Apple® II GS

X - Basic

Apple II GS

X-Basic

Basic Erweiterung für den Apple II GS

Dokumentation: A.Niederstadt

Stand: Oktober 1987
X-Basic Version: 1.0

Die Software ist nur lauffähig auf Geräten der Apple II GS Serie.

Das Programm X-Basic und das zugehörige Handbuch sind urheberrechtlich geschützt. Das Handbuch und die Diskette dürfen ohne vorherige Genehmigung durch die Autoren weder ganz noch teilweise kopiert oder auf ein anderes Medium übertragen werden.

Das Nutzungsrecht bezieht sich ausschließlich auf den Gebrauch der Software auf oder in Zusammenhang mit nur jeweils einem Computer.

Änderungen an Dokumentation und Programm, die dem softwaretechnischen Fortschritt dienen, sind jederzeit ohne Vorankündigung möglich.

ProDOS ist ein durch das Urheberrecht geschütztes Programm der Firma APPLE Computer Inc., das an die Autoren zur alleinigen Verwendung mit der X-Basic Diskette lizensiert ist. Die APPLE Software darf weder auf eine andere Diskette kopiert, noch ohne das X-Basic in ein elektronisches Medium eingelesen werden.

Das Apple Symbol, Apple, Applesoft, ProDOS und Macintosh sind Warenzeichen der Apple Computer Inc.
SuperSonic ist ein Warenzeichen der MIDIdeas Inc.
Program Writer ist ein Warenzeichen der Firma The Software Touch

Hinweis:

Die X-Basic Diskette ist nicht mit einem Softwareschutz versehen. Die Autoren gestatten die Anfertigung von beliebig vielen Kopien der X-Basic Software ausschließlich zum persönlichen Gebrauch des Käufers.

Inhaltsverzeichnis

Vorwort

Über dieses Handbuch

Dieses Buch enthält eine vollständige Beschreibung der Basic-Erweiterung "X-Basic©" für Ihren Apple II GS. Es soll Ihnen den Umgang mit diesem neuen Zusatz zum bekannten Applesoft©-Basic ermöglichen und Ihnen anhand einiger Beispiele die Fähigkeiten und Einsatzmöglichkeiten dieser Sprache demonstrieren.

X-Basic wird Ihnen damit erstmals einen problemlosen Zugriff auf die neuen Hardwareeigenschaften Ihres Apple II GS ermöglichen und Ihnen so den Zugang zu einer neuen Welt erschließen. Um X-Basic und dieses Buch wirklich in vollem Umfang nutzen zu können, sollten Sie sich bereits mit Applesoft Basic auseinandergesetzt haben und wissen, wie Programme syntaktisch und inhaltlich aufzubauen sind.

Andererseits werden Sie feststellen, daß der Gebrauch von X-Basic sehr einfach ist und Ihnen eine Menge Freude machen wird. Wenn Sie sich mit den hier im Buch befindlichen Kapiteln und speziell mit den aufgelisteten Beispielen befaßt haben, wird es für Sie kein Problem mehr sein, die neuen Möglichkeiten Ihres Apple II GS zu nutzen.

Dieses Handbuch teilt sich in folgende Kapitel ein :

Kapitel 1: Was ist X-Basic ?
Eine allgemeine Übersicht über X-Basic und seine Elemente. Hier werden die Eigenschaften und Befehlsgruppen von X-Basic erläutert.

Kapitel 2: Die X-Basic Befehle
Eine Übersicht über die Funktionen und Auswirkungen der X-Basic Befehle. Was leisten die neuen Befehle?

Kapitel 3: Referenzteil
Referenzteil - hier werden Syntax, Parameter und Arbeitsweise aller Befehle erläutert. Dies ist das für den Programmierer entscheidende Kapitel.

Kapitel 4: Programmieren in X-Basic
Hier befinden sich die "Listings" einiger Beispielprogramme in eine "lesbare" Form gebracht.

Anhang A zeigt alle X-Basic Befehle in alphabetischer Reihenfolge mit einem Seitenindex für den Referenzteil.

Anhang B gibt eine Übersicht aller Dateien, die sich auf der X-Basic Diskette befinden.

Kapitel 1 Was ist X-Basic ?

X-Basic - ein Steckbrief

X-Basic ist ein völlig neues Programiersprachenelement speziell für den Apple II GS. Der Name X-Basic steht dabei für "e**X**tended **Basic**", was eigentlich schon einen Teil seiner Eigenschaften charakterisiert. X-Basic ist nämlich eine Erweiterung des sicherlich schon sehr bekannten und leistungsstarken Applesoft Basic.

Es ist in der Anwendung genauso einfach wie Applesoft, aber es erschließt dem Benutzer die vielfältigen Möglichkeiten des Apple II GS. Besondere Sorgfalt wurde dabei den für diesen Apple II namengebenden Bereichen gewidmet, also **G**rafik und **S**ound.

Ebenso sind natürlich auch andere nützliche Befehle in den neuen Befehlskatalog aufgenommen worden, zum Beispiel solche, die zur Maussteuerung und für den Aufbau von Ein- und Ausgaben nützlich sind. Insgesamt stehen dem Benutzer von X-Basic 54 neue Basicbefehle zur Verfügung, die alle in ganz normale Basicprogramme aufgenommen werden können.

Durch seine spezielle Syntax erlaubt X-Basic teilweise sogar, daß bereits bestehende Programme durch Hinzu-

fügen nur eines einzelnen Zeichens den entsprechenden neuen Befehl ausführen und damit zum Beispiel die hochauflösende Grafik im GS Modus benutzen. Auf diese Art stehen sowohl dem Einsteiger, dem versierten Anwender als auch dem fortgeschrittenen Basic Programmierer fast alle neuen Möglichkeiten des Apple II GS sofort zur Verfügung, ohne daß eine Folge von komplizierten Toolboxaufrufen und die Benutzung eines entsprechenden Entwicklungssystems notwendig sind.

Die sechs Ebenen von X-Basic

X-Basic besteht, wenn man die Befehle in ihren Funktionen so einteilt, wie wir es getan haben, aus sechs Bereichen oder Ebenen. Dabei ist mit dem Begriff Ebene hier keine hierarchische Struktur gemeint, obwohl es auch soche Strukturen sowohl in X-Basic als auch in Applesoft Basic gibt.

Alle hier aufgeführten Ebenen oder Bereiche sind im Prinzip gleichwertig. Wie später gezeigt wird, greifen einige Bereiche dabei jedoch zum Teil ineinander über oder können als Teilbereich eines anderen gelten. Trotzdem - wir haben damit versucht, eine Aufteilung zu finden, die es dem Anwender etwas leichter macht, diese neuen Sprachelemente zu verstehen und zu beherrschen.

Grafik

Der erste und vielen Benutzern vielleicht wichtigste Bereich sind die neuen Grafikbefehle. Dazu stehen insgesamt 17 neue Befehle zur Verfügung, die einen unkomplizierten Zugriff auf die neue Hiresgrafik mit 320 x 200 Punkten und die Superhiresgrafik mit 640 x 200 Punkten des Apple II GS erlauben.
Eine Reihe von Befehlen arbeitet dabei genauso, wie die Grafikbefehle älterer Apple II Modelle, nur eben in den neuen Grafikmodi des Apple II GS. Andere Befehle ermöglichen dagegen grundsätzlich Neues, so z.B. die Erzeugung von Rechtecken und Kreisen durch nur einen Befehl.
Darüber hinaus stehen noch Befehle zum Löschen des Bildschirms, zum Arbeiten mit Sprites - grafischen Symbolen oder Figuren - und zum Speichern und Laden von Bildern zur Verfügung. Mit diesen Befehlen kann jeder Benutzer eine große Vielfalt von grafischen Effekten und Möglichkeiten seines Apple II GS in optimaler Weise nutzen.

Farben

Der nächste Bereich bezieht sich auf den Umgang mit Farben. Hierzu stehen 7 neue Befehle zur Verfügung, die es dem Benutzer ermöglichen, auf die 4096 Farben des Apple II GS zuzugreifen. Außer den Funktionen, die

auch schon von der Grafik und den Farben älterer Apple II Modelle bekannt sind, gibt es einige zusätzliche neue Funktionen, die den Umgang mit der Farbenvielfalt deutlich vereinfachen. Am interessantesten sind dabei diejenigen Befehle, die das Arbeiten mit Farbpaletten ermöglichen und unterstützen.
Weiter bietet X-Basic zwei Möglichkeiten, die bisher einzigartig sind, nämlich die Darstellung von 16 Farben auch in der Superhiresgrafik (640x200 Punkte) und die Benutzung des "Hardware Fillmodus", einer Besonderheit der Apple II GS Hardware zum sehr schnellen Füllen von Grafikflächen. Dies eröffnet dem Benutzer und Programmierer Möglichkeiten von Darstellungen, die bislang von keiner anderen Software und keiner anderen Programmiersprache für den Apple II GS angeboten werden.

Text

Genauso leistungsstark wie die Grafik- und Farbbefehle sind die neuen Textbefehle. Diese Befehle erlauben es erstmals, direkt von Basic aus Texte in der Hires- und Superhiresgrafik darzustellen, ohne vorher sehr umständliche "Shape"-Definitionen (siehe Applesoft Basic) durchzuführen.
Es stehen zwei komplette Zeichensätze zur Verfügung, die beide vom Benutzer oder Programmierer frei umgestaltet werden können. Alle Zeichen werden durch zwei

verschiedene Befehle auf der Grafik angezeigt, die eine Darstellung nach den jeweiligen Erfordernissen und Wünschen des Anwenders erlauben. Damit wird ohne großen Aufwand eine dem Macintosh-ähnliche Darstellung von Texten möglich. Weiter sind natürlich alle Möglichkeiten der Manipulation, die auch vom Applesoft her bekannt sind, in der Apple II GS Grafik zugänglich.

Input

Mit den neuen "Input"-Befehlen stehen jetzt einige sehr einfache Möglichkeiten der Programmsteuerung und der Dateneingabe zur Verfügung. Die Befehle sind so gehalten, daß auch der Einsteiger auf unkomplizierte Weise Text- und Steuereingaben durch Tastatur und Maus programmieren kann.
Damit läßt sich z.B. ein einfaches Zeichenprogramm oder mausgesteuerte Menüauswahl mühelos realisieren, ohne daß eine genaue Kenntniss der sonst notwendigen Maschinenadressen erforderlich ist. Auch hier erlauben die Befehle eine Programmierung ähnlich der des Macintosh, was besonders den Programmierern zugute kommt, die solche Möglichkeiten mit dem Apple II auf Grund der bislang sehr komplizierten Programmierung noch nicht genutzt haben.

Sound

Der Sound - d.h. die akustischen Fähigkeiten - ist eine der wesentlichen Eigenschaften des Apple II GS. Doch bisher war es für den Anwender und den Basicprogrammierer so gut wie nicht möglich, diese Eigenschaften zu nutzen. Mit den Sound Befehlen des X-Basic ist diese Möglichkeit nun gegeben. Dabei können sowohl Klänge von externen Tonquellen benutzt- als auch durch entsprechende Progammsteuerung sämtliche Fähigkeiten des Soundprozessors angesprochen werden.
Es genügt im Extremfall ein Basicprogramm mit nur drei Zeilen, um auf der Tastatur des Apple II GS Klavier oder Orgel zu spielen. Der Sound-Befehl ist zwar einer der komplexesten Befehle des X–Basic und erfordert sicher vom Programmierer einige Übung, aber die Resultate werden jede Anstrengung lohnen. Durch die hohe Geschwindigkeit und die fast unbegrenzten Manipulationsmöglichkeiten der Töne werden selbst professionelle Ansprüche zufriedengestellt.

Sonstige Programmierhilfen

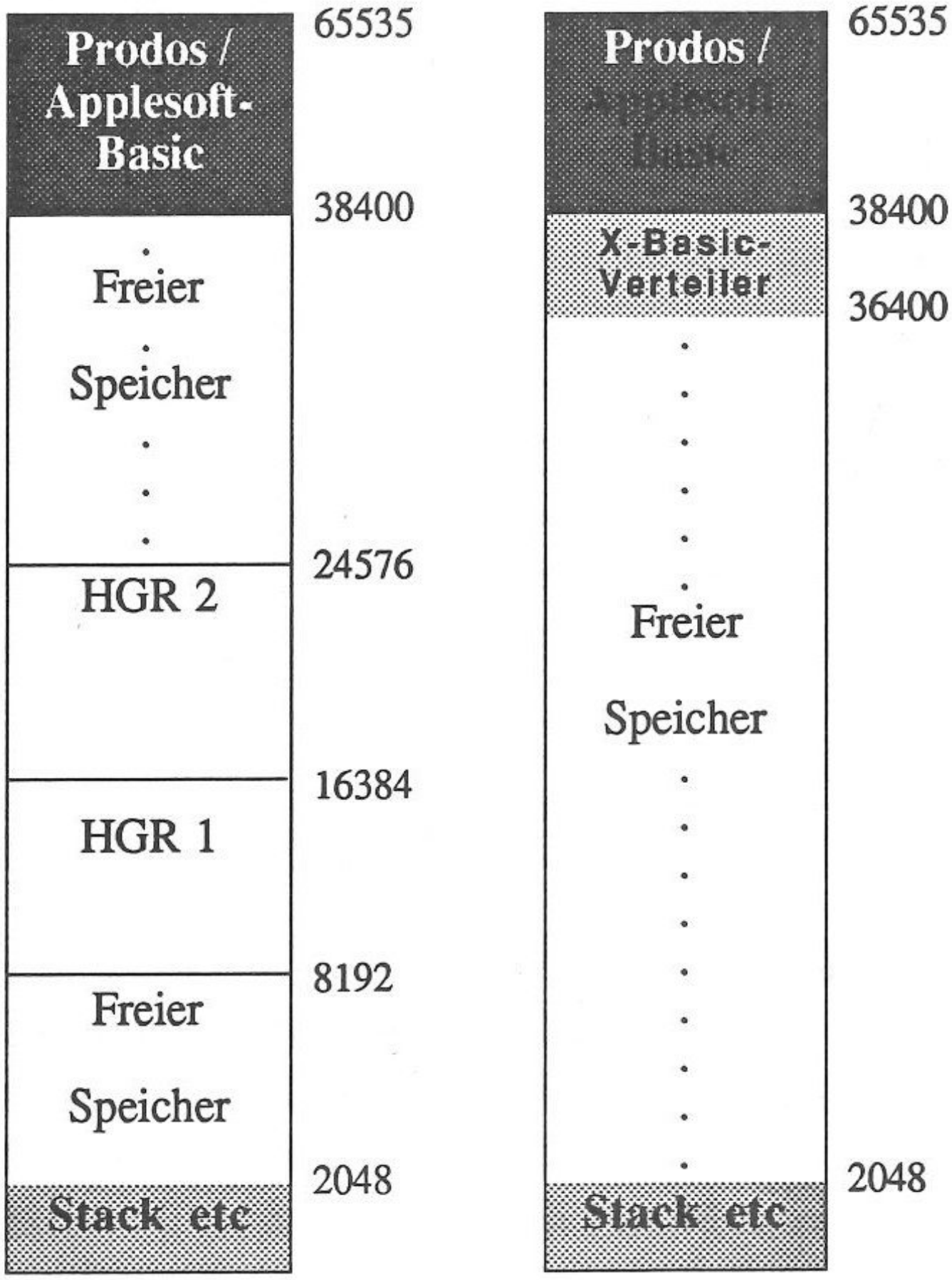
Unter dem Begriff sonstige Programmierhilfen haben wir die Befehle des X-Basic zusammengefaßt, die sich nach unserer Meinung nicht eindeutig einem der vorhergehenden Bereiche zuordnen lassen.
Das soll allerdings keinesfalls heißen, daß diese Befehle

weniger interessant oder leistungsfähig wären als die Anderen. So finden sich hier Befehle, die die Größe von "Fenstern" festlegen können oder die Verwaltung mehrerer Fenster ermöglichen. Außerdem gibt es nützliche Dinge wie Aufruf von Datum und Uhrzeit der eingebauten Uhr oder Aufruf von Monitor-Programmen.
Einer der mächtigsten Befehle des X-Basic ist ebenfalls hier dokumentiert, nämlich der Aufruf von Toolbox Routinen, was dem erfahrenen Programmierer den Zugang zu allen Möglichkeiten der höheren Programmierung gestattet.
Die hier gesammelten Befehle runden die Leistungsfähigkeit von X-Basic ab und stellen allen Anwendern ein komfortables und allen Anforderungen gewachsenes Programmiersystem zur Verfügung.

Wie arbeitet X-Basic ?

X-Basic ist eine vollständig in 16 Bit Assembler geschriebene Erweiterung des bekannten Applesoft Basic. Das hat zur Folge, daß es fast vollständig im 16 Bit Adressraum des Apple II GS liegt und den Arbeitspeicher für Basicprogramme praktisch nicht belegt.
Bisher war es so, daß Programme, die über Grafikausgabe verfügten, nur etwa ein Drittel des gesammten 64 K Speichers benutzen konnten. Aus "historischen" Gründen befand sich nämlich der Bildschirmspeicher für die

Grafik mitten im Arbeitsspeicher des Apple II. Das hatte zur Folge, daß ein Basicprogramm im ungünstigsten Fall nur etwa 5 K Bytè groß werden durfte. Das nachfolgende Diagramm soll diesen Umstand etwas anschaulicher darstellen:



Aufteilung des Arbeitspeichers früher/heute.

