.1%,Xm.2%,Ym.2%

Return

Procedure F.fillpro

'Dies ist die Fill-Prozedur

Mouse X,Y,K

If K=1

'Falls die linke Maustaste gedrückt wurde, fülle an der aktuellen
'Mausposition

Fill X,Y

If Mirror%=1

'Falls eine Spiegelfunktion eingeschaltet ist, errechne die weiteren
'Füllkoordinaten und fülle dort ebenfalls.

'Aus dem unter "FILL" beschriebenen Grund (Dauer des Füllens bei
'"offenen" Füllmustern) ist das gespiegelte Füllen mit Vorsicht
'anzuwenden, da z.B. bei der Drei-Achsen-Spiegelung auf jeden Fall
'vier Mal gefüllt wird. Dies läßt sich nur durch "Control/Shift/
'Alternate" nach jedem abgeschlossenen Füllprozess unterbrechen.
'Dadurch wird dann allerdings das Programm beendet und das/die
'aktuelle(n) Bild(er) geht/gehen dadurch verloren.

Fill 639-X,Y

Endif

If Mirror%=2

Fill X,399-Y

Endif

If Mirror%=3

Fill 639-X,399-Y

Endif

If Mirror%=4

Fill 639-X,Y

Fill X,399-Y

Fill 639-X,399-Y

Endif

Endif

Return

Procedure Pl.spr

'Pl.spr = Plot Spray

' = Einzelpunkte sprühen

'Mit dieser Prozedur wird einer der beiden Sprüh-Modi ausgeführt.

'Unter "Plot-Spray" ist zu verstehen, daß auf der eingestellten Patch-

'fläche schwarze Einzelpunkte gesprüht werden. Die Patchfläche ist der

'aufblinkende Rahmen, der während des Sprühens dargestellt wird. Die

'Größe dieses Patches (patch;engl.:Flicken) kann wie bei den Patch-Funk-

'tionen durch die Tasten F7 und F8 oder im Grafik-Menü verändert werden.

'D.h., daß bei entsprechender Patchgröße der gesamte Bildschirm gleich-

'zeitig besprüht werden kann. Außerdem kann durch die drei möglichen

'Maustastenvariationen (links, rechts, beide) die Dicke der Sprühpunkte

'dreifach variiert werden.

Graphmode 3

Defline linstil%,lndik,0,0

Box X-Halbe%,Y-Halbe%,X+Halbe%,Y+Halbe%

Box X-Halbe%,Y-Halbe%,X+Halbe%,Y+Halbe%

'Hier wird der Patch-Rahmen dargestellt

Graphmode Grafmod%

If K>0

 If K=1

 Pcircle X-Halbe%+Spr.x,Y-Halbe%+Spr.y,0.4

 Endif

 If K=2

 Pcircle X-Halbe%+Spr.x,Y-Halbe%+Spr.y,1

 Endif

 If K=3

 Deffill ,2,8

 Pcircle X-Halbe%+Spr.x,Y-Halbe%+Spr.y,2

 Deffill ,Fstil%,Fmuster%

 Endif

'je nach Maustastenauswahl werden hier drei verschieden große "Pcircle"
'gezeichnet. Die Zufallswerte, die in den Variablen "Spr.x" und "Spr.y"
'enthalten sind, werden in "Verteiler" produziert.


```

Endif
Return
Procedure Pt.spr

'Pt.spr = Pattern Spray
'      = Füllmuster sprühen
'Diese Proedur erfüllt die gleiche Funktion wie die vorhergehende.
'Der Unterschied besteht darin, daß hier mit dem eingestellten Füll-
'muster gefüllte "Pboxes" ohne Rahmen (s "Procedure A.srecp") auf den
'Bildschirm gesprüht werden. Ebenfalls in drei verschiedenen Größen
'und auf der gesamten Fläche der eingestellten Patchgröße. Hiermit
'lassen sich sehr interessante Effekte erzielen.
  Graphmode 3
  Defline lnstil%,lndik,0,0
  Box X-Halbe%,Y-Halbe%,X+Halbe%,Y+Halbe%
  Box X-Halbe%,Y-Halbe%,X+Halbe%,Y+Halbe%
  Graphmode Grafmod%
  If K>0
    @Rahmen(0)
    If K=1
      'Pbox X-Halbe%+Spr.x,Y-Halbe%+Spr.y,X-Halbe%+Spr.x,Y-Halbe%+Spr.y
    Endif
    If K=2
      Pbox X-Halbe%+Spr.x,Y-Halbe%+Spr.y,X-Halbe%+Spr.x+1,Y-
Halbe%+Spr.y+2
    Endif
    If K=3
      Pbox X-Halbe%+Spr.x,Y-Halbe%+Spr.y,X-Halbe%+Spr.x+2,Y-
Halbe%+Spr.y+4
    Endif
    @Rahmen(1)
  Endif
Return
Procedure Marker

'Das GEM des Atari stellt eine Komfortfunktion zur Verfügung, mit der
'bestimmte Markierungssymbole an bestimmte Positionen gezeichnet werden
'können (s."POLYLINE,POLYFILL,POLYMARK"). Diese Möglichkeit wird hier
'dazu verwendet, um diese "Marker" als Grafik-Element zu benutzen.

```

'Dazu wird einfach nur das Null-Feld der Koordinatenarrays mit einem
'Wert belegt und im Befehl "Polymark" nur eine Position angegeben.
'In diesem Fall verhält sich der Marker wie z.B. die vom Programm er-
'zeugten Patches.

```
Defmark ,Marker.typ%,Halbe%*2
```

DEFMARK

Mit "Defmark" kann die Farbe, der Typ und die Größe der Markierungssymbole bestimmt werden.

Als "Marker" stehen Ihnen 6 Typen zur Auswahl:

- Typ 1 = ein Ein-Pixel-Punkt
- Typ 2 = ein Plus-Zeichen
- Typ 3 = ein sechszackiger Stern
- Typ 4 = ein liegendes Rechteck
- Typ 5 = ein Diagonalkreuz bzw. X
- Typ 6 = eine liegende Raute

Alle Marker besitzen unveränderlich eine Liniendicke von einem Pixel. Die Größe ist nicht kontinuierlich veränderbar, sondern in 20er Schritten. Es ist also zwecklos, z.B. die Größe 28 einstellen zu wollen, da trotzdem die Größe 20 gezeichnet wird. Ein Beispiel zu diesem Befehl finden Sie unter "POLYLINE,POLYFILL,POLYMARK".

```
Graphmode 3
Xmark(0)=X
Ymark(0)=Y
Polymark 1,Xmark(),Ymark()
Polymark 1,Xmark(),Ymark()
If K=1
  Graphmode Grafmod%
  Polymark 1,Xmark(),Ymark()
Endif
Return
Procedure Markermenu
```

'Dieses ist die Prozedur zur Erstellung des Auswahlmenüs für die
'zu verwendenden Markierungssymbole

```
Get 250,150,390,280,Backmenu$
@Grow.shrink.box(0,330,320,45,45,250,150,140,130)
```

'In der "Procedure Grow.shrink.box" befindet sich eine GEM-Funktion, mit
'der die so dynamisch wirkenden sich öffnenden und schließenden Boxes
'erzeugt werden, wie sie zum Beispiel auf dem Desktop beim Öffnen eines
'Diskettenfensters zu beobachten sind. Weiter Erläuterungen dazu finden
'Sie in "Grow.shrink.box".

```
Deftext 1,1,0,9
@Set_menuback
Deffill ,0,0
For J=0 To 2
  For I=0 To 1
    Inc M.mk
    Defmark ,M.mk,20
    Pbox 265+I*55,165+J*33,320+I*55,198+J*33
    Xmark(0)=292+I*55
    Ymark(0)=181+J*33
    Polymark 1,Xmark(),Ymark()
  Next I
Next J
M.mk=0
Repeat
  Mouse X,Y,K
Until K=0
Repeat
  Until Mousek
Repeat
```

'Die Menü-Analyse ist an anderer Stelle bereits beschrieben worden.

```
Mouse X,Y,K
Mark.x=Int((X-265)/55)+1
Mark.y=Int((Y-165)/33)
If Mark.x>=1 And Mark.x<=2 And Mark.y>=0 And Mark.y<=2
```

```

Marker.typ%=Mark.x+Mark.y*2
Endif
Until K=0
Put 250,150,B.ackmenu$,3
Return
Procedure T.ext

```

'Wieder eine interessante Prozedur ist die zur Darstellung von Grafik-
'text auf dem Bildschirm.

'Der eingegebene Textstring kann auf vier verschiedene Arten editiert
'werden. Mit der Backspace-Taste (ASCII-Wert 8) wird jeweils das letzte
'Zeichen gelöscht. Mit den vier Cursortasten kann zusätzlich

'

' a) die optische Stringlänge bestimmt werden

' Pfeil-links-Taste = verkürzen

' Pfeil-rechts-Taste = verlängern

'

' b) der Dehnungsmodus bestimmt werden

' Pfeil-unten-Taste = Wechsel zwischen beiden Modi

'

' Mit der Pfeil-oben-Taste kann die Normaldarstellung wieder ein-
' gestellt werden.

'

'Nähere Erläuterungen und ein Beispiel hierzu finden Sie unter "TEXT".

'Die Verwaltung dieser Cursortasten-Funktionen finden Sie in "Procedure
'T_extlen".

```

If Fkey.2%<>18 And Fkey.2%<>27 And Fkey.2%<>13 And Fkey.2%<>8 And
Fkeys%<>0

```

'Wenn weder die "Tab"-Taste (ASCII-Wert 18), noch die "Esc"-Taste
'(ASCII-Wert 27), noch die "Return"-Taste (ASCII-Wert 13), noch die
'"Backspace"-Taste (ASCII-Wert 8), aber trotzdem irgendeine andere
'Taste (ASCII-Wert <>0) gedrückt wurde,...

```

If (Fkey%<>72 And Fkey%<>75 And Fkey%<>77 And Fkey%<>80) And Fkey.2%
>0

```

'...aber diese andere Taste keine der Cursortasten war,...

```
T_char$=Chr$(Fkey.2%)
T_string$=T_string$+T_char$
```

'...dann wandle den ASCII-Wert der gedrückten Taste in ein Stringzeichen
'um und füge es dem Textstring an.

'Es können auch Sonderzeichen eingegeben werden ("Control"+Taste).

```
'
' Anmerkung:
' Es lassen sich übrigens überall im Programm (außer in der
' "Flimmerphase" der Blockoperationen) zur pixelgenauen Maussteuerung
' auch folgende Desktop-Funktionen verwenden:
```

```
' "Alternate/Pfeiltasten"      = Mauszeiger in 8-Pixel-Schritten
'                               in Pfeilrichtung bewegen
' "Alternate/Shift/Pfeiltasten" = Mauszeiger in Pixelschritten in
'                               in Pfeilrichtung bewegen
' "Alternate/Insert"           = linke Maustaste wird simuliert
' "Alternate/ClrHome"          = rechte Maustaste wird simuliert
' "Alternate/Insert/
'   Pfeiltasten"               = linke Maustaste gedrückt und
'                               gleichzeitig bewegen (8 Pixels)
' "Alternate/Shift/Insert/
'   Pfeiltasten"               = linke Maustaste gedrückt und
'                               gleichzeitig bewegen (1 Pixel)
' "Alternate/ClrHome/
'   Pfeiltasten"               = rechte Maustaste gedrückt und
'                               gleichzeitig bewegen (8 Pixels)
' "Alternate/Shift/ClrHome/
'   Pfeiltasten"               = rechte Maustaste gedrückt und
'                               gleichzeitig bewegen (1 Pixel)
```

```
Endif
Endif
If K=2 Or Fkeys%=254
```

'Wenn die rechte Maustaste oder die "Delete"-Taste (ASCII-Wert 254)
'gedrückt wurde, lösche den Textstring

```

T_string$=""
Repeat
  D.um$=Inkey$
Until D.um$=""

```

'Diese Repeat/Until-Schleife leert den Tastaturpuffer

```

Endif
If Fkey.2%=8 And Len(T_string$)>0

```

'Wenn die "Backspace"-Taste gedrückt wurde, verkürze den String um ein
'(das letzte) Zeichen, aber nur, wenn der String größer als ein Zeichen
'ist. Wird versucht mit "Mid\$" im Minusbereich ein Zeichen zu erfassen
'(was natürlich nicht funktionieren kann), wird eine Fehlermeldung
'ausgegeben, die das Programm evtl. abbricht und das aktuelle Bild
'zerstört.

```

  T_string$=Mid$(T_string$,1,(Len(T_string$)-1))
Endif
Graphmode 3
Text X,Y,T_grow%(0),T_string$
Text X,Y,T_grow%(0),T_string$
If K=1 Or Fkey.2%=13
  Graphmode Grafmod%
  Text X,Y,T_grow%(0),T_string$
Endif

```

Return

Procedure T.extmenu

```

  Get 250,150,390,280,B.ackmenu$
  T.menstart:
  @Grow.shrink.box(0,385,320,45,45,250,150,140,130)
  Deftext 1,17,0,6
  Defline 0,1,0,0
  @Set_menuback
  Restore T.m.text
  Deffill ,0,0
  For J=0 To 1
    For I=0 To 1
      Deftext 1,Random(15),0,6
      Pbox 266+I*55,165+J*55,319+I*55,210+J*55
    
```

```
Text 268+I*55,180+J*55,"TEXT-"
Read T.men_text$
Text 268+I*55,200+J*55,T.men_text$
Next I
Next J
Repeat
Until Mousek
Repeat
  Mouse M.en_x,M.en_y,Button
Until Button=0
Tt.xp=Int((M.en_x-266)/55)+1
Tt.yp=Int((M.en_y-162)/55)
Tt.menpkt=0
If Tt.xp>=1 And Tt.xp<=2 And Tt.yp=>0 And Tt.yp<1
  Tt.menpkt=Tt.xp+Tt.yp*2
Endif
If Tt.menpkt=1 Or Tt.menpkt=2
  Deftext 1,1,0,6
  @Set_menuback
  Deftext 1,1,0,9
  Graphmode 2
  Box 281,190,301,210
  Box 280,189,302,211
  Box 341,190,361,210
  Box 340,189,362,211
  If Tt.menpkt=1
    Text 280,180,"TEXTART"
  Else
    Text 274,180,"TEXTHÖHE"
  Endif
  Text 285,205,"+"
  Text 345,205,"-"
  Deftext 1,T.extvar(2),0,T.extvar(4)
  Text 266,252,90,"TEXT"
  If Tt.menpkt=1
    Print At(40,13);T.extvar(2)
  Else
    Print At(40,13);T.extvar(4)
  Endif
Repeat
```

```
Until Mousek Or (X<250 Or Y<150 Or X>390 Or Y>280)
Graphmode 1
Repeat
  Mouse X,Y,K
  If K>0
    If X>281 And X<301 And Y>190 And Y<210 And Tt.menpkt=1
      Inc T.extvar(2)
      If T.extvar(2)>31
        T.extvar(2)=0
      Endif
    Endif
    If X>341 And X<361 And Y>190 And Y<210 And Tt.menpkt=1
      Dec T.extvar(2)
      If T.extvar(2)<0
        T.extvar(2)=31
      Endif
    Endif
    If X>281 And X<301 And Y>190 And Y<210 And Tt.menpkt=2
      Inc T.extvar(4)
      If T.extvar(4)>26
        T.extvar(4)=2
      Endif
    Endif
    If X>341 And X<361 And Y>190 And Y<210 And Tt.menpkt=2
      Dec T.extvar(4)
      If T.extvar(4)<2
        T.extvar(4)=26
      Endif
    Endif
    Pause 7
    If Tt.menpkt=1
      Print At(40,13);T.extvar(2)'
    Else
      Print At(40,13);T.extvar(4)'
    Endif
    Deftext 1,T.extvar(2),0,T.extvar(4)
    @Rahmen(0)
    Pbox 263,212,378,268
    @Rahmen(1)
    Text 266,252,90,"TEXT"
```



```
Endif
Until X<250 Or Y<150 Or X>390 Or Y>280
Goto T.menstart
Endif
If Tt.menpkt=3
  @Set_menuback
  Deftext 1,1,0,6
  Text 329,219,"EAST"
  Deftext ,,900,
  Text 323,204,"NORTH"
  Deftext ,,1800,
  Text 309,213,"WEST"
  Deftext ,,2700,
  Text 317,227,"SOUTH"
  Box 327,209,370,223
  Box 313,162,327,207
  Box 270,209,313,223
  Box 313,225,327,268
  Deffill ,2,2
  Pbox 270,162,308,204
  Pbox 332,162,370,204
  Pbox 270,228,308,268
  Pbox 332,228,370,268
  Pbox 317,211,323,221
  Deftext 1,1,0,13
  Repeat
  Until Mousek Or (X<250 Or Y<150 Or X>390 Or Y>280)
  Graphmode 3
  Deffill ,2,8
  Repeat
    Mouse X,Y,K
    If K>0
      If X>327 And X<370 And Y>209 And Y<223
        T.extvar(3)=0
        Pbox 327,209,370,223
        Pause 5
        Pbox 327,209,370,223
      Endif
      If X>313 And X<327 And Y>162 And Y<207
        T.extvar(3)=900
```

```
Pbox 313,162,327,207
Pause 5
Pbox 313,162,327,207
Endif
If X>270 And X<313 And Y>209 And Y<223
  T.extvar(3)=1800
  Pbox 270,209,313,223
  Pause 5
  Pbox 270,209,313,223
Endif
If X>313 And X<327 And Y>225 And Y<268
  T.extvar(3)=2700
  Pbox 313,225,327,268
  Pause 5
  Pbox 313,225,327,268
Endif
Pause 7
Endif
Until X<250 Or Y<150 Or X>390 Or Y>280
Goto T.menstart
Endif
If Tt.menpkt=4
  @Set_menuback
  Deftext 1,16,0,6
  Text 278,180,"TEXTFARBE"
  Deftext ,1,,12
  Text 281,221,"WHITE"
  Text 281,252,"BLACK"
  Box 277,234,364,256
  Graphmode 3
  Deffill ,2,8
  Pbox 277,203,364,225
  Repeat
  Until Mousek Or (X<250 Or Y<150 Or X>390 Or Y>280)
  Repeat
    Mouse X,Y,K
    If K>0
      If X>277 And X<364 And Y>234 And Y<256
        T.extvar(1)=1
        Pbox 277,234,364,256
```

```
    Pause 5
    Pbox 277,234,364,256
Endif
If X>277 And X<364 And Y>203 And Y<225
    T.extvar(1)=0
    Pbox 277,203,364,225
    Pause 5
    Pbox 277,203,364,225
Endif
Pause 7
Endif
Until X<250 Or Y<150 Or X>390 Or Y>280
Goto T.menstart
Endif
Put 250,150,B.ackmenu$,3
Return
Procedure Set_menuback
    Graphmode 1
    Deffill ,0,0
    Pbox 250,150,390,280
    Box 251,151,389,279
    Deffill ,2,4
    Pbox 254,154,386,276
    Deffill ,0,0
    Pbox 260,160,380,270
Return
Procedure Areaprocs
```

'Im Hauptmenü hat man die Möglichkeit, einen gespeicherten
'Bildschirmausschnitt
'zu laden und in das aktuelle Bild einzusetzen. Damit im Anschluß
'an diesen
'Prozess, der über die gleiche Prozedur gesteuert wird, wie die
'MOVE/COPY-Funktionen,
'kein Fehler auftritt, muß hier analysiert werden, ob es sich beim
'Betreten
'der Block-Prozedur um eine der beiden Blockfunktionen handelt, oder ob
'nur ein Bildausschnitt geladen wurde.

```
    If Block.flag%=2
```

'wenn ein Bildausschnitt geladen wurde, gehe in die Placierungsprozedur
'und schalte anschließend den COPY-Modus wieder aus.

```
@Setareas
Areas%=0
C.opy%=0
Block.flag%=0
Goto Nonedo
Endif
@Rubbing
If Norubb%=0
  Rubbflag%=0
  If Block.flag%<>1
```

'"Block.flag%=2" wurde vorher schon abgefangen. Es bleibt nur noch
'"Block.flag%=1" oder "Block.flag%<>1" übrig. "Block.flag%=1" würde
'bedeuten, daß ein Bildausschnitt schon definiert ist und gespeichert
'werden soll.

```
Get Xl%,Yl%,Xx%,Yy%,Mc.part$
```

'Ist das nicht der Fall, kann nur noch eine der beiden Blockoperationen
'(MOVE/COPY) gemeint sein. Dann speichere den gewählten Ausschnitt
'im Rasterpuffer.

```
Endif
@Setareas
```

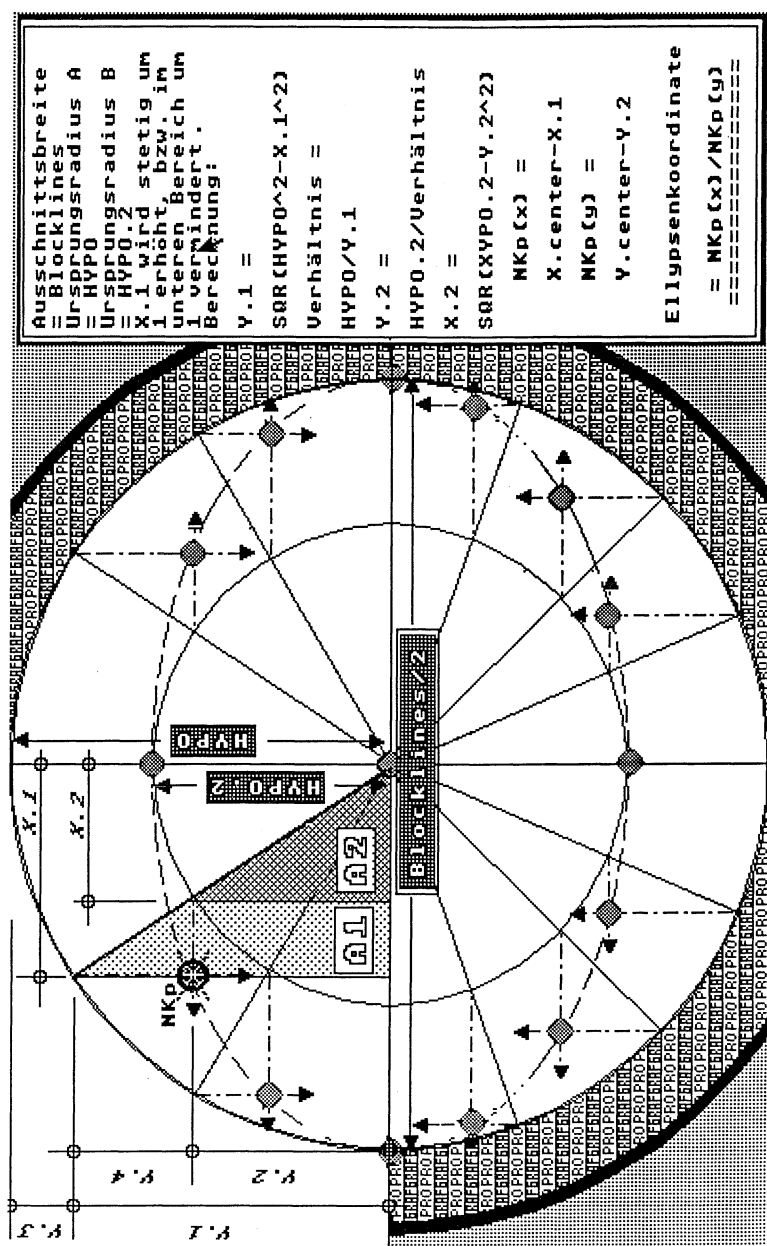
'...gehe zur Placierungsroutine

```
Endif
Nonedo:
Return
Procedure Set.clipping
```

'dem Begriff "Clipp" sind sie bis hierher schon öfter begegnet.
'In dieser
'Prozedur wird der Bereich definiert, an dessen Rändern alle Grafikaus-
'gaben abgeschnitten (clipping; engl.:Ausschnitt) werden sollen. Die

'Rasterfunktionen "SELF" und "MOVE/COPY" lassen sich damit allerdings
'nicht kontrollieren.

```
@Clipprec(2)
Deftext 1,1,0,6
Graphmode 3
Repeat
    Mouse Cl.xl%,Cl.yl%,Button%
    Text Cl.xl%,Cl.yl%,"CLIPP-BOX SETZEN"
    Text Cl.xl%,Cl.yl%,"CLIPP-BOX SETZEN"
Until Button%>0
@Rubbself
Cl.xr%=Xx%
Cl.yr%=Yy%
@Set.paras
Clipp.set%=0
Return
```



Procedure Setareas

 @Clipprec(2)

 Mc.done%=0

 If Block.flag%=1

'Wie in "Areaprocs" beschrieben, wird hier ein Bildausschnitt auf
'Diskette gespeichert.

 Fileselect "*.blo", ".blo", Selected\$

 If Selected\$<>"\BLO" And Selected\$>"" And Selected\$<>"\"

 Defmouse 2

 E.xtender\$="BLO"

 @Extend

 Get Xl%,Yl%,Xx%,Yy%,Mc.part\$

'Nachdem eine Datei bestimmt wurde, in die der Ausschnitt gespeichert
'werden soll, speichere den Ausschnitt im Rasterpuffer...

 Bsave Selected\$,Varptr(Mc.part\$),Len(Mc.part\$)

'...und speichere seinen Inhalt auf Diskette ab

 Defmouse 0

 Endif

 Block.flag%=0

 Areas%=0

 C.opy%=0

'Die Operation ist beendet, kehre zur Hauptschleife zurück.

 Goto S.hownoblock

 Endif

 Block.ops=0

 Showarea:

 Repeat

'Hier wird der Schwebe-Effekt des Ausschnitts produziert

 Mouse X.offset%,Y.offset%,Button%

 Put Xl%,Yl%,Mc.part\$,6

```

Put Xl%,Yl%,Mc.part$,6
Key$=Inkey$
K.ey%=Asc(Left$(Key$))+Asc(Right$(Key$))
Until Mousek Or K.ey%=54 Or K.ey%=72 Or K.ey%=75 Or K.ey%=80 Or K.ey%=
77 Or K.ey%=100 Or K.ey%=104 Or K.ey%=108 Or K.ey%=112

```

'Die "Flimmer"-Schleife wird nur verlassen, wenn eine der folgenden
'Tasten gedrückt wird:

'- Maustaste	=> Block bewegen
'- "Esc"-Taste (ASCII=27, hier: *2=54)	=> rufe Hauptmenü auf
'- Pfeil-links-Taste (SCAN=75)	=> Block vertikal spiegeln
'- Pfeil-unten-Taste (SCAN=80)	=> Block horizontal spiegeln
'- Pfeil-oben-Taste (SCAN=72)	=> Block-Tonnenwölbung
'- Pfeil-rechts-Taste(SCAN=77)	=> Block-Sinuswelle
'- Shift/Pfeil-links (SCAN=104)	=> Block horizontal verkürzen
'- Shift/Pfeil-oben (SCAN=112)	=> Block vertikal verkürzen
'- Shift/Pfeil-unten (SCAN=100)	=> Block horizontal dehnen
'- Shift/Pfeil-rechts(SCAN=108)	=> Block vertikal dehnen

```

On Error Gosub B_lockerror
If K.ey%=54

```

'Wenn die "Esc"-Taste gedrückt wurde, verlasse diese Prozedur

```

Goto S.hownoblock
Endif
If Xx%-Xl%<2 Or Yy%-Yl%<2 And Block.ops=0

```

'Um Errors zu vermeiden, muß der Block in diesem Fall auf eine
'Minimalgröße festgelegt werden (2*2 Pixels).

```

Alert 1,"Block ist zu klein !!|(Minimal: 2 * 2)",1," OKAY ",Backaler
t%

```

```

Goto S.hownoblock
Endif

```

'Die Bedeutung der folgenden acht "Gosubs" finden Sie oben beschrieben.
'Wenn das Programm die jeweilige Sonderbehandlung durchgeführt hat, geht
'es wieder in die "Flimmerschleife zurück (Goto Showarea).


```
If K.ey%=80
    Block.ops=1
    @Block.h.mir
    Goto Showarea
Endif
If K.ey%=75
    Block.ops=1
    @Block.v.mir
    Goto Showarea
Endif
If K.ey%=72
    Block.ops=1
    @Block.y.elly
    Goto Showarea
Endif
If K.ey%=77
    Block.ops=1
    @Block.x.fade
    Goto Showarea
Endif
If K.ey%=100
    Block.ops=1
    @Block.h.grow
    Goto Showarea
Endif
If K.ey%=112
    Block.ops=1
    @Block.h.shrink
    Goto Showarea
Endif
If K.ey%=104
    Block.ops=1
    @Block.v.shrink
    Goto Showarea
Endif
If K.ey%=108
    Block.ops=1
    @Block.v.grow
    Goto Showarea
Endif
```

On Error

If M.ove%=1 And K.ey%=0 And Block.ops=0

'Falls "MOVE" angeklickt wurde, lösche den Bereich, aus dem der Block
'"herausgeschnitten" werden soll.

```

Graphmode 1
Deffill ,0,0
@Rahmen(0)
Pbox Xl%,Yl%,Xx%,Yy%
@Rahmen(1)
@Set.paras
Endif
Block.ops=0
Repeat

```

'Dies ist die Darstellungs-Schleife. Sie wird solange durchlaufen, bis
'der in der letzten Schleife gedrückte Mausknopf wieder losgelassen
'wird. Bleibt er gedrückt, kann der Block über den Bildschirm bewegt
'werden. Die Verfahrensweise dieser Darstellungsart wurde an anderer
'Stelle bei einem ähnlichen Fall bereits beschrieben.

```

Mouse X.x%,Y.y%,Button%
If Showlines%=1
  X=X.x%
  Y=Y.y%
  @S.howlines
Endif
If Const.lines%=1
  @C.onst.lines
Endif
If Netzmod%=1
  X.x%=Int(X.x%/Netzweite%)*Netzweite%
  Y.y%=Int(Y.y%/Netzweite%)*Netzweite%
Endif
Put (X.x%-(X.offset%-Xl%)),(Y.y%-(Y.offset%-Yl%)),Mc.part$,6
Put (X.x%-(X.offset%-Xl%)),(Y.y%-(Y.offset%-Yl%)),Mc.part$,6
Until Button%=0 Or Button%=3
If Button%=1 Or Button%=3
  Mc.done%=1

```

```

    If Button%=3
        Put Xl%,Yl%,Mc.part$,3
    Endif
Endif
If Button%=0 And K.ey%=0
    @Changemod
        Put      (X.x%-(X.offset%-Xl%)),(Y.y%-(Y.offset%-
Yl%)),Mc.part$,Rastermod%
    Endif
    S.hownoblock:
    @G.etflip
    If Flip.done%=1 And S.wapper%=1

```

'wenn beim Screenwechsel "MOVE" angeschaltet war, wurden "M.ove%" und
 "'C.opy%" getauscht. Nach vollendeter Aktion auf der neuen Screen
 'wird nun wieder zu "MOVE" zurückgeschaltet. Ab hier geht's auf der
 'neuen Screen wieder mit "MOVE" weiter.

```

        S.wapper%=0
        Swap M.ove%,C.opy%
    Endif
    Flip.done%=0
Return
Procedure B_lockerror
    If Err=8
        Alert 3,"Speicher ist leider voll !!|Evtl. kleineren Ausschnitt
|wählen !",1," OKAY ",Backalert%
    Else
        Error Err
    Endif
    Erase Mir.mem$( )
    @P.utflip
    Resume S.hownoblock
Return
Procedure Block.h.grow

```

'Prozedur zur Horizontal-Dehnung des Blockes

```

Erase Ygd%( )
Dim Ygd%(0)

```

'Die Dimensionierung von "Ygd%" hat keine weitere Bedeutung. Durch die
'Programmlänge ist bei der mir vorliegenden Version ein Fehler in der
'Adressenverwaltung aufgetreten. Dieser Fehler wird durch die
'Dimensionierung umgangen.

Blocklines%=Yy%-Yl%

'Stelle fest, wieviel Pixels der Block hoch ist...

Dim Mir.mem\$(Blocklines%)

'...bereite ein Feld mit dementsprechend vielen Elementen vor...

For I=1 To Blocklines%

Get Xl%,Yl%+I-1,Xx%,Yl%+I,Mir.mem\$(I)

'...zerschneide den Block in so viele horizontale Pixelzeilen...

Next I

Cls

'...lösche den Bildschirm, damit der Hintergrund nicht in den neuen
'Block mit aufgenommen wird...

For I=1 To Blocklines%

Put Xl%,Yl%+(I*2),Mir.mem\$(I),3

'...setze den neuen Block zusammen...

Next I

Ygd%(0)=Yl%+2*Blocklines%

'...errechne den neuen unteren Rand des Blockes...

If Ygd%(0)>399

'...falls er über den unteren Bildschirmrand hinausgeht, schneide
'den neuen Block dort ab...

```
Ygd%(0)=399
Endif
Get Xl%,Yl%,Xx%,Ygd%(0),Mc.part$

'...übertrage den neuen Block in den Rasterpuffer...

aP.utflip

'...und setze das aktuelle Bild wieder auf die Screen.

Erase Mir.mem$()
Return
Procedure Block.h.shrink

'Diese Prozedur ist größtenteils identisch mit der vorhergehenden. In
'diesem Fall wird jedoch der neue Block 1:2 vertikal verkürzt

Blocklines%=Yy%-Yl%
Dim Mir.mem$(Blocklines%)
For I=1 To Blocklines%
    Get Xl%,Yl%+I-1,Xx%,Yl%+I,Mir.mem$(I)
Next I
Cls
For I=1 To Blocklines%
    Put Xl%,Yl%+Int(I/2),Mir.mem$(I),3
Next I
Get Xl%,Yl%,Xx%,Yl%+Int(Blocklines%/2),Mc.part$
aP.utflip
Erase Mir.mem$()
Return
Procedure Block.v.grow

'Das gleiche Verfahren wie in "Block.h.grow". Hier wird die Breite des
'Blockes gedehnt.

Erase Xgd%()
Dim Xgd%(0)
Blocklines%=Xx%-Xl%
Dim Mir.mem$(Blocklines%)
```

```

For I=1 To Blocklines%
  Get Xl%+I-1,Yl%,Xl%+I,Yy%,Mir.mem$(I)
Next I
Cls
For I=1 To Blocklines%
  Put Xl%+I*2,Yl%,Mir.mem$(I),3
Next I
Xgd%(0)=Xl%+Blocklines%*2
If Xgd%(0)>639
  Xgd%(0)=639
Endif
Get Xl%,Yl%,Xgd%(0),Yy%,Mc.part$
@P.utflip
Erase Mir.mem$()
Return
Procedure Block.v.shrink

```

'Wieder das gleiche wie in "Block.h.shrink". Diesmal wird die Breite
'des Blockes verkürzt.

```

Blocklines%=Xx%-Xl%
Dim Mir.mem$(Blocklines%)
For I=1 To Blocklines%
  Get Xl%+I-1,Yl%,Xl%+I,Yy%,Mir.mem$(I)
Next I
Cls
For I=1 To Blocklines%
  Put Xl%+Int(I/2),Yl%,Mir.mem$(I),3
Next I
Get Xl%,Yl%,Xl%+Int(Blocklines%/2),Yy%,Mc.part$
@P.utflip
Erase Mir.mem$()
Return
Procedure Block.h.mir

```

'Prinzipiell unterscheidet sich auch diese Prozedur nicht von den
'vorhergehenden. Nur, daß hier die einzelnen Bildstreifen horizontal
'in der umgekehrten Reihenfolge des Aufnehmens wieder auf den
'Bildschirm gebracht, also horizontal gespiegelt werden.

```

Blocklines%=Yy%-Yl%
Dim Mir.mem$(Blocklines%)
For I=1 To Blocklines%
    Get Xl%,Yl%+I-1,Xx%,Yl%+I,Mir.mem$(I)
Next I
Cls
For I=1 To Blocklines%
    Put Xl%,Yy%-I,Mir.mem$(I),3
Next I
Get Xl%,Yl%,Xx%,Yy%,Mc.part$
@P.utflip
Erase Mir.mem$()
Return
Procedure Block.v.mir
    Blocklines%=Xx%-Xl%
    Dim Mir.mem$(Blocklines%)
    For I=1 To Blocklines%
        Get Xl%+I-1,Yl%,Xl%+I,Yy%,Mir.mem$(I)
    Next I
    Cls
    For I=1 To Blocklines%
        Put Xx%-I,Yl%,Mir.mem$(I),3
    Next I
    Get Xl%,Yl%,Xx%,Yy%,Mc.part$
    @P.utflip
    Erase Mir.mem$()
Return
Procedure Block.y.elly

```

'Dieses ist vielleicht die interessanteste Prozedur des ganzen
'Programms.

'Nicht weil sie besonders ausgeklügelt wäre, sondern weil mit dieser Art
'der Grafikprogrammierung eine neue Dimension eingeführt wird.

'Wir kennen

'diese Tricks alle schon aus dem Fernsehen. Alle wissen, wie es
'aussieht,

'aber nur wenige wissen, wie es funktioniert: die dreidimensionale
'Darstellung.

'Nicht etwa von mathematisch errechneten räumlichen Figuren, sondern von

'komplexen Bildern. Dieses Programm ist in seiner ganzen Konzeption
 'nicht
 'dazu geeignet, umfangreiche Prozesse dieser Art zu verwalten. Es ist ja
 'ein Grafikprogramm und keine 3D-Bastelstube. Trotzdem können diese
 'Routinen
 'eine Anregung sein, in dieser Richtung eigene Ideen zu entwickeln und
 'zu verwirklichen.
 'Diese Routine hat nun die Aufgabe, den durch MOVE/COPY gewählten Bild-
 'ausschnitt anhand des Einsatzes vom guten alten Pythagoras in ein
 'tonnen-ähnlich gewölbtes Bild zu verwandeln. Dabei kann die Neigung
 'dieser Tonne
 'frei gewählt werden. Es kann also sowohl eine kreisförmige, als auch
 'eine Ellipsenförmige Projektion entstehen.

Blocklines%=Xx%-Xl%

Dim Mir.mem\$(Blocklines%)

Alert 2,"Mit oder ohne Rand?",0," MIT | OHNE ",Backalert%

'Um die Sache etwas interessanter zu machen, kann gewählt werden, ob die
 'Tonne oben und unten einen schwarzen Rand erhalten soll oder nicht.

If Backalert%=1

Sub Yl%,3

Add Yr%,3

Endif

Color 1

Defline 0,3,0,0

If Backalert%=1

Line Xl%,Yl%,Xx%,Yl%

Line Xl%,Yy%,Xx%,Yy%

'Wenn ja, werden hier zwei dicke Linien auf den oberen, bzw. unteren
 'Rand des Blockes gezeichnet.

Endif

For I=1 To Blocklines%

Get Xl%+I-1,Yl%-1,Xl%+I,Yy%+1,Mir.mem\$(I)

'Die Aufnahme der einzelnen Zeilen ist uns schon aus den vorherigen
 'Routinen bekannt.


```
Next I
Cls
Defline ,1,,
Print At(2,2);"Bitte Neigung bestimmen !"
```

'Ab hier wird nun eine Schleife aufgebaut, die eine wandelbare Ellipse
'zeichnet und auf Mausbewegungen bzw. einem anschließenden
'Maustastendruck
'wartet. Damit läßt sich die Tonnen-Neigung bestimmen.

```
Graphmode 3
Pause 10
Xrad%=Blocklines%/4
Repeat
  Mouse Ell.xx,Ell.yy,K
  Yrad%=Abs(200-Ell.yy)
  Ellipse 320,200,Xrad%,Yrad%
  Ellipse 320,200,Xrad%,Yrad%
Until Mousek
Cls
Blockrows%=Yy%-Yl%
```

'In diesem Fall benötigen wir außer der Blockbreite auch die Blockhöhe,
'um hinterher errechnen zu können, wo der untere Rand des entstehenden
'Tonnenbildes liegt

```
Xl%=320-(Blocklines%/4)
Yl%=0
```

'Um nicht soviel umrechnen zu müssen, bekommt der neue Block ein
'Position,
'die sich leichter in den mathematischen Vorgang einbinden läßt.

```
H_ypo%=Blocklines%/4
H_ypo2%=Yrad%
```

'Für die Ellipsenberechnung brauchen wir die Radien der beiden Kreise,
'die die Lage der Ellipsenbahn bestimmen.
'Das ist einmal der vorher bestimmte Ellipsen-Y-Radius und zum anderen

'die Breite der Tonne (Tonnendurchmesser=Blockbreite/2)) geteilt durch 2
'(d/2=r), also Blockbreite/4.

Center.x%=Xl%+H_ypo%

Center.y%=Yl%+H_ypo%

'Hier wird der gemeinsame Mittelpunkt beider Ursprungskreise errechnet.

@Tinny

Kr.yr%=Yl%+2*Yrad%+Blockrows%+1

'Nach Abschluß der Berechnungen und der Tonnenzeichnung wird wieder der
'untere Rand des neuen Blockes berechnet.

If Kr.yr%>398

'Falls dieser außerhalb des Bildschirms liegt, wird der neue Block am
'unteren Bildrand abgeschnitten,

Kr.yr%=398

Endif

Get Xl%,Yl%+1,Xl%+2*H_ypo%,Kr.yr%+1,Mc.part\$

'Übernahme des neuen Blockes in den Rasterpuffer

@P.utflip

'Aktuelle SCREEN neu setzen

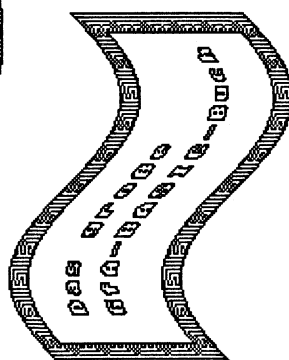
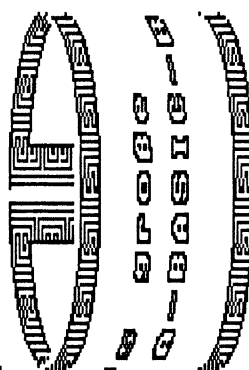
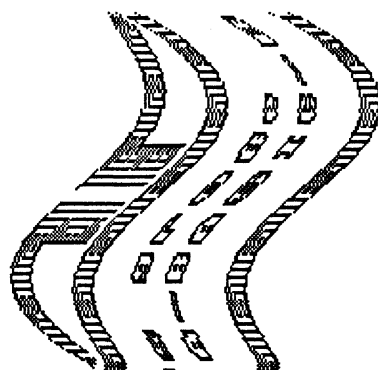
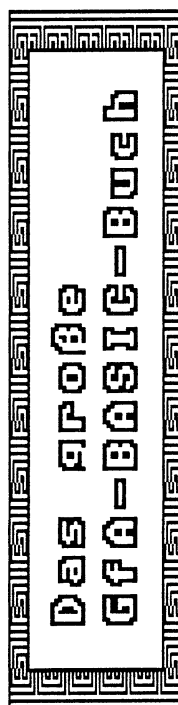
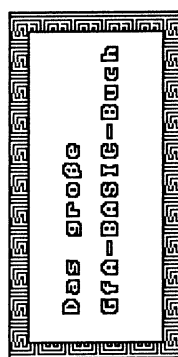
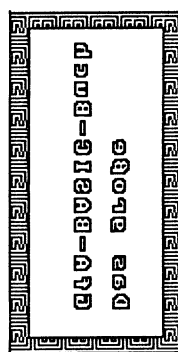
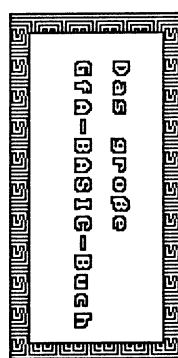
Mem.index%=0

'In "Tinny" wird eine Zählvariable eingesetzt. Damit beim nächsten Mal
'nichts schiefgeht, muß diese wieder auf Null gesetzt werden.

