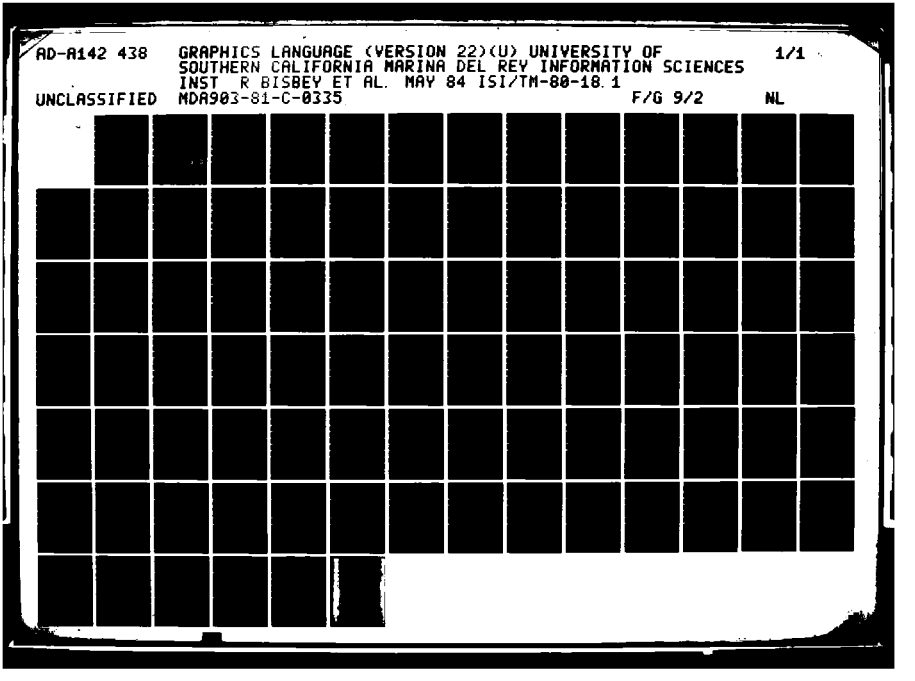
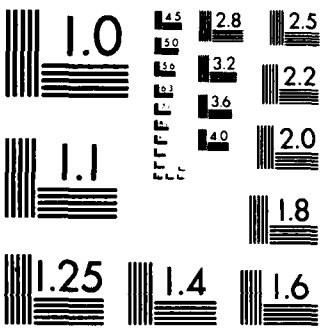

AD-A142 438 GRAPHICS LANGUAGE (VERSION 22)(U) UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY INFORMATION SCIENCES INST R BISBEY ET AL. MAY 84 ISI/TM-80-18.1
1/1
UNCLASSIFIED MDA903-81-C-0335
F/G 9/2
NL



ISI Technical Manual
ISI/TM-80-18.1
May 1984

Richard Bisbey II
Dennis Hollingworth
Benjamin Britt

University of Southern California

Graphics Language (Version 2.2)

INFORMATION SCIENCES INSTITUTE
213/822-1511
4676 Admiralty Way/Marina del Rey/California 90292-6695

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ISI/TM-80-18.1	2. GOVT ACCESSION NO. AD-A142438	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Graphics Language (Version 2.2)		5. TYPE OF REPORT & PERIOD COVERED Technical Manual 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR*(s)* Richard Bisbey II, Dennis Hollingworth, and Benjamin Britt		8. CONTRACT OR GRANT NUMBER*(s)* MDA903 81 C 0335
9. PERFORMING ORGANIZATION NAME AND ADDRESS USC/Information Sciences Institute 4676 Admiralty Way Marina del Rey, CA 90292-6695		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE May 1984 13. NUMBER OF PAGES 88
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

This document is approved for public release and sale; distribution is unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

.....

18. SUPPLEMENTARY NOTES

.....