Unter Verwendung der neuen Hires- und Superhires-grafik mit X-Basic stehen dem Programmierer nun ca. 32 K Byte ausschließlich für das Basicprogramm zur Verfügung, da der Speicher für die neue Grafik genau wie X-Basic im 16 Bit Adressbereich des Apple II GS liegen.
Wie das Diagramm zeigt, stellt X-Basic dem Programmierer und Anwender daher erheblich mehr durchgehenden Speicher zur Verfügung, als es bisher der Fall war.

Die X-Basic Befehle werden einfach anstelle der alten Applesoft Befehle in das Basicprogramm eingegeben und als solche durch Voranstellen des Applesoftzeichens "&" gekennzeichnet. Im Programmbetrieb werden sie dann über den X-Basic Verteiler weitergeleitet und im 16 Bit-Modus interpretiert. Und damit kommen wir schon zum nächsten Punkt, nämlich :

Wie leistungsfähig ist X-Basic ?

Wie schon erwähnt, ist X-Basic vollständig in 16 Bit Assembler geschrieben und umfaßt mehr als 7500 Programmzeilen. Seine Leistungsfähigkeit demonstriert sich zum einen in seiner Geschwindigkeit und zum anderen in der Vielfalt seines Befehlsvorrats.
Damit wird es erstmals auch Einsteigern und normalen Basicprogrammierern ermöglicht, in vollem Umfang auf

die neuartigen Grafik- und Soundfähigkeiten des Apple II GS zuzugreifen. Außerdem ist X-Basic vollständig kompatibel zum Appelsoft Basic und ermöglicht durch seine in Diagramm 1 dargestellte Speicherplatzbenutzung das Erstellen von wesentlich komplexeren und umfangreicheren Programmen als bisher möglich.
Zusammenfassend läßt sich sagen, daß X-Basic die ideale Erweiterung des bewährten Applesoft Basic für alle Anwender, Einsteiger und Basicprogrammierer ist, die in die neue Welt des Apple II GS, der Grafik und des Sounds einsteigen möchten!

Wir empfehlen übrigends dem Benutzer von X-Basic, seine Programme mit dem "Program Writer" der Firma "The Software Touch" zu schreiben und zu editieren. Dieser mausgesteuerte fullscreen - Editor wird mit dem "&&" Komando aufgerufen und ist zum X-Basic Interpreter vollständig kompatibel.

Kapitel 2 Die X-Basic Befehle

Allgemeine Beschreibung

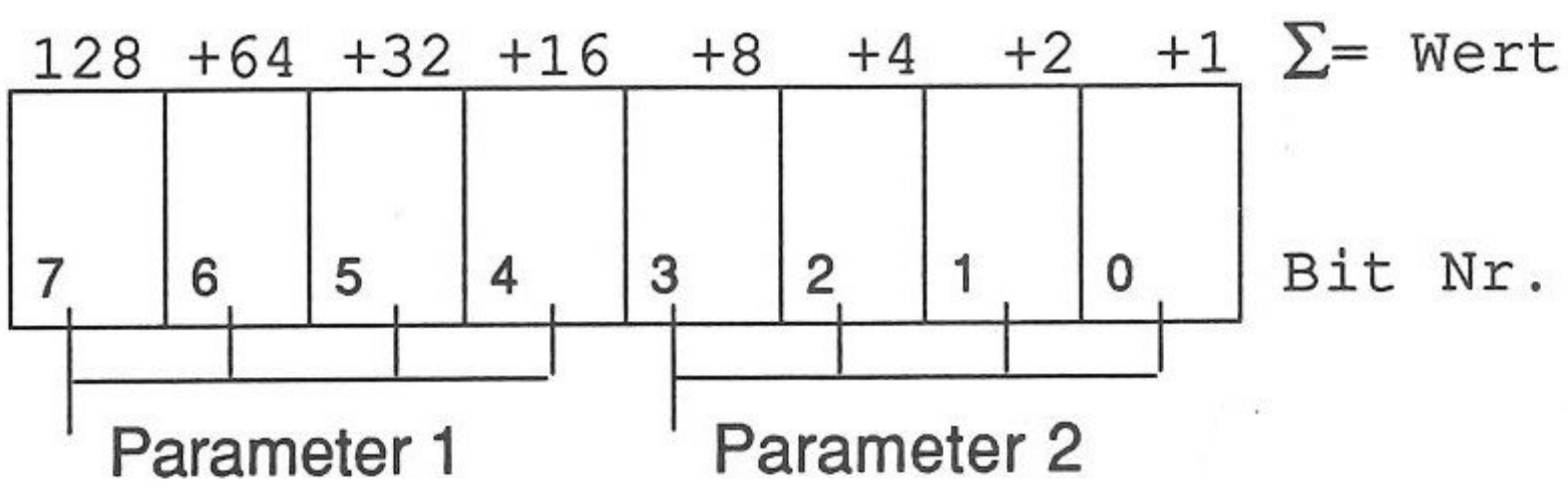
X-Basic besteht aus einer Erweiterung des Appelsoft Befehlsvorrats um 54 Befehle, die das Arbeiten mit den neuen Grafik- und Soundfähigkeiten des Apple II GS ermöglichen. Die Befehle werden genau wie Appelsoft Befehle in den Programmtext eingefügt, nur mit dem Unterschied, daß ihnen das Applesoftzeichen "&" vorangestellt wird.

In der folgenden Beschreibung der X-Basic Befehle ist grundsätzlich zu beachten:

- Angaben in eckigen Klammern [ ] sind optional
- Runde Klammern ( ) gehören zum Befehlssyntax und müssen mit angegeben werden.
- Parameter, die ein "$" Zeichen am Schluß führen, enthalten eine String- (Zeichen) Variable.
- Einige Befehle - nämlich die, die einem Applesoft - Token (Schlüsselwort) entsprechen, benötigen keine Klammern.
- Befehlsparameter in Großbuchstaben enthalten in der Regel (Ausnahmen siehe Referenzteil) eine bitweise Adressierung, die durch ein oder zwei Bytes übergeben werden.

Zur näheren Erläuterung der zusammengesetzten Parameterübergaben soll hier der Aufbau eines Bytes anhand der folgenden Skizze kurz beschrieben werden. Detailliertes Verständnis von Bits und Bytes ist nicht notwendig, für Interressenten sei hier auf entsprechende Fachliteratur verwiesen.

Aufbau eines Bytes :



Ein einzelnes Bit kann nur den Wert 0 oder 1 annehmen; das ganze Byte (8 Bit) kann also Werte zwischen 0 und 255 annehmen. Setzt man die Bits 2 und 3 auf den Wert 1, so ergibt sich der Gesamtwert 12.
Ein Byte kann aber auch zwei Parameter übergeben: Um z.B. dem Parameter 1 den Wert 12 zuzuweisen und dann X-Basic zu übergeben, müssen zunächst die Bits 6 und 7 auf 1 gesetzt und für das Endergebnis mit 16 multipliziert werden.

Die X-Basic Befehle wurden von uns in sechs Gruppen eingeteilt. Dabei haben wir versucht, eine Einteilung nach Funktionen einzuhalten. Jedoch kann man einige Gruppenelemente, d.h. Befehle, ohne weiteres auch als Unter- oder Teilbereich einer anderen Gruppe betrachten. In diesem Kapitel werden die Befehle der einzelnen Gruppen vorgestellt und ihre jeweilige Funktion kurz erläutert.

Die Grafikbefehle

In diesem Abschnitt werden die einzelnen Grafik- und Zeichenbefehle vorgestellt. Die Reihenfolge, in der die Befehle hier aufgeführt sind, ist rein willkürlich und hängt **nicht** mit der Reihenfolge zusammen, in der die Befehle eventuell innerhalb eines Progamms aufgerufen werden müssen. (Informationen dazu im Kapitel 3.)

& GR

Der Befehl GR schaltet den Bildschirm vom Text in den Grafikmodus der GS-Hiresgrafik um. Dabei wird der Bildschirminhalt des Grafikbildschirms nicht verändert. Gleichzeitig werden alle anderen Grafikparameter auf ihre Standardwerte eingestellt und die X-Y-Position des HPLOT-Befehls wird auf 0,0 gesetzt. Ein analoger Effekt konnte in Applesoft

nur durch eine Folge von "Poke" erzeugt werden. (Umkehrbefehl siehe & Text.)

& HGR([-] color)

Der Befehl HGR entspricht in seiner Funktion im Prinzip dem HGR Befehl des Applesoft Basic, mit dem Unterschied, daß es keine zwei Grafikseiten gibt. Die 320x200 Punkte Grafik wird initialisiert und der Bildschirm wird mit der in "color" bezeichneten Farbe gelöscht. Ansonsten sind im HGR Befehl alle Funktionen des GR Befehls enthalten. Eine Besonderheit ist die Angabe einer negativen Farbnummer, die den sogenannten Hardware Fillmodus aktiviert, der im nächsten Kapitel noch genauer erläutert wird.

& HGR2([-] color)

Dieser Befehl initialisiert die Superhiresgrafik mit 640x200 Punkten und löscht den Bildschirm mit der in "color" angegebenen Farbe. Hier besitzt das negative Vorzeichen eine andere Sonderfunktion - es ermöglicht das Arbeiten mit 16 Farben! Ansonsten sind auch hier alle Zusatzfunktion des GR Befehls inbegriffen.

& CLEAR(color)

Der CLEAR Befehl löscht den Grafikbildschirm in der mit "color" angegebenen Farbe. Hier tritt eine Besonderheit von X-Basic auf: Abhängig von dem mit dem MODE Befehl eingestellten Modus wird der Bildschirm initialisiert:

a) Für den 640 Punkte Superhires Betrieb mit 16 Farben

b) Abhängig vom Zustand der ersten Bildschirmzeile im

- 640 Punkte Modus mit 4 Farben
- 320 Punkte Modus mit 16 Farben

& AND

Dieser Befehl schaltet den Grafikbetrieb auf seine Standarteinstellung "AND". Werden mit den Befehlen HPLOT, DRAW, BOX, BOXFILL, CIRCLE, PRINT und SPRITE neue Bildschirminhalte erzeugt, so ersetzen diese den alten Inhalt vollständig in der für sie definierten Farbe. Gegenstück hierzu ist der "OR" Modus.

& OR

Der OR Befehl ist das Gegenstück zum AND Befehl. In diesem Modus werden entsprechend dem

XOR des Applesoft für die Befehle HPLOT, DRAW, BOX, BOXFILL, CIRCLE, PRINT und SPRITE die bereits bestehenden Grafikinhalte berücksichtigt und die neuen Punkte in der jeweiligen Komplementärfarbe gezeichnet.

& HPLOT [TO] x1,y1 **[TO** x2,y2]...

HPLOT zeichnet analog zum Applesoft Basic einen Punkt oder eine Linie zu den angegebenen Koordinaten. Eine Besonderheit liegt hier in dem Umstand, daß die Koordinaten auch außerhalb des Bildschirms oder des Grafikfensters liegen können. Im Gegensatz zu Applesoft treten hierbei keine Doppellinien oder Fehler im Programmablauf auf. Diese Behandlung von Koordinatenangaben gilt auch für alle anderen Zeichenbefehle!

& BOX(links,oben **TO** rechts,unten)

Der Befehl BOX erzeugt ein Rechteck mit den durch "links, oben" und "rechts, unten" definierten Bildschirmkoordinaten. Dabei wird nur der Rahmen in der durch HCOLOR definierten Farbe gezeichnet und der durch das Rechteck umschlossene Raum bleibt unverändert.

& BOXFILL(links,oben **TO** rechts,unten)

BOXFILL arbeitet genau wie BOX, aber mit dem Unterschied, daß der durch das Rechteck umschlossene Raum gleich mit der durch HCOLOR definierten Farbe gefüllt wird. Der dort vorhandene Bereich der Grafik wird also gelöscht.

& BOXPUSH (addr;links,oben **TO** rechts,unten)

BOXPUSH schiebt den durch "links,oben" und "rechts,unten" bezeichneten Rechteckbereich der Grafik komplett in eine Art *Zwischenspeicher* mit der Adresse "addr", von wo er mit dem Befehl BOXPULL wieder abgerufen und in die aktuelle Grafik eingesetzt werden kann.

& BOXPULL (addr;links,oben **TO** rechts,unten)

BOXPULL ist das Gegenstück zu BOXPUSH und kopiert den durch "links,oben" und "rechts,unten" bezeichneten Rechteckbereich des *Zwischenspeichers* mit der Anfangsadresse "addr" komplett in die aktuelle Grafik.
Mit diesen beiden Befehlen lassen sich äußerst interessante Anwendungen gestalten, so z.B. sehr schnelle Einblendungen von Menüs (Macintosh ähnliche Benutzeroberfläche) oder Kopieren und

Vervielfältigen von grafischen Elementen.

& CIRCLE (x,y,a,b,C,D)

Der CIRCLE Befehl ermöglicht die Erzeugung von nahezu beliebigen Kreisfiguren an der Bildschirmposition x,y. Durch "a,b" werden die Ellipsenradien definiert, d.h. wenn a=b erhält man einen Kreis, ansonsten eine Ellipse. Diese kann zusätzlich um ihren Mittelpunkt gedreht werden. Mit dem "C" Parameter kann bestimmt werden, ob eine volle Ellipse bzw. ein voller Kreis oder nur ein Ausschnitt gezeichnet werden soll. Außerdem ist mit CIRCLE die Darstellung von Vielecken möglich!

& FILL (x,y,color)

Der FILL Befehl ist für viele grafische Anwendungen einer der interessantesten. Er füllt jede geometrische Gestalt ausgehend von den Koordinaten x,y mit der durch "color" vorgegebenen Farbe. Dabei wird der Füllvorgang erst abgebrochen, wenn kein freier Punkt innerhalb eines geschlossenen Linienzuges gefunden wird.

& SCRN (x,y,color)

Der SCRN Befehl ermittelt den Farbwert des Bild-

schirmpunktes mit den Koordinaten x,y und speichert diesen zur Weiterverwendung in "color" ab. Damit lässt sich also feststellen, ob Bildpunkte auf eine bestimmte Farbe gesetzt sind (für Animationen), oder es lassen sich sehr schnell Spiegeleffekte erzeugen.

& SPRITE (x,y,def$)

SPRITE zeichnet an der durch x,y bezeichneten Bildschirmposition eine durch "def$" definierte Figur d.h. ein grafisches Symbol (Sprite). Dabei ist "def$" eine genau 129 Zeichen lange Zeichenkette - ein sogenannter String. Dieser String enthält dann die Information für ein aus 16 * 16 Punkten aufgebautes grafisches Symbol, mit dem sich beispielsweise sehr einfach bewegte Grafik darstellen läßt.

& PIC LOAD name$

PIC LOAD lädt ein Hires / Superhires Bild von einem Festspeicher (Festplatte/Diskette) in den Bildschirmspeicher des Apple II GS. Vorlagen können mit X-Basic selbsterstellte Bilder sein oder auch solche, die mit Paintworks oder Deluxe Paint II erstellt wurden.

& PIC SAVE name$

Der Befehl PIC SAVE ist das Gegenstück zu PIC LOAD und speichert dementsprechend den Inhalt des Bildschirmspeichers auf einen externen Festspeicher. Die so gespeicherten Bilder enthalten alle Informationen über Zeilenmodus (320/640 Punkte), Farbmodus und Farbpaletten. Zum Ausdruck und zur Weiterverarbeitung solcher Dateien können Programme wie Paintworks oder Deluxe Paint II benutzt werden.

Die Farben

& COLOR= [-]border,[-] text, [-]hintergrund

Der Befehl COLOR definiert mit den in "border, text, hintergrund" angegebenen Parametern die Farben für den Bildschirmrand, die Textausgabe mit dem PRINT Befehl und die Farbe für den Hintergrund. Eine Besonderheit hierbei ist, daß negative Werte sich auf die Darstellung des PRINT Befehls im Textmodus beziehen und positive Werte den entsprechenden Effekt im Grafikmodus bewirken.

& HCOLOR= color

HCOLOR= ist eine dem Applesoft Befehl nahezu entsprechende Anweisung. Hiermit wird die jeweilige Farbe "color" für die Grafikbefehle HPLOT, DRAW, BOX, BOXFILL und CIRCLE gesetzt. Wenn HCOLOR nicht definiert wird, so werden die vorgenannten Befehle mit der in den Befehlen HGR oder HGR2 definierten Farbe "color" ausgeführt und sind damit praktisch unsichtbar.

& MODE (mode)

Der MODE Befehl ist eine der absoluten Besonderheiten von X-Basic. Er wirkt sich zwar nur in der Superhiresgrafik (640x200 Punkte) aus, aber dort um so wirksamer. MODE erlaubt nämlich die Umschaltung von der normalen Darstellung mit 4 Farben in eine Darstellung gleicher Auflösung mit 16 Farben. Dies ist eine Möglichkeit, die sich bislang nur mit X-Basic realisieren läßt!