Erase Mir.mem\$()

Return

Procedure Tinny



'In dieser Abbildung wurden andere Variablennamen als in den folgenden
 'Prozeduren verwendet. Der beschriebene Algorithmus (Verfahrensweise)
 'ist ebenfalls nicht identisch. Die Grafik soll in groben Umrissen das
 'Prinzip dieser Methode erkennbar machen.
 'Grundsätzlich handelt es sich jedoch um den gleichen Vorgang.
 'Es existieren zwei Ursprungskreise. Der Durchmesser des einen ergibt
 'sich aus der halben Ausschnittsbreite. Den Durchmesser des anderen
 'können Sie im Programm frei bestimmen.
 'Nach dem in der Grafik dargestellten Verfahren können jetzt die
 'einzelnen
 'Umfangspunkte (Ausschnittsbreite=200) der Ellipse errechnet werden, an
 'welche die Bildstreifen der Reihe nach angelegt werden.
 'Dazu ist für jeden einzelnen dieser Bildpunkte diese Berechnung
 'auszuführen.

```
For I%=-1 Downto (H_ypo%*(-1))
```

'Zuerst 50 Punkte vom Scheitelpunkt aus in Richtung des Schnittpunktes
 'des Kreises mit der horizontalen Mittelachse.

```

X.left1%=Center.x%+I%
Y.left1=Center.y%-Sqr(H_ypo%^2-(I%*(-1))^2)
Verhaeltnis=H_ypo%/Y.left1
Y.left2%=H_ypo2%/Verhaeltnis
Put X.left1%,Y.left2%+1,Mir.mem$(Mem.index%),Rastermod%
Inc Mem.index%
If Mem.index%=H_ypo%
  I%=(H_ypo%*(-1))
Endif
Next I%
Mem.index%=3*H_ypo%
For I%=H_ypo% Downto 1
```

'Anschließend die rechte obere ViertelEllipse vom rechten Schnittpunkt
 'des Kreises mit der horizontalen Mittelachse aus in Richtung
 'Scheitelpunkt.

```

X.left1%=Center.x%+I%
Y.left1=Center.y%-Sqr(H_ypo%^2-(I%*(-1))^2)
Verhaeltnis=H_ypo%/Y.left1
```

```

Y.left2%=H_ypo2%/Verhaeltnis
Put X.left1%,Y.left2%+1,Mir.mem$(Mem.index%),Rastermod%
Inc Mem.index%
If Mem.index%=4*H_ypo%
  I%=1
Endif
Next I%
Mem.index%=H_ypo%
For I%=(H_ypo%*(-1))) To -1

```

'50 Punkte vom linken Schnittpunkt des Kreises mit der horizontalen
'Mittelachse aus in Richtung Fußpunkt.

```

X.left1%=Center.x%+I%
Y.left1%=Center.y%+Sqr(H_ypo%^2-(I%*(-1))^2)
Verhaeltnis=H_ypo%/Y.left1
Y.left2%=H_ypo2%/Verhaeltnis
Put X.left1%,Y.left2%+1,Mir.mem$(Mem.index%),Rastermod%
Inc Mem.index%
If Mem.index%=2*H_ypo%
  I%=1
Endif
Next I%
For I%=1 To H_ypo%

```

'und zuletzt nochmal 50 Punkte vom Fußpunkt aus in Richtung rechter
'Schnittpunkt des Kreises mit der horizontalen Mittelachse.

```

X.left1%=Center.x%+I%
Y.left1%=Center.y%+Sqr(H_ypo%^2-(I%*(-1))^2)
Verhaeltnis=H_ypo%/Y.left1
Y.left2%=H_ypo2%/Verhaeltnis
Put X.left1%,Y.left2%+1,Mir.mem$(Mem.index%),Rastermod%
Inc Mem.index%
If Mem.index%=3*H_ypo%
  I%=H_ypo%
Endif
Next I%
Return
Procedure Block.x.fade

```

'Eine weitere interessante Routine zur verzerrten Darstellung von
 'Bildern
 'ist diese. Der gewählte Bildausschnitt wird hier an eine Sinuswelle
 'angelegt,
 'deren Höhe Sie selbst bestimmen können.

```
Blocklines%=Xx%-Xl%
Blockrows%=Yy%-Yl%
Dim Mir.mem$(Blocklines%)
For I=1 To Blocklines%
  Get Xl%+I-1,Yl%,Xl%+I,Yy%,Mir.mem$(I)
Next I
Cls
```

'Bis hierhin ist alles schon bekannt.

```
Print At(2,2);"Bitte Sinushöhe bestimmen !"
```

'Ab hier wird eine Schleife gebaut, innerhalb derer durch Mausbewegung
 'und anschließendem Maustastendruck die Höhe der Sinuswelle bestimmt
 'werden kann.

```
Graphmode 3
Defline 0,3,2,2
Repeat
  Mouse X,Ysin,K
  Line 320,Ysin,320,399-Ysin
  Line 320,Ysin,320,399-Ysin
  Print At(2,4);Abs(Ysin-(399-Ysin))''
Until Mousek
Cls
Graphmode 1
Defline 0,1,0,0
Well.h%=Abs(Ysin-(399-Ysin))
```

'Wellenhöhe errechnen...

```
X.lang%=0
```

'...den Zähler für die einzelnen Bildstreifen auf Null...

St.x%=Xl%

'...linke Randkoordinate doppeln (sie wird noch als Zählerstart
'gebraucht)

Yl%=200-(Blockrows%/2)

'...Block in der Bildschirmmitte placieren...

Yy%=Yl%+Blockrows%

'...untere Randkoordinate ermitteln.

For I=Pi To 3*Pi Step Pi/((Xx%-Xl%)/2)

'Pi-Step-Schleife für Sinus

Inc X.lang%

'Den Streifenzähler +1...

Well.p%=Int(Well.h%*Sin(I)+0.5)

'...die einzelnen Punkte der Welle errechnen...

Yl.sin%=Well.p%+Yl%

'...die neue obere Y-Koordinate relativ zur alten Y-Blockkoordinate
'ermitteln...

Add St.x%,1

'...relative X-Koordinate +1...

Put St.x%,Yl.sin%,Mir.mem\$(X.lang%),Rastermod%

'...und den jeweiligen Bildstreifen auf die neue Position setzen.

```
If X.lang%=Xx%-Xl%
```

```
'Falls die Schleife so oft durchlaufen wurde, wie Anzahl der  
'einzelnen Bildstreifen vorhanden sind...
```

```
I=3*Pi
```

```
'...verlasse die Schleife (Schleifenendwert = 3*Pi).
```

```
Endif
```

```
Next I
```

```
Sinyo%=Yl%-Well.h%
```

```
Sinyu%=Yy%+Well.h%
```

```
'Den neuen oberen und unteren Blockrand errechnen...
```

```
If Sinyo%<0
```

```
    Sinyo%=0
```

```
Endif
```

```
If Sinyu%>399
```

```
    Sinyu%=399
```

```
Endif
```

```
'...und evtl. am Bildschirmrand abschneiden.
```

```
Get Xl%,Sinyo%,Xx%,Sinyu%,Mc.part$
```

```
'Neuen Block in den Rasterpuffer...
```

```
@P.utflip
```

```
'...und aktuelle SCREEN wieder auf den Bildschirm bringen.
```

```
Erase Mir.mem$()
```

```
Return
```

SIN, COS, TAN

"Sin" übergibt den Sinuswert des in Klammern angegebenen RAD-Winkels. Soll der Sinus einer GRAD-Zahl ermittelt werden, ist der Winkel wie folgt anzugeben: (bezogene GRAD-Zahl * $\text{Pi} / 180$)

"Cos" übergibt den Cosinuswert des Winkels nach dem gleichen Muster wie oben beschrieben.

"Tan" übergibt den Tangenswert des Winkels nach gleichem Muster wie oben.

z.B.:

```
For I=0 To 90 Step 15
  Print "Sinus  ";I;" GRAD : ";Sin(I*Pi/180)
  Print "Cosinus ";I;" GRAD : ";Cos(I*Pi/180)
  Print "Tangens ";I;" GRAD : ";Tan(I*Pi/180)
Next I
For I=-Pi+0.0000001 To Pi Step Pi/90
  Print "Sinus  ";I;" RAD : ";Sin(I)
  Print "Cosinus ";I;" RAD : ";Cos(I)
  Print "Tangens ";I;" RAD : ";Tan(I)
Next I
U=Inp(2)
Edit
```

ATN

Durch "Atn" läßt sich der dem angegebenen Tangenswert entsprechende Winkel in RAD ermitteln. Der zurückgegebene Wert liegt zwischen $-\text{Pi}/2$ und $\text{Pi}/2$. Um einen GRAD-Winkel zu erhalten, ist der entsprechende Tangenswert wie folgt anzugeben:

(bezogener Tangens * $180 / \text{Pi}$)

z.B.:

In diesem kleinen Programm werden Sie zuerst zur Eingabe zweier Kathetenlängen aufgefordert. Es sollen die Tangenswerte zu den beiden Hypothenusen/Katheten-Winkeln (A;B) eines rechtwinkligen Dreiecks anhand der Behauptung "Tangens A = b / a"; "Tangens B = a / b" ermittelt werden, aus denen anschließend die entsprechenden GRAD-Winkel errechnet und daran anschließend der Tangens und die Längen der Katheten zurückgerechnet werden. Anschließend wird das entsprechende Dreieck gezeichnet.

```

Do
Cls
Input "Kathetenlänge a :",Kathete.a
Input "Kathetenlänge b :",Kathete.b
A.tangens=Kathete.b/Kathete.a
B.tangens=Kathete.a/Kathete.b
Print "Tangens A : ";A.tangens
Print "Tangens B : ";B.tangens;Chr$(10)
A.winkel=Atn(A.tangens)*180/Pi
B.winkel=Atn(B.tangens)*180/Pi
Print "Winkel A : ";A.winkel
Print "Winkel B : ";B.winkel;Chr$(10)
Print "Zur Kontrolle :";Chr$(10)
Na.tangens=Tan(A.winkel*Pi/180)
Nb.tangens=Tan(B.winkel*Pi/180)
Print "Tangens A neu : ";Na.tangens
Print "Tangens B neu : ";Nb.tangens;Chr$(10)
Print "Summe aller Winkel (180) :";90+A.winkel+B.winkel
Hypothenuse=Sqr(Kathete.a^2+Kathete.b^2)
N.kathete.a=Hypothenuse*Sin(B.winkel*Pi/180)
N.kathete.b=Hypothenuse*Cos(B.winkel*Pi/180)
Print "Hypothenuse : ";Hypothenuse
Print "Kathete a neu : ";N.kathete.a
Print "Kathete b neu : ";N.kathete.b
Plot 620,380
Draw To 620,380-(N.kathete.b) To 620-(N.kathete.a),380 To 620,380
Text 615,380-(N.kathete.b/2),"a"

```

```

Text 620-(N.kathete.a/2),385,"b"
Text 620-(N.kathete.a/2),380-(N.kathete.b/2),"c"
Text 620-Len(Str$(B.winkel))*8,380-N.kathete.b-2,B.winkel
Text 620-N.kathete.a-Len(Str$(A.winkel))*8,380,A.winkel
Flaeche=N.kathete.a*N.kathete.b/2
Text 620-(N.kathete.a/2),380-(N.kathete.b/4),"F: "+Str$(Flaeche)
Repeat
Until Len(Inkey$)
Loop

```

PI

In den vorangegangenen Beispielen wurde des öfteren die Kreiszahl "Pi" eingesetzt. Diese Zahl ist im GfA-BASIC in der reservierten Variablen "Pi" (3.1415926536) enthalten. Diese Variable läßt sich, ebenso wie "Err" und "Fatal", nicht verändern. Zuweisungen werden zwar angenommen, jedoch der Variableninhalt wird dadurch nicht verändert.

Ein kleines "Denk"mal: Der geschichtliche Ursprung dieser Zahl ist nicht genau erforscht worden. Es wird angenommen, daß sie schon vor ca. 3500 Jahren den alten Ägyptern bekannt gewesen ist, da sich aus dem Verhältnis der Seitenneigung zur Grundfläche der meisten ägyptischen Pyramiden dieser Wert ableiten läßt. Auf jeden Fall ist sie eine der genialsten menschlichen Errungenschaften. Ihr Wert ergibt sich außerdem, wenn man einen exakten Kreis zeichnet, seinen Durchmesser anhand eines Fadens festhält und die Länge dieses Fadens an den Kreisumfang anlegt. Der Faden paßt genau 3.1415926536 mal in den Kreisumfang (Umfang=Durchmesser*Pi). Die Neuentdeckung dieser Zahl wird einem Mathematiker des antiken Griechenlands zugesprochen, der dieses Faden- und Kreisspiel das erste Mal spielte. Die Auswirkungen dieser Zahl auf die Entwicklung der menschlichen Kultur sind weitreichender, als man es ihr vom Anschein her zutraut. Die heutige Computertechnologie wäre ohne das Wissen um diese Zahl nicht vorstellbar. Wenn es sie nicht geben würde, müßte man sie schnellstens entdecken.

LOG, LOG10

"Log" gibt Ihnen den natürlichen (Neper'schen) Logarithmus des in Klammern angegebenen numerischen Ausdrucks zurück. (Basis "e" => 2.718281828 => Euler'sche Zahl)

"Log10" gibt Ihnen den dekadischen (Briggs'schen) Logarithmus des in Klammern angegebenen numerischen Ausdrucks zurück. (Basis 10)

z.B.:

A=322.571

Print Log(A)'''Log10(43.883)

Ausgabe: 5.776323267 1.6422963099

DEFFN FN

Eine sehr komfortable Möglichkeit, mathematische oder alphanumerische Funktionen kompakt zu definieren, bietet dieser Befehl. Ihm kann optional (wahlfrei; "entweder so, oder so") eine Liste von Variablenamen in Klammern angefügt werden, deren Inhalte innerhalb der Funktion verarbeitet werden können. Diese Parameterliste kann ohne weiteres Variablen unterschiedlicher Typen (Real, Integer, String etc.) beinhalten. Innerhalb der Funktion sind dann allerdings nur Operationen erlaubt, die dem verwendeten Funktionstypen entsprechen. D.h., wenn die Funktion als Stringtyp deklariert wurde (Variablenname + angehängtes "\$"), können nur Stringoperationen ausgeführt werden. Bei numerischen Funktionstypen können folglich nur numerische Operationen angewandt werden.

In der Wahl des Funktionsnamens werden Ihnen keinerlei Einschränkungen auferlegt. Sie können auch BASIC-Befehlsnamen oder Namen schon existierender Variablen verwenden. Dies ist möglich, da der Funktionsname untrennbar mit der unter ihm definierten Funktion in Bezug steht. Die Verwendung der Namen von Feldvariablen ist allerdings nicht gestattet. Haben Sie eine Parameterliste vorgesehen, müssen beim Funktionsaufruf

dementsprechend viele und dem jeweiligen Variablentyp entsprechende Werte, Strings, andere Variablen oder Ausdrücke übergeben werden. Entsprechen die übergebenen Parametertypen nicht der Liste oder werden zuviele, bzw. zuwenige Parameter übergeben, erscheint eine Fehlermeldung. In der Funktion müssen die angegebenen Variablen nicht verwendet werden (Dummies), und es können natürlich Variablen in der Funktion verwendet werden, die nicht in der Parameterliste stehen. Beim Aufruf der Funktion werden die zu dem Zeitpunkt aktuellen Inhalte der darin verwendeten Variablen angenommen und weiterverarbeitet.

Die Länge einer Funktion wird durch die maximale Eingabezeilenlänge (256 Zeichen) beschränkt. Reicht dieser Platz nicht aus, um eine Funktion vollständig zu definieren, können aus einer Funktion heraus andere Funktionen aufgerufen werden, d.h., daß man Funktionen beliebig miteinander verketteten kann. Eine Funktion kann also ohne weiteres ausschließlich aus Funktionsaufrufen und deren Verknüpfungen bestehen! Die Inhalte einer Funktion können entweder über einen Ausgabe-Befehl (Print, Write, Text, Out) ausgegeben, durch eine Zuweisung einer dem Funktionstyp entsprechenden Variablen übergeben oder z.B. in Bedingungsabfragen integriert werden. Nur der Funktionsaufruf ohne einen dieser weiterverarbeitenden Operationen ist nicht möglich, da das BASIC dann annimmt, es handle sich um einen Prozedur-Aufruf und die Fehlermeldung "Procedure nicht gefunden" ausgibt. Dem Interpreter ist es völlig egal, an welcher Programmstelle Sie eine Funktion definieren oder ob der Funktionsaufruf vor der Definition steht. Zu Beginn des Programmlaufs durchsucht der Interpreter den Programmtext nach evtl. vorhandenen "Defn". Der Funktionsinhalt ist dem Interpreter vor Ausführung der ersten Zeile bereits bekannt. Es ist also möglich, eine Funktion zu Programmbeginn aufzurufen, deren Definition erst in der letzten Programmzeile steht.

"Fn" bzw. das Kürzel "@" ist der entsprechende Funktionsaufruf. Bei einer evtl. vorhandenen Parameterliste sind diesem Aufruf die von der Funktion erwarteten Werte oder Strings, bzw. deren Variablen in Klammern anzufügen. Wie oben erwähnt, ist dabei

darauf zu achten, daß diese Angaben den in der Parameterliste aufgeführten Variablentypen entsprechen.

z.B.:

Es soll der gemeinsame Schwerpunkt mehrerer Flächen ermittelt werden. Die einzelnen Flächen sind Rechtecke verschiedener Größe, die beliebig auf dem Bildschirm placiert werden können. Die Anzahl der Flächen kann beliebig groß sein. Zuerst erscheint ein kleiner Kreis. Dieser stellt den Nullpunkt dar, auf den sich anschließend die Koordinatenangaben beziehen. Placieren Sie ihn an einer beliebigen Position und drücken Sie die Maustaste einmal kurz. Der Nullpunkt ist fixiert. Zu Anfang einer jeden Rechteckdefinition erscheinen die zum Nullpunkt relativen Koordinaten, sowie die Nummer der aktuellen Fläche. Drücken Sie die linke Maustaste, wird die aktuelle Mausposition als Ausgangspunkt für die Rechteckbestimmung angenommen. Halten Sie die Maustaste gedrückt und bestimmen Sie die Größe der Fläche. An Mauszeigerposition werden nun die Breite und die Höhe der Fläche angezeigt. Wenn das Rechteck Ihren Vorstellungen entspricht, lassen Sie die Maustaste los und die Fläche wird gezeichnet. Mit einem Klick auf die rechte Maustaste beenden Sie die Eingabe und der gemeinsame Schwerpunkt der bisher eingegebenen Flächen wird gezeichnet und seine zum Nullpunkt relativen X/Y-Koordinaten werden angezeigt. Drücken Sie nun eine Maustaste, können Sie den nächsten Flächensatz eingeben oder das Programm beenden.

Beispiel 1:

```
Do  
  Cls  
  Clear  
  Deffill ,2,2
```

```

I=0
Graphmode 3
Defline 1,3
Repeat
  Mouse Xnull,Ynull,K
  Circle Xnull,Ynull,7
  Circle Xnull,Ynull,7
Until Mousek
Graphmode 2
Defline 1,1
Circle Xnull,Ynull,7
Text Xnull+8,Ynull-8,"Nullpkt.(0/0)"
Pause 20
Do
  Graphmode 3
  Inc I
  Repeat
    Mouse X,Y,K
    Text X,Y,Str$(X-Xnull)+"/"+Str$(Ynull-Y)+" (" +Str$(I)+")"
    Text X,Y,Str$(X-Xnull)+"/"+Str$(Ynull-Y)+" (" +Str$(I)+")"
  Until Mousek
  Repeat
    Mouse Xx,Yy,Kk
    Box X,Y,Xx,Yy
    Box X,Y,Xx,Yy
    Text Xx,Yy,Str$(Abs(Xx-X))+"/"+Str$(Abs(Yy-Y))
    Text Xx,Yy,Str$(Abs(Xx-X))+"/"+Str$(Abs(Yy-Y))
  Until Kk<>1
  Exit If Kk=2
  Breite=Abs(Xx-X)
  Hoehe=Abs(Yy-Y)
  X.diff=Min(X,Xx)-Xnull
  Y.diff=Ynull-Max(Y,Yy)
  Graphmode 2
  Xlo=@X.lo
  Ylo=@Y.lo
  Xru=@X.ru
  Yru=@Y.ru
  Pbox Xlo,Ylo,Xru,Yru
  Defline 1,1

```



```

Line Xlo,Ylo,Xru,Yru
Line Xlo,Yru,Xru,Ylo
F.laeche=@Flaeche
Add Areas,F.laeche
X.halbe=@Halbe(Breite)+X.diff
Y.halbe=@Halbe(Hoehe)+Y.diff
Add An.xn,@Multi(X.halbe)
Add An.yn,@Multi(Y.halbe)
Loop
Graphmode 1
Sx=An.xn/Areas
Sy=An.yn/Areas
Defline 4,1
Line 0,Ynull-Sy,640,Ynull-Sy
Line Xnull+Sx,0,Xnull+Sx,399
Defline ,3
Circle Xnull+Sx,Ynull-Sy,7
Print At(1,1);"SX : ";Sx
Print At(1,2);"SY : ";Sy
Repeat
  If Mousek
    Alert 2,"Nächste Eingabe ?      ",1,"JA|NEIN|ENDE",Antwort
  Endif
  If Antwort=3
    Edit
  Endif
Until Antwort=1
Loop
Deffn Flaeche=Hoehe*Breite
Deffn Halbe(Seite)=Seite/2
Deffn Multi(H.seite)=H.seite*F.laeche
Deffn X.lo=Xnull+X.diff
Deffn Y.lo=Ynull-Hoehe-Y.diff
Deffn X.ru=Xnull+X.diff+Breite
Deffn Y.ru=Ynull-Y.diff

```


Beispiel 2:

```

For I=4 Downto 0
  If @A=69
    Print "ENDE"
    Pause 100
    Edit
  Else
    On I Gosub 1,2,3,4
    Print Bb$
    Print @B$(Bb$)
    Print @A
  Endif
Next I
Procedure 1
  Bb$="EINS"
Return
Procedure 2
  Bb$="ZWEI"
Return
Procedure 3
  Bb$="DREI"
Return
Procedure 4
  Bb$="VIER"
Return
Defn B$(B.b$)=B.b$+Str$(@A)
Defn A=34*I+Asc(Bb$)

```

In den folgenden beiden Procedures erfolgt die Anpassung der Polyline-Koordinaten an die aktuelle Patch- bzw. Rubbgröße.

Procedure D.efpatch

```

Plnx(0)=X-Halbe%
Plny(0)=Y-Halbe%
Plnx(1)=X+Halbe%
Plny(1)=Y-Halbe%
Plnx(2)=X+Halbe%
Plny(2)=Y+Halbe%
Plnx(3)=X-Halbe%

```

```

Plny(3)=Y+Halbe%
Plnx(4)=X-Halbe%
Plny(4)=Y-Halbe%
Plnx(5)=X+Halbe%
Plny(5)=Y-Halbe%

```

Return

Procedure D.ef.patch.2

```

Pl.nxx(0)=Xl%
Pl.nyy(0)=Yl%
Pl.nxx(1)=Xx%
Pl.nyy(1)=Yl%
Pl.nxx(2)=Xx%
Pl.nyy(2)=Yy%
Pl.nxx(3)=Xl%
Pl.nyy(3)=Yy%
Pl.nxx(4)=Xl%
Pl.nyy(4)=Yl%
Pl.nxx(5)=Xx%
Pl.nyy(5)=Yl%

```

Return

Procedure Verteiler

|

'Dieses ist die Hauptverteiler-Schleife.

'Durch sie werden alle Grafikoperationen und Abfragen gesteuert.

|

Do

@Getfkeys

|

'Zuerst wird in "Getfkeys" nachgefragt, ob eine Taste gedrückt wurde.

'"Getfkeys" ist die Abfrage-Prozedur, von der aus die Menüaufrufe, Einstellungsveränderungen und Sonderfunktionen verteilt werden.

|

Halbe%=Patchsize%/2

|

'"Patchsize%" wird hier halbiert, damit der Mauszeiger im Patchmodus in der Mitte der Figur erscheint.

|

Spr.x=Random(2*Halbe%)

Spr.y=Random(2*Halbe%)

```
'Die Koordinaten der Sprühpunkte werden zufällig gewählt und in die
'aktuelle Patchgröße eingepaßt.
'
Mirflag%=1
Mouse X,Y,K
'
'Dieses sind die Mauskoordinaten, die in den meisten
'Grafikoperationen als Ausgangspunkt dienen.
'
If Border%=1
'
'Falls der "BORDER"-Modus an ist...
'
If X<1 And Y<300 And Y>100
'
'..und mit dem Mauszeiger der linke Bildschirmrand berührt wird,
'
Click%=0
@Clicks
'
'warte auf eine Maustastenbedienung...
'
'In der Prozedur "Clicks" wird eine GEM-Funktion verwendet,
'durch
'die es möglich ist, das Programm abhängig von der Anzahl der
'Mausklicks zu steuern. Sie kennen diese Funktion vom Desktop,
'wenn Sie ein Programm starten oder eine Datei zum Lesen öffnen
'wollen. Dazu müssen Sie den berühmten Doppelklick ausführen.
'Genau dieser Doppelklick kann mit dieser GEM-Funktion abgefragt
'werden.
'Wird bestimmt, daß nur bei einem Doppelklick etwas geschehen
'soll, wird das Programm bei einem einfachen Mausklick einfach
'fortgesetzt.
'
If Click%=2
@Alterscreen
'
'...und verzweige gegebenenfalls zur Alternativ-SCREEN-
'Anzeige.
```

```

    '
    Endif
Endif
If X<1 And Y>398
    '
    'Wird die linke untere Bildschirmecke berührt...
    '
    Click%=0
    @Clicks
    If Click%=2
        Inc Clou
        If Clou=1
            R.emem=D.rawmode%
            D.rawmode%=10
            '
            '...merke Dir bei einem Doppelklick beim ersten Aufruf den
            'gerade eingestellten Grafik-Operations-Modus, und schalte
            '"SELF" ein.
            '
        Endif
        If Clou=2
            Clou=0
            D.rawmode%=R.emem
            '
            'Wird diese Ecke das nächste Mal berührt und doppelt
            'geklickt, schalte die zuletzt gemerkte Grafikoperation
            'wieder ein.
            'Beim nächsten Mal wird wieder "SELF" aufgerufen usw...
            '
        Endif
    Endif
Endif
If X>638 And Y<300 And Y>100
    '
    'Wird der rechte Bildschirmrand im mittleren Bereich berührt...
    '
    Click%=0
    @Clicks
    If Click%=2
        '
    Endif
Endif
```

```
'...und doppelt geklickt...
'
If Flipp.flag%=0
    Get 0,0,639,399,Screen.eins$
    @Starter
    Put 0,0,Screen.eins$,3
Endif
If Flipp.flag%=1
    Get 0,0,639,399,Screen.zwei$
    @Starter
    Put 0,0,Screen.zwei$,3
Endif
'
'...bringe die aktuelle SCREEN in den ihr zustehenden Raster-
'speicher und rufe den Editor auf (@Starter). Nach Editor-Ende
'bringe die zuletzt aktuelle SCREEN wieder auf den Bildschirm.
'
Endif
Endif
If X>638 And Y>398
'
'Ist die rechte untere Bildschirmecke berührt...
'
Click%=0
@Clicks
If Click%=2
'
'...und ein Dippelklock, ääh Doppelklick ausgeführt worden...
'
Inc Koordi
If Koordi=1
    Showkoordi=1
'
'...schalte das Koordinaten-Anzeige-Flag beim ersten Mal
'auf "On"...
'
Endif
If Koordi=2
    Koordi=0
    Showkoordi=0
```

```

        '
        '...und beim nächsten Mal auf "Off" usw...
        '
    Endif
Endif
Endif
If X>270 And X<370 And Y>398
    '
    'Wurde der untere Bildschirmrand im mittleren Bereich berührt...
    '
    Click%=0
    @Clicks
    If Click%=2
        @Showhelp
        '
        '...und doppelt geklickt, rufe das Grafik-Einstellungsmenü
        'auf.
        '
    Endif
Endif
Endif
If Showkoordi=1
    '
    'Ist das Koordinaten-Anzeige-Flag gesetzt, zeige die Koordinaten
    'an.
    '
    Deftext 1,1,0,13
    Graphmode 3
    Text 3,15,Str$(X)+" / "+Str$(Y)
    Text 3,15,Str$(X)+" / "+Str$(Y)
    @Set.paras
Endif
If Netzmod%=1
    '
    'Ist der "NETZMODUS" an, rechne die Mauskoordinaten in das
    'entsprechende Raster um.
    '
    X=Int(X/Netzweite%)*Netzweite%
    Y=Int(Y/Netzweite%)*Netzweite%
Endif

```

```

If Showlines%=1
    '
    'Ist "FADENKREUZ MOBIL" an, zeige es ständig an der aktuellen
    'Mausposition.
    '
    @Clipprec(2)
    @S.howlines
    @Set.paras
Endif
If Const.lines%=1
    '
    'Ist "FADENKREUZ FIX" eingeschaltet, zeige es ständig an der
    'definierten Position.
    '
    @Clipprec(2)
    @C.onst.lines
    @Set.paras
Endif
If (Areas%=1 And K=1) Or Bl.load%=1
    '
    'Ist eine der beiden Raster-Funktionen (MOVE/COPY) eingeschaltet
    'oder wurde ein Block von Diskette geladen, gehe zur Block-
    'Prozedur.
    '
    Bl.load%=0
    @Areaprocs
Endif
If Areas%=0
    '
    'Ist weder MOVE noch COPY eingeschaltet, gehe zu der eingestellten
    'Grafikoperation.
    '
    On D.rawmode% Gsub F.srecp,L.srecp,A.srecp,F.rrecp,L.rrecp,A.rrec
p,F.cp,L.cp,A.cp,Icomp,F.srecr,L.srecr,A.srecr,F.rrecr,L.rrecr,A.rrecr,F
.cr,L.cr,A.cr,S.lin,F.illpro,Pl.spr,Pt.spr,Marker,T.ext
Endif

```

ON GOSUB

Mit diesem Befehl lassen sich mehrere Prozeduren zu eine Liste zusammenfassen, die dann je nach Inhalt des numerischen Ausdrucks oder der Variablen, der/die an "On Gosub" übergeben werden muß, aufgerufen werden. Der Befehl "On A Gosub 1,2,3" ersetzt also gleichzeitig drei If-Abfragen, in denen man zuerst den Inhalt von "A" erfragen müßte, um dementsprechend zu der gewünschten Prozedur verzweigen zu können. Diese If-Abfragen würden dann so aussehen:

```
If A = 1
Gosub 1
Endif
If A = 2
Gosub 2
Endif
If A = 3 ,
  Gosub 3
Endif
```

"On Gosub" kann seine Entscheidung darüber, zu welcher Prozedur es verzweigen soll, natürlich nur in Einerschritten ab 1 aufwärts treffen. Eine "A=0"-Verzweigung ist also nicht möglich und eine Werte-Staffelung, die bei größer 1 beginnt, ebensowenig. Dieser Befehl verzweigt grundsätzlich in der Reihenfolge ab 1 aufwärts zu den in der Liste angegebenen Prozeduren. Die Verzweigung erfolgt also nach folgendem Schema:

```
Wert  >= 1 < 2 ==> 1. Prozedur der Liste
Wert  >= 2 < 3 ==> 2. Prozedur der Liste
usw.
```

Tritt ein Wert, der größer als die Anzahl der angegebenen Prozeduren oder der Wert Null auf, wird das Programm in der auf "On Gosub" folgenden Programmzeile fortgesetzt.

z.B.

```

Do
  A=Random(4)
  On A GOSUB 1,2,3
Loop
Procedure 1
  Print "Proc 1"
Return
Procedure 2
  Print "Proc 2"
Return
Procedure 3
  Print "Proc 3"
Return

```

```

If Mirror%>0 And Areas%=0

```

```


```

```

'Ist eine der "SPIEGEL"-Funktionen eingeschaltet und weder MOVE
'noch COPY an, gehe zu der entsprechenden Spiegel-Prozedur.

```

```


```

```

Mirflag%=0

```

```

On Mirror% GOSUB Mir.1,Mir.2,Mir.3,Mir.4

```

```

Endif

```

```

Norubb%=0

```

```

Loop

```

```

Return

```

```


```

```

'Die folgenden vier Prozeduren dienen der Ermittlung der jeweiligen
'Spiegelkoordinaten. Wurden diese ermittelt, werden die jeweiligen
'Grafikoperationen erneut zum Zeichnen der Spiegelfiguren aufgerufen.

```

```


```

```

Procedure Mir.1

```

```

  X=639-X

```

```

  Xl%=639-Xl%

```

```

  Xx%=639-Xx%

```

```

  @D.efpatch

```

```

  Rubbflag%=1

```

```

On D.rawmode% Gosub F.srecp,L.srecp,A.srecp,F.rrecp,L.rrecp,A.rrecp,F.
cp,L.cp,A.cp,Iconp,F.srecr,L.srecr,A.srecr,F.rrecr,L.rrecr,A.rrecr,F.cr,
L.cr,A.cr,S.lin,F.fillpro,Pl.spr,Pt.spr,Marker,T.ext

```

```
Return
```

```
Procedure Mir.2
```

```
Y=399-Y
```

```
Yl%=399-Yl%
```

```
Yy%=399-Yy%
```

```
@D.efpatch
```

```
Rubbflag%=1
```

```

On D.rawmode% Gosub F.srecp,L.srecp,A.srecp,F.rrecp,L.rrecp,A.rrecp,F.
cp,L.cp,A.cp,Iconp,F.srecr,L.srecr,A.srecr,F.rrecr,L.rrecr,A.rrecr,F.cr,
L.cr,A.cr,S.lin,F.fillpro,Pl.spr,Pt.spr,Marker,T.ext

```

```
Return
```

```
Procedure Mir.3
```

```
X=639-X
```

```
Y=399-Y
```

```
Xl%=639-Xl%
```

```
Xx%=639-Xx%
```

```
Yl%=399-Yl%
```

```
Yy%=399-Yy%
```

```
@D.efpatch
```

```
Rubbflag%=1
```

```

On D.rawmode% Gosub F.srecp,L.srecp,A.srecp,F.rrecp,L.rrecp,A.rrecp,F.
cp,L.cp,A.cp,Iconp,F.srecr,L.srecr,A.srecr,F.rrecr,L.rrecr,A.rrecr,F.cr,
L.cr,A.cr,S.lin,F.fillpro,Pl.spr,Pt.spr,Marker,T.ext

```

```
Return
```

```
Procedure Mir.4
```

```
@Mir.1
```

```
@Mir.2
```

```
@Mir.1
```

```
Return
```

```
Procedure Getfkeys
```

```
!
```

```
'Hauptabfrage-Prozedur
```

```
!
```

```
@Clipprec(2)
```

```
Graphmode 1
```

```
Mouse X,Y,K
```

```
Helpin%=0
```

```
'
'Menüindex auf Null setzen
'
If Y<50 And Help.on%=1 And K>0
'
' Ist das Grafikmenü auf und der Mauszeiger über ihm und wird
' gleichzeitig eine Maustaste gedrückt...
'
Helpin%=Int(X/64)+1
'
' ...errechne nach altbekanntem Muster den Index des Menüfeldes, über
' dem sich der Mauszeiger befindet.
'
Endif
Keyget$=Inkey$
'
'Tastaturabfrage
'
Fkey.1%=Asc(Right$(Keyget$))
Fkey.2%=Asc(Left$(Keyget$))
'
'Die beiden mit Inkey$ möglichen Zeichen splitten.
'
Fkeys%=Fkey.1%+Fkey.2%
If Fkeys%>0 Or Help.on%>0
'
' Wenn eine Taste gedrückt wurde oder das Grafikmenü offen ist...
'
Repeat
Until Inkey$=""
'
' ...leere erstmal den Tastaturspeicher.
'
If Fkeys%=54 And Help.on%=0
'
' Ist das Grafikmenü geschlossen und wird die "Esc"-Taste gedrückt,
' rufe das Hauptmenü auf.
'
@Mainmenue
Endif
```

```
If (Fkeys%=59 And Fkey.1%=59) Or Helpin%=1
'
'Wurde F1 gedrückt oder das erste Menüfeld angeklickt, gehe zur
'Füllmusterindex-Incrementierung,
'
@Uppattern
Endif
If (Fkeys%=60 And Fkey.1%=60) Or Helpin%=2
'
'F2 gedrückt oder 2. Feld angeklickt ==> Füllmuster-
'Decrementierung
'
@Downpattern
Endif
If (Fkeys%=61 And Fkey.1%=61) Or Helpin%=3
'
'F3 gedrückt oder 3. Feld angeklickt ==> Liniendicke-
'Incrementierung
'
@Updick
Endif
If (Fkeys%=62 And Fkey.1%=62) Or Helpin%=4
'
'F4 gedrückt oder 4. Feld angeklickt ==> Liniendicke-
'Decrementierung
'
@Downdick
Endif
If (Fkeys%=63 And Fkey.1%=63) Or Helpin%=5
'
'F5 gedrückt oder 5. Feld angeklickt ==> Linienstil-Auswahl
'
@Lnstyle
Endif
If (Fkeys%=64 And Fkey.1%=64) Or Helpin%=6
'
'F6 gedrückt oder 6. Feld angeklickt ==> Linienendformen-Auswahl
'
@Lnform
Endif
```

```
If (Fkeys%=65 And Fkey.1%=65) Or Helpin%=7
'
'F7 gedrückt oder 7. Feld angeklickt ==> Patchgrößen-
'Incrementierung
'
@Uppatch
Endif
If (Fkeys%=66 And Fkey.1%=67) Or Helpin%=8
'
'F8 gedrückt oder 8. Feld angeklickt ==> Patchgrößen-
'Decrementierung
'
@Downpatch
Endif
If (Fkeys%=67 And Fkey.1%=67) Or Helpin%=9 Or (Fkeys%=18 And Help.on
%=1)
'
'F9 gedrückt oder 9. Feld angeklickt ==> Auswahl der
'Grafikoperationen
'
@Setdrawmode
Endif
If (Fkeys%=68 And Fkey.1%=68) Or Helpin%=10
'
'F10 gedrückt oder 10. Feld angeklickt ==> Grafikmodus-Auswahl
'
@Setgrafmode
@Changemod
Endif
If Help.on%=1 And K=1
'
'Setze Schrittweite zur Patchgrößen-Einstellung fest.
'
@Setsteps
Endif
If Fkeys%=18 And Help.on%=0
'
'Ist das Grafikmenü geschlossen und wird die "Tab"-Taste gedrückt,
'öffne das Grafikmenü.
'
```

```

    @Showhelp
Endif
If D.rawmode%=25 And Areas%=0
    '
    'Ist "TEXT" eingeschaltet, frage die Cursortasten nach
    'Veränderungen der Textlänge durch die vier Cursortasten.
    '
    @T_extlen
Endif
If Fkeys%=98 And Fkey.1%=98
    '
    'Wird die "Help"-Taste gedrückt, zeige den aktuellen Infotext.
    '
    @H.elptext
Endif
If Fkeys%=71 And Help.on%=0
    '
    'Ist das Grafikmenü aus und wurde die "ClrHome"-Taste gedrückt,
    'frage nach, ob das aktuelle Bild gelöscht werden soll, und lösche
    'es gegebenenfalls.
    '
    Alert 2,"BILD LOESCHEN ??",2," OKAY | NEIN ",Backalert%
    If Backalert%=1
        Zufall%=Random(2)
        For I=1 To 26
            If Zufall%=0
                Print At(1,25);Chr$(13)
            Else
                Print At(1,1);Chr$(27);"I"
            Endif
        Next I
        If Flipp.flag%=0
            Get 0,0,639,399,Screenfix.eins$
            Get 0,0,639,399,Screen.eins$
        Else
            Get 0,0,639,399,Screenfix.zwei$
            Get 0,0,639,399,Screen.zwei$
        Endif
    Endif
Endif
Endif

```

```
If Fkeys%=97 And Help.on%=0
```

```
,
```

```
'Ist das Grafikmenü aus und wurde die "Undo"-Taste gedrückt, frage  
'nach, ob die seit der letzten "Insert"-Tastenbedienung gezeichnete  
'Sequenz gelöscht werden soll, und lösche sie gegebenenfalls.
```

```
,
```

```
Alert 2,"LETZTE SEQUENZ| LÖSCHEN ?",2," OKAY | NEIN ",Backalert%
```

```
If Backalert%=1
```

```
  If Flipp.flag%=0
```

```
    Put 0,0,Screenfix.eins$,3
```

```
    Get 0,0,639,399,Screen.eins$
```

```
  Else
```

```
    Put 0,0,Screenfix.zwei$,3
```

```
    Get 0,0,639,399,Screen.zwei$
```

```
  Endif
```

```
Endif
```

```
Endif
```

```
If Fkeys%=82 And Help.on%=0
```

```
,
```

```
'Ist das Grafikmenü aus und wurde die "Insert"-Taste gedrückt,  
'frage nach, ob das aktuelle Bild gespeichert werden soll, und  
'speichere es gegebenenfalls.
```

```
'Dieses Grafikprogramm verfügt über zwei Speicher zu jeder SCREEN.
```

```
'In dem einen Speicher werden SCREENs nur abgespeichert, wenn sie
```

```
'als erhaltenswert befunden wurden. Ist das geschehen, kann
```

```
'anschließend beliebig auf dem Bild "herumgepfuscht" werden.
```

```
'Gefällt Ihnen das nach der Fixierung gezeichnete Bild nicht,
```

```
'drücken Sie einfach die "Undo"-Taste, und das zuletzt fixierte
```

```
'Bild wird wieder sauber dargestellt. Dieses "Tafelputzen"
```

```
'ermöglicht Ihnen weitgehenden Spielraum in der Gestaltung der
```

```
'Grafik. Sie können so verschiedene Füllmuster, Formen
```

```
'und Grafikeffekte ausprobieren, ohne das schon fixierte, darunter
```

```
'befindliche Bild zu zerstören.
```

```
,
```

```
Alert 2,"AKTUELLES BILD| FIXIEREN ?",2," OKAY | NEIN ",Backalert%
```

```
If Backalert%=1
```

```
  If Flipp.flag%=0
```

```
    Get 0,0,639,399,Screenfix.eins$
```

```
    Get 0,0,639,399,Screen.eins$
```

```
  Else
```

```

        Get 0,0,639,399,Screenfix.zwei$
        Get 0,0,639,399,Screen.zwei$
    Endif
Endif
Endif
Endif
@Set.paras
'
'Nach aller Anstrengung, die ein Menüaufruf mit sich bringt, werden
'zum Schluß die vom Anwender definierten Operationsmodi und
'Einstellungen restauriert.
'
Return
Procedure T_extlen
'
'Prozedur zur Einstellung der Textlänge
'
If Fkeys%=72
    T_grow%(0)=0
Endif
If Fkeys%=75 Or Fkeys%=77 Or Fkeys%=80
    T_grow%(0)=T_grow%(1)
    If Fkeys%=75
        Sub T_grow%(0),4
        If T_grow%(0)<0
            Add T_grow%(0),8
        Endif
    Endif
    If Fkeys%=77
        Add T_grow%(0),4
        If T_grow%(0)<0
            Sub T_grow%(0),8
        Endif
    Endif
    If Fkeys%=80
        Inc T_sign%(0)
        If T_sign%(0)=2
            T_sign%(0)=0
            T_grow%(0)=Abs(T_grow%(0))
        Endif
    Endif
Endif

```



```

    If T_sign%(0)=1
        T_grow%(0)=T_grow%(0)*(-1)
    Endif
Endif
T_grow%(1)=T_grow%(0)
Endif
Return
Procedure C.const.lines
'
'Prozedur zur Darstellung des "FADENKREUZ FIX"
'
Defline 0,1,0,0
Color Lncol%
Deftext 1,1,0,6
Graphmode 3
Text Hor.ko%,Ver.ko%+8,Co.txt$
Text Hor.ko%,Ver.ko%+8,Co.txt$
Line 0,V.line%,639,V.line%
Line H.line%,0,H.line%,399
Line 0,V.line%,639,V.line%
Line H.line%,0,H.line%,399
Return
Procedure S.howlines
'
'Prozedur zur Darstellung des "FADENKREUZ MOBIL"
'
Defline 0,1,0,0
Color Lncol%
Graphmode 3
Line 0,Y,639,Y
Line X,0,X,399
Line 0,Y,639,Y
Line X,0,X,399
Return
Procedure Set.paras
'
'Prozedur zur Restauration der Einstellungen
'
Graphmode Grafmod%
Defline Lnstil%,Lndick%,Lnstart%,Lnende%

```

```

Deffill ,Fstil%,Fmuster%
Color Lncol%
Dextext T.extvar(1),T.extvar(2),T.extvar(3),T.extvar(4)
@Clipprec(C.lipp%)
Return
Procedure Mainmenue
  '
  'Vorbereitung zum Hauptmenü-Aufruf
  '
  Garbage%=Fre(9)
  @G.etflip
  Get 0,0,319,199,F.screen$(0)
  Get 320,0,639,199,F.screen$(1)
  Get 0,200,319,399,F.screen$(2)
  Get 320,200,639,399,F.screen$(3)
  @Grafmenue
Return
  '
  'Es folgen 11 Prozeduren zur Einstellung der gewünschten
  'Grafikmenüfunktionen
  'Diese werden jeweils dann aufgerufen, wenn vom Anwender entweder die
  'entsprechende Funktionstaste (F1-F10) gedrückt oder das entsprechende
  'Menüfeld im offenen Grafikmenü angeklickt wurde.
  '
Procedure Uppattern
  Inc Fmuster%
  If (Fmuster%=25 And Fstil%=2) Or (Fmuster%=12 And Fstil%=3) Or
(Fmuster%=1 And Fstil%=0)
    Add Fstil%,1
    If Fstil%=1
      Fstil%=2
    Endif
    Fmuster%=1
  Endif
  If Fstil%=4 And Fmuster%>1
    Fstil%=0
    Fmuster%=0
  Endif
  If Help.on%=1
    Deffill ,Fstil%,Fmuster%

```

```
Pbox 14,62,123,107
Pause 6
Endif
Return
Procedure Downpattern
Dec Fmster%
If Fmster%<=0
Dec Fstil%
If Fstil%<0
Fstil%=4
Fmster%=2
Endif
If Fstil%=1
Fstil%=0
Endif
If Fstil%=3
Fmster%=11
Endif
If Fstil%=2
Fmster%=24
Endif
Endif
If Help.on%=1
Deffill ,Fstil%,Fmster%
Pbox 14,62,123,107
Pause 6
Endif
Return
Procedure Updick
Add Lndick%,2
If Lndick%>40
Lndick%=40
Endif
If Help.on%=1
Defline 0,Lndick%,0,0
Pause 2
Color 1
Line 145,84,244,84
Endif
Return
```

Procedure Downdick

Sub Lndick%,2

If Lndick%<0

Lndick%=0

Endif

If Help.on%=1

Pause 2

Deffill ,0,0

Pbox 140,62,249,107

Defline 0,Lndick%,0,0

Color 1

Line 145,84,244,84

Endif

Return

Procedure Lnstyle

Lncol%=1

Color Lncol%

Inc Lnstil%

If Lnstil%=7

Lnstil%=0

Endif

If Lnstil%=0

Lncol%=0

Color Lncol%

Endif

If Help.on%=1

Deffill ,0,0

Pbox 266,62,312,107

Defline Lnstil%,1,0,0

Line 269,84,309,84

Pause 6

Endif

Return

Procedure Lnform

Inc Lnstart%

If Lnstart%=3

Inc Lnende%

Lnstart%=0

If Lnende%=3

Lnende%=0

```
Endif
Endif
If Help.on%=1
  Deffill ,0,0
  Pbox 329,62,375,107
  Color 1
  Defline 1,6,Lnstart%,Lnende%
  Line 334,84,370,84
Endif
Pause 6
Return
Procedure Setsteps
  Graphmode 2
  Deftext 1,1,0,6
  Text 398,73,99,"1 5 10 20 50"
  If X>385 And X<500 And Y>60 And Y<77
    Graphmode 3
    Instep%=0
    Instep%=Int(X/23)-16
    If Instep%=1
      Schrittweite%=Instep%
      Text 398,73,99," 5 10 20 50"
    Endif
    If Instep%=2
      Schrittweite%=5
      Text 398,73,99,"1 10 20 50"
    Endif
    If Instep%=3
      Schrittweite%=10
      Text 398,73,99,"1 5 20 50"
    Endif
    If Instep%=4
      Schrittweite%=20
      Text 398,73,99,"1 5 10 50"
    Endif
    If Instep%=5
      Schrittweite%=50
      Text 398,73,99,"1 5 10 20 "
    Endif
  Endif
Endif
```

Return

Procedure Uppatch

Add Patchsize%,Schrittweite%

If Patchsize%>650

Patchsize%=650

Endif

If Help.on%=1

Graphmode 1

Deftext 1,16,0,20

Text 426,100,0,Str\$(Patchsize%)+ " "

Pause 4

Endif

Return

Procedure Downpatch

Sub Patchsize%,Schrittweite%

If Patchsize%<0

Patchsize%=0

Endif

If Help.on%=1

Graphmode 1

Deftext 1,16,0,20

Text 426,100,0,Str\$(Patchsize%)+ " "

Pause 4

Endif

Return

Procedure Setdrawmode

Icnx%=10

Icny%=10

,

'Dieses Variablenpaar beinhaltet die X- und Y-Position des "SELF"-

'Icons. Ist dieses Icon während der Arbeit "verlorengegangen" (aus dem

'Bildschirmbereich hinausgeschoben und nicht wiedergefunden), kann es

'nach einem erneuten "SELF"-Aufruf im "Fundbüro" (Icnx%=10,Icny%=10 =>

'linke obere Bildschirmecke) abgeholt werden.

