oettle & reichler
datentechnik

Schießgrabenstr. 28 a
8900 Augsburg 1

Tel.: (0821) 15 46 32

GLIB

Graphic Extension

Handbuch

Inhaltsverzeichnis:

Einführung:

GLIB stellt ähnlich wie das BIOS eines CP/M-Betriebssystems eine grafische Schnittstelle zwischen Anwenderprogrammen und dem RGB-Grafik-Modul dar. Als Schnittstelle empfängt GLIB grafische Befehle und setzt sie in Aufrufe um, die der Grafik-Prozessor uPD 7220 'versteht'. Dabei wurde nicht einfach irgendeine eigene maßgeschneiderte Grafik-Norm entwickelt, sondern die von Digital Research erstellte Norm GSX-80 (Hier und im weiterem: CP/M, GSX-80 und DR-Graph ist ein Warenzeichen von Digital Research) stand dabei Pate. Da schon viele Programmpakete auf dieser Norm aufbauen und in Zukunft wohl noch sehr viele hinzukommen werden, erschien uns diese Norm als sinnvoll. So besitzen alle Programmprodukte von DR (CB80, MT+, PLI) seit neuestem spezielle Grafik-Erweiterungen, die auf GSX-80 oder eben der GLIB aufbauen. Zudem besitzt GLIB noch eine Reihe von Befehlen, die über die geforderte Norm hinausgehen (Zoom, DMA-Transfers, flexiblere Adressierungsarten, Ellypsen ...).

GLIB belegt ca. 12.5 kByte Speicher und ist in reinem Assembler Z-80-Code geschrieben worden. Dadurch ergibt sich ein sehr kompakter Code unter höchsten Verarbeitungsgeschwindigkeiten im Gegensatz zu ähnlichen Programmpaketen, die meist unter höheren Sprachen erstellt wurden. GLIB ist auf allen Z-80-Systemen (auch nicht CP/M-Systeme) in Verbindung mit dem RGB-Grafik-Modul einsetzbar. Zudem ist GLIB durch seine Trennung in Code- und Datensegmente ROM-Programmierbar. Als RAM-Speicher werden dabei ca. 1.5 kByte benötigt.

Anwender, die bereits über eine GSX-80 Lizenz verfügen, können natürlich GLIB als einen ihrer grafischen Treiber einsetzen. Wir halten jedoch den Erwerb der Lizenz für überflüssig, da alle Funktionen des GDOS-Teils von GSX-80 in der GLIB implementiert sind. Zudem werden in dieser Konstellation die flexiblen Adressierungsarten der GLIB nicht ausgenützt. Spezielle Hilfsprogramme dienen dem Installieren der GLIB (ähnlich GENGRAF unter GSX-80).

Die GLIB - Bestandteile:

GLIB gliedert sich in folgende drei Programmteile:

GLIB: Dieser Teil ist der Rumpf des Grafik-Treibers. Um ihn herum werden je nach Anwendung die beiden andernen Teile angeordnet, die Anwender- und Betriebssystemspezifische Informationen enthalten.

GLIBDATA: Dieser Teil enthält Anwenderspezifische Informationen über das Grafik-System. Hier kann z.B. definiert werden wieviel Farbebenen zur Verfügung stehen, welches Video-Format eingesetzt wird, welche I/O-Ports belegt werden...

GLIBGSX: Soll GLIB als GSX-80 kompatible Version eingesetzt werden, so wird dieser File benötigt, der die grafischen Betriebssystemaufrufe tätigt. Dieser Teil entspricht unter GSX-80 dem GDOS.

Standardanpassung der GLIB:

Standardmäßig wird GLIB mit folgender Konfiguration ausgeliefert:

- Portadressen des RGB-Moduls 30h - 3Fh
- Non-Interlaced Video-Format 704 x 288 Punkte
- Schwarz/Weiß-Betrieb, eine Farbebene (rot) wird angesprochen
- Eingabeeinheiten (Locator, Valuator, Choice und String) über seriellen Tastaturkanal System 8000
- Druckeranpassung (Hardcopy-Funktion) für NEC 8023 oder Itho 8510 604 Punkte horizontal über parallele Centronics-Schnittstelle des Systems 8000

Alle mitgelieferten Programme sind auf diese Werte eingestellt und mit diesen Einstellungen sofort lauffähig. Sollte jedoch eine Anpassung notwendig werden, so muß diese im GLIBDATA-File vorgenommen werden (s. dort).

GLIB-Konfigurationen:

GLIB ist im wesentlichen in vier verschiedenen Konfigurationen einsetzbar:

- Als GSX-80 kompatible Version. Alle Aufrufe werden über BDOS-Calls getätigt. GLIB ist dabei 100-prozentig GSX-80 kompatibel.

- Unter einem bestehendem GSX-80-System kann GLIB als ein grafischer Treiber unter mehreren andern dienen.

- GLIB kann auch direkt mit Anwenderprogrammen verlinkt werden. Dadurch wird eine völlige Betriebssystem-Unabhängigkeit erreicht. Somit ist der Einsatz auch unter nicht CP/M-kompatiblen Betriebssystemen möglich.

- GLIB kann auch in eigenständigen grafischen System in einen ROM gebrannt werden.

GLIB als GSX-80 kompatible Version:

Diese Anwendung dürfte wohl die gebräuchlichste sein. Hiermit wird eine völlig kompatible GSX-80 Betriebssystemumgebung geschaffen. Über das Hilfsprogramm SETGLIB wird GLIB installiert.

```
A>SETGLIB
```

GLIB lagert sich nun als eine 'Resident System Extension' RSX unterhalb des normalen BDOS an. Die verfügbare TPA verringert sich dabei um die Länge der GLIB von ca. 12.5 k. GLIB fängt nun alle BDOS-Calls ab und leitet alle Funktionsnummern ungleich 115 an das eigentliche BDOS weiter. Aufrufe mit einer Funktionsnummer von 115 in Register <C> werden als Grafik-Befehle interpretiert und in die eigentliche GLIB zur Ausführung umgeleitet. Nun kann ein Anwenderprogramm gestartet werden, das sich auf diese Betriebssystemerweiterung stützt.

GLIB stellt dabei für Anwenderprogramme ein GSX-80-System mit erweitertem Funktionsumfang dar. GLIB bleibt nun so lange resis-

tent im Speicher bestehen, bis durch das Hilfsprogramm RESGLIB diese wieder gelöscht wird.

```
A>RESGLIB
```

Dadurch wird GLIB aus dem Speicher entfernt und es steht wieder die volle TPA-Länge zur Verfügung.

Installation einer GSX-kompatiblen GLIB-Version:

SETGLIB wird standardmäßig als schwarz/weiß Version im non-interlaced-Betrieb (704 x 288 Punkte) ausgeliefert. Näheres s. GLIBDATA-File. Sollte eine Anpassung notwendig sein (z.B. auf Farb-Version), so ist ein neuer SETGLIB-File nach folgendem Schema zu erstellen:

1. Der mitgelieferte GLIBDATA.ASM Source-File ist mit Hilfe eines Editors entsprechend zu modifizieren.

2. Assemblieren des GLIBDATA-Files mit dem Assembler RMAC. Soll ein anderer Assembler eingesetzt werden, so sind entsprechende Teile des GLIBDATA-Files auf den neuen Code umzuschreiben.

   ```
   A>RMAC GLIBDATA
   ```

3. Nun muß der neue File mit der eigentlichen GLIB und dem File GLIBGSX mit Hilfe eines Linkers verbunden werden:

   ```
   A>LINK GLIBGSX OP = GLIBGSX,GLIB,GLIBDATA
   ```

   Der Linker schafft nun durch die Anweisung OP in eckigen Klammern aus den drei Eingangsfiles einen File GLIBGSX.PRL, also einen File der 'page relocatable' (=PRL) ist. D.h. dieser File ist noch nicht auf eine bestimmte Speicheradresse fixiert, sondern er kann 256 Bytes - Weise verschoben werden.

4. Dieser neu gewonnene File muß nun in den Namen GLIBGSX.RSX umbenannt werden. Ein evt. schon existierender File mit diesem Namen muß vorher gelöscht werden. Dadurch wird der File GLIBGSX zum Einsatz als resistente Betriebssystemerweiterung RSX vorbereitet (s. CP/M Plus Operating System Hand-

buch 'GENCOM' und 'RSX's').

5. Am Programm SETGLIB befindet sich schon die Standard-GLIB-RSX. Diese muß nun durch folgenden Aufruf vom Rumpfteil des Programmes SETGLIB entfernt werden:

   A>GENCOM SETGLIB

   GENCOM, ein Hilfsprogramm unter CP/M-Plus, entfernt nun die ursprüngliche RSX von SETGLIB.

6. Nun kann die neue GLIB-RSX-Version an SETGLIB gehängt werden. Dies geschieht wiederum unter GENCOM:

   A>GENCOM SETGLIB GLIBGSX

7. Nun steht eine neue SETGLIB-Version bereit, die alle im GLIBDATA-File vorgenommenen Änderungen enthält. Durch den Aufruf von SETGLIB wird GLIB aktiviert:

   A>SETGLIB

Schritt 2 bis 7 laufen automatisch nach dem Aufruf der mitgelieferten Submit-Datei GLIBGSX.SUB ab:

A>SUBMIT GLIBGSX

Inhalt von GLIBGSX.SUB:

```
RESGLIB
RMAC GLIBDATA
LINK GLIBGSX OP = GLIBGSX,GLIB,GLIBDATA
ERA GLIBGSX.RSX
REN GLIBGSX.RSX = GLIBGSX.PRL
GENCOM SETGLIB
GENCOM SETGLIB GLIBGSX
SETGLIB
```


GLIB unter GSX-80:

Anwender die eine GSX-80 Lizenz besitzen und auch andere Grafik-Treiber z.B. für Drucker, Plotter o.ä. einsetzen wollen, können GLIB auch in dieser Konfiguration anwenden. Dazu dient der mitgelieferte File GLIB.PRL. Im ASSIGN.SYS-File ist der Name 'GLIB' unter der Nummer 01 - 10 entsprechend einzutragen. Näheres siehe im GSX-80 Handbuch von Digital Research.

Installation der GLIB unter GSX:

GLIB.PRL wird standardmäßig als schwarz/weiß Version im non-interlaced-Betrieb (704 x 288 Punkte) ausgeliefert. Näheres s. GLIBDATA-File. Sollte eine Anpassung notwendig sein (z.B. auf Farb-Version), so ist ein neuer GLIB.PRL-File nach folgendem Schema zu erstellen:

1. Der mitgelieferte GLIBDATA.ASM Source-File ist mit Hilfe eines Editors entsprechend zu modifizieren. Dabei ist zu beachten, daß die Marke 'GSXKOMP' im GLIBDATA-File unbedingt auf 'FALSE' gesetzt wird, um die Koordinatenumrechnung Virtuell -> Device abzuschalten, da diese Funktion nun vom GDOS des GSX-80 übernommen wird.

2. Assemblieren des GLIBDATA-Files mit dem Assembler RMAC. Soll ein anderer Assembler eingesetzt werden, so sind entsprechende Teile des GLIBDATA-Files auf den neuen Code umzuschreiben.

   ```
   A>RMAC GLIBDATA
   ```

3. Nun muß der neue File mit der eigentlichen GLIB mit Hilfe eines Linkers verbunden werden:

   ```
   A>LINK GLIB OP = GLIB,GLIBDATA
   ```

   Der Linker schafft nun durch die Anweisung OP in eckigen Klammern aus den zwei Eingangsfiles den File GLIB.PRL, so wie ihn GSX-80 benötigt.

Direktes Binden mit Anwenderprogrammen:

Steht kein CP/M Plus System zur Verfügung, so ist der Einsatz der GLIB als GSX-80 kompatible Version nicht möglich. Es besteht jedoch auch die Möglichkeit GLIB direkt mit einem Anwenderprogramm zu einem einzigen Programm zu verbinden (natürlich auch unter CP/M Plus). Dazu dienen die Files GLIB.REL und GLIBDATA.REL. Der Einsprungspunkt für die GLIB ist direkt der Anfang des GLIB.REL-Files. Es kann auch auf die Marke 'GLIB', die als öffentlich ('public') definiert ist, bezug genommen werden. Diese Marke deckt sich mit dem Start des GLIB-Files. Das Anwenderprogramm definiert diese Marke als extern. Durch das Binden des Anwenderprogramms und der beiden GLIB-Teile steht ein fertiges Programm zur Verfügung:

```
A>LINK  YOURFILE.COM L100 = YOURFIL1,YOURFIL2,... ,GLIB,GLIBDATA
```

Installation:

Zur Installation dieser Version muß der GLIBDATA nach den gleichen Regeln wie oben verändert und assembliert werden.

Eigenständiger Einsatz als ROM-Version:

GLIB ist auch in einen ROM programmierbar. Dazu sind die beiden Files GLIB.REL und GLIBDATA.REL evt. mit weiteren Steuerprogrammen zu verlinken. Diese sind in Code- und Datensegmente aufgeteilt. Das Code-Segment (CSEG) kann evt. mit eigenen zusätzlichen Programmen in einen ROM gebrannt werden. Alle Daten-Segmente DSEG müssen im RAM-Bereich des Systems stehen. Die Installation geschieht durch Modifikation des GLIBDATA-Files wie oben.

Befehlsvorrat der GLIB:

Folgendes Kapitel beschäftigt sich noch in sehr allgemeiner Form mit dem Befehlsvorrat der GLIB. Dadurch soll ein Überblick über die grundlegenden Fähigkeiten der GLIB gewonnen werden.

Zeichengrundelemente:

Wichtigste Grundelemente einer grafischen Schnittstelle sind die grafischen Zeichenbefehle, d.h. diejenigen Befehle die den Video-Speicher dirket manipulieren und ein Bild auf dem Schirm erstellen.

1. Line: Das Zeichnen einer Linie gehört zu den wichtigsten Zeichen-Grund-Elementen einer grafischen Ein-Ausgabe dar. Eine Linie wird durch ihren Anfangs- und durch ihren Endpunkt definiert. Das Zeichnen mehrerer Linien in einem einzigen Aufruf ist möglich (Polyline).

2. Text: GLIB kann den standard Ascii-Zeichensatz deutsch/international in 16 verschiedenen Größen und 8 Richtungen schreiben.

3. Füllgebiet: Ein großer Rechenaufwand ist beim Ausfüllen eines Gebiets mit einem bestimmbaren Muster notwendig. Das Gebiet wird durch anzugebende Stützpunkte begrenzt. Es kann unter 66 verschiedenen Füllmustern gewählt werden. Besondere Anforderungen an die Form des Füllgebiets bestehen nicht. GLIB füllt wirklich jede beliebig geformte Figur.

4. Marker-Symbole: Marker-Symbole dienen dem hervorheben oder markieren bestimmter Punkte des Bildschirms. GLIB kennt 6 verschiedenen Markersymbole.

5. Zellgebiet: Die beiden Zell-Funktionen der GLIB dienen dem Auslesen bzw. Einschreiben eines frei definierbaren Musters in den Video-Speicher. GLIB weitet das Muster automatisch auf die geforderte physikalische Größe des Zellen-Gebiets aus, d.h. es paßt die Figur an die Auflösung des Bildschirms an.

6. DMA-Zugriff: Zum schnellen dirketen Ein- und Auslesen des Videospeichers kann dieser dirket zwischen CPU- und Video-Speicher transferiert werden.

7. GDP's: Sogenannte 'General Drawing Primitives' kombinieren die Grundelemente Linie und Füllgebiet um oft benötigte Figuren mit einfachen Aufrufen selbstständig ausführen zu lassen. Es stehen GDP's zum Zeichenen von Balken, Bogenstücken, Kuchenstücken und Kreisen zur Verfügung. Alle Kreisfunktionen können entsprechend verzerrt werden um Ellypsen zu zeichnen.

Steuerelemente:

1. Größe: Die Größe von Text und Marker-Symbolen kann unabhängig voneinander in 16 verschiedene Stufen definiert werden.

2. Farbe: Bei Farbgeräten kann zwischen 8 verschiedenen Farben, bei schwarz/weiß-Geräten zwischen zwei verschiedenen Farben gewählt werden. Die Farbe läßt sich getrennt für Text, Marker, Linie und Füllgebiete definieren.

3. Position: Die Position der auszugebenden Figuren wird direkt beim Aufruf der Funktion festgelegt.

4. Richtung: Die Richtung bestimmt sich meist direkt durch Anfangs- und Endpunkte der Figur. Doch oft ist auch eine explizite Richtungsangabe erforderlich. Diese erfolgt in Winkelgrad im mathematisch positiven Drehsinn.

5. Art: Die Art der zu zeichenden Linien, Texte, Marker und Füllgebiete muß vor dem Aufruf zum zeichnen der Figur getätigt werden. GLIB kennt:

* 66 Füllmuster
* 16 Linienarten
* 8 Textarten
* 6 Markersymbole

6. Schreibmodus: Die Art wie bei Zeichenbefehlen der Video-Speicher manipuliert wird bestimmen folgende vier Schreibmodi:

* Replace
* Transparent
* Complement XOR
* Erase

7. Sonstiges: Natürliche existiert noch eine Vielzahl weiterer Steuerelemente, die sich nicht in so allgemeiner Art definieren lassen.