& LINEMODE (zeile,mode)

Der Befehl LINEMODE erlaubt noch weitergehende Manipulationen der Grafik als alle vorhergehenden. Er ermöglicht es, jede mit der Position "zeile" gekennzeichnete Grafikzeile abhängig

von der in "mode" enthaltenen Größe entweder im 320 Punkte oder 640 Punkte Modus aufzubauen. Dabei wird ein automatischer Koordinatenausgleich bewirkt, so daß eine gerade Linie, die solche unterschiedlichen Zeilenbereiche kreuzt, auch eine gerade Linie bleibt.
Weiter bewirkt dieser Befehl eine gleichzeitige Farbdefinition der betreffenden Zeile. Die Zeile kann entweder mit einem automatischen Color Fill versehen werden (sehr schnell), oder sie bekommt eine bestimmte Farbpalette zugeordnet.

& PANEL ON

PANEL ON ist ein Befehl, der dem Anwender das Arbeiten mit Farben und Paletten deutlich vereinfacht. PANEL ON blendet dem Benutzer ein Kontrollfeld mit den zur Verfügung stehenden 4 oder 16 Farben auf dem Grafikbildschirm ein. Es befinden sich drei Farbregler für die drei Grundfarben in diesem Kontrollfeld, mit denen sich jeder beliebige Farbton mischen läßt. Dies ermöglicht ein problemloses und komfortables Erstellen von Farbpaletten.

& RESTORE palette, color1...color16

RESTORE lädt die Farbpalette "palette" mit den

16 Farbwerten "color1...color16" aus dem Bildschirmspeicher in die Variablen "color1...color16". Prinzipiell ist es damit möglich, in jeder Bildschirmzeile eine andere Farbpalette (von insgesamt 16) zu benutzen.

& STORE palette, color1...color16

Der Befehl STORE ist das Gegenstück zu RESTORE. Er speichert die Palette "palette" mit den 16 Farbwerten "color1...color16" in den Bildschirmspeicher. Bemerkung dazu: Wird die Grafik gespeichert, so werden auch alle Informationen über die Farbpaletten mitgespeichert (Siehe PIC SAVE).

Die Textbefehle

& PRINT AT (htab,vtab);text$

Der Befehl PRINT AT schreibt den in "text$" enthaltenen Text an der durch "htab,vtab" vorgegebenen Position in die Hires- (ergibt 40 Zeichen/Zeile) oder Superhiresgrafik (ergibt 80 Zeichen). Siehe dazu auch COLOR= und CHARSET. Texte können hier in Tabulatorschritten positioniert werden - die Ausgabe erfolgt sehr schnell.

Der Zeichenhintergrund wird dabei voll überschrieben, AND/OR Modus werden bei diesem Befehl beachtet.

& CHARSET (set)

CHARSET bestimmt den für die Ausgabe verwendeten Zeichensatz. X-Basic stellt zwei Zeichensätze zur Verfügung, "set" kann daher den Wert 0 oder 1 annehmen. Standardzeichensatz entsprechend dem Apple II GS Zeichensatz ist der Satz 0. Dieser wird bei Aufruf der Befehle GR, HGR oder HGR2 automatisch in den Satz 1 kopiert.

& CHARL (ascii,def$)

Der Befehl CHARL lädt die Definition des Zeichens mit dem ASCII Wert "ascii" in die Variable "def$". Das ermöglicht z.B. schnelle Information über den Aufbau eines Zeichens, um anschließend mit CHARS eigene Zeichen zu generieren.

& CHARS (ascii,def$)

CHARS ist das Gegenstück zu CHARL und speichert das in "def$" definierte Zeichen unter

dem ASCII Wert "ascii" in den entsprechenden Zeichensatz. Damit lassen sich leicht eigene Zeichensätze generieren und nahezu alle beliebigen Sonderzeichen darstellen.

& DRAW AT ([#]x,[#]y);text$

DRAW AT ist ein alternativer Befehl zu PRINT AT und zeichnet den Inhalt von "text$" an der mit x,y bezeichneten Position des Bildschirms. DRAW AT ist zwar in der Ausgabe etwas langsammer als PRINT AT, ermöglicht aber z.B. Rotation, inverse Darstellung und beliebige Positionierung. OR/AND Modus wird ebenfalls beachtet. Die Ausgabe erfolgt in der mit HCOLOR= gewählten Farbe. Die Eingabe des Zeichens "#" vor den x,y Koordinaten interpretiert diese als Tabulatorpositionen. Der Hintergrund unter den Zeichen bleibt erhalten!

& ROT= a,b,c

Der Befehl ROT bestimmt die Rotation der Ausgabe von DRAW AT. Standardeinstellung von "a,b,c" ist "1,1,1" und erzeugt eine Ausgabe entsprechend dem PRINT AT Befehl.

Andere Parameter erzeugen Rotation um 90° bzw. um jeweils 180° um die Figurenachsen.

& INVERSE

INVERSE setzt den DRAW Modus auf inverse Darstellung. Das bedeutet, daß mit dem DRAW AT Befehl gezeichneter Text nur den Hintergrundbereich eines Zeichens darstellt. AND oder OR Modus bleiben unbeeinflußt. INVERSE und NORMAL wirken nur auf den DRAW AT Befehl !

& NORMAL

Der NORMAL Befehl ist das Gegenstück zum INVERSE Befehl. Er bewirkt die Zurücknahme des INVERSE Modus, und die Zeichen werden wieder "normal" dargestellt, d.h. es werden wieder nur die Zeichen selbst gezeichnet.

Die Input Befehle - Tastatur und Maus

& GET key

GET arbeitet prinzipiell wie der von Applesoft bekannte GET Befehl. Er hält den Programmablauf an und wartet, bis eine Taste gedrückt wird. Dann

beginnen jedoch schon die Unterschiede: X-Basic GET liest nämlich nicht das Zeichen selbst, sondern dessen ASCII Code und speichert diesen in "key". Außerdem berücksichtigt GET automatisch den Zustand der *Optionstaste* und der *Offenen Apfeltaste*.
Wenn gleichzeitig die *Optionstaste* gedrückt wurde, addiert GET automatisch den Wert 128 zum ASCII Code der gedrückten Taste und wenn die *offfene Apfeltaste* gedrückt wurde, wird 256 zum Tastencode addiert.
Außerdem wird die Bewegung der Maus abgefragt. Für die Mausbewegung gelten die Tastenschlüssel "up, down, left, right, return". Damit läßt sich z.B. schon auf einfache Weise eine brauchbare Maussteuerung in Programme einbauen.

& INKEY (key)

Der INKEY Befehl arbeitet in der Tastaturbehandlung wie der GET Befehl. Im Gegensatz zum GET wird aber der Programmablauf nicht angehalten, und die Mausbewegung wird nicht abgefragt.

& MOUSEINIT (links,oben **TO** rechts,unten;x,y)

MOUSEINIT initialisiert die Maus im durch "links, oben" und "rechts, unten" definierten Fenster und

setzt die Mausposition (Mauszeiger) auf die Koordinaten x,y.

& READ x,y,button

Dieser READ Befehl des X-Basic sollte auf keinen Fall mit dem READ Befehl von Applesoft verwechselt werden. X-Basic READ liest nämlich die jeweilige Mausposition in die Koordinaten x,y ein und fragt den Status der Maus- und Apfeltasten ab. Dabei kann READ vier verschiedene Tastenzustände unterscheiden.
Die unterscheidbaren Zustände sind: Taste nicht gedrückt, Taste gerade gedrückt, Taste gerade losgelassen und Taste wird festgehalten. Damit kann der Programmierer mühelos eine Macintosh-ähnliche Maussteuerung in seinen Programmen konstruieren.

Die Soundbefehle

& SOUNDINIT (number)

Der Befehl SOUNDINIT initialisiert den Sound-DOC (Soundprozessor) des Apple II GS mit der in "number" angegebenen Anzahl von Oszillatoren.

& SOUND (A,B,C,D)

Der SOUND Befehl ist trotz seiner scheinbaren Einfachheit einer der komplexesten Befehle des X–Basic. Das hängt damit zusammen, daß bis auf den Faktor "B" jeder der anderen Faktoren in jedem seiner Bits eine andere bestimmte Information an den Soundprozessor überträgt. So transportiert alleine der Faktor "D" z.B. 7 verschiedene Informationen zum SoundDOC. Die Codierung der Parameter erfordert daher einige Übung und eine gründliche Durcharbeitung des Referenzteils. Andererseits erlaubt der Soundbefehl auch, schon mit einem "Dreizeiler" auf der Tastatur des Apple II GS Klavier oder Orgel zu spielen.

& DIGI (slot,M)

Der DIGI Befehl ist nur in Verbindung mit einer SuperSonic Digitizer Karte anwendbar. DIGI digitalisiert dann eine 32 K Byte Tonfolge und speichert diese in die Superhiresgrafik. Dabei gibt "slot" den Slot an, in den die Digitizerkarte eingebaut wurde. Der Parameter "M" legt die Dauer des Digitalisierens sowie die Ansteuerung fest: Auf Tastendruck oder durch Signaleingang.

& WAVE (page,name$)

WAVE lädt eine digitalisierte Tonfolge von einem Festspeicher in den mit "page" bezeichneten Bereich des SoundRAM. Dabei enthält "name$" den kompletten Pfadnamen der betreffenden Datei.

& SOUND END

Die Anweisung SOUND END schaltet den Sound-DOC (Soundprozessor) des Apple II GS wieder ab. Dieser Befehl sollte immer verwendet werden, um Störungen im weiteren Betrieb zu vermeiden.

Die sonstigen Programmierhilfen

& * [(mon$)]

Die Anweisung * ohne optionalen Parameter "mon$" ist im Prinzip gleichwertig mit der Applesoft Anweisung *CALL-151*. Mit dem Parameter "mon$" wird der in "mon$" enthaltene Monitor-befehl (siehe Apple II Referenz-Manual) sofort ausgeführt und anschließend wieder ins Basic zurückgesprungen.

& TEXT

TEXT ist das eigentliche Gegenstück zum GR Befehl und schaltet vom Grafikbildschirm auf den Textbildschirm um.

& PUSH (page,len)

Der Befehl PUSH kopiert z.B. eine komplette Grafikseite von 32 K Byte Größe in den Sound-RAM. Die Größe "page" gibt dabei den Startpunkt im SoundRAM an und "len" die Länge des zu kopierenden Bereichs. Das Gegenstück zu PUSH ist der Befehl PULL.

& PULL (page,len)

PULL kopiert z.B. den Inhalt von 32 K Byte Sound-RAM in die Hiresgrafik. ("page" und "len" siehe PUSH!) Die Befehle PUSH und PULL eignen sich hervorragend, um Bilder, die zwischenzeitlich manipuliert werden sollen, kurzfristig wiederherzustellen, oder grundsätzlich, um mit mehreren Grafikfenstern gleichzeitig arbeiten zu können.

& PEEK bank,addr,cont$

Der PEEK Befehl ist prinzipiell mit dem PEEK

Befehl des Applesoft vergleichbar. Allerdings kann der X-Basic PEEK gleich einen ganzen String "cont$" von 255 Zeichen Länge aus einer beliebigen Speicherbank "bank" von der Adresse "addr" holen.

& POKE bank,addr,cont$

POKE ist das Gegenstück zu PEEK und erlaubt das Ablegen eines ganzen Strings von 255 Zeichen Länge im Speicher mit der Adresse "addr" in der Bank "bank".

& GWINDOW (links,oben **TO** rechts,unten)

Die Anweisung GWINDOW definiert den Rechteckbereich, der durch "links,oben" und "rechts,unten" gegeben ist, als Fenster für die Grafikausgabe. Die Zeichenbefehle HPLOT, DRAW, BOX, BOXFILL, SCRN und CIRCLE wirken nur in diesem Bereich.

& TWINDOW (htabl,vtabo **TO** htabr,vtabu)

TWINDOW bestimmt die Größe des Ausgabefensters im Textmodus. Diese Funktion mußte in Applesoft durch eine Folge von POKE aufgerufen werden und ist hier deutlich vereinfacht.

& CALL ToolBox, A, [b], [c],...

Der Befehl CALL erlaubt den direkten Aufruf von ToolBox Routinen und damit Zugriff auf die ganze Vielfalt und die Geschwindigkeit der Apple II GS ROM- und Systemroutinen. Allerdings ist hier auch die genaue Kenntnis der Apple II GS Toolbox und ihrer Aufrufe notwendig, was in aller Regel dem versierteren Programmierer vorbehalten ist.
Detaillierte Informationen dazu sind den entsprechenden Referenzmanuals über die Apple II GS Toolbox zu entnehmen. (z.B. Addison Wesley Verlag)

& TIME (monat,tag,jahr;stunde,minute)

TIME spricht die im Apple II GS eingebaute Echtzeituhr an und überträgt das aktuelle Datum und die aktuelle Uhrzeit in die entsprechenden Variablen. Diese Möglichkeit ist für alle Programme interessant, in denen Zeit eine Rolle spielt, angefangen bei Spielen bis zu Rechnungen.

& VER= number

Die VER= Anweisung zeigt dem Anwender die jeweilige Versionsnummer seines X-Basic. Dies

dient zur Kontrolle, um dem Anwender zu gewährleisten, daß er immer mit der jeweils aktuellen Version dieses Produkts arbeitet.

& ERR (line, berror, terror)

Die ERR Anweisung lädt im Falle eines Programmfehlers die Zeilennummer, den Fehlercode und bei Toolboxzugriffen den Toolboxfehlercode in die Parameter "line, berror, terror", was dem Anwender die Fehlerbeseitigung erheblich vereinfacht.

& END

END beendet die Arbeit mit X-Basic und entfernt das Basic vollständig aus dem Speicher. Dieser Befehl sollte angewandt werden, um Fehler in anderen Programmen - die möglicherweise nicht ausreichend abgesichert sind - zu vermeiden.

Kapitel 3 Referenzteil

Die X-Basic Parameter

Allgemeine Beschreibung

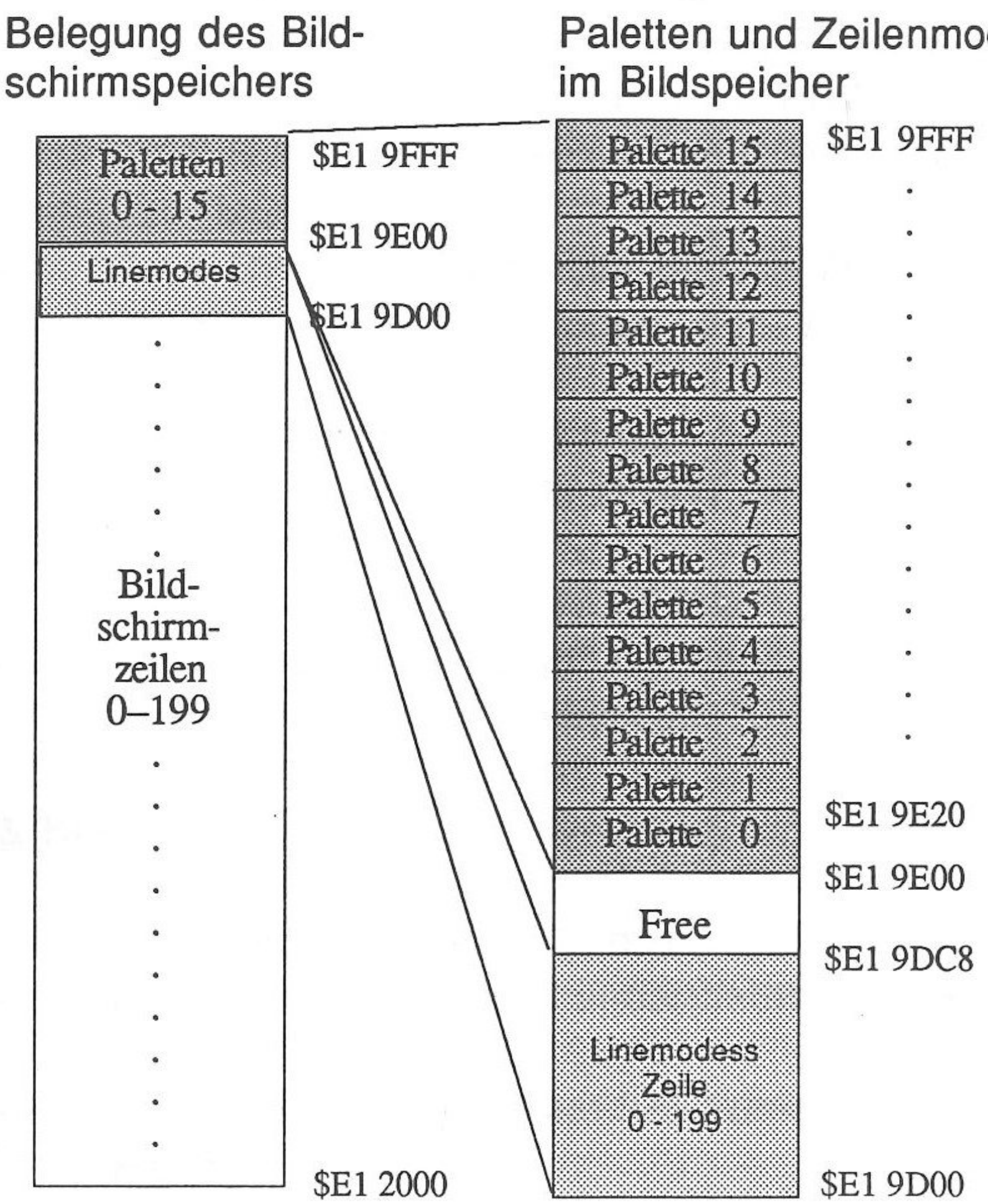
Dieses Kapitel soll dem X-Basic Benutzer eine möglichst genaue Darstellung der einzelnen Befehle geben. Damit ist nicht gemeint, wie sie sich im Programm nach außen auswirken, sondern was die Befehle im Apple II GS intern bewirken. Das heißt, die Parameter - ob nun für den Anwender sichtbar oder nicht - sollen hier erläutert werden.