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

command and control graphics, computer graphics, device-independent graphics system, high-level graphics language, language interface, on-line map display

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

(OVER)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

20. ABSTRACT (continued)

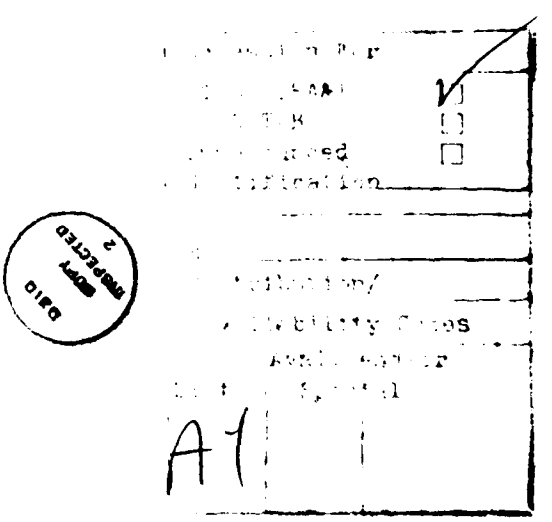
This document defines the language interface to Version 2.2 of a device-independent graphics system intended to facilitate the use of graphics in the command and control environment. The system homogeneously supports graphics terminals of widely varying capability, configured with one or more different programs running on separate computers connected via a C2 communications network. These include both calligraphic and bit-map displays as well as plotters. The set of graphics primitives defined here provides a core upon which device-independent application-tailored graphics packages can be built. Advanced graphics features such as color, shading, and input from multiple devices are supported in a manner that permits use of the same application program with devices not supporting those features. The graphics system performs the appropriate feature mapping to support the connected display device. For example, an application program may specify the color of a line segment. The system maps the specified color into some suitable color supported by the device or--in the case of a monochromatic display--into the single color of that device. An enquiry capability is also provided, permitting the application program to determine the characteristics of the connected display device and fully exploit all of its capabilities.

ISI Technical Manual
ISI/TM-80-18.1
May 1984

Richard Bisbey II
Dennis Hollingworth
Benjamin Britt

University of Southern California

Graphics Language (Version 2.2)



INFORMATION SCIENCES INSTITUTE

213/822-1511
4676 Admiralty Way/Marina del Rey/California 90292-6695

This research is supported by the Defense Advanced Research Projects Agency under Contract No. MDA903 81 C 0335. Views and conclusions contained in this report are the authors' and should not be interpreted as representing the official opinion or policy of DARPA, the U.S. Government, or any person or agency connected with them.

Contents

Acknowledgments

We wish to acknowledge Danny Cohen for his invaluable help and guidance in defining GL and Robert F. Sproull for his development of the OMNIGRAPH system, after which the syntax of GL is modeled. Also, thanks to Rick Shiffman who wrote and documented the TOPS-20 Interlisp Interfaces.

General Information

This document defines the language interface to Version 2.2 of a device-independent graphics system intended to facilitate the use of graphics in the command and control environment. The system homogeneously supports graphics terminals of widely varying capability, configured with one or more different programs running on separate computers connected via a C2 communications network. These include both calligraphic and bit-map displays as well as plotters. The set of graphics primitives defined here provides a core upon which device-independent application-tailored graphics packages can be built. Advanced graphics features such as color, shading, and input from multiple devices are supported in a manner that permits use of the same application program with devices not supporting those features. The graphics system performs the appropriate feature mapping to support the connected display device. For example, an application program may specify the color of a line segment. The system maps the specified color into some suitable color supported by the device or--in the case of a monochromatic display--into the single color of that device. An enquiry capability is also provided, permitting the application program to determine the characteristics of the connected display device and fully exploit all of its capabilities.

The graphics system provides constructs for

- Initiating and terminating a connection with the desired display device.
- Allocating a graphics output area of a specified aspect ratio on the display device viewing surface.
- Defining a viewport (subarea within the allocated area of the display surface) and user coordinate system to be mapped to that viewport.
- Creating, merging, destroying, displaying, and erasing named segments.
- Generating graphics entities such as lines, dots, text, arcs, and shaded polygons and sectors.
- Controlling display characteristics of graphics elements (e.g., intensity, color, text face and shading parameters).
- Accepting various forms of input from the terminal.
- Retrieving device/system status information.
- Sending and receiving device-specific orders.
- Storing pictures in and retrieving pictures from Graphics Files.

This document is a language reference manual for Graphics Language. Appendixes contain specifics regarding particular implementations of this language for specific languages and operating systems.

The following comments are useful for understanding Graphics Language.