Eingabeelemente:

GLIB unterstützt auch grafische Eingabeelemente, die interaktive Grafik-Anwendungen möglich machen. Aus welcher Hardwareumgebung die jeweiligen Eingabeelemente kommen lässt sich im GLIBDATA-File bestimmen. Standardmäßig erfolgen alle Eingaben über den seriellen Tastaturkanal des Systems 8000.

1. Locator: Die Locator-Einheit dient zum positionieren eines grafischen Cursors auf dem Bildschirm. Allgemein lassen sich mit der Locator-Einheit bestimmte Positionen des Bildschirms angeben. Locator-Einheiten können eine Tastatur, ein Lichtgriffel, eine Maus, ein Joystick oder ein grafisches Tablett sein.

2. Valuator: Die Valuator-Einheit dient zum Verändern von Variablen durch aufaddieren oder subtrahieren. Normalerweise dienen als Valuator-Einheit bestimmte Funktionstasten einer Tastatur. Z.b. erhöht die 'Pfeil nach oben Taste' den Wert des Valuators, die 'Pfeil nach unten Taste' ernierdrigt seinen Wert.

3. String: Die String-Einheit ermöglicht die Eingabe von ASCII-Zeichen oder Texten ähnlich wie die CP/M-Console-Einheit.

4. Choice: Die Choice-Einheit löst bestimmte Funktionen im Anwenderprogramm aus. Dazu dienen in der Regel bestimmte Funktionstasten der Tastatur.

Adressierungsarten:

Ein immer wieder auftretendes Problem stellt die Adressierung bestimmter Bildschirmpositionen z.B. zum Zeichnen einer Linie dar. Als Grundregel gilt: Das linke untere Eck des Bildschirms entspricht dem Punkt (0,0). Adressiert wird im mathematisch gebräuchlichen Sinn. D.h. die X-Achse ist die horizontale und die Y-Achse ist die vertikale Achse des Schirms. Steigende X-Werte befinden sich also rechts vom Ursprung, steigende Y-Werte oberhalb vom Ursprung.

Längen:

Der weiter unten erwähnten Koordinatenumrechnung unterliegen alle Eingangskoordinaten im ptsin-Feld und alle Ausgangskoordinaten im ptsout-Feld. Die relative Adressierungsart ist jedoch nur bei Koordinatenangaben sinnvoll. Ausnahmen bestehen daher in bestimmten Fällen bei Längenangaben (Texthöhe, Markerhöhe, Radiusangaben). Daher unterbleibt bei Längenangaben automatisch die relative Koordinatenumrechnung des ptsin-Feldes. Die Umrechnung virtuell - device findet jedoch nach wie vor statt.

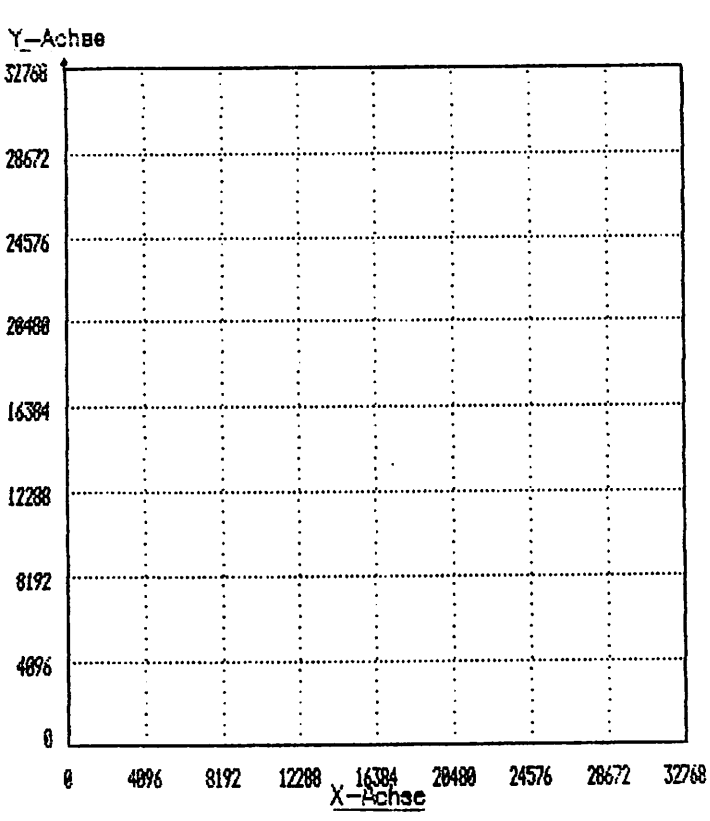
Winkel:

Bestimmte Funktionen erfordern die Angabe eines Winkelwertes. Diese Angaben erfolgen alle im mathematisch positiven Sinne. D.h. 0 Grad entspricht der Ost-Richtung einer Landkarte, 90 Grad entspricht Norden usw. Alle Winkelangaben müssen in zehntel Grad (0 - 3600) erfolgen.

GLIB kennt vier verschiedene Adressierungsmethoden, die sich durch ESC-Befehle umschalten lassen:

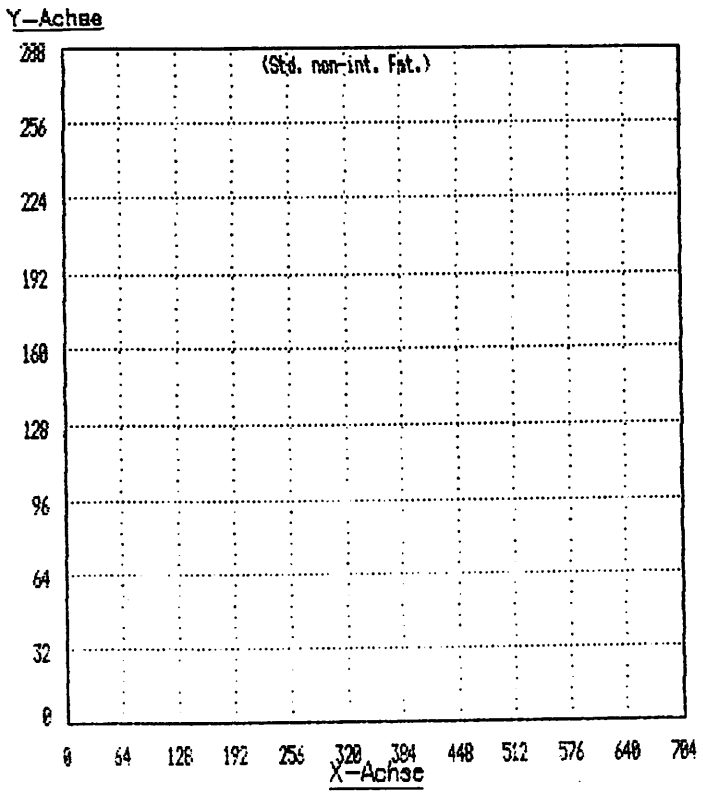
1. Virtuell - absolut:

In der virtuellen absoluten Adressierungsart steht dem Programmierer ein 'virtueller' Adressierungsraum von 32768 x 32768 Punkten zur Verfügung. Dabei braucht man sich um die tatsächlich existierenden Bildpunkte nicht zu kümmern. GLIB übernimmt die Umrechnung der Koordinaten auf die zur Verfügung stehenden Bildpunkte. Dadurch läßt sich eine Programmierung erreichen, die weitgehend vom gewählten Bildschirmformat unabhängig ist. Eine Koordinate von (0,0) entspricht in jedem Falle dem linken unteren Eck des Schirms, eine Koordinate von (16384,16384) exakt der Mitte des Bildschirms usw.



2. Device - absolut:

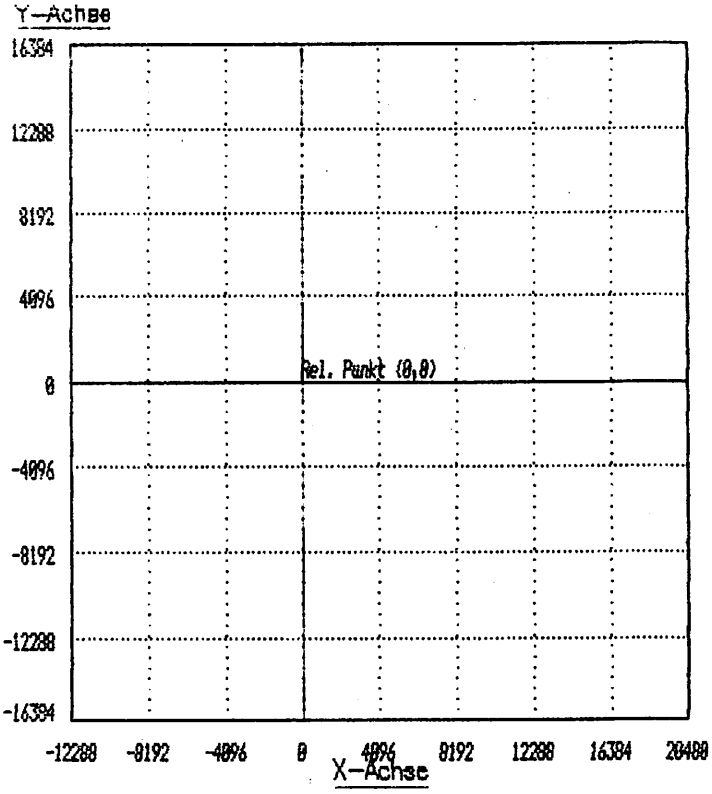
Diese Adressierungsart unterläßt die Umrechnung der virtuellen Koordinaten in die tatsächlich vorhandenen Bildpunkte. Jede Koordinate wird als absolut angesehen, d.h. sie entspricht direkt dem Bildpunkt auf dem Schirm. Das linke untere Eck des Schirms entspricht in jedem Fall dem Punkt (0,0), das rechte obere Eck des Schirms entspricht dem Punkt (XPIX,YPIX).



3. Virtuell - relativ:

In dieser Adressierungsart steht dem Programmierer auch ein virtueller Adressraum zur Verfügung, der auf die tatsächlich existierenden Bildpunkte zurückgerechnet wird. Doch befindet sich der Ursprung (0,0) des Koordinatensystems nicht im linken unteren Eck des Bildschirms, sondern alle Angaben beziehen sich auf einen frei wählbaren relativen Stützpunkt, der den Ursprung darstellt. Daher sind auch negative Koordinaten zulässig, je nach Wahl des relativen Stützpunkts. Der große Vorteil dieser Adressierungsart liegt in der allgemeinen Definition einer Figur, die durch entsprechendes Verschieben des Stützpunkts auf jede beliebige Stelle des Schirms plaziert werden kann. Wird der Stützpunkt verschoben, so unterliegt seine Adressierung auch der relativen Koordinatenumrechnung, d.h. relative zum vorhergehenden Stützpunkt.

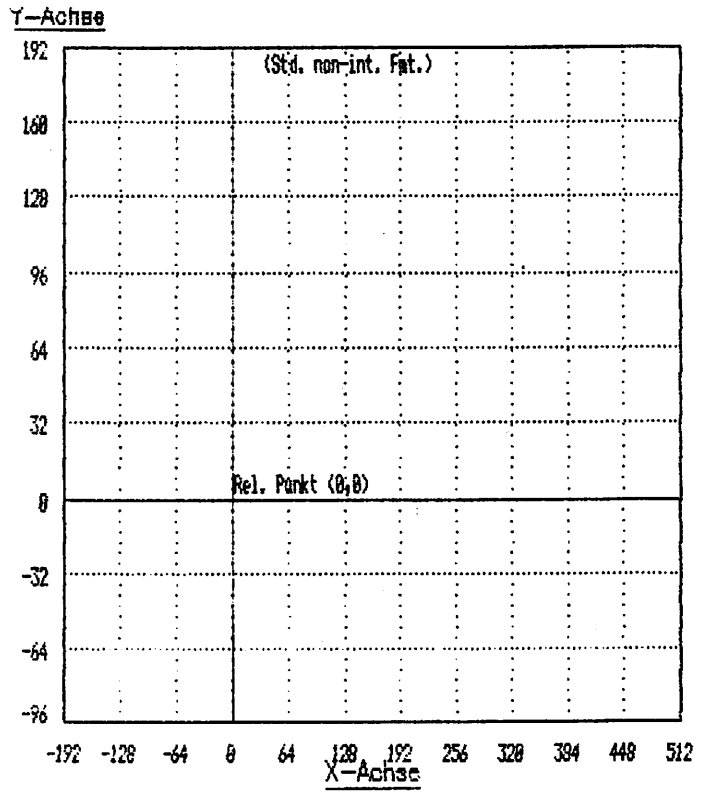
3: Virtuell – relative



4. Device - relativ:

Diese Methode entspricht Adressierungsart 3 mit dem Unterschied, daß alle Koordinaten keiner virtuellen - device Umrechnung unterworfen werden (s. 2).

4: Device – relative



Verdoppelung des Y-Adreßbereichs durch 'DOUBLEY':

Standardmäßig arbeitet GLIB mit einer Bildschirmauflösung von 704 x 288 Bildpunkten, die den nach der CCIR-Norm arbeitenden Bildschirmgeräten (50 Hz Bildwechsel, 15,7 kHz Zeilenfrequenz) angepaßt ist. Wünschenswert wäre jedoch eine höhere Zeilenauflösung, um einen quadratisch skallierten Bildpunktabstand (dx/dy = 1) zu erreichen. Diese Forderung ist jedoch für CCIR-Schirme nur durch Anwendung der Interlaced-Technik (Zwei-Bild-Verfahren) zu erreichen. Die Interlaced-Technik zeigt jedoch starke Flimmererscheinungen, da nur noch mit einer Zeilenfrequenz von 25 Hz pro Halbbild gearbeitet wird. Auch der Einsatz von Bildschirmgeräten mit langnachleuchtendem Phosphor (P39) zeigt noch leichtes Flimmern.

Aus diesem Grund wird wohl der Einsatz des Standardformats von 704 x 288 Punkten am meisten angewendet werden. Der Nachteil dieser Methode ist nicht zu übersehen: Alle Y-Werte erscheinen auf dem Bildschirm um das doppelte gestreckt. Ein Linie mit der Steigung 1 (dy/dx = 1) erscheint nicht unter einem Winkel von 45 Grad, sondern entsprechend steiler. Ein Kreis erscheint nicht rund sondern als seitlich abgeflachte Ellypse (daher wurde in der GLIB nicht einfach die Kreisfunktionen des GDC 7220 herangezogen, GLIB errechnet einen Kreis softwaremäßig).

Doch GLIB bietet eine besondere Option an. Im GLIBDATA-File kann eine Marke 'DOUBLEY' umgesetzt werden. Ist diese Marke auf einen Wert ungleich 0 gesetzt, so verdoppelt GLIB den Y-Adreßraum. Anwenderprogrammen erscheint es nun, als lägen doppelt so viele in Y-Richtung adressierbare Punkte vor. Ein tatsächlich vorhandenes Bildschirmformat von 704 x 288 Punkten erscheint bei dieser Option als 704 x 576 Punkte. Anwenderprogramme können nun mit diesem Adreßbereich rechnen, der nach entsprechender Justage des Bildschirms quadratisch skallierte x- und y-Abstände zeigt. GLIB rechnet unmittelbar vor dem Zeichnen einer Figur die Y-Koordinate auf die tatsächlichen Bildpunkte um (/2). Dadurch reduziert sich die wirkliche Auflösung natürlich um die Hälfte, also auf die tastächlich vorhandenen Bildpunkte. Geraden mit der Steigung 1 erscheinen nun unter einem Winkel von 45 Grad und Kreise erscheinen 'kreisrund'.

Bildschirm-Farben:

Die Standardversion der GLIB arbeitet mit einer monochromen RGB-Grafik-Karte zusammen. Bei Farbausführungen wird nur die Rot-Ebenen angesprochen, daher ist eine Anpassung im GLIBDATA-File erforderlich.