Dazu einige Vorbemerkungen: Grundsätzlich sind für alle vom Benutzer definierbaren Parameter Zahlenwerte zwischen -32768 bis + 65535 möglich. Diese werden von X-Basic automatisch in die entsprechenden Modulo Zahlen umgerechnet. Dezimalstellen werden bei entsprechenden Befehlen ignoriert.

Anweisungen, die String-Variable enthalten, sollten vom Anwender nicht im Direktmodus angewandt werden, da diese den Tastaturpuffer des Apple II GS blockieren können. Im Programmbetrieb tritt dieser Effekt nicht auf!

Der Zugriff auf RAM-Disk Slot 3, Drive 2 ist mit X-Basic nicht mehr möglich, da der Speicherplatz belegt ist.

Zum besseren Verständnis der Arbeitsweise von X-Basic soll das folgende Diagramm den Aufbau des Bildschirmspeichers darstellen:



Die Grafik

& GR

- Initialisiert X-Basic für den Grafikbetrieb
- Schaltet auf Hires / Superhires Grafik um
- Setzt die Rotationsparameter auf 1,1,1
- Setzt Bildschirmkoordinaten für HPLOT auf 0,0
- Setzt den COLOR Parameter für PRINT AT auf x,0,0 - x hat keine Auswirkung im Grafikbetrieb
- Setzt MODE auf 0
- Setzt GWINDOW auf volle Bildschirmgröße, abhängig vom Zeilenmodus der ersten Zeile (320/640 Punkte)
- Setzt Zeichenmodus auf AND
- Schaltet für PRINT und DRAW Befehle auf den Zeichensatz 0 um
- Setzt DRAW Modus auf NORMAL
- Zeichensatz 0 wird in Zeichensatz 1 kopiert (0 = Apple Standartzeichensatz "US")

& HGR([-] color)

- Alle Funktionen wie GR und zusätzlich:
- Kopiert die X-Basic Standardfarbpalette in die Palette 0
- Setzt alle Grafikzeilen auf 320 Punkte Modus und Palette 0

- Löscht den ganzen Bildschirm mit der in "color" angegebenen Farbe, für die gilt :
  "color" positiv (0...15) => Normalbetrieb
  "color" negativ (-16,-0) => Hardware Fillmodus (Hardware Fillmodus bedeutet, daß jeweils nur der erste Punkt einer Zeile gesetzt wird und die Hardware den Rest der Zeile bis zum nächsten, explizit gesetzten Punkt mit der Farbe "color" gefüllt wird. Sehr schnell !)
- Setzt HCOLOR auf "color"

& HGR2([-] color)

- Alle Funktion wie GR und zusätzlich :
- Kopiert die X-Basic Standardfarbpalette in die Palette 0
- Setzt alle Grafikzeilen auf 640 Punkte Modus und Palette 0
- Löscht den ganzen Bildschirm mit der in "color" angegebenen Farbe, für die gilt :
  "color" positiv (0...3) => Normalbetrieb
  "color" negativ (-16...-1) => 16 Farben möglich und MODE wird auf 1gesetzt (-16 = - 0!)
- Setzt HCOLOR auf "color"

& CLEAR(color)

- Löscht den Bildschirm mit der Farbe "color" abhängig von MODE, d.h. :
  Wenn MODE = 1, dann 640 Punkte, 16 Farben, wenn MODE = 0, dann 320 Punkte - 16 Farben / 640 Punkte - 4 Farben, abhängig vom Status der ersten Bildschirmzeile, bzw "color" = 0, wenn Hardware Fillmodus.

& AND

- Schaltet auf "AND", (wirksam bei HPLOT, DRAW, BOX, BOXFILL, CIRCLE, PRINT und SPRITE). Gegenstück hierzu ist der "OR" Modus.

& OR

- Schaltet in den OR (XOR) Modus um - beeinflußt die gleichen Befehle wie AND.

& HPLOT [TO] x1,y1 **[TO** x2,y2]...

- Zeichnet Punkte (ohne TO) oder Linien an den Koordinaten x_n,y_n, mit x,y = -32768...65535 und n = Anzahl der Bildpunkte 1...n.

& BOX(links,oben **TO** rechts,unten)

- Zeichnet ein Rechteck mit den Eckpunktkoordinaten "links,oben" und "rechts,unten".

& BOXFILL(links,oben **TO** rechts,unten)

- Befehl wie BOX, nur das Rechteck wird automatisch in der mit HCOLOR= gewählten Farbe gefüllt.

& BOXPUSH (addr;links,oben **TO** rechts,unten)

- Schiebt den durch die Koordinaten "links,oben" und "rechts,unten" bezeichneten Bereich des Bildschirms an den mit "addr" bezeichneten Platz im SoundRAM. Die Wahl der Koordinaten "links, oben" und "rechts,unten" ist anlog zum HPLOT etc. "addr" = 0...65535 entsprechend der tatsächlichen Anfangsadresse im SoundRAM

& BOXPULL (addr;links,oben **TO** rechts,unten)

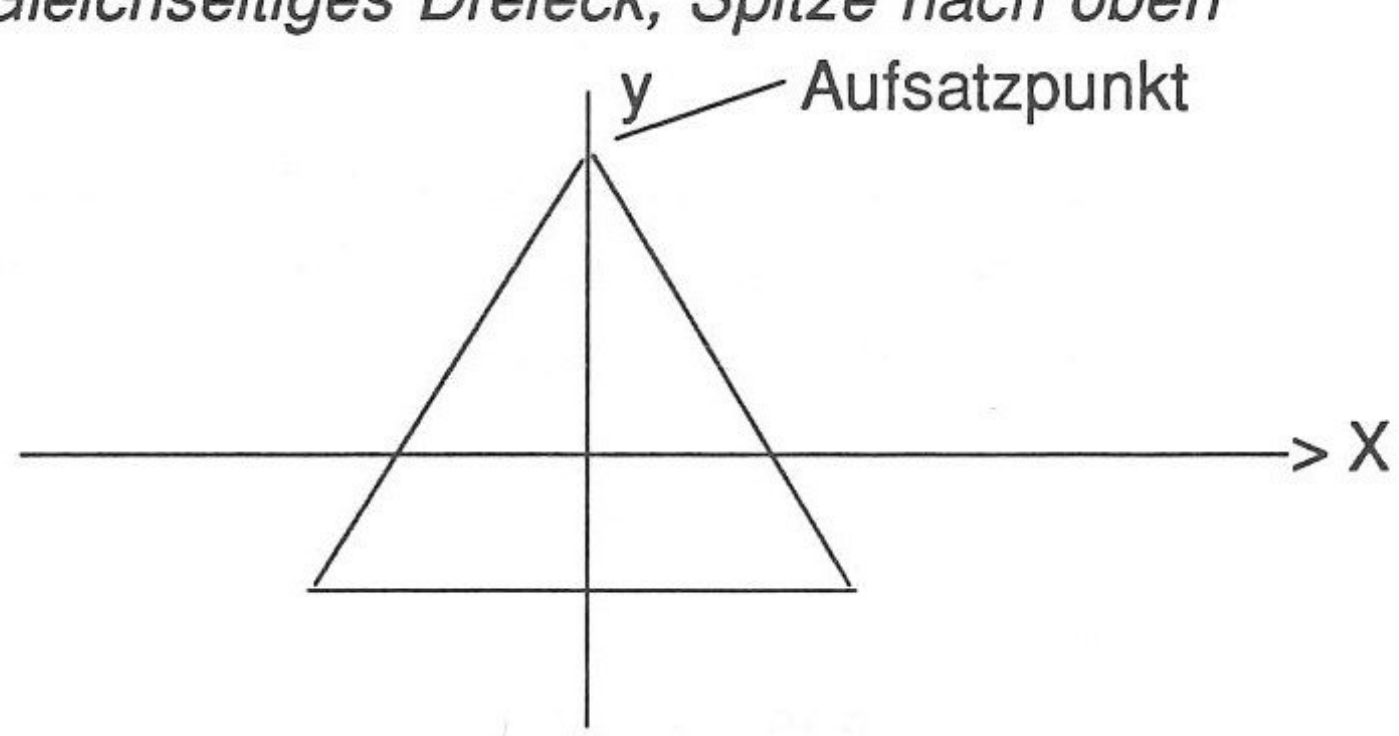
- Gegenstück zum BOXPUSH; holt den durch die Koordinaten "links,oben" und "rechts,unten" bezeichneten Bereich in die aktuelle Grafik. "addr" wie bei BOXPUSH. Wenn "links,oben" und "rechts,unten" vom vorhergehenden BOXPUSH abweichend angegeben werden, können damit

Bildschirmbereiche verschoben werden.

& CIRCLE (x,y,a,b,C,D)

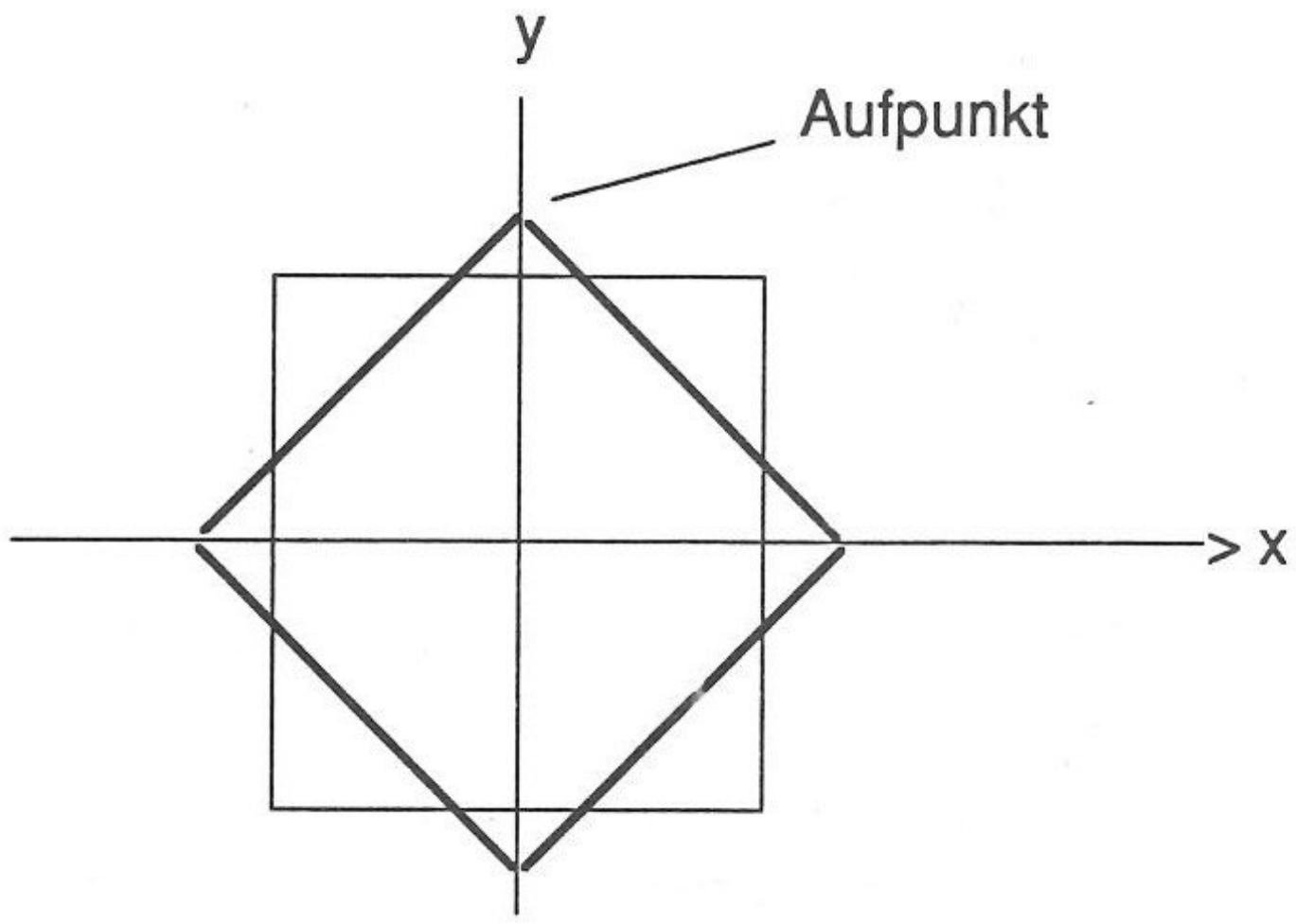
- x,y Koordinaten des Kreis-/Ellipsen-/Vieleckmittelpunktes.
- a = Große Halbachse, b = Kleine Halbachse; a = b für Kreis oder regelmäßiges Vieleck
- C = ta*256+te, C ist ein 2 Byte großer Parameter; erstes Byte "ta*256" gibt den Anfangspunkt und "te" den Endpunkt an; d.h. nach der wievielten Teilung das Zeichnen beendet werden soll.
- D = dr*256+t, D ist ebenfalls ein 2 Byte großer Parameter. Das erste Byte "dr*256" gibt die Drehung des Anfangspunktes, und damit die Gesamtrotation des Körpers an; das zweite Byte "t" gibt die Anzahl der Teilungen an.
- Beispiele :

Gleichseitiges Dreieck, Spitze nach oben



& CIRCLE (x, y, 50, 50, 0*256+3, 0*256+3)

d.h. an Position x,y wird ein regelmäßiges (50=50!) Vieleck mit Startpunkt 0 über 3 Teilungen gezeichnet, die Rotation des Körpers ist 0 und er besteht aus drei "Eckpunkten".
Bemerkung : die Parameter ta,te,dr,t sollten sich im Bereich zwischen 0...255 bewegen! Für einen Vollkreis / eine Vollellipse sind "te" und "t" mit 0 (oder ab ca. 50) zu wählen.
Regelmäßiges Viereck, Spitze nach oben



& CIRCLE (x, y, 50, 50, 0*256+4, 0*256+4)
Befehl wie Dreieck nur jetzt 4 Teilungen. Soll das Viereck um 90° gedreht werden (dünne Linien), sieht der Befehl so aus :
& CIRCLE (x, y, 50, 50, 0*256+3, 64*256+3)
64 ist ein Viertel von 256 entsprechend 90°!
Weitere Beispiele in Kapitel 4.

& FILL (x,y,color)

- Füllt einen geschlossenen Linienzug (auch Hintergrund) mit der Farbe "color" beginnend an der Position x,y. Der Wert von HCOLOR bleibt unverändert.

& SCRN (x,y,color)

- Ermittelt den Farbwert des Bildpunktes x,y und überträgt diesen in "color".
- "color" = 0...15 im 320 Punkte Modus
- "color" = 0...3 im 640 Punkte Modus

& SPRITE (x,y,def$)

- Zeichnet die durch "def$" definierte Figur an Position x,y; "def$" ist ein 129 Zeichen langer String, der linear alle Farbdefinitionen enthält, die für ein 16*16 großes Raster benötigt werden.
  1 Zeichen = 1 Byte = 2 Bildpunkte. Byte 0 enthält den Farbwert für Hintergrund im OR Modus!
  Beispiel : Hintergrundfarbe = 0
  Farbwert des ersten Bildpunktes = 5
  Farbwert des zweiten Bildpunktes = 10
  Farbwert des dritten Bildpunktes = 12
  Farbwert des vierten Bildpunktes = 7
  dann sieht "def$" wie folgt aus :

def$ = CHR$(255) + CHR$(5*16+10) + CHR$ (12*16+7)... usw. bis 129 Zeichen voll sind. SPRITE nicht im Direktmodus benutzen!

& PIC LOAD name$

- PIC LOAD lädt ein komplettes Hires-/Superhires-bild (incl. Linemodes und Paletten) vom Massen-speicher. Wenn nur der Bildname in "name$" geschrieben wird, muß ein PREFIX gesetzt sein. Nach Möglichkeit nicht im Direktmodus verwenden oder wenn, dann keine Variable für name$ son-dern explizit den ganzen Pfadnamen angeben!

& PIC SAVE name$

- Gegenstück zu PIC LOAD, speichert komplettes Hires-/Superhiresbild (incl. Linemodes und Palet-ten) auf Massenspeicher. Wenn nur der Bildname in "name$" geschrieben wird, muß ein PREFIX gesetzt sein. Bilder können mit Paintworks oder DeluxePaint weiterbearbeitet werden. Befehl nach Möglichkeit nicht im Direktmodus verwenden oder wenn, dann keine Variable für name$ sondern explizit den ganzen Pfadnamen angeben!