1. The run-time environment consists of

   (1A) An application program written in FORTRAN, BLISS, LISP, C or some other programming language supported by the graphics system,

   (1B) The graphics system,

   (1C) The display system.

   The interface between (1A) and (1B) is defined within this document. It should be thought of as a set of subroutine/procedure calls, rather than as a programming language.

   The interface between (1B) and (1C) depends on the order codes of the particular device on which the graphics is being generated, and may, but does not necessarily, include transnet communication.

2. The language separates device-independent from device-dependent issues. Thus, issues like [a] the size and position of the TERMINAL display-viewport on the CRT, [b] repainting of a storage tube, and [c] function key assignment are all handled independently of application program interaction.

3. The user (i.e., the programmer using Graphics Language) supplies sufficient information to the graphics system to allow it to identify and connect to the device, and also requests allocation of a graphics output area of desired aspect-ratio on the display surface via the INITIATE command. He specifies either explicitly or implicitly his coordinate system, the WINDOW, and separately, a sub-area within the allocated area on the CRT surface to be used, the VIEWPORT. The system always maps the entire WINDOW onto the entire VIEWPORT, even if this transformation is not a conformal mapping (i.e., introduces "stretching"). For details of these specifications, see WINDOW and VIEWPORT.

4. The smallest nameable display entity is a segment. Segments have user-assigned unique IDs. Segments are created by issuing an OPEN call followed by any sequence of graphic primitive calls (e.g., MOVE, DRAW, DOT, TEXT) and terminated by CLOSE. Once a segment has been created, it can be made visible by issuing POST, or invisible by UNPOST. A segment may be POSTed or UNPOSTed many times. Segments may be MERGEd into a single segment. When the segment is no longer needed, it is KILLed to release memory associated with its graphic primitives and to free the ID.

5. The Graphics Language deals with absolute transformed display segments only. All transformations (e.g., those resulting from the VIEWPORT/WINDOW relation) are performed when the segment is defined. No transformation may be applied to segments

already generated. As a result, the effect of motion can be achieved only by replacing already displayed entities with newly generated ones.

6. The system maintains status information that includes important parameters that are available to the user. These include error reports, scope size, display system capabilities, etc. A complete list of these is shown later, under the description of the ENQUIRE command.

Graphics Language Calls

This section contains information on the various Graphics Language (GL) calls available to the application programmer. The calls are grouped in ten sections according to their role in creating the user's graphic output. Each call is presented in terms of its intended effect; any side effects that may result; the order, type, and value range of required parameters; and any error conditions that might result from improper use of the call.

[1] DEVICE CONNECTION INITIATION AND TERMINATION

The following GL calls are used to initiate and terminate a connection with a display device. Initiating a connection results in the binding of various components of the graphics system with the application program and the requested display device. It also establishes a number of defaults for various GL commands discussed later in this document. These defaults include the user's coordinate system, the area on the display surface on which the user's graphic output will appear, and the display attributes for graphic primitives such as text and vectors. The user may override these defaults or establish his own default values via commands discussed in subsequent portions of this document.

INITIATE (CONFIGURATION-STRING, ASPECT-RATIO)

Establish a connection to and initialize the display device, reset all buffers and parameters to their default values, and retrieve device-specific information from the display device. The first parameter is a character string, the second is a real number.

CONFIGURATION-STRING specifies system configuration and display-device information, i.e., the type and location of graphic system modules, the display device type, location, and connection protocol. The format of this information depends on the environment in which the system is implemented. See Appendix F for further information.

ASPECT-RATIO specifies the desired aspect ratio for the area on the display surface to be allocated for this connection as a real number. Values less than 0. indicate that the default aspect ratio for the viewing surface should be used; values greater than 0. indicate the ratio of the width to the height of the desired area. Thus, the value .5 requests a working area in which the vertical size is twice the horizontal size, and a value of 2.0 requests a working area in which the horizontal size is twice the vertical size. An area of the desired aspect ratio, the *allocation-viewport*, will be assigned to the application from the available working area on the device surface. It will be given an address range of (0.,0.,W,H) for purposes of subsequent VIEWPORT calls where the smaller of the pair (W,H) is 1.0 and the larger is proportionally greater according to the aspect-ratio requested. The values of W and H are made available to the user via elements 14 and 15 of the enquiry status information (see ENQUIRE). The physical size and actual location of the allocation-viewport is implementation and

connection dependent. Its size in centimeters is made available to the application via words 12 and 13 of the enquiry information.