Farben werden bei der GLIB über eine Farbindextabelle adressiert. Jedem Farb-Index 0-7 (0-1) läßt sich ein bestimmter Farbwert zuordnen, der in der Farb-Index-Tabelle definiert ist. GLIB stellt Befehle zum umdefinieren der Farbindex-Tabelle zur Verfügung. Beim Erst-Aufruf der GLIB (Open Glib) ist die Farb-Index-Tabelle folgendermaßen definiert:

Voreinstellung der Farb-Index-Tabelle:

Monochrom:	Index:	Farbe:
	0	black
	1	white (=red)

Farbe:	Index:	Farbe:
	0	black
	1	red
	2	green
	3	blue
	4	cyan
	5	yellow
	6	magenta
	7	white
	8-n	white

Schreibmodi:

GLIB kennt vier verschiedene Schreibmodi. Beim Erstaufruf ist Schreibmodus 1 (replace) eingestellt. Die Schreibmodi können durch eine Art boolsche Funktion erklärt werden. Dabei bedeudet:

Mask:	Das Zeichenpattern z.B. einer Linie
Old:	Das ursprüngliche Pixelmuster des Videospeichers
Fore:	Die per Frabindex angwählte Zeichenfarbe
Back:	Die Hintergrundfarbe (entspricht immer Farbindex 0)
New:	Das Pattern des Videospeichers nach der Operation

Replace:

Im Replace-Modus spielt das ursprüngliche Bit-Muster des Video-Speichers keine Rolle. Das neue Bitmuster ersetzt das ursprüngliche Muster völlig. Stellen, an denen das neue Bitmuster auf 0 steht werden aktiv mit der Hintergrundfarbe (=BACK) Index 0 belegt. Stellen an denen die Mask den Wet 1 besitzt werden auf die momentane Schreibfarbe (=FORE) gesetzt.

NEW = (FORE and MASK) or (BACK and not MASK)

Transparent:

Im Transparent-Modus werden nur diejenigen Pixel verändert, wo die Pixel-Maske auf 1 steht. Dies ist der schnellste Schreibmodus, da die wenigsten Schreibbefehle gegeben werden müssen. Dieser Modus sollte daher nach Möglichkeit immer eingesetzt werden:

NEW = (FORE and MASK) or (OLD and not MASK)

Komplement:

Dieser Modus, auch XOR-Modus genannnt, führt eine logische XOR-Operation zwischen neuem und unrsprünglichen Bildmuster durch. Zweimaliges anwenden dieser Funktion stellt wieder den Ausgangszustand her.

NEW = (FORE and MASK) xor OLD

Erase:

Im Erase-Modus (=Löschen) wird unabhängig von der gewählten Vordergrundfarbe die Punkte auf die Hintergrundfarbe gesetzt, bei denen die Maske den Wert 1 hat.

NEW = (BACK and MASK) or (OLD and not MASK)

Geschwindigkeitsoptimiertes Zeichnen:

Die Geschwindigkeit, mit der die Zeichenoperationen ablaufen, hängt natürlich von der Geschwindigkeit des Steuerprozessors, der Software, und der Komplexität der Figur ab. Auf diese Parameter hat der Programmierer keinen Einfluß. Hinzu kommen jedoch bestimmte Größen, die durch geschicktes programmieren den Zeitbedarf zum Aufbau einer Figur deutlich herabsetzen können:

Schreibmodus:

Der gewählte Schreibmodus hat eine sehr große Bedeutung in Bezug auf die Geschwindigkeit. Im Replace- und im Erase-Modus wird für alle drei Farbebenen getrennt eine Schreiboperation durchgeführt, unabhängig von der momentanen Schreibfarbe (jede Farbe wird entweder gesetzt oder gelöscht). Im Transparent-Modus und Complement-Modus hingegen werden nur für die momentanen Vordergrundfarben Schreib-Operationen durchgeführt. Dieser Modus bringt erhebliche Geschwindigkeitsvorteile bei Anwahl nur einer Grundfarbe oder zweier Mischfarben mit sich.

Farbe:

Im Replace-, Erase-Modus wird für jede Farbebene getrennt ein Schreibbefehl durchgeführt. Die beiden anderen Schreibmodi hingegen zeichnen nur die aktive Vordergrundfarben. Wird z.B. Farbe Rot angewählt, so muß dafür nur eine Schreiboperation durchgeführt werden. Bei Farbe Gelb sind dazu schon zwei Schreiboperationen nötig (Rot und Grün). Bei Farbe Weiß sind drei Schreiboperationen nötig, dies benötigt also die gleiche Zeit wie im Replace- oder Erase-Modus. D.h. im Transparent- und im Complement-Modus bestimmt die angewählte Schreibfarbe die Geschwindigkeit des Bildaufbaus.

Füllmuster:

Zum Füllen stellt GLIB ein Vielzahl verschiedener Füllmuster bereit, deren Zeichengeschwindigkeit sich in besonderen Fällen erheblich beschleunigen lassen:

Hollow: Natürlich benötigt diese Füllart am wenigsten Zeit, da die Figur nicht gefüllt sondern nur umrandet wird.

Solid: Diese Füllart ist die schnellste Füllmethode überhaupt. Hier wird zum Füllen des Gebiets nicht die Linienfunk-

tion des GDC 7220 angesprochen, sondern ein spezielles WDAT-Kommando. Beim Zeichnen einer Linie schreibt der GDC einen Punkt pro Schreibzyklus. Nach dem Beenden der Linie muß die nächste Farbe angewählt werden und ein neuer Schreibbefehl wird gestartet. Doch der GDC bietet auch die Möglichkeit mit einem Schreibbefehl ein Wort d.h. 16 Pixel auf einmal zu schreiben. Bedingung dafür ist, daß alle Bits des Bitmusters auf 0 oder 1 gesetzt sind. Dies genau ist bei dieser Füllart der Fall, da kein Muster geschrieben sondern das ganze Füllgebiet auf 1 gesetzt wird. Ein zweiter erheblicher Geschwindigkeitsvorteil besteht darin, daß durch die besonderen Hardware-Gegebenheiten des RGB-Grafik-Moduls in diesem Fall gleich alle angewählten Farbebenen in einem Schreibbefehl parallel gesetzt werden könnne, d.h. die Schreibbefehle für die einzelen Farben müssen nicht hintereinandergeschaltet werden. GLIB erkennt diesen Fall und füllt das Gebiet entsprechend schneller. Ein Einschränkung jedoch besteht dabei. Das WDAT-Kommando kann nur angewendet werden, falls der Start des zu füllenden Gebiets auf eine ganze Wortadresse des GDC fällt (Dot-Adress = 1) und mit einer Wortadresse endet. Ist dies nicht der Fall, so zeichnet GLIB das Füllgebiet bis zur nächsten ganzen Wortadresse konventionell per Linienkommando, dann per WDAT-Kommando unter paralleler Anwahl aller benötigten Farben bis zur letzten ganzen Wortadresse und beendet schließlich das Füllen durch einen Linien-Befehl.

Hatch: Ähnliche Fälle wie unter der oben genannten Füllart treten bei den Schrafurmustern auf. Schrafurart 1, 8, 25 und 32 setzen sich aus horizontalen Linien zusammen, die durchwegs auf 1 oder 0 gesetzt sind. Diese Fälle erkennt GLIB und führt das beschleunigte WDAT aus.

Halftone: Ähnliches gilt für die Grauwerte-Füllmuster, bei denen Teile des Musters aus durchgezogenen 0 oder 1 Linien aufgebaut ist (z.B. 1, 2, 4, ...).

Display-Off:

Durch das Abschalten des Bildschirms (ESC 23) wird erreicht, daß der GDC nicht die inaktiven Blanking-Phasen (Dunkelsteuerung des Bildes an den Rändern) des Bildes abwarten muß, um den Video-Speicher zu manipulieren. Nachdem das Bild steht, kann der Bildschirm über ESC 22 wieder eingeschaltet werden.

Aufruf-Konvention der GLIB:

Alle GLIB-Funktionen werden über einen einzigen Einsprungpunkt aus aufgerufen. In GSX-kompatibler Version ist dies die BDOS Funktion 115, beim direkten Binden mit Anwenderprogrammen ist dies direkt der Start der GLIB (erster Op-Code), die auch über die als öffentlich definierte Marke 'GLIB' erreichbar ist.

Registerpaar <DE> muß beim Einsprung auf den sogenannten Parameter Block zeigen, der die Adressen verschiedener Ein- und Ausgabefelder zum Austausch von Informationen enthält. Ein Parameter Block ist folgendermaßen aufgebaut:

Parameter Block:

PB:	Adresse des Control-Feldes 'contrl'
PB+2:	Adresse des Eingangs-Parameter-Feldes 'intin'
PB+4:	Adresse des Eingangs-Koordinaten-Feldes 'ptsin'
PB+6:	Adresse des Ausgangs-Parameter-Feldes 'intout'
PB+8:	Adresse des Ausgangs-Koordinaten-Feldes 'ptsout'

Alle Daten die über diese Felder übergeben werden sind 16 Bit Integer Zahlen (Ausnahme DMA Read, Write).

Da GLIB intern immer mit absoluten device Koordinaten rechnet, erfolgt als erstes immer eine Koordinatenumrechnung. Wieviele Koordinaten übergeben werden steht in contrl(2). Das aufrufende Programm muß in jedem Fall diesen Wert auf die richtige Größe setzen. Werden keine Koordinaten übergeben, so muß hier der Wert Null eingetragen werden. Entsprechendes gilt für die GLIB, die durch setzen von contrl(3) angibt, wieviele Ausgangskoordinaten umgerechnet und übergeben werden.

Control-Feld 'contrl':

Dieses sog. Control-Feld enthält Informationen über die Art des Befehls, wieviele Eingangskoordinaten und wieviele Ausgangskoordinaten übergeben werden und sonstige Steuerungsinformationen.

contrl(1)	Das erste Wort bestimmt den auszuführen Op-Code
contrl(2)	Dieses Wort enthält die Anzahl der Eingangs-Koordinatenpaare im ptsin-Feld. Jedes Koordinatenpaar besteht aus einer X- und einer Y-Koordinate. Daher

enthält das ptsin-Feld doppelt soviel Worte als in contrl(2) angegeben oder 4mal so viel Bytes.

contrl(3) Dieses Wort wird immer von der GLIB gesetzt und enthält ähnlich wie contrl(2) die Anzahl der Ausgangskoordinatenpaare im ptsout-Feld.

contrl(4-n) Diese Stellen enthalten Op-Code abhängige Informationen

Eingangsparmeter-Feld 'intin':

Dieses Feld 'intin' (= integer in) enthält allgemeine Parameter zu Steuerungszwecken, und muß je nach Op-Code gesetzt werden

Eingangskoordinaten-Feld 'ptsin':

Diese Feld 'ptsin' (= points in) enthält die Eingangskoordinatenpaare des Anwenderprogramms für die GLIB. Wieviel Koordinatenpaare übergeben werden bestimmt contrl(2). Alle Werte im ptsin-Feld unterliegen der oben aufgeführten Koordinatenumrechnung.

Ausgangsparameter-Feld 'intout':

Dieses Feld dient der GLIB zum Übergeben von Parametern an das Anwenderprogramm nach einem Rücksprung aus der GLIB.

Ausgangskoordinaten-Feld 'ptsout':

Über dieses Feld übergibt GLIB Koordinatenpaare an das aufrufende Programm. Wieviele Paare übergeben gibt GLIB in contrl(3) an. All diese Koordinaten unterliegen ebenfalls einer Koordinatenumrechnung im umgekehrten Sinne zur Eingangskoordinatenumrechnung, d.h. z.B. die Device-Koordinaten der GLIB werden wieder auf virtuelle Koordinaten zurückgerechnet, absolute Koordinaten werden wieder auf relative Koordinaten umgerechnet.

Installation der GLIB über den GLIBDATA-File:

Sollte die Standard-Version der GLIB nicht den Anforderungen des Anwenders entsprechen, so muß GLIB durch ändern des GLIBDATA-Files angepßasst werden. Der GLIBDATA-File enthält Initialisierungsparameter verschiedenster Art, die sich nach den Bedürfnissen des Anwenders verändern lassen.

GLIBDATA liegt im GLIBDATA.ASM-File in Assembler-Source-Form vor. Dieser Source File muß verändert werden, durch den Assembler RMAC assembliert werden und schließlich mit der eigentlichen GLIB durch eine Linker verbunden werden.

1. Einfache Anpassung über Equate-Anweisungen:

Am Anfang des GLIBDATA-Files befinden sich einige Equate-Anweisungen, mit deren Hilfe sich auch für in Assemblerprogramierung ungeübte Anwender die GLIB rasch und einfach anpassen läßt. Durch einfaches umsetzen der 'TRUE' in 'FALSE' Anweisungen werden bestimmte Teile der GLIB automatisch durch 'conditionaling assembling' umgeschrieben.

Port-Base-Adresse: Das RGB-Grafik-Modul, mit der die GLIB zusammenarbeitet, belegt 16 aufeinanderfolgende Portadressen. Die Standardausführung belegt die Adressen 30h bis 3Fh. In Systemen mit mehereren RGB-Karten z.B. zur Trennung der alphanumerischen von der grafischen Ausgabe kann ein Umsetzen dieser Adresse z.B. auf 0F0h erforderlich werden. Dazu ist diese Basis-Adresse entsprechend bei der Marke 'RGBBASE' anzupassen.

Interlaced-Betrieb: Im GLIBDATA-File sind zwei verschiedene Videoformate definiert. Das Standard-Format arbeitet im non-interlaced Betrieb mit einer Auflösung von 704 x 288 Punkten. Durch Angabe von 'TRUE' bei der Marke 'INTERLCD' wird auf das zweite vordefinierte Bildschirmformat umgeschaltet. Es arbeitet mit einer Auflösung von 608 x 429 Punkten im Interlaced-Betrieb. Allerdings ist dabei die Verwendung eines lang nachleuchtenden Monitors sehr empfehlenswert. Beide Bildschirmformate dienen zur

Ansteuerung eines Monitors mit 50 Hz Bildwechselfrequenz und einer Zeilenfrequenz von 15.7 kHz (GDC Taktrate 16 MHz).

Farb-Betrieb: GLIB benötigt Informationen, wieviele Farbebenen vorliegen. Standardmäßig ist GLIB auf eine schwarz-weiß Version entsprechend einer Farbebene voreingestellt. Durch umsetzen der Marke 'RGBCAP' auf 'TRUE' wird der GLIBDATA-File auf die 8-Farben-Version umgestellt (3 Farbebenen).

CPU/EPC-System: GLIB benötigt weiter hardwareabhängige Informationen über die Druckeransteuerung und die Eingabekanäle des Systems. Durch setzen der Marke 'CPU' auf 'TRUE' wird im GLIBDATA-File der Betrieb unter dem CPU-Modul eingestellt. Erfolgt die Angabe von 'FALSE' (std.), so erfolgt die Einstellung auf den Einplatinenrechner EPC. Arbeitet GLIB mit Steuermodulen anderer Hersteller zusammen, so ist diese Angabe unnötig, da weitergehende Änderungen im GLIBDATA-File notwendig werden.

Double-Y: Durch setzen der Marke 'DOUBLEY' auf 'FALSE' unterbleibt die Verdoppelung des Y-Adressebereichs um einen quadratisch skallierten Punkteabstand zu erhalten. Durch setzen der Marke auf 'TRUE' wird diese GLIB-Option eingeschaltet.

GSX-kompatibel: Je nach Anwendung unterscheiden sich die Adressierungsarten der GLIB beim Start des Treibers. Soll eine GSX-kompatible Version erstellt werden, so ist bei der Marke 'GSXKOMP' der Wert 'TRUE' einzutragen. Soll GLIB als ein Treiber unter einem bestehendem GSX-System arbeiten so ist die Marke mit 'FALSE' zu kennzeichnen.

2. Adressierungsart beim Start:

Unter der Marke 'VIRSTR' und 'ABSSTR' kann angegeben werden, mit welcher Adressierungsart die GLIB beim Start (Open-Glib) arbeiten

soll. Ist 'VIRSTR' auf einen Wert gleich Null gesetzt, so wird beim Start der device-Adressmodus eingeschaltet, alle anderen Werte schalten den virtuellen Adressmodus ein. Ist 'ABSSTR' auf einen Wert gleich Null gesetzt, so wird ist beim Start der relative Adressmodus eingstellt, andernfalls der absolute Modus. Je nach Anwendungsfall können diese Marken umgesetzt werden.

3. Video-Format:

Standardmäßig sind im GLIBDATA-File zwei Video-Formate vordefiniert, die durch die Marke 'INTERLCD' umgeschaltet werden können. Sind andere Formate gewünscht (andere Zeilen- und Bildwechselfrequenzen des Monitors, anderer GDC-Takt, andere Auflösung) so müssen die horizontal- und vertical-Synchronwerte umdefiniert werden. Im GLIBDATA-File kann das gesammte Reset-Kommando an den GDC 7220 durch relativ einfache Equate-Anweisungen umdefiniert werden. Dazu ist allerdings ein Studium des Datenblatts des GDC's notwendig.

4. Eingabe-Einheiten:

Die vier Eingabeinheiten der GLIB (Locator, Valuator, String, Choice) lassen sich über den GLIBDATA-File an die spezielle Hardware-Umgebung des Anwenders anpassen. Standardmäßig führen alle vier Einheiten auf den seriellen Tastaturkanal des Systems 8000 (serieller STI-Kanal). Zudem sind die Funktionstasten der Valuator, Locator und Choice-Einheit per Tabellen definierbar.

5. Druckeranpassung:

Die Hardcopy-Funktion der GLIB arbeitet standardmäßig mit einem Drucker NEC 8023 oder Itho 8510 zusammen. Der Drucker wird über den parallelen Centronics-Kanal des Systems 8000 angesprochen. Dies kann durch anpassen der Drucker-Ausgaberoutinen geändert werden. Andere Drucker benötigen evt. andere Kodes zur Umschaltung in den Grafik-Modus. Dies geschieht durch ändern der Werte in der Tabelle 'LSTINI'.