Die Farben

& COLOR= [-]border, [-] text, [-]hintergrund

- [-]border, [-] text, [-] hintergrund positiv => Farben für den Rand, Text und Hintergrund im Textmodus
- [-]border, [-] text, [-] hintergrund negativ => Farben für den Grafik PRINT AT Befehl, "border" wird ignoriert!

& HCOLOR= color

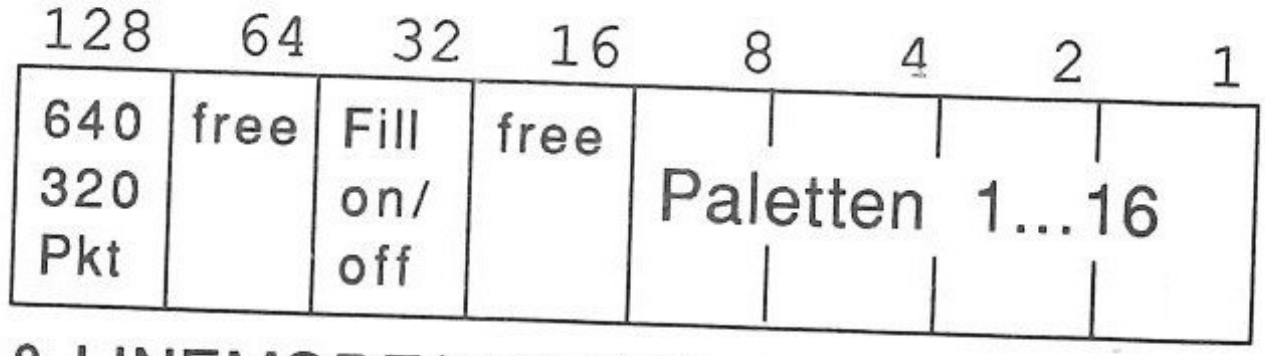
- HCOLOR= setzt Farben für HPLOT, DRAW, BOX, BOXFILL und CIRCLE. Wenn HCOLOR nicht definiert wird, so werden die vorgenannten Befehle mit der in den Befehlen HGR oder HGR2 definierten Farbe "color" ausgeführt und sind damit praktisch unsichtbar.

& MODE (mode)

- Schaltet den Zeichenmodus der 640 Punkte Grafik um:
- "mode"=1 => 640 Punkte, 16 Farben (Achtung: Grafikbefehle werden wie im 320 Punkte Modus behandelt!)
- "mode"=0 => 640 Punkte, 4 Farben

& LINEMODE (ypos,mode)

- Definiert die durch "ypos" angegebene Zeile im Zeilenmodus "mode"
  Beispiel : Bildzeile 100, 640 Punkte, Palette 3

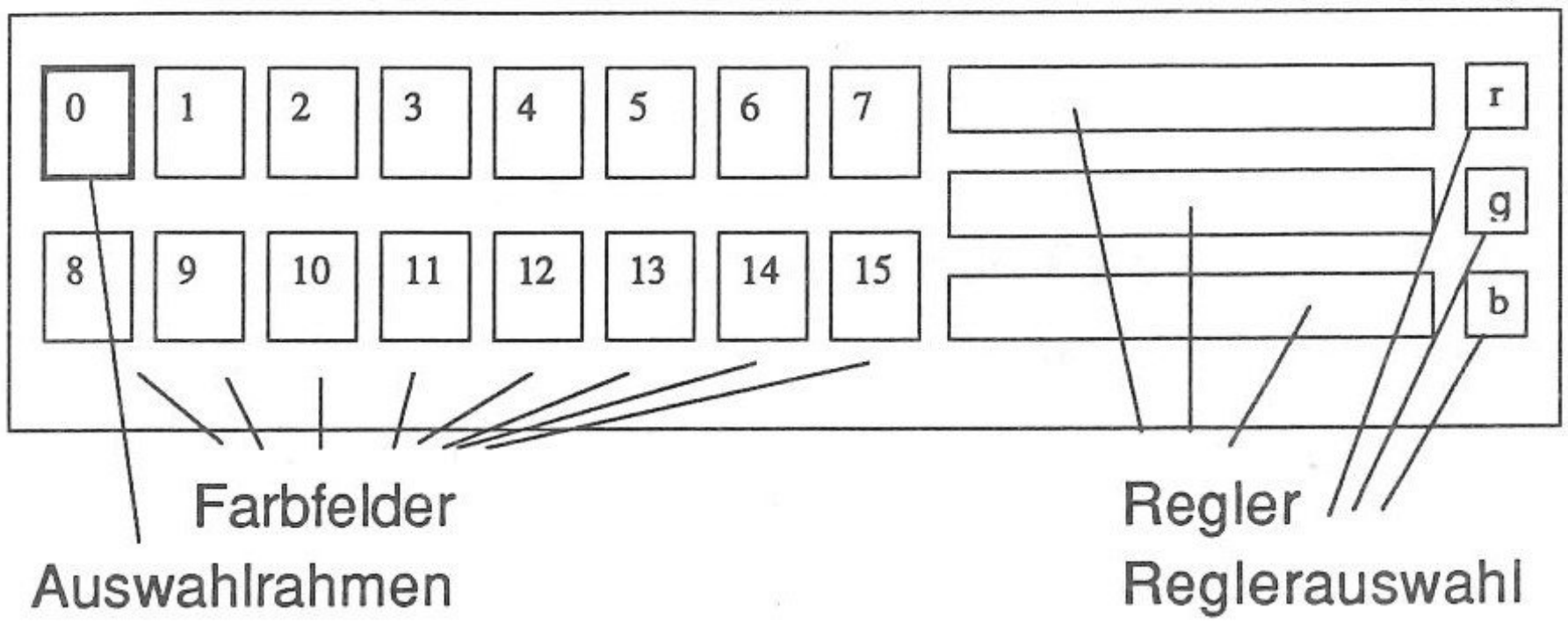


  & LINEMODE(100,{128+0+3})
- Bildzeile 149, 320 Punkte, Fill on, Palette 5
  & LINEMODE(149,{0+32+5})
- Hardware-Fill nur im 320 Punkte Modus möglich!

& PANEL ON

- PANEL ON blendet dem Benutzer ein Kontrollfeld mit den zur Verfügung stehenden 4 oder 16 Farben auf dem Bildschirm ein. Außerdem befinden sich drei Farbregler für die drei Grundfarben in diesem Kontrollfeld, mit denen sich jede beliebige Farbe aus 4096 möglichen mischen läßt. Das folgende Diagramm zeigt das auf dem Grafikbildschirm eingeblendete Kontrollfeld.

Diagramm: Farbpanel



Bedienung des Panels:

- Zahlentasten 1...9 positionieren das Panel auf dem Bildschirm. 1=> ganz oben; 9 => ganz unten.
- "esc" beendet PANEL ON ohne Übernahme der Änderungen.
- "return" beendet PANEL ON mit Übernahme der Änderungen.
- Pfeiltasten ↑, ↓ wählen die drei Farbregler aus (r=rot, g=grün, b=blau)
- Pfeiltasten ←,→ wählen die Farben dunkler / heller (16 Stufen)
- ♂ - Taste + ↑, ↓,←,→ wählen mit dem roten Auswahlrahmen die einzustellende Farbe
- Option Taste + ↑, ↓,←,→ wählen mit dem grünen Auswahlrahmen die einzustellende Palette(0...15).

Achtung! Paletten 14 und 15 werden von PANEL ON benötigt!

Funktion der "space" Taste (Leertaste):

- Erster Tastendruck kopiert den Farbwert in den "Zwischenspeicher" und markiert diese Farbe mit einem blauen Rahmen.
- Zweiter Tastendruck setzt den Farbwert vom "Zwischenspeicher" in die mit Pfeiltasten ausgewählte Farbnummer und Palette ein; der blaue Markierungsrahmen wird wieder gelöscht.
- Alle Cursortastenfunktionen können auch mit der Maus bedient werden.

& RESTORE palette, color1...color16

- RESTORE holt die 16 Farbwerte der Palette "palette" (palette = 0...15) in die Variablen "color1...color16".

& STORE palette, color1...color16

- Speichert die 16 Farbwerte "color1...color16" in die durch "palette" angegebene Palette.
- "color" ist eine Mischfarbe aus rot(0...15), grün(0...15) und blau(0...15)
- Beispiel : Mit rot =7, grün = 2 und blau = 14 erhält man den Farbwert der Mischfarbe "color" durch Addition der Komponenten nach folgendem Verfahren:
  color = *7*256* + *2*16* + 14**1*.

Achtung ! Es müssen alle 16 Fabwertparameter "color1...color16" definiert sein, um zu vernünftigen Ergebnissen zu kommen. (Sonst Syntaxerror)

Die Textbefehle

& PRINT AT (htab,vtab);text$

- Schreibt die in "text$" enthaltenen Zeichen in die Grafik (siehe auch CHARSET und COLOR=)
- Jedes Zeichen ist dabei 8*8 Punkte groß.
- Schreibt im Modus der ersten Bildschirmzeile (320/640 Punkte =40/80 Zeichen pro Zeile), in der die Ausgabe beginnt. Alle weiteren Zeilenmodi werden beim jeweiligen PRINT ignoriert.
- AND und OR Modus werden berücksichtigt.
- Farbauswahl nur mit COLOR= möglich.
- Möglichst nicht im Direktmodus verwenden!

& CHARSET (set)

- "set"=0 wählt den Zeichensatz 0 aus
- "set"=1 wählt den Zeichensatz 1 aus
- Die Befehle GR, HGR und HGR2 kopieren den Zeichensatz 0 in den Zeichensatz 1. Zeichensatz 0 entspricht beim Start von X-Basic dem Standard-

zeichensatz des Apple II GS (US-Zeichensatz)

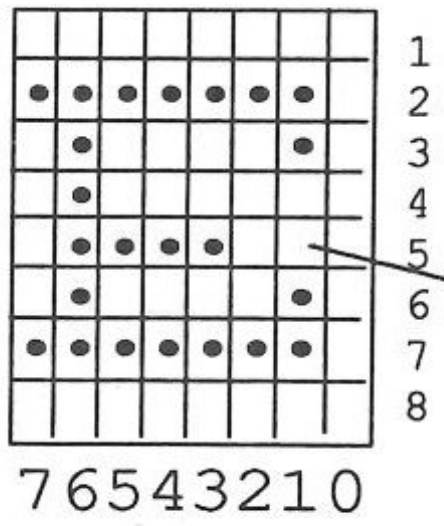
& CHARL (ascii,def$)

- Holt die Definition des Zeichens mit dem ASCII Code "ascii" in die Variable "def$".
- Zeichensatz 0: ascii = 0...127
- Zeichensatz 1: ascii = 128...255
- Nicht im Direktmodus verwenden!

& CHARS (ascii,def$)

- Definiert das Zeichen mit dem ASCII Code "ascii" entsprechend "def$" und lädt es in den entsprechenden Zeichensatz.
- Zeichensatz 0: ascii = 0...127
- Zeichensatz 1: ascii = 128...255
- Nicht im Direktmodus verwenden!
- "def$" ist ein 8 Zeichen langer String, bei dem jedes Bit linear einem gesetzten oder nicht gesetzten Punkt eines Zeichens aus einer 8*8 Matrix entspricht.

Beispiel : Buchstabe "E"



8*8 Bitmatrix : Jede Zeile ein Zeichen aus 8 Bit

```
128  64  32  16  8  4  2  1
  0   1   1   1  1  0  0  0
```

Zeichen für obiges Muster in Zeile 5 = CHR$ (64+32+16+8)

=> def$ = CHR$(0) + CHR$(254) + CHR$(66) + CHR$(64) + CHR$(120) + CHR$(66) + CHR$(254) + CHR$(0)

- Nicht im Direktmodus verwenden!

& DRAW AT ([#]x,[#]y);text$

- Zeichnet den Text in "text$" an der Position x,y mit der in HCOLOR= definierten Farbe.
- # vor Positionsangaben x,y interpretiert x,y als Tabulatoren (Tabulatorschritt entspricht 8 Punkten)
- AND und OR Modus möglich
- Zeilenmodi werden immer beachtet.
- Weiteres siehe ROT, INVERSE, NORMAL

& ROT= a,b,c

- Bestimmt die Rotation der Ausgabe mit DRAW AT (Standard = 1,1,1)
- Wertebereich für a,b,c: -1 oder 1

Andere Werte werden Modulo umgerechnet.

- a =-1 => Rotation um180° horizontal nach links
- b =-1 => Rotation um 180° vertikal nach oben
- c =-1 => Rotation um 90° nach oben + 180° nach unten
- alle Rotationen können überlagert, d.h. gleichzeitig ausgeführt werden.

& INVERSE

- Setzt den DRAW Modus auf inverse Darstellung. Nur der Hintergrund wird gezeichnet.
- AND oder OR Modus bleiben unbeeinflußt.
- INVERSE und NORMAL wirken nur auf den DRAW AT Befehl !

& NORMAL

- Setzt DRAW AT auf normale Darstellung zurück.

Die Input Befehle - Tastatur und Maus

& GET key

- Wartet bis Taste gedrückt wird (Programm *steht*) und holt den ASCII Code in "key"

- Taste + Option Taste => key = key + 128
- Taste + ć - Taste => key = key + 256
- Mausabfrage :

  Bewegung links => key = 8
  Bewegung rechts => key = 21
  Bewegung oben => key = 11
  Bewegung unten => key = 10
  Maustaste => key = 13

& INKEY (key)

- Funktion wie GET, jedoch ohne Mausabfrage und ohne Warten auf Tastatureingabe (Programm läuft weiter)
- Ohne Tastendruck => key = -1

& MOUSEINIT (links,oben **TO** rechts,unten;x,y)

- Initialisiert die Maus im definierten Fenster und setzt die Mausposition (Mauszeiger) auf die Koordinaten x,y.

& READ x,y,button

- Liest die Mausposition in x,y und den Maustasten-status in "button"
- button = 0 => Maustaste nicht gedrückt
- button = 1 => Maustaste gerade gedrückt

- button = 2 => Maustaste gerade losgelassen
- button = 3 => Maustaste wird gehalten
- button = 0...3 + 32 => zusätzlich Shift - Taste
- button = 0...3 + 64 => zusätzlich ć - Taste
- button = 0...3 + 128 => zusätzlich Option - Taste
- button = 0...3 + 192 => zusätzlich ć - Taste und Option - Taste
- button = 0...3 + 224 => zusätzlich ć - Taste, Shift und Option - Taste

Die Soundbefehle

& SOUNDINIT (number)

- Initialisiert den SoundDOC mit "number" Oszillatoren. "number" = 1...32

& SOUND (A,B,C,D)

-	A = wait *256+nr,
	wait = 0...255;
	nr = 0...31;
		2 Byte großer Parameter mit :
		Warte Zyklus, nur ONE SHOT
		Oszillator Nummer

-	B = 0...65535	Frequenz in Herz
-	C = vol*256+page, vol = 0...255;	2 Byte großer Parameter mit : Lautstärke des Oszillators

page = 0…255; Startadresse der wavetable im SoundRAM (richtet sich nach der wavetable)

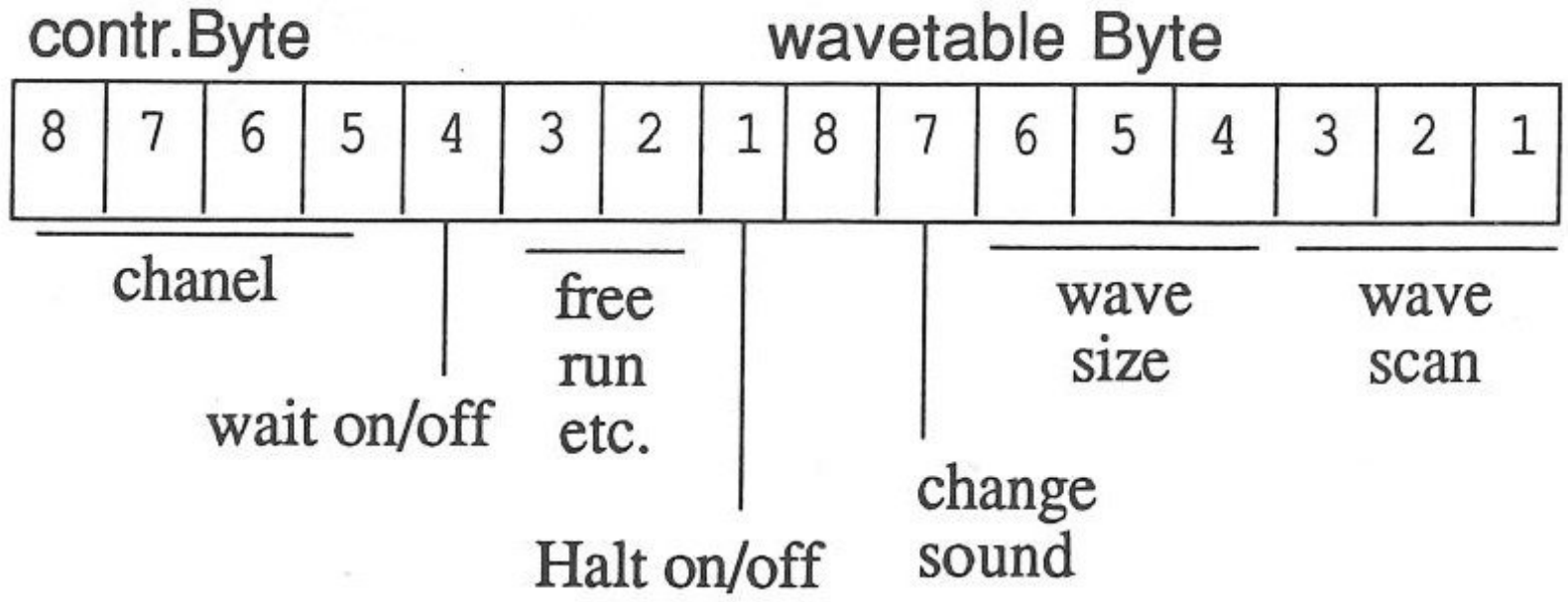
- D = contr*256+wavetable, 2 Byte großer Parameter mit folgender Belegung der Bytes :
contr Byte :
Bit 1 : Halt on/off => 1 = Halt aus 0 = Halt ein
Bit 2…3 : nimmt Werte von 0…3 an
0 = FREE RUN, d.h. Tonfolge wird kontinuierlich wiederholt
1 = ONE SHOT, d.h. Tonfolge wird nur einmal gespielt
2 = Synchron- oder Amplitudenmodulation, abhängig von der Aufrufreihenfolge der Oszillatoren. Oszillator 2 und 3 ergibt synchron, Oszillator 1 und 2 ergeben in dieser Reihenfolge Amplitudenmodulation
3 = SWAP Modus, d.h. sobald der Oszillator seine Tonfolge abgearbeitet hat, wird der nächste Oszillator eingeschaltet.
Bit 4 : Abfrage, ob "wait"-zyklus aus Parameter A beachtet werden soll; 0 = kein wait, 1 = waitzyklus ausführen.(nur ONE SHOT Modus)
Bit 5…8 : Kanalsteuerung links/rechts (0…15)