,

Clou=0

If Help.on%=1 And Icnflag%=1

,

'Dieses Programm bietet die Möglichkeit für "Schnellader" auf das

'grafische Grafikoperationsmenü (Menü mit kleinen Bildern) zu
 'verzichten.
 'Das Programm fragt ganz zu Anfang nach, ob sich die Grafikmenü-
 'Icons
 '("M_icons") auf der Diskette befinden. Tun sie das nicht, kann das
 'Grafik-Menü trotzdem bedient werden. Dazu müssen Sie in das
 'Menüfeld
 '"Draw Modus" klicken. Klicken Sie mit der linken Taste, wird die
 'nächste
 'Operation eingeschaltet, bei Klick mit der rechten Taste, die vor
 'der
 'aktuellen Operation liegende. Die Reihenfolge wird Ihnen schnell in
 '"Fleisch und Blut" übergehen. Es ist übrigens dieselbe, in der die
 'kleinen
 'Bildchen im "Draw Modus"-Menü (falls vorhanden) gesetzt werden.
 'Sollte Ihnen kein Satz Menü-Icons zur Verfügung stehen, können Sie
 'sich diese Bildchen auf einfache Weise selbst malen.
 'Dazu benötigen Sie zuerst einen Ordner auf Ihrer Programmdiskette,
 'dem Sie den Namen "M_icons" geben. Nachdem das vollbracht ist,
 'starten Sie das Programm, gehen zum Icon-Editor und stellen diesen
 'auf das Raster 44 * 44 Punkte ein. Nun können Sie Ihre Menü-Icons
 'entwerfen. Haben Sie eins entworfen, klicken Sie die "Sa"-Box des
 'Editors an und speichern das Bild in dem angegebenen Ordner unter
 'dem Namen "MICON_" + des Indexes des jeweiligen Menüfeldes, welches
 'Sie gerade entworfen
 'haben, ab (z.B. für das erste Menüfeld "Quadrat voll m. Rand" den
 'Namen "MICON_1" usw.). Jetzt können Sie das nächste Icon entwerfen
 'und abspeichern usw... Die Namen finden Sie auch am Schluß des
 'Programms in der entsprechenden Data-Zeile.
 'Damit Sie die richtige Reihenfolge einhalten, hier die Menüfelder
 'und ihre Indizes (Nummern):
 '
 ' Quadrat,voll ,m. Rand = 1
 ' Quadrat,leer = 2
 ' Quadrat,voll ,o. Rand = 3
 ' Rundeck,voll ,m. Rand = 4
 ' Rundeck,leer = 5
 ' Rundeck,voll ,o. Rand = 6
 ' Kreis ,voll ,m. Rand = 7
 ' Kreis ,leer = 8

' Kreis ,voll ,o. Rand = 9

'Die Größe der ersten neun Grafikfiguren läßt sich mit den
'Menüfeldern

'"Patch Plus" und "Patch Minus" oder durch Druck auf die Tasten "F7"
'und "F8" (auch während des Zeichnens) verändern.

'Self- ,defined,Icon = 10

'Die "SELF"-Funktion übernimmt das gerade im Raster-Puffer
'befindliche Bild (auch die durch MOVE/COPY gewählten Ausschnitte
'und mit dem Icon-Editor entworfenen Bilder) und verwendet sie als
'"Pinself". Drücken Sie die linke Maustaste, wird das Icon auf dem
'Bildschirm im jeweils eingestellten Grafikmodus gezeichnet. Mit der
'rechten Maustaste wird es über den Bildschirm bewegt.

' S.rubb ,voll ,m. Rand = 11

' S.rubb ,leer = 12

' S.rubb ,voll ,o. Rand = 13

' R.rubb ,voll ,m. Rand = 14

' R.rubb ,leer = 15

' R.rubb ,voll ,o. Rand = 16

' E.rubb ,voll ,m. Rand = 17

' E.rubb ,leer = 18

' E.rubb ,voll ,o. Rand = 19

'Diese neun Figuren erwarten einen Klick mit der linken Maustaste an
'einer beliebig gewählten Position. Halten Sie die Maustaste
'gedrückt,

'und ziehen Sie die jeweilige Figur auf. Soll die geöffnete Figur
'gezeichnet werden, lassen Sie die Maustaste einfach wieder los.

'Soll sie gelöscht werden, drücken Sie zusätzlich zur schon
'gedrückten linken auch die rechte Maustaste, und lassen Sie
'anschließend die linke Taste zuerst wieder los und dann die rechte.
'Danach können Sie wieder eine neue Box öffnen.

'LINES = 20

'Im "LINE"-Modus gibt es drei Möglichkeiten. Mit der linken
'Maustaste

'werden Freihandstriche gezeichnet, mit der rechten können Sie einen
'Punkt markieren und eine gerade Linie placieren. Drücken Sie
'anschließend an die rechte zusätzlich die linke Maustaste, wird die
'placierte Linie gezeichnet.

'FILL = 21

'Durch einen Klick mit der linken Maustaste wird der unter dem
'Mauszeiger
'liegende Bereich mit dem eingestellten Füllmuster gefüllt.

'Plot- , SPRAY = 22

'Patt- , SPRAY = 23

'Diese beiden Funktionen lassen sich ebenfalls durch die Kombination
'der Maustasten variieren. Mit der linken Taste werden kleine Sprüh-
'punkte, mit der rechten mittelgroße und mit beiden gleichzeitig
'große Sprühpunkte erzeugt.

'MARKER = 24

'Die Funktion "Marker" besitzt ein kleines Untermenü, daß ebenso wie
'das Textuntermenü entweder durch nochmaliges Drücken der Tab-Taste
'(keine Menü-Icons vorhanden) aufgerufen wird (wenn der Menütext
'"MARKER"

'im Fenster erscheint) oder durch Klick mit beiden Tasten auf das
'MARKER-Icon im Grafikmenü. Sie können anschließend eines der sechs
'verfügbaren Marker-Symbole auswählen und damit auf die gleiche Art
'zeichnen, wie Sie es vom Patch-Modus (die ersten neun Figuren) her
'kennen.

'TEXT = 25

'Die "Text"-Funktion hat gleich mehrere Untermenüs. Um das Text-
'Hauptmenü aufzurufen, verfahren Sie genauso wie unter "MARKER"
'beschrieben. Anschließend erscheint ein vierteiliges Menü. Hier
'können Sie die nötigen TextEinstellungen vornehmen (Größe, Art,
'Farbe, Richtung). Klicken Sie das gewünschte Feld an und es
'erscheint ein neues Mini- Menü. Stellen Sie den gewünschten Effekt
'ein und schieben Sie anschließend

'den Mauszeiger zur Seite. Es erscheint das Text-Hauptmenü wieder.
 'Wollen Sie dieses ebenfalls wieder verlassen, klicken Sie neben das
 'Menü auf die Screen.

,

'Nachdem Sie die einzelnen Menü-Icons entworfen und abgespeichert
 'haben, können Sie ihre eigenen Menübildchen vom nächsten
 'Programmstart
 'an bei Aufruf des "Draw Modus"-Menüs jederzeit bewundern.

,

'Das Programm fragt bei Programmstart auch auf der Diskette nach, ob
 'ein Titelbild mit dem Namen "PRE_PRO.PIC" vorhanden ist. Ist es das
 'nicht, können Sie Ihr eigenes Titelbild entwerfen. Dieses Titelbild
 'wird vom nächsten Programmstart an als erstes Bild geladen und in
 'beide SCREEN-Speicher gepackt.

'Zeichen Sie dazu ein normales Fullscreen-Bild und speichern es
 'anschließend

'mit "SAVE SOME" (Hauptmenüpunkt "DISKY") unter dem oben angegebenen
 'Namen ab. Da die Save-Funktion selbsttätig den Extender ".PRO"
 'setzt,

'müssen Sie nun nur noch mit der Funktion "NAME SOME" den Extender
 'ändern. Fertig ist das individuelle Titelbild.

,

'Von nun an ist dieses Programm Ihr ganz persönliches Grafik-
 'Programm.

,

```
@Grow.shrink.box(0,542,85,10,10,200,122,258,258)
```

```
Graphmode 1
```

```
Defline 0,1,0,0
```

```
Deffill ,0,0
```

```
Pbox 200,122,458,382
```

```
Box 201,123,457,381
```

```
Box 202,124,456,380
```

```
Deffill ,2,4
```

```
Pbox 204,126,454,378
```

```
For J=0 To 4
```

```
  For I=0 To 4
```

```
    Inc Icnindex
```

```
    Box 211+(I*47),133+(J*47),211+47+(I*47),133+47+(J*47)
```

```
    Put 212+(I*47),134+(J*47),M.icon$(Icnindex)
```

```
  Next I
```

```

Next J
Icnindex=0
I.ntodraw:
Repeat
Until Mousek
Repeat
    Mouse X,Y,K
    Dmx.index=Int((X-211)/47)+1
    Dmy.index=Int((Y-133)/47)
    '
    'Stelle die Spalte und Zeile des Menüs fest, auf der sich der
    'Mauszeiger befindet.
    '
    If Dmx.index>=1 And Dmx.index<=5 And Dmy.index>=0 And Dmy.index<=4
        '
        'Wenn sich der Mauszeiger innerhalb des Menüs befindet...
        '
        D.rawmode%=Dmx.index+Dmy.index*5
        '
        '...ermittle den Operationsmodus zu dem gewählten Feld.
        'Mit dieser kleinen Formel lassen sich auch die Menüpunkte
        'zweidimensionaler Menüs in der richtigen Reihenfolge
        'durchnummerieren.
        '
        Put 519,63,M.icon$(D.rawmode%),3
    Endif
    If K=3 And D.rawmode%=24
        '
        'Da die beiden Menüpunkte "MARKER" und "TEXT" jeweils ein, bzw.
        'mehrere
        'Untermenüs besitzen, können diese mit einem Klick auf beide
        'Maustasten aufgerufen werden.
        '
        @Markermenu
        @Grow.shrink.box(1,330,320,45,45,250,150,140,130)
        Goto I.ntodraw
    Endif
    If K=3 And D.rawmode%=25
        '
        'Siehe oben.

```

```

      ,
      @T.extmenu
      @Grow.shrink.box(1,375,320,45,45,250,150,140,130)
      Goto I.ntodraw
    Endif
  Until K=0
  Get 0,0,639,120,Helparea$
  @P.utflip
  Put 0,0,Helparea$,3
Endif
If Help.on%=0 Or (Help.on%=1 And Icnflag%=0)
  ,
  'Ist das Grafikmenü geschlossen bzw. ist es offen aber es sind keine
  'Menü-Icons vorhanden, dann verzweige in diesen Teil.
  ,
  If K=1 Or Fkeys%=67
    ,
    'Wird die linke Maustaste über dem Grafikmenüpunkt "Draw Modus",
    'oder die Taste "F9" gedrückt erhöhe den Operationsindex
    '"D.rawmode%" um eins.
    ,
    Inc D.rawmode%
    If D.rawmode%>25
      D.rawmode%=1
    Endif
  Endif
  If K=2
    ,
    'Wenn die rechte Maustaste über dem Menüpunkt "Draw Modus"
    'gedrückt wird vermindere den "D.rawmode%" um eins.
    ,
    Dec D.rawmode%
    If D.rawmode%<1
      D.rawmode%=25
    Endif
  Endif
  If (D.rawmode%=24 Or D.rawmode%=25) And Help.on%=1 And Fkeys%=18
    ,
    'Ist das Grafikmenü geöffnet und es steht entweder "MARKER" oder
    '"TEXT" im Feld und es wird die "Tab"-Taste gedrückt...

```

```
'
If Icnflag%=1
'
'...dann setze entweder das aktuelle Menü-Icon ein, falls es
'vorhanden ist (Icnflag%=1).
'
'
Put 519,63,M.icon$(D.rawmode%),3
Else
'
'oder, falls es nicht vorhanden ist, zeige den Text der
'aktuellen Operation im Fenster an.
'
Graphmode 1
Deftext 1,1,0,4
Text 520,75,D.mode$(D.rawmode%,1)
Text 520,85,D.mode$(D.rawmode%,2)
Text 520,95,D.mode$(D.rawmode%,3)
Endif
If D.rawmode%=24
@Markermenu
Else
@T.extmenu
Endif
Endif
Endif
R.emem=D.rawmode%
If Help.on%=1
If Icnflag%=1
Put 519,63,M.icon$(D.rawmode%),3
Else
Graphmode 1
Deftext 1,1,0,4
Text 520,75,D.mode$(D.rawmode%,1)
Text 520,85,D.mode$(D.rawmode%,2)
Text 520,95,D.mode$(D.rawmode%,3)
Endif
Pause 10
Endif
Return
```

Procedure Setgrafmode

```

Inc Grafmod%
If Grafmod%=5
    Grafmod%=1
Endif
If Help.on%=1
    Graphmode 1
    Deftext 1,1,0,4
    If Grafmod%=1
        Text 583,80,0,"REPLACE"
        Text 583,93,0," MODUS "
    Endif
    If Grafmod%=2
        Text 583,80,0,"TRANSP."
        Text 583,93,0," MODUS "
    Endif
    If Grafmod%=3
        Text 583,80,0," X O R "
        Text 583,93,0," MODUS "
    Endif
    If Grafmod%=4
        Text 583,80,0,"INVERS-"
        Text 583,93,0,"TRANSP."
    Endif
Endif
Pause 20

```

Return

Procedure Setnewmenu

```


```

```

'In dieser Prozedur werden bei jedem Grafikmenü-Aufruf die aktuellen
'Einstellungen ermittelt und der Menü-Inhalt dementsprechend neu
'gezeichnet.

```

```

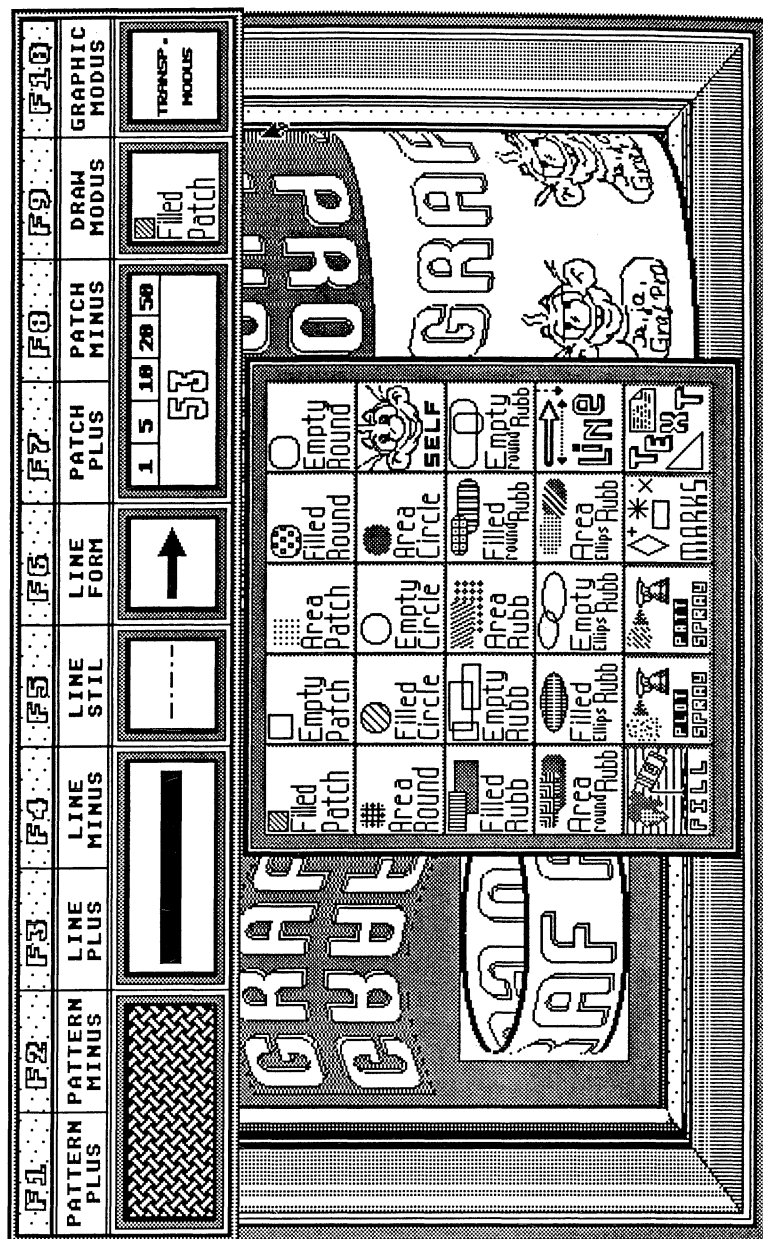

```

```

@Clipprec(2)
Graphmode 1
Deffill ,Fstil%,Fmuster%
Color 1
Pbox 14,62,123,107
Defline 0,Lndick%,0,0
Line 145,84,244,84

```

```
Deffill ,0,0
Pbox 266,62,312,107
If Lnstil%=0
    Color 0
Endif
Defline Lnstil%,1,0,0
Line 269,84,309,84
Color 1
Pbox 329,62,375,107
Defline 1,6,Lnstart%,Lnende%
Line 334,84,370,84
Deftext 1,16,0,20
Text 426,100,0,Str$(Patchsize%)+ " "
If Icnflag=1
    Put 519,63,M.icon$(D.rawmode%),3
Else
    Deftext 1,1,0,4
    Text 520,75,D.mode$(D.rawmode%,1)
    Text 520,85,D.mode$(D.rawmode%,2)
    Text 520,95,D.mode$(D.rawmode%,3)
Endif
Deftext 1,1,0,4
If Grafmod%=1
    Text 583,80,0,"REPLACE"
    Text 583,93,0," MODUS "
Endif
If Grafmod%=2
    Text 583,80,0,"TRANSP."
    Text 583,93,0," MODUS "
Endif
If Grafmod%=3
    Text 583,80,0," X O R "
    Text 583,93,0," MODUS "
Endif
If Grafmod%=4
    Text 583,80,0,"INVERS-"
    Text 583,93,0,"TRANSP."
Endif
@Set.paras
Return
```



Procedure Showhelp

```

'
'Prozedur zur Darstellung des Grafikmenüs.
'
Help.on%=1
@G.etflip
If Helpstartflag%=1
    For I%=0 To 120 Step 10
        Put 0,-120+I%,Helparea$,3
    Next I%
Endif
If Helpstartflag%=0
    Color 1
    Graphmode 1
    Defline 1,1,0,0
    Deffill ,2,2
    Pbox 0,0,639,120
    Deffill ,0,0
    Pbox 4,4,635,116
    Box 1,1,638,120
    Box 4,4,635,116
    Box 5,5,634,115
    F.co%=0
    Line 5,23,634,23
    Line 5,25,634,25
    Line 5,51,634,51
    Line 5,53,634,53
    Deffill ,2,15
    Pbox 5,5,633,22
    For I%=5 To 572 Step 63
        Graphmode 1
        Inc F.co%
        If F.co%=5 Or F.co%=6 Or F.co%=9 Or F.co%=10
            Line I%,5,I%,115
            Line I%+63,5,I%+63,115
            Deffill ,2,4
            Pbox I%+3,56,I%+60,112
            Deffill ,0,0
            Pbox I%+7,60,I%+56,108
            Box I%+8,61,I%+55,107
        Endif
    Next I%
Endif

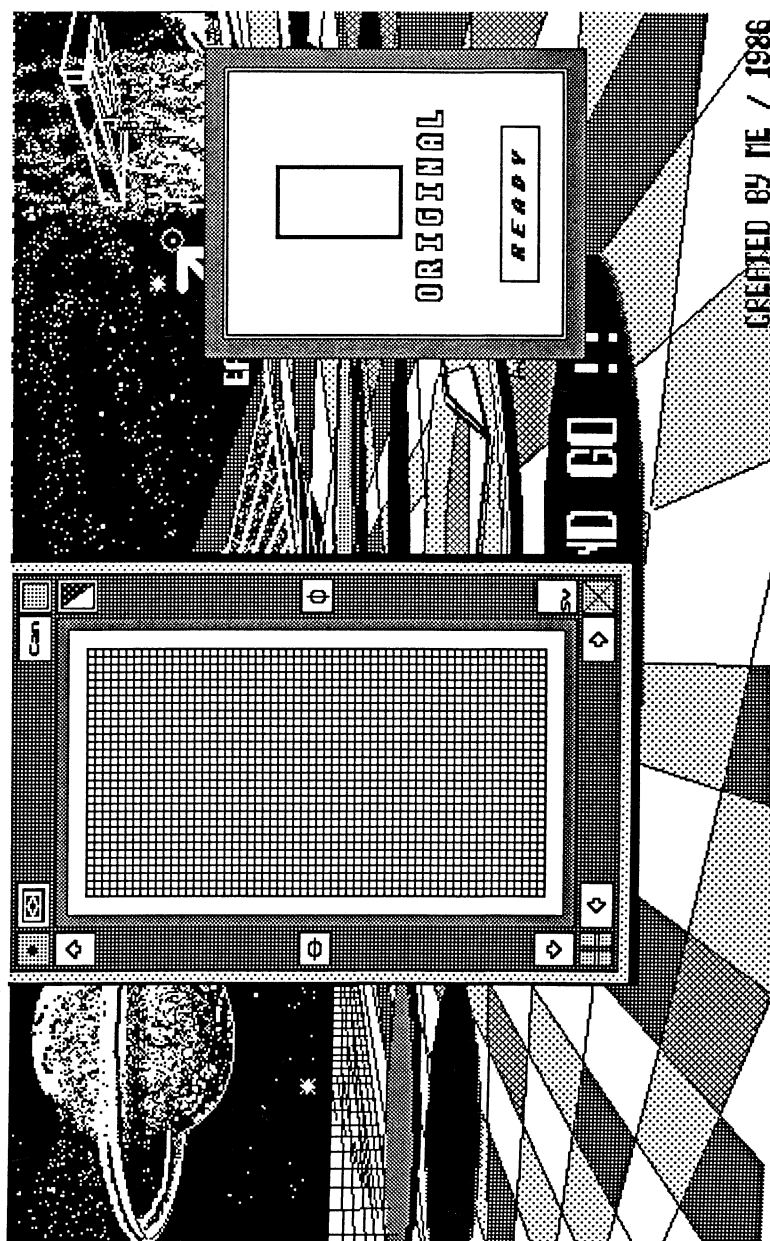
```

```

    Box I%+9,62,I%+55,107
Endif
If F.co%=1 Or F.co%=3 Or F.co%=7
    Line I%,5,I%,50
    Line I%+63,5,I%+63,50
    If F.co%=3
        Line I%,51,I%,115
    Endif
    Deffill ,2,4
    Pbox I%+3,56,I%+123,112
    Deffill ,0,0
    Pbox I%+7,60,I%+119,108
    Box I%+8,61,I%+118,107
    Box I%+9,62,I%+118,107
Endif
Graphmode 2
Deftext 1,17,0,8
Text I%+13,18,0,"F"+Str$(F.co%)
Deftext ,16
Text I%+13,18,0,"F"+Str$(F.co%)
Next I%
Deftext 1,0,0,6
Restore Grafmenutext
For I%=1 To 20
    Read X.tx%,Y.tx%,M.tx$
    Text X.tx%,Y.tx%,0,M.tx$
Next I%
Line 391,77,500,77
For I%=1 To 108 Step 22
    Line 391+I%,63,391+I%,77
Next I%
Deftext 1,1,0,6
Text 398,73,99,"1  5  10 20 50"
Get 0,0,639,120,Helparea$
Helpstartflag%=1
Endif
@Setnewmenu
Graphmode 1
Do
    @Getfkeys

```

```
Exit If (Fkeys%>68 Or (Y>120 And K>0)) And Fkey.1%<>49
Loop
Get 0,0,639,120,Helparea$
For I%=120 To 0 Step -12
    Put 0,-120+I%,Helparea$,3
Next I%
@P.utflip
Help.on%=0
Return
Procedure Changemod
'
'Prozedur zur Anpassung des Rastermodus an den gewählten GRAPHMODE.
'
'
If Grafmod%=1
    Rastermod%=3
Endif
If Grafmod%=2
    Rastermod%=7
Endif
If Grafmod%=3
    Rastermod%=6
Endif
If Grafmod%=4
    Rastermod%=12
Endif
Return
```



Procedure Starter

```

'
'Einstiegsprozedur zum Editor-Unterprogramm.
'
'Anhand des in das Grafikprogramm integrierten Icon- und Pattern-
'Editor
'soll an verschiedenen Stellen versucht werden, das Grundprinzip der
'objektbezogenen Programmierung zu verdeutlichen.
'Der erste Vorteil eines objektbezogenen Programms ist seine
'Anpassungsfähigkeit an veränderte Situationen und Erfordernisse.
'Das Editor-Programm ist nicht vollständig objektiviert, da seine
'Verwendung auf den Monochrom-Modus beschränkt ist. Grundsätzlich
'ließe sich dieses Programm in relativ kurzer Zeit so umschreiben, daß
'es sowohl im HIRES-, als auch im MIDRES- bzw. LOWRES-Modus lauffähig
'ist. Die aufwendigste Änderung wäre in diesem Fall die Anpassung der
'Grafikausgaben zum
'Screen-Aufbau, sowie der Einbau einer Möglichkeit zur Farbgestaltung.
'Im Hinblick auf seine vielseitige Verwendbarkeit wären diese
'Änderungen des Editor-Programms jedoch einer Programm-Neuentwicklung
'vorzuziehen. Bei Benutzung des Editors werden Sie feststellen, daß
'sein gesamtes Konzept
'stark an die Desktop-Window-Funktionen "Schließen, Öffnen, Verändern,
'Bewegen, Scrollen" usw. angelehnt ist. Grundsätzlich ist also auch
'eine derart komfortable Programmgestaltung mit diesem BASIC möglich.
'Das erste Kriterium für diese Art der Programmierung ist vor allem
'die Geschwindigkeit,
'mit welcher eine Programmiersprache die notwendigen Befehle umsetzt.
'Da dieses BASIC diesbezüglich alles in den Schatten stellt, was
'bisher an BASIC-Interpretern verfügbar war, kann man sich also nun
'auch an etwas gewagtere Aufgaben heranwagen.
'
Color 1
Defline 0,1,0,0
Graphmode 1
Alert 2,"          Was soll editiert werden ?          ",0," ICONS
|PATTERN|ABBRUCH",Backalert%
If Backalert%=3
  Goto E.dout
Endif
E.dstart:

```

```

If Backalert%=1
  Bbbb=0
  Alert 2,"Altes Raster beibehalten?",0," OKAY | NEIN ",Bbbb
  If Bbbb=1 And Pat.flag%=1
    Alert 3,"Geht nicht !!!|Pattern-Raster können nicht|übernommen
werden !",1," OKAY ",Aaaa
    Bbbb=2
  Endif
  Pat.flag%=0
  If Bbbb=1 And Zeilen<10 And Spalten<10
    Alert 3,"Geht nicht !!!|Es muß wenigstens einmal ein|Icon-Raster
definiert werden!",1," OKAY ",Aaaa
    Bbbb=2
  Endif
  If Bbbb=2
    Erase Raster%()
    @Set.rasterform(1)
    @Set.rasterform(2)
    Dim Raster%(Spalten,Zeilen)
  Endif
Endif
If Backalert%=2
  Bbbb=0
  Alert 2,"Altes Muster beibehalten?",0," OKAY | NEIN ",Bbbb
  If Bbbb=1
    Endif
  If Bbbb=1 And Spalten<>16 And Zeilen<>16
    Alert 3,"Geht nicht !!!|Raster entspricht nicht dem|Pattern-Format
!",1," OKAY ",Aaaa
    Goto E.start
  Endif
  If Bbbb=2
    '
    'Wurde beim Editor-Aufruf "Pattern" gewählt, wird hier das Raster
    'der Füllmuster als Editor-Raster bestimmt.
    'Die Variablen "Zeilen" und "Spalten" beinhalten anschließend das
    'angegebene Rasterformat.
    '
    Erase Raster%()
    Zeilen=16

```

```
    Spalten=16
    Dim Raster%(16,16)
Endif
    Pat.flag%=1
Endif
R.b.3=26
R.b.5=49
R.b.2=15
'
'Zur Ghost-Darstellung des Fensters bei seiner Eröffnung benötigen wir
'die Randbreiten, die das eigentliche Editor-Eingabe-Fenster umrahmen.
'
If Bbbb=1
'
'Wird der Icon-Editor zum zweiten Mal aufgerufen,kann gewählt werden
'ob das alte Editor-Raster mitsamt Inhalt erhalten werden soll oder
'ob ein neues Raster definiert wird.
'Soll das alte Raster beibehalten werden, werden hier die beim
'Verlassen des Editors zwischengespeicherten Editor-Grund-
'Definitionen wieder in die Arbeitsvariablen eingesetzt.
'
Xl=Xl.2
Yl=Yl.2
X=X.2
Y=Y.2
Zeilen=Zeilen.2
Spalten=Spalten.2
Li=Li.2
Ob=Ob.2
Re=Re.2
Un=Un.2
Breit=Breit.2
Hoch=Hoch.2
Endif
If Bbbb=2
'
'Wird der Pattern-Editor aufgerufen, wird hier seine Position
'bestimmt.
'
X=220
```

```

Y=110
Endif
Graphmode 1
Deffill ,2,4
Pbox 460,100,620,300
Deffill ,0,0
Pbox 470,110,610,290
Box 472,112,608,288
Get 460,100,620,300,E.ditinfo$
Do
  '
  'Dieses ist die Hauptschleife des Editors. In ihr werden die
  'Vorarbeiten und die Abbruchbedingungen erledigt.
  '
  @P.utflip
  Put 460,100,E.ditinfo$,3
  If Raus%=1
    '
    '  ''
    '
    C.ancel%=1
    Goto E.dloopout
  Endif
  If Bbbb=2
    '
    'Wurde der Icon-Editor aufgerufen, gehe zur Prozedur der
    'Größendefinition des Windows.
    '
    @Rubberbox.1
  Endif
  @Busy
  '
  '"Busy" ist eine kleine Prozedur, die auf dem Bildschirm die
  'Mitteilung
  '"BUSY" (engl.: beschäftigt) ausgibt, solange das Programm mit
  'zeitaufwendigen Arbeiten beschäftigt ist.
  '
  @Initvar.w1
  '

```


'In "Initvar.w1" werden sämtliche Objektdaten an die Window-Grunddefinition angepaßt. Wurde das Fenster definiert, verändert oder bewegt, müssen dort die Objektdaten auf den veränderten Zustand eingestellt werden.

'Diese Objektdaten-Verwaltung nennt man auch "Objektliste", da es eine

'Liste aller für die einzelnen Objekte wichtigen Koordinaten ist.

'Der Prozedurname "Initvar.w1" deutet an, daß es mit relativ

'geringem Aufwand sogar möglich ist, mehrere Windows (w1, w2, w3

'usw.) gleichzeitig zu verwalten. Das Hauptproblem bei der

'Verwaltung mehrerer

'Windows ist die Feststellung, welches Fenster (engl.:window) zur

'Zeit aktiv ist (oben liegt) und die Grafikverwaltung zur Einhaltung

'der Reihenfolge, in der die Fenster übereinander liegen. Mit guten

'Ideen wird Ihnen zu diesem Problem sicher eine Lösung einfallen.

,

Fenster=1

,

'Merke Dir, daß das Fenster geöffnet ist.

,

@Setplane

,

'In "Setplane" werden die in "Initvar.w1" festgelegten Koordinaten

'in eine handfeste Grafik umgesetzt, die dann das Gesamtfenster

'einschließlich aller Bedienungsfunktionen darstellt.

,

@Draw.edpic

,

'"draw.edpic" zeichnet den Editor-Inhalt entsprechend der aktuellen

'Window-Position und -Größe.

,

@Ready

,

'Nachdem die zeitaufwendigen Arbeiten abgeschlossen sind, bringt

'"Ready" die Mitteilung "READY" (engl.:fertig) auf den Bildschirm.

,

@Mausabfrage

,

'Gehe zur Abfrage-Prozedur.

,

```

E.dloopout:
Exit If C.ancel%=1
,
'Soll der Editor verlassen werden, fülle das Array, das den Editor-
'Inhalt aufnimmt, mit dem aktuellen Editor-Inhalt...
,
@Busy
@Initarray
@Ready
Fenster=0
,
'...und merke Dir, daß das Fenster von nun an geschlossen ist.
,
Loop
If Pat.flag%=1
,
'War der Pattern-Editor aktiv, frage nach, ob die Pattern-Daten
'ausgegeben werden sollen...
,
Alert 2,"Sollen die Patterndaten|ausgegeben werden ?      ",1," OKA
Y | NEIN ",B.alert%
If B.alert%=1
,
'...und gebe sie gegebenenfalls aus.
,
Alert 2,"Monitor, Drucker oder|Diskette ?      ",1,"MONITOR|DRUCK
ER|DISKETTE",Backalert%
If Backalert%=1
  Deffill 1,0,0
  Deftext 1,16,0,6
  Pbox 200,100,440,300
  Box 202,102,438,298
  Text 208,112,222,"PATTERN-DATAS"
  For I=0 To 15
    Text 208,130+I*10,"Patternzeile "+Str$(I)+" : "
    Text 380,130+I*10,Str$(Patword%(I))
  Next I
  Text 228,295,"(Maustaste drücken)"
  Repeat
  Until Mousek

```

```
Endif
If Backalert%=2
  For I=0 To 15
    Lprint "Zeile ";I;" : ";Patword%(I)
  Next I
Endif
```

LPRINT

Wenn Sie sich zu den glücklichen Besitzern eines Druckers zählen dürfen, haben Sie hier einen Befehl, der (außer der Positionierung "At(XP,YP)") in seiner Syntax und Verwendung mit dem "Print"-Befehl identisch ist. Der einzige Unterschied ist der, daß "Lprint" die gewünschten Zeichen nicht auf dem Bildschirm, sondern auf dem Drucker ausgibt. Um den Druckkopf bei Bedarf steuern zu können, sollten Sie sich das Handbuch zu Ihrem Drucker etwas genauer anschauen. Dort finden Sie dann die notwendigen Steuerbefehle, um z.B. den "Print"-Zusatz "At(XP,YP)" auf diese Weise zu simulieren. Zusätzliche Informationen zu "Lprint" können Sie, wie gesagt, auch unter "PRINT" nachlesen.

LLIST

Wollen Sie Ihr Programm-Listing auf den Drucker übertragen, geben Sie diesen Befehl ein. Ihm werden keine Parameter angefügt. Findet der Interpreter diesen Befehl, haben Sie anschließend nicht mehr die Möglichkeit, die Druckerausgabe des Listings zu unterdrücken. Es sei denn, Sie schalten den Drucker "OFF LINE" oder ganz einfach aus. Nach ca. einer halben Minute meldet sich dann Ihr Programm wieder. Je nach Druckertyp und -marke kann diese Unterbrechung während des Druckvorgangs unterschiedliche Auswirkungen haben. Die meisten Drucker wird es nicht stören, wenn sie in ihrer Arbeit brutal unterbrochen werden. Einige jedoch könnten es Ihnen übel nehmen. Sie werden sicher schon wissen, ob ihr Drucker zu dieser mißmutigen Gattung gehört. Grundsätzlich ist jedoch bei einem Ausschalten des Druckers eine kurze Frist von ca. 3-5

Sekunden einzuhalten, bevor Sie das Gerät wieder einschalten. Diese Besinnungspause gehört sozusagen zur Wartung des Druckers. Er wird sie Ihnen mit einem langen Druckerleben vergelten.

LPOS

Mit der Funktion "Lpos" können Sie den aktuellen Standort des virtuellen Druckkopfes im Zeichenpuffer Ihres Druckers ermitteln. Da dieser nicht mit der aktuellen Position des physikalischen Schreibkopfes (der den Lärm macht) übereinstimmen muß, kann es unter Umständen wichtig sein, die gerade im Speicher aktuelle Position zu erfahren. "Lpos" wird in Klammern ein beliebiger Wert angefügt. Dieser sogenannter "Dummy" hat keine Bedeutung.

z.B.:

```
For i=1 to 50
  Lprint Lpos(1);
Next i
A=Lpos(1)
Print A
```

Sie kennen sicher eine andere Art Dummys durch Photo- oder Filmaufnahmen von Crash-Tests neuer Fahrzeugkonstruktionen. Die Gummipuppen, die anstatt von Menschen Unfälle überstehen müssen, werden ebenfalls "Dummys" genannt. Die Bedeutung des Wortes ist im weiten Sinn: "So tun, als ob" (dummy, engl.: Attrappe/Schwindel). Dieser Ausdruck wird in der Computer-Umgangssprache recht häufig verwendet. Seit heute wissen Sie, was damit gemeint ist.

If Backalert%=3

,

'Falls die Option "Diskette" gewählt wurde, werden die
'Daten der einzelnen Füllmusterzeilen als "Data"-Zeile auf
'Diskette geschrieben. Den Dateinamen, unter welchem
'diese Zeile gespeichert werden soll,

```

'können Sie frei wählen. Allerdings wird der Extender
'".LST" vom Programm auf jeden Fall nachgestellt, selbst
'wenn Sie einen anderen Extender benennen. Hierdurch
'haben Sie die Möglichkeit, diese Data-
'Zeile mit "Merge" in jedes beliebige Programm
'einzubinden, ohne
'die nervenaufreibende Datentipperei auf sich nehmen zu müssen.
'Da der Mauszeiger sowie die Sprites sich ebenfalls aus
'6*16 Punkten zusammensetzen, können Sie auf diese
'Weise "spielend" Mausformen und Sprites in Hülle und
'Fülle entwerfen und in ihren eigenen Programmen
'verwenden.
'
Fileselect "*.LST",".LST",Selected$
If Selected$>"" And Selected$<>"\" And Selected$<>".LST"
    E.xtender$=".LST"
    @Extend
    Open "O",#1,Selected$
    Print #1;"Data ";
    For I=0 To 15
        Print #1;Str$(Patword%(I));",";
    Next I
    Print #1;Chr$(13)
    Close #1
Endif
Endif
Endif
C.ancel%=0
Fenster=0
Raus%=0
'
'Nachfolgend merkt sich das Programm die Grunddefinition
'des Windows, um sie beim nächsten Mal, falls der alte Editor
'wieder benutzt werden soll, wieder einsetzen zu können.
'
Zeilen.2=Zeilen
Spalten.2=Spalten
Li.2=Li
Ob.2=Ob

```

```

Re.2=Re
Un.2=Un
Breit.2=Breit
Hoch.2=Hoch
E.dout:
@Set.paras
Return
Procedure Set.rasterform(S.size%)
  '
  'In dieser Prozedur wird der Hintergrund für die Dragbox
  'gezeichnet, mit welcher die Rastergröße bestimmt wird.
  '
  Get 200,120,440,260,Gridback$
  Graphmode 1
  Deffill ,0,0
  Pbox 200,120,440,260
  Box 201,121,439,259
  Box 205,125,435,255
  Deftext 1,16,0,13
  If S.size%=1
    Text 220,150,"Anzahl Rasterzeilen :"
    R.index%=1
  Endif
  If S.size%=2
    Text 215,150,"Anzahl Rasterspalten :"
    R.index%=2
  Endif
  Deffill ,2,4
  Pbox 220,160,420,200
  Deffill ,0,0
  Pbox 225,165,415,195
  Box 227,167,413,193
  Box 302,220,340,245
  Box 303,221,339,244
  @Dragbox(1,229,169,6,22,229,169,182,22)
  Put 200,120,Gridback$,3
  R.index%=0
Return
Procedure Mausabfrage
Do

```

```

'
'In dieser Schleife bleibt das Programm, solange nicht die
'ganz rechts oben oder die links daneben liegenden
'"Can"(cel)-Box angeklickt wird (s.Schleifenende).
'Wird die "OK"-Box (ganz rechts oben) angeklickt wird der
'aktuelle Editor-Inhalt in das Aufnahme-Array
'"Raster%(Spalten,Zeilen)"
'Übertragen und das entworfenene Icon-Muster ist verfügbar.
'Wird "Can" angeklickt, wird der Editor sofort verlassen. Das
'aktuelle Icon wird nicht übernommen.
'
Repeat
'
'In dieser Schleife bleibt das Programm, solange keine
'Maustaste gedrückt wird.
'
Mouse X,Y,K
If X>472 And X<608 And Y>112 And Y<288 And Pat.flag%=1
'
'Wird der Mauszeiger in den "ORIGINAL"-Ausschnitt bewegt und ist
'gleichzeitig der Pattern-Editor aktiv (Pat.flag%=1)...
'
@Busy
@Initarray
@Patedit
Deffill ,4,1
Pbox 478,Mic.b.yr+25,602,282
@Ready
'
'zeige das bis hierher entworfenene Füllmuster als Beispiel in
'dieser Box.
'
Endif
Until Mousek
Mouse X,Y,K
If X>S.d.b.xl And X<S.d.b.xr And Y>S.d.b.yl And Y<S.d.b.yr
'
'Befindet sich bei Maustastendruck der Mauszeiger in der "S.d.b"-
'Box (Scroll.Down.Box)...
'

```

```

@Busy
@Initarray
|
|...Übernehme den aktuellen Editor-Inhalt...
|
For I=Zeilen Downto 1
  For J=0 To Spalten-1
    Raster%(J,I)=Raster%(J,I-1)
  Next J
Next I
|
|... und verschiebe die Rasterzeilen im "Raster"-Array um eine
|Zeile nach unten.
|
For I=0 To Spalten-1
  Raster%(I,0)=0
Next I
|
|Lösche die oberste Zeile...
|
@Clearplane
@Drawraster
@Draw.edpic
|
|...und zeichne den Inhalt neu.
|
@Ready
Endif
If X>S.u.b.xl And X<S.u.b.xr And Y>S.u.b.yl And Y<S.u.b.yr
|
|Wurde die "Scroll.Up.Box" angeklickt...
|
@Busy
@Initarray
|
|...merke Dir den aktuellen Editorinhalt,...
|
For I=0 To Zeilen-1
  For J=0 To Spalten-1
    Raster%(J,I)=Raster%(J,I+1)
  
```



```

    Next J
Next I
'
'...verschiebe die Rasterzeilen um eine Zeile nach oben,...
'
For I=0 To Spalten-1
    Raster%(I,Zeilen)=0
Next I
'
'...lösche die unterste Zeile...
'
@Clearplane
@Drawraster
@Draw.edpic
'
'...und zeichne den Editorinhalt neu.
'Der Aufruf der Prozeduren "Initarray","Initvar.w1", "Clearplane",
'Drawraster" und "draw.edpic" wird Ihnen im folgenden Listing
'immer wieder begegnen.
'Die Funktionen dieser Prozeduren soll hier einmal aufgeführt
'werden.
'Anschließend wird davon ausgegangen, daß Sie bei diesen Aufrufen
'wissen, worum es sich handelt. Die Funktion von "Initvar.w1"
'wurde bereits weiter oben beschrieben.
'"Initarray" : Durch eine Schleife wird festgestellt, welche der
'einzelnen Felder des Editor-Rasters mit schwarz belegt
'sind. Dementsprechend werden die den Punkten zugeordneten
'Elemente des Aufnahme-Arrays "Raster%()" entweder mit
'einer 1 (schwarz=gesetzt) oder mit einer 0 (weiß=nicht
'gesetzt) gefüllt.
'"Clearplane": Es wird einfach die Editor-Eingabefläche und das
'Fenster, in welchem die Icon- bzw. Pattern-Originalgrößen gezeigt
'werden, mit einer weißen Box überdeckt.
'"Drawraster": An "Clearplane" schließt sich in jedem Fall diese
'Prozedur an, welche die ebenfalls simple Aufgabe hat, die Linien
'des Editor-Rasters zu zeichnen.
'"Draw.edpic": Diese Prozedur wird aufgerufen, um den Inhalt des
'Editors nach evtl. Änderungen der veränderten Editorfläche
'entsprechend wieder neu in das Editor-Raster zu zeichnen.
'

```

```

@Ready
Endif
If X>S.l.b.xl And X<S.l.b.xr And Y>S.l.b.yl And Y<S.l.b.yr
'
'Das gleiche Verfahren wie unter "S.d.b" und "S.d.u". Diesmal
'wird der Editorinhalt nach links verschoben.
'
@Busy
@Initarray
For I=0 To Spalten-1
  For J=0 To Zeilen-1
    Raster%(I,J)=Raster%(I+1,J)
  Next J
Next I
For I=0 To Zeilen-1
  Raster%(Spalten,I)=0
Next I
@Clearplane
@Drawraster
@Draw.edpic
@Ready
Endif
If X>S.r.b.xl And X<S.r.b.xr And Y>S.r.b.yl And Y<S.r.b.yr
'
'Das gleiche wie oben. Inhalt wird nach rechts verschoben.
'
@Busy
@Initarray
For I=Spalten Downto 1
  For J=0 To Zeilen-1
    Raster%(I,J)=Raster%(I-1,J)
  Next J
Next I
For I=0 To Zeilen-1
  Raster%(0,I)=0
Next I
@Clearplane
@Drawraster
@Draw.edpic
@Ready

```

```

Endif
If X>Sa.b.xl And X<Sa.b.xr And Y>Sa.b.yl And Y<Sa.b.yr
'
'Die "Save.Box" wurde angeklickt.
'
Alert 2,"Abspeichern ?           ",1," OKAY | NEIN ",Backalert%
If Backalert%=1
  If Pat.flag%=0
    '
    'Soll der Inhalt gespeichert werden und ist der Icon-Editor
    'aktiv, dann gehe in die "Iconsaver"-Prozedur.
    '
    @Iconsaver
  Else
    '
    'Ist der Pattern.Editor aktiv, verzweige zu "Patternsaver".
    '
    @Patternsaver
  Endif
Endif
Endif
If X>M.v.b.xl And X<M.v.b.xr And Y>M.v.b.yl And Y<M.v.b.yr
'
'Die "Mirror.Vertikal.Box" wurde gewählt.
'In diesem Fall wird die Spiegelung nicht hier, sondern in
'"draw.edpic" vollzogen (s. dort).
'
@Busy
M.v.flag=1
'
'Ist das "M.v.flag" gleich 1, weiß das Programm in "draw.edpic",
'was es zu tun hat.
'
@Initarray
@Clearplane
@Drawraster
@Draw.edpic
M.v.flag=0
@Ready
Endif

```

```

If X>M.h.b.xl And X<M.h.b.xr And Y>M.h.b.yl And Y<M.h.b.yr
'
'Wurde die "Mirror.Horizontal.Box" gewählt...
'
@Busy
@Initarray
For I=0 To (Zeilen/2)-1
  For J=0 To Spalten-1
    Swap Raster%(J,I),Raster%(J,(Zeilen-1)-I)
  Next J
Next I
'
'...werden in dieser Doppelschleife die von unten aufsteigenden
'Editorzeilen mit den von oben fallenden vertauscht. D.h., die
'erste Zeile wird mit der letzten vertauscht, die zweite mit der
'vorletzten usw., bis die Zeilen sich in der Mitte treffen. In der
'Prozedur "Dragbox" wurde vorherbestimmt, daß die Anzahl der
'Zeilen des Icon-Editors nur gerade Werte annehmen kann. Deshalb
'geht diese "Umtausch-Aktion" immer auf. Im Pattern-Editor ist die
'Zeilenanzahl sowieso gerade (16).
'
@Clearplane
@Drawraster
@Draw.edpic
@Ready
Endif
If X>Cle.b.xl And X<Cle.b.xr And Y>Cle.b.yl And Y<Cle.b.yr
'
'"Clear.Box" (unter der "Okay.Box") wurde angeklickt. Lösche das
'Fenster.
'
@Busy
@Clearplane
@Initarray
@Drawraster
@Ready
Endif
If X>Can.b.xl And X<Can.b.xr And Y>Can.b.yl And Y<Can.b.yr
'
'Die "Cancel.Box" wurde gewählt. Setze das Abbruchflag.