As a result of this call, various device and connection specific information, including the information specified above, is made available to the application program via the ENQUIRE call.

Possible errors: ERR-06 Device not known to the system or not available.

RELEASE

Release the display device and system modules being used. Terminate any networ[illegible]nections.

Possible errors: None.

[2] VIEWING AREA AND COORDINATE SYSTEM SELECTION

The following commands specify the subarea *(viewport)* of the allocation-viewport in which the user's graphic output will appear and the X,Y bounds of the user's coordinate system *(window)* to be mapped to that viewport. Graphic primitives that extend outside of the specified X,Y range of the window will be clipped at the specified boundaries; portions outside of the window/viewport boundary will not appear on the display surface. The viewport/window pair allows the application programmer to both scale and translate his pictures to any area of the display surface allocated to his program. Depending upon the values of the window and the viewport, the user's picture may be distorted (stretched) in either the X or the Y direction. Choice of appropriate window values allows the application to selectively view and/or enlarge particular portions of a specified picture.

VIEWPORT (XL, YB, XR, YT)

Select a subarea of the allocation-viewport in which subsequent graphics will appear. All the arguments are real numbers and must range from 0. to the maximum for the allocation-viewport assigned during the INITIATE call as identified via words 14 and 15 of the enquiry information.

The largest left- and down-justified square of the allocation-viewport is defined by the value range (0.,0.,1.,1.). In this coordinate system the top half of this square is defined by (0.,.5,1.,1.).

This call does not affect any segment (or part thereof) already generated. The current beam position of an OPEN segment remains at the same location in the allocation-viewport after a VIEWPORT call as it was prior to that call. The values for the current beam position within the old coordinates are adjusted to correspond to the values for the new VIEWPORT call.

The system initialization default is VIEWPORT(0.,0.,1.,1.). Thus, if no VIEWPORT call is issued by the application program, the largest left- and down-justified square from the allocation-viewport is used.

It is not legal to have XL = XR, YB = YT, XL>XR, or YB>YT.

Possible errors:	ERR-07	The specified values are out of the allowable range as defined by the allocation-viewport or XL = XR, YB = YT, XL>XR or YB>YT.

WINDOW (XL, YB, XR, YT)

Define the user coordinate system for the current VIEWPORT. Values are specified as real numbers. The identified coordinate range is mapped onto the viewport. User graphics outside the specified coordinate range will be clipped at the window boundaries to include only that portion within the WINDOW coordinates.

The call does not affect any segment (or part thereof) already generated. The current beam position of an OPEN segment in the old coordinate range is adjusted to correspond to the coordinates of the new WINDOW call. The call may be reissued whenever needed, even within a segment.

The system initialization default is WINDOW(0.,0.,1.,1.). Thus, if no WINDOW call is issued by the application program, any vectors, text, etc., with start and end points within this value range will appear on the screen while those with start or end points outside of this range will be clipped at the window edges.

It is acceptable to have XL>XR or YB>YT, in order to achieve *mirroring*. However, XL = XR or YB = YT is not acceptable and results in an error.

Possible errors: ERR-07 XL = XR or YB = YT.

[3] SEGMENT SPECIFICATION

The following GL calls allow the application programmer to name and define the contents of an individual GL *segment*. A segment is a named collection of GL primitives (lines, dots, text, arcs, polygons, sectors, etc.). Once created, segments can be merged with other segments, renamed, made visible or invisible, highlighted, made touch sensitive, and destroyed. These operations are discussed in section 4 of this document.

Segment Identification

A given set of graphic primitives is associated with a segment by virtue of the following two calls that both delimit the bounds of the segment and assign a name by which the application program can subsequently refer to that segment.

OPEN (N)

Initiate specification of a segment with ID N. All subsequently specified graphic primitives (lines, text, dots, etc.) are to be associated with the segment named N until a CLOSE is issued, i.e., segment N consists of all graphics primitives specified between the OPEN and CLOSE calls. The ID, N, is an integer between 1 and 32000. If any other segment is still open when an OPEN is issued, then a CLOSE (see below) is implied for that segment.