****
OPEN GLIB: Initialisieren des GLIB-Treibers
****

OPEN GLIB initialisiert den Treiber mit den Werten des 'GLIBDATA'-Files und des Eingabe-Feldes 'intin'. Die CP/M-Konsole wird über 'TRMINI' initilisiert (s. GLBDATA-File). Das RGB-Modul wird entsprechend programmiert und der Bildschirm wird gelöscht. Informationen über den Treiber werden über die Felder 'intout' und 'pstout' an das aufrufende Programm weitergegeben.

Input:	contrl(1)	Opcode = 1
	contrl(2)	0
	intin	Voreinstellungen für GLIB
	intin(2)	Linien-Typ
	intin(3)	Polyline Farb-Index
	intin(4)	Marker Typ
	intin(5)	Polymarker Farb-Index
	intin(6)	Text-Art
	intin(7)	Text Farb-Index
	intin(8)	Fill interior style
	intin(9)	Fillstyle Index
	intin(10)	Fillcolor Index
Output:	contrl(3)	6
	contrl(5)	Länge von intout = 45
	intout	Konstanten von GLIB
	intout(1)	Max. adrs. Punkte horizontal (Start = 0)
	intout(2)	Max. adrs. Punkte vertikal (Start = 0)
	intout(3)	= 1 (CRT-Device)
	intout(4)	Breite eines Punktes in Micrometer
	intout(5)	Höhe eines Punktes in Micrometer
	intout(6)	Anzahl der Zeichen-Größen = 16
	intout(7)	Anzahl der Linien-Arten = 16
	intout(8)	Anzahl der Linien-Breiten = 1
	intout(9)	Anzahl der Marker-Typen = 6
	intout(10)	Anzahl der Marker-Größen = 16
	intout(11)	Anzahl der Text-Arten = 8
	intout(12)	Anzahl der Grauwerte = 32
	intout(13)	Anzahl der Schrafur-Arten = 32
	intout(14)	Anzahl der Farben (2 bei SW, 8 bei RGB)
	intout(15)	Anzahl der 'general drawing primitives'

	(GDP's) = 4
intout(16-25)	Liste der GDP-Arten:
intout(16)	Bar GDP = 1
intout(17)	Arc GDP = 2
intout(18)	Pie slice GDP = 3
intout(19)	Circle GDP = 4
intout(20-25)	= -1, keine weiteren GDP's
intout(26-35)	Liste der GDP-Attribute:
intout(26)	Bar-GDP = Area-Fill = 3
intout(27)	Arc-GDP = Poly-Line = 0
intout(28)	Pie-GDP = Area-Fill = 3
intout(29)	Circle-GDP = Area-Fill = 0
intout(30-35)	= -1, keine weiteren GDP's
intout(36)	Farb-Flag: = 0 bei S/W, = 1 bei RGB
intout(37)	Text-Rotation-Capability-Flag: = 1
intout(38)	Area-Fill-Flag: = 1
intout(39)	Read-Cell-Array-Flag: = 1
intout(40)	Anzahl der gleichzeitig darstellbaren Farben (2/8)
intout(41)	Anzahl der Locator-Einheiten: = 1
intout(42)	Anzahl der Valuator-Einheiten: = 1
intout(43)	Anzahl der Choice-Einheiten: = 1
intout(44)	Anzahl der String-Einheiten: = 1
intout(45)	Workstation Type: = 1 (Input/Output)
ptsout(1)	= 0
ptsout(2)	= minimale Zeichen-Höhe
ptsout(3)	= 0
ptsout(4)	= maximale Zeichen-Höhe
ptsout(5)	= minimale Linien-Breite
ptsout(6)	= 0
ptsout(7)	= maximale Linien-Breite
ptsout(8)	= 0
ptsout(9)	= 0
ptsout(10)	= minimale Marker-Höhe
ptsout(11)	= 0
ptsout(12)	= maximale Marker-Höhe

Voreinstellungen bei Erst-Aufruf der GLIB:

Alpha/Graphic:	Graphic-Modus
Text-Größe:	minimal
Text-Richtung:	90 Grad

Linien-Breite:	minimal
Marker-Größe:	minimal
Schreib-Modus:	replace
Input-Modus:	Request-Modus für alle vier Eingabe-Einheiten
Zoom:	x 1
Pan:	linker oberer Bildschirmrand
Adressierung:	Virtuell absolut (s. GLIBDATA)
Relativer Punkt:	X = 0, Y = 0

****

CLOSE GLIB: Beenden der graphischen Ein/Ausgabe

****

Diese Funktion löscht den Bildschirm, schaltete den Alpha-Modus ein und reinitialisiert die CP/M-Konsole über 'TRMEXI' (s. GLIBDATA-File). Dies Funktion sollte immer als letzter Zugriff auf die GLIB aufgerufen werden.

Input:	contrl(1)	Opcode = 2
	contrl(2)	= 0
Output:	contrl(3)	= 0

****

CLEAR GLIB: Löschen des Bildschirms

****

Diese Funktion reinitialisiert das RGB-Modul (Bildschirm löschen). Die Alpha-Cursor-Position und die Pan-Position wird auf 0,0 zurückgesetzt.

Input:	contrl(1)	Opcode = 3
	contrl(2)	= 0
Output:	contrl(3)	= 0

****
UPDATE GLIB: Darstellen aller graphischen Ausgaben:
****

Diese Funktion wurde implementiert um Kompatibilität mit dem GSX-Standard zu bewahren. Sie hat für die GLIB keine Bedeutung, da alle graphischen Ausgaben sofort dargestellt werden.

Input:	contrl(1)	Opcode = 4
	contrl(2)	= 0
Output:	contrl(3)	= 0

****
ESCAPE:
****

Allgemein:

Input:	contrl(1)	Opcode = 5
	contrl(2)	Anzahl der Eingangs-Koordinaten
	contrl(6)	ESCAPE-Funktionsnummer
	intin	Funktionsabhängige Parameter
	ptsin	Eingangs-Koordinaten
Output:	contrl(3)	Anzahl der Ausgabe-Koordinaten
	contrl(5)	Anzahl der Ausgabe-Parameter
	intout	Feld für Ausgangs-Parameter
	ptsout	Feld für Ausgangs-Koordinaten

****
ESC1: Abfrage der adressierbaren Zeichen-Zellen:
****

Diese Funktion übergibt dem aufrufendem Programm die Anzahl der Reihen (row's) und Spalten (column's) auf die der Alpha-Cursor positioniert werden kann.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0

	contrl(6)	Function ID = 1
Output:	contrl(3)	= 0
	intout(1)	Anzahl der Reihen - 1 (Start = 0)
	intout(2)	Anzahl der Spalten -1 (Start = 0)

****

ESC2: Einschalten des graphischen Betriebsart

****

Diese Funktion setzt die GLIB in den grafischen-Modus, gleichzeitig wird der Bildschirm gelöscht. Der einzige Unterschied zwischen Alpha- und Graphic-Modus besteht darin, daß im Alpha-Modus die augenblickliche Cursor-Position durch invertieren des darunterliegenden Zeichens hervorgehoben wird. Im Graphic-Modus unterbleibt diese Anzeige.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0
	contrl(6)	= Function ID = 2
Output:	contrl(3)	= 0

****

ESC3: Abschalten des graphischen Betriebsart

****

Diese Funktion schaltet den Alpha-Modus ein, d.h. die momentane Alpha-Cursor-Position wird durch invertieren des darunterliegenden Zeichens angezeigt. Zudem wird der Bildschirm gelöscht.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0
	contrl(6)	= Function ID = 3
Output:	contrl(3)	= 0

****
ESC4: Alpha Cursor nach oben
****

ESC4 bewegt den Alpha-Cursor eine Reihe nach oben. Befindet sich der Cursor schon in der obersten Reihe, so geschieht nichts.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0
	contrl(6)	Function ID = 4
Output:	contrl(3)	= 0

****
ESC5: Alpha Cursor nach unten
****

ESC5 bewegt den Alpha-Cursor eine Reihe nach unten. Befindet sich der Cursor schon in der untersten Reihe, so geschieht nichts.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0
	contrl(6)	Function ID = 5
Output:	contrl(3)	= 0

****
ESC6: Alpha Cursor nach rechts
****

ESC6 bewegt den Alpha-Cursor eine Spalte nach rechts. Befindet sich der Cursor schon in der äußerst rechten Position, so geschieht nichts.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0
	contrl(6)	Function ID = 6
Output:	contrl(3)	= 0

****
ESC7: Alpha Cursor nach links
****

ESC7 bewegt den Alpha-Cursor eine Spalte nach links. Befindet sich der Cursor schon in der äußerst linken Position, so geschieht nichts.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0
	contrl(6)	Function ID = 7
Output:	contrl(3)	= 0

****
ESC8: Alpha Cursor Home
****

ESC8 bewegt den Alpha-Cursor auf die Home-Position (Reihe = 0, Spalte = 0), also auf den linken oberen Bildschirmrand.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0
	contrl(6)	Function ID = 8
Output:	contrl(3)	= 0

****
ESC9: Löschen bis Bildschirm-Ende
****

ESC9 löscht von der augenblicklichen Cursor-Position bis zum Bildschirmende. Die Cursor-Position bleibt unverändert.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0
	contrl(6)	Function ID = 9
Output:	contrl(3)	= 0

****
ESC10: Löschen bis Zeilen-Ende
****

ESC10 löscht von der augenblicklichen Cursor-Position bis zum Zeilenende. Die Cursor-Position bleibt unverändert.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0
	contrl(6)	Function ID = 10
Output:	contrl(3)	= 0

****
ESC11: Direkte Alpha-Cursor-Positionierung:
****

Über ESC11 wird der Alpha-Cursor direkt über Reihen- und Spaltenangabe adressiert. Adressen außerhalb der gültigen Bereiche werden auf maximale Werte gesetzt.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0
	contrl(6)	Function ID = 11
	intin(1)	Zeile (row) -1
	intin(2)	Spalte (clm) -1
Output:	contrl(3)	= 0

****
ESC12: Ausgabe von Alpha-Text:
****

Diese Funktion gibt einen Alpha-Text-String an der augenblicklichen Cursor-Position aus. Der Text wird mit den derzeitigen Alpha-Attributen (z.B. Invers-Video) ausgegeben. Der Alpha-Text wird immer in Zeichenrichtung 0 Grad mit minimaler Höhe geschrieben.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0

	contrl(4)	Anzahl der auszugebenden Zeichen im Strings
	contrl(6)	Function ID = 12
	intin	Text-String in ASCII, ein Wort pro Zeichen
Output:	contrl(3)	= 0

****
ESC13: Reverse Video ON
****

Aller nachfolgender Text wird in invers Video dargestellt, d.h. schwarze Zeichen auf hellem Hintergrund.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0
	contrl(6)	Function ID = 13
Output:	contrl(3)	= 0

****
ESC14: Reverse Video OFF
****

Aller nachfolgender Text wird normal dargestellt, d.h. helle Zeichen auf schwarzem Hintergrund.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0
	contrl(6)	Function ID = 14
Output:	contrl(3)	= 0

****
ESC15: Abfrage der augenblicklichen Cursor-Position
****

Diese Funktion gibt die Position des Alpha-Cursors wieder.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0

	contrl(6)	Function ID = 15
Output:	contrl(3)	= 0
	intout(1)	Zeilenummer (row) -1
	intout(2)	Spaltennummer (column) -1

****
ESC16: Abfrage des grafischen Tablett-Zustands
****

Diese Funktion wurde implementiert um Kompatibilität mit dem GSX-Standard zu bewahren. Sie hat für die GLIB keine Bedeutung, da diese Funktion über die Befehle 28 - 31 verwirklicht wird.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0
	contrl(6)	Function ID = 16
Output:	contrl(3)	= 0
	intout(1)	Tablett-Zustand = 0

****
ESC17: Hardcopy
****

Diese Funktion dient zum Anfertigen einer Hard-Copy des Bildschirm-Inhalts auf den Drucker. Für die Anpassung an den verwendeten Drucker s. GLIBDATA-File. Die Hardcopy-Funktion kann während ihrem Lauf nicht abgebrochen werden. daher ist es unbedingt nötig vor dem Aufruf der Funktion den Drucker bereit zu machen.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0
	contrl(6)	Function ID = 17
Output:	contrl(3)	= 0

Anmerkung: Alle die in diesem Handbuch gezeigten Diagramme wurden über die Hardcopy-Funktion der GLIB gewonnen.

****
ESC18: Setzen des graphischen Cursors
****

ESC18 setzt den graphischen Cursor auf eine definierte XY-Koordinate. Der graphische Cursor wird grundsätzlich im XOR-Modus geschrieben, um den darunterliegenden Bildschirminhalt nicht zu zerstören. Der graphische Cursor besitzt keine Attribute wie Farbe, Linienart und Größe. Der Cursor wird durch zwei sich kreuzende horizontale und vertikale Linien von jeweils 17 Pixel Länge dargestellt. Der Kreuzungspunkt der Linien stellt die Koordinate des Cursors dar:

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 1
	contrl(6)	Function ID = 18
	ptsin(1)	x-Koordinate des Cursors
	intin(2)	y-Koordinate des Cursors
Output:	contrl(3)	= 0

****
ESC19: Löschen des letzten graphischen Cursors
****

ESC19 löscht den zuletzt definierten graphischen Cursor. Da der Cursor im XOR-Mode gezeichnet wird, bleibt der unter dem Cursor befindliche Bildschirminhalt erhalten. Existiert kein Cursor so findet auch keine Operation statt.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0
	contrl(6)	Function ID = 19
Output:	contrl(3)	= 0

****
ESC20: Hardware-Zoom
****

Diese Funktion stellt den Hardware-Zoom-Faktor ein. Beim Zoomen wird je nach Faktor nur ein Ausschnitt des Gesamtbildes dargestellt, der sich über die PAN-Funktionen verschieben läßt.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0
	contrl(4)	Hardware-Zoom-Faktor
	contrl(6)	Function ID = 20
Output:	contrl(3)	= 0

Beim Hardware-Zoom-Faktor werden nur die zwei LSB-Bits berücksichtigt, d.h. z.B. Zommfaktor 1 = binär 1h = ASCII '1' = ASCCI 'A'.

Faktor:	Zoom:
0	1-fach d.h. natürliche Darstellung
1	2-fach
2	4-fach
3	8-fach

****
ESC21: Bestimmen des Bildschirmausschnitts (PAN direct)
****

Diese Funktion definiert den Bereich des dargestellten Bildschirmausschnitts beim Hardware-Zoomen. Es wird dabei das linke untere Eck des 'Fensters' definiert.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 1
	contrl(6)	Function ID = 21
	ptsin(1)	X-Koordinate des linken unteren Ecks
	ptsin(2)	Y-Koordinate des linken unteren Ecks
Output:	contrl(3)	= 0

****
ESC22: Display ON
****

ESC22 schaltet den Bildschirm zur Anzeige wieder an (s. ESC23).

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0
	contrl(6)	Function ID = 22
Output:	contrl(3)	= 0

****
ESC23: Display OFF
****

Diese Funktion schaltet die Anzeige des Video-Speicher-Inhalts ab. Keine GLIB-Funktion wird von ESC23 beeinflußt. Durch diese Funktion vollzieht sich die Video-Speicher-Manipulation wesentlich schneller. Zudem bleibt ein Bildaufbau unbemerkt für den Anwender. Das Wiedereinschalten des Bildschirms geschieht über Funktion ESC22.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0
	contrl(6)	Function ID = 23
Output:	contrl(3)	= 0

****
ESC24: PAN up
****

Folgende fünf PAN-Funktionen dienen dem Verschieben des Bildschirm-Fensters beim Hard-Ware-Zoomen. PAN-UP bewegt das Bildschirmfenster um einen Pixel (Zeile) nach oben.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0
	contrl(6)	Function ID = 24
Output:	contrl(3)	= 0

****
ESC25: PAN down
****

Bewegt das Bildschirmfenster um einen Pixel (Zeile) nach unten.

```
Input:    contrl(1)      Opcode = 5
          contrl(2)      = 0
          contrl(6)      Function ID = 25

Output:   contrl(3)      = 0
```

****
ESC26: PAN right
****

Bewegt das Bildschirmfenster um 16 Pixel (1 Wort) nach rechts.

```
Input:    contrl(1)      Opcode = 5
          contrl(2)      = 0
          contrl(6)      Function ID = 26

Output:   contrl(3)      = 0
```

****
ESC27: PAN left
****

Bewegt das Bildschirmfenster um 16 Pixel (1 Wort) nach links.

```
Input:    contrl(1)      Opcode = 5
          contrl(2)      = 0
          contrl(6)      Function ID = 27

Output:   contrl(3)      = 0
```


****
ESC28: PAN home
****

Bewegt das Bildschirmfenster auf die Ausgangsposition linker oberer Bildschirmrand.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0
	contrl(6)	Function ID = 28
Output:	contrl(3)	= 0

****
ESC29: Direktes auslesen des Videospeichers (DMA Read)
****

Diese Funktion dient zum direkten auslesen des Videospeichers in den CPU-Arbeitsspeicher. Somit kann z.B. das momentane Video-Bild auf Diskette abgespeichert werden. Pro Aufruf wird eine bestimmbare Bildschirmzeile Y (row) ausgelesen. Die Angabe von Y geschieht in den Device-Koordinaten des uPD 7220. Y=0 entspricht dabei der obersten Bildschirmzeile. Um den Gesammt-Bildinhalt auszulesen muß der Aufruf unter Inkrementierung der Zeilennummer entsprechend oft wiederholt werden.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0
	contrl(4)	DMA-Farb-Ebene
	contrl(5)	Zeilenadresse Y (von Y=0 bis (Ymax)-1)
	contrl(6)	Function ID = 29
Output:	contrl(3)	= 0
	intout	Bitmuster (!) der ausgelesene Bildschirmzeile; Start mit X=0 bis (Xmax)-1

DMA-Farb-Ebene:

Dieser Wert bestimmt die Farb-Ebene die ausgelesen werden soll. Ungültige Werte werden in Farb-Ebene rot umgewandelt.

contrl(4):	Farb-Ebene:
1	rot
2	grün
3	blau
andere	rot

****
ESC30: Direktes beschreiben des Videospeichers (DMA Write)
****

Diese Funktion dient zum direkten Beschreiben des Videospeichers vom CPU-Arbeitsspeicher. Somit kann z.B. ein auf Diskette abgespeichertes Bild zur Anzeige gebracht werden. Pro Aufruf wird eine bestimmbare Bildschirmzeile Y (row) beschrieben. Alle Angaben erfolgen analog zur Funktion DMA Read (ESC29).