```

```

'
C.ancel%=1
Endif
If X>Inv.b.xl And X<Inv.b.xr And Y>Inv.b.yl And Y<Inv.b.yr
'
'Auch diese Funktion wird nicht an Ort und Stelle erledigt,
'sondern
'in "Initarray". Dort müssen bei jeder Window-Aktion die einzelnen
'Punkte des Editor-Inhalts festgestellt werden. Um diese Punkte zu
'invertieren, braucht man nur die zu Bedingung ändern, anhand
'welcher
'das Programm entscheidet, ob ein Punkt als gesetzt registriert
'wird.
'Im Normalfall wird eine 1 in das Aufnahme-Array in das Element
'gesetzt, welches den Punkt repräsentiert. Ist der Editor-Punkt
'weiß, wird eine 0 eingetragen. Trägt man nun eine 1 ein, wenn der
'Punkt weiß, und
'eine 0, wenn der Punkt schwarz ist, hat man den Editor-Inhalt
'invertiert.
'
@Busy
Invflag=1
@Initarray
@Clearplane
@Drawraster
@Draw.edpic
Invflag=0
@Ready
Endif
If X>R.b.xl And X<R.b.xr And Y>R.b.yl And Y<R.b.yr
'
'Die "Rubber.Box" wurde geklickt. Das Wort "Rubber" ist der
'englischen Sprache entliehen. Es hat die Bedeutung
'"wischen/rädieren" (to rubb). Wenn in diesem Programm häufiger
'dieser Ausdruck benutzt wird, so ist damit immer eine Box
'gemeint, die mit ihrer linken oberen Ecke feststeht
'und deren rechte untere Ecke dem Mauszeiger folgt (außer "Ellyps
'Rub" im Grafik-Menü). Damit läßt sich in diesem Fall das Fenster
'in der Größe verändern. Die "Rubber.Box" ist hier ganz rechts
'unten.

```

```
'Wenn Sie das "Rubb"-Feld des Editors anklicken, müssen Sie die
'Maustaste solange gedrückt halten, bis im "ORIGINAL"-Fenster die
'Meldung "READY"
'ausgegeben wird. Es erscheint nun eine "Ghost"-Box, die Ihren
'Bewegungen mit dem Mauszeiger bis zu einer bestimmten Grenze
'folgt. Lassen Sie nun den Mausknopf wieder los, wird das gesamte
'Fenster einschließlich des Editor-Inhalts in der neuen Größe
'gezeichnet. Um sich ein Bild zu machen, was das Programm in
'dieser kurzen Zeichenphase alles zu
'leisten hat, schauen Sie sich bitte die Prozeduren "Setplaine",
'"Initarray" und "Draw.edpic" an. Die Geschwindigkeit, mit der
'dieser Interpreter eine derart umfassende Arbeit erledigt, ist
'unbestreitbar tief beeindruckend. Neben seiner komfortablen
'Sprache und seinen kaum
'zu überschauenden Anwendungsmöglichkeiten ist dies wohl der
'schwerwiegendste Vorteil, den das GfA-BASIC gegenüber anderen
'Interpretern anzubieten hat.
```

```
'
```

```
@Busy
```

```
@Initarray
```

```
'
```

```
' Zuerst stelle den Editor-Inhalt fest...
```

```
'
```

```
@Rubberbox.2
```

```
'
```

```
'...anschließend lasse die Editorgröße vom Anwender neu
'bestimmen...
```

```
'
```

```
Get 460,100,620,300,E.ditinfo$
```

```
'
```

```
'...speichere das "ORIGINAL"-Window...
```

```
'
```

```
@P.utflip
```

```
'
```

```
'...setze die aktuelle Grafik-SCREEN...
```

```
'
```

```
Put 460,100,E.ditinfo$,3
```

```
'
```

```
'...setze das "ORIGINAL"-Window wieder an seinen Platz...
```

```
'
```

```
@Initvar.w1
|
|... restauriere die Objekt-Daten...
|
@Setplane
@Draw.edpic
|
|...und zeichne das Editor-Window neu.
|
@Ready
Endif
If X>Clo.b.xl And X<Clo.b.xr And Y>Clo.b.yl And Y<Clo.b.yr
|
|Sobald die "Close.Box" (ganz links oben) gewählt wurde...
|
@Busy
@Initarray
Gxl=Li
Gyl=Ob
Gxr=Re
Gyr=Un
|
|...merke Dir die alten Koordinaten für die "Grow.shrink.box"...
|
X.ed=Int(180/(Spalten/2))
Y.ed=Int(180/(Zeilen/2))
Xl=40
Yl=40
X=Xl+X.ed*Spalten
Y=Yl+Y.ed*Zeilen
|
|...errechne die neuen...
|
@Grow.shrink.box(0,Gxl,Gyl,Gxr,Gyr,Xl,Yl,X,Y)
|
|...produziere eine "Growbox"...
|
Get 460,100,620,300,E.ditinfo$
@P.utflip
|...bringe die aktuelle SCREEN auf den Bildschirm...
```

```

Put 460,100,E.ditinfo$,3
@Initvar.w1
'
'...restauriere die Objektdaten...
'
@Setplane
@Draw.edpic
'
'...und zeichne den Editor neu.
'
@Ready
Endif
If X>D.b.xl And X<D.b.xr And Y>D.b.yl And Y<D.b.yr
'
'Die "Drag.Box" wurde gewählt. "Drag" ist wieder ein englisches
'Wort und bedeutet soviel wie "ziehen/schleifen/schleppen". In der
'englischen GEM-Dokumentation wird dieses Wort häufiger verwendet.
'Damit wird im allgemeinen das Bewegen von "Ghosticons"
'(Geisterbild) beschrieben.
'Ein Ghosticon ist zum Beispiel ein kleiner Rahmen, den man über
'den Bildschirm schiebt, um z.B. auf dem Desktop eine Datei zum
'Mülleimer zu befördern.
'In diesem Fall haben Sie die kleine Box ganz links unten im
'Editor-Rahmen angeklickt. Halten Sie die Maustaste gedrückt, bis
'das "Ghosticon" erscheint. Anschließend können Sie die
'Editorfläche mit immer noch gedrückter Maustaste an eine andere
'Bildschirmposition bewegen. Lassen Sie die Taste los, wird der
'Editor dort neu gezeichnet.
'
@Busy
@Initarray
@Dragwindow.1
Get 460,100,620,300,E.ditinfo$
@P.utflip
Put 460,100,E.ditinfo$,3
@Initvar.w1
@Setplane
@Draw.edpic
Endif
Xx=Int((X-Li)/Breit)

```



```

Yy=Int((Y-Ob)/Hoch)
'
'Diese beiden Editorpunkt-Indizierungen haben Sie an anderer Stelle
'zu einem ähnlichen Anlaß (Menüfeldbestimmung) kennengelernt.
'Hier wird festgestellt, auf welchem der vielen kleinen Editorpunkte
'sich der Mauszeiger befindet.
'
If Xx<Spalten And Xx>=0 And Yy<Zeilen And Yy>=0
'
'   Befindet er sich innerhalb der Editorfläche...
'
'   @Setpoint
'
'   ...gehe zu der Routine, die die Punkte zeichnet (linke Taste)
bzw.
'   löscht (rechte Taste).
'
Endif
Exit If (X>Ok.b.xl And X<Ok.b.xr And Y>Ok.b.yl And Y<Ok.b.yr) Or C.ancel%=1
'
'Wurde die "Okay.Box" oder die "Cancel.Box" angeklickt, verlasse die
'Schleife und damit den Editor. Die "Okay.Box" ist das kleine Feld
'ganz rechts oben im Editor-Rahmen.
'
Loop
If C.ancel%=0
'
'   Nur wenn die "Okay.Box" gewählt wurde...
'
'
If Pat.flag%=0
  Get Mic.b.xl+3,Mic.b.yl+3,Mic.b.xr-3,Mic.b.yr-3,Mc.part$
  '
  '...und der Icon-Editor aktiv war, übertrage das entworfene
  'Bildchen in den Raster-Puffer "Mc.part$".
  '
Endif
If Pat.flag%=1
'

```

```

'War der Pattern-Editor aktiv...
'
@Busy
@Initarray
@Patedit
'
'...aktualisiere das Array und schreibe die Daten (in "Patedit")
'in das GEM-Array.
'
@Ready
Endif
Endif
Raus%=1
Return
Procedure Busy
  Graphmode 1
  Deffill ,0,0
  Pbox 500,255,580,275
  Deftext 1,5,0,6
  Text 510,267,60,"BUSY"
Return
Procedure Ready
  Graphmode 1
  Deffill ,0,0
  Pbox 500,255,580,275
  Deftext 1,5,0,6
  Text 510,267,60,"READY"
Return
Procedure Clearplane
  Deffill ,0
  Pbox Li+1,Ob+1,Re-1,Un-1
  Pbox Mic.b.xl+1,Mic.b.yl+1,Mic.b.xr-1,Mic.b.yr-1
  Deffill ,1
Return
Procedure Drawraster
  For J=Li To Re Step Breit
    Line J,Ob,J,Un
  Next J
  For J=Ob To Un Step Hoch
    Line Li,J,Re,J

```

Next J

Return

Procedure Rubberbox.1

!

'Die folgenden drei Routinen erzeugen in Zusammenarbeit eine
'"Rubberbox".

'Das GEM bietet zwar eine solche an, aber diese ist nur in ihrer
'maximalen Größe bestimmbar. Als maximale Größe wird vom GEM der
'Bildschirmrand angenommen. Da jedoch in diesem Editorprogramm auch
'die maximale Größe definierbar sein, mußte hierfür eine eigene
'"Rubberbox" konstruiert werden.

!

Deftext 1,1,0,6

Graphmode 3

Deffill ,2,1

Repeat

Mouse X,Y,K

Text X,Y,0,"EDIT-BOX SETZEN"

Text X,Y,0,"EDIT-BOX SETZEN"

Until K=1

Xl=X

If Xl<20+Spalten

Xl=20+Spalten

Endif

If Xl>440-(2*R.b.5+5*Spalten)

Xl=440-(2*R.b.5+5*Spalten)

Endif

Yl=Y

If Yl>415-(2*R.b.5+5*Zeilen)

Yl=415-(2*R.b.5+5*Zeilen)

Endif

If Yl<40

Yl=40

Endif

@Watchbounds

Return

Procedure Rubberbox.2

Graphmode 3

Deffill ,2,1

Do

```

Exit If Mousek=1
Loop
@Watchbounds
Return
Procedure Watchbounds
Do
  Mouse X,Y,K
  If X<Xl+5*Spalten
    X=Xl+5*Spalten
  Endif
  If Y<Yl+5*Zeilen
    Y=Yl+5*Zeilen
  Endif
  If X>435
    X=435
  Endif
  If Y>390
    Y=390
  Endif
  Rubbxl=Xl-R.b.5
  Rubbyl=Yl-R.b.5
  Rubbxr=X+(R.b.5-R.b.3)+11
  Rubbyr=Y+(R.b.5-R.b.3)+30
  Pbox Rubbxl,Rubbyl,Rubbxr,Rubbyr
  Box Rubbxl,Rubbyl,Rubbxr,Rubbyr
  Pbox Rubbxl,Rubbyl,Rubbxr,Rubbyr
  Box Rubbxl,Rubbyl,Rubbxr,Rubbyr
  Exit If K=0
Loop
Graphmode 1
Return
Procedure Setplane
'
'Dieses ist die schon erwähnte Prozedur, in welcher sämtliche
'Einzelheiten, die das Editor-Erscheinungsbild ausmachen, entsprechend
'der in "Initvar.w1" definierten Koordinaten gezeichnet wird.
'
Deffill ,1,1
Pbox Pl.5.xl,Pl.5.yl,Pl.5.xr,Pl.5.yr
Box Pl.5.xl,Pl.5.yl,Pl.5.xr,Pl.5.yr

```

```

Deffill ,2,1
Pbox Pl.4.xl,Pl.4.yl,Pl.4.xr,Pl.4.yr
Box Pl.4.xl,Pl.4.yl,Pl.4.xr,Pl.4.yr
Deffill ,2,6
Pbox Pl.3.xl,Pl.3.yl,Pl.3.xr,Pl.3.yr
Box Pl.3.xl,Pl.3.yl,Pl.3.xr,Pl.3.yr
Deffill ,2,4
Pbox Pl.2.xl,Pl.2.yl,Pl.2.xr,Pl.2.yr
Box Pl.2.xl,Pl.2.yl,Pl.2.xr,Pl.2.yr
Deffill ,0,0
Pbox Pl.1.xl,Pl.1.yl,Pl.1.xr,Pl.1.yr
Box Pl.1.xl,Pl.1.yl,Pl.1.xr,Pl.1.yr
@Drawraster
Deffill ,0
Garbage%=Fre(0)
Pbox S.l.b.xl,S.l.b.yl,S.l.b.xr,S.l.b.yr
Pbox S.r.b.xl,S.r.b.yl,S.r.b.xr,S.r.b.yr
Pbox S.u.b.xl,S.u.b.yl,S.u.b.xr,S.u.b.yr
Pbox S.d.b.xl,S.d.b.yl,S.d.b.xr,S.d.b.yr
Pbox Sa.b.xl,Sa.b.yl,Sa.b.xr,Sa.b.yr
Pbox M.v.b.xl,M.v.b.yl,M.v.b.xr,M.v.b.yr
Pbox M.h.b.xl,M.h.b.yl,M.h.b.xr,M.h.b.yr
Pbox Can.b.xl,Can.b.yl,Can.b.xr,Can.b.yr
Pbox Cle.b.xl,Cle.b.yl,Cle.b.xr,Cle.b.yr
Deffill ,2,7
Box Cle.b.xl+2,Cle.b.yl+2,Cle.b.xr-2,Cle.b.yr-2
Line Cle.b.xl+2,Cle.b.yl+2,Cle.b.xr-2,Cle.b.yr-2
Fill Cle.b.xr-4,Cle.b.yl+4
Deffill ,0,0
Pbox Inv.b.xl,Inv.b.yl,Inv.b.xr,Inv.b.yr
Pbox Mic.b.xl,Mic.b.yl,Mic.b.xr,Mic.b.yr
Box S.l.b.xl,S.l.b.yl,S.l.b.xr,S.l.b.yr
Box S.r.b.xl,S.r.b.yl,S.r.b.xr,S.r.b.yr
Box S.u.b.xl,S.u.b.yl,S.u.b.xr,S.u.b.yr
Box S.d.b.xl,S.d.b.yl,S.d.b.xr,S.d.b.yr
Box M.v.b.xl,M.v.b.yl,M.v.b.xr,M.v.b.yr
Box M.h.b.xl,M.h.b.yl,M.h.b.xr,M.h.b.yr
Box Can.b.xl,Can.b.yl,Can.b.xr,Can.b.yr
Box Cle.b.xl,Cle.b.yl,Cle.b.xr,Cle.b.yr
Box Inv.b.xl,Inv.b.yl,Inv.b.xr,Inv.b.yr

```

```

Box Inv.b.xl+3,Inv.b.yl+3,Inv.b.xr-3,Inv.b.yr-3
Box Inv.b.xl+3,Inv.b.yl+3,Inv.b.xr-4,Inv.b.yr-4
Box Sa.b.xl,Sa.b.yl,Sa.b.xr,Sa.b.yr
Box Mic.b.xl+1,Mic.b.yl+1,Mic.b.xr-1,Mic.b.yr-1
Pbox Ok.b.xl,Ok.b.yl,Ok.b.xr,Ok.b.yr
Box Ok.b.xl,Ok.b.yl,Ok.b.xr,Ok.b.yr
Deffill ,2,2
Pbox Ok.b.xl+2,Ok.b.yl+2,Ok.b.xr-2,Ok.b.yr-2
Box Ok.b.xl+2,Ok.b.yl+2,Ok.b.xr-2,Ok.b.yr-2
Pbox Clo.b.xl,Clo.b.yl,Clo.b.xr,Clo.b.yr
Box Clo.b.xl,Clo.b.yl,Clo.b.xr,Clo.b.yr
Pbox D.b.xl,D.b.yl,D.b.xr,D.b.yr
Box D.b.xl,D.b.yl,D.b.xr,D.b.yr
Pbox R.b.xl,R.b.yl,R.b.xr,R.b.yr
Box R.b.xl,R.b.yl,R.b.xr,R.b.yr
Graphmode 2
Deffill ,1,1
Deftext 1,0,0,13
Text Clo.b.xl+5,Clo.b.yr+1,0,Chr$(249)
Defftext 1,16,0,13
Text 490,Mic.b.yr+20,100,"ORIGINAL"
Defftext 1,17,0,13
Text 490,Mic.b.yr+20,100,"ORIGINAL"
Defftext 1,1,0,4
Text Can.b.xl+3,Can.b.yr-7,(Can.b.xr-Can.b.xl)-5,"Can"
Text Sa.b.xl+1,Sa.b.yr-4,"Sv"
Defftext 1,0,0,13
Text S.l.b.xl+8,S.l.b.yr-2,0,""
Text S.r.b.xl+8,S.r.b.yr-2,0,""
Text M.v.b.xl+5,M.v.b.yr-2,0,Chr$(237)
Defftext ,,900,
Text Inv.b.xr-6,Inv.b.yr-6,0,""
Text S.d.b.xr-2,S.d.b.yr-7,0,""
Text S.u.b.xr-2,S.u.b.yr-8,0,""
Text M.h.b.xr-3,M.h.b.yr-5,0,Chr$(237)
Line R.b.xl,R.b.yl,R.b.xr,R.b.yr
Line R.b.xr,R.b.yl,R.b.xl,R.b.yr
Line D.b.xl,D.b.yl+8,D.b.xr,D.b.yl+8
Line D.b.xl+8,D.b.yl,D.b.xl+8,D.b.yr
Line D.b.xl,D.b.yl+10,D.b.xr,D.b.yl+10

```

```

Line D.b.xl+10,D.b.yl,D.b.xl+10,D.b.yr
Graphmode 1
Deftext 1,0,0,13
Return
Procedure Setpoint
  '
  'Hier werden die einzelnen Editorpunkte gezeichnet, bzw. gelöscht.
  '
  Mouse X,Y,K
  X.index%=Int((X-Li)/Breit)
  Y.index%=Int((Y-Ob)/Hoch)
  '
  'Wieder die Punktbestimmung.
  '
  If X.index%<0
    X.index%=0
  Endif
  If Y.index%<0
    Y.index%=0
  Endif
  If X.index%>Spalten-1
    X.index%=Spalten-1
  Endif
  If Y.index%>Zeilen-1
    Y.index%=Zeilen-1
  Endif
  '
  'In den vorangegangenen vier Abfragen wurde festgestellt, ob sich der
  Mauszeiger
  'außerhalb der Editor-Zeichenfläche befindet.
  '
  If K=1
    '
    'Wurde die linke Maustaste gedrückt...
    '
    Raster%(X.index%,Y.index%)=1
    '
    '...setze das entsprechende Array-Element auf 1.
    '
    @Set.black

```

```

    '
    'Gehe zur "Schwarzpunkt"-Routine.
    '
Endif
If K=2
    '
    'Wurde die linke Taste gedrückt...
    '
    Raster%(X.index%,Y.index%)=0
    '
    'setze das Element auf 0...
    '
    @Set.white
    '
    '...und gehe zur "Weißpunkt"-Routine.
    '
Endif
Return
Procedure Set.white
    '
    'Hier werden die weißen Punkte (gelöscht) gezeichnet. Die Bezeichnung
    '"Mic.b" bedeutet übrigens "Micro.Box". Damit ist die kleine Fläche
    'im "ORIGINAL"-Fenster gemeint, in welcher das entworfenen Bild in
    'seiner realen Größe erscheint.
    '
    Deffill ,0,0
    @Rahmen(0)
    Pbox Li+2+((Xx)*Breit),Ob+2+((Yy)*Hoch),Li+Breit-2+((Xx)*Breit),Ob+Hoc
h-2+((Yy)*Hoch)
    Pbox Mic.b.xl+3+Xx,Mic.b.yl+3+Yy,Mic.b.xl+3+Xx,Mic.b.yl+3+Yy
    @Rahmen(1)
Return
Procedure Set.black
    '
    'Die schwarzen Punkte (gesetzt).
    '
    Deffill ,2,8
    Pbox Li+2+((Xx)*Breit),Ob+2+((Yy)*Hoch),Li+Breit-2+((Xx)*Breit),Ob+Hoc
h-2+((Yy)*Hoch)
    Pbox Mic.b.xl+3+Xx,Mic.b.yl+3+Yy,Mic.b.xl+3+Xx,Mic.b.yl+3+Yy

```


Return

Procedure Draw.edpic

```

|
|   'Die Editor-Inhalt-Zeichenroutine.
|
|   Deffill ,1,1
|
|   'Setze das Füllmuster auf Schwarz.
|
|   For Z.eile=0 To Zeilen-1
|       |
|       'Gehe eine Zeile nach der anderen durch.
|       |
|       For S.palte=0 To Spalten-1
|           |
|           'Und innerhalb jeder Zeile jeden einzelnen Punkt.
|           |
|           If Raster%(S.palte,Z.eile)=1
|               |
|               'Steht das dem jeweiligen Punkt entsprechende Array-Element auf
|               '1 (gesetzt)...
|               |
|               If M.v.flag=1
|                   |
|                   'zeichne, falls die Vertikalspiegel-Funktion gewünscht
|                   'wurde...
|                   |
|                   Pbox Li+2+(((Spalten-1)-S.palte)*Breit),Ob+2+(Z.eile*Hoch),Li+
|                   Breit-2+(((Spalten-1)-S.palte)*Breit),Ob+Hoch-2+(Z.eile*Hoch)
|                   Pbox Mic.b.xl+3+((Spalten-1)-S.palte),Mic.b.yl+3+Z.eile,Mic.b.
|                   xl+3+((Spalten-1)-S.palte),Mic.b.yl+3+Z.eile
|                   |
|                   '...die einzelnen Punkte von rechts nach links...
|                   |
|               Else
|                   |
|                   '...bzw., wenn nicht gespiegelt werden soll...
|                   |
|                   Pbox Li+2+(S.palte*Breit),Ob+2+(Z.eile*Hoch),Li+Breit-2+(S.pal
|                   te*Breit),Ob+Hoch-2+(Z.eile*Hoch)

```

```

        Pbox Mic.b.xl+3+S.palte,Mic.b.yl+3+Z.eile,Mic.b.xl+3+S.palte,M
ic.b.yl+3+Z.eile
        '
        '...von links nach rechts.
        '
        Endif
    Endif
    Next S.palte
    Next Z.eile
Return
Procedure Initarray
    '
    'Das Array, welches den Inhalt des Editors aufzunehmen hat, wird hier
    'definiert.
    '
    If Invflag=1
        '
        'Falls die "Invers.Box" (rechts neben der "Close.Box") angeklickt
        'wurde,
        '
        Setflag=0
        '
        ' ...registriere nur die weißen Punkte (Editorpunkte ohne Inhalt).
        '
    Else
        Setflag=1
        '
        'Sonst registriere die schwarzen.
        '
    Endif
    For Z.eile=0 To Zeilen-1
        '
        'Gehe wieder jede einzelne Zeile durch...
        '
        For S.palte=0 To Spalten-1
            '
            'und darin jeden einzelnen Punkt.
            '
            If Point((S.palte*Breit)+Li+(Breit/2),(Z.eile*Hoch)+Ob+(Hoch/2))=S
etflag

```

```

      'Ist der Punkt entsprechend dem oben bestimmten Flag (1 oder
0)...
      Raster%(S.palte,Z.eile)=1
      '...setze das Array-Element auf 1.
Else
      Raster%(S.palte,Z.eile)=0
      'Entspricht der Punkt nicht dem Flag, setze das Element auf 0.
Endif
Next S.palte
Next Z.eile

```

POINT

Noch eine BASIC-Funktion. Mit "Point" läßt sich im HIRES-Modus feststellen, ob ein bestimmter Bildschirmpunkt weiß oder schwarz ist. Dazu übergeben Sie der Funktion in Klammern X- und Y-Koordinate des Punktes, dessen Status untersucht werden soll. Ist der Punkt gesetzt, liefert "Point" den Wert 1 zurück. Ist der Punkt leer, wird eine 0 zurückgegeben. Im MIDRES- und LOWRES-Modus wird Ihnen durch "Point" der Farbindex des getesteten Punktes zurückgegeben (LOWRES=16 Farben 0-15 / MIDRES=4 Farben 0-3).

z.B.

```

Deffill 1,2,2
For I=1 To 400
  X=Random(100)
  Y=Random(100)
  Pbox X,Y,X+5,Y+5
Next I
For I=0 To 99
  For J=0 To 99

```

```
      If Point(J,I)=1
        Color 0
      Else
        Color 1
      Endif
      Plot I,J
    Next J
  Next I
  Pause 200
  Edit
```

PLOT

Mit "Plot" können einzelne Punkte auf dem Bildschirm ausgegeben werden. Es werden in die X- und Y-Koordinate des Punktes übergeben, der gesetzt werden soll. Bis auf die Ausnahme, daß die Linienendformen beide als rund definiert wurden, wird der Plot-Befehl von der Definition der Liniendicke nicht betroffen. Es wird grundsätzlich ein einzelnes Pixel gesetzt. Die Farbdefinition durch den "Color"-Befehl wirkt sich allerdings auch auf "Plot" aus,

z.B.:

```
Deffill ,2,4
Pbox 0,0,639,399
For I=0 To 1000
  A=Random(20)
  B=Random(4)-1
  C=Random(3)-1
  X=Random(640)
  Y=Random(400)
  Deline ,A,B,B
  Color C
  Plot X,Y
Next I
U=Inp(2)
Edit
```

DRAW TO TO

Der Grafikbefehl "Draw" ohne die Erweiterung "To" arbeitet exakt genauso wie "Plot". Es wird ein Koordinatenpaar angegeben, wo der Punkt gezeichnet werden soll.

"Draw To" zeichnet eine Linie vom letzten Punkt aus, der mit "Draw", "Line" oder "Plot" gezeichnet wurde. Wird "Draw To" eingesetzt, ohne das einer dieser Befehle einen Endpunkt markiert hat, wird die Linie vom Koordinaten-Nullpunkt (linke obere Bildschirmecke) aus zu der Koordinate gezeichnet, welche in "Draw To" angegeben wurde. Es lassen sich durch das Bindeglied "To" auch mehrere Koordinatenpaare zu einer Liste verbinden, die dann der Reihenfolge nach mit Linien verbunden werden.

z.B.

Draw 100,100

Draw To 110,100 To 110,110 To 100,110 To 100,100

Beispiel 2:

Damit Sie sich nicht soviel Arbeit machen müssen, die einzelnen Punktkoordinaten einzutippen, folgt nun ein kleines Hilfsprogramm. Wenn Sie es gestartet haben, können Sie mit einem Druck auf die linke Maustaste einzelne Punkte zeichnen. Es sind zwei Arrays dimensioniert, die die jeweilige X- und Y-Koordinate des gerade gezeichneten Punktes festhalten. Jedes dieser beiden Arrays hat 1000 Elemente, d.h., daß Sie maximal 1000 Punkte zeichnen können. Haben Sie diese Anzahl erreicht, oder drücken Sie die rechte Maustaste, werden die beiden Arrays in Form von Data-Zeilen unter dem Namen "Draw.Lst" auf Diskette abgespeichert. Anschließend wird das erste Programm beendet. Laden Sie nun mit "Merge" die geschriebene "Data"-Datei in den Arbeitsspeicher und starten Sie das zweite Programm. Sie werden sehen, daß Ihre Zeichnung nun anhand der Data-Zeilen und des "Draw"-Befehls neu auf den Bildschirm gebracht wird.

PROGRAMM 1

```
Dim Px(1000),Py(1000)
Do
  Mouse X,Y,K
  If K=1
    Pause 1
    Plot X,Y
    Inc Count
    Px(Count)=X
    Py(Count)=Y
  Endif
  Exit If K=2 Or Count=1000
Loop
Open "O",#1,"DRAW.LST"
Print #1;"D.rawkoos:";Chr$(13)
For J=1 To 100
  Print #1;"Data ";
  For I=1 To 10
    Inc Ci
    Print #1;Str$(Px(Ci));",";Str$(Py(Ci));
    If I<10
      Print #1;",";
    Endif
  Next I
  Print #1;Chr$(13)
Next J
Print #1;"Data ";Str$(1111)
Close #1
Edit
```

PROGRAMM 2

```
Dim Px(1000),Py(1000)
Deffill,2,4
Restore D.rawkoos
Read Px(1),Py(1)
Plot Px(1),Py(1)
For I=2 To 1000
```

```
Read Px(I),Py(I)
```

Pbox Px(I),Py(I),Px(I)+10,Py(I)+10 ersetzen Sie die Zeile mit dem Draw-Befehl durch diese Pbox-Zeile und schon haben Sie einen neuen Effekt.

```
Draw To Px(I),Py(I)
Exit If Px(I)=1111
Next I
U=Inp(2)
Edit
```

```
Return
```

```
Procedure Dragwindow.1
```

```
@E.dragbox
```

```
Xl=Dx+R.b.2+11
```

```
Yl=Dy-Draghoch-R.b.2-11
```

```
X=Dx+Dragbreit+R.b.5+11
```

```
Y=Dy
```

```
Return
```

```
Procedure E.dragbox
```

```
Deffill ,2,1
```

```
Graphmode 3
```

```
Dragbreit=Re-Li
```

```
Draghoch=Un-Ob
```

```
Repeat
```

```
Mouse Dx,Dy,Dk
```

```
If Dx+Dragbreit>383
```

```
Dx=383-Dragbreit
```

```
Endif
```

```
If Dy-Draghoch<=70
```

```
Dy=70+Draghoch
```

```
Endif
```

```
If Dx<=20
```

```
Dx=20
```

```
Endif
```

```
If Dy>380
```

```
Dy=380
```

```
Endif
```

```
Ghostxl=Dx-R.b.3/2
```

```

Ghostyl=Dy-Draghoch-2*R.b.2-R.b.3
Ghostxr=Dx+Dragbreit+2*R.b.2+R.b.3
Ghostyr=Dy+2*R.b.2+10
Pbox Ghostxl,Ghostyl,Ghostxr,Ghostyr
Box Ghostxl,Ghostyl,Ghostxr,Ghostyr
Pbox Ghostxl,Ghostyl,Ghostxr,Ghostyr
Box Ghostxl,Ghostyl,Ghostxr,Ghostyr
Until Dk=0
Graphmode 1
Return
Procedure Patedit

'Hier wird das Füllmuster an die entsprechenden GEM-Arrays übergeben.
'Auf das "wie" soll hier nicht weiter eingegangen werden.
'Ausführlichere
'Informationen zum Thema "GEM" folgen im Anschluß an das Listing.
'
Erase Patword%()
Dim Patword%(15)
Dpoke Contrl+2,0
Dpoke Contrl+6,16
Dpoke Contrl+12,2
For J=0 To 15
  For I=0 To 15
    If Raster%(I,J)=1
      Add Patword%(J),2^(15-I)
    Endif
  Next I
  Dpoke Intin+J*2,Patword%(J)
Next J
Vdisys 112
Return
Procedure Initvar.w1
'
'Die "Objektliste". Das A und O der Objektprogrammierung.
'
'Die Objektnamen haben folgende Bedeutung:
'
'      Li      = "Links"; der linke Rand des Editorfeldes
'      Ob      = "Oben"; der obere Rand des Editorfeldes

```


'	Breit	=	Die Breite der einzelnen Editorpunkte ergibt sich aus der Breite der Definitionsbox geteilt durch die Anzahl der Rasterzellen
'	Hoch	=	Die Höhe der einzelnen Editorpunkte ergibt sich aus der Höhe der Definitionsbox geteilt durch die Anzahl der Rasterzeilen
'	Re	=	"Rechts"; der rechte Rand des Editorfeldes ergibt sich aus "Li" + "Breit" * Anzahl der
'	Un	=	Spalten "Unten"; der untere Rand des Editorfeldes ergibt sich aus "Ob" + "Hoch" * Anzahl der Zeilen
'			
'	XL.2	=	Die Definitionsboxkoordinaten werden für den
'	YL.2	=	nächsten Aufruf des Editors zwischengespeichert
'			
'	X.2	=	
'	Y.2	=	
'			
'	R.b.1	=	"Rand.Breite.1"
'	R.b.2	=	Die "R.b"-Variablen stellen die Breite der
'	R.b.3	=	einzelnen Ränder, die das Editorfeld umgeben dar. Z.B. der
'			
'	R.b.4	=	weiße Rand direkt um das Feld herum ist "R.b.1" usw.
'			
'	R.b.5	=	
'			
'	Clo.b	=	"Close.Box"; die Box links oben
'	D.b	=	"Drag.Box"; die Box links unten
'	R.b	=	"Rubber.Box"; die Box rechts unten
'	Ok.b	=	"Okay.Box"; die Box rechts oben
'	S.u.b	=	"Scroll.Up.Box"; die Box links unter "Clo.b"
'	S.d.b	=	"Scroll.Down.Box"; die Box links über "D.b"
'	M.h.b	=	"Mirror.Vertikal.Box"; die Box links in der Mitte
'			
'	S.l.b	=	"Scroll.Left.Box"; die Box rechts neben "D.b"
'	S.r.b	=	"Scroll.Right.Box"; die Box links neben "R.b"
'	Mic.b	=	"Micro.Box"; die kleine Box im "ORIGINAL"-Fenster
'			
'	M.v.b	=	"Mirror.Horizontal.Box"; die Box rechts in der Mitte
'			

```

'      Cle.b      = "Clear.Box"; die Box rechts unter "Ok.b"
'      Sa.b       = "Save.Box"; die Box rechts über "R.b"
'      Can.b      = "Cancel.Box"; die Box links neben "Ok.b"
'      Inv.b      = "Invers.Box"; die Box rechts neben "Clo.b"
'
'      Pl.1       = "Plain.1"
'      Pl.2       = Die "Plain"-Variablen beinhalten die
'                  Koordinaten
'      Pl.3       = der einzelnen Großflächen, die insgesamt den
'      Pl.4       = Hintergrund des Editors darstellen. "Pl.5" ist
'      Pl.5       = die größte Fläche. Sie ist der Schatten rechts
'                  und unten von "Pl.4".
'
Li=Xl
Ob=Yl
D.iverr=1
On Error Goto Div.cero
Breit=Int(((X-R.b.2-11)-Xl)/Spalten)
Hoch=Int(((Y-R.b.2-11)-Yl)/Zeilen)
On Error
Re=Breit*Spalten+Li
Un=Hoch*Zeilen+Ob
Xl.2=Xl
Yl.2=Yl
X.2=X
Y.2=Y
R.b.1=10
R.b.2=R.b.1+6
R.b.3=R.b.2+23
R.b.4=R.b.3+6
R.b.5=R.b.4+4
Clo.b.xl=Li-R.b.3+3
Clo.b.yl=Ob-R.b.3+3
Clo.b.xr=Li-R.b.2-3
Clo.b.yr=Ob-R.b.2-3
D.b.xl=Clo.b.xl
D.b.yl=Un+R.b.2+3
D.b.xr=Clo.b.xr
D.b.yr=Un+R.b.3-3
R.b.xl=Re+R.b.2+3

```

R.b.yl=D.b.yl
R.b.xr=Re+R.b.3-3
R.b.yr=D.b.yr
Ok.b.xl=R.b.xl
Ok.b.yl=Clo.b.yl
Ok.b.xr=R.b.xr
Ok.b.yr=Clo.b.yr
S.u.b.xl=Clo.b.xl
S.u.b.yl=Clo.b.yr+3
S.u.b.xr=Clo.b.xr
S.u.b.yr=S.u.b.yl+21
S.d.b.xl=S.u.b.xl
S.d.b.yl=D.b.yl-24
S.d.b.xr=S.u.b.xr
S.d.b.yr=S.d.b.yl+21
M.h.b.xl=Clo.b.xl
M.h.b.yl=Ob+((Un-Ob)/2)-8
M.h.b.xr=Clo.b.xr
M.h.b.yr=M.h.b.yl+16
S.l.b.xl=D.b.xr+3
S.l.b.yl=D.b.yl
S.l.b.xr=S.l.b.xl+23
S.l.b.yr=D.b.yr
S.r.b.xl=R.b.xl-25
S.r.b.yl=R.b.yl
S.r.b.xr=S.r.b.xl+23
S.r.b.yr=R.b.yr
Mic.b.xl=540-(Spalten/2)-2
Mic.b.yl=170-(Zeilen/2)-2
Mic.b.xr=Mic.b.xl+Spalten+5
Mic.b.yr=Mic.b.yl+Zeilen+5
M.v.b.xl=R.b.xl
M.v.b.yl=M.h.b.yl
M.v.b.xr=R.b.xr
M.v.b.yr=M.h.b.yr
Cle.b.xl=R.b.xl
Cle.b.yl=S.u.b.yl
Cle.b.xr=R.b.xr
Cle.b.yr=S.u.b.yr
Sa.b.xl=Cle.b.xl

```

Sa.b.yl=S.d.b.yl
Sa.b.xr=Cle.b.xr
Sa.b.yr=S.d.b.yr
Can.b.xl=S.r.b.xl
Can.b.yl=Ok.b.yl
Can.b.xr=S.r.b.xr
Can.b.yr=Ok.b.yr
Inv.b.xl=S.l.b.xl
Inv.b.yl=Ok.b.yl
Inv.b.xr=S.l.b.xr
Inv.b.yr=Ok.b.yr
Pl.1.xl=Li-R.b.1
Pl.1.yl=Ob-R.b.1
Pl.1.xr=Re+R.b.1
Pl.1.yr=Un+R.b.1
Pl.2.xl=Li-R.b.2
Pl.2.yl=Ob-R.b.2
Pl.2.xr=Re+R.b.2
Pl.2.yr=Un+R.b.2
Pl.3.xl=Li-R.b.3
Pl.3.yl=Ob-R.b.3
Pl.3.xr=Re+R.b.3
Pl.3.yr=Un+R.b.3
Pl.4.xl=Li-R.b.4
Pl.4.yl=Ob-R.b.4
Pl.4.xr=Re+R.b.4
Pl.4.yr=Un+R.b.4
Pl.5.xl=Li-R.b.4-1
Pl.5.yl=Ob-R.b.4-1
Pl.5.xr=Re+R.b.5
Pl.5.yr=Un+R.b.5

```

Return

Procedure Cinema

!

'Das "Kino"-Unterprogramm.

'In diesem Programm werden nur einige Teile erklärt, da davon

'ausgegangen

'wird, daß Sie den größten Teil nach Lektüre der in den anderen

'Programnteilen ausgeführten Erläuterungen allein bewältigen können.

!

```

Bigloop:
Color 1
Defline 0,1,0,0
Graphmode 1
Kintopp.quit%=0
Do
  Alert 2,"Film laden, zeichnen          |oder zurück ?          ",
0," LADEN |ZEICHNEN| ZURÜCK ",Back.alert%
  If Back.alert%=1
    @P_icload
    If Selected$<>"\" And Selected$<>".FLM" And Selected$<>""
      @Showpics
    Endif
  Endif
  If Back.alert%=2
    @Getpicinits
    @Get_pics
    If L.oaddflag%=1
      L.oaddflag%=0
      Goto Bigloop
    Endif
    If Kintopp.quit%=1
      Goto K.intoppquit
    Endif
    If Trommelflag%=1
      Counter%=0
      @G.ettinny
      @Showpics
      Trommelflag%=0
      Goto K.intoppquit
    Endif
    @Showpics
  Endif
  If Back.alert%=3
    Kintopp.quit%=1
  Endif
  K.intoppquit:
  Exit If Kintopp.quit%=1
Loop
Erase P_icy$()

```

```

P_icscroll$=""
@Set.paras
Return
Procedure Getpicinits
  Deffill ,3,6
  Pbox 2,2,637,397
  Deffill ,0,0
  Pbox 96,51,544,88
  Deftext 1,16,0,26
  Text 100,80,440," C I N E M A "
  Deffill ,2,2
  Fill 320,53
  Alert 1,"Richten Sie anschließend die|Größe des Eingabe-/Ausgabe-|Fens
ters ein !",1," OKAY ",Bbbbb
  Graphmode 3
  Deftext 1,1,0,6
  Repeat
    Mouse Scr_lft%,Scr_upn%,Button%
    If Scr_upn%<65
      Scr_upn%=65
    Endif
    If Scr_upn%>360
      Scr_upn%=360
    Endif
    Text Scr_lft%,Scr_upn%,"SET WINDOW"
    Text Scr_lft%,Scr_upn%,"SET WINDOW"
  Until Mousek
  Repeat
    Mouse Scr_rgt%,Scr_dwn%,Button%
    If Scr_rgt%<Scr_lft%+10
      Scr_rgt%=Scr_lft%+10
    Endif
    If Scr_dwn%<Scr_upn%+10
      Scr_dwn%=Scr_upn%+10
    Endif
    If Scr_dwn%>380
      Scr_dwn%=380
    Endif
    Box Scr_lft%,Scr_upn%,Scr_rgt%,Scr_dwn%
    Box Scr_lft%,Scr_upn%,Scr_rgt%,Scr_dwn%

```

Until Button%=0

Graphmode 1

Get Scr_lft%,Scr_upn%,Scr_rgt%,Scr_dwn%,Dummy\$

,

'Nachdem eine "Leinwand"-Größe bestimmt wurde, muß festgestellt

'werden, wieviel Speicherplatz jedes einzelne Bild im "Get"-

'Rasterformat benötigt.

'Dazu wird einmal der definierte Ausschnitt in einen "Get"-String

'übernommen.

'Alle weiteren Bilder, die ja alle an der gleichen Position liegen,

'nehmen exakt denselben Speicherplatz in Anspruch. Würde sich allein

'die Position verändern, während die Breite und Höhe des Ausschnittes

'(in Pixeln) gleichbleibt, würde sich auch der Bedarf an Speicherplatz

'verändern, da die Rasteroperation "Get" nur ganze Wortgrößen

'aufnehmen kann. D.h., auch

'wenn Sie die Ausschnittsgröße frei wählen können, wird im Speicher

'trotzdem das letzte Bit (links) des Wortes, in dem sich der

'Ausschnittsanfang befindet als Start angenommen. Ebenso verhält es

'sich mit dem rechten Rand des Ausschnittes. Dort wird das erste Bit

'(rechts) des Wortes als Ende angenommen, wo sich der von Ihnen

'gewählte Ausschnittsrand befindet.