Beispiel: contr soll waitzyklus ausführen, Tonfolge im ONE SHOT Modus spielen, kein Halt Bit setzen und rechts spielen => contr = (0+1*2+1*8+0*16)

(Kursive Faktoren zur Berechnung der Bitposition)
wavetable Byte :
Bit 1...3: wave scan, d.h. abhängig vom Wert der Bits wird jeweils jedes, jedes zweite, jedes vierte Byte etc. der Tonfolge gespielt.
0 = jedes Byte
1 = jedes zweite Byte
2 = jedes vierte Byte
...7 = jedes128 te Byte
Bit 4...6: wave size : Größe der Tonfolge
0 = 1, 1 = 2, 2 = 4,...,7 = 128
Bit 7 : Change Sound, d.h. wenn Bit = 1, dann wird Soundbefehl ignoriert.
Bit 8 : nicht belegt.

Beispiel: wavetable soll jedes Byte spielen, Tonfolge ist 32 K Byte, und Soundbefehl soll ausgeführt werden.

=> wavetable = (0*64+7*8+0) (Kursive Faktoren zur Berechnung der Bitposition)
Gesamtgröße D ergibt sich für die beiden Beispiele zu : contr*256+wavetable =>
D= (0+1*2+1*8+0*16)*256 + (0*64+7*8+0)



Bemerkung zur Startposition einer wavetable: Die Startposition darf nur ein ganzahliges Vielfaches der wavetablegröße betragen, d.h. eine 512 Byte große wavetable kann nur an den page-Adressen 0, 2, 4,6 etc. liegen!

& DIGI (slot,M)

- Digitalisiert 32 K Tonfolge (wavetable) von der SuperSonic Digitizer Karte in Slot "slot" (1...6) in die Hires Grafik.
- "M" ist ein zwei Byte großer Parameter,
  Byte 1 = 1 (andere Werte werden ignoriert) bedeutet: Starte Digitalisieren auf Tastendruck
  Byte 1 = 0 bedeutet: Starte Digitalisieren auf Signaleingang
- Byte 0 = 12...255 gibt die Digitalisierungsdauer an. 12 ist Minimum wegen der Einschwingzeit des AD Wandlers
- Für eine 32 K Byte große Tonfolge mit Start auf Tastendruck bedeutet das: M = 1*256+255

- Eine Tonfolge kann mit PIC SAVE abgespeichert werden.

& WAVE (page,name$)

- Lädt wavetable mit Namen "name$" in den mit page (0...255) angegebenen Bereich des Sound-RAM. (Siehe PREFIX unter SAVE/LOAD)

& SOUND END

- schaltet den SoundDOC (Soundprozessor) des Apple II GS wieder ab.

Die sonstigen Programmierhilfen

& * [mon$]

- Ohne optionalen Parameter "mon$" identisch mit *CALL -151*. Mit dem Parameter "mon$" wird die in "mon$" enthaltene Monitoranweisung direkt ausgeführt und nach Basic zurückgekehrt.

& TEXT

- Schaltet vom Grafikbildschirm auf den Textbild-schirm um.

& PUSH (page,len)

- Kopiert 32 K Grafikbildschirm in den SoundRAM (rotierend)
- page = 0...255, d.h. wenn 32 K und Startpunkt page = 250, dann wird der *rest* an den Anfang des RAM geschrieben
- "len" = 1...128, wobei 128 = 32 K Byte. Wenn len = 126, dann wird die Grafik ohne Paletten gespeichert!

& PULL (page,len)

- Kopiert 32 K Byte SoundRAM in den Bildschirmspeicher. "page" und "len" siehe PUSH

& PEEK bank,addr,cont$

- Holt einen String mit Länge 255 aus bank 0...255 von Adresse 0...65535, sonst gleiche Funktion wie im Applesoft Basic
- Nicht im Direktmodus verwenden!

& POKE bank,addr,cont$

- Speichert einen String mit Länge 255 in bank 0...255 an Adresse 0...65535, sonst gleiche Funktion wie im Applesoft Basic

- Nicht im Direktmodus verwenden!

& GWINDOW (links,oben **TO** rechts,unten)

- Setzt Grafikfenster auf angegebene Größe
- Befehle HPLOT, DRAW, BOX, BOXFILL, FILL, SCRN und CIRCLE wirken nur in diesem Fenster.

& TWINDOW (htabl,vtabo **TO** htabr,vtabu)

- Bestimmt die Größe des Textfensters. Sonst Funktion wie Applesoft Poke

& CALL A,B, [c], [d],...

- Zugriff auf die Toolbox
- A = tool + func*256
  B = in + out*256
  [c] = [tostack]
  [d] = [fromstack]

& TIME (monat,tag,jahr;stunde,minute)

- Holt Datum und Uhrzeit in die entsprechenden Variablen.

& VER= number

- Holt die Versionsnummer von X-Basic (als Integerzahl, also das 100 fache) in "number"

& ERR (line, berror, terror)

- berror (Basic error) enthält 254, wenn ein toolbox error aufgetreten ist, sonst identisch mit Peek(222)
- terror (Toolbox error) enthält den toolbox error, nur wenn berror =254, sonst nicht interpretierbar
- line enthält die Zeilennummer, in der der Fehler aufgetreten ist.

& END

END beendet die Arbeit mit X-Basic und entfernt das Basic vollständig aus dem Speicher. Dieser Befehl sollte angewandt werden, um Fehler in anderen Programmen - die möglicherweise nicht ausreichend abgesichert sind - zu vermeiden.

Kapitel 4 Programmieren in X-Basic

Einige Beispiele

Programm LINE

```
100  E = 52: GOTO 190  -- Sprung zur Einstellung 320. Grafik
110  X1 =  RND (1) * W -- Anfangspunkte berechnen
     :X2 =  RND (1) * W
     :Y1 =  RND (1) * 200
     :Y2 =  RND (1) * 200
120  X3 =  RND (1) * 8 - 4 -- Bewegungsrichtung berechnen
     :X4 =  RND (1) * 8 - 4
     :Y3 =  RND (1) * 8 - 4
     :Y4 =  RND (1) * 8 - 4
130  FOR I = 1 TO 512 -- 512 x Linien zeichnen
     : &  HCOLOR=  INT ( RND (1) * F + 1)
     : &  HPLOT X1,Y1 TO X2,Y2
     : & INKEY(E)
     : IF E <  >  - 1 THEN 190 -- Ist eine Taste gedrückt ?
140  X1 = X1 + X3  -- Punkte bewegen
     : IF X1 < 0 OR X1 > W THEN X3 =  - X3 -- Im Fenster ?
     :X1 = X1 + X3 + X3 -- Sonst Bewegungsrichtung ändern
150  X2 = X2 + X4
     : IF X2 < 0 OR X2 > W THEN X4 =  - X4
     :X2 = X2 + X4 + X4
160  Y1 = Y1 + Y3
     : IF Y1 < 0 OR Y1 > 199 THEN Y3 =  - Y3
     :Y1 = Y1 + Y3 + Y3
170  Y2 = Y2 + Y4
     : IF Y2 < 0 OR Y2 > 199 THEN Y4 =  - Y4
     :Y2 = Y2 + Y4 + Y4
180  NEXT -- 512te Linie ?
     : &  CLEAR ( RND (1) * F)
     : &  COLOR=  RND (1) * F,15,2 -- Ramenfarbe verändern
     : GOTO 110 -- Rücksprung zum zeichnen
190  IF E = 52 THEN F = 16 -- war Taste = "4"
     : &  HGR ( RND (1) * F) -- Umschaltung auf 320 Pkte
```

```
      :W = 320
      : GOTO 110
200   IF E = 56 THEN F = 4 -- war Taste = "8"
      : &  HGR2 ( RND (1) * F) -- Umschaltung auf 640 Pkte
      :W = 640
      : GOTO 110 -- Rücksprung zum zeichnen
210   IF E = 27 THEN  &  TEXT -- ESC für beenden
      : END
220   IF E = 32 THEN  GET E$ -- SPACE für Pause
      : GOTO 140
230   GOTO 140 -- Rücksprung in FOR - NEXT Schleife
```

Programm MOVER1

```
100   &  HGR (0) -- Initialisiert Hires
      : &  STORE 0,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
110 X1 =  RND (1) * 320 -- Berechnung der Startpkte
      :X2 =  RND (1) * 320
      :Y1 =  RND (1) * 200
      :Y2 =  RND (1) * 200
120 X3 =  RND (1) * 8 - 4 -- Berechnung der Bewegungsrichtg
      :X4 =  RND (1) * 8 - 4
      :Y3 =  RND (1) * 8 - 4
      :Y4 =  RND (1) * 8 - 4
130   FOR J = 1 TO 20 -- 20 x 15 Linien zeichnen
      : FOR I = 15 TO 1 STEP  - 1
      : &  HCOLOR= I
      : GOSUB 190
      : &  HPLOT X1,Y1 TO X2,Y2
      : & INKEY(E)
      : ON E = 27 GOTO 300 -- ESC Taste
      : IF E = 32 THEN  GET E$ -- SPACE Taste
140 X1 = X1 + X3 --Punkt bewegen
      : IF X1 < 0 OR X1 > 319 THEN X3 =  - X3 -- im Fenster ?
      :X1 = X1 + X3 + X3 -- sonst Bewegungsrichtung ändern
150 X2 = X2 + X4
      : IF X2 < 0 OR X2 > 319 THEN X4 =  - X4
      :X2 = X2 + X4 + X4
160 Y1 = Y1 + Y3
```

```
     : IF Y1 < 0 OR Y1 > 199 THEN Y3 =  - Y3
     :Y1 = Y1 + Y3 + Y3
170 Y2 = Y2 + Y4
     : IF Y2 < 0 OR Y2 > 199 THEN Y4 =  - Y4
     :Y2 = Y2 + Y4 + Y4
180  NEXT -- Ende der FOR - NEXT Schleife "I"
     : NEXT -- Ende der FOR - NEXT Schleife "J"
     : &  CLEAR (0)
     : GOTO 130
190  &  RESTORE 0,A,B,C,D,E,F,G,H,K,L,M,N,O,P,Q,R
200  &  STORE 0,0,B + 1,C + 1,D + 1,E + 1,F + 1,G + 1,H +
     1,K + 1,L + 1,M + 1,N + 1,O + 1,P + 1,Q + 1,R + 1
     : RETURN -- Jede Farbe in Palette um 1 erhöhen
300  &  TEXT -- Ende
     : END
```

Programm MOVER2

```
100  &  HGR (0) -- initialisiert Hires
     : &  OR
     :AN = 80 -- Linien Anzahl kann > oder < gemacht werden
     :GX = 320 -- Grafikweite
     :GY = 200 -- Grafikzeilen größe
     :F = 15 -- Farben Anzahl
     : DIM X1(AN),Y1(AN),X2(AN),Y2(AN),CO(AN)
110 X1(0) =  INT ( RND (1) * GX) -- Punkte berechnen
     :X2(0) =  INT ( RND (1) * GX)
     :Y1(0) =  INT ( RND (1) * GY)
     :Y2(0) =  INT ( RND (1) * GY)
     :CO(0) = 1 -- Farbe für ersten Punkt
120 H1 =  INT ( RND (1) * 8) - 4 -- Bewegung berechnen
     :H2 =  INT ( RND (1) * 8) - 4
     :V1 =  INT ( RND (1) * 8) - 4
     :V2 =  INT ( RND (1) * 8) - 4
130  IF H1 = 0 OR H2 = 0 OR V1 = 0 OR V2 = 0 THEN 120
140  FOR P1 = 0 TO AN - 1 -- Je nach AN Lienen Zeichnen
     :P2 = P1 + 1
     : GOSUB 250 -- Linie Zeichnen, Taste abfragen
     : GOSUB 190 -- Punkt bewegen
```

```
      : NEXT
      :P1 = AN -- Letzte Linie
      :P2 = 0 -- Erste Linie
150 P = P1 -- P1 merken
      :P1 = P2 -- P1 gleichsetzen mit P2
      : GOSUB 250 -- Letzte Linie weiter bewegen
      :P1 = P -- P1 wieder zurückhohlen
      : GOSUB 190 -- Ertste Linieweglöschen
      : GOSUB 250 -- Punkt bewegen
160 P1 = P1 + 1 -- P1 Züklich erhöhen
      : IF P1 > AN THEN P1 = 0
170 P2 = P2 + 1 -- P2 Züklich erhöhen
      : IF P2 > AN THEN P2 = 0
180  GOTO 150 -- Weiter Züklich durchlaufen
190 X1(P2) = X1(P1) + H1
      : IF X1(P2) > GX OR X1(P2) < 0 THEN H1 = - H1
      : GOTO 190
200 X2(P2) = X2(P1) + H2
      : IF X2(P2) > GX OR X2(P2) < 0 THEN H2 = - H2
      : GOTO 200
210 Y1(P2) = Y1(P1) + V1
      : IF Y1(P2) > GY OR Y1(P2) < 0 THEN V1 = - V1
      : GOTO 210
220 Y2(P2) = Y2(P1) + V2
      : IF Y2(P2) > GY OR Y2(P2) < 0 THEN V2 = - V2
      : GOTO 220
230 CO(P2) = CO(P1) + 1 -- Farbe Züklich erhöhen
      : IF CO(P2) > F THEN CO(P2) = 1
240  RETURN
250  &  HCOLOR= CO(P1)
      : &  HPLOT X1(P1),Y1(P1) TO X2(P1),Y2(P1)
      : & INKEY(E)
      : IF E =  - 1 THEN  RETURN
260  IF E = 27 THEN  &  TEXT -- ESC für Ende
      : END
270  IF E = 32 THEN  GET E$ -- Space für kurze Pause
      : RETURN
280  RETURN
```

Programm MOVER3

```
100 …
105  &  STORE 0,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
110 … -- weiter wie MOVER2
250 …
260  &  RESTORE
     0,FA,FB,FC,FD,FE,FF,FG,FH,FI,FJ,FK,FL,FM,FQ,FO,FP
270  &  STORE 0,0,FB + 1,FC + 1,FD + 1,FE + 1,FF + 1,FG +
     1,FH + 1,FI + 1,FJ + 1,FK + 1,FL + 1,FM + 1,FQ + 1,FO +
     1,FP + 1 -- Jede Farbe um 1 erhöhen
     : & INKEY(E)
280  IF E = 27 THEN  &  TEXT -- ESC für Ende
     : END
290  IF E = 32 THEN  GET E$ -- Space für warten
     : RETURN
300  RETURN
```

Programm MOVER4

```
100  …-- weiter wie MOVER2
240  …
250  &  HCOLOR= CO(P1)
     : &  HPLOT X1(P1),Y1(P1) TO X2(P1),Y2(P1)
260  &  RESTORE -- Alle Farben lesen
     0,FA,FB,FC,FD,FE,FF,FG,FH,FI,FJ,FK,FL,FM,FQ,FO,FP
270  &  STORE --Erste Farbe nach hinten bewegen
     0,0,FC,FD,FE,FF,FG,FH,FI,FJ,FK,FL,FM,FQ,FO,FP,FB
     : & INKEY(E)
280  IF E = 27 THEN  &  TEXT
     : END
290  IF E = 32 THEN  GET E$
     : RETURN
300  RETURN
```