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0
	contrl(4)	DMA-Farb-Ebene
	contrl(5)	Zeilenadresse Y (von Y=0 bis (Ymax)-1)
	contrl(6)	Function ID = 30
	intin	Bitmuster (!) der zu schreibenden Bildschirmzeile; Start mit X=0 bis (Xmax)-1
Output:	contrl(3)	= 0

****
ESC31: Virtual adressing ON
****

GLIB rechnet die Ein- und Ausgangskoordinaten in ptsin bzw. ptsout entsprechend um. Bei virtueller Adressierung werden die Koordinaten im Bereich von 0 bis 32767 eingegeben und automatisch in Device-Koordinaten umgerechnet. Die virtuelle Koordinate 32767 entspricht dabei der maximal zulässigen Device-Koordinate. ESC31 schaltetet den virtuellen Adress-Modus ein (Startwert beim Aufruf der GLIB). Der umgekehrte Befehl zu ESC31 ist ESC32 (Device adressing ON).

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0


```
          contrl(6)      Function ID = 31

Output:   contrl(3)      = 0
```

****
ESC32: Device adressing ON
****

Bei diesem Adress-Modus entfällt die Umrechnung von virtuellen zu device-Koordinaten. Alle diesem Befehl folgenden Koordinaten werden in device-Units erwartet, wie sie beim Öffnen des Treibers (Open GLIB) definiert wurden (intout(1) und intout(2)). Ein Überschreiten dieser Werte ist nicht zulässig. Das Gegenstück zu dieser Funktion ist ESC31 (virtual adressing ON).

```
Input:    contrl(1)      Opcode = 5
          contrl(2)      = 0
          contrl(6)      Function ID = 32

Output:   contrl(3)      = 0
```

****
ESC33: Absolut adressing ON
****

In diesem Adress-Modus werden die Ein- und Ausgangskoordinaten im Gegensatz zum relativen Adress-Modus als absolut angesehen, d.h. z.B. eine X-Koordinate von 0 entspricht dem äußerst linken Rand des Bildschirms. Negative Koordinaten sind in diesem Modus unzulässig.

```
Input:    contrl(1)      Opcode = 5
          contrl(2)      = 0
          contrl(6)      Function ID = 33

Output:   contrl(3)      = 0
```


****
ESC34: Relativ adressing ON
****

In diesem Adress-Modus werden die Ein- und Ausgangskoordinaten im Gegensatz zum absoluten Adress-Modus als relativ zu einem definierbaren Stützpunkt angesehen, d.h. z.B. eine X-Koordinate von 0 entspricht nicht unbedingt dem äußerst linken Rand des Bildschirms, sondern der Koordinate des Stützpunkts. Vorteil dieses Adress-Modus ist die allgemeine Definition von Grafiken die an beliebiger Stelle des Bildschirms durch verändern des Stützpunkts plaziert werden können. Im relativen Adress-Modus sind auch negative Koordinaten (virtuell oder device) zulässig.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 0
	contrl(6)	Function ID = 34
Output:	contrl(3)	= 0

****
ESC35: Define relativ adressing point
****

Diese Funktion bestimmt den Stützpunkt im relativen Adress-Modus. Ist beim Aufruf dieser Funbktion schon der relative Adress-Modus eingeschaltet, so unterliegt auch die neuen Koordinaten des relativen Punkts zuvor noch einer relativen Umrechnung an Hand des zuvor gültigen relativen Punktes.

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 1
	contrl(6)	Function ID = 35
	ptsin(1)	X-Koordinate des Stützpunkts
	ptsin(2)	Y-Koordinate
Output:	contrl(3)	= 0

****
ESC36: Exzentrizität der Kreisfunktionen
****

Diese Funktion erlaubt es anstatt der Kreisfunktionen (Arc, Pie und Circle) beliebig geformte Ellypsen zu erstellen. Als Parameter werden zwei Angaben benötigt. Parameter 1 gibt an, um das wievielfache der ursprüngliche Kreis (oder Kreisstück) in X-Richtung gestaucht wird. Gültige Werte sind 0 - 255. Ein Wert von 0 (Startwert) schaltet die Ellypsen-Funktion aus und zeichnet einen Kreis (keine Stauchung). Der zweite Parameter hat die gleiche Bedeutung für die Y-Achse.

Beispiel:

Kreisfunktion: P1 = P2 = 0 (default)

X = cos(phi)
Y = sin(phi)

Ellypse 1: P1 = 2; P2 = 0

X = (cos(phi)) / 2
Y = sin(phi)

Ellypse 2: P1 = 5; P2 = 3

X = (cos(phi)) / 5
Y = (sin(phi)) / 3

Input:	contrl(1)	Opcode = 5
	contrl(2)	= 1
	contrl(6)	Function ID = 36
	intin(1)	P1 staucht X-Werte (0-255, 0 = aus)
	intin(2)	P2 staucht Y-Werte (0-255, 0 = aus)
Output:	contrl(3)	= 0

****
POLYLINE: Zeichnen einer Polylinie
****

Diese Funktion erstellt eine Poly-Linie. Der Startpunkt der Linie ist der erste Punkt im ptsin-Feld. Die Linien werden zwischen aufeinanderfolgenden Punkten im ptsin-Feld gezogen. Das Verbinden des letzten Punktes mit dem Start-Punkt erfolgt nicht, ist dies dennoch gewünscht so ist zusätzlich die Start-Koordinate als letzter Punkt zu übergeben. Ein einzelnes Koordinatenpaar wird nicht gezeichnet. Eine Linie mit der Länge 0 (Xn = Xn+1, Yn = Yn+1) wird als Punkt dargestellt. Alle Linienattribute wie Linien-Farbe, Linien-Art und Linien-Breite werden berücksichtigt. Maximal sind 256 Koordinatenpaare erlaubt, sollte die Linie aus mehr Stütz-Punkten aufgebaut sein, so muß der Aufruf mit neuen Werten wiederholt werden.

Input:	contrl(1)	Op-Code = 6
	contrl(2)	Anzahl der Koordinatenpaare n mit n <= 256
	ptsin(1)	X-Koord. des Startpunkts
	ptsin(2)	Y-Koord. des Startpunkts
	ptsin(3)	X-Koord. des zweiten Punkts
	ptsin(4)	Y-Koord. des zweiten Punkts
	.	
	.	
	ptsin(2n-1)	X-Koord. des letzten Punkts
	ptsin(2n)	Y-Koord. des letzten Punkts
Output:	contrl(3)	= 0

****
POLYMARKER: Setzen von Markern
****

Diese Funktion zeichnet Marker auf die Punkte die im ptsin-Feld definiert wurden. Alle Markerattribute wie Marker-Farbe, Marker-Art und Marker-Größe werden berücksichtigt. Maximal sind 256 Koordinatenpaare pro Aufruf erlaubt.

Input:	contrl(1)	Op-Code = 7
	contrl(2)	Anzahl der Koordinatenpaare n

mit n <= 256

	ptsin(1)	X-Koord. des ersten Markers
	ptsin(2)	Y-Koord. des ersten Markers
	ptsin(3)	X-Koord. des zweiten Markers
	ptsin(4)	Y-Koord. des zweiten Markers
	.	
	.	
	ptsin(2n-1)	X-Koord. des letzten Markers
	ptsin(2n)	Y-Koord. des letzten Markers
Output:	contrl(3)	= 0

****

TEXT: Schreiben von Text

****

TEXT schreibt ASCII-Text startend bei der spezifizierten X,Y-Position auf den Bildschirm. Die X,Y-Koordinate ist das linke untere Ecke des ersten Zeichens. Alle undefinierten Zeichen werden als Leerzeichen dargestellt. Alle Textattribute wie Text-Farbe, Text-Art, Text-Größe und Text-Richtung werden berücksichtigt. Jedes Wort (= 2 Bytes) im intin-Feld enthält nur ein ASCII-Zeichen. Zu beachten ist, daß die Länge des Textes nicht den zulässigen Bildschirmrand überschreitet (abhängig von Text-Größe und -Richtung).

Input:	contrl(1)	Op-Code = 8
	contrl(2)	Anzahl der Koordinatenpaare = 1
	contrl(4)	Anzahl der Zeichen (= Worte) im String
	ptsin(1)	X-Koord. des Startpunkts des Strings
	ptsin(2)	Y-Koord. des Startpunkts des Strings
	intin	String in ASCII, ein Wort pro Zeichen
Output:	contrl(3)	= 0

****
AREA FILL: Ausfüllen eines Polygons
****

Diese sehr komplexe Funktion füllt ein beliebig (!) geformtes Polygon mit einem bestimmbaren Muster aus. Die Eckpunkte des Polygons werden durch das ptsin-Array festgelegt. Alle Füllattribute wie Füll-Farbe, Füll-Index und Füll-Art werden berücksichtigt. Maximal sind 256 Koordinatenpaare pro Aufruf erlaubt. Ein Polygon mit einem Null-Inhalt (3 aufeinanderfolgende Punkt gleich) wird als Punkt dargestellt. Ein Polygon mit nur einem Endpunkt wird nicht dargestellt.

Input:	contrl(1)	Op-Code = 9
	contrl(2)	Anzahl der Koordinatenpaare n des Polygons mit n <= 256
	ptsin(1)	X-Koord. des ersten Eckpunkts
	ptsin(2)	Y-Koord. des ersten Eckpunkts
	ptsin(3)	X-Koord. des zweiten Eckpunkts
	ptsin(4)	Y-Koord. des zweiten Eckpunkts
	.	
	.	
	ptsin(2n-1)	X-Koord. des letzten Eckpunkts
	ptsin(2n)	Y-Koord. des letzten Eckpunkts
Output:	contrl(3)	= 0

****
CELL ARRAY: Füllen eines Zellengebiets
****

Diese Funktion veranlaßt GLIB ein rechteckiges Feld zu zeichnen, das aus bestimmbaren Zellenelementen aufgebaut ist. Die Dimension des Rechtecks wird durch seinen unteren linken und oberen rechten Eckpunkt im ptsin-Feld festgelegt. Sein Inhalt, also dir Farbe der einzelnen Zellen (Color-Index) bestimmt das intin-Feld. Jede Zeile im intin-Feld wird durch eine entsprechende Zellbreite automatisch auf die Gesammtbreite des Rechtecks aufgeweitet. Entsprechendes gilt für die Spalten. Die Breite der einzelnen Zellelemente wird gleich groß gewählt. Entsprechendes gilt für ihre Höhe. Höhe und Breite einer Zelle können jedoch unterschiedliche Werte annehmen.

Input:	contrl(1)	Op-Code = 10
	contrl(2)	Anzahl der Koordinatenpaare = 2
	contrl(7)	Anzahl der Zell-Spalten (columns) >= 1
	contrl(8)	Anzahl der Zell-Reihen (rows) >= 1
	contrl(9)	Pixel Operation
	intin	Farb-Index Feld für die Zellelemente. Reihenweise angeordnet beginnend in der obersten Reihe. Ein Wort pro Zellelement.
	ptsin(1)	X-Koord. des linken unteren Eckpunkts
	ptsin(2)	Y-Koord. des linken unteren Eckpunkts
	ptsin(3)	X-Koord. des rechten oberen Eckpunkts
	ptsin(4)	Y-Koord. des rechten oberen Eckpunkts
Output:	contrl(3)	= 0

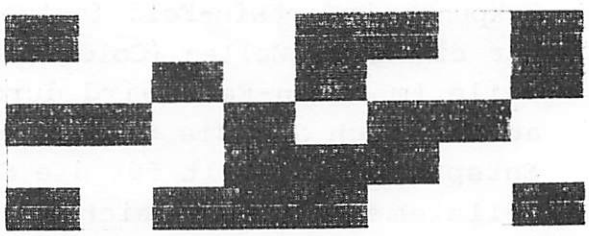
Pixel Operation:
Normalerweise wird bei allen Funktionen der durch den Befehl SET WRITING MODE (s. Funktion 32) definierte Schreib-Modus verwendet. Die CELL ARRAY Funktion jedoch erlaubt die Definition eines eigenen nur für diesen Befehl gültigen Schreibmodus:

contrl(9):	Pixel-Operation:
1	replace
2	transparent
3	complement xor
4	reset

Beispiel:
Folgendes Beispiel zeigt ein Zellenelement mit 5 Reihen zu 8 Spalten. D.h. das Zellgebiet besteht aus folgenden 40 Elementen, die wortweise im intin-Feld abgelegt wurden:

rechtes oberes Eck ptsin(3,4)

1. Reihe:	1,0,0,0,1,1,0,1
2. Reihe:	0,0,1,0,1,0,0,1
3. Reihe:	1,1,0,1,0,1,1,0
4. Reihe:	0,0,0,1,1,1,0,0
5. Reihe:	1,0,1,1,1,0,0,1



linkes unteres Eck ptsin(1,2)

****
Allgemeine Zeichen Primitive: GDP's
****

Die allgemeinen Zeichen Primitive (GDP's) erzeugen selbstständig oft wiederkehrende Figuren wie Rechtecke, Kreissegmente, 'Kuchenstücke' und Kreise. Kreise werden dabei sehr schnell per Software errechnet (reiner Assemblercode), da die Kreisfunktion des GDP's 7220 nur bei quadratisch skallierten Bildschirmen anzuwenden sind.

****
Balken-GDP:
****

Diese GDP-Funktion dient zum Anfertigen von Rechtecken und Balkendiagrammen. Die Dimension des Rechtecks ergibt sich aus seinem unterem linken und oberen rechten Eckpunkt. Alle Area-Fill-Attribute werden auch beim Bar-GDP verwendet.

Input:	contrl(1)	Opcode = 11 (GDP)
	contrl(2)	Anzahl der Koordinatenpaare = 2
	contrl(6)	Bar-GDP identfier = 1
	ptsin(1)	X-Koord. des linken unteren Eckpunkts
	ptsin(2)	Y-Koord. des linken unteren Eckpunkts
	ptsin(3)	X-Koord. des rechten oberen Eckpunkts
	ptsin(4)	Y-Koord. des rechten oberen Eckpunkts
Output:	contrl(3)	= 0

****
Kreisbogen-GDP:
****

Diese Funktion zeichnet Kreisbögen (bzw. Ellypsen s. ESC 36). Der Kreisbogen wird mit den momentanen Linienattributen gezeichnet. Anzugeben sind Mittelpunkt, Radius, Start- und Endwinkel des Bogens. Beliebige Werte sind für Start- und Endwinkel zulässig. Der Startwinkel darf auch größer als der Endwinkel sein. Die Winkelangaben erfolgen in 10tel Grad-Stufen. Ein Radius von 0 ist nicht zulässig. Die Radiusangabe wird keiner relativen Koordina-

tenumrechnung unterworfen, da diese Größe ein Länge und keine Koordinate darstellt.