'Dieses etwas komplizierte Verfahren soll Sie jedoch nicht weiter

'beschäftigen. Als "Get"- und "Put"-Benutzer in diesem BASIC haben Sie

'sich mit diesem Problem auch nicht zu befassen. Grundsätzlich soll

'damit nur erklärt werden, warum ein und dieselbe Ausschnittsgröße an

'unterschiedlichen Bildschirmpositionen auch verschieden viel

'Speicherplatz benötigt.

,

Scr_hei%=Scr_dwn%-Scr_upn%

Scr_wid%=Scr_rgt%-Scr_lft%

P_icspace%=Len(Dummy\$)

,

'Die Länge des "Get"-Strings ist der Speicherplatz in Byte, den ein

'Bild benötigt.

,

Max_pics%=Int((Fre(0)-10000-P_icspace%)/P_icspace%)

,

'Man kann natürlich nicht beliebig viele Bilder gleichzeitig

'definieren. Dazu muß erstmal festgestellt werden, wieviel

'Speicherplatz zur Zeit im Hauptspeicher überhaupt frei ist

```
'("Fre(0)"). Von dieser Bytezahl werden anschließend sicherheitshalber
'10.000 Byte und eine Ausschnittsgröße wieder zurückgegeben. Der
'verbleibende Rest wird durch den Speicherbedarf
'eines einzelnen Bildes geteilt. Damit erhalten wir die maximale
'Anzahl der Bilder, die gleichzeitig zur Verfügung stehen können.
```

```
,
```

```
@Set.pics
```

```
,
```

```
'Anschließend wird die gewünschte Bilderzahl von Ihnen festgelegt...
```

```
,
```

```
Graphmode 1
```

```
Erase P_icy$( )
```

```
Dim P_icy$(P_icys%+1)
```

```
,
```

```
'...und ein Stringfeld eingerichtet, das die einzelnen Bilder
```

```
'aufnehmen soll.
```

```
,
```

```
P_icscroll$=""
```

```
Return
```

```
Procedure Set.pics
```

```
Graphmode 1
```

```
Deffill ,0,0
```

```
Pbox 200,120,440,260
```

```
Box 201,121,439,259
```

```
Box 205,125,435,255
```

```
Deftext 1,16,0,13
```

```
Text 215,150,"Anzahl der Bilder :"
```

```
Deffill ,2,4
```

```
Pbox 220,160,420,200
```

```
Deffill ,0,0
```

```
Pbox 225,165,415,195
```

```
Box 227,167,413,193
```

```
Box 282,220,360,245
```

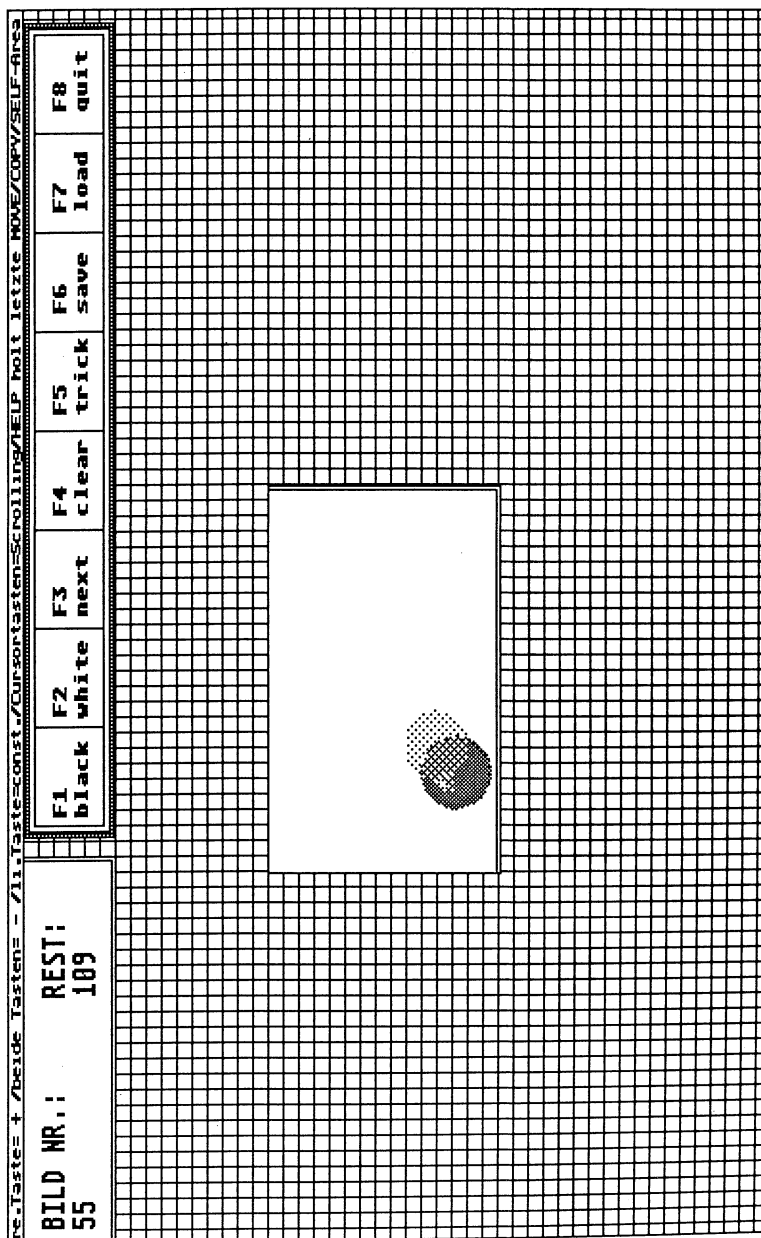
```
Box 283,221,359,244
```

```
Box 285,223,357,242
```

```
R.index%=3
```

```
@Dragbox(1,229,169,1,22,229,169,182,22)
```

```
Return
```

Procedure Get_pics

```

  '
  'Wir befinden uns nun quasi im "Atelier und Schneiderraum". Hier werden
  'die einzelnen Bilder gezeichnet bzw. auf verschiedene Weise editiert.
  '
  S.til%=2
  P.att%=8
  Radius%=1
  Defline 0,1,0,0
  Cls
  Deffill ,3,6
  Pbox 2,2,637,397
  Deffill ,0,0
  Pbox 2,8,200,56
  Box 2,8,198,54
  Deffill ,2,6
  Pbox 210,8,630,56
  Deffill ,0,0
  Pbox 213,11,627,53
  Pbox 2,0,637,8
  Deftext 1,1,0,6
  Restore M_enu.1
  For I=0 To 7
    Box 216+51*I,14,267+51*I,50
    Read M.enu.1$
    Read M.enu.2$
    Text 220+52*I,30,M.enu.1$
    Text 220+52*I,41,M.enu.2$
  Next I
  Deftext 1,1,0,4
    Text    3,6,"re.Taste=      +      /beide      Tasten=      -
/li.Taste=const./Cursortasten=Scrolling/HELP holt letzte MOVE/COPY/SELF-
Area"
  Pbox Scr_lft%-2,Scr_upn%-2,Scr_rgt%+4,Scr_dwn%+4
  Pbox Scr_lft%-2,Scr_upn%-2,Scr_rgt%+2,Scr_dwn%+2
  Deffill ,2,8
  Bildzahl=0
  Print At(2,2);"BILD NR.:      REST:"
  For Counter%=0 To P_icys%-1
    '

```

```
'Diese Schleife wird so oft durchlaufen, wie Bilder vorhanden sind,
' bzw. bis eine Abbruchfunktion aufgerufen wird.
```

```
'
```

```
Inc Bildzahl
```

```
Print At(2,3);Counter%+1
```

```
Print At(17,3);P_icys%-Counter%-1'
```

```
Zeichnen:
```

```
Graphmode 3
```

```
Deffill ,2,1
```

```
@C_lipprec2(0,Scr_lft%,Scr_upn%,Scr_rgt%,Scr_dwn%)
```

```
'
```

```
'Das "Clipping Rectangle" wird auf das Bildformat eingestellt, aber
' noch nicht aktiviert.
```

```
'
```

```
Repeat
```

```
'
```

```
  'Hier bleibt das Programm bis eine Maus- oder Tastaturtaste
  ' gedrückt wird.
```

```
'
```

```
  Mouse X,Y,Button%
```

```
  Pcircle X,Y,Radius%
```

```
  Pcircle X,Y,Radius%
```

```
  Key%=Inkey%
```

```
  Key%=Asc(Left$(Key%))+Asc(Right$(Key%))
```

```
  If Key%>58 Or Key%<67
```

```
    Hitkey%=1
```

```
  Endif
```

```
Until Mousek Or Hitkey%=1
```

```
Deffill ,S.til%,P.att%
```

```
@C_lipprec2(1,Scr_lft%,Scr_upn%,Scr_rgt%,Scr_dwn%)
```

```
'
```

```
""Clipping Rectangle" wird aktiviert.
```

```
'
```

```
Graphmode 1
```

```
Mouse X,Y,Button%
```

```
If Hitkey%=0
```

```
  Key%=Inkey%
```

```
  Key%=Asc(Left$(Key%))+Asc(Right$(Key%))
```

```
Endif
```

```
Hitkey%=0
```

```

Xpos%=Int((X-216)/51)+1
Ypos%=Int((Y-14)/36)+1
|
'Die bekannte Menüfeldindizierung.
|
If Key%=59 Or (Xpos%=1 And Ypos%=1 And Button%=1)
|
|   'Wird "F1" gedrückt oder das erste Menüfeld angeklickt, schalte auf
|   'schwarze Zeichenfarbe.
|
|   @Rahmen(1)
|   S.til%=2
|   P.att%=8
|   Deffill ,2,8
|   Goto Zeichnen
Endif
If Key%=60 Or (Xpos%=2 And Ypos%=1 And Button%=1)
|
|   '"F2" oder 2.Menüfeld = weiße Zeichenfarbe.
|
|   @Rahmen(0)
|   S.til%=0
|   P.att%=0
|   Deffill ,0,0
|   Goto Zeichnen
Endif
If Key%=61 Or (Xpos%=3 And Ypos%=1 And Button%=1)
|
|   '"F3" oder 3.Menüfeld = "photographiere" das aktuelle Bild.
|
|   Get Scr_lft%,Scr_upn%,Scr_rgt%,Scr_dwn%,P_icy$(Counter%)
|   Pause 10
|   Goto N_ext
Endif
If Key%=62 Or (Xpos%=4 And Ypos%=1 And Button%=1)
|
|   '"F4" oder 4.Menüfeld = setze eine weiße Box auf das Bild =
|   'löschen.
|
|   Deffill ,0,0

```

```

Pbox Scr_lft%-1,Scr_upn%-1,Scr_rgt%+1,Scr_dwn%+1
Deffill ,2,8
Goto Zeichnen
Endif
If Key%=63 Or (Xpos%=5 And Ypos%=1 And Button%=1)
|
|   ""F5" oder 5.Menüfeld = Auswahl des Trickfilm-Modus.
|   |
|   Alert 2,"Welcher Trick:                ",0,"BUBBLE|TROMMEL| KEIN
ER",Back%
|   If Back%=1
|       T.rickflag%=1
|   Endif
|   If Back%=3
|       T.rickflag%=0
|   Endif
|   If Back%=2
|       Alert 1,"Ich brauche dazu einen Grafik-|Ausschnitt. Befindet sic
h eine|Grafik auf Screen Eins ?",1," JA |NEIN",Vvvv
|       If Vvvv=1
|           Counter%=P_icys%
|           Trommelflag%=1
|           Goto N_ext
|       Else
|           Alert 1,"Laden oder zeichnen Sie bitte|erstmal eine Grafik auf
| SCREEN EINS !",1," OKAY ",Backalert%
|       Endif
|   Endif
Endif
If Key%=64 Or (Xpos%=6 And Ypos%=1 And Button%=1)
|
|   ""F6" oder 6.Menüfeld = speichern.
|   |
|   Alert 2,"Was soll gespeichert werden ?",0," FILM | BILD |ABBRUCH",
Ssss
|   If Ssss=<3
|       Dec Counter%
|       If Ssss=1
|           @P_icsave
|           If Selected$<>"\" And Selected$<>".FLM" And Selected$>""

```

```

        Counter%=P_icys%
        T.rickflag%=0
    Endif
Else
    @Bildsaver
Endif
Goto N_ext
Endif
Endif
If Key%=65 Or (Xpos%=7 And Ypos%=1 And Button%=1)
    ,
    ""F7" oder 7.Menüfeld = laden.
    ,
    Alert 2,"Was soll geladen werden? | 1. FILM|2. BLOCK/ICON/BILD ",
0," 1. | 2. |ABBRUCH",T.oload
    If T.oload=1
        Alert 1,"Durch den Ladevorgang geht|die aktuelle Edition|unwider
ruflich verloren !",1," NEIN | OKAY ",Backalert%
        If Backalert%=2
            L.oadflag%=1
            Counter%=P_icys%
            Goto N_ext
        Endif
    Endif
    If T.oload=2
        Alert 2,"Bitte wählen Sie:                ",0," BLOCK| ICON | B
ILD ",Mmmm
        If Mmmm=1
            @Blockloader
        Endif
        If Mmmm=2
            @Iconloader
        Endif
        If Mmmm=3
            @Bildloader
        Endif
        Key%=98
    Endif
Endif
If Key%=66 Or (Xpos%=8 And Ypos%=1 And Button%=1)

```

```

      '
      '"F8" oder 8.Menüfeld = Abbruch oder Film zeigen.
      '
      Alert 2,"Film soweit zeigen,|total quit oder|zurück ?",3,"ZEIGEN|
QUIT |ZURÜCK",B.ackalert%
      If B.ackalert%=2
          Kintopp.quit%=1
          Counter%=P_icsys%
          Goto N_ext
      Endif
      If B.ackalert%=1
          Zeigen%=1
          Counter%=P_icsys%
          Goto N_ext
      Endif
  Endif
  '
  'Die folgenden drei Abfragen beziehen sich auf die vier Cursor- bzw.
  'Pfeiltasten. Wird während des Zeichnens eine davon gedrückt, wird
  'der Inhalt des Bildes in die der Pfeilrichtung entsprechenden
  'Richtung verschoben (ge"scrollt").
  '
  If Key%=72
      Get Scr_lft%,Scr_upn%+1,Scr_rgt%,Scr_dwn%+1,P_icscroll$
      Put Scr_lft%,Scr_upn%,P_icscroll$
      Goto Zeichnen
  Endif
  If Key%=75
      Get Scr_lft%+1,Scr_upn%,Scr_rgt%+1,Scr_dwn%,P_icscroll$
      Put Scr_lft%,Scr_upn%,P_icscroll$
      Goto Zeichnen
  Endif
  If Key%=77
      Get Scr_lft%-1,Scr_upn%,Scr_rgt%-1,Scr_dwn%,P_icscroll$
      Put Scr_lft%,Scr_upn%,P_icscroll$
      Goto Zeichnen
  Endif
  If Key%=80
      Get Scr_lft%,Scr_upn%-1,Scr_rgt%,Scr_dwn%-1,P_icscroll$
      Put Scr_lft%,Scr_upn%,P_icscroll$

```

```

    Goto Zeichnen
Endif
If Key%=98
    '
    'Wird während des Zeichenvorgangs die "Help"-Taste gedrückt, wird
    'das (falls vorhanden) letzte Bild, welches mit dem Grafikprogramm
    'in den Rasterspeicher geholt wurde (MOVE/COPY- oder SELF-
    'Ausschnitt bzw. Block- oder Iconload) an der Mauszeigerposition
    'angezeigt. Sie können dieses Bild nun in das Film-Bild einsetzen.
    'Sollten Sie mehrere solcher Ausschnitte benötigen, können Sie
    'über die "Load"-Funktion im "Kino"-Menü beliebige Ausschnitte
    '(die Sie allerdings vorher dort gespeichert haben müßten) in Ihr
    'Film-Atelier übertragen. Sie können also ohne weiteres die
    'einzelnen Film-Bilder mit dem Komfort des Grafikprogramms
    'entwerfen, abspeichern und anschließend in Ihren Trickfilm
    'einbauen.
    '
    Repeat
        Mouse X,Y,K
        Put X,Y,Mc.part$,6
        Put X,Y,Mc.part$,6
    Until Mousek
    Put X,Y,Mc.part$,6
    Get Scr_lft%,Scr_upn%,Scr_rgt%,Scr_dwn%,P_icscroll$
    Put X,Y,Mc.part$,6
    Put Scr_lft%,Scr_upn%,P_icscroll$,3
    Pause 10
    Goto Zeichnen
Endif
If Button%=0 And Key%=0
    Goto Zeichnen
Endif
If Button%=2
    '
    'Wird während des Zeichnens die rechte Maustaste gedrückt,
    'vergrößert sich der Zeichenpunkt.
    '
    Inc Radius%
Endif
If Button%=3 And Radius%>0

```



```

'
'Drücken Sie beide Maustasten, wird er wieder kleiner.
'
Dec Radius%
Endif
If T.rickflag%=1
'
'Haben Sie den "Bubble"-Trick gewählt, wird hier das Bild eines
'Ballons mit Glanz und Schatten erzeugt, das auf Maustastendruck
' (links, rechts oder beide) gezeichnet wird. Müssen Sie im
'normalen Zeichenmodus die einzelnen Bilder selber weiterschalten,
'wird das hier automatisch erledigt. Es kann so zum Beispiel der
'Effekt eines in der Luft herumwirbelnden Ballons erzielt werden.
'
@Rahmen(0)
Deffill ,0,0
Pbox Scr_lft%,Scr_upn%,Scr_rgt%,Scr_dwn%
Deffill ,2,6
Pcircle X+Radius%/12,Y+Radius%/12,Radius%+Radius%/12
Deffill ,2,5
Pcircle X+Radius%/15,Y+Radius%/15,Radius%+Radius%/15
Deffill ,2,4
Pcircle X,Y,Radius%
Deffill ,2,3
Pcircle X-Radius%/5,Y-Radius%/5,Radius%/3
Deffill ,2,2
Pcircle X-Radius%/4,Y-Radius%/4,Radius%/5
Deffill ,2,1
Pcircle X-Radius%/3.9,Y-Radius%/3.9,Radius%/8
Deffill ,0,0
Pcircle X-Radius%/3.9,Y-Radius%/3.9,Radius%/11
Get Scr_lft%,Scr_upn%,Scr_rgt%,Scr_dwn%,P_icy$(Counter%)
@Rahmen(1)
Goto N_ext
Endif
Pcircle X,Y,Radius%
Goto Zeichnen
N_ext:
Next Counter%
Counter%=Bildzahl

```

```

Color 1
@C_lipprec2(0,Scr_lft%,Scr_upn%,Scr_rgt%,Scr_dwn%)
'
'Clipping wieder ausgeschaltet.
'
@Rahmen(1)
If Kintopp.quit%=0 And Loadflag%=0 And Zeigen%=0 And Trommelflag%=0
    Alert 2,"Film abspeichern,|oder erstmal sehen ?",1," SHOW | SAVE ",B
ackalert%
    If Backalert%=2
        @P_icsave
    Endif
Endif
Return
Procedure G.ettinny
'
'Prinzipiell ist dies die gleiche Prozedur, wie Sie sie in
'"Block.y.elly" kennengelernt haben. Diese wurde nur an die speziellen
'Erfordernisse angepaßt.
'
@Clipprec(0)
Graphmode 1
Cls
Defline 0,1,0,0
Print At(2,2);"Bitte Neigung bestimmen !"
Graphmode 3
Repeat
    Mouse Ell.xx,Ell.yy,K
    If Ell.yy<170
        Ell.yy=170
    Endif
    Yrad%=Ell.yy-170
    If Yrad%>30
        Yrad%=30
    Endif
    Ellipse 320,170+Yrad%,50,Yrad%
    Ellipse 320,170+Yrad%,50,Yrad%
Until Mousek
Counter%=200
Scr_wid%=100

```

```

Scr_hei%=80+(2*Yrad%)
Put 0,0,Screen.eins$,3
Erase P_icy$()
Dim P_icy$(200)
Alert 1,"Bitte Ausschnitt zeigen !      ",1," OKAY ",Backalert%
Pause 10
Defline 0,1,0,0
Graphmode 3
Repeat
    Mouse Xx%,Yy%,Button%
    Box Xx%-200,Yy%-80,Xx%,Yy%
    Box Xx%-200,Yy%-80,Xx%,Yy%
Until Mousek
Xl%=Xx%-200
Yl%=Yy%-80
Blocklines%=200
Alert 2,"Mit oder ohne Rand?",0," MIT | OHNE ",Backalert%
Color 1
Graphmode 1
If Backalert%=1
    Line Xl%,Yl%-1,Xx%,Yl%-1
    Line Xl%,Yy%+1,Xx%,Yy%+1
Endif
Erase Mir.mem$()
Dim Mir.mem$(Blocklines%)
For I%=1 To 200
    Get Xl%+I%-1,Yl%-1,Xl%+I%,Yy%+1,Mir.mem$(I%)
Next I%
Cls
Deffill ,2,4
Pbox 0,0,639,399
Deffill ,2,2
Pbox 20,0,619,359
Deffill ,0,0
@Rahmen(0)
Pbox 240,0,400,150
@Rahmen(1)
Print At(2,24);"      Bitte Geduld ! Es müssen noch      Einzelbildaufna
hmen gemacht werden.      "
Blockrows%=80

```

```

XL%=270
YL%=0
H_ypo%=50
H_ypo2%=Yrad%
Center.x%=XL%+H_ypo%
Center.y%=YL%+H_ypo%
Rastermod%=3
Line 0,YL%,600,YL%
Kr.yr%=YL%+2*Yrad%+81
If Kr.yr%>399
    Kr.yr%=399
Endif
Counter%=200
For Ii%=1 To Counter%
    '
    "Counter%" wurde vorher auf 200 Bilder festgelegt. Sooft wird
    'diese Schleife durchlaufen.
    '
    'Es werden hier von einem Bildausschnitt, den Sie von der Grafik-
    'SCREEN gewählt haben, 200 Einzelbilder gemacht (konstante
    'Ausschnittsbreite). Dieser Ausschnitt wird zu jeder Einzelaufnahme
    'um ein Pixel "weitergedreht".
    'Nach Fertigstellung werden Sie dann zur Betrachtung Ihres
    'Kunstwerks eingeladen ("Hit the Key"). Sie bekommen eine rotierende
    'Trommel zu sehen, auf deren Außenfläche sich der von Ihnen gewählte
    'Bildausschnitt befindet. Sie werden den verblüffenden Effekt
    'feststellen, daß Sie gleichzeitig in der Trommelwölbung Ihr Bild
    'von hinten und von vorne sehen können.
    '
    Mir.mem$(0)=Mir.mem$(Counter%)
    '
    'Das erste Element des Arrays, das die einzelnen Bildstreifen (200)
    'enthält, wird hier bei jedem Durchlauf mit dem Inhalt des letzten
    'Elements aufgefüllt.
    '
    For Jj%=Counter% Downto 1
        '
        'In dieser zweiten Schleife werden nun die einzelnen Bildstreifen
        'jeweils um ein Element nach vorne verschoben.
        'Dadurch wird erreicht, daß jeder einzelne Bildstreifen bei der

```

```

'Aufnahme einmal vorne im Bild war und sich die anderen in Ihrer
'korrekten Reihenfolge daran anschließen. Dies ist der Rotations-
'Effekt.
'
Mir.mem$(Jj%)=Mir.mem$(Jj%-1)
Next Jj%
@Tinny
'
'Anschließend wird jede der 200 Trommeln in "Tinny" errechnet.
'
Print At(36,24);200-Ii%'
Mem.index%=0
Get 270,1,370,Kr.yr%+1,P_icy$(Ii%)
Next Ii%
P_icspace%=Len(P_icy$(1))
Scr_lft%=320-(Scr_wid%/2)
Scr_upn%=200-(Scr_hei%/2)
Scr_rgt%=Scr_lft%+Scr_wid%
Scr_dwn%=Scr_upn%+(2*Yrad%)+80
Mem.index%=0
Erase Mir.mem$()
Return
Procedure P_icload
Fileselect "\*.flm","",Selected$
If Selected$<>"\" And Selected$<>".FLM" And Selected$>""
'
'Soll ein Film geladen werden...
'
Defmouse 2
Open "I",#1,Selected$
'
'...öffne die Film-Datei...
'
Identifier$=Input$(30,#1)
'
'...lese die ersten 30 Byte (Info-Block)...
'
Nums1%=Val?(Identifier$)
Counter$=Left$(Identifier$,Nums1%)
Counter%=Val(Counter$)

```

```

|
|...filtere daraus die Anzahl der Bilder, die der Film enthält...
|
Identifizier$=Right$(Identifizier$,30-Nums1%-1)
Nums2%=Val?(Identifizier$)
P_icspace$=Left$(Identifizier$,Nums2%)
P_icspace%=Val(P_icspace$)
|
|...die Bytegröße eines einzelnen Bildes...
|
Identifizier$=Right$(Identifizier$, (30-Nums1%-1)-Nums2%-1)
Nums3%=Val?(Identifizier$)
Scr_hei$=Left$(Identifizier$,Nums3%)
Scr_hei%=Val(Scr_hei$)
|
|...die Ausschnittshöhe...
|
Identifizier$=Right$(Identifizier$, ((30-Nums1%-1)-Nums2%-1)-Nums3%-1)
Nums4%=Val?(Identifizier$)
Scr_wid$=Left$(Identifizier$,Nums4%)
Scr_wid%=Val(Scr_wid$)
|
|...und die Ausschnittsbreite.
|
Erase P_icy$()
Dim P_icy$(Counter%+1)
On Error Gosub Disk.error
For I=0 To Counter%-1
    P_icy$(I)=Space$(P_icspace%)
    |
    'Bereite die einzelnen Array-Elemente auf die Speicherung vor...
    |
    P_icy$(I)=Input$(P_icspace%,#1)
    |
    '...lade die einzelnen Bilder und weise Sie den Elementen zu.
    |
Next I
Defmouse 0
No.dat:
On Error

```

```

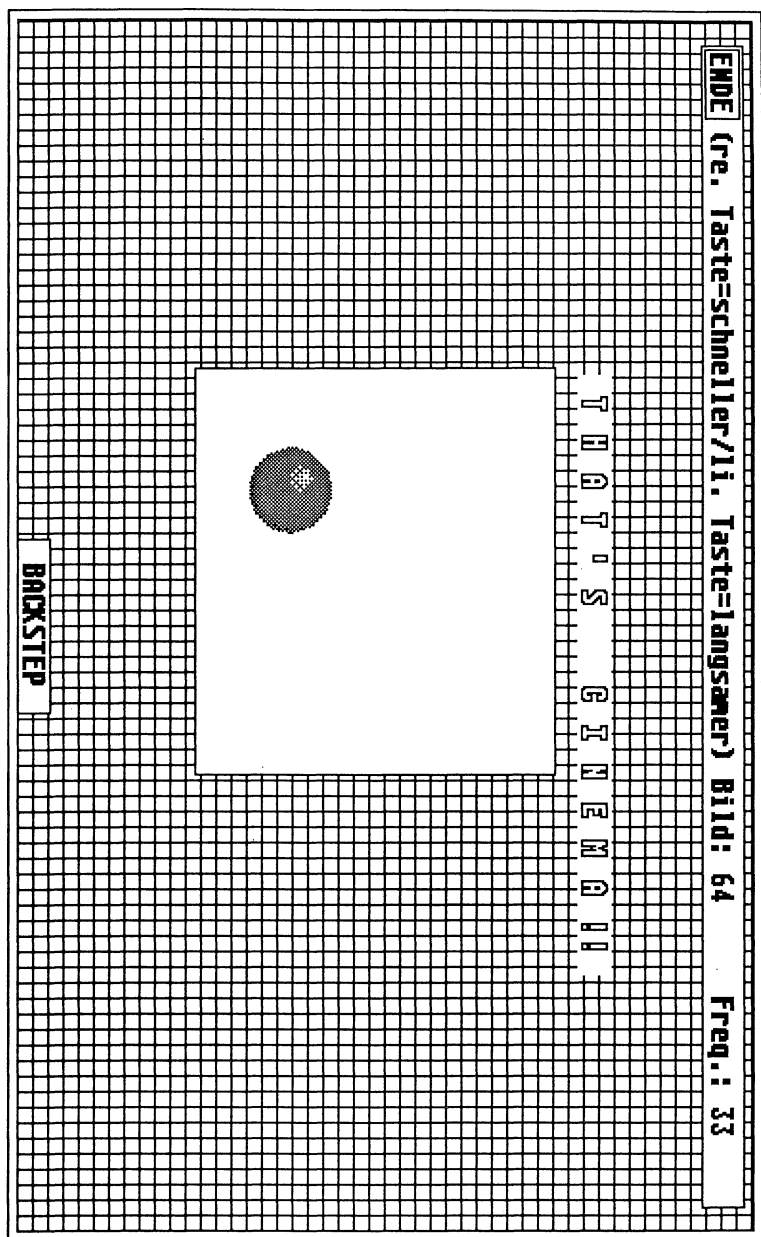
Close #1
Scr_lft%=320-(Scr_wid%/2)
Scr_upn%=200-(Scr_hei%/2)
Scr_rgt%=Scr_lft%+Scr_wid%
Scr_dwn%=Scr_upn%+Scr_hei%
'
'Berechne die Standort-Koordinaten der "Leinwand".
'
Endif
'
'Das in dieser Prozedur verwendete Verfahren, die Filmdatei und die
'einzeln Komponenten zu analysieren und zu laden, ist nicht sehr
'effektiv. Wesentlich einfacher wäre es, die einzelnen Bilder und die
'Info-Tabelle mit "Write #1" zu schreiben und mit "Input #1" wieder
'einzulesen. Das Analyseverfahren wurde jedoch absichtlich kompliziert
'gestaltet, um zu verdeutlichen, wie unter Umständen solche Dateien
'aufgebaut sein können. Sie werden es in Ihrem Programmiererdasein
'nicht immer nur mit GfA-BASIC und all seinen Vorzügen zu tun haben,
'sondern evtl. in die Verlegenheit kommen, eine Datei auf die hier
'demonstrierte Art "auseinanderzunehmen".
'
Return
Procedure P_icsave
'
'Hier wird der Film abgespeichert.
'
Startsave:
If Counter%*P_icspace%+30<Dfree(0)
'
'Allerdings nur, wenn der auf der Diskette verfügbare Speicherplatz
'auch dazu ausreicht. Es wäre schade um den schönen Film, wenn durch
'unvorhersehbare Errormeldungen das Programm abgebrochen würde.
'
Defmouse 2
K_ontrol$=Str$(Counter%)+""|"+Str$(P_icspace%)+""|"+Str$(Scr_hei%)+""|"+
+Str$(Scr_wid%)
K_ontrol$=K_ontrol$+Space$(30-Len(K_ontrol$))
'
'Es werden die oben bereits erwähnten ersten 30 Byte der Datei
'(Info-Block) in einem String zusammengefaßt.

```

```

'
Fileselect "\*.flm","",Selected$
E.xtender$="FLM"
@Extend
Open "0",#1,Selected$
Print #1,K_ontrol$;
'
'Schreibe den Info-String in die Datei...
'
For I=0 To Counter%
'
'...und anschließend alle Bilder.
'
Print #1,P_icy$(I);
Next I
Close #1
Defmouse 0
Goto S.aveout
Else
Alert 3,"Nicht genug Platz auf der|Diskette!",1,"NOCHMAL|ABBRUCH|LÖS
CHEN",Backal%
Endif
If Backal%=1
Goto Startsave
Endif
If Backal%=3
@K.ill
Alert 2,"Datei ist gelöscht !|Film abspeichern ?",1,"OKAY | NEIN ",
Uuuu
If Uuuu=1
Goto Startsave
Endif
Endif
S.aveout:
Return

```

Procedure Showpics

```

'
'Der "Vorführraum".
'
Graphmode 1
Deffill ,2,4
Pbox 0,0,639,399
Deffill ,0,0
Pbox 35,145,605,255
Deffill ,2,1
Pbox 40,150,600,250
Graphmode 2
Deftext 1,21,0,26
Text 50,205,540," H I T   T H E   K E Y ! "
Graphmode 1
Repeat
Until Mousek Or Len(Inkey$)
Deffill ,0,0
For I=320 To 630 Step 32
    Pbox 300-(Abs(320-I)),0,I+21,399
Next I
Cls
Pbox 2,2,637,397
Deffill ,3,6
Pbox 7,7,632,392
Deffill ,0,0
Pbox 20,12,620,34
Pbox 275,375,365,392
Box 22,14,58,32
Deftext 1,1,0,13
Graphmode 2
Text 23,29,"ENDE (re. Taste=schneller/li. Taste=langsamer) Bild:
    Freq.:"
Text 287,389,"BACKSTEP"
Graphmode 1
Deffill ,2,4
Deftext 1,16,0,13
Deffill ,0,0
Text Scr_lft%-20,Scr_upn%-40," T H A T ' S   C I N E M A ! ! "
Pbox Scr_lft%-25,Scr_upn%-25,Scr_rgt%+25,Scr_dwn%+25
```

```
Do
|
|Endlosschleife. D.h., der Film fängt immer wieder von vorne an.
|
For Bild%=1 To Counter%
|
|Schleife für den ganzen Filmdurchlauf.
|
T%=Timer
Pause P.ause%/2
Mouse X%,Y%,Button%
Put Scr_lft%,Scr_upn%,P_icy$(Bild%)
|
|Zeige die Bilder in ihrer Reihenfolge.
|
If Y%=399
|
|Wird während der Vorführung der untere Bildschirmrand berührt,
|halte das Programm an und warte auf einen Mausklick
|
|(Standbild). Lassen Sie den Mauzeiger am unteren Bildschirmrand
|und drücken Sie mehrmals die Maustaste, erhalten Sie eine
|Einzelbild-Projektion.
|
|
|
|@Clicks
Endif
If Button%=1
|
|Wird die linke Maustaste gedrückt...
|
If Y%<32
If X%<58
|
|...und der Mauszeiger befindet sich auf der "Ende"-Box...
|
Bild%=Counter%
Goto No.show
|
|...beende die Vorführung.
|
```

```

    Endif
Endif
If Y%>375
    If X%>275 And X%<365
        |
        |'...wenn der Mauszeiger sich auf der "Backstep"-Box
        |'befindet...
        |
        |
        Sub Bild%,2
            |
            |'...vermindere den Bild-Durchlaufzähler um 2. (Bildrücklauf)
            |
            If Bild%<=0
                Bild%=0
            Endif
            Goto C_onti
        Endif
    Endif
Endif
If Button%=1 And P.pause%<30
    |
    |'Ist keiner der oberen Fälle eingetroffen und die linke
    |'Maustaste gedrückt, verlangsame den Film (Slow-Motion).
    |
    |
    Inc P.pause%
Endif
If Button%=2 And P.pause%>=1

    |'Wird die rechte Maustaste gedrückt, laß ihn wieder schneller
    |'laufen. Es ist je nach Bildgröße nur eine bestimmte maximale
    |'Bildfrequenz möglich. Schneller geht's dann wirklich nicht
    |'mehr.

    Dec P.pause%
Endif
C_onti:
T_time%=Timer-T%
Print At(57,2);"      "
Print At(57,2);Bild%+1'

```

```

    If T_time%>0
        Frequence%=Int(10*200/(Timer-T%)+0.5)/10
        Print At(72,2);Frequence%'
    Endif
    No.show:
Next Bild%
Exit If Button%=1 And X%<58 And Y%<32
Loop
Alert 2,"Film abspeichern,|Film neu zeichnen,|oder total Quit ?",1," S
AVE | NEU | QUIT",B.alert%
If B.alert%=1
    Backalert%=0
    @P_icsave
Endif
If B.alert%=3
    Kintopp.quit%=1
Endif
Return
Procedure Bildloader
    Fileselect "%*.bil",".bil",Selected$
    If Selected$<>"\BIL" And Selected$>"" And Selected$<>"\"
        Defmouse 2
        @D.size(Selected$)
        Mc.part$=Space$(Datsize%)
        Bload Selected$,Varptr(Mc.part$)
        Defmouse 0
    Endif
Return
Procedure Bildsaver
    Fileselect "%*.bil",".bil",Selected$
    If Selected$<>"\BIL" And Selected$>"" And Selected$<>"\"
        Defmouse 2
        E.xtender$="BIL"
        @Extend
        Get Scr_lft%,Scr_upn%,Scr_rgt%,Scr_dwn%,Mc.part$
        Bsave Selected$,Varptr(Mc.part$),Len(Mc.part$)
        Defmouse 0
    Endif
Return
Procedure Iconloader

```

```

Fileselect "*.icn",".icn",Selected$
If Selected$<>"\*.ICN" And Selected$>"" And Selected$<>"\
  Defmouse 2
  @D.size(Selected$)
  Mc.part$=Space$(Datsize%)
  Bload Selected$,Varptr(Mc.part$)
  Defmouse 0
Endif
Return
Procedure Iconsaver
Fileselect "*.icn",".icn",Selected$
If Selected$<>"\*.ICN" And Selected$>"" And Selected$<>"\
  Defmouse 2
  E.xtender$="ICN"
  @Extend
  Get Mic.b.xl+3,Mic.b.yl+3,Mic.b.xr-3,Mic.b.yr-3,Mc.part$
  Bsave Selected$,Varptr(Mc.part$),Len(Mc.part$)
  Defmouse 0
Endif
Return
Procedure Blockloader
Fileselect "*.blo",".blo",Selected$
If Selected$<>"\*" And Selected$<>"*.BLO" And Selected$>""
  Defmouse 2
  @D.size(Selected$)
  Mc.part$=Space$(Datsize%)
  Bload Selected$,Varptr(Mc.part$)
  Defmouse 0
Endif
Return
Procedure Patternsaver
Fileselect "*.pat",".pat",Selected$
If Selected$<>"\*.PAT" And Selected$>"" And Selected$<>"\
  Defmouse 2
  E.xtender$="PAT"
  @Extend
  @Busy
  @Initarray
  @Patedit
  Bsave Selected$,Varptr(Patword%(0)),64

```

```

@Ready
Defmouse 0
Endif
Return
Procedure Dragbox(Dragflag%,Dragxinit%,Dragyinit%,Dragbreite%,Draghoch%,X
bound%,Ybound%,Boundbreite%,Boundhoch%)

```

'Sie werden im Programm mehrmals aufgefordert, anhand eines Schiebers
'bestimmte Werte einzustellen (z.B. NETZMODUS AN). Dieser Schieber
'wird in dieser Prozedur erzeugt.

```

Dragstart:
Graphmode 3
Color 1
Defline 0,1,0,0
If Dragflag%>0
Repeat
Box Dragxinit%,Dragyinit%,Dragxinit%+Dragbreite%,Dragyinit%+Draghoc
h%
Box Dragxinit%,Dragyinit%,Dragxinit%+Dragbreite%,Dragyinit%+Draghoc
h%
Until Mousek
Endif
Repeat
Mouse Dragx%,Dragy%,Dragkey%
If Dragx%>Xbound%+Boundbreite%
Dragx%=Xbound%+Boundbreite%
Endif
If Dragy%>Ybound%+Boundhoch%
Dragy%=Ybound%+Boundhoch%
Endif
If Dragx%<Xbound%+Dragbreite%
Dragx%=Xbound%+Dragbreite%
Endif
If Dragy%<Ybound%+Draghoch%
Dragy%=Ybound%+Draghoch%
Endif
Box Dragx%-Dragbreite%,Dragy%-Draghoch%,Dragx%,Dragy%
Box Dragx%-Dragbreite%,Dragy%-Draghoch%,Dragx%,Dragy%
If R.index%=0

```

```

    Netzweite%=Int((Dragx%-Dragxinit%-6)/3)
    Print At(40,15);Netzweite'
Endif
If R.index%=1
    Zeilen=Int((Dragx%-Dragxinit%+28)/2.6)
    If Zeilen Mod 2<>0
        Inc Zeilen
    Endif
    Print At(40,15);Zeilen'
Endif
If R.index%=2
    Spalten=Int((Dragx%-Dragxinit%+23)/3.4)
    Print At(40,15);Spalten'
Endif
If R.index%=3
    P_icys%=Int((Dragx%-Dragxinit%)/((Boundbreit%)/Max_pics%))
    Print At(39,15);P_icys'
Endif
Until Dragkey%=0
If R.index%=3 And P_icys%<2
    Alert 0,"Zuwenig !!!",1,"OKAY",Backalert%
    Goto Dragstart
Endif
Dragxl%=Dragx%-Dragbreit%
Dragyl%=Dragy%-Draghoch%
Dragxr%=Dragx%
Dragyr%=Dragy%
Return
Procedure Rubbself

'Wenn Sie im Grafikprogramm Boxen "aufziehen" müssen ("Rubb"-Modus oder
'MOVE/COPY-Funktionen), dann werden die dehnbaren Rahmen, die dem
'Mauszeiger folgen, in dieser Prozedur erzeugt.

Do

    'In dieser Schleife bleibt das Programm solange, bis entweder eine
    'Mautaste gedrückt, oder der Draw-Modus ("D.rawmod%") verändert
    'wurde.

```



```
Kk%=D.rawmode%
@Getfkeys
If Fkeys%=67 Or Kk%<>D.rawmode%
    Goto Rubbself.out
Endif
@Clipprec(2)
Mouse Xx%,Yy%,Kk%
```

'Die "Border"-Funktionen sind Ihnen bereits aus "Verteiler" bekannt.

```
If Border%=1
    If Xx%<1 And Yy%<300 And Yy%>100
        Click%=0
        @Clicks
        If Click%=2
            @Alterscreen
        Endif
    Endif
    If Xx%<1 And Yy%>398 And Clipp.set%=0 And Areas%=0
        Click%=0
        @Clicks
        If Click%=2
            Inc Clou
            If Clou=1
                R.emem=D.rawmode%
                D.rawmode%=10
            Endif
            If Clou=2
                Clou=0
                D.rawmode%=R.emem
            Endif
        Endif
        Goto Rubbself.out
    Endif
    If Xx%>638 And Yy%<300 And Yy%>100
        Click%=0
        @Clicks
        If Click%=2
            If Flipp.flag%=0
                Get 0,0,639,399,Screen.eins$
```

```

    @Starter
    Put 0,0,Screen.eins$,3
Endif
If Flipp.flag%=1
    Get 0,0,639,399,Screen.zwei$
    @Starter
    Put 0,0,Screen.zwei$,3
Endif
Endif
Endif
If Xx%>638 And Yy%>398
    Click%=0
    @Clicks
    If Click%=2
        Inc Koordi
        If Koordi=1
            Showkoordi=1
        Endif
        If Koordi=2
            Koordi=0
            Showkoordi=0
        Endif
    Endif
Endif
Endif
If X>270 And X<370 And Y>398
    Click%=0
    @Clicks
    If Click%=2
        @Showhelp
    Endif
Endif
Endif

```

'Ebenso die Netzmodus- und Fadenkreuz-Funktionen.

```

If Netzmod%=1
    Xx%=Int(Xx%/Netzweite%)*Netzweite%-Int(Netzweite%/2)
    Yy%=Int(Yy%/Netzweite%)*Netzweite%-Int(Netzweite%/2)
Endif
If Help.on%=0

```

```

If Showlines%=1
  X=Xx%
  Y=Yy%
  @S.howlines
Endif
If Const.lines%=1
  @C.onst.lines
Endif
Endif
Graphmode 1
Exit If Kk%=1 Or Fkeys%>0 Or D.rawmode%<11 Or D.rawmode%>19
Loop
Xl%=Xx%
Yl%=Yy%
If Fkeys%=0 And (D.rawmode%>10 And D.rawmode%<20) Or Areas%=1 Or Clipp
.set%=1

```

'Wenn keine Tastatur-Taste gedrückt wurde und entweder eine "Rubb"-
'Funktion oder "MOVE/COPY" eingeschaltet ist...

```

Graphmode 3
Norubb%=0
Rubbflag%=0
If Areas%=0 And Clipp.set%=0
  Rubbflag%=1
Endif
Defline 1,1,0,0
Deftext 1,0,0,6
Do

```

'...dann bleibe in dieser Schleife, bis entweder keine oder beide
'Maustasten gedrückt wird.

```

Mouse Xx%,Yy%,Kk%
If Xx%<Xl%
  Xx%=Xl%
Endif
If Yy%<Yl%
  Yy%=Yl%
Endif

```

'Wenn der Mauszeiger sich über dem Startpunkt oder links davon
'befindet, reduziere die Koordinaten auf den Nullpunkt.

If Netzmod%=1

Xx%=Int(Xx%/Netzweite%)*Netzweite%-Int(Netzweite%/2)

Yy%=Int(Yy%/Netzweite%)*Netzweite%-Int(Netzweite%/2)

If Netzweite% Mod 2<>0

Xx%=Xx%-1

Yy%=Yy%-1

Endif

Endif

If Areas%=1 Or Clipp.set%=1

'Wenn MOVE/COPY an ist, zeichne eine Doppelbox in XOR.

Box Xl%,Yl%,Xx%,Yy%

Box Xl%,Yl%,Xx%,Yy%

Endif

If Areas%=0 And Clipp.set%=0

If D.rawmode%>10 And D.rawmode%<14

'Doppelbox mit scharfen Ecken.

Box Xl%,Yl%,Xx%,Yy%

Box Xl%,Yl%,Xx%,Yy%

Endif

If D.rawmode%>13 And D.rawmode%<17

'Doppelbox mit runden Ecken.

Rbox Xl%,Yl%,Xx%,Yy%

Rbox Xl%,Yl%,Xx%,Yy%

Endif

If D.rawmode%>16 And D.rawmode%<20

'Doppel-Ellipse.