Segments are always initialized with the segment default conditions in effect (color and text attributes) when this call is issued. These default conditions are set during system initialization to the values indicated in the discussion of the particular calls (see COLOR and TEXTFACE below). They may be changed via the DEFAULT-COLOR and DEFAULT-TEXT calls discussed in section 9 of this document.

When this GL call is issued the current beam position is undefined, and its value, as available through an ENQUIRE call, is not necessarily valid.

Possible errors:	ERR-02	System table overflow.
	ERR-16	No scope selected.

CLOSE

Terminate specification of the currently open segment. If a segment with the same ID, N, already exists, it is replaced by this segment, which assumes its display attributes. If the old segment was visible, (i.e., POSTed), its replacement will be too. If the old one was HIGHLIGHTed, the new segment will also be highlighted. Otherwise the segment is initially invisible and nonhighlighted. If there is no open segment, no action is taken.

Possible errors: ERR-02 System table overflow.

Display Mode Specification

The following set of GL calls indicates the preferred manner for subsequently specified graphic primitives to appear on the display surface. Whether or not the displayed graphic primitives actually appear according to the specified mode settings depends upon the capabilities of the graphics system and the display device.

COLOR (I, R, G, B)

Set the intensity and chromaticity for the rest of this segment. I, R, G, and B are real numbers. I determines the intensity.

I = 1. Highest available intensity (same for I > 1.)

I = 0. Lowest available intensity (same for I < 0.)

For any 0. < I < 1. the system will choose an appropriate intensity level.

The R, G, B values determine the chromaticity (hue and saturation). Since chromaticity is uniquely determined by only two of the R, G, B values, at least one of the values must be zero. If the user specifies all three values greater than zero, the system maps at least one to zero by the following computation:

COLOR (I, R-min(R,G,B), G-min(R,G,B), B-min(R,G,B))

The ENQUIRE call (see Section 7) may be used to find the available intensities and colors. The initialization default for each segment is COLOR(1.,0.,0.,0.).

Possible errors: ERR-02 System table overflow.
ERR-03 No open segment.

INTENSITY (I)

Set the intensity level for the remainder of this segment. I is specified as a real number.

I = 1. Highest available intensity (same for I > 1.)

I = 0. Lowest available intensity (same for I < 0.)

For any 0. < I < 1. the system will choose an appropriate intensity level. This call is equivalent to COLOR(I.,0.,0.,0.). The ENQUIRE call (see Section 7) may be used to determine the range of intensities available. The initialization default for each segment is INTENSITY(1.).

Possible errors:

ERR-02	System table overflow.
ERR-03	No open segment.

TEXTFACE (MASK, NAME, QUALITY, HEIGHT, WIDTH, VERTICAL-SPACING, HORIZONTAL-SPACING)

Select a text face/font based upon the indicated attributes. Use the VERTICAL-SPACING and HORIZONTAL-SPACING values (if specified) to perform intercharacter spacing.

The MASK parameter indicates which of the subsequent variables are set and, hence, are to be used in the text font selection process or intercharacter spacing. If a variable is not set, then the default for that field is used. Values of the MASK parameter range from 0 to 63 as follows:

MASK = 0: None of the subsequent parameters are to be included in the text font selection process; use the device default text face/font. The default intercharacter spacing values for the selected font are to be used for positioning characters.

= 1: The text NAME parameter is set and may be included in the font selection process.

= 2: The text QUALITY parameter is set and may be included in the font selection process.

= 4: The HEIGHT parameter is set and may be included in the font selection process.

= 8: The WIDTH parameter is set and may be included in the font selection process.

= 16: The VERTICAL-SPACING parameter is set; intercharacter vertical spacing is to be performed according to the value of the VERTICAL-SPACING parameter.

= 32: The HORIZONTAL-SPACING parameter is set; intercharacter horizontal spacing is to be performed according to the value of the HORIZONTAL-SPACING parameter.

Combinations of these values are used to specify that more than one of the parameters are to be included in the font selection/intercharacter positioning process. The manner in which specific parameters are employed in the font selection process is described below.

The NAME parameter indicates the name of the desired text face to be used, e.g., BODONI, NEWS GOTHIC READER, TIMES ROMAN (see Appendix E). If the value is not used in the font selection process (i.e., the corresponding