Input:	contrl(1)	Opcode = 11 (GDP)
	contrl(2)	Anzahl der Koordinatenpaare = 4
	contrl(6)	Arc-GDP identfier = 2
	intin(1)	Startwinkel in 10tel Grad
	intin(2)	Endwinkel in 10tel Grad
	ptsin(1)	X-Koord. des Mittelpunkts
	ptsin(2)	Y-Koord. des Mittelpunkts
	ptsin(3)	undefiniert, keine Angabe
	ptsin(4)	undefiniert, keine Angabe
	ptsin(5)	undefiniert, keine Angabe
	ptsin(6)	undefiniert, keine Angabe
	ptsin(7)	Radius (keine rel. K.-Umrechnung.)
Output:	contrl(3)	= 0

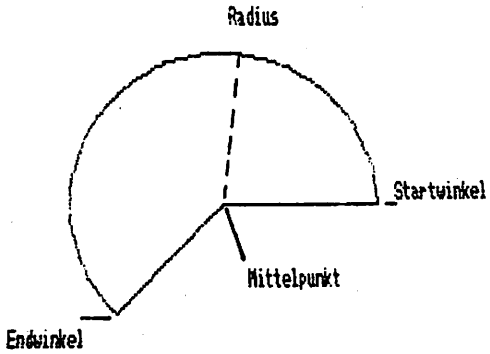
****
Pie-GDP:
****

Diese Funktion zeichnet Kuchenstücke (sog. Pie's). Der Pie-Slice wird mit den momentanen Area-Fill-Attributen gezeichnet. Es gelten die gleichen Eingabe-Werte wie für den Kreis-Bogen GDP (auch Ellypsen möglich).

Input:	contrl(1)	Opcode = 11 (GDP)
	contrl(2)	Anzahl der Koordinatenpaare = 4
	contrl(6)	Pie-GDP identfier = 3
	intin(1)	Startwinkel in 10tel Grad
	intin(2)	Endwinkel in 10tel Grad
	ptsin(1)	X-Koord. des Mittelpunkts
	ptsin(2)	Y-Koord. des Mittelpunkts
	ptsin(3)	undefiniert, keine Angabe
	ptsin(4)	undefiniert, keine Angabe
	ptsin(5)	undefiniert, keine Angabe
	ptsin(6)	undefiniert, keine Angabe
	ptsin(7)	Radius (keine rel. K.-Umrechnung.)

	ptsin(8)	undefiniert, keine Angabe
Output:	contrl(3)	= 0

Beispiel für ein Kuchenstück mit Füllart 'hollow':



****
Kreis-GDP:
****

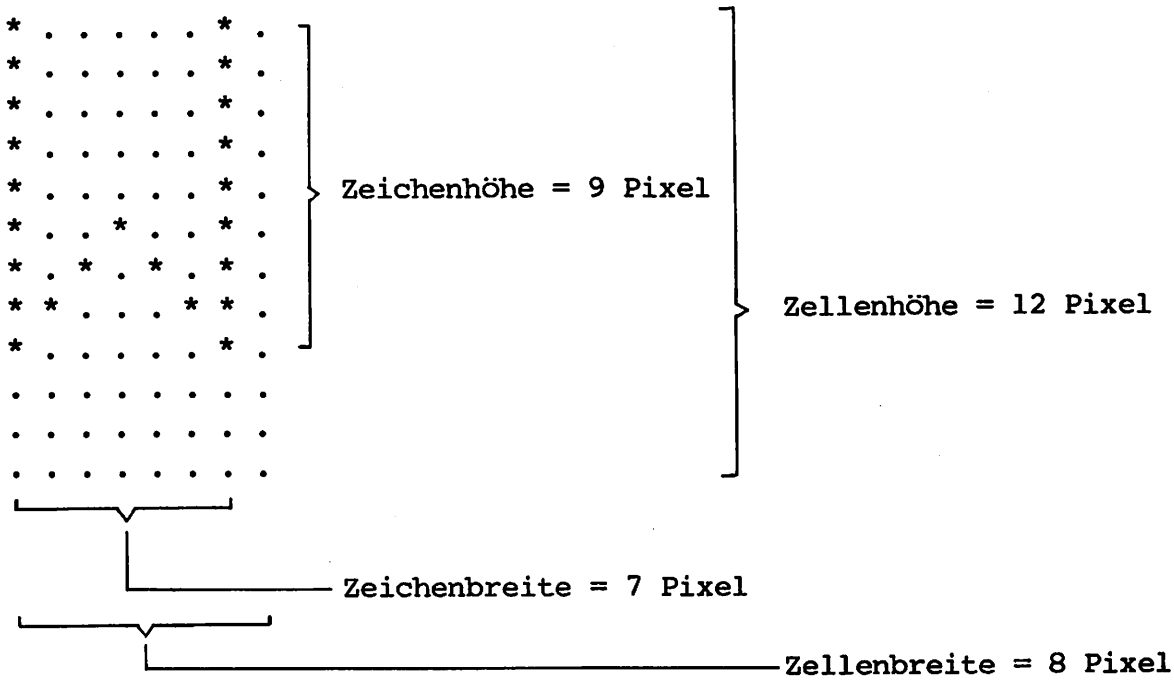
Diese Funktion zeichnet Vollkreise oder Ellypsen (s. ESC 36). Der Kreis wird dabei mit den momentanen Area-Fill-Attributen gezeichnet. Als Angaben werden Mittelpunkt und Radius des Kreises benötigt. Die Radius-Angabe unterliegt keiner relativen Koordinatenumrechnung.

Input:	contrl(1)	Opcode = 11 (GDP)
	contrl(2)	Anzahl der Koordinatenpaare = 3
	contrl(6)	Circle-GDP identfier = 4
	ptsin(1)	X-Koord. des Mittelpunkts
	ptsin(2)	Y-Koord. des Mittelpunkts
	ptsin(3)	undefiniert, keine Angabe
	ptsin(4)	undefiniert, keine Angabe
	ptsin(5)	Radius (keine rel. K.-Umrechnung.)
	ptsin(6)	undefiniert, keine Angaben
Output:	contrl(3)	= 0

****
SET CHARACTER HEIGHT: Setzen der Zeichen-Höhe
****

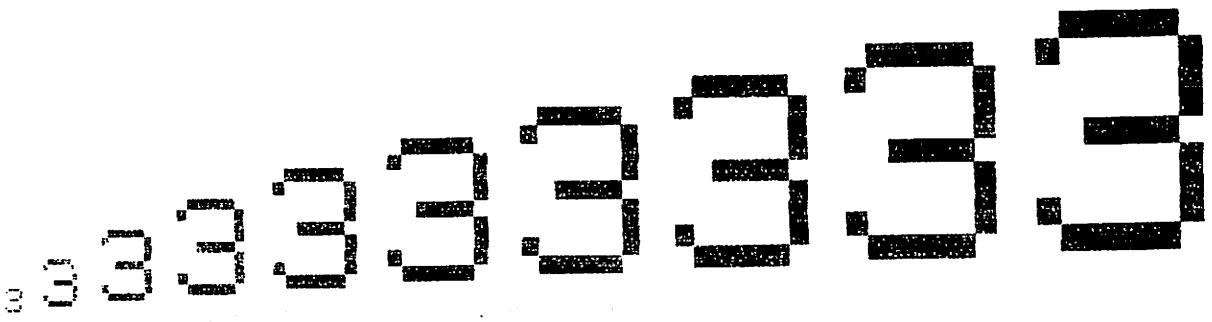
Dieser Befehl definiert die Größe der Text-Zeichen. Die Angabe erfolgt stufenlos durch die Höhe des Zeichens in Device-Koordinaten. Sie unterliegt keiner relativen Koordinaten-Umrechnung. Unter der GLIB sind 16 verschiedenen Zeichengrößen möglich (1-16-fach der normalen Größe von 12 Pixel). Eine Zeichengröße kleiner als die minimale Höhe wird auf die minimale Höhe aufgerundet. Sonst werden nicht-darstellbare Zwischenstufen auf die nächst niedrigere Zeichengröße abgerundet. GLIB gibt die tatsächlich angewählte Zeichengröße an das aufrufende Programm zurück. Dabei wird zwischen der Größe der Zelle des Zeichens (Cell height, width) und der Größe des Zeichens selbst (Character height, width) unterschieden.

Folgendes Diagramm soll am Beispiel des Buchstabens 'W' bei minimaler Zeichengröße den Zusammenhang zwischen Zeichen- und Zellengröße verdeutlichen.



Input:	contrl(1)	Opcode = 12 (Set character height)
	contrl(2)	Anzahl der Koordinatenpaare = 1
	ptsin(1)	undefiniert, keine Angabe

	ptsin(2)	Angeforderte Zeichengröße (unterliegt keiner rel. Koord.-Umrechnung)
Output:	contrl(3)	Anzahl der Ausgangskoord.-Paare = 2
	ptsout(1)	Momentane neue Zeichenbreite
	ptsout(2)	Momentane neue Zeichenhöhe
	ptsout(3)	Momentane neue Zellenbreite
	ptsout(4)	Momentane neue Zellenhöhe



Beispiel für die Zeichenhöhen 9 Pixel (x1) bis 90 Pixel (x10):

****

SET CHARACTER UP VECTOR: Definieren der Text-Richtung

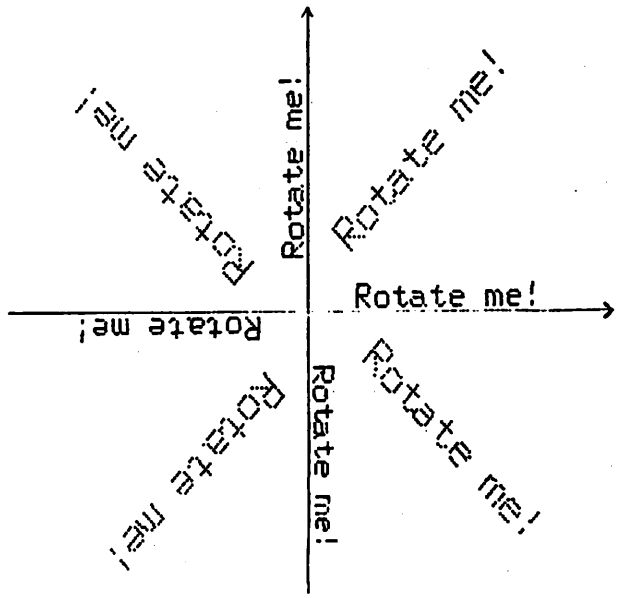
****

Diese Operation definiert die Schreibrichtung der Text-Zeichen (Winkel der Grundlinie) . Die Angabe erfolgt in zehntel Winkelgrad (0 - 3600). GLIB kann Text in 8 verschiedenen Zeichenrichtungen schreiben (0, 45, 90, 135, 180, 225, 270, 315). Daher werden Zwischenwerte in der Winkelangabe auf die nächstbeste Winkelstufe auf- oder abgerundet. Dieser Winkel wird von GLIB zurückgegeben. Winkelangaben größer gleich 360 Grad sind unzulässig und setzen die Zeichenrichtung auf 0 Grad.

Input:	contrl(1)	Opcode = 13 (Set character up vector)
	contrl(2)	= 0
	intin(1)	Gewünschte Richtung in 10tel Grad (0-3600)
Output:	contrl(3)	= 0

contrl(5) = 1

intout(1) Tatsächliche neue Zeichenrichtung in 10tel Grad



Beispiel für die 8 Textrichtungen von 0 bis 315 Grad in 45 Grad Schritten.

****

SET COLOR REPRESENTATION: Definieren eines Farbindex

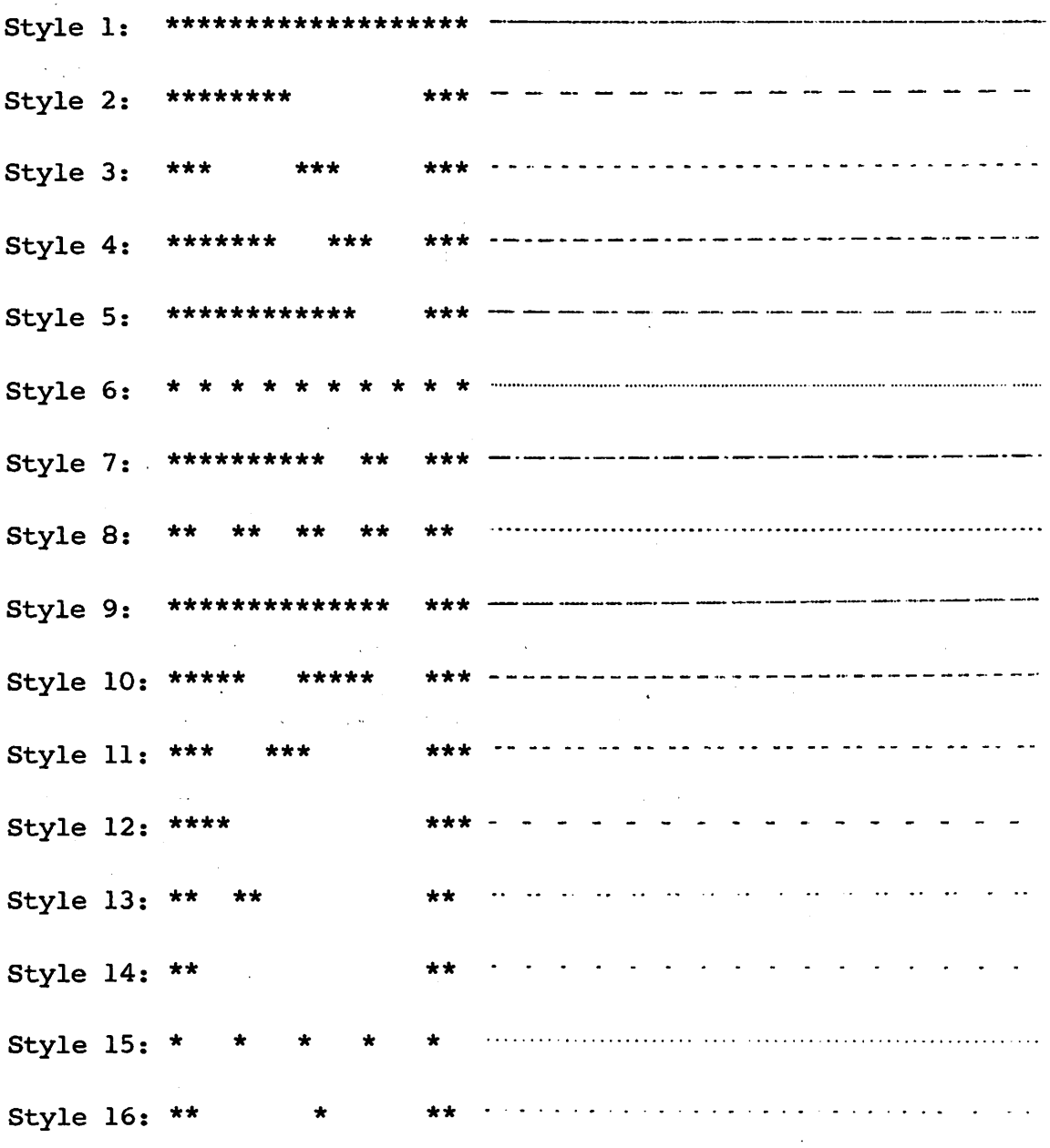
****

Diese Operation definiert zu einem bestimmten Farbindex seinen zugehörigen Farbwert. Die Angabe erfolgt dabei in zehntel Prozent des Rot-, Blau- und Grünanteils. GLIB setzt jeden Farbanteil ungleich 0 auf 100% . Existiert der angesprochene Farb-Index nicht, so findet keine Operation statt. Bei monochromen Geräten gibt es nur zwei Indizes 0 und 1. In der Farbausführung sind acht Indizes 0 - 7 erlaubt.

Input:	contrl(1)	Opcode = 14 (Set color representation)
	contrl(2)	= 0
	intin(1)	Angewählter Farbindex
	intin(2)	Zugehöriger Rot-Anteil (In zehntel Prozent 0-1000)
	intin(3)	Zugehöriger Grünanteil
	intin(4)	Zugehöriger Blauanteil
Output:	contrl(3)	= 0

****
SET POLYLINE LINETYPE: Definieren der Linienart
****

Diese Operation setzt die Linienart für alle nachfolgenden Linienoperationen (z.B. POLYLINE, ARC ...) fest. GLIB kennt 16 verschiedene Linienarten. Falls die gewünschte Linienart nicht existiert wird Linienart 1 (durchgezogen) eingestellt.



Input:	contrl(1)	Opcode = 15 (Set polyline linetype)
	contrl(2)	= 0
	intin(1)	Gewünschte Linienart (Line-Style)

Output:	contrl(3)	= 0
	intout(1)	Tatsächlich angwählte Linienart

****
SET POLYLINE LINE WIDTH: Definieren der Linienbreite
****

Diese Funktion ist in der GLIB nicht implementiert, jedoch aus Gründen der Kompatibilität zu GSX-80 als Dummy-Funktion aufrufbar. Sind verschiedene Liniearten gewünscht, so sind diese sehr einfach durch mehrfaches Zeichnen der selben Linie mit 1 Pixel Versatz möglich.