Xrad%=Abs(Xl%-Xx%)

Yrad%=Abs(Yl%-Yy%)

```

    Ellipse Xx%,Yy%,Xrad%,Yrad%
    Ellipse Xx%,Yy%,Xrad%,Yrad%
Endif
Endif
If Showkoord=1

'Falls die Koordinaten angezeigt werden sollen (im Border-Modus
'Bildschirmcke links unten berührt und Doppelklick ausgeführt),
'errechne die Abstände und zeige Sie an.

Deftext 1,1,0,13
L.o.h%=Abs(Xx%-Xl%)
L.o.v%=Abs(Yy%-Yl%)
L.o.d%=Sqr(L.o.h%^2+L.o.v%^2)
L.o.l$="X:"+Str$(L.o.h%)+ " Y:"+Str$(L.o.v%)+ " D:"+Str$(L.o.d%)
Text Xx%-Int(Xx%/4.5),Yy%-Int((Yy%-200)/15),L.o.l$
Text Xx%-Int(Xx%/4.5),Yy%-Int((Yy%-200)/15),L.o.l$
Text 3,15,Str$(Xx%)+ " / "+Str$(Yy%)
Text 3,15,Str$(Xx%)+ " / "+Str$(Yy%)
Endif
If Kk%=3
    Norubb%=1
Endif
Exit If Kk%=0 Or Kk%=3
If Showlines%=1
    X=Xx%
    Y=Yy%
    @S.howlines
Endif
If Const.lines%=1
    @C.onst.lines
Endif
Loop
Rubbself.out:
@Set.paras
Endif
Return
Procedure Divbycero
    If Err=0

```

```

    Alert 3,"Es muß wenigstens einmal eine|Rastergröße bestimmt werden !
",1," OKAY ",Backalert%
Endif
Resume E.dstart
Return
Procedure Disk.error
    If Err=26
        Alert 1,"Anscheinend ist in der|angegebenen Datei nichts|mehr zu fin
den!|Evtl. nochmal versuchen !?",1," OKAY ",Backalert%
    Else
        Alert 1,"Es ist ein Diskettenfehler|aufgetreten !|Nochmal von vorn !
|Aber bitte mit Gefühhl !",1," OKAY ",Backalert%
    Endif
    Kintopp.quit%=1
    Resume No.dat
Return
Procedure L.oad.error
    If Err=-33
        Alert 1,"Die angegebene Datei ist|auf der aktuellen Diskette|  nic
ht zu finden !|Evtl. Diskette wechseln !",2,"Nochmal|Weiter",Backalert%
    Endif
    If (Err<>-33 And E.rorop<>1) Or (Err<>-33 And E.rorop=1 And Err<>10)
        Alert 1,"Ich weiß nicht genau, was|Sie falsch gemacht haben !|Auf je
den Fall war es| nicht richtig !!",1," OKAY ",Backalert%
    Endif
    If E.rorop=0
        Startflag%=1
        Resume Next
    Endif
    If E.rorop=1
        If Err=10
            Alert 3,"Die Menü-Icon-
Dateien befinden|sich nicht auf der aktuellen|Diskette !|Evtl. Diskette
wechseln !",2,"Neustart| Weiter ",Backalert%
        Endif
        If Backalert%=1
            Erase M.icon$( )
            Resume Enter.0
        Else
            Resume Enter.1

```

```

    Endif
Endif
If E.rrorp=2
    If Backalert%=1
        Resume Enter.1
    Else
        Resume Enter.2
    Endif
Endif
Return
Procedure S.ave.error
    Alert 1,"Ich weiß nicht genau, was|Sie falsch gemacht haben !|Auf jede
n Fall war es| nicht richtig !!",1," OKAY ",Backalert%
    Startflag%=1
    Resume Next
Return
Procedure Div.cero
    If Err=0
        Alert 3,"Es ist eine Null-Division| aufgetreten! |Die gewünschte Fu
nktion kann|nicht ausgeführt werden!",1," OKAY ",Backalert%
    Else
        E.rror=Err
        Error E.rror
    Endif
    If D.iverr=1
        Resume E.dstart
    Else
        Resume Next
    Endif
Return
Procedure Accessory

```

'Wenn Sie ein Bild ausdrucken lassen wollen, meldet sich eine Alert-Box, die Sie danach fragt, ob die Desktop-Accessories aufgerufen werden sollen (evtl. zur Druckeranpassung). Wenn beim Systemstart Accessories im Speicher installiert wurden, können diese mit dieser Prozedur aufgerufen und bedient werden. Da unter dem Abbruch-Menü ("DO IT") auch ein Menüpunkt "NOT" existiert, können Sie also die Accessories auch aufrufen, ohne eine SCREEN drucken zu wollen. Durch "NOT" kehrt das Programm wieder zur normalen Zeichenebene zurück.

'Bei dieser Prozedur handelt es sich um die GEM-Pulldown-Menü-
 'Funktionen, die über verschiedene BASIC-Befehle angesprochen werden
 'können. Dieses soll hier jedoch nicht näher erläutert werden, da wir
 'uns mit diesem Thema an anderer Stelle ausführlicher auseinander-
 'setzen werden.

```

Pause 10
@P.utflip
Erase Q_utter%()
Dim Q_utter%(1)
Menu M_feld$()
On Menu Gosub Menue
Q_utter%(0)=0
Do
  On Menu
  Exit If Q_utter%(0)>0
Loop
Menu Kill
Return
Procedure Menue
  If M_feld$(Menu(0))=" N O W "
    Q_utter%(0)=1
  Endif
  If M_feld$(Menu(0))=" N O T "
    Q_utter%(0)=2
  Endif
  Menu Off
Return
Procedure Extend

```

'In dieser kleinen Hilfsprozedur wird bei der Eingabe von Dateinamen
 'überprüft, ob der vom Anwender eingegebene Extender dem im Programm
 'vorgegebenen (E.xtender\$) entspricht. Tut er das nicht, wird der
 'falsch eingegebene Extender vom Programm durch den vorgegebenen
 'ersetzt.

```

If Instr(Selected$,".")=0
  Selected$=Selected$+"."+E.xtender$
Endif
If Right$(Selected$,3)<>E.xtender$

```



```

P_os=Instr(Selected$,".")
Selected$=Left$(Selected$,P_os)+E.xtender$
Endif
Return
Procedure K.ill

```

'Eine weitere Hilfsprozedur ist diese. Wenn Sie während Ihrer
'Editorarbeit eine Datei löschen möchten und nicht den ganzen
'Pfadnamen mit der Hand eintippen wollen, geben Sie im Direkt-Modus
'einfach "aK.ill" ein, und Sie können die zu löschende Datei aus-
'wählen. Klicken Sie die "OK"-Box an, wird die Datei gelöscht. Aller-
'dings müssen Sie diese Prozedur im Arbeitsspeicher haben, um sie
'aufrufen zu können!

```

Fileselect "*.*", "", F.kill$
If F.kill$ <> "" And F.kill$ <> ""
    Kill F.kill$
Endif
Return

```

'Es folgen einige reine GEM-Funktionen, für die es keinen BASIC-Befehl
'gibt. Diese Funktionen werden hier nur andeutungsweise erklärt. Für
'weitere Informationen zu GEM schauen Sie bitte im entsprechenden
'Kapitel nach.

Procedure Showmouse

'Mit dieser Funktion wird der mit "Hidemouse" deaktivierte Mauszeiger
'wieder sichtbar gemacht. Geben Sie bei "Dpoke Intin" statt der 0
'eine 1 ein, müssen, um den Mauszeiger wieder sichtbar zu machen,
'genausoviel "Showmouse"- wie "Hidemouse"-Aufrufe stattfinden. Soll
'der Zeiger grundsätzlich bei jedem "Showmouse"-Aufruf sichtbar sein,
'muß der Aufruf-Zähler ausgeschaltet werden (Intin,0).

```

Dpoke Contrl+2,0
Dpoke Contrl+6,1
Dpoke Contrl+12,2
Dpoke Intin,0
Vdisys 122
Return

```

Procedure Hidemouse

'Unter Umständen kann es notwendig werden, den Mauszeiger vollständig
'vom Bildschirm zu entfernen (z.B. im Grafikprogramm beim SCREEN-
'Laden). Da die BASIC-Funktion "Defmouse" den Mauszeiger nicht völlig
'ausschaltet, sondern, selbst wenn Sie den Defmouse-String auf Null
'setzen, nur die Form und Maske verändert, ist diese GEM-Funktion
'manchmal sehr von Nutzen.

Dpoke Contrl+2,0
Dpoke Contrl+6,0
Dpoke Contrl+12,2
Vdisys 123

Return

Procedure Clipprec(Cl.stat%)

'Im Programm wird häufiger diese Prozedur aufgerufen. Sie bewirkt, wie
'schon erwähnt, daß alle Grafikausgaben auf einen bestimmten Bild-
'schirmausschnitt beschränkt werden. Alles, was über diesen Bereich
'hinausgeht, wird abgeschnitten. Bei der Parameterübergabe wird hier
'nur eine lokale Variable angesprochen. Dies hat den Grund, daß hier
'nur die jeweiligen Ausschnitte an- oder ausgeschaltet werden, ohne
'jedesmal die ganze Koordinatenliste mit Übergeben zu müssen. "An- und
'ausschalten" ist hier eigentlich nicht die richtige Bezeichnung. Es
'wird nämlich nur zwischen der Clipp-Box, die kleiner ist als das
'Bildschirmformat und einer anderen, die exakt die Bildschirmgröße
'hat, hin- und hergeschaltet.

Dpoke Contrl+2,4
Dpoke Contrl+6,1
Dpoke Contrl+12,2
Dpoke Intin,1
If Cl.stat%=1
 Dpoke Ptsin,Cl.xl%
 Dpoke Ptsin+2,Cl.yl%
 Dpoke Ptsin+4,Cl.xr%
 Dpoke Ptsin+6,Cl.yr%
Vdisys 129

Else

Dpoke Ptsin,0

```
Dpoke Ptsin+2,0
Dpoke Ptsin+4,639
Dpoke Ptsin+6,399
Vdisys 129
Endif
Return
Procedure C_lipprec2(Cl.stat,Cl.xl,Cl.yl,Cl.xr,Cl.yr)
```

'Soll eine Clipp-Box neu definiert werden, wird diese Prozedur aufgerufen. Sie ist in den Array-Initialisierungen identisch mit der vorangegangenen Prozedur. Dieser Prozedur muß allerdings eine komplette Koordinatenliste übergeben werden.

```
Dpoke Contrl+2,4
Dpoke Contrl+6,1
Dpoke Contrl+12,2
Dpoke Intin,Cl.stat
Dpoke Ptsin,Cl.xl
Dpoke Ptsin+2,Cl.yl
Dpoke Ptsin+4,Cl.xr
Dpoke Ptsin+6,Cl.yr
Vdisys 129
Return
Procedure Rahmen(Visibel%)
```

'Das GfA-BASIC sieht keine Möglichkeit vor, eine gefüllte Pbox, bzw. 'Prbox, Pcircle, Pellipse oder Polyfill zu erzeugen, ohne das der 'Rahmen sichtbar ist. Mit dieser Funktion können Sie den Rahmen der 'Pbox usw. ein- und ausschalten (Visibel=1 => an / Visibel=0 => aus).

```
Dpoke Contrl+2,0
Dpoke Contrl+6,1
Dpoke Contrl+12,2
Dpoke Intin,Visibel%
Vdisys 104
Return
Procedure G_rubberbox(Rubbxlo%,Rubbylo%,Rubbxmin%,Rubbymin%)
```

'Dieses ist die Original-GEM-Rubberbox. Ihr wird die Koordinate der

'linken oberen Ecke der gewünschten Rubberbox, sowie die minimale
 'Breite und Höhe übergeben, die die Box einnehmen darf.
 'Diese Prozedur kann nur bei gedrücktem Mausknopf aufgerufen werden.
 'Solange Sie einen Mausknopf gedrückt halten, folgt die rechte untere
 'Ecke der Box dem Mauszeiger. Lassen Sie die Maustaste los, ist die
 'Funktion beendet.
 'In den Variablen "Rubbxl" und "Rubbyl" befinden sich anschließend die
 'Koordinaten der linken oberen Boxecke (unverändert), und in den Vari-
 'ablen "Rubbxr%" und "Rubbyr%" finden Sie die Koordinaten der rechten
 'unteren Boxecke zum Zeitpunkt des "released button" (release, engl.:
 'loslassen / button, engl.: Knopf).

```
Dpoke Gintin,Rubbxlo%
Dpoke Gintin+2,Rubbylo%
Dpoke Gintin+4,Rubbxmin%
Dpoke Gintin+6,Rubbymin%
Gemsys 70
Rubbxl%=Rubbxlo%
Rubbyl%=Rubbylo%
Rubbxr%=Rubbxl%+Dpeek(Gintout+2)
Rubbyr%=Rubbyl%+Dpeek(Gintout+4)
```

Return

Procedure Grow.shrink.box(Gs.flag%,Litx%,Lity%,Litb%,Lith%,Bigx%,Bigy%,B
 igb%,Bigh%)

'In dieser Prozedur werden die lustigen und so effektiv dynamisch
 'wirkenden Springboxen produziert, wie Sie sie beim Öffnen von Dateien
 'oder Fenstern auf dem Desktop beobachten können.

'Der Prozedur wird eine ganze Reihe von Parametern übergeben.

- ' 1. Das Gs.flag% besagt, welche Funktion aufgerufen werden
 ' soll
- ' 0 = Growbox (grow, engl.:wachsen)
- ' 1 = Shrinkbox (shrink, engl.:schrumpfen)
- ' In beiden Fällen gibt es eine große und eine kleine Box.
- ' Bei Growbox ist die kleine der Ausgangspunkt und die
 ' große der Zielpunkt.
- ' Bei Shrinkbox ist es genau umgekehrt.
- ' 2. Litx% enthält die X-Koordinate der linken oberen Ecke der
 ' kleinen Box.
- ' 3. Lity% ist die Y-Koordinate derselben Ecke.

- ' 4. Litb% enthält die Breite der kleinen Box in Pixels.
- ' 5. Lith% enthält deren Höhe.
- ' 6./7./8./9. enthalten die entsprechenden Werte für die große Box.

```

Dpoke Gintin,Litx%
Dpoke Gintin+2,Lity%
Dpoke Gintin+4,Litb%
Dpoke Gintin+6,Lith%
Dpoke Gintin+8,Bigx%
Dpoke Gintin+10,Bigy%
Dpoke Gintin+12,Bigb%
Dpoke Gintin+14,Bigh%
If Gs.flag%=1
  Gemsys 74
Else
  Gemsys 73
Endif

```

```
Return
```

```
Procedure Formalert(Alicon,Defbutton,Alboxtext$,Albutttext$)
```

```

'Hier finden Sie die Original-GEM-Alertbox.
'Wenn Sie probieren möchten, was man mit dieser Box alles anstellen
'kann, machen Sie sich bitte auf eine Reihe Abstürze gefaßt, bis Sie
'die Alertbox-Grundgesetze kennengelernt haben. Die BASIC-Alertbox
'kann keinen Absturz verursachen, da die Eingaben vom Interpreter
'überprüft und gegebenenfalls gekürzt oder verändert werden.
'Diese Box dagegen ist ein wahres Absturzwunder, wenn man etwas
'leichtsinnig mit ihr umgeht. Hat man allerdings erstmal die wunden
'Punkte erkannt, ist sie wesentlich variabler als die BASIC-Alertbox
'(drücken Sie im Programm doch mal beim Zeichnen die "Help"-Taste).
'Dieser Prozedur werden die gleichen Parameter übergeben, wie Sie sie
'vom "Alert"-Befehl her kennen. Allerdings in einer anderen Reihen-
'folge. Diese Reihenfolge in der Variablenliste oben können Sie aller-
'dings nach eigenem Gutdünken verändern. Die Parameter sind dann beim
'Prozedur-Aufruf allerdings der neuen Reihenfolge entsprechend zu
'übergeben. Schauen wir uns die Struktur des "Alert"-Befehls an:
'
'Alert, Index des Info-Icons, Boxtext, Index des schwarz umrandeten
'Buttons (Default-Button), Buttontext, Rückgabeveriable für den tat-
'sächlich geklickten Button.

```

```

'
'Der Aufruf dieser Prozedur sieht ganz ähnlich aus:
'
'@Formalert(Index des Info-Icons, Index des Default-Buttons, Boxtext,
'ButtonText)
'
'Die Rückgabeveriable existiert hier nicht. Diese wird nach Abschluß
'der Boxbedienung definiert. Und zwar können Sie, nachdem das Programm
'aus dieser Prozedur zurückgekehrt ist, den Index des tatsächlich ge-
'klickten Buttons aus der Variablen "A.select" auslesen.
'
'Als Trennungszeichen der einzelnen Zeilen im Boxtext bzw. der Worte
'des Buttontextes gilt hier ebenfalls wie in "Alert" der senkrechte
'Strich.
'Statt maximal 30 Zeichen pro Zeile des Boxtextes können Sie hier
'maximal 40 Zeichen eingeben (im Normalfall).
'Und statt 8 Zeichen je Buttontext sind hier 10 Zeichen erlaubt (im
'Normalfall).

Alstring$="["+Str$(Alicon)+"] ["+Alboxtext$+"] ["+Albutton$+"]"+Chr$(0)
Lpoke AddrIn,VarPtr(Alstring$)
Dpoke GintIn,Defbutton
Gemsys 52
A.select=Dpeek(GintOut)
Return
Procedure Clicks
Dpoke GintIn,2
Dpoke GintIn+2,3
Dpoke GintIn+4,1
Gemsys 21
Click%=Dpeek(GintOut)
Return
Procedure Smalltext(Txtsize%)

txtsize% = 1 ==> 8*8 - FONT    /    txtsize% = 2 ==> 8*16 -
FONT

Match$=Mki$(&HA000)+Mki$(&H2009)+Mki$(&H4E75)
A%=VarPtr(Match$)
M.adr%=C:A%()
```

'Dieses kleine Maschinenprogramm nutzt ein sogenannte "Line A"-
'Variable zur Feststellung der Fonttable-Adresse. Dazu werden einem
'String die Mnemocodes der entsprechenden Maschinenbefehle (Hexa) als
'Mki\$-String übergeben.

' \$A000 = Bereichsadressen

' \$2009 = Maschinenbefehl (Datentransfer A0 => D0)

' \$4E75 = RTS

'Anschließend beherbergt der String ein Mini-Maschinenprogramm, das

'dann über den Sprungbefehl "C:" aufgerufen wird.

C:

Mit "C:" kann ein C-kompiliertes Maschinenprogramm angesprungen werden. Es wird dazu die Adresse der Routine und eine evtl. vorhandene Parameterliste übergeben. Sind keine Parameter vorhanden, kann eine leere Klammer (s.o.) übergeben werden. Sollen 32-Bit-Parameter übergeben werden, ist dem jeweiligen Parameter das Kürzel "L:" (wie auch unter "GEMDOS" beschrieben) voranzustellen. Ohne Kennung werden grundsätzlich Words übergeben. In der Variablen, die dem C:-Befehl vorangestellt wird, ist nach Rückkehr zum BASIC der Inhalt von D0 (Longword) zu finden.

CALL

Ein altbekannter Sprungbefehl, der Maschinenprogramme (assembliert) aufruft, ist "Call". Hier wird keine Variable vorangestellt, sondern nur die Adresse der Maschinenroutine übergeben, zu welcher verzweigt werden soll. Ebenso wie bei "C:" ist auch die Übergabe einer Parameterliste möglich (alle Werte in Longs). Dabei wird dem Maschinenprogramm auf dem Stack ein 16-Bit-Wert übergeben, der die Anzahl der übergebenen Parameter angibt und anschließend ein 32-Bit-Wert, der die Adresse der Routine beinhaltet.

z.B.:

```
Call Maschine(11,2,Varptr(X$))
```

```
Fontadr%=Lpeek(M.adr%+Txtsize%*4)
```

'Die ermittelte Adresse wird dieser Vdisys-Routine übergeben und damit
'der gewählte Fontsatz eingeschaltet.

```
Lpoke Intin,Fontadr%
```

```
Dpoke Contrl+2,0
```

```
Dpoke Contrl+6,2
```

```
Dpoke Contrl+10,102
```

```
Vdisys 5
```

```
Return
```

```
Procedure D.size(Flnm$)
```

```
Erase S_howsize%()
```

```
Dim S_howsize%(1)
```

```
If Flnm$=""
```

```
Fileselect "\*.***", "", Flnm$
```

```
S_howsize%(0)=1
```

```
Endif
```

```
If Flnm$<>""
```

```
Dta$=Space$(44)
```

```
Dat.add%=Varptr(Dta$)
```

```
Buff%=Gemdos(&H1A,L:Dat.add%)
```

```
Datnam$=""
```

```
A%=Gemdos(&H4E,L:Varptr(Flnm$),&H0)
```

```
Datsize%=Lpeek(Dat.add%+26)
```

```
For I=30 To 43
```

```
  If Peek(I+Dat.add%)>32
```

```
    Datnam$=Datnam$+Chr$(Peek(I+Dat.add%))
```

```
  Endif
```

```
Next I
```

```
For I=30 To 43
```

```
  Poke I+Dat.add%,0
```

```
Next I
```

```
Endif
```



```

If S_howsize%(0)=1
  B_oxtext$="Die Datei|"+Flnm$+"|hat eine Größe von|      "+Str$(Datsize
%)+" Byte!"
  Alert 1,B_oxtext$,1," OKAY ",B.alert%
Endif
Return
Procedure Schluss
  Alert 2,"PROGRAMM VERLASSEN ?",2," OKAY | NEIN ",B.alert%
  If B.alert%=1
    Edit
  Else
    Cont
  Endif
Return
Procedure H.elptext
  If Areas%=1
    F.txt$=" MOVE / COPY . Mit der linken Maustaste = Ecke links ob
en festlegen."
    B.txt$="Box      aufziehen.Curs.tast.      bzw.
Shift/Curs.tast.=Sonderfunktionen.>RETURN<"
  Endif
  If Areas%=0
    If D.rawmode%=>1 And D.rawmode%<10
      F.txt$="PATCH-
FUNKTIONEN. Zeichnen mit der linken Maustaste. Mit den Funktions- "
      B.txt$="Tasten Größe,Muster,Linen-Art,-Dicke,und Stil verändern.
      >RETURN<"
    Endif
    If D.rawmode%=10
      F.txt$="SELF-Funktion.linke Taste=zeichnen /rechte Taste =bewegen.
Es kann mit "
      B.txt$="Icons      und      Blöcken      gezeichnet      werden.
>RETURN<"
    Endif
    If D.rawmode%=>11 And D.rawmode%<20
      F.txt$="RUBBER-FUNKTIONEN.Linke Taste=Startpunkt setzen und Box
aufziehen.Beide"
      B.txt$="Tasten = CANCEL
      >RETURN<"
    Endif

```

```

    If D.rawmode%=20
        F.txt$="LINE-FUNKTION. Linke Taste=Freihand / rechte Taste=gerade
        Linie zeigen /"
        B.txt$="beide Tasten = gerade Linien zeichnen
    >RETURN<"
    Endif
    If D.rawmode%=21
        F.txt$=" FILL-FUNKTION . Es werden Flächen mit dem eingestellten
        Füllmuster aus-"
        B.txt$="gefüllt. Linke Taste = Auslöser.
    >RETURN<"
    Endif
    If D.rawmode%=22 Or D.rawmode%=23
        F.txt$="SPRAY-Funktionen.Plot-Spray sprüht schwarze Punkte.Patt-
        Spray sprüht das"
        B.txt$="eingestellte Muster.Tastenvariationen=verschiedene
        Punktstärken.>RETURN<"
    Endif
    If D.rawmode%=24
        F.txt$="MARKER-FUNKTION. Die Markergröße richtet sich nach der
        Patch-Größe. Aus-"
        B.txt$="wahl = beide Tasten über dem Markermenüfeld drücken.
    >RETURN<"
    Endif
    If D.rawmode%=25
        F.txt$="TEXT-FUNKTION.Texteinstellung=beide Tasten über dem Textme
        nüfeld drücken"
        B.txt$="linke Taste=String setzen/rechte Taste=String löschen.
    >RETURN<"
    Endif
    Endif
    @Formalert(0,1,F.txt$,B.txt$)
Return
T.m.text:
Data ART,HÖHE,WINK.,FARBE
Il_text:
Data "micon_1","micon_2","micon_3","micon_4","micon_5","micon_6","micon_
7","micon_8","micon_9","micon_10","micon_11","micon_12","micon_13","micon_
14"

```

Der Befehl "Sound" ermöglicht die Modifikation eines der drei Sound-Register. Dazu werden fünf Parameter, jeweils durch ein Komma getrennt, übergeben.

Parameter 1 = Soundregisternummer (1-3)
 Parameter 2 = Lautstärke (0-15)
 Parameter 3 = Oktave (1-8)
 Parameter 4 = Note (1-12)
 Parameter 5 = Haltedauer des Tones

Statt der Parameter 3 und 4 kann ein gemeinsamer Parameter übergeben werden, dem dann das Nummernzeichen "#" vorangestellt werden muß. Der Wert dieses Parameters errechnet sich aus dem Wert 125000, der durch die Frequenz eines Tones (z.B. A = 440 Hertz)+0.5 geteilt wird. Der Befehl für diesen Ton "A" sieht dann z.B. so aus: Sound 1,12,#284,50

WAVE

Mit "Wave" können die verschiedenen Einstellregister im Yamaha-Soundchip angesprochen werden. Es werden ebenfalls fünf Parameter übergeben.

Parameter 1 = Sound- und Rauschregister an/aus. Es existieren drei Register für die Tonerzeugung und drei für die Geräuscherzeugung. Der Wert des ersten Parameters errechnet sich aus der Bitstellung eines 6Bit-Wertes. Das erste Bit ist für das erste Soundregister, das zweite Bit für das zweite Soundregister, das dritte Bit für das dritte Soundregister, das vierte Bit für das erste Rauschregister, das fünfte Bit für das zweite Rauschregister und das sechste und letzte Bit ist für das dritte Rauschregister zuständig. Je nachdem, welches Bit gesetzt ist, wird das entsprechende Register ein- oder ausgeschaltet.

z.B. &X101011 oder 43

1. Soundregister an
2. Soundregister an
1. Rauschregister an
3. Rauschregister an

2.Parameter = Hüllkurvenzuordnung Nach dem gleichen Bit-Schema wie oben wird angegeben, welchem Register die aktuelle Hüllkurve zugeordnet werden soll.

3.Parameter = Bestimmung der Hüllkurvenform:

Linear fallend	= 1-3
Linear steigend	= 4-7
Sägezahn fallend	= 8
Linear fallend	= 9
Dreieck anfangs fallend	= 10
linear fallen /laut	= 11
Sägezahn steigend	= 12
Liniear steigend/haltend	=13
Dreieck anfangs steigend	= 14
Linear steigend	= 15

Parameter 4 = Dehnung der Hüllkurve

0 = keine Dehnung je größer der Wert, um so länger die Kurve.

Parameter 5 = Haltedauer bis zur Programmfortsetzung

Wer schon einmal versucht hat, sich mit dem Yamaha-Sound-chip, der sich im Atari ST befindet, näher auseinanderzusetzen, wird meine Verlegenheit, darüber etwas schreiben zu müssen, sicher verstehen. Selbst eingehende Kenntnisse dieser Materie garantieren nicht unbedingt den gewünschten Effekt. Deshalb finden Sie im folgenden ein kleines Programm, mit welchem Sie spielerisch versuchen sollten, dem Geheimnis auf die Spur zu kommen. Das Programm ist in seiner Grundstruktur möglichst einfach gehalten, sodaß Sie hoffentlich die Herausforderung annehmen, es nach eigenen Vorstellungen und Erfahrungen so zu verändern, daß hinterher auch "Musik" dabei herauskommt.

@Bild_1

Do

Repeat

Mouse X,Y,K

Wave 0,0

```

Sound 0,0
If K=0
  X1=0
  Y1=0
  X1=Int((X-43)/48)+1
  Y1=Int((Y-153)/17)+1
  If X1>0 And X1<19 And Y1>0 And Y1<13
    If X1>0 And X1<9
      Octave=X1
    Endif
    If Y1>0 And Y1<13
      Tone=13-Y1
    Endif
    Wave P.1,P.2,P.3,P.4,P.5
    Sound Kanal,Vol,Tone,Octave-1,Dauer
    Sound Kanal-1,Vol,Tone,Octave,Dauer
    Sound Kanal+1,Vol,Tone,Octave+1,Dauer
  Endif
Endif
Print At(10,7);"Sound ";Kanal'Vol'Octave'13-Tone'Dauer'''
Print At(10,8);"Wave ";P.1'P.2'P.3'P.4'P.5'''
Key=Asc(Right$(Inkey$))
If Key=59
  @S.chluss
Endif
Until Mousek Or (Key>59 And Key<66)
Mouse X,Y,K
If (Y>42 And Y<78) Or (Key>58 And Key<66)
  @Einstellung
Endif
Loop
Procedure Einstellung
  X2=Int((X-35)/40)+1
  If X2=1 Or Key=59
    @S.chluss
  Endif
  If (X2=2 And K=1) Or Key=60
    Inc P.1
  Endif
  If (X2=3 And K=1) Or Key=61

```

```
    Inc P.2
Endif
If (X2=4 And K=1) Or Key=62
    Inc P.3
Endif
If (X2=5 And K=1) Or Key=63
    Inc P.4
Endif
If (X2=6 And K=1) Or Key=64
    Inc P.5
Endif
If (X2=7 And K=1) Or Key=65
    Inc Kanal
    If Kanal>3
        Kanal=3
    Endif
Endif
If (X2=8 And K=1) Or Key=66
    Inc Dauer
Endif
If (X2=9 And K=1) Or Key=67
    Inc Vol
    If Vol>15
        Vol=15
    Endif
Endif
If (X2=2 And K=2) Or Key=60
    Dec P.1
    If P.1<0
        P.1=0
    Endif
Endif
If (X2=3 And K=2) Or Key=61
    Dec P.2
    If P.2<0
        P.2=0
    Endif
Endif
If (X2=4 And K=2) Or Key=62
    Dec P.3
```

```
If P.3<0
  P.3=0
Endif
Endif
If (X2=5 And K=2) Or Key=63
  Dec P.4
  If P.4<0
    P.4=0
  Endif
Endif
If (X2=6 And K=2) Or Key=64
  Dec P.5
  If P.5<0
    P.5=0
  Endif
Endif
If (X2=7 And K=2) Or Key=65
  Dec Kanal
  If Kanal<0
    Kanal=0
  Endif
Endif
If (X2=8 And K=2) Or Key=66
  Dec Dauer
  If Dauer<0
    Dauer=0
  Endif
Endif
If (X2=9 And K=2) Or Key=67
  Dec Vol
  If Vol<0
    Vol=0
  Endif
Endif
Wave P.1,P.2,P.3,P.4,P.5
Pause 2
Return
Procedure Bild_1
  Vol=15
  Deffill ,2,4
```



```
Pbox 10,10,619,379
Box 9,9,620,380
Deffill ,0,0
Pbox 18,18,435,371
Deffill ,2,8
Deftext ,16,2700,26
Text 510,100,"SOUND-CHECK"
Deftext 0,1,0,13
Restore F.tasten
For I=40 To 400 Step 40
  Pbox I,40,I+30,80
  Read F.txt$
  Graphmode 2
  Text I+6,57,F.txt$
  Graphmode 1
Next I
Graphmode 2
Deftext ,0,0,4
Text 44,70,"QUIT"
Text 84,70,"P.1"
Text 124,70,"P.2"
Text 164,70,"P.3"
Text 204,70,"P.4"
Text 244,70,"P.5"
Text 284,70,"CHAN"
Text 324,70,"HOLD"
Text 364,70,"VOL"
Text 404,70,"TEST"
Defline 1,4
Graphmode 1
Box 40,150,430,357
Defline 1,1
For J=91 To 379 Step 48
  For I=170 To 340 Step 17
    Line 43,I,427,I
  Next I
  Line J,153,J,357
Next J
Deffill 1,0
Defline 1,4
```

```
Pbox 163,145,307,157
Box 163,145,307,157
Pbox 22,150,45,357
Box 22,150,45,357
Deftext 1,0,0,4
Restore Octavetext
For I=1 To 8
    Read Oct.txt$,Txt.ko
    Text Txt.ko,166,Oct.txt$
Next I
Deftext 1,0,0,6
Text 190,154,"O C T A V E"
Restore Noten
For I=166 To 353 Step 17
    Read Not.txt$
    Text 27,I,Not.txt$
Next I
Return
Procedure S.chluss
    Pause 10
    Alert 2,"Programm-Ende ?",2,"OKAY|NEIN",Backalert
    If Backalert=1
        Print Chr$(7)
        Edit
    Endif
Return
Octavetext:
Data first,50,second,97,third,149,fourth,194,fifth,245,sixth,293,seventh
,335,eighth,385
Noten:
Data B,A,H,G#,G,F#,F,E,D#,D,C#,C
F.tasten:
Data F1,F2,F3,F4,F5,F6,F7,F8,F9,F10
```

MIN / MAX

Mit diesen beiden Funktionen können entweder der kleinste Wert oder String aus einer Liste von Ausdrücken oder Variablen, bzw. der größte Wert aus einer solchen Liste ermittelt werden.

Diese Liste wird der jeweiligen Funktion in Klammern übergeben.

```
z.B.:  B=3*11
        A=Min(1,5,8,2^3,B,11,7)
        Print A

        Ausgabe : 1 (der kleinste Wert)

        B$="ab"
        A$=Min("A","a",B$,B$+"X","B","c")
        Print A$
```

Ausgabe : A (der vom ASCII-Wert her kleinste String)

Würde in diesen beiden Beispielen für "Min" der Befehl "Max" eingesetzt werden, würde der jeweils größte Wert, bzw. String der Liste ausgegeben.

TRON / TROFF

Mit "Tron" kann der Programmablauf verfolgt werden. Wird nur der Befehl allein, ohne Zusatz verwendet, werden die Programmzeilen, die aktuell bearbeitet werden, auf dem Bildschirm ausgegeben. Das kann allerdings manchmal äußerst störend wirken. Um das zu umgehen, gibt es zwei Möglichkeiten.

1. Sie öffnen eine Diskettendatei (s."OPEN") und fügen dem Befehl anschließend die Kanalnummer der Datei an. Dadurch wird die Zeilenausgabe in die betreffende Datei geschrieben. Anschließend kann das Listing mit "Input\$" wieder eingelesen werden.
2. Sie öffnen die virtuelle "LST:"-Datei und fügen ebenfalls anschließend die gewählte Kanalnummer dem Befehl an. Dadurch erreichen Sie, daß die Programmzeilen auf dem Drucker ausgegeben werden.

z.B.:

```
Open "",#1,"LST:"  
Tron #1  
-- ab hier folgt der Programmteil,  
-- der überwacht werden soll.  
Troff
```

Sie können den "Tron"-Befehl, sowie auch den "Troff"-Befehl an jeder beliebigen Stelle im Programm einsetzen und somit auch kleine Programmteile oder einzelne Zeilen überwachen. Der "Troff"-Befehl schließt die Programmüberwachung ab. Im Anschluß an diesen Befehl ist ihr Programm wieder im Normalmodus. Dem "Troff"-Befehl braucht keine Kanalnummer übergeben zu werden. Bei Anwendung dieser Ausweichmöglichkeiten aus dem Direktmodus achten Sie bitte darauf, daß die Dateien auch wieder ordnungsgemäß geschlossen werden.

PULLDOWN-MENÜS UND WINDOWS

Wer sich bisher schon mal mit dem Thema "GEM-Programmierung" befaßt hat, wird annähernd ermessen können, was es heißt, BASIC-Befehle zu entwickeln, die die Erstellung von Pulldown-Menüs und Fenstern dermaßen vereinfachen, wie es mit diesem BASIC möglich ist. Als Grundlage zur Erläuterung dieser Befehle soll uns ein Programm dienen, das mit einfachen Mitteln ein Pulldown-Menü und die Verwendung von bis zu vier verschiedenen Fenster ermöglicht. Der Begriff "einfach" ist in diesem Fall relativ zu betrachten. Der Einsatz dieser Befehle birgt schon einige Klippen, die erst einmal, und das gilt speziell für Einsteiger, vorsichtig zu umgehen sind. Auch dieses Programm birgt noch einige Mängel, deren Beseitigung findigen Köpfen überlassen bleiben soll. Es war in der Kürze der Zeit nicht möglich, alle notwendigen Maßnahmen bzw. alle Möglichkeiten, die diese Befehle und Funktionen erfordern bzw. bieten, vollständig auszuloten. In groben Zügen sind jedoch die wichtigsten Verfahrensweisen implementiert. Wer sich ausgiebiger damit befassen möchte, dem sei das "Große GEM-Buch zum ATARI ST" (Data Becker) wärmstens ans Herz gelegt. Dieses Beispielprogramm hat nichts mit dem vorher beschriebenen

Grafikprogramm zu tun. Beide Programme sind also getrennt voneinander zu betrachten.

@Inits

In Inits werden alle notwendigen Vorbereitungen zum Einsatz der Menü und Windowtechnik getroffen.

On Menu Key Gosub K.ey

ON MENU KEY GOSUB

Mit diesem Befehl wird angegeben, wohin das Programm verzweigen soll, wenn eine Taste auf der Tastatur betätigt wurde.

On Menu Gosub M.enu

ON MENU GOSUB

Zu der Prozedur, die bei "On Menu Gosub" angegeben wird, verzweigt das Programm, wenn eine Menü-Ereignis (event;engl.:Ereignis) stattgefunden hat. Wurde also ein Menüpunkt angeklickt, steht in der betreffenden Prozedur, welche Maßnahmen in den einzelnen Fällen getroffen werden sollen.

On Menu Message Gosub M.essage

ON MENU MESSAGE GOSUB

Diesem Befehl wird ebenfalls eine Prozedur angegeben. Zu dieser verzweigt das Programm, wenn eines von vielen verschiedenen anderen Ereignissen eingetreten ist. Hauptsächlich wird dieser Befehl zur Analyse von Window-Events eingesetzt. Die Events, die hier erfaßt werden können, werden unter "MENU" erläutert.

Do

On Menu

ON MENU

Mit "On Menu" sagen Sie dem Programm, daß es "nachschaauen" soll, ob ein Ereignis eingetreten ist. Wird dieser Befehl nicht verwendet, wird auch in den Fällen, daß Sie mit "On Menu Gosub" bestimmte Prozeduren benannt haben, keine dieser Prozeduren aufgerufen. Daher sollte "On Menu" so häufig wie möglich im Programm stehen. Vorzugsweise ist dazu eine Hauptschleife zu bilden, die das Programm ständig durchläuft und so bei jedem Durchlauf einmal auf "On Menu" stößt. Nur dann kann entsprechend der eingetretenen Ereignisse reagiert werden.

Loop

Procedure Inits

```
Dim X%(4),Y%(4),B%(4),H%(4)
```

Da wir vier Fenster erstellen wollen, brauchen wir zuerst vier numerische Felder, die die Koordinaten der einzelnen Fenster aufzunehmen haben. Das sind zwei Felder für die X- bzw. Y-Koordinaten der linken oberen Ecken der Fenster. Diese Koordinaten sind entweder relativ zum Bildschirm-Nullpunkt oder, wenn Window 0 aktiviert wurde, relativ zur linken oberen Ecke von dessen Arbeitsfläche. Window 0 ist kein Fenster, das Sie in der Größe definieren können. Es bildet nur den Hintergrund zu einem evtl. vorhandenen Pulldownmenü. Würde das Window 0 nicht geöffnet und Sie benutzen keine weiteren Fenster würde die Menüleiste von Printbefehlen überschrieben werden. Öffnen Sie dagegen Window 0, wird der Bildschirm am oberen Rand einfach um 18 Pixels abgeschnitten. In diesem 18 Pixels breiten Streifen hat jetzt die Kopfzeile des Pulldownmenüs Platz und alle Ausgabebefehle nehmen nun Pixel 0/18 des gesamten Bildschirms als Ursprung an. Das zweite Koordinatenpaar besteht aus der Breite des jeweiligen Fensters (B%), sowie aus seiner Höhe (H%). Dabei ist jeweils die Gesamtfläche des Fensters gemeint. Eigentlich sind diese Arrays nicht notwendig, da dieselben Informationen die hier abgelegt werden, auch aus der "Windtab"-Tabelle (s.u.) entnommen werden können. Da diese

Tabelle aber nur über Peeks und Pokes angesprochen werden kann, ist es in Einzelfällen sinnvoller (und handlicher), sich solch ein Koordinatenfeld selbst anzulegen.

```
For I=1 To 4
```

Diese Felder werden willkürlich mit den Koordinaten und Maßen gefüllt, die für die erste Eröffnung ausschlaggebend sein sollen.

```
X%(I)=10+I*10
Y%(I)=30+I*5
B%(I)=440
H%(I)=300
Next I
Dim M.enuarray$(60)
```

Es wird ein Text-Array angelegt, das wenigstens soviele Elemente haben muß, wie anschließend an Menüpunkten definiert werden soll.

```
Restore Men_txt.1
```

Die einzelnen Menüpunkttexte werden am besten in Datazeilen abgelegt, aus denen Sie mit der folgenden Schleife ausgelesen werden.

```
For I=0 To 60
  Read M.enuarray$(I)
  Exit If M.enuarray$(I)="+++++"

```

Durch diese Schleife werden die einzelnen Data-Strings dem Menütextfeld zugeordnet

```
Next I
M.enuarray$(I)=""
M.enuarray$(I+1)=""
Men_txt.1:
Data PROGRAMM,Information
Data -----
```

```

Data      ,      ,      ,      ,      ,      ,      ,      ""
Data windows,OPENW 1,OPENW 2,OPENW 3,OPENW 4,""
Data options,Showareas,Umschaltstatus,Maustastenstatus,""
Data quit,Back to GfA-BASIC-Editor,""
Data +++++

```

Hierbei ist zu beachten, daß das erste Menü folgenden Aufbau hat: 1. String = Menütitel (evtl. Programmname) 2. String = Beliebige Überschrift (hier: "Information"). Unter diesem Menüpunkt können Sie eine beliebige Programm-Funktion einordnen. Da es aber der einzige für Sie verwendbare Menüpunkt in diesem Menü ist, eignet sich am besten die Ausgabe eines Info-Blockes zu Ihrem Programm. 3. String = Reihe von Minuszeichen (bzw. Bindestrichen). Die Anzahl der Striche bestimmt hier die Menübreite. Da in dieses Menü grundsätzlich die evtl. vorhandenen Desk-Accessories aufgeführt werden, sollten Sie hier die Länge der einzelnen Accessory-Titel berücksichtigen. 4. - 9. String = Sechs Leerstrings, die als Platzhalter für die Accessories dienen. Es dürfen keine Nullstrings übergeben werden! 10. String = Nullstring. Dieser Nullstring gilt als Abschluß für jede Menüreihe. An diesen ersten Menüaufbau werden nun die von Ihnen frei benennbaren weiteren Menüs angehängt. Diese sind so aufgebaut, daß sich an den Menütitel (der String, der immer in der Menüleiste sichtbar bleibt) die einzelnen Menüpunktbezeichnungen anschließen. Dabei muß wieder, wie beim ersten Menü, ein Nullstring ("") den Abschluß zu jedem einzelnen Menü bilden. Nachdem alle Menüeinträge gelesen wurden, müssen als Endmarkierung dem Ganzen noch zwei Nullstrings angehängt werden (""). Da nicht immer genau feststeht, wieviel einzelne Data-Strings gelesen werden müssen, empfiehlt sich der hier verwendete Aufbau, der die Lese-Schleife grundsätzlich verläßt, wenn ein, ebenfalls von Ihnen frei benennbarer Schlußstring gelesen wird, der dann jedoch nicht in das Menü eingefügt wird. Sie haben dann nur darauf zu achten, daß das String-Array genügend Elemente besitzt.

```

Dpoke Contrl+2,0
Dpoke Contrl+6,1
Dpoke Contrl+12,1
Dpoke Intin,1

```



```
Vdisys 106
Dpoke Contrl+2,0
Dpoke Contrl+6,1
Dpoke Contrl+12,1
Dpoke Intin,9
Vdisys 107
```

Mit diesen beiden Funktionen wird die Schriftart und -höhe des Textes eingestellt, der in der Menüleiste oder im Fensterrahmen ausgegeben wird.

```
Deftext ,1,0,13
```

Mit dieser "Deftext"-Einstellung können auch die "Print"-Ausgaben im Fenster modifiziert werden.

```
Deffill ,2,4
Pbox 0,19,639,399
Print At(25,1);"Menü- und Fensterdemo in GfA-BASIC"
Pause 100
Menu M.enuarray$( )
```

MENU (Array\$)

Mit diesem Befehl wird dem Interpreter gesagt, wo er im Speicher die einzelnen Menüeinträge findet. Sie übergeben ihm dazu in Klammern den Namen des Stringarrays, das die Menüttexte enthält.

```
@Init.fenster
Openw 0
```

OPENW

"Openw" öffnet das Window, dessen Nummer (handle) Sie übergeben. In diesem Fall wurde als Rahmen das Window 0 geöffnet. Den Grund dafür haben Sie weiter oben bereits erfahren.

Return

Procedure K.ey

Print "Scan- + ASCII-Code : ";Menu(14)

MENU

Dieses ist ein eindimensionales numerisches Array, in welches, je nach aufgetretenem Ereignis, bestimmte Informationen abgelegt werden. Jedes Element dieses Feldes kann unter Angabe seines Indexes ausgelesen werden. Allerdings steht in ein und demselben Punkt nicht immer eine spezielle Information, sondern es können, je nachdem welches Ereignis eingetreten ist, verschiedene Informationen wechselweise darin enthalten sein. Menu(5) enthält zum Beispiel, wenn eine Window-Beeinflussung stattgefunden hat, die X-Koordinate des betreffenden Fensters. Wenn aber einer der beiden Schieber (weiße Box rechts über bzw. unten links neben dem jeweiligen grauen Scrollbalken) bewegt wurde, kann man unter demselben Punkt die Position des Schiebers innerhalb der umfassenden Box (relativ zum Nullpunkt der jeweiligen Box = 0-1000) erfahren. Diese von Fall zu Fall variierenden Einträge finden allerdings nur in den Punkten 1-8 statt. Um sich über den jeweiligen Inhalt dieser Elemente größere Klarheit zu verschaffen, empfiehlt sich nochmals die Lektüre des bereits erwähnten GEM-Buches. Im Rahmen des vorliegenden Buches läßt sich eine erschöpfende Behandlung dieses Themas nicht verwirklichen.