Programm DRAW

```
100  &  HGR (0) -- initialisiert Hires
     : &  OR
```

```
      : &  HCOLOR= 10
110   X = 0   :Y = 0 -- x/y Position auf 0
      :F = 15:XR = 1 -- max. Farben 15
      :YR = 1:ZR = 1 -- ROT= Variablen
      :C = 1  :S = 0 -- C=Color, S=Screen color
120 A$ =  CHR$ (1) + " APPLE //gs " +  CHR$ (1)
130   & MOUSEINIT(0,0 TO 312,192;X,Y)
140   &  ROT= XR,YR,ZR
      : &  DRAW  AT (X,Y);A$
150   &  READ XP,YP,B -- Mausabfrage
      : & INKEY(E) -- Tastaturabfrage
160   IF E =  - 1 AND XP = X AND YP = Y AND B = 0 THEN 150
170   &  DRAW  AT (X,Y);A$
      :Y = YP:X = XP
180   IF B THEN  &  AND -- Wenn Bottum dann Draw
      : &  DRAW  AT (X,Y);A$
      : &  OR
190   IF E =  - 1 THEN 140
200 E$ =  CHR$ (E)
210   IF E = 27 THEN  &  TEXT : END -- ESC Ende
220   IF E$ = "4" THEN  &  HGR (0) -- Taste "4" => 320 Mode
      :F = 15: &  OR
      :C = 1:S = 0
      : &  HCOLOR= C
230   IF E$ = "8" THEN  &  HGR2 (0) -- Taste "8" => 640 Mode
      :F = 3 : &  OR
      :C = 1  :S = 0
      : &  HCOLOR= C
240   IF E$ = "I" THEN  &  INVERSE -- "I" für Inverse
250   IF E$ = "N" THEN  &  NORMAL -- "N" für Normal
270   IF E$ = "C" THEN  &  HCOLOR= C -- "C" Draw Farbe
      :C = C + 1
      : IF C > F THEN C = 0
280   IF E$ = "S" THEN  &  CLEAR (S) -- "S" Screen Farbe
      :S = S + 1
      : IF S > F THEN S = 0
290   IF E$ = "Q" THEN ZR = 1 -- Drehfaktoren festlegen
300   IF E$ = "W" THEN ZR =  - 1
310   IF E$ = "E" THEN XR =  - 1
320   IF E$ = "R" THEN XR = 1
330   IF E$ = "T" THEN YR = 1
```

```
340  IF E$ = "Z" THEN YR =  - 1
350  GOTO 140
```

Programm BOXFILL

```
100  E = 52 : GOTO 240
110  X1 = 160
     :Y1 = 100
     :X2 = 160
     :Y2 = 100
120  & MOUSEINIT(0,0 TO W,199;X1,Y1)
130  &  OR  : &  DRAW  AT (X1,Y1);"^B" -- MauseZeichen
140  &  READ X,Y,B
     : & INKEY(E)
150  IF X = X1 AND Y = Y1 AND E =  - 1 AND B = 0 THEN 140
160  &  DRAW  AT (X1,Y1);"^B"
     :X1 = X:Y1 = Y
170  IF B = 1 THEN  & BOXFILL(X2,Y2 TO X1,Y1) -- Taste los
     : GOTO 130
180  IF B = 2 THEN X2 = X1 -- Taste geradegedrückt
     :Y2 = Y1
     : GOTO 130
190  IF B = 3 THEN  & BOX(X1,Y1 TO X2,Y2) -- Taste gedrückt
     : & BOX(X1,Y1 TO X2,Y2)
     : GOTO 130
200  IF E = 67 THEN FA = (FA + 1) * (FA < 15) -- "F"
     : ON FA = SA GOTO 200
     : &  HCOLOR= FA
210  IF E = 27 THEN  &  TEXT  -- ESC
     : END
220  IF E = 70 THEN  & FILL(X1,Y1,FA) -- "F" => Fill
230  IF E = 83 THEN  &  CLEAR (FA) --"S"
     :SA = FA
     :E = 67: GOTO 200
240  IF E = 52 THEN  &  HGR (0)
     : &  HCOLOR= 1
     :FA = 1:SA = 0
     :W = 319
     : GOTO 110
```

```
250  IF E = 56 THEN  &  HGR2 (0)
     : &  HCOLOR= 1
     :FA = 1:SA = 0
     :W = 639
     : GOTO 110
260  GOTO 130
```

Programm FILL1

```
100  &  HGR (0)
     : &  HCOLOR= 1
     :F = 1
110 X1 = 160:Y1 = 100
     :X2 = 160
     :Y2 = 100
111  & MOUSEINIT(0,0 TO 319,199;X1,Y1)
120  &  OR  : &  HPLOT X1,Y1 TO X2,Y2
     : &  READ X,Y,B
     : & INKEY(E)
130  IF E = 13 OR B THEN  &  AND -- RETURN oder Taste
     : &  HPLOT X1,Y1 TO X2,Y2
     : &  OR
     :X2 = X1
     :Y2 = Y1
     : GOTO 120
140  IF E = 32 THEN  &  HPLOT X1,Y1 TO X2,Y2 I-- SPACE um
     :X2 = X1 -- Line neu anzusetzen
     :Y2 = Y1
     : GOTO 120
150  &  HPLOT X1,Y1 TO X2,Y2
     :X1 = X:Y1 = Y
160  IF E = 67 THEN F = (F + 1) * (F < 15)
     : &  HCOLOR= F
210  IF E = 27 THEN  &  TEXT : END -- ESC
220  IF E = 70 THEN  & FILL(X1,Y1,F) -- "F"
230  IF E = 83 THEN  &  CLEAR (F) -- "S"
     :E = 67: GOTO 160
240  GOTO 120
```

Programm FILL2

```
100 E = 52   : GOTO 220
110 X1 = 160:Y1 = 100
     :X2 = 160
     :Y2 = 100
120  & MOUSEINIT(0,0 TO W,199;X1,Y1)
130  &  OR   : &  DRAW  AT (X1,Y1);"^B"
140  &  READ X,Y,B
     : & INKEY(E)
     : IF X = X1 AND Y = Y1 AND E =  - 1 AND B = 0 THEN 140
150  &  DRAW  AT (X1,Y1);"^B"
     :X1 = X:Y1 = Y
160  IF B = 3 THEN  &  AND -- Linie zeichnen
     : &  HPLOT X1,Y1 TO X2,Y2
     :X2 = X1
     :Y2 = Y1
170  IF B = 2 THEN X2 = X1 -- Koordinate merken
     :Y2 = Y1
     : GOTO 130
180  IF E = 67 THEN FA = (FA + 1) * (FA < F)
     : ON FA = SA GOTO 180
     : &  HCOLOR= FA
190  IF E = 27 THEN  &  TEXT
     : END
200  IF E = 70 THEN  & FILL(X1,Y1,FA)
210  IF E = 83 THEN  &  CLEAR (FA)
     :SA = FA
     :E = 67
     : GOTO 180
220  IF E = 52 THEN  &  HGR (0)
     : &  HCOLOR= 1
     :FA = 1
     :SA = 0
     :F = 15
     :W = 319
     : GOTO 110
230  IF E = 56 THEN  &  HGR2 (0)
     : &  HCOLOR= 1
     :FA = 1
```

```
      :SA = 0
      :F = 4
      :W = 639
      : GOTO 110
240   GOTO 130
```

Programm CIRCLE

```
100 E = 52
      : GOTO 240
110 X1 = 160 -- Startposition
      :Y1 = 100
      :X2 = 160
      :Y2 = 100
120   & MOUSEINIT(0,0 TO W,199;X1,Y1) -- Mousefenster setzen
130   &  OR
      : &  DRAW  AT (X1,Y1);"^B"
140   &  READ X,Y,B
      : & INKEY(E)
150   IF X = X1 AND Y = Y1 AND E =  - 1 AND B = 0 THEN 140
160   &  DRAW  AT (X1,Y1);"^B"
      :X1 = X
      :Y1 = Y
170   IF B = 1 THEN  &  AND
      : & CIRCLE(X2,Y2,X2 - X,Y2 - Y,0,0) -- Foller Kreis
      : &  OR
      : GOTO 130
180   IF B = 2 THEN X2 = X
      :Y2 = Y
      : GOTO 130
190   IF B = 3 THEN  & CIRCLE(X2,Y2,X2 - X,Y2 - Y,32,32)
      : & CIRCLE(X2,Y2,X2 - X,Y2 - Y,32,32) -- Schneller Kr.
      : GOTO 130
200   IF E = 67 THEN FA = (FA + 1) * (FA < F)
      : ON FA = SA GOTO 200
      : &  HCOLOR= FA
210   IF E = 27 THEN  &  TEXT
      : END
220   IF E = 70 THEN  & FILL(X1,Y1,FA) -- "F"
```

```
230  IF E = 83 THEN  &  CLEAR (FA)
     :SA = FA
     :E = 67
     : GOTO 200
240  IF E = 52 THEN  &  HGR (0)
     : &  HCOLOR= 1
     :FA = 1
     :SA = 0
     :F = 15
     :W = 319
     : GOTO 110
250  IF E = 56 THEN  &  HGR2 (0)
     : &  HCOLOR= 1
     :FA = 1
     :SA = 0
     :F = 4
     :W = 639
     : GOTO 110
260  GOTO 130
```

Programm PIECE

```
100  &  HGR (0)
     :X = 1
     : &  HCOLOR= X
     :RX = 100 -- Radien
     :RY = 50
     :MX = 160
     :MY = 100
110  FOR Y = 0 TO 255
     :X = X + 1
     : IF X > 15 THEN X = 1
120  &  HCOLOR= X
     : & CIRCLE(MX,MY,RX,RY,16,Y * 256 + 32) -- Halb Ellipse
     : NEXT
130  &  RESTORE 0,Q,W,E,R,T,Z,U,I,O,P,A,S,D,F,G,H -- Palette
     : &  STORE 0,0,W,H,E,R,T,Z,U,I,O,P,A,S,D,F,G -- Züklen
     : & INKEY(E)
     : IF E =  - 1 THEN 130
```

```
140  &  TEXT
     : END
```

Programm POLY.1

```
100  &  HGR (0)
     :F = 1
110  FOR I = 0 TO 255
     : &  HCOLOR= F
     :F = F + 1
     : IF F > 15 THEN F = 1
120  & CIRCLE(160,100,I,I,3,3 + I * 256) -- Dreieck
     : NEXT
130  &  RESTORE 0,Q,W,E,R,T,Z,U,I,O,P,A,S,D,F,G,H
     : &  STORE 0,0,W,H,E,R,T,Z,U,I,O,P,A,S,D,F,G
     : & INKEY(E)
     : IF E =  - 1 THEN 130
140  &  TEXT
     : END
```

Programm POLY.3.MOUSE

```
100  &  HGR (0)
     :F = 1
110  FOR I = 0 TO 255
     : &  HCOLOR= F
     :F = F + 1
     : IF F > 15 THEN F = 1
120  & CIRCLE(160,100,I,I,3,3 + I * 256) -- Dreieck
     : NEXT
130  &  GET E
     : ON E = 8 GOTO 150 -- Maus links
     : ON E = 21 GOTO 140 -- Mause rechts
     : ON E = 27 GOTO 160
     : GOTO 130
140  &  RESTORE 0,Q,W,E,R,T,Z,U,I,O,P,A,S,D,F,G,H
     : &  STORE 0,0,H,W,E,R,T,Z,U,I,O,P,A,S,D,F,G
     : GOTO 130
```

```
150   &  RESTORE 0,Q,W,E,R,T,Z,U,I,O,P,A,S,D,F,G,H
      : &  STORE 0,0,E,R,T,Z,U,I,O,P,A,S,D,F,G,H,W
      : GOTO 130
160   &  TEXT
      : END
```

Programm POLY.8

```
100   &  HGR (0)
      :F = 1
110   FOR I = 0 TO 200
      : &  HCOLOR= F
      :F = F + 1
      : IF F > 15 THEN F = 1
120   & CIRCLE(160,100,I,I,8,8 + I * 256) -- Achteck
      : NEXT
130   &  RESTORE 0,Q,W,E,R,T,Z,U,I,O,P,A,S,D,F,G,H
      : &  STORE 0,0,W,H,E,R,T,Z,U,I,O,P,A,S,D,F,G
      : & INKEY(E)
      : IF E =  - 1 THEN 130
140   &  TEXT
      : END
```

Programm SPRITE

```
100   &  HGR (15)
      : & MOUSEINIT(0,0 TO 320,200;0,0)
      : &  OR
      :X = 0
      :Y = 0
      :S = 0
110   PRINT  CHR$ (4)"BLOAD SPRITE.DATA,A$300,L$80"
      : &  PEEK 0,768,B$ -- Sprite Daten holen
      :A$ =  CHR$ (0) +  LEFT$ (B$,128) -- Hintergrung def.
120   & SPRITE(X,Y,A$)
130   &  READ XP,YP,B
      : & INKEY(E)
      : IF E <  >  - 1 THEN 160
```

```
140  ON B GOTO 210,210,210 -- Sprite zeichnen
     : IF X = XP AND Y = YP THEN 130 -- Teste ob Mausebeweg.
150  & SPRITE(X,Y,A$)
     :X = XP
     :Y = YP
     : GOTO 120
160  IF E = 83 THEN S = S + 1
     : IF S > 15 THEN S = 0
170  IF E = 27 THEN  &  TEXT
     : END
180  IF E = 83 THEN  &  HGR (S)
     : &  OR
     : GOTO 120
190  IF E = 13 THEN 120
200  GOTO 130
210  & SPRITE(X,Y,A$) -- Sprite unwiederuflich Speichern
     : &  AND
     : & SPRITE(XP,YP,A$)
     : &  OR
     :X = XP
     :Y = YP
     : GOTO 120
```

Programm DEMO

```
100  DIM X(32),Y(32)
     : GOTO 130
110  & INKEY(E) -- Tastatur abfragen
     : IF E =  - 1 THEN  RETURN
120  &  TEXT
     : END
130  &  HGR2 (0)
140  REM ---------------------------------- rahmen
150  &  COLOR=  - 1, - 2, - 16
151  &  PRINT  AT (28,11);"Welcome Apple IIgs grafik"
160  &  PRINT  AT (31,13);"supported by XBASIC"
170  &  HCOLOR= 3
     : &  HPLOT 1,1
180  FOR I = 1 TO 80 STEP 4 -- Vierkantspirale
```

```
190  &  HPLOT  TO 640 - 2 * I,I
200  &  HPLOT  TO 640 - 2 * I,200 - I - 2
210  &  HPLOT  TO 2 * I + 2,200 - I - 2
220  &  HPLOT  TO 2 * I + 2,I + 4
230  GOSUB 110: NEXT
240  REM ------------------------------- labyrinth
250  &  HGR2 (2)
260 X = 320
     :Y = 100
     :C = 1
270  &  HCOLOR= C
280  &  HPLOT X,Y
290  FOR I = 1 TO 300
300 X = X + ( RND (1) > 0.5) * 20
     : &  HPLOT  TO X,Y
310 Y = Y - ( RND (1) > 0.5) * 10
     : &  HPLOT  TO X,Y
320 X = X - ( RND (1) > 0.5) * 20
     : &  HPLOT  TO X,Y
330 Y = Y + ( RND (1) > 0.5) * 10
     : &  HPLOT  TO X,Y
340  IF X > 600 OR X < 40 OR Y > 180 OR Y < 20 THEN X = 320
     :Y = 150
     : &  HPLOT X,Y
     :C = C + 1
     : IF C = 2 OR C > 3 THEN C = 0
350  &  HCOLOR= C
     : GOSUB 110
     : NEXT
360  REM ------------------------------- kringel
370  &  HGR2 (1)
380  &  COLOR=  - 1, - 3, - 1
     : &  PRINT  AT (30,11);"Welcome to XBASIC"
390  &  COLOR=  - 1, - 2, - 1
     : &  PRINT  AT (26,12);"full support of GS-grafic"
400  FOR K = 1 TO 40
410 C =  INT ( RND (1) * 3)
     : IF C = 1 THEN C = 0
420 R =  INT ( RND (1) * 17 + 8)
430 X =  INT ( RND (1) * 515 + 63)
440 Y =  INT ( RND (1) * 150 + 25)
```

```
450  IF X + 2.5 * R < 220 OR X - 2.5 * R > 420 THEN 480
460  IF Y + R < 80 OR Y - R > 120 THEN 480
470  GOTO 430
480  GOSUB 660
     : GOSUB 110
     : NEXT
490  REM ---------------------------------- rechteck
500  &  CLEAR (0)
510  FOR K = 1 TO 200
520  &  HCOLOR=  RND (1) * 3 + 1
530 X =  RND (1) * 540 + 20
540 Y =  RND (1) * 170 + 25
550 S =  RND (1) * 5 + 1
560  GOSUB 710
     : GOSUB 110
     : NEXT
570  REM ---------------------------------- dreieck
580  &  HGR2 (3)
590  FOR K = 1 TO 150
600 X =  RND (1) * 540 + 40
610 Y =  RND (1) * 150 + 25
620 R =  INT ( RND (1) * 256)
630 S =  RND (1) * 20 + 10
640  &  HCOLOR= 1
     : GOSUB 780
     : GOSUB 110
     : NEXT
650  GOTO 130
660  REM ---------------------------------- Kreiszeichnen
670  &  HCOLOR= C
     : & CIRCLE(X,Y,R * 2.5,R,32,32) -- 32 Eck fast Kreis
     : RETURN
710  REM ---------------------------------- Draw BOX
720  & BOX(X,Y TO X + 40,Y - 3 * S)
     : RETURN
780  REM ---------------------------------- DRAW Triaera
790  & CIRCLE(X,Y,S,S,3,3 + R * 256)
     : RETURN
```