Input:	contrl(1)	Opcode = 16 (Set line width)
	contrl(2)	Anzahl der Koord.-Paare = 1
	ptsin(1)	Gewünschte Linienbreite
	ptsin(2)	undefiniert, keine Angabe nötig
Output:	contrl(3)	Anzahl der Ausg.-Koordinaten
	ptsout(1)	Tatsächlich angewählte Liniebreite = 1
	ptsout(2)	undefiniert = 0

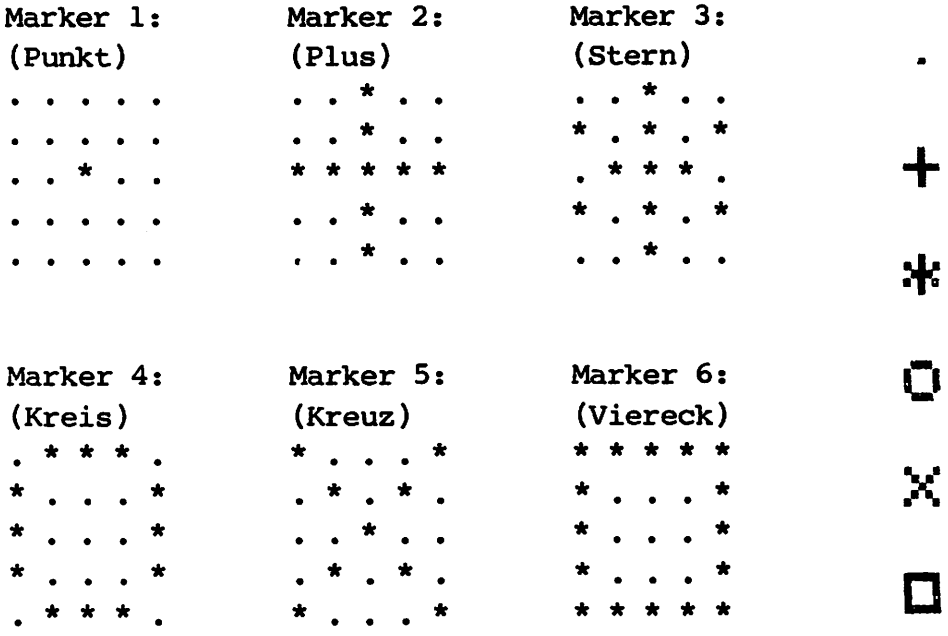
****
SET POLYLINE COLOR INDEX: Anwahl des Linien-Farbindex
****

Diese Funktion definiert die Farbe aller nachfolgenden Linienoperationen mit Hilfe eines Farbindex an Hand der Farb-Index-Tabelle. Falls der gewünschte Farbindex nicht existiert wird der maximale Farbindex angewählt.

Input:	contrl(1)	Opcode = 17 (Set polyline color)
	contrl(2)	= 0
	intin(1)	Gewünschter Farbindex für Linien
Output:	contrl(3)	= 0
	intout(1)	Tatsächlich angewählter Farbindex

****
SET POLYMARKER TYPE: Definieren des Marker-Symbols
****

Diese Funktion definiert das zu verwendente Marker-Symbol für alle nachfolgenden Marker-Befehle. Unter GLIB sind sechs verschiedene Marker-Symbole möglich. Falls das gewünschte Markersymbol nicht existiert wird Marker-Symbol Nr. 3 (Stern) verwendet.



Input:	contrl(1)	Opcode = 18 (Polymarker type)
	contrl(2)	= 0
	intin(1)	Gewünschtes Markersymbol (1-6)
Output:	contrl(3)	= 0
	intout(1)	Tatsächlich selektierte Symbol-Nr.

****
SET POLYMARKER SCALE: Definieren der Marker-Größe
****

Diese Operation setzt die Größe der Markersymbole fest. Ähnlich wie unter der Funktion 'Text height' kann die Größe stufenlos angegeben werden, wobei GLIB auf die nächst niedrigere von 16 verschiedenen Größen abrundet. Die minimale Markergröße beträgt 5 Pixel Höhe und Breite. Wird die minimal zulässige Größe unterschritten so wird auf diese Höhe aufgerundet.

Input:	contrl(1)	Opcode = 19 (Set polymarker scale)
	contrl(2)	Anzahl der Koordinatenpaare = 1
	ptsin(1)	undefiniert, keine Angabe nötig
	ptsin(2)	Angeforderte Markergröße (unterliegt keiner rel. Koord.-Umrechnung)
Output:	contrl(3)	Anzahl der Ausgangskoord.-Paare = 1
	ptsout(1)	undefiniert = 0
	ptsout(2)	Tatsächlich angewählte Markergröße

****
SET POLYMARKER COLOR INDEX: Anwahl des Marker-Farbindex
****

Diese Funktion definiert die Farbe aller nachfolgenden Markeroperationen mit Hilfe eines Farbindex an Hand der Farb-Index-Tabelle. Falls der gewünschte Farbindex nicht existiert wird der maximale Farbindex angewählt.

Input:	contrl(1)	Opcode = 20 (Set marker color)
	contrl(2)	= 0
	intin(1)	Gewünschter Farbindex für Marker-Symbole
Output:	contrl(3)	= 0
	intout(1)	Tatsächlich angewählter Farbindex

****
SET TEXT FONT: Anwahl der Textart
****

GLIB kennt acht verschiedene Textarten (fonts). Existiert die gewünschte Text-Art nicht so wird Font Nr. 8 angewählt.

International Text Font 1: {[]}!\~
Deutsch Text Font 2: äÄüÜöÖß
Unterstrichen/int. Text Font 3: {[]}!\~
Unterstrichen/deutsch Text Font 4: äÄüÜöÖß
Cursiv/int. *Text Font 5: {[]}!\~*
Cursiv/deutsch *Text Font 6: äÄüÜöÖß*
Cursiv/unterst./int. *Text Font 7: {[]}!\~*
Cursiv/unterst./dtsch. *Text Font 8: äÄüÜöÖß*

Input:	contrl(1)	Opcode = 21 (Set hardware text font)
	contrl(2)	= 0
	intin(1)	Gewünschte Textart
Output:	contrl(3)	= 0
	intout(1)	Tatsächlich angewählte Textart

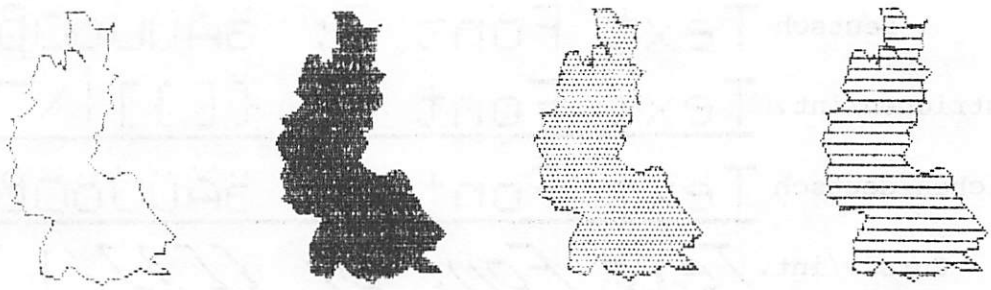
****
SET TEXT COLOR INDEX: Anwahl des Text-Farbindex
****

Diese Funktion definiert die Farbe aller nachfolgenden Textoperationen mit Hilfe eines Farbindex an Hand der Farb-Index-Tabelle. Falls der gewünschte Farbindex nicht existiert wird der maximale Farbindex angewählt.

Input:	contrl(1)	Opcode = 22 (Set text color)
	contrl(2)	= 0
	intin(1)	Gewünschter Farbindex für Text
Output:	contrl(3)	= 0
	intout(1)	Tatsächlich angewählter Farbindex

****
SET FILL INTERIOR STYLE: Füll-Art definieren
****

Diese Funktion selektiert die Füll-Art für alle nachfolgenden Füllfunktionen (Area-Fill, Pie, Circle ...). Folgende Füllarten sind möglich:



0: Umranden (hollow) 1: Ausfüllen (solid) 2: Grauwerte (halfton) 3: Strichmuster (hatch)

Falls die gewünschte Füllart nicht existiert wird Füllart Nr. 0 umranden gewählt.

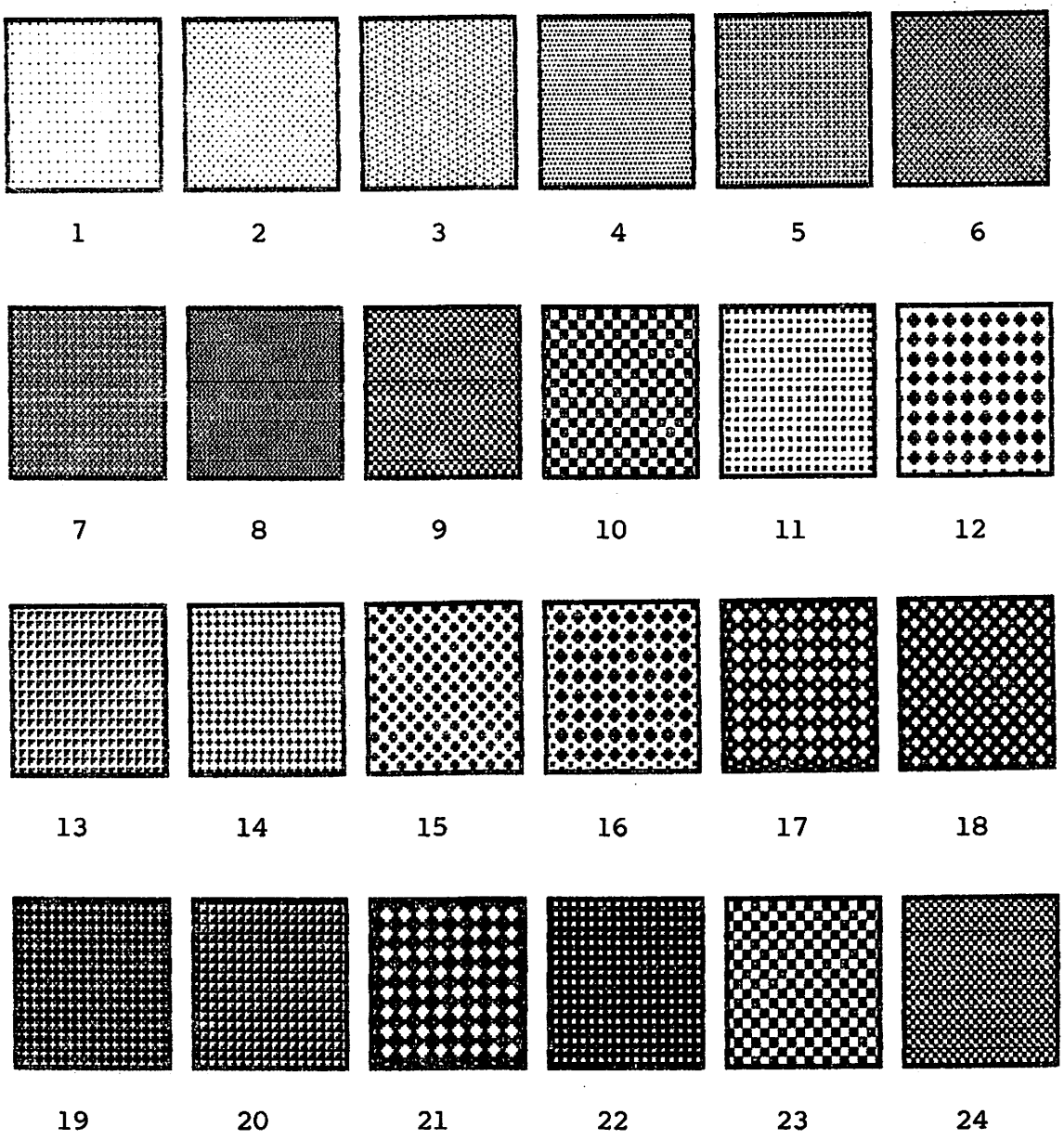
Input:	contrl(1)	Opcode = 23 (Set fill interior style)
	contrl(2)	= 0
	intin(1)	Gewünschte Füll-Art (0-3)
Output:	contrl(3)	= 0
	intout(1)	Tatsächlich angewählte Füllart

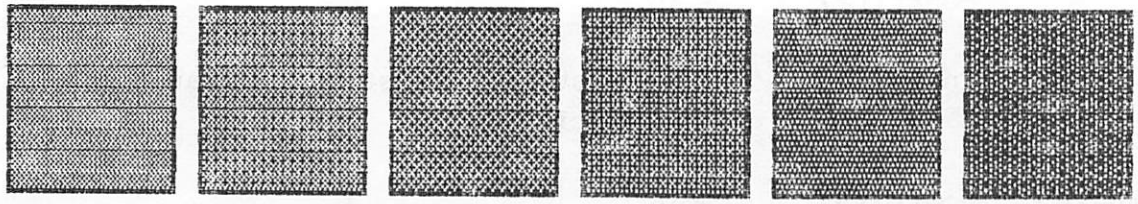
****
SET FILL INDEX: Definieren des Füll Musters
****

Diese Funktion definiert einen Füllindex je nach angewählter Füllart. Der Füllindex hat keine Bedeutung bei der Füllart 0 (Umranden) und 1 (Ausfüllen). Es sind 32 Füllmuster anwählbar (1-32). Falls das angewählte Muster nicht existiert wird Füllindex 1 angewählt. Das Füllmuster bezieht sich auf die Grauwerte falls Füllart 2 angewählt wurde, bei Füllart 3 auf die Strichmuster.

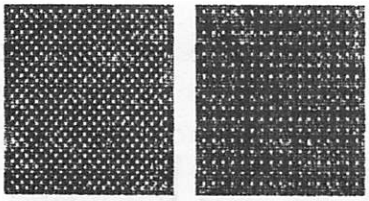
Input:	contrl(1)	Opcode = 24 (Set fill style)
	contrl(2)	= 0
	intin(1)	Gewünschtes Füllmuster für Graustufen oder Stricheln
Output:	contrl(3)	= 0
	intout(1)	Tatsächlich angewählter Füll-Index

Grauwerte (halftone patterns):



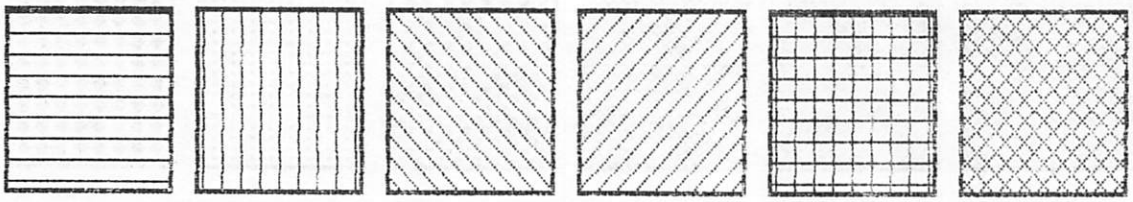


25 26 27 28 29 30

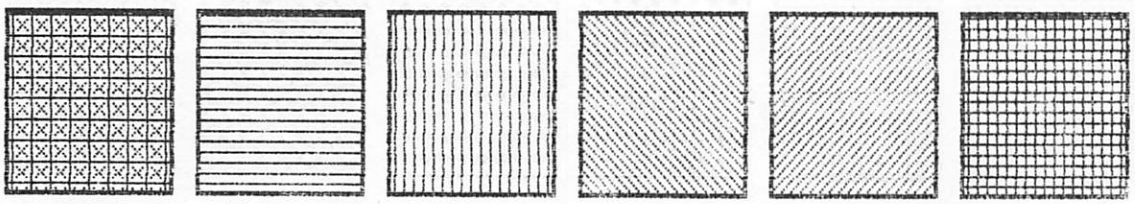


31 32

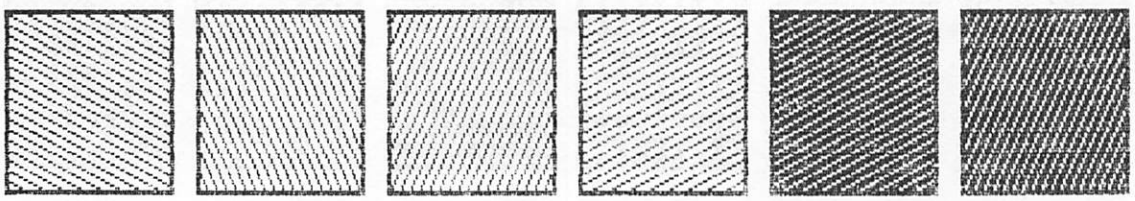
Strichmuster (hatch styles):



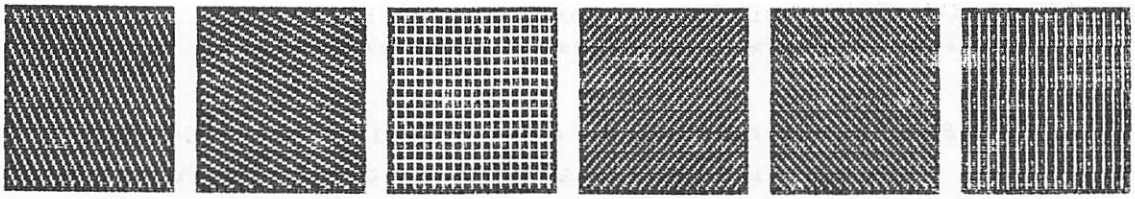
1 2 3 4 5 6



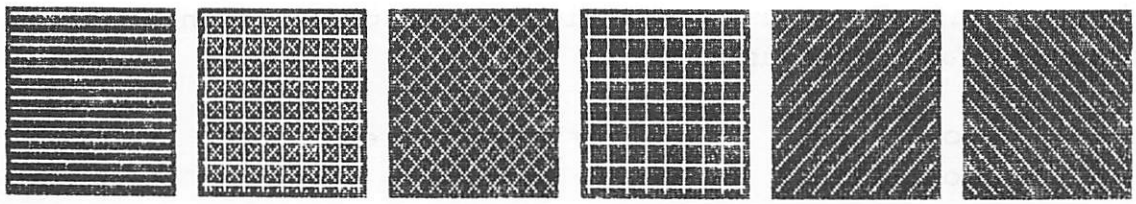
7 8 9 10 11 12



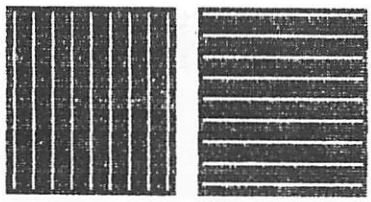
13 14 15 16 17 18



19 20 21 22 23 24



25 26 27 28 29 30



31 32

****

SET FILL COLOR INDEX: Anwahl des Füll-Farbindex

****

Diese Funktion definiert die Farbe aller nachfolgenden Fülloperationen (Areafill, Pie, Circle ...) mit Hilfe eines Farbindex an Hand der Farb-Index-Tabelle. Falls der gewünschte Farbindex nicht existiert wird der maximale Farbindex angewählt.