Die Inhalte von "Menu(x)" sind:

- Menu(-1) = liefert die Adresse des aktuellen Menü-Objektbaumes
- Menu(0) = der Index des angeklickten Menüpunktes
- Menu(1) = Pufferindex (gibt an, was sich ereignet hat)
10 = (MN_SELECTED) Pulldown-Menü wurde ausgewählt.

Menu(4) = Titelfindex

Menu(5) = Menüpunktindex

20 = (WM_REDRAW) Windowbereich neu zeichnen. Koordinaten und Größe in Menu(5),Menu(6),Menu(7),Menu(8).

21 = (WM_TOPPED) ein Window soll aktiviert werden. In Menu(4) steht das Handle des angeklickten Fensters.

22 = (WM_CLOSED) Window soll geschlossen werden. Schließfeld links oben wurde gewählt (==> Closew).

23 = (WM_FULLED) Window soll maximal geöffnet werden. Vollfeld rechts oben wurde gewählt (==> Fullw).

24 = (WM_ARROWED) Window-Inhalt soll gescrollt werden. Es wurde einer der Pfeile oder einer der beiden grauen Scrollbalken angeklickt. In diesem Fall steht in Menu(5) der Index des gewählten Objektes.

0 = ganze Seite nach oben

1 = ganze Seite nach unten

2 = eine Zeile nach oben

3 = eine Zeile nach unten

4 = ganze Seite nach links

5 = ganze Seite nach rechts

6 = ein Zeichen nach links

7 = ein Zeichen nach rechts

25 = (WM_HSLID) Horizontaler Schieber wurde bewegt. In diesem Fall steht in Menu(5) die relative Position Position des Schiebers zu dem Nullpunkt der umgebenden Box (links außen). Dieses ist ein Wert zwischen 0 und 1000.

26 = (WM_VSLID) Vertikaler Schieber wurde bewegt. Es gilt das gleiche wie bei "HSLID". Der Nullpunkt der V-Box liegt oben.

27 = (WM_SIZED) Das Größenfeld rechts unten wurde gewählt. Es steht in Menu(5) die X-

Koordinate, in Menu(6) die Y-Koordinate, in Menu(7) die neue Breite und in Menu(8) die neue Höhe.

28 = (WM_MOVED) Der lange graue Balken über dem Fenster wurde angeklickt. Das Fenster soll bewegt werden. Die neue X-Koordinate steht wieder in Menu(5), die neue Y-Koordinate in Menu(6), die Breite (unverändert) in Menu(7) und die ebenfalls unveränderte Höhe in Menu(8).

- Menu(2) = nicht bekannt (1 oder 0 ??)
 - Menu(3) = nicht bekannt (0 ??)
 - Menu(4) = Handle
 - Menu(5) = X-Koordinate oder Schieberposition
 - Menu(6) = Y-Koordinate
 - Menu(7) = Breite
 - Menu(8) = Höhe
 - Menu(9) = Eventflag
 - Menu(10) = Maus-X-Koordinate zum Zeitpunkt des Ereignisses.
 - Menu(11) = Maus-Y-Koordinate zum Zeitpunkt des Ereignisses. Diese beiden Koordinaten sind "echt". D.h., sie beziehen sich nicht auf den Window-Nullpunkt, sondern auf den realen Bildschirm-Nullpunkt. Bei geöffneten Fenster werden von "Mousex", "Mousey" und "Mouse X,Y,K" immer die Koordinaten relativ zum jeweiligen Fenster-Nullpunkt ausgegeben.
 - Menu(12) = Maustasten-Status bei Event.
(1=linke;2=rechte;3=beide)
 - Menu(13) = "Switch"-Tastenstatus
-
- Shift-Taste rechts = 1
 - Shift-Taste links = 2
 - Control-Taste = 4
 - Alternate-Taste = 8

Es sind Kombinationen von allen vier Tasten möglich.

z.B. "Control" + "Shift"-links = 6
 "Alternate" + "Shift"-rechts + "Control" = 13
 usw.

Menu(14) = 16Bit-Wert, der im oberen Byte den Scan-Code und im unteren Byte den ASCII-Code der gerade gedrückten Taste liefert. (Umformung s. unten im Programm)
 Menu(15) = Anzahl der Mausklicks, die zum Ereignis führten.

```
Taste$=Mki$(Menu(14))
Print "Scan-Code : "Peek(Varptr(Taste$))
Print "ASCII-Code : "Peek(Varptr(Taste$)+1)
```

Umformung (Splitting) von Menu(14) in Scan- und ASCII-Code. Manchmal kann es von Nutzen sein, dieses Splitting nicht vorzunehmen, da die durch Menu(14) gelieferte 16Bit-Zahl zu jeder Zeit eine unzweifelhafte Identifizierung der gedrückten Taste ermöglicht. Es gibt keine Taste, die denselben 2-Byte-Wert hat wie eine andere.

Return

Procedure M.enu

```
If M.enuarray$(Menu(0))="Information"
  If Handle%>0
    @Fuller
    Clearw Handle%
    Cls
    Print "Sie können:"
    Print "  4 Fenster gleichzeitig auf den Bildschirm bringen."
    Print "  Die Fenster öffnen, schließen, bewegen. "
    Print "  Die Randbereiche (Pfeile, Scrollbalken etc.) bedienen."
    Print "  Sämtliche Tasten abfragen."
    Print "  Bis zu 5 Headlines in eine Menüleiste eintragen."
    Print "  Beliebige viele Menüpunkte aufführen und auswählen."
  Else
    Alert 1," Bitte zuerst ein Fenster|öffnen !",1,"OKAY",Abc
  Endif
```

```

Endif
If M.enuarray$(Menu(0))="Umschaltstatus" And Handle%>0
  Print "Status der Umschalttasten >Control/Shift/Alternate< : '"Menu
(13)
Endif
If M.enuarray$(Menu(0))="Maustastenstatus" And Handle%>0
  Print "Status der Maustasten (li=1,re=2,beide=3) : '"Menu(12)
Endif
If M.enuarray$(Menu(0))="Showareas" And Handle%>0
  @Get.workarea
  Print "Arbeitsfläche: '"Workx' 'Worky' 'Workb' 'Workh
  @Get.wholearea
  Print "Gesamtfläche: '"Wholx' 'Wholey' 'Wholeb' 'Wholeh
  Print "aktuelles Window-Handle : '"Handle%
Endif
If M.enuarray$(Menu(0))="Back to GfA-BASIC-Editor"
  Closew 0

```

CLOSEW

Durch "Closew" wird das Fenster mit dem angegebenen Handle geschlossen.

```

Closew 1
Closew 2
Closew 3
Closew 4
Cls
Menu Kill

```

MENU KILL

"Menu Kill" deaktiviert das Pulldown-Menü. Es kann keine Auswahl mehr vorgenommen werden, obwohl der Menütext stehen bleibt. In diesem Fall wurde das Hauptwindow 0 schon geschlossen und der Bildschirm mit "Cls" komplett bereinigt. Sie können danach wieder, wie gewohnt im normalen TOS-Modus arbeiten.

```
Edit
Endif
If M.enuarray$(Menu(0))="OPENW 1"

'Da die Menü-Indizes nicht ohne Unterbrechung fortlaufend sind, wird
'am besten der angeklickte Menü-Eintrag als String abgefragt.

Handle%=1
@Resize
Endif
If M.enuarray$(Menu(0))="OPENW 2"
Handle%=2
@Resize
Endif
If M.enuarray$(Menu(0))="OPENW 3"
Handle%=3
@Resize
Endif
If M.enuarray$(Menu(0))="OPENW 4"
Handle%=4
@Resize
Endif
Infow Handle%,"Sie haben "+M.enuarray$(Menu(0))+ " gewählt."
```

INFOW

Sie haben die Möglichkeit, dem laufenden Programm über eine Informationszeile, die direkt unter dem oberen grauen Balken liegt, Hinweistexte oder ähnliches beizusteuern. Diese Infozeile kann mit "Infow" definiert werden. Dazu wird dem Befehl das Handle des betreffenden Fensters übergeben und daran anschließend nach einem Komma der Text oder das String-Konstrukt, das Sie in dieser Zeile ausgeben möchten. Soll die Infozeile vollständig gelöscht werden, übergeben Sie an "Infow" einen Nullstring ("").

Menu Off

MENU OFF

"Menu Off" sollte immer eingesetzt werden, wenn ein Menü geöffnet und ein Menüpunkt gewählt wurde. Dadurch wird der bei der Anwahl schwarz unterlegte Menütitel wieder auf weiß geschaltet. Wurde kein Punkt gewählt und außerhalb des Menüs ins Window geklickt, ist diese Maßnahme nicht nötig. Eine weitere Funktion hat dieser Befehl nicht.

Return

Procedure M.message

```

  If Handle%>0
    Cls
    Print "Menu 0 :";Menu(0);" Menu 1 :";Menu(1);" Menu 2 :";Menu(2)
    Print "Menu 3 :";Menu(3);" Menu 4 :";Menu(4);" Menu 5 :";Menu(5)
    Print "Menu 6 :";Menu(6);" Menu 7 :";Menu(7);" Menu 8 :";Menu(8)
    Print "Menu 9 :";Menu(9);" Menu 10 :";Menu(10);" Menu 11 :";Menu(1
1)
    Print "Menu 12 :";Menu(12);" Menu 13 :";Menu(13);" Menu 14 :";Menu
(14)
    Print "Menu 15 :";Menu(15);" Menu -1 (Objektbaumadresse) :";Menu(-
1)
  Endif

```

'Die folgenden Abfragen beziehen sich auf das jeweils stattgefundenereignis. Die einzelnen Bedeutungen wurden oben bereits erklärt.

```

W.event=Menu(1)
If W.event=21
  @Topper
Endif
If W.event=23
  @Fuller
Endif
If W.event=24
  @Arrowscroll
Endif
If W.event=25
  @Slider

```



```

Endif
If W.event=26
  @Slider
Endif
If W.event=22
  @Closer
Endif
If W.event=27
  @Sizer
Endif
If W.event=28
  @Mover
Endif
If W.event=20 And Handle%>0
  Print "Screen ";Menu(4);" : ";Menu(5)''Menu(6)''Menu(7)''Menu(8)
Endif
Return
Procedure Closer
  Closew Handle%
Return
Procedure Topper
  Handle%=Menu(4)
  Openw Handle%
  Clearw Handle%

```

CLEARW

Das Fenster, dessen Handle hiermit übergeben wird, wird gelöscht. Der Befehl "Cls" kann ebenfalls verwendet werden. Allerdings wird dadurch ausschließlich das aktuelle Window gelöscht.

```

Return
Procedure Fuller

```

'In dieser Prozedur würde eigentlich alleine der Befehl "Fullw
'Handle%" ausreichen. Der mir vorliegende Interpreter war allerdings
'nicht in Stimmung, diesen Befehl korrekt auszuführen. "GfA-System-
'technik" wurde über diesen Mißstand bereits informiert.

```

X%(Handle%)=0
Y%(Handle%)=19
B%(Handle%)=639
H%(Handle%)=380
Closew Handle%

```

'Wenn nicht durch "Fullw", dann eben mit der Hand! Dem Koordinaten-
'Array werden dazu die, auf das aktuelle Fenster bezogenen Elemente
'die maximalen Window-Koordinaten einverleibt.

```

@Defsize(1,X%(Handle%),Y%(Handle%),B%(Handle%),H%(Handle%))
Openw Handle%
Clearw Handle%
Infow Handle%,"Volle Größe !"
Return
Procedure Arrowscroll
  Infow Handle%,"Pfeil:"+Str$(Menu(5))+" Fenster:"+Str$(Menu(4))
Return
Procedure Slider
  If W.event=25
    Infow Handle%,"Der H-Schieber wurde nach "+Str$(Menu(5))+" bewegt !"
  Else
    Infow Handle%,"Der V-Schieber wurde nach "+Str$(Menu(5))+" bewegt !"
  Endif
Return
Procedure Sizer
  Dpoke Windtab+8+(Handle%-1)*12,Menu(7)
  Dpoke Windtab+10+(Handle%-1)*12,Menu(8)

  'Falls das Größenfeld rechts unten gewählt wurde, werden den ent-
  'sprechenden Elementen der "Windtab"-Tabelle (dazu unten mehr) die
  'neuen Window-Maße übergeben...
  '
  B%(Handle%)=Menu(7)
  H%(Handle%)=Menu(8)

  '...und die Koordinaten-Liste aktualisiert.

  Closew Handle%
  Openw Handle%

```

Clearw Handle%

Infow Handle%,"Das Fenster wurde in der Größe verändert"

Return

Procedure Mover

Dpoke Windtab+4+(Handle%-1)*12,Menu(5)

Dpoke Windtab+6+(Handle%-1)*12,Menu(6)

B.breite=Menu(7)

If B.breite>639-Menu(5)

B.breite=639-Menu(5)

Endif

If B.breite<42

B.breite=42

Endif

Dpoke Windtab+8+(Handle%-1)*12,B.breite

Dpoke Windtab+10+(Handle%-1)*12,Menu(8)

X%(Handle%)=Menu(5)

Y%(Handle%)=Menu(6)

B%(Handle%)=Menu(7)

H%(Handle%)=Menu(8)

'Das Fenster wurde bewegt. Die Windtab-Tabelle und das eigene Info-
'Feld werden aktualisiert und das Fenster mit den neuen Koordinaten
'neu gezeichnet.

Closew Handle%

Openw Handle%

Clearw Handle%

Infow Handle%,"Das Fenster wurde bewegt"

Return

Procedure Init.fenster

'Damit wir es nicht immer mit anonymen Windtab-Indizes zu tun haben,
'geben wir den einzelnen Tabellenpunkten einen Namen, der uns jeweils
'Auskunft gibt, um welchen speziellen Eintrag es geht.

Handef=Windtab

Parts=Windtab+2

Windx=Windtab+4

Windy=Windtab+6

```
Windb=Windtab+8
Windh=Windtab+10
```

WINDTAB

Diese Tabelle ist keine BASIC-Funktion an sich. Sie hat eine ähnliche Funktion wie die reservierte Variable "Err". Genau wie in dieser werden hier Werte abgelegt (ausschließlich 16Bit-Werte), die uns über bestimmte Zustände Auskunft geben.

Sie kann jedoch noch etwas mehr. Sie können in vier Worte (2Byte) dieser Tabelle selber eintragen, welche Rand-Objekte jedes einzelnen Fensters gezeichnet werden sollen, also aktiv sind.

Dies geschieht durch:

```
Für Window 1 ==> Windtab+2
      Window 2 ==> Windtab+14
      Window 3 ==> Windtab+26
      Window 4 ==> Windtab+38
```

Diesen Windtab-Stellen wird ein 12-Bit-Wert übergeben. Durch einzelnes Setzen oder Addition der Bitwerte können die Komponenten bestimmt werden:

1	= der NAME (Fenstername)	==> BIT 1
2	= das CLOSE-Feld links oben	==> BIT 2
4	= das FULL-Feld rechts oben	==> BIT 3
8	= der MOVE-Balken oberer Randbereich	==> BIT 4
16	= die Infozeile unter dem MOVE-Balken	==> BIT 5
32	= das SIZE-Feld rechts unten	==> BIT 6
64	= der Aufwärts-Pfeil rechts	==> BIT 7
128	= der Abwärtspfeil rechts	==> BIT 8
256	= der VSLIDE-Balken rechts (vertikalschieber	==> BIT 9
512	= der Links-Pfeil unten	==> BIT 10
1024	= der Rechts-Pfeil unten	==> BIT 11
2048	= der HSLIZE-balken unten (Horizontalschieber)	==> BIT 12

Um nicht jedesmal die einzelnen Werte ausrechnen zu müssen, kann man die Komponenten auch so bestimmen:

Dpoke Windtab+2,&X111111111111 ==> es werden alle Komponenten von Window 1 aktiviert

Dpoke Windtab+2,&X001001001001 ==> es wird nur NAME,MOVER,PFEIL hoch und PFEIL links von Window 1 gesetzt.

usw.

Wenn Infow oder Titlew gesetzt ist, läßt sich das durch Weglassen der Bits in der Windtab-Tabelle nicht rückgängig machen. Um die Namenszeile und Infozeile vollständig zu löschen, müssen Titlew mit " " und Infow mit "" angegeben werden. Anschließend wirkt sich der Eintrag in "Windtab" auch darauf aus. Wenn Titlew mit "" angegeben wird, läßt sich auch der graue MOVE-Balken nicht mehr aktivieren.

Der Aufbau der Windtab-Tabelle sieht folgendermaßen aus:

Windtab	=	Window-Handle 1
Windtab+2	=	Attribute für Window 1 (s.o.)
Windtab+4	=	X-Koordinate für Window 1
Windtab+6	=	Y-Koordinate für Window 1
Windtab+8	=	Breite Window 1
Windtab+10	=	Höhe Window 1
Windtab+12		
bis		
Windtab+22	=	die entsprechenden Angaben für Window 2
Windtab+24		
bis		
Windtab+34	=	die entsprechenden Angaben für Window 3
Windtab+36		
bis		
Windtab+46	=	die entsprechenden Angaben für Window 4

Die Windtab-Stellen 48 und 50 können ignoriert werden.

```

Windtab+52 = Bildschirm-X-Koordinate
Windtab+54 = Bildschirm-Y-Koordinate
Windtab+56 = Bildschirmbreite
Windtab+58 = Bildschirmhöhe
Windtab+60 = X-Schnittpunkt der vier Fenster
Windtab+62 = Y-Schnittpunkt der vier Fenster
Windtab+64 = X-Nullpunkt für Grafikbefehle
Windtab+68 = Y-Nullpunkt für Grafikbefehle

```

Return

Procedure Defsize(P.aktiv,Xlo,Ylo,Xbr,Yho)

'Hier werden die einzelnen Windtab-Elemente aktualisiert.

P.aktiv=&X111111111111

Dpoke Parts+(Handle%-1)*12,P.aktiv

Dpoke Windx+(Handle%-1)*12,Xlo

Dpoke Windy+(Handle%-1)*12,Ylo

Dpoke Windb+(Handle%-1)*12,Xbr

Dpoke Windh+(Handle%-1)*12,Yho

Return

Procedure Get.koordie

Screenx=Dpeek(Windtab+52)

Screeny=Dpeek(Windtab+54)

Screenb=Dpeek(Windtab+56)

Screenh=Dpeek(Windtab+58)

Crossx=Dpeek(Windtab+60)

Crossy=Dpeek(Windtab+62)

Cerox=Dpeek(Windtab+64)

Ceroy=Dpeek(Windtab+66)

Return

Procedure Defwindow

Openw Handle%

Clearw Handle%

Titlew Handle%," D E M O Window "+Str\$(Handle%)

TITLEW

Sie können dem jeweiligen Window eine Überschrift geben. Der Befehlsaufbau ist identisch mit dem unter "INFOW" beschriebenen. Soll keine Überschrift gesetzt werden, übergeben Sie einen Leerstring. Ein Nullstring ("") bewirkt, daß das Fenster nicht mehr durch Anklicken des grauen MOVE-Balkens bewegt werden kann.

```
Infow Handle%,"Dieses ist die INFO-Zeile für Window "+Str$(Handle%)
```

```
Pause 50
```

```
Return
```

```
Procedure Resize
```

```
  @Defsize(1,X%(Handle%),Y%(Handle%),B%(Handle%),H%(Handle%))
```

```
  @Defwindow
```

```
Return
```

```
Procedure Get.workarea
```

'Mit dieser reinen GEM-Funktion kann die Größe und Position der
'verfügbaren Arbeitsfläche des jeweiligen Fensters ermittelt werden.

```
Dpoke Gintin,Handle%
```

```
Dpoke Gintin+2,4
```

```
Gemsys 104
```

```
Workx=Dpeek(Gintout+2)
```

```
Worky=Dpeek(Gintout+4)
```

```
Workb=Dpeek(Gintout+6)
```

```
Workh=Dpeek(Gintout+8)
```

```
Return
```

```
Procedure Get.wholearea
```

'Hier wird die Größe und Position der gesamten Fensterfläche (ein-
'schließl. der Randelemente) ermittelt.

```
Dpoke Gintin,Handle%
```

```
Dpoke Gintin+2,6
```

```
Gemsys 104
```

```
Wholex=Dpeek(Gintout+2)
```

```
Wholey=Dpeek(Gintout+4)
```

```
Wholeb=Dpeek(Gintout+6)
```

```
Wholeh=Dpeek(Gintout+8)  
Return
```

ON MENU IBOX GOSUB / ON MENU OBOX GOSUB

Mit diesen beiden Befehlen können vier bestimmte Bildschirmbereiche definiert werden, die von GEM ständig daraufhin überwacht werden, ob der Mauszeiger sie betritt bzw. verläßt. "On Menu Ibox" kann gleichzeitig zwei Bereiche verwalten, die dann über GEM melden, wenn der Mauszeiger sich in ("Ibox") ihnen befinden. Dasselbe gilt für zwei Bereiche, die melden, wenn sie vom Mauszeiger verlassen werden (out - "Obox").

Außerdem wird diesen Befehlen gleichzeitig der Name einer Prozedur übergeben zu welcher das Programm verzweigen soll, wenn diese genannten Ereignisse eingetreten sind.

Syntax:

On Menu Ibox Index,X-Koordinate,Y-Koordinate,Breite,Höhe Gosub Prozedur-
name

Für "On Menu Obox Gosub" gilt dieselbe Syntax.

Die Parameter bedeuten:

Index	=	Nummer des spezifizierten Bereichs
X-Koordinate	=	X-Koordinate des Bereichs
Y-Koordinate	=	Y-Koordinate des Bereichs
Breite	=	Breite des Bereichs
Höhe	=	Höhe des Bereichs
Prozedurname	=	Name der Prozedur, zu welcher bei Eintritt des Ereignisses verzweigt werden soll.

Anhang A: GEM,GEMDOS,BIOS,XBIOS

GEM - (AES) - FUNKTIONEN

GEMSYS

Das GEM (Graphic Environment Manager) besteht aus zwei Teilen, die beide verschiedene Aufgaben zu erfüllen haben. Dem AES (Application Environment System) und dem VDI (Virtual Device Interface).

Das AES hat die Aufgabe, die grafikorientierte Arbeit mit dem Gerät (Windows, Menüs, Formulare, Alertboxen) zu unterstützen, während das VDI für das Bereitstellen komfortabler Grafik- bzw. Ein- und Ausgabebefehle zuständig ist. Diese beiden GEM-Bereiche werden über zwei verschiedene Aufruf-Befehle angesprochen. Der eine davon ist "Gemsys". Mit diesem Befehl werden die AES-Funktionen aufgerufen.

Um diesen Befehl einsetzen zu können, müssen Sie noch einiges über die Organisation der verschiedenen Arrays erfahren, die vom GEM speziell zur Aufnahme aller notwendigen Parameter eingerichtet werden.

Im AES gibt es sechs solcher Felder:

GCONTRL = Aufnahme und Ausgabe der Grundinformationen, die das GEM zur Bearbeitung eines AES-Aufrufs benötigt. Dieses sind die Funktionsnummer sowie Angaben über die Größe der anderen Felder.

Gcontrl . = Funktionsnummer

Gcontrl+2 = Größe des Gintin-Feldes (Word)

Gcontrl+4 = Größe des Gintout-Feldes (Word)

Gcontrl+6 = Größe des AddrIn-Feldes (Longword)

Gcontrl+8 = Größe des Addrout-Feldes (Longword)

Global = nicht interessant (nur bei Multitasking)

GINTIN = In diesem Feld werden der Funktion entsprechende Informationen an das System weitergegeben (z.B. das Fensterhandle, auf das sich die Operation auswirken soll). Hier handelt es sich um Integerwerte (Words) die von Ihnen je nach gewünschter Funktion an das AES übergeben werden. Die Anzahl und jeweilige Bedeutung der Elemente dieses Feldes ist von der aufgerufenen Funktion abhängig.

GINTOUT = In diesem Feld werden bestimmte Informationen von der Funktion zurückgegeben. Es handelt sich ebenfalls um Integer-Words. Die Bedeutung der Elemente ist ebenfalls von der Funktion abhängig.

ADDRIN = Falls von der Funktion Adressen benötigt werden, auf die die Funktion zugreifen soll (z.B. Objektbaumadressen), müssen diese als Integer-Longword (4Byte) in Addrin übergeben werden.

ADDROUT = Hier werden Adressen, die vom AES funktions-spezifisch ermittelt werden, an den Programmierer, bzw. das Programm zurückgegeben.

Die Anzahl der benötigten Parameter ist auch bei diesen beiden Feldern von der Funktion abhängig.

Die "Gcontrl"-Elemente müssen nicht in jedem Fall bedient werden. Wollen Sie eine neue Funktion aufrufen, müssen Sie dem Befehl "Gemsys" nachstehend die Funktionsnummer übergeben. Soll nur der Inhalt der anderen Felder verändert, aber die zuletzt verwendete Funktion beibehalten werden, dann übergeben Sie nur die veränderten Parameter an die übrigen Felder und rufen ohne Funktionsnummernangabe die alte Funktion nur mit "Gemsys" auf.

Zusätzlich kann noch eine konstante Variable darüber abgefragt werden, an welcher Stelle im Speicher das erste Element des ersten AES-Feldes (Gcontrl) steht:

GB

Die Verwendung dieser Adresse ist in GfA-BASIC fast unnötig, da die oben beschriebenen Feldnamen schon die jeweiligen AES-Adressen beinhalten. Falls es notwendig werden sollte, können die Anfangsadressen der einzelnen Felder auch so bestimmt werden:

```
Procedure G.inits
  G.contrl% = Lpeek(Gb)
  G.intin%  = Lpeek(Gb+8)
  G.intout% = Lpeek(Gb+12)
  G.addrin% = Lpeek(Gb+16)
  G.addrout% = Lpeek(Gb+20)
Return
```

Die hier definierten Variablen beinhalten nun exakt dieselben Werte, wie Sie sie in den vorher beschriebenen BASIC-Feldbezeichnungen finden. In diesem Fall wäre es gleichgültig, ob Sie "Dpoke Gcontrl,104" oder "Dpoke G.contrl%, 104" verwenden. Es wird in beiden Fällen dieselbe Adresse angesprochen.

Es folgen nun einige AES-Funktionen. Es wurden hier nicht alle aufgeführt, da es erstens in der Kürze der verbleibenden Zeit nicht möglich war, diese vollständig an die GfA-Struktur anzupassen. Einige andere Funktionen wurden auch schon in den vorangegangenen Teilen ansatzweise erläutert.

Procedure Gethandle

'Übergibt in "Handle" das Handle des aktiven Fensters

Gemsys 77

Handle=Dpeek(Gintout)

Return

Procedure Windfind(Xwindow,Ywindow)

'Übergibt in "Windowhandle" das Handle des Fensters, das sich unter
'dem Mauszeiger befindet. Dabei sind "echte" Mauskoordinaten zu ver
'wenden, die sich auch auf den Bildschirm-Nullpunkt beziehen.

'Sollten Sie mal nicht wissen, woher Sie diese beziehen können, können

'Sie auch die Systemadressen 9952 und 9954 abfragen. In diesen werden

```
'vom System ständig die X-Koordinate (X=Dpeek(9952) und die Y-Ko-
'ordinate (Y=Dpeek(9954) abgelegt. Den Status der Maustasten erhalten
'Sie aus dem Inhalt der Adresse 3581. Von dem hierin enthaltenen Wert
'müßßen Sie dann nur noch den Wert 248 subtrahieren (K=Dpeek(3581)-
'248).
```

```
Dpoke Gintin,Xwindow
```

```
Dpoke Gintin+2,Ywindow
```

```
Gemsys 106
```

```
Windowhandle=Dpeek(Gintout)
```

```
Return
```

```
Procedure Movebox(Mov.breit,Mov.hoch,Mov.xstart,Mov.ystart,Mov.xziel,Mov
.yziel)
```

```
'Es kann eine Box definiert werden, die von einer Startposition aus-
'gehend, sich zu einer Zielposition hinbewegt (huscht). Die Variablen-
'namen sprechen für sich.
```

```
Dpoke Gintin,Mov.breit
```

```
Dpoke Gintin+2,Mov.hoch
```

```
Dpoke Gintin+4,Mov.xstart
```

```
Dpoke Gintin+6,Mov.ystart
```

```
Dpoke Gintin+8,Mov.xziel
```

```
Dpoke Gintin+10,Mov.yziel
```

```
Gemsys 72
```

```
Return
```

```
Procedure D.ragbox(D.br,D.ho,D.x,D.y,B.x,B.y,B.br,B.ho)
```

```
'Hiermit kann ein Rechteck definiert werden, daß sich nur innerhalb
'eines anderen definierten Rechtecks bewegen kann. Das erste Rechteck
'folgt den Bewegungen, die Sie mit dem Mauszeiger ausführen. Verläßt
'der Mauszeiger den Bereich des umgrenzenden Rechtecks, bleibt die
'kleine Box an dessen Rand stehen. Diese Funktion kann nur bei ge-
'drücktem Mausknopf aufgerufen werden, da sie bei gelöstem Mausknopf
'wieder beendet wird. Die "D."-Parameter geben die Breite, Höhe, X-
'und Y-Koordinate des kleinen Rechtecks an, während die "B."-Parameter
'die X-, Y-Koordinaten, sowie die Breite und Höhe des umfassenden
'Rechtecks beschreiben.
'Zum Abschluß der Funktion wird in "X.fin" und "Y.fin" die Position
'des verschiebbaren Rechtecks zum Zeitpunkt des Funktionsendes zurück-
'gegeben.
```

```
Dpoke Gintin,D.br
Dpoke Gintin+2,D.ho
Dpoke Gintin+4,D.x
Dpoke Gintin+6,D.y
Dpoke Gintin+8,B.x
Dpoke Gintin+10,B.y
Dpoke Gintin+12,B.br
Dpoke Gintin+14,B.ho
Gemsys 71
X.fin=Dpeek(Gintout+2)
Y.fin=Dpeek(Gintout+4)
Return
```

Procedure Mausstate

'Diese Funktion liefert die "echten" Mauskoordinaten und den Status
'der Maustasten, sowie den Status der Umschalttasten (s. "MENU").

```
Gemsys 79
Mausx=Dpeek(Gintout+2)
Mausy=Dpeek(Gintout+4)
Mausk=Dpeek(Gintout+6)
Flipkey=Dpeek(Gintout+8)
```

Return

Procedure Mouseevent(Outin,Ev.boxxl,Ev.boxyl,Ev.boxbreit,Ev.boxhoch)

'Es kann eine einzelne Box definiert werden, die bei Funktionsaufruf
'daraufhin überwacht wird, ob der Mauszeiger sie betritt (Outin=0)
'oder verläßt (Outin=1). Die Funktion wird erst dann abgebrochen, wenn
'das Ereignis eingetreten ist. Solange wartet AES mit der Programm-
'fortführung.

'Es werden anschließend die Mauskoordinaten der Maustasten- sowie der
'Umschalttastenstatus zurückgegeben.

```
Dpoke Gintin,Outin
Dpoke Gintin+2,Ev.boxxl
Dpoke Gintin+4,Ev.boxyl
Dpoke Gintin+6,Ev.boxbreit
Dpoke Gintin+8,Ev.boxhoch
Gemsys 22
Mausx=Dpeek(Gintout+2)
Mausy=Dpeek(Gintout+4)
Mausk=Dpeek(Gintout+6)
```

```
Switchk=Dpeek(Gintout+8)
```

```
Return
```

Procedure Keyevent

'AES wartet nach Aufruf dieser Funktion solange, bis eine Taste gedrückt wird. Anschließend wird in "Keycode" der Code der Taste abgelegt (16-Bit-Wert ist wie "On Menu Key" zu behandeln).

```
Gemsys 20
```

```
Keycode=Dpeek(Gintout)
```

```
Return
```

Procedure Buttonevent(Clicks,Ev.button,Ev.bstat)

'Bei Aufruf wartet AES bis eine Maustaste betätigt wurde. Dabei kann mit "Clicks" die Anzahl der Mausklicks festgelegt werden, bei welcher die Funktion abgebrochen wird. Geben Sie also hier eine 4 an, müßten Sie sehr schnell viermal hintereinander klicken, ehe das Programm weitermacht. Ob Sie das schaffen, ist sehr fraglich. Also Vorsicht! Mit "Ev.button" übergeben Sie den Index der Maustaste, die zur Auslösung des Ereignisses führen soll. Sie können die Funktion darauf beschränken, daß nur dann reagiert wird, wenn eine bestimmte Maustaste gedrückt wird (linke Taste=1; rechte Taste=2; beide Tasten=3; keine Taste=0). In "Ev.bstat" kann angegeben werden, welchen Zustand die Taste einnehmen soll, um das Ereignis zu initialisieren (0=nicht gedrückt; 1=gedrückt). Anschließend wird in "Clicks" (diesmal global) die tatsächliche Anzahl der Klicks zurückgegeben. Außerdem wieder die Mauskoordinaten zum Zeitpunkt des Ereignisses, der Maustastenstatus und der Wechseltastenstatus.

```
Dpoke Gintin,Clicks
```

```
Dpoke Gintin+2,Ev.button
```

```
Dpoke Gintin+4,Ev.bstat
```

```
Gemsys 21
```

```
Clicks=Dpeek(Gintout)
```

```
Mausx=Dpeek(Gintout+2)
```

```
Mausy=Dpeek(Gintout+4)
```

```
Mausk=Dpeek(Gintout+6)
```

```
Switchk=Dpeek(Gintout+8)
```

```
Return
```

*GEM - (VDI) - FUNKTIONEN**VDISYS*

Das "Virtual Device Interface" gibt Ihnen die Möglichkeit, auf verschiedene komfortable Funktionen zur Grafik- oder Textausgabe bzw. -einstellung zuzugreifen. Es existieren ebenso wie unter "Gemsys" verschiedene Arrays, die mit den entsprechenden Werten gefüllt werden müssen und andere, die funktionspezifische Rückmeldungen enthalten. Anders als bei den AES-Aufrufen müssen hier grundsätzlich die Contrl-Elemente mit übergeben werden. Wollen Sie allerdings nur den Inhalt der übrigen Felder verändern und die gleiche Funktion erneut aufrufen, können Sie die Definition des Contrl-Feldes beim zweiten Aufruf vernachlässigen und rufen die zuletzt aktive Funktion wieder mit "Vdisys" ohne Funktionsnummernangabe auf. Wird die Funktionsnummer geändert, muß jedesmal der Contrl-Block neu definiert werden. Im "VDI" gibt es ebenfalls fünf Parameterfelder, die nur zum Teil die gleiche Bedeutung haben, wie unter "GEMSYS" beschrieben.

CONTRL

Hierin werden wieder Grundinformationen übergeben, aus denen VDI ersehen kann, welche Funktion Sie auf welche Art ausführen lassen möchten.

- Contrl = Funktionsnummer (auch Opcode genannt)
- Contrl+2 = Anzahl der Elemente im "Ptsin"-Array, die für die jeweilige Funktion relevant sein sollen.
- Contrl+4 = Enthält nach Abschluß der Funktion die Anzahl der Ptsout-Elemente (Var=Dpeek(Contrl+4)).
- Contrl+6 = Anzahl der "Intin"-Elemente
- Contrl+8 = Enthält nach Abschluß der Funktion die Anzahl der Intout-Elemente (Var=Dpeek(Contrl+8)).
- Contrl+10 = Spezifische Identifikationsnummer
- Contrl+12 = Handle des anzusprechenden Windows

INTIN

In dieses Feld, dessen Länge von Fall zu Fall sehr variieren kann, werden funktionspezifische Parameter übergeben (Integer-Word), die der Funktion angeben, welche Art der Ausführung Sie wünschen bzw. welche Werte zur Verarbeitung übernommen werden sollen.

PTSIN

Hier werden von Ihnen ggf. Koordinaten eingetragen, an denen die gewünschte Aktion stattfinden soll.

INTOUT

Hier werden von der Funktion bestimmte Werte zurückgegeben. Die Bedeutung der Werte ist von der jeweiligen Funktion abhängig.

PTSOUT

= Von der Funktion werden Koordinaten oder Werte ausgegeben, die nach Abschluß der jeweiligen Funktion Aufschluß darüber geben, welche Einstellungen aktuell enthalten sind. Dieses Array wird nicht grundsätzlich von VDI bedient. Wann welche Werte ermittelt werden können, ist wieder von der jeweiligen Funktion abhängig.

Einige VDI- und AES-Funktionen haben Sie im Verlauf des Buches bisher schon kennengelernt. Wie Sie z.B. im Menu/Window-Demoprogramm sehen konnten, kann die Grafik-Ausgabe, sowie das jeweilige Schriftbild beeinflusst werden. Rufen Sie die folgenden Funktionen z.B. mit Handle=1 auf, können Sie sogar die Darstellung des Desktops, der Fileselectbox oder der Alertbox verändern. Achten Sie dann aber bitte darauf, daß die jeweiligen Veränderungen auch noch eine Benutzung

der Objekte zulassen. Was damit im einzelnen gemeint ist, werden Sie erfahren, wenn Sie mit diesen Funktionen etwas herumexperimentieren. Es kann nicht vermieden werden, hier nochmals auf das "Große GEM-Buch zum ATARI ST" zu verweisen, aus dem fast sämtliche GEM-Funktionen ersichtlich sind. Der "Contrl"-Block in diesen Prozeduren ist bereits vollständig eingestellt. Sie brauchen daran also keine Veränderungen vorzunehmen. Die zu verändernden Werte werden jeweils bei Prozeduraufruf als Parameter übergeben.

Procedure Textrotor(Handle,Angle)

 'Es kann die Textrichtung eingestellt werden (s. auch "DEFTTEXT"/Param.3).

 Dpoke Contrl+2,0
 Dpoke Contrl+6,1
 Dpoke Contrl+12,Handle
 Dpoke Intin,Angle
 Vdisys 13

Return

,

Procedure Textgraph(Handle,Grafikmode)

 'Einstellung des Grafikmodus (s. auch "GRAPHMODE").

 Dpoke Contrl+2,0
 Dpoke Contrl+6,1
 Dpoke Contrl+12,Handle
 Dpoke Intin,Grafikmode
 Vdisys 32

Return

,

Procedure Textart(Handle,Textart)

 'Veränderung der Textart (s. auch "DEFTTEXT"/Param.2).

 Dpoke Contrl+2,0
 Dpoke Contrl+6,1
 Dpoke Contrl+12,Handle
 Dpoke Intin,Textart
 Vdisys 106

Return

,

Procedure Testpix(Handle,Xpixel,Ypixel)

'Austesten von Bildschirmpunkten (s. auch "POINT").

'In "Pxset" wird zurückgegeben, ob der Punkt gesetzt ist oder nicht.

'In "Pxcoll" steht anschließend der Farbwert des Punktes.

Dpoke Contrl+2,2

Dpoke Contrl+6,0

Dpoke Contrl+12,Handle

Dpoke Ptsin,Xpixel

Dpoke Ptsin+2,Ypixel

Vdisys 105

Pxset=Dpeek(Intout)

Pxcoll=Dpeek(Intout+2)

Return

,

Procedure F.interior(Handle,F.stil,F.muster)

'Einstellung des Füllmusters (s. auch "DEFFILL").

Dpoke Contrl+2,0

Dpoke Contrl+6,1

Dpoke Contrl+12,Handle

Dpoke Intin,F.muster

Vdisys 24

Dpoke Contrl+2,0

Dpoke Contrl+6,0

Dpoke Intin,F.stil

Vdisys 23

Return

,

Procedure Selfline(Handle,L.typ,L.def)

'Einstellung der Linienart (s. auch "DEFLINE"/Param.1).

'Mit "L.typ" = 1-6 wird der vorgegebene Stil bestimmt.

'Mit "L.typ" = 7 und Angabe von "L.def" wird der selbstdefinierte

'Linienstil (1-32767) übernommen.

Dpoke Contrl+2,0

Dpoke Contrl+6,1

Dpoke Contrl+12,Handle

Dpoke Intin,L.typ

Vdisys 15

Dpoke Contrl+12,Handle

Dpoke Intin,L.def

Vdisys 113

Return

,

Procedure Diabar(Handle,Barxl,Baryl,Barxr,Baryr)

'Zeichnet ausgefüllte Box an angegebener Position (s. auch "PBOX").

Dpoke Contrl,11

Dpoke Contrl+2,4

Dpoke Contrl+6,0

Dpoke Contrl+10,1

Dpoke Contrl+12,Handle

Dpoke Ptsin,Barxl

Dpoke Ptsin+2,Baryl

Dpoke Ptsin+4,Barxr

Dpoke Ptsin+6,Baryr

Vdisys 11

Return

,

Procedure Charboxh(Handle,Boxhoehe)

'Einstellung der Texthöhe (s. auch "DEFTXT"/Param.4).

Dpoke Contrl+2,0

Dpoke Contrl+6,1

Dpoke Contrl+12,Handle

Dpoke Intin,Boxhoehe

Vdisys 107

Return

,

Procedure E.col(Reg,Rot,Gruen,Blau)

'Farbeinstellung (RGB) (s. auch "SETCOLOR").

Dpoke Contrl+2,0

Dpoke Contrl+6,4

Dpoke Contrl+12,2

Dpoke Intin,Reg

Dpoke Intin+2,Rot

Dpoke Intin+4,Gruen

Dpoke Intin+6,Blau

Vdisys 14

Return

GEMDOS-FUNKTIONEN

GEMDOS

Ein weiterer Befehl bzw. eine Adreßvariable, der (neben BIOS und XBIOS) die Möglichkeit bietet, nach Herzenslust im System herumzuspazieren. Mit dem Angebot dieser Möglichkeiten steht GfA-BASIC tatsächlich allein auf weiter Flur. Was sich mit diesen drei Befehlen alles anstellen läßt, übersteigt die spontane Phantasie. Der Atari ST besitzt drei verschiedene Ebenen im Betriebssystem, die jede für sich spezielle Aufgaben hat. Über entsprechende Opcodes lassen sich so z.B. Disketten formatieren, eigene Fonttabellen laden, Sektoren von Diskette lesen/schreiben usw...

Eine dieser Ebenen ist das GEMDOS (Graphic Environment Manager Disk Operating System). Mit der Übergabe einer Funktionsnummer an den Befehl "Gemdos" können nun verschiedene Funktionen ausgelöst werden. Zur Aufnahme eines evtl. Rückgabewertes bzw. der Fehlernummer wird eine beliebige numerische Variable bestimmt. Dieser Variablen wird das Ergebnis der jeweiligen Funktion "zugewiesen". Es handelt sich hier um jene Werte die im Normalfall im Datenregister D0 eingetragen werden. Die Syntax:

Var=Gemdos(Opcode,Liste von Parametern)

Die Anzahl der zu übergebenden Parameter ist jeweils abhängig von der aufgerufenen Systemroutine. Um den Aufbau der Funktionen zu erfahren, sei Ihnen hier ein weiteres Buch empfohlen, das genauen Aufschluß über die einzelnen Routinen gibt. Es handelt sich um das DATA-BECKER-Buch "ATARI ST Intern" (Englisch, Gerits, Brückmann), das sich ausschließlich mit der Anwendung solcher und ähnlicher Funktionen beschäftigt. In diesem Buch wird ausführlich zu der jeweiligen Funktion erläutert, in welcher Art und Zusammensetzung die Parameter übergeben werden müssen.

Werden von der Funktion Longwords erwartet (z.B. bei Adres-
senübergaben), kann dem entsprechenden Wert ein "L:" vorange-
stellt werden, woraus die Funktion erkennt, das ein 32-Bit-Wert
übergeben wird. Wird den Werten kein L: vorangestellt, werden
die übergebenen Werte als 16Bit-Words betrachtet.

Procedure Setdrive(Desdrv)

'Es kann ein beliebige Laufwerk (1-15) zum aktuellen Laufwerk bestimmt
'werden. Nach Abschluß der Funktion liegt in "A" der Index des vorher
'aktiven Laufwerks.

A=Gemdos(&HE,Desdrv)

Return

,

Procedure Z.printer(S.string\$)

'Es wird ein Stringausdruck übergeben, der dann auf dem Bildschirm
'an der Cursorposition ausgegeben wird.

A=Gemdos(&H9,L:Varptr(S.string\$))

Return

,

Procedure Diskbuffer

'Gibt in "A" die Adresse des Disk-Buffers (512 Byte) zurück und
schreibt

'den aktuellen Inhalt auf den Bildschirm.

A=Gemdos(&H2F)

Print A

For I=0 To 512

Print Chr\$(Peek(A+I));

Next I

Return

,

Procedure Conout(Zeichen\$)

'Gibt ein einzelnes Zeichen auf dem Bildschirm aus.

Aa=Asc(Zeichen\$)

A=Gemdos(&H2,Aa)

Return

,

Procedure Conin

'Wartet, bis ein einzelnes Zeichen über die Tastatur eingegeben wird
'und gibt das Zeichen anschließend auf dem Bildschirm aus.

A=Gemdos(&H1)

```

Zeichen$=Chr$(Dpeek(Varptr(A)+4))
Print Zeichen$'A
Return

```

BIOS-FUNKTIONEN

BIOS

Bei "Bios" handelt es sich prinzipiell um denselben Vorgang. Nur das hier eine andere TOS-Ebene angesprochen wird, nämlich das "BIOS".

Procedure Sondertasten(Keystat)

```

'Wird Keystat = -1 gesetzt, koennen die Sondertasten abgefragt werden.
'   A=1 ==> linke Shift-Taste wurde gedruickt
'   A=2, ==> rechte " " " "
'   A=4 ==> Contrl-Taste / A=8 ==> Alternate-Taste
'   A=16 ==> CapsLock an
'   Kombination z.B. A=22 ==> CapsLock an + Shift links + Contrl
'EIn positiver Wert wird als Sondertasten-Modus uebernommen.
A=Bios(11,Keystat)

```

Return

!