Programm FUNKTION

```
100  &  HGR2 (1):N$ = ""
     : GOTO 130
110  & INKEY(E) --Tastatur abfragen
     : IF E =  - 1 THEN  RETURN
120  &  TEXT
     : END
130  &  HCOLOR= 0 -- Koordinaten kreuz malen
     : &  HPLOT 50,120
     : &  HPLOT  TO 590,120
140  &  HPLOT  TO 585,117
     : &  HPLOT  TO 590,120
     : &  HPLOT  TO 585,123
150  &  HCOLOR= 0
     : &  HPLOT 320,190
     : &  HPLOT  TO 320,20
160  &  HPLOT  TO 315,23
     : &  HPLOT  TO 320,20
     : &  HPLOT  TO 325,23
170  &  COLOR=  - 1, - 16, - 1
     : &  PRINT  AT (5,1);N$ + "Funktion:"
180  &  HCOLOR= 3
     : &  HPLOT 35,18
     : &  HPLOT  TO 110,18
190  DEF  FN A(X) =  SIN (X) *  LOG ( ABS (X))
200 F$ = "f(x) = sin (x) * log (abs(x))"
210  &  COLOR=  - 1, - 2, - 1
     : &  PRINT  AT (2,3);N$ + F$
220  &  HCOLOR= 2
     : &  HPLOT 50,119
230  FOR I = 50 TO 580 STEP 5 -- Funktion1 malen
240  &  HPLOT  TO I + 2, FN A(I / 15) * 15 + 120
250  GOSUB 110 -- Tastatur abfragen
     : NEXT
260  DEF  FN A(X) =  SIN (X) / X
270 F$ = "f(x) = sin (x) / x"
280  &  COLOR=  - 1, - 16, - 1
     : &  PRINT  AT (2,4);N$ + F$
290  &  HCOLOR= 0
```

```
     : &  HPLOT 50,119
300  FOR I = 50 TO 580 STEP 5 -- Funktion2 malen
310  &  HPLOT  TO I + 2, FN A(I / 15) * 275 + 120
320  GOSUB 110
     : NEXT
330  DEF  FN A(X) = X ^ 3 - 2 * X ^ 2 + 3 * X - 5 / X ^ 2 -
     2 * X
340 F$ = "f(x) = x^3 - 2x^2 + 3x - 5 / x^2 - 2x
350  &  COLOR=  - 1, - 3, - 1
     : &  PRINT  AT (2,5);N$ + F$
360  &  HCOLOR= 3
     : &  HPLOT 52, FN A(50 / 20 - 15.8) / 40 + 125
370  FOR I = 50 TO 580 STEP 10 -- Funktion3 malen
380  &  HPLOT  TO I + 2, FN A(I / 20 - 15.8) / 40 + 125
390  GOSUB 110
     : NEXT
420  &  COLOR=  - 1, - 2, - 1
     : &  PRINT  AT (49,23);"Bitte eine Taste druecken"
430  GOSUB 110 -- Warteschleife bis Taste
     : GOTO 430
```

Programm SWEEPER

```
100  HOME
     : &  HGR (0)
     :C = 1
110  &  STORE 0,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
120  &  CLEAR (0)
     :X =  INT ( RND (1) * 5) + 2
130  FOR A = 0 TO 159 STEP X * 2
140  &  HCOLOR= C
     :C = C + 1
     : IF C > 15 THEN C = 1
150  &  HPLOT A,0 TO 159 - A,99 -- Muster zeichnen
160  &  HPLOT 319 - A,0 TO 159 + A,99
170  &  HPLOT A,199 TO 159 - A,100
180  &  HPLOT 319 - A,199 TO 159 + A,100
190  NEXT
200  FOR B = 0 TO 96 STEP X + 2
```

```
210   &  HCOLOR= C
      :C = C + 1 -- Farbe erhöhen
      : IF C > 15 THEN C = 1
220   &  HPLOT 0,99 - B TO 159,B
230   &  HPLOT 159,B TO 319,99 - B
240   &  HPLOT 0,100 + B TO 159,199 - B
250   &  HPLOT 319,100 + B TO 159,199 - B
260   NEXT
270   IF C > 3 THEN C = C - 3
      : GOTO 270
280   FOR I = 0 TO 100
290   & INKEY(E)
      : IF E <  >  - 1 THEN  &  TEXT
      : END
300   &  RESTORE -- Farbpallete wandern lassen
      0,FA,FB,FC,FD,FE,FF,FG,FH,FI,FJ,FK,FL,FM,FQ,FO,FP
310   &  STORE 0,0,FB - C,FC - C,FD - C,FE - C,FF - C,FG -
      C,FH - C,FI - C,FJ - C,FK - C,FL - C,FM - C,FQ - C,FO -
      C,FP - C
      : NEXT
      : GOTO 120
```

Programm TREFFER

```
100   DIM X(32),Y(32)
      : GOTO 130
110   & INKEY(E) --abfrage ob eine Taste gedrückt wurde
      : IF E =  - 1 THEN  RETURN
120   &  TEXT
      : END
130   &  HGR2 (0)
140 I$ = "Willkommen zum XBASIC"
150 B$ = " fuer Apple IIgs GRAFIK"
160   &  COLOR=  - 1, - 3, - 16 -- Farbe 3 und 0 !!!
      : &  PRINT  AT (17,2);I$ + B$
170   X = 270
      :Y = 100
```

```
180  FOR R = 4 TO 48 STEP 4 -- 11x Kreise und füllen
190   &  HCOLOR= 3
      : GOSUB 380
200   & FILL(X,Y + R - 2,1 +( INT (R / 8) = R / 8))
      : NEXT
210 I$ = "****************************************"
220   &  COLOR=  - 1, - 1, - 16
      : &  PRINT  AT (1,0);I$ + I$
      : &  PRINT  AT (1,24);I$ + I$
230   &  HCOLOR= 1 -- Hexenhaus malen
      : &  HPLOT 500,120 TO 600,120 TO 600,70 TO 500,70
240   &  HPLOT  TO 550,45 TO 600,70 TO 500,120 TO 500,70 TO
      600,120
250   &  HCOLOR= 1
      : &  HPLOT 501,120 TO 501,70
260   &  HCOLOR= 1
      : &  HPLOT 599,120 TO 599,70
270   &  HCOLOR= 1
      : &  HPLOT 550,95
280   & FILL(550,72,3)  Hexenhaus ausfüllen
290   & FILL(597,95,2)
300   & FILL(550,118,3)
310   & FILL(503,95,2)
320   & FILL(550,50,3)
330   &  HCOLOR= 3
      : &  HPLOT 100,30
340   &  HPLOT  TO 270,100 TO 107,26 TO 100,30 -- Pfeil
350   & FILL(107,28,1)
360   &  PRINT  AT (21,22);"Ein Volltreffer fuer Ihren GS"
370   GOSUB 110
      : GOTO 370
380   REM ------------------------------- Kreiszeichnen
390   & CIRCLE(X,Y,R * 2.5,R,0,0)
      : RETURN
```

Programm MAKE.SOUND.DATA

```
110  RESTORE
     :AN = 14 --Ton Anzahl
     : DIM FR(210,AN),DA(210),S1(AN),S2(AN),P(10)
     : FOR I = 0 TO 6
     :T(I) = 2 ^ I -- Oktaven Berechnung
     : NEXT
     :F1 = 0
     :PO = 0
130  READ DA(F1)
     : ON DA(F1) = 0 GOTO 300 -- Lesen bis Dauer=0
     : PRINT DA(F1); TAB( 4);":";
     : IF  SGN (DA(F1)) =  - 1 THEN P(PO) = F1 -- Dauer -
     :PO = PO + 1 -- Dann merken (für Wiederholungen)
131 DA(F1) =  ABS (DA(F1))
     : FOR F2 = 1 TO AN -- all 14 Töne hohlen
     : READ TN$
     :T$ =  LEFT$ (TN$,2)
     :O =  VAL ( RIGHT$ (TN$,1))
     :O =  ABS (O - 1)
139  REM Noten umwandeln in Frequenzen + Oktaven
140  IF T$ = ".." THEN FR(F1,F2) = 0
150  IF T$ = "C " THEN FR(F1,F2) = 32.703 * T(O)
160  IF T$ = "C#" OR T$ = "Db" THEN FR(F1,F2) = 34.648 *
     T(O)
170  IF T$ = "D " THEN FR(F1,F2) = 36.708 * T(O)
180  IF T$ = "D#" OR T$ = "Eb" THEN FR(F1,F2) = 38.891 *
     T(O)
190  IF T$ = "E " THEN FR(F1,F2) = 41.203 * T(O)
200  IF T$ = "F " THEN FR(F1,F2) = 43.654 * T(O)
210  IF T$ = "F#" OR T$ = "Gb" THEN FR(F1,F2) = 46.249 *
     T(O)
220  IF T$ = "G " THEN FR(F1,F2) = 48.999 * T(O)
230  IF T$ = "G#" OR T$ = "Ab" THEN FR(F1,F2) = 51.913 *
     T(O)
240  IF T$ = "A " THEN FR(F1,F2) = 55.000 * T(O)
250  IF T$ = "A#" OR T$ = "Bb" OR T$ = "B " THEN FR(F1,F2) =
     58.270 * T(O)
260  IF T$ = "H " THEN FR(F1,F2) = 61.735 * T(O)
```

```
270  PRINT TN$;"!";
     : NEXT
     : PRINT "  -  ";F1
     :F1 = F1 + 1
     : GOTO 130
300  PRINT  CHR$ (4)"STORE SOUND.DATA" -- Frequ. Abspeichern
     : END
500  REM
     ================================================================ 0 =
501  DATA -08,...,...,...,...,...,...,...,...,C 2,...,...,...,...,...
502  DATA  08,...,...,...,...,...,...,...,...,...,E 2,...,...,...,...
503  DATA  08,...,...,...,...,...,...,...,...,...,...,G 2,...,...,...
504  DATA  04,...,...,...,...,...,...,...,...,...,...,...,C 3,...,...
505  DATA  04,...,...,...,...,...,...,...,...,...,...,...,...,C 3,...
506  DATA  04,...,...,...,...,...,...,...,...,...,...,...,...,...,C 3
507  DATA  04,...,...,...,...,...,...,...,C 3,...,...,...,...,...,...
508  DATA  08,...,...,...,...,...,...,...,...,C 3,...,...,...,...,...
509  DATA  08,...,...,...,...,...,...,...,...,...,H 2,...,...,...,...
510  DATA  08,...,...,...,...,...,...,...,...,...,...,C 3,...,...,...
511  DATA  08,...,...,...,...,...,...,...,...,...,...,...,G 2,...,...
```

Programm PLAY1

```
100  & WAVE(0,"/XBASIC/WAVE.TABELS/HOTWAVE")
110  PRINT  CHR$ (4)"RESTORE SOUND.DATA" -- Frequ. Laden
     : PRINT "<I'AM PLAYING>"
120  FOR I = 1 TO AN
     :S1(I) = 200 * 256 -- Volumen = 200
     :S2(I) = 10 * 256 + 7 * 8 + 5 -- Wait/on Shot/Size/Scan
     : NEXT
     : FOR I = 1 TO 8
     :S1(I) = S1(I)
     :S2(I) = S2(I) + 16 * 256 - 0 -- Kanal Links
     : NEXT
125  & SOUNDINIT(15)
130  FOR F3 = 0 TO P(3) - 1 -- Wiederholung 1
     : FOR F4 = 1 TO AN
     : IF FR(F3,F4) <  > 0 THEN Wenn Frequ.=0 dann kein Ton
     & SOUND(F4,FR(F3,F4),S1(F4),S2(F4))
```

```
140  NEXT
     : FOR I = 1 TO 1300 / DA(F3) -- Warte Dauer
     : NEXT
     : NEXT
150  FOR F3 = P(1) TO P(2) - 1 -- Wiederholung 2
     : FOR F4 = 1 TO AN
     : IF FR(F3,F4) <  > 0 THEN
     & SOUND(F4,FR(F3,F4),S1(F4),S2(F4))
160  NEXT
     : FOR I = 1 TO 1300 / DA(F3)
     : NEXT
     : NEXT
170  FOR F3 = P(3) TO P(6) - 1 -- Wiederholung 3
     : FOR F4 = 1 TO AN
     : IF FR(F3,F4) <  > 0 THEN
     & SOUND(F4,FR(F3,F4),S1(F4),S2(F4))
180  NEXT
     : FOR I = 1 TO 1300 / DA(F3)
     : NEXT
     : NEXT
190  FOR F3 = P(4) TO P(5) - 1 -- Wiederholung 4
     : FOR F4 = 1 TO AN
     : IF FR(F3,F4) <  > 0 THEN
     & SOUND(F4,FR(F3,F4),S1(F4),S2(F4))
200  NEXT
     : FOR I = 1 TO 1300 / DA(F3)
     : NEXT
     : NEXT
```

Programm PLAY2

```
100  & WAVE(0,"/XBASIC/WAVE.TABELS/SYNTH")
110  …-- Weiter wie PLAY 1--
```

Programm PLAY3

```
100  & WAVE(0,"/XBASIC/WAVE.TABELS/SYNTH")
     : & WAVE(128,"/XBASIC/WAVE.TABELS/HOTWAVE")
110 …
120  FOR I = 1 TO AN
     :S1(I) = 200 * 256 -- Volumen 200
     :S2(I) = 10 * 256 + 7 * 8 + 5 -- Wait/on Shot/Size/Scan
     : NEXT
     : FOR I = 1 TO 8
     :S1(I) = S1(I) + 128 -- 2 Wavetabel addresse
     :S2(I) = S2(I) + 16 * 256
     : NEXT
125  …-- Weiter wie PLAY 1--
```

Programm PLAY4

```
100  & WAVE(0,"/XBASIC/WAVE.TABELS/NUMBER.WAVE")
     : & WAVE(128,"/XBASIC/WAVE.TABELS/LETTER.WAVE")
110  …
120  FOR I = 1 TO AN
     :S1(I) = 200 * 256 + 128
     :S2(I) = 10 * 256 + 4 * 8 + 3
     : NEXT
     : FOR I = 1 TO 2
     :S1(I) = S1(I) - 50 * 256 - 128 -- Volumen Kleiner
     :S2(I) = S2(I) + 16 * 256 + 8
     : NEXT
     : FOR I = 3 TO 8
     :S1(I) = 0
     :S2(I) = 0
     : NEXT
125  …-- Weiter wie PLAY 1--
```

Anhang A Alphabetische X-Basic Befehlsliste

Alphabetische X-Basic Befehlsliste (Fortsetzung)

30	& MOUSEINIT (L,T TO R,B;x,y)	60
31	& NORMAL	59
32	& OR	46
33	& PANEL ON	53
34	& PEEK bank,addr,P$	66
35	& PIC LOAD name$	51
36	& PIC SAVE name$	51
37	& POKE bank,addr,P$	66
38	& PRINT AT (htab,vtab);text$	56
39	& PULL (page,len)	66
40	& PUSH (page,len)	66
41	& READ x,y,button	60
42	& RESTORE palette,F1…F16	55
43	& ROT= a,b,c	58
44	& SCRN (x,y,color)	50
45	& SOUND END	65
46	& SOUND (A,B,C,D)	61
47	& SOUNDINIT (nr)	61
48	& SPRITE (x,y,def$)	50
49	& STORE palette,F1…F16	55
50	& TEXT	65
51	& TIME (MM,TT,JJ;H,MIN)	67
52	& TWINDOW (htabL,vtabT TO htabR,vtabB)	67
53	& VER= nr	68
54	& WAVE (page,name$)	65

Anhang B Die X-Basic Diskette

Die X-Basic Diskette enthält die folgenden Dateien:

/XBASIC/

1) PRODOS (prodos 8 Ver.1.4)
2) BASIC.SYSTEM
3) XBASIC

/XBASIC/GRAFIK.DEMOS/

1) LINE
2) MOVER1
3) MOVER2
4) MOVER3
5) MOVER4
6) DRAW
7) BOXFILL
8) FILL1
9) FILL2
10) CIRCLE
11) PIECE
12) POLY.1
13) POLY.3MOUSE
14) POLY.8
15) SPRITE
16) DEMO
17) FUNKTION
18) SWEEPER
19) TREFFER

/XBASIC/MUSIK.DEMOS/

1) MAKE.SOUND.DATA
2) SOUND.DATA
3) PLAY1
4) PLAY2
5) PLAY3
6) PLAY4
7) MOOD.MAKE
8) MOOD.DATA
9) MOOD.PLAY

/XBASIC/WAVE.TABLES/

1) HOTWAVE
2) SYNTH
3) NUMBER.WAVE
4) LETTER.WAVE
5) VOICE

/XBASIC/PROGRAM.DEMOS/

(Nicht im Handbuch gelistet !)

1) MUSIC.EDITOR
2) SPRITE.EDITOR
3) XPAINT

HI[illegible] BE[illegible] DET[illegible] ICH[illegible] IE
X-BASIC DISKETTE !

↓

Nur unverletzte Siegel garantieren
den Einwandfreien Zustand der Ware !