Input:	contrl(1)	Opcode = 25 (Set fill color)
	contrl(2)	= 0
	intin(1)	Gewünschter Farbindex für Füll-Operat.
Output:	contrl(3)	= 0
	intout(1)	Tatsächlich angewählter Farbindex

****
INQUIRE COLOR REPRESENTATION: Ermitteln des Farbwerts
****

Bei dieser Funktion gibt GLIB zu einem bestimmten Farb-Index die zugehörigen Farbintensitäten an, so wie sie durch die Funktion 14 (Set color representation) definiert wurden. Wird der maximal zulässige Farb-Index überschritten, so erfolgen die Angaben für den maximalen Index. Je nach Wert von intin(2) gibt diese Funktion Auskunft über dir tatsächlichen oder über die ehemals gewünschten Farbwerte Auskunft.

Input:	contrl(1)	Opcode = 26 (Inquire color)
	contrl(2)	= 0
	intin(1)	Angesprochener Farbindex
	intin(2)	Gewünschte Information 0: Ehemals gewünschter Farbwert 1: Tatsächlich selektierter Farbwert
Output:	contrl(3)	= 0
	intout(1)	Tatsächlich angewählter Farbindex
	intout(2)	Rot-Intensität in 10tel Prozent (0 - 1000)
	intout(3)	Grün Intensität
	intout(4)	Blau Intensität

****
INQUIRE CELL ARRAY: Lesen eines Zellen-Rechtecks
****

Diese Funktion veranlaßt GLIB die Farbwerte eines rechteckigen Feldes auszulesen. Dies stellt die Umkehr-Operation zu Opcode 10 (Cell array) dar. Die physikalische Größe des auszulesenden Rechtecks wird durch seinen unteren linken und oberen rechten Eckpunkt im ptsin-Feld festgelegt. Wieviele Elemente tasächlich übergeben werden (Auflösung) hängt von der Größe des intout-Feldes ab, d.h. wieviel Farbinformationen es fassen kann (= contrl(4) / 2). Wieviele Spalten- bzw. Reihenelemnte übergeben werden ergibt sich aus der Größe des intout-Feldes und aus der Anzahl der Reihen in contrl(7).

Input:	contrl(1)	Op-Code = 27 (Inquire cell)
	contrl(2)	Anzahl der Koordinatenpaare = 2
	contrl(4)	Länge des intout-Feldes in Byte -> /2 = Anzahl der Werte die gelesen werden
	contrl(7)	Anzahl der Spalten (Zeilen) die ausgelesen werden sollen >= 1
	ptsin(1)	X-Koord. des linken unteren Eckpunkts
	ptsin(2)	Y-Koord. des linken unteren Eckpunkts
	ptsin(3)	X-Koord. des rechten oberen Eckpunkts
	ptsin(4)	Y-Koord. des rechten oberen Eckpunkts
Output:	contrl(3)	= 0
	contrl(8)	Anzahl der Elemente pro Reihe (=Spalten)
	contrl(9)	Anzahl der Reihen
	contrl(10)	Fehler-Flag: 0: Falls keine Fehler auftraten 1: Falls der Farb-Index eines Elements an Hand der derzeitigen Farb-Index-Tabelle nicht ermittelt werden konnte
	intout	Farb-Index Feld für die Zellelemente. Reihenweise angeordnet beginnend in der obersten Reihe links. Ein Wort pro Zellelement. Ein Wert von 00ffh gibt an, daß der Farbindex für dieses Element nicht ermittelt werden konnte.

****

INPUT LOCATOR: Ermitteln der Locator-Position

****

Diese Funktion dient zur Angabe einer X-Y-Position auf dem Bildschirm per grafischem Cursor und speziellen Cursor-Positionier-Funktionstasten. Ein graphischer Cursor wird auf dem Bildschirm per Cursor-Tasten vom Anwender bewegt, bis er sich auf der gewünschten Position befindet. Diese Position wird dem aufrufendem Programm übermittelt. Im GLIBDATA-File kann die Eingabe der Zeichen ('LOCST', 'LOCIN') und deren Bedeudtung ('LOCCHA') verändert werden. Standardmäßig werden Zeichen vom seriellen Tastatur-

kanal des System 8000 hereingeholt. Die Eingabe kann natürlich auch über die CP/M-Einheiten Console oder Auxiliary oder über eine Maus, Joy-Stick oder Tablett nach entsprechender Hardwareerweiterung erfolgen. Standardmäßig sind als Loacator-Tasten die bekannten WordStar Steuerzeichen implementiert.

	Langsam (Eine Position)	Schnell (Zehn Positionen)
Locator nach oben:	^E	^R
Locator nach unten:	^X	^C
Locator nach links:	^S	^A
Locator nach rechts:	^D	^F

Request Modus:

Beim Einsprung in die Routine wird der grafische Cursor auf die angegebenen Werte gesetzt. Der Cursor wird entsprechend den oben aufgeführten Steurzeichen über den Bildschirm bewegt. Seine Position wird dem aufrufendem Programm übergeben, sobald ein ungültiges Cursor-Steuerzeichen eingegeben wird (Terminator z.B. Schalter einer Maus). In diesem Fall wird die Anzeige des grafischen Cursors wieder aufgehoben, das Terminierungszeichen wird ebenfalls an das aufrufende Programm übergeben.

Input:	contrl(1)	Op-Code = 28 (Return locator position)
	contrl(2)	Anzahl der Koordinatenpaare = 1
	ptsin(1)	Start X-Koord. des Locators
	ptsin(2)	Start Y-Koord. des Loactors
Output:	contrl(3)	Anzahl der Ausgangskoord. = 1
	contrl(5)	Anzahl der Terminierungszeichen = 1
	intout(1)	Terminierungszeichen
	ptsout(1)	Neue X-Koord. des Locators
	ptsout(2)	Neue Y-Koord. des Locators

Abfrage Modus:

Beim Einsprung in die Routine wird kein grafischer Cursor angezeigt. Es wird auf eine Eingabe von der Locator-Einheit geprüft. Falls ein gültiges Cursor-Steuerzeichen eingeben wird, so erfolgt die Übermittlung der neuen Koordinate an das Programm. Andere Eingaben werden als Terminator-Zeichen interpretiert und ebenfalls übergeben. Erfolgt keine Eingabe von der Locator-Einheit so passiert nichts. Es wird weder ein Terminierungszeichen noch eine neue Koordinate übergeben.

Input:	wie im Request-Modus	
Output:	contrl(3)	Anzahl der Ausgangskoordinaten: 0 = keine Koordinatenänderung 1 = Neue Koordinatenwerte
	contrl(5)	Anzahl der Terminierungszeichen: 0 = kein Terminator 1 = Terminierungszeichen wird zurückgegeben
	intout(1)	Terminierungszeichen falls vorhanden
	ptsout	Angabe falls neue Koordinaten
	ptsout(1)	Neue X-Koord. des Locators
	ptsout(2)	Neue Y-Koord. des Locators

****
INPUT VALUATOR: Übergeben des Wertes von der Zähleinheit
****

Der Valuator stellt eine Größe dar, die vom Anwender erhöht oder erniedrigt werden kann. Wie die Zeichen vom Anwender zur Valuator-Routine gelangen bestimmt 'VALUST' und 'VALUIN' im GLIBDATA-File. Welche Zeichen gültige Valuator-Zeichen sind, bestimmt die Tabelle 'VALCHA'. Standardmäßig werden die Valuator-Zeichen vom seriellen Tastaturkannal des System 8000 hereingeholt, als Steuerzeichen dienen standardmäßig die bekannten WordStar-Kodes:

	Langsam: (1ne Position)	Schnell: (10 Positionen)
Valuator erhöhen:	^E	^S
Valuator erniedrigen:	^X	^D

Request Modus:

Der Valuator wird entsprechend den eingegebenen Steuerzeichen vom Valuator-Device solange erniedrigt oder erhöht, bis ein ungültiges Zeichen (Terminierungszeichen) auftritt.

Input:	contrl(1)	Op-Code = 29 (Return valuator value)
	contrl(2)	= 0
	intin(2)	Startwert des Zählers
Output:	contrl(3)	= 0
	contrl(5)	Anzahl der Terminierungszeichen = 1
	intout(1)	Neuer Wert des Zählers
	intout(2)	Terminierungszeichen

Abfrage Modus:

In diesem Modus wird getestet, ob ein Zeichen von der Valuatoreinheit empfangen werden kann. Wird kein Zeichen eingegeben so erfolgt sofort ein Rücksprung. Ansonsten wird der Wert des Valuators angeglichen bzw. das Terminierungszeichen übermittelt, je nach dem ob gültige Steuercodes vorliegen oder nicht.

Input:	Wie im Request-Modus	
Output:	contrl(3)	= 0
	contrl(5)	Status: 0: Keine Eingabe 1: Zähler verändert 2: Terminierungszeichen
	intout(1)	Neuer Wert des Zählers falls Eingabe
	intout(2)	Terminierungszeichen falls Eingabe

****
INPUT CHOICE: Funktionstasten übergeben
****

Diese Funktion übergibt den Wert der Funktionstasten von der Choice-Einheit. Diese Einheit kann im GLIBDATA-File definiert werden ('CHOIST', 'CHOIIN'). Sie ist normalerweise der serielle Tastaturkanal des System 8000. Welche Werte als Funktionstatsten-Werte interpretiert werden kann in der Tabelle 'CHOICH' in GLIBDATA-File definiert werden. Dort kann ein Code-Bereich definiert werden, der alle gültigen Funktionstatsen umfasst. Normalerweise werden alle Werte von 0F0h bis 0FEH (einschließlich) als Funktionstasten interpretiert. Die Übergabe des Funktionstasten-Werts erfolgt nach Rücksetzen von Bit 7 des eingegebenen Werts. Wird z.B. eine Taste gedrückt, die den Code 0A0h erzeugt, so wird dies als Funktionstaste interpretiert und das Zeichen 020h an das aufrufende Programm übergeben.

Funktionstastencode:	Übermittelter Wert:
0F0h - 0FEh	000h - 07Eh

Request Modus:

Beim Einsprung in diese Funktion wird solange gewartet, bis eine gültige Funktionstaste eingegeben wurde. Der Werte der Funktionstaste wird übergeben.

Input:	contrl(1)	Op-Code = 30 (Return choice key)
	contrl(2)	= 0
	intin(1)	Funktionstasten Einheit Nr. = 1
Output:	contrl(3)	= 0
	contrl(5)	= 1
	intout(1)	Funktionstasten Wert von 0 bis 7Fh

Abfrage Modus:

Im Abfrage Modus wird geprüft, ob eine Funktionstaste betätigt wurde. Erfolgt keine Eingabe so passiert nichts, erfolgt eine gültige Eingabe, so

wird der Wert der Funktionstaste übermittelt.

Input: Wie im Request-Modus

Output:	contrl(3)	= 0
	contrl(5)	Status:
		0: Keine Eingabe
		1: Funktionstaste wurde betätigt
		2: Andere Taste wurde betätigt (Terminator)
	intout(1)	Funktionstasten-Wert falls eingegeben
	intout(2)	Terminierungszeichen falls andere Eingabe

****
INPUT STRING: Eingabe einer Zeichenkette
****

Diese Funktion erlaubt es Zeichenfolgen von der String-Einheit hereinzuholen und, falls gewünscht (Echo-Mode), sofort auf den Bildschirm auszugeben. Die String-Einheit kann im GLIBDATA-File definiert werden ('STRGST', 'STRGIN'). Die Stringeinheit ist normalerweise der serielle Tastaturkannal des System 8000.

Request-Modus:

Diese Operation übergibt eine Zeichenfolge von der String-Einheit. Es werden solange Zeichen empfangen, bis entweder der String-Puffer voll ist oder ein <CR> eingegeben wurde. Im Echo-Modus wird der eingegebene Text mit allen Textattributen (Richtung, Größe, Farbe ...) sofort am Bildschirm ausgegeben.

Input:	contrl(1)	Op-Code = 31 (Return string)
	contrl(2)	= 0 falls kein Echo-Modus
		= 1 falls Echo-Modus
	intin(1)	Stringeinheit Nr. = 1
	intin(2)	Maximale Stringlänge in Worten
		-> = max. Anzahl der Zeichen
	ptsin(1)	Start X-Koord. der Ausgabe auf Bildschirm falls Echomode
	ptsin(2)	Y-Koord. der Ausgabe

Output:	contrl(3)	= 0
	contrl(5)	Länge des Ausgang-Strings: 0: Keine Eingabe, nur <CR> >0: Anzahl der Zeichen im String (1 = 2 Bytes reserviert = 1 Zeichen)
	intout	String, ein Wort pro ASCII

Abfrage Modus:

Im Abfrage-Modus wird geprüft, ob ein Zeichen eingegeben wurde. Erfolgt keine Eingabe so passiert nichts. Erfolgt eine Eingabe so wird das Zeichen in intout abgespeichert und geprüft, ob ein weiteres Zeichen vorhanden ist. Die Routine wird abgebrochen falls:

* Keine weitere Eingabe erfolgt
* Ein <CR> empfangen wird
* Der intout-Puffer voll ist

Input:	contrl(1)	Opcode = 31 (Input string)
	contrl(2)	= 0
	intin(1)	Stringeinheit Nr. = 1
	intin(2)	Maximale String Länge in Worten -> = max. Anzahl der Zeichen
Output:	contrl(3)	= 0
	contrl(5)	Länge des Ausgang-Strings: 0: Keine Eingabe, nur <CR> >0: Anzahl der Zeichen im String (1 = 2 Bytes reserviert = 1 Zeichen)
	intout	String, falls Eingabe 1 Wort pro ASCII

****
SET WRITING MODE: Bestimmen des Schreibmodus
****

Diese Operation bestimmt die Art, wie der Bildspeicher beim Schreiben manipuliert wird. Der eingestellte Schreibmodus hat für alle nachfolgenden Schreibbefehle (Text, Area-Fill, Linie, Marker

1 = replace modus
2 = transparent modus (set)
3 = XOR (complement) modus
4 = erase modus

Ungültige Werte werden in Schreibmodus 1 (Replace) umgewandelt.

Input:	contrl(1)	Opcode = 32 (Set writing mode)
	contrl(2)	= 0
	intin(1)	Gewünschter Schreibmodus 1 - 4
Output:	contrl(3)	= 0
	intout(1)	Tatsächlich gewählter Modus

****
SET INPUT MODE: Setzen des Input-Modus
****

Diese Operation bestimmt den Input-Modus für die jeweilige Eingabe-Einheit Locator, Valuator, Choice und String. Für jede Einheit kann getrennt der Request- oder der Sample-Modus (= Abfrage-Modus) definiert werden. Im Request-Modus wird so lange in der jeweiligen Routine gewartet, bis eine gültige Eingabe von der jeweiligen Einheit erfolgt. Im Sample-Modus übergibt die jeweilige Funktion den Status der angesprochenen Einheit ohne auf gültige Eingaben zu warten.

Input:	contrl(1)	Opcode = 33 (Set input mode)
	contrl(2)	= 0
	intin(1)	Logische Eingabe-Einheit: 1 = Locator 2 = Valuator

		3 = Choice 4 = String
	intin(2)	Gewünschter Eingabe-Modus 1 = Request Modus 2 = Sample Modus (Abfrage)
Output:	contrl(3)	= 0
	intout(1)	Tatsächlich gewählter Modus