Procedure Sektormod(Rwflag%,Buffer\$,Anzahl%,Recno%,Dev%)

```

'z.B. @Sektormod(0,Space$(1200),4,10,0)
'Liest und schreibt Sektoren von/auf Diskette.
' "Rwflag" = Funktions-Identifikator
'   0 = Sektoren lesen
'   1 = Sektoren schreiben
'   2 = Sektoren lesen / Diskwechsel ignorieren
'   3 = Sektoren Schreiben / Diskwechsel ignorieren
' "Buffer$"= Stringpuffer, in den hineingeschrieben- bzw. aus dem
'   herausgelesen werden soll. Er muB auf jeden Fall so groB
'   sein, daB alle Daten der Sektoren darin Platz haben.
'   Wenn Sie also 3 Sektoren lesen moechten, muB er
'   3*512=1536 Byte groB sein.
' "Anzahl" = Anzahl der Sektoren, die hintereinander
'   gelesen/geschrieben werden sollen.

```

```
' "Recno" = Gibt an, mit welchem logischen Sektor (1-720) begonnen
'           werden soll.
' "Dev"    = Die Laufwerksnummer (0=A; 1=B; 2=Harddisk)
A%=Bios(4,Rwflag%,L:Varptr(Buffer$),Anzahl%,Recno%,Dev%)
Print Buffer$
Return
'
Procedure Diskparameter(Flopnr)
'Gibt die Adresse des "DPB" (Diskparameterblock) für das angegebene
'Laufwerk (0=A; 1=B) zurück.
'    Der DPB ist folgendermaßen aufgebaut:
'    Word 1 = Sektorgröße in Bytes (512)
'    Word 2 = Clustergröße in Sektoren (2)
'    Word 3 = Clustergröße in Bytes (1024)
'    Word 4 = Directorylänge in Sektoren
'    Word 5 = FAT (Free Allocation Table) in Sektoren
'    Word 6 = Sektornummer des 2. FAT
'    Word 7 = Sektornummer des 1. Datenclusters
'            (cluster;engl.:Traube)
'    Word 8 = Anzahl der Datencluster auf Diskette
Parablock=Bios(7,Flopnr)
For I=0 To 14 Step 2
    Diskpara$=Diskpara$+Str$(Dpeek(Parablock+I))+ " "
Next I
Print Diskpara$
Return
'
Procedure Changedisk(Flopnr)
'Ermittelt, ob zwischenzeitlich die Diskette gewechselt wurde.
'Dazu wird in "Flopnr" die Laufwerksnummer übergeben.
'In "Changestat" wird ein Wert zurückgegeben:
'    0 = Diskette wurde nicht gewechselt
'    1 = möglicherweise Diskettenwechsel
'    2 = Diskette wurde gewechselt
Changestat=Bios(9,Flopnr)
Return
```

XBIOS-FUNKTIONEN

XBIOS

Auch die Syntax der "Xbios"-Aufrufe ist identisch mit den vorangegangenen. Diesmal werden die "XBIOS"-Routinen des TOS (Tramiel Operating Sysytem) aufgerufen.

Procedure Keytakter(Verz,Takt)

'Setzt die Reaktionsverzögerung und Repeattakt für Tastaturklick.

'1 steht für sehr schnell, je größer die Zahl, um so langsamer.

'Wenn für Repeattakt 0 gesetzt wird, sind nur einzelne Tastenklicks

'möglich.

A=Xbios(35,Verz,Takt)

Return

,

Procedure Setpalette(Colbuff\$)

'Es kann mit einem Befehl die gesamte Farbpalette ausgewechselt

'werden.Dazu muß ein String übergeben werden, der nacheinander die

'16 Farbwerte in Wortgröße enthält.

'z.B.:

'For I=1 To 16

' Farbe=Random(511)

' Palette\$=Palette\$+Mki\$(Farbe)

' Next I

' @Setpalette(Palette\$)

A=Xbios(6,L:Varptr(Colbuff\$))

Return

,

Procedure Getreso

'ermittelt die aktuelle Bildschirmauflösung

' A = 0 ==> niedrige Auflösung (LOWRES=320*200 / 16 Farben)

' A = 1 ==> mittlere Auflösung (MIDRES=640*200 /4 Farben)

' A = 2 ==> hohe Auflösung (HIGHRES=640*400 / monochrom)

A=Xbios(4)

Return

,

Procedure Midiout(Anzahl,S.tring\$)

'Hiermit wird ein String an den Midi-Out-Port gesendet.

'Die Zeichen dieses Strings müssen als Stringvariable übergeben werden.

'"Anzahl" gibt die Anzahl der zu sendenden Zeichen -1 an.

A=Xbios(12,Anzahl,L:Varptr(S.string\$))

Return

Anhang A: Prozeduren in "GRAF PRO"

Prozedurname	Seite
A.CP.....	212
A.CR	219
A.RRECP.....	210
A.RRECR.....	218
A.SRECP.....	209
A.SRECR.....	217
ACCESSORY.....	384
ALTERSCREEN.....	133
AREAPROCS	241
BILDLOADER	373
BILDSAVER	373
BLOCK.H.GROW	249
BLOCK.H.MIR	252
BLOCK.H.SHRINK	251
BLOCK.V.GROW	251
BLOCK.V.MIR	253
BLOCK.V.SHRINK	252
BLOCK.X.FADE	259
BLOCK.Y.ELLY	253
BLOCKLOADER.....	374
BUSY	330
B_LOCKERROR	249
C.ONST.LINES.....	287
CHANGEMOD	305
CINEMA.....	348
CLEARPLANE	330
CLICKS	391
CLIPPREC.....	387
C_LIPPREC2.....	388

D.EF.PATCH.2.....	272
D.EFPATCH.....	271
D.SIZE	393
DISK.ERROR	383
DISK.OPS	151
DIV.CERO.....	384
DIVBYCERO.....	382
DOWNDICK.....	290
DOWNPATCH.....	292
DOWNPATTERN	289
DRAGBOX.....	375
DRAGWINDOW.1.....	343
DRAW.EDPIC	337
DRAWICONRASTER.....	343
DRAWRASTER	330
E.DRAGBOX	343
EXTEND	211
F.CP	218
F.ILLPRO	229
F.LIPSERVICE.....	136
F.RRECP	209
F.RRECR	217
F.SRECP	206
F.SRECR	216
FORMALERT.....	390
G.ETFLIP	135
G.ETTINNY	362
G.RUBBERBOX	388
GETFKEYS.....	280
GETPICINITS	350
GET_PICS.....	354
GET_SIZE	180
GRAF̄MENUE.....	73
GROW.SHRINK.BOX	389

H.ELPTEXT	394
HIDEMOUSE	387
ICONP.....	212
ICONSAVER.....	374
INITARRAY.....	338
INITS.....	407
INITVAR.W1.....	344
K.ILL.....	386
L.CP	211
L.CR	219
LOAD.ERROR.....	383
L.RRECP	210
L.RRECR	218
L.SRECP.....	208
L.SRECR	217
LNFORM.....	290
LNSTYLE.....	290
MENUE	
M.ENUTEXT.1	143
M.ENUTEXT.2?	142
M.ENUTEXT.3?	150/151
M.ENUTEXT.4	151
MAINMENUE	288
MARKER.....	231
MARKERMENU.....	232
MAUSABFRAGE	316
MIR.1	279
MIR.1.LINE.1.....	227
MIR.1.LINE.2.....	228
MIR.2.....	280
MIR.2.LINE.1.....	228
MIR.2.LINE.2.....	228
MIR.3.....	280
MIR.3.LINE.1.....	228
MIR.3.LINE.2.....	228
MIR.4.....	280
MIR.4.LINE.1.....	228

N.NAME_ERROR.....	198
P.UTFLIP	135
PATEDIT.....	344
PATTERNSAVER	374
PL.SPR.....	230
PRE.LOADING.....	35
PREDEFS	205
PT.SPR.....	231
P_ICLOAD	365
P_ICSAVE	367
RAHMEN.....	388
READY.....	330
RUBBERBOX.1	331
RUBBERBOX.2	331
RUBBGOING.....	215
RUBBSELF	376
S.AVE.ERROR.....	384
S.LIN.....	220
SCHLUSS.....	394
SCREENSET	126
SET.BLACK	336
SET.CLIPPING.....	142
SET.CONST	143
SET.GRID.....	144
SET.INITS.....	138
SET.PICS.....	287
SET.RASTERFORM.....	316
SET.SPECIALS.....	147
SET.WHITE	336
SETAREAS.....	245
SETDRAWMODE	292
SETGRAFMODE.....	300
SETNEWMENU	300
SETPLANE.....	332
SETPOINT.....	335
SETSTEPS.....	291
SET_MENUBACK.....	241

SHOWHELP.....	303
SHOWLINES.....	287
SHOWMOUSE.....	386
SHOWPICS.....	370
SL.SOME	198
SMALLTEXT.....	391
STARTER	307
T.EXT.....	234
T.EXTMENU.....	236
TINNY.....	256
T_EXTLEN.....	286
UPDICK	289
UPPATCH.....	292
UPPATTERN.....	288
VERTEILER.....	272
WATCHBOUNDS	332

Anhang B: Bedienungsanleitung zum "Graf Pro"

Funktionstasten

Im Grafikteil :

- F1 = Füllmuster +
- F2 = Füllmuster -
- F3 = Liniendicke +
- F4 = Liniendicke -
- F5 = Linenstil
- F6 = Linienendformen
- F7 = Patch-Größe +
- F8 = Patch-Größe -
- F9 = Grafik-Auswahl (+)
- F10 = Grafik-Modus (+)

Andere Tasten:

- Undo = seit dem letzten Insert gezeichnete Sequenz löschen
- Insert=aktuelle Screen fixieren
- Help=Evtl. Helptexte
- Clr/Home=aktuelle Screen löschen
- Esc=Hauptmenü aufrufen
- Tab=Grafik-Einstellungsmenü aufrufen

Die Tasten können auch während des Zeichnens bedient werden.

Menübedienung:

Im Hauptmenü:

Entsprechende Menüfelder anklicken.

!! alle Funktionen sind gleichzeitig aktivierbar !!

Menupkt. 'QUIT' = Programmende

Menupkt. 'INITS':

1. vier Spiegelfunktionen und Spiegel Aus
2. zwei Hilfslinien-Funktionen und Aus
3. Netzmodus An/Aus

Im Netzmodus kann ein beliebiges Raster eingestellt werden. Sämtliche Grafikfunktionen (auch Text !!) arbeiten dann in diesem Raster.

Menupkt. 'EDITS':

1. Icon Editor mit max. 60*80 Punkten
2. Pattern-Editor mit 16*16 Punkten

Das damit entworfene Füllmuster kann anschließend im Mustermenü aufgerufen werden.

Der jeweils letzte Editor-Inhalt läßt sich nach Rückkehr zum Grafikprogramm wieder installieren.
Funktionen:

Eckpunkt links oben = Window-Maximierung

Punkt rechts daneben= Revers

Eckpunkt rechts oben= Editorinhalt wird übernommen
und Rückkehr zum Hauptproramm

Punkt links daneben = CANCEL. Editorinhalt wird
nicht übernommen. Rückkehr zum
Hauptprogramm

Punkt unter Eckpunkt

oben rechts = CLEAR

Punkte links und

rechts in der Mitte = H/V-Spiegel

Punkt rechts unten = Rubber (Taste gedrückt halten
und Box variieren.)

Punkt darüber = Editorinhalt speichern

Eckpunkt links unten= Draggar (Taste gedrückt halten
und Box verschieben

Vier Pfeilpunkte = Inhalt scrollen

Menupkt. 'KINO':

Kino wird weiter unten genauer erklärt.

Menupkt. 'SPEZI':

1. die 'üblichen' Move/Copy - Funktionen,
bis man dabei auf die Cursor Tasten drückt.
Hat man einen Block mit MOVE oder COPY
definiert, beginnt er zu flimmern. Jetzt
können mit den vier Cursortasten Sonder-
funktionen ausgelöst werden:

Cursor-links-Taste = Vertikal spiegeln
Cursor-unten-Taste = Horizontal spiegeln
Cursor-oben-Taste = Tonnenbild
Cursor-rechts-Taste=Sinus-Bild
Shift/Cursor-links = Block horizontal verkleinern
Shift/Cursor-oben = Block vertikal verkleinern
Shift/Cursor-rechts= Block horizontal dehnen
Shift/Cursor-unten = Block vertikal dehnen

Nachdem diese Funktionen ausgeführt sind,
können die neuen Blöcke gesetzt werden.
Schaltet man auf die andere Screen um, wird der
Block dort wieder aktiviert. Der Block bleibt
auch nach dem Setzen(einmaliger Tastendruck)
solange durch Umschalten zur anderen Screen aktiv,
bis man die Maustaste zweimal hintereinander
bedient. Es kann aber auch nach einmaligem Tasten-
druck wieder ein neuer Block definiert werden.
Dann geht allerdings der bisher aktive Block
verloren.

Wenn man die MOVE/COPY-Funktionen wieder aus-
schaltet und mit SELF weiterarbeitet, kann man
mit diesen Blöcken sogar malen.

2. BLOCK Laden / Speichern

Die Funktion fordert bei SAVE auf, einen Bildaus-
schnitt zu definieren. Dieser Ausschnitt wird dann
mit dem Extender 'BLO' auf Diskette gespeichert.
Mit LOAD können Auschnitte von Diskette geladen
werden

3. CLIPPING setzen / CLIPP. an/aus

Man kann einen Bildschirmbereich definieren (wie bei MOVE/COPY oder den Rubb-Funktionen wird die linke Maustaste gedrückt und eine Box aufgezogen. Beide Tasten = UNDO).

Alle Grafikfunktionen außer SELF und MOVE/COPY werden dann an den Rändern dieses Feldes abgeschnitten. Man kann also nur noch innerhalb dieses Feldes zeichnen oder schreiben.

4. BORDER On/Off

Die Bildschirmränder können aktiviert werden. Kommt man während des Zeichnens an bestimmte Punkte am Bildrand, wartet das Programm auf einen Tastendruck. Wird dann zweimal schnell geklickt (wie auf dem Desktop beim Datei-Eröffnen), werden Sonderfunktionen ausgelöst.

Da man während des Zeichnens sowieso immer eine Taste gedrückt hält, stört dieser Umstand sehr wenig. Er zeigt seine Wirkung erst wenn man mit nicht gedrückter Maustaste die Ränder berührt. Der Mauszeiger flackert dann nicht mehr und man kann jetzt doppelklicken.

Ausgelöst werden:

Maus im mittleren Bereich am linken Bildrand:

=> es erscheint die 'andere' Screen. Man kann sie mit der Maus bewegen. Sie verschwindet wieder, wenn man noch einmal klickt.

Maus in der linken, unteren Ecke:

=> Es wird auf SELF-Funktion umgeschaltet.

Die vorher aktive Funktion (außer MOVE/COPY) wird zwischengespeichert und kann mit noch-

maligem Doppelklick (nach Berührung der Bildschirmecke) wieder zurückgeholt werden.

Maus in der rechten unteren Ecke:

==> Es werden die Mauskoordinaten in der linken oberen Bildschirmecke angezeigt.
Außerdem werden bei Rubb-Funktionen und dem Straight-Line-Modus die x/y-Entfernungen der Maus zum Ursprung und die Länge der Diagonalen angezeigt (mathematisch korrekt nach Phytagoras). Nochmal rechte untere Ecke berühren und Doppelklick = aus.

Maus im mittleren Bereich des unteren Randes:

==> Das Einstellmenü wird aufgerufen.

Maus im mittleren Bereich des rechten Randes:

==> Der Editor wird aufgerufen.

Menupkt. 'FLIPP'

Es wird auf die andere Screen umgeschaltet.
Noch nicht fixierte MOVE/COPY-Blöcke können so von einer Screen in die andere gebracht werden.

Menupkt. 'DISKY'

LOAD / SAVE / KILL und NAME - Funktionen

Menupkt. 'PRINT'

Hardcopy der aktuellen SCREEN (Druckeranpassung kann mit Desktop-Accessory vorgenommen werden). Accessories können aufgerufen werden.

Im Einstell-Menü:

Das Titelfeld jeweils anklicken oder Funktionstasten drücken.

Besonderheit bei Feld 'Draw-Modus':

Dieser Menüpunkt kann unabhängig davon, ob Menü-Icons geladen wurden, bedient werden.

Wenn Menü-Icons geladen wurden, muß nur in das Titelfeld geklickt werden. Es erscheint das Grafik-Menü.

Hier ist eine Besonderheit bei den Menüfeldern MARKER und TEXT zu beachten! Wird nur eine Maustaste gedrückt, wird die Funktion einfach nur angewählt.

Werden beide Tasten gleichzeitig gedrückt, erscheint ein Untermenü.

Wurden keine Menü-Icons geladen, können mit den beiden Maustasten im Titelfeld die Draw-Modi vor und zurückgewählt werden.

Wieder Sonderfall MARKER und TEXT:

Steht einer dieser beiden Draw-Titel in der Box, muß die 'Tab'-Taste nochmal gedrückt werden, um die Untermenüs aufzurufen.

Besonderheit SELF:

Mit dieser Funktion können beliebige Icons oder Bildausschnitte, die vorher mit dem Editor entworfen bzw. mit MOVE/COPY (s.Menüpkt. 'SPEZI') definiert wurden (nach der Definition den Block einfach aus dem Bild schieben und MOVE/COPY wieder ausschalten), als Zeichen-Element benutzt werden. Diese Funktion arbeitet ebenfalls in allen vier Grafik-Modi (Replace, Transp., XOR, Revers).

Man kann auch beliebig Icons oder Blöcke laden und damit zeichnen.

Achtung:

Wenn man Icons oder Blöcke aus dem Bild hinaus-schiebt, findet man sie evtl. nicht wieder. Sollte Ihnen das passieren, brauchen Sie nur nochmal "SELF" anzuwählen und das Icon erscheint wieder in der linken oberen Bildschirmecke.

Bedienungsfunktionen der einzelnen Grafikoperationen:

Alle "Patch"-Funktionen (1-9) werden durch Druck auf die linke Maustaste ausgelöst.

SELF-Funktion (10) : Mit der linken Taste wird das Icon gezeichnet mit der rechten Maustaste wird es bewegt, ohne zu zeichnen.

"Rubb"-Funktionen (11-19):

Mit der linken Taste wird der Startpunkt definiert.

Anschließend zusätzlicher Druck auf die rechte Taste (also beide Tasten zugleich) löscht die Box.

Dabei muß darauf geachtet werden, daß anschließend die

linke Taste zuerst wieder losgelassen wird.

Drücken Sie nur die linke Taste und haben eine Box

aufgezogen, wird diese gezeichnet, sobald die Taste

wieder losgelassen wird.

"Line"-Modus (20): linke Taste = Freihandzeichnen
rechte Taste= Gerade Linie definieren
anschließender Druck zusätzlich auf die rechte Maustaste zeichnet die Linie.

"Fill"-Modus (21): linke Maustaste löst den Füllprozeß aus.

"Spray"-Modus(22 u. 23):

Im Spraymodus ist folgendes zu beachten:

Die Größe des Spraybereichs richtet sich nach der eingestellten "Patch"-Größe.

Es sind alle drei Maustasten mit Funktionen belegt.

linke Maustaste = kleine Punkte sprühen

rechte Taste = etwas größere Punkte sprühen
 beide Tasten = dicke Punkte sprühen

Es gibt zwei verschieden Spraymodi:

1. Plot-Spray = sprüht einfarbig schwarze Punkte
2. Patt-Spray = sprüht das eingestellte Füllmuster

"Marker" (24) Es kann mit sechs verschiedenen Markertypen gezeichnet werden. Die Größe der Marker richtet sich nach der eingestellten "Patch"-Größe. Allerdings lassen sich Marker nur sprungweise in der Größe verändern, nicht kontinuierlich.

"Text" (25) Es kann Grafik-Text ausgegeben werden. Druck auf linke Maustaste setzt den String, ohne ihn zu löschen.
 Der String wird auch durch Drücken der "Return"-Taste gesetzt. Rechte Maustaste oder "Delete"-Taste löscht den String. "Backspace"-Taste löscht jeweils das letzte Zeichen.
 Außerdem sind auch hier die vier Cursortasten belegt.
 Cursor-links-Taste = String in der Länge verkürzen
 (Achtung: die Stringlänge fängt bei Null an).
 Cursor-rechts-Taste = String dehnen.
 Cursor-unten-Taste = Dehnungsmodus kann bestimmt werden.

Im Wechsel:

Wort- oder Zeichendehnung.

Cursor-oben-Taste = Normale Textlänge wieder einstellen.

"KINO":

Einstiegspunkt ist die Definition einer Screen-Größe (Taste gedrückt halten und Box aufziehen). Anschließend kann bestimmt werden, wieviele Bilder gezeichnet bzw. bearbeitet werden sollen. Hier richtet sich die maximale Anzahl der möglichen Bilder nach der vorher definierten Bildgröße und dem verfügbaren Speicherplatz.

Trickfilm-Editor:

Er erklärt sich weitgehend selbst.

Die Plot-Größe kann während des Zeichnens durch die Maustasten-Kombinationen variiert werden:

linke Taste = konstant

rechte Taste = größer

beide Tasten = kleiner

Funktionstasten sind während des Zeichnens aktiv.

F1= Zeichenfarbe schwarz

F2= Zeichenfarbe weiß

F3= aktuelles Bild 'knipsen' / Bildzähler +1

F4= Zeichenfenster clearen

F5= Tricks wählen

Trick BUBBLE:

Es wird eine glänzende Kugel dargestellt.

Größe kann wie oben mit Tasten variiert werden. Der Bildzähler incrementiert automatisch, solange Maustaste gedrückt ist.

Trick TROMMEL:

Lassen Sie sich überraschen und legen Sie schon mal eine Leerdiskette bereit.

F6= SAVE - Film / Einzelbilder speichern.

F7= LOAD - Es können entweder Icons, Blöcke, Einzelbilder

oder Filme geladen werden.

Der Film wird automatisch gestartet.

Icons, Blöcke und Einzelbilder können nach dem Laden durch Drücken der HELP-Taste in den Editor geholt und mit Maustastendruck gesetzt werden.

F8 = Weiter zur Vorführung oder Rückkehr zum Hauptprogramm

Ausserdem kann:

1. das aktuelle Bild mit den vier Cursortasten gescrollt werden
2. mit der "Help"-Taste der zuletzt im Grafikprogramm aktive Block, bzw. das SELF-Icon in den Editor geholt werden.

Picture-Show:

Der editierte Film wird vorgeführt.

Mit den beiden Maustasten kann die Geschwindigkeit gesteuert werden.

Standbild ==> mit dem Mauszeiger unteren Bildschirmrand berühren.

Einzelbildschaltung ==> unteren Rand berühren und gleichzeitig klicken.

Bilddrücklauf ==> Mauszeiger auf 'Backstep' und Maustaste drücken.

Anhang C: Fehlermeldungen

EDITOR-FEHLERMELDUNGEN:

TEXT:

WHILE OHNE WEND
REPEAT OHNE UNTIL
DO OHNE LOOP
FOR OHNE NEXT
WEND OHNE WHILE
LOOP OHNE DO
NEXT OHNE FOR
IF OHNE ENDIF
ENDIF OHNE IF
ELSE OHNE IF
ELSE OHNE ENDIF
EXIT OHNE SCHLEIFE
PROCEDURE OHNE RETURN
PROCEDURE IN SCHLEIFE
PROCEDURE DOPPELT DEFINIERT
RETURN OHNE PROCEDURE
MARKE DOPPELT DEFINIERT
LOCAL NUR IN PROCEDURE
LOCAL NICHT IN SCHLEIFE
FUNKTION DOPPELT DEFINIERT
GOTO IN/AUS FOR-NEXT ODER PROCEDURE
RESUME IN FOR-NEXT-SCHLEIFE
RESUME OHNE PROCEDURE
SYNTAX FEHLER
ZEILE ZU LANG

BASIC-FEHLERMELDUNGEN:

ERR :	TEXT:
0	DIVISION DURCH NULL
1	ÜBERLAUF
2	ZAHL NICHT INTEGER -2147483648...2147483648
3	ZAHL NICHT BYTE 0...255
4	ZAHL NICHT WORT 0...65535
5	QUADRATWURZEL NUR FÜR POSITIVE ZAHLEN
6	LOGARITHMEN NUR FÜR ZAHLEN > 0
7	UNBEKANNTER FEHLER
8	SPEICHER VOLL
9	FUNKTION ODER BEFEHL NOCH NICHT MÖGLICH
10	STRING ZU LANG. MAX. 32767 ZEICHEN
11	KEIN GfA-BASIC V1.0 PROGRAMM
12	PROGRAMM ZU LANG, SPEICHER VOLL. NEW
13	KEIN GfA-BASIC. PROGRAMMFILE ZU KURZ. NEW
14	FELD ZWEIMAL DIMENSIONIERT
15	FELD NICHT DIMENSIONIERT
16	FELD INDEX ZU GROß
17	DIM INDEX ZU GROß
18	FALSCH ANZAHL INDIZES
19	PROCEDURE NICHT GEFUNDEN
20	LABEL NICHT GEFUNDEN
21	BEI OPEN NUR ERLAUBT: "I"INPUT, "O"OUTPUT, "R"RANDOM, "A"APPEND, "U"UPDATE
22	FILE SCHON GEÖFFNET
23	FILE # FALSCH
24	FILE NICHT GEÖFFNET
25	FALSCH EINGABE. KEINE ZAHL
26	FILEENDE ERREICHT EOF
27	ZUVIEL PUNKT FÜR POLYLINE/POLYFILL. MAX.128
28	FELD MUß EINDIMENSIONAL SEIN
29	ANZAHL PUNKTE GRÖßER ALS FELD
30	MERGE - KEIN ASCII-FILE
31	MERGE - ZEILE ZU LANG - ABRUCH
32	==> SYNTAX NICHT KORREKT
33	MARKE NICHT DEFINIERT

34	ZUWENIG DATA
35	DATA NICHT NUMERISCH
36	SYNTAXFEHLER IN DATA. "" PAARWEISE VERWENDEN
37	DISKETTE VOLL
38	BEFEHL IM DIREKTMODUS NICHT MÖGLICH
39	PROGRAMMFEHLER. KEIN GOSUB MÖGLICH
40	CLEAR NICHT MÖGLICH IN FOR-NEXT-SCHLEIFEN ODER PROZEDUREN
41	CONT NICHT MÖGLICH
42	ZUWENIG PARAMETER
43	AUSDRUCK ZU KOMPLEX
44	FUNKTION NICHT DEFINIERT
45	ZUVIELE PARAMETER
46	PARAMETER FALSCH. MUß ZAHL SEIN
47	PARAMETER FALSCH. MUß STRING SEIN
48	OPEN "R" SATZLÄNGE FALSCH
50	KEIN "R"-FILE
51	NUR EIN FIELD ZU EINEM OPEN "R" MÖGLICH
52	FIELDS GRÖßER ALS SATZLÄNGE
53	ZU VIELE FIELDS (MAX. 9)
54	GET/PUT FIELD-STRINGSLÄNGE FALSCH
55	GET/PUT SATZNUMMER FALSCH
60	SPRITE-STRINGSLÄNGE FALSCH
90	FEHLER BEI LOCAL
91	FEHLER BEI FOR
92	RESUME (NEXT) NICHT UNMÖGLICH. FATAL, FOR ODER LOCAL
100	C COPYRIGHT 1986, GFA SYSTEMTECHNIK

TOS-FEHLERMELDUNGEN:

ERR :

TEXT:

- 1 * ALLGEMEINER FEHLER
- 2 * DRIVE NOT READY. ZEITÜBERSCHREITUNG
- 3 * UNBEKANNTER BEFEHL
- 4 * CRC FEHLER, DISK-PRÜFSUMME FALSCH
- 5 * BAD REQUEST. UNGÜLTIGER BEFEHL
- 6 * SEEK ERROR. SPUR NICHT GEFUNDEN
- 7 * UNKNOWN MEDIA. FALSCHER BOOTSEKTOR
- 8 * SEKTOR NICHT GEFUNDEN
- 9 * KEIN PAPIER
- 10 * SCHREIBFEHLER
- 11 * LESEFEHLER
- 12 * ALLGEMEINER FEHLER 12
- 13 * DISKETTE SCHREIBGESCHÜTZT
- 14 * DISKETTE WURDE GEWECHSELT
- 15 * UNBEKANNTES GERÄT
- 16 * BAD SEKTOR (VERIFY)
- 17 * ANDERE DISKETTE EINLEGEN
- 32 * UNGÜLTIGE FUNKTIONSNUMMER
- 33 * DATEI NICHT GEFUNDEN
- 34 * PFADNAME NICHT GEFUNDEN
- 35 * ZUVIELE DATEIEN OFFEN
- 36 * ZUGRIFF NICHT MÖGLICH
- 37 * UNGÜLTIGES HANDLE
- 39 * SPEICHER VOLL
- 40 * UNGÜLTIGE SPEICHERBLOCKADRESSE
- 46 * UNGÜLTIGE LAUFWERKSBEZEICHNUNG
- 49 * KEINE WEITEREN DATEN
- 64 * GEMDOS BEREICHSFEHLER. SEEK FALSCH?
- 65 * INTERNER GEMDOS-FEHLER
- 66 * KEIN BINÄRPROGRAMMFILE
- 67 * SPEICHERBLOCKFEHLER

BOMBEN-FEHLERMELDUNGEN:

ERR :	TEXT:
102	2 BOMBEN - BUS ERROR. VIELLEICHT PEEK ODER POKE FALSCH
103	3 BOMBEN - ADRESS ERROR. UNGERADE WORTADRESSE! BEI DPOKE, DPEEK, LPOKE ODER LPEEK
104	4 BOMBEN - ILLEGAL INSTRUCTION. AUSFÜHRUNG EINES UNGÜLTIGEN 68000-MASCHINENBEFEHLS
105	5 BOMBEN - DIVIDE BY CERO. DIVISION DURCH NULL IN 68000-MASCHINENSPRACHE
106	6 BOMBEN - CHK-EXEPTION. 68000-INTERRUPT DURCH CHK-BEFEHL
107	7 BOMBEN - TRAPV-EXEPTION. 68000-INTERRUPT DURCH TRAPV-BEFEHL
108	8 BOMBEN - PRIVILEGE VIOLATION. 68000-INTERRUPT DURCH AUSFÜHRUNG EINES PRIVILIGIERTEN BEFEHLS
109	9 BOMBEN - TRACE-EXEPTION. 68000-TRACE-INTERRUPT

Anhang D: Übersicht der BASIC-Befehle

Befehlsname	Abkürzung	Seite
ABS		224
ADD	Abk.: Ad	105
ALERT	Abk.: A	119
ARRAYFILL	Abk.: Ar	203
ARRPTR		59
ASC		103
ATN		263
BIN\$ * &X		87
BIOS		436
BLOAD	Abk.: Bl	39
BOX	Abk.: B	98
BSAVE	Abk.: Bs	200
C:		389
CALL	Abk.: Ca	389
CHAIN	Abk.: Ch	182
CHDIR	Abk.: Chd	177
CHDRIVE	Abk.: Chdr	180
CHR\$		103
CIRCLE	Abk.: C	212
CLEAR	Abk.: Cle	49
CLEARW	Abk.: Cle W	415
CLOSE	Abk.: Cl	189
CLOSEW	Abk.: Cl W	412
CLS		127
COLOR	Abk.: Co	75
CONT	Abk.: Con	48
CONTRL		429
COS		262
CVI * CVL * CVS * CVF * CVD		92
DATA	Abk.: D	56
DATE\$		109
DEC		105
DEFFILL	Abk.: Deff	85
DEFFN		266
DEFLINE	Abk.: De	78
DEFLIST	Abk.: Deflis	184

DEFMARK	Abk.: Defm	232
DEFMOUSE	Abk.: Defmo	43
DEFTXT	Abk.: Deft	99
DFREE		197
DIM	Abk.: Di	40
DIM?		42
DIR		178
DIR\$		178
DIV		105
DO * LOOP	Abk.: Do, L	107
DPEEK		70
DPOKE		67
DRAW * TO	Abk.: Dr	339
EDIT	Abk.: Ed	48
ELLIPSE	Abk.: Ell	198
END		48
EOF		190
ERASE	Abk.: Er	43
ERR		167
ERROR	Abk.: Err	169
EXIST		37
EXIT IF	Abk.: E	107
EXP		225
FATAL		167
FIELD * AS	Abk.: Fi, As	197
FILES	Abk.: File	179
FILESELECT	Abk.: Filese	152
FILL	Abk.: Fi	132
FIX		116
FN	Abk.: @	266
FOR * NEXT	Abk.: F, N	53
FORM INPUT	Abk.: F input	165
FRAC		116
FRE		58
FULLW	Abk.: Fu	
GB		425
GEMDOS		434
GEMSYS		423
GET		81
GET #		196
GOSUB	Abk.: Go bzw. @	35
GOSUB "Procedure"(Liste)	Abk.: Go bzw. @	50
GOTO label	Abk.: Got	38

GRAPHMODE	Abk.: G	73
HARDCOPY	Abk.: H	124
HEX\$ * &H		87
IF * ELSE * ENDIF	Abk.: I, El, En	36
INC	Abk.: In	105
INFOW	Abk.: Inf	413
INKEY\$		111
INP		93
INPUT	Abk.: Inp	162
INPUT\$		163
INSTR		171
INT		115
INTIN		430
INTOUT		430
KILL	Abk.: K	153
LEFT\$		170
LEN		111
LET	Abk.: Le	169
LINE	Abk.: Li	144
LINE INPUT	Abk.: Li input	164
LIST	Abk.: Lis	182
LLIST	Abk.: Ll	313
LOAD	Abk.: Loa	181
LOCAL	Abk.: Loc	52
LOC		191
LOG * LOG10		266
LPEEK		70
LPOKE	Abk.: Lp	67
LPOS		314
LPRINT	Abk.: Lpr	313
LSET	Abk.: Ls	172
MAX		400
MENU		408
MENU array\$()	Abk.: Me array\$()	407
MENU KILL	Abk.: Me Kill	412
MENU OFF	Abk.: Me Off	414
MID\$		171
MIN		400
MKDIR	Abk.: Mk	176
MKI\$ * MKL\$ * MKS\$ * MKF\$ * MKD\$		91
MOUSE	Abk.: M	113

MOUSEK		114
MOUSEX		114
MOUSEY		114
MUL	Abk.: Mu	105
NAME * AS	Abk.: Na, As	175
NEW		49
OCT\$ * &O		87
ON * GOSUB		278
ON BREAK GOSUB		36
ON ERROR GOSUB		165
ON MENU		404
ON MENU GOSUB		403
ON MENU IBOX GOSUB		422
ON MENU KEY GOSUB		403
ON MENU MESSAGE GOSUB		403
ON MENU OBOX GOSUB		422
OPEN	Abk.: O	187
OPENW	Abk.: O W	407
OUT	Abk.: Ou	94
PAUSE	Abk.: Pa	118
PBOX	Abk.: Pb	97
PCIRCLE	Abk.: Pc	211
PEEK		70
PELLIPSE	Abk.: Pe	219
PI		265
PLOT	Abk.: Pl	338
POINT		337
POKE	Abk.: Po	67
POLYFILL	Abk.: Polyf	207
POLYLINE	Abk.: Pol	207
POLYMARK	Abk.: Polym	207
POS		158
PRBOX	Abk.: Prb	210
PRINT	Abk.: P bzw. ?	154
PRINT USING	Abk.: P Using	159
PROCEDURE * RETURN	Abk.: Pro, Ret	35
PROCEDURE "Name"(lokale Variablenliste)		50
PSAVE	Abk.: Ps	181
PTSIN		430
PTSOUT		430
PUT	Abk.: Pu	81
PUT #	Abk.: Pu	196

UPPER\$		55
VAL		96
VAL?		96
VARPTR		57
VDISYS		429
WAVE	Abk.: Wa	394
WINDTAB		418
WHILE * WEND	Abk.: W, We	106
WRITE	Abk.: Wr	160
XBIO\$		438

Anhang E: Programmänderungen für ATARI

Das gesamte in diesem Buch aufgeführte Grafikprogramm ist auf den ATARI ST mit 1 MB Arbeitsspeicher zugeschnitten. Besitzer eines ST mit einem Speicher von 512 KB müssen auf die Möglichkeit der Trickfilmedition verzichten. Im übrigen ist die 260er Version weitgehend identisch mit der Megaversion. Voraussetzung ist ebenfalls das ROM-TOS.

1. Programmteile löschen

Folgende Prozeduren und Programmzeilen müssen vollständig gelöscht werden:

```
procedure: cinema
           getpicinit
           set_pics
           g.ettinny
           p_icsave
           showpics
           set.pics
           bildsaver
           bildloader
```

Programmzeilen:

```
           procedure grafmenue
           Zeile 6 - 12
           Zeile 34, 35, 37, 38
```

```
procedure screenset
           Zeile 2 - 5
           Zeile 7 - 13
           Zeile 19
```

procedure setinits

Zeile 4 - 10

Zeile 38 - 40

Zeile 104 - 107

procedure set.specials

Zeile 5 - 8

Zeile 28 und 29

Zeile 129

procedure sl.some

Zeile 97 - 100

Zeile 110 - 113

procedure pre.def

Zeile 2

procedure mainmenue

Zeile 3 - 6

2. Einfügungen

Folgende Programmzeile müssen unbedingt eingefügt werden:

Anfang - vor das Programm:

```
if fre(0)<140000
on break gosub s.system
alert 1,"Der Speicherplatz reicht|nicht aus",1,"Bye, Bye", Dummy%
gosub s.system
else
goto r.un
endif
proc s.system
edit
Return
r.un:
```


Einfügen in:

in procedure set.areas zwischen Zeile 34 und 39:

```
if k.ey>0
if (((xx%-xl%)/8)+2)*16)*(yy%-yl%)=<(fre(0)-5000)
```

zwischen Zeile 74 und 75:

```
else
alert 1,"Der Block ist zu groß!|Nicht genügend
Speicherplatz!",1,"ok",Dummy%
endif
endif
```

Ersetzen:

in procedure grafmenue Zeile 61:

```
alert 1,"Funktion nur in Mega-Version|möglich!",1,"ok",Dummy%
```

Fehlende Programmzeilen der Procedure PREDEFs:

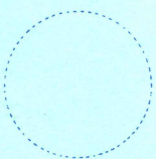
```
dim m_feld$(15),patwr$(16)
restore m_feldtext
for i=0 to 15
read m_feld$(i)
exit if m_feld$(i)="XXX"
next i
m_feld$=""
```

DATA BECKER-Original-Software vollständig dokumentiert! Dieses Superbuch bietet erstmalig einen schnellen, kompetenten und umfassenden Einstieg in das neue Super-BASIC des ATARI ST. Lassen Sie sich von einem DATA BECKER-Profi zeigen, wie man Spitzenprogramme in GfA-BASIC schreibt. Das in diesem Buch dokumentierte Super-Malprogramm enthält Features, die bisher nicht für möglich gehalten wurden. Ohne schwer begreifliche Assemblerroutrinen eintippen zu müssen, bekommen Sie hier Know-how pur. Alles natürlich erläutert und mit vielen Ideen für eigene Erweiterungen. Hier ein Ausschnitt aus den über 100 Features:



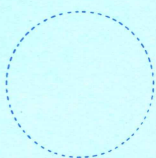
- Blöcke markieren, vergrößern, verkleinern, drehen, spiegeln und invertieren
- Malen mit beliebigen Bildschirmabschnitten
- automatische Anpassung an verfügbaren Speicherplatz
- Spraydose mit 4 Strichstärken
- 6 verschiedene Strichstärken und Stricharten
- X-Ray-Line-Modus
- Spezielle Vertikal- und Horizontal-Line-Funktionen
- 5 Polygon-/Polymarker-Darstellungen
- Horizontales und vertikales Spiegeln
- Volles DIN-A4-Arbeitsblatt ausdruckbar, 2. Grafikseite für Bibliotheken nutzbar
- Optionales Koordinatennetz
- Mustereditor und Malen mit selbsterstellten Mustern
- 36 Standardmuster
- Zoom-Modus
- UNDO-Funktion bis zu vier Schritte rückwärts

Lüers
GfA Painter
ca. 200 Seiten, DM 39,-
ISBN 3-89011-249-8



DM	Pf	für Postgirokonto Nr.
29,-	—	789 - 436

Absender: _____



Für Vermerke des Absenders

Empfängerabschnitt

DM	Pf
29,-	—
für Postgirokonto Nr. 789 - 436	
Absender (mit Postleitzahl)	
Verwendungszweck Diskette z. Buch	
376 222	

Zahlkarte

(Mit Schreibmaschine, Tinte oder Kugelschreiber deutlich ausfüllen)

DM	Pf	(DM-Betrag in Buchstaben wiederholen)
29,-	—	Neunundzwanzig
für DATA BECKER GmbH		
Merowingerstr. 30		
Postgirokonto Nr.		789 - 436
Postleitzahl		
4000 Düsseldorf		
Postvermerk		

Einlieferungschein

— Bitte sorgfältig aufbewahren —

DM	Pf
29,-	—
für DATA BECKER GmbH	
Merowingerstr. 30	
4000 Düsseldorf	
Postgirokonto Nr.	
789 - 436	
Postvermerk	

LADEN, STARTEN - KLAR!

Für alle diejenigen, die sich fleißiges Abtippen ersparen und trotzdem das Graphic Construction Set nutzen wollen, gibt es einen Ausweg:

Mit der nebenstehenden Post-Zahlkarte einfach die Diskette zum Buch bestellen!

Diskette zum Buch
„Das große GfA-BASIC-Buch“
DM 29,-

Einlieferungsschein

(nicht zu Mitteilungen an den Empfänger benutzen)

Gebühr für die Zahlkarte

(wird bei der Einlieferung bar erhoben)

bis 10 DM 90 Pf
über 10 DM (unbeschränkt) 1,50 DM

Bedienen Sie sich der Vorteile
eines eigenen Postgirokontos!
Auskunft hierüber erteilt jedes Postamt

Für Mitteilungen an den Empfänger

Feld
für
postdienstliche
Zwecke

Diskette zum Buch
„Das große GfA-BASIC-Buch“
376 222

DAS STEHT DRIN:

Verlassen Sie sich nicht auf Gerüchte, profitieren Sie lieber vom DATA BECKER-Know-How. Keine nackten Befehlsübersichten, sondern wirklich brauchbares Material liefert dieses Buch in Hülle und Fülle. Dem Autor ist es erstmalig gelungen, ein wirklich spannendes Buch zum neuen GfA-BASIC zu schreiben. So lernen Sie nicht nur schnell und kompetent das GfA-BASIC und die Systemprogrammierung des ATARI ST kennen, sondern erhalten zusätzlich noch ein komplettes GRAPHIC CONSTRUCTION SET, das mit den gebotenen Features garantiert einmalig ist.

Aus dem Inhalt:

- Alle Kommandos des GfA-EDITORS
- Strukturierte Programmierung kontra Spaghetti-Code
- Professionelle Programmerstellung anhand des GCS
- Trickfilm-Modus in Echtzeit
- Patch-Grafik-Design
- 3-D-Animationsgrafik
- Pattern- und Grafik-Editor
- Verzerren, Spiegeln, 3-D-Trommel-Generator
- Clipping, Blockverarbeitung, Netzmodus
- 2 aktive/2 passive Grafikseiten
und vieles mehr

UND GESCHRIEBEN HAT DIESES BUCH:

Uwe Litzkendorf, Student der Architektur, ist begeisterter BASIC-Programmierer und hat durch den ständigen Kontakt mit dem Programmierer des GfA-BASIC die Möglichkeit gehabt, schon in der Entwicklungsphase wirklich alle Features kennenzulernen. Diese Kompetenz ist voll in Buch und Programm eingeflossen.

ISBN 3-89011-222-6

Korrekturblatt zu "Das große GFA-BASIC Buch"

1. Auflage Stand: 07.05.87

Auf Seite 75 des Buches ist ein Beispiel-Listing nicht grau unterlegt. Die Zeilen bis einschließlich 'Edit' gehören zum Programm-Listing. Die Zeile 'Color 1' gehört jedoch nicht dazu. Falls Sie die ersten Zeilen mit abgetippt haben, müssen Sie diese wieder entfernen.

Auf Seite 128 ist ebenfalls ein Beispiel-Listing nicht grau gekennzeichnet. es handelt sich hier um die obersten vier Programmzeilen (Do ... Loop).

Zwei Zeilen der Prozedur Disk.ops sind versehentlich in ein grau unterlegtes Beispiel-Listing geraten. Dies sind die Zeilen:

```
If N.ewname$>""  
If Left$(N.ewname$)<>" "
```

auf Seite 170 oben. Diese beiden Zeilen sind an der entsprechenden Stelle im Listing einzufügen.

Folgende Programmteile sind für die 260er Version zu löschen.

Procedure Cinema	vollständig
Procedure Getpicinits	vollständig
Procedure Get_pics	vollständig
Procedure G.ettinny	vollständig
Procedure P_icload	vollständig
Procedure P_icsave	vollständig
Procedure Showpics	vollständig
Procedure Set.pics	vollständig
Procedure Bildsaver	vollständig
Procedure Bildloader	vollständig

Procedure Grafmenue

Zeile 6 (If Menstartflag%=1) bis incl. Zeile 12 (If Menstartflag%=0)
Zeile 34 (Get 0,0,639...) und Zeile 35 (Get 0,359,639...)
Zeile 37 (Menstartflag%=1) und Zeile 38 (Endif)

Procedure Screenset

Zeile 2 (Zufall%=...) bis einschließlich Zeile 5 (Endif)
Zeile 7 (If Zufall%=0) bis einschließlich Zeile 13 (If Zufall%=1)
Zeile 19 (Endif)

Procedure Set.inits

Zeile 4 (If Initstartflag%=1) bis einschließlich Zeile 10 (If Initstartflag%=0)
Zeile 38 (Dim Initmenu\$(142)) bis einschließlich Zeile 40 (Endif)
Zeile 104 (For I=0 To...) bis einschließlich Zeile 107 (Next I)

Procedure Set.specials

Zeile 5 (If Spec.startflag%=1) bis einschließlich Zeile 8 (If Spec.startflag%=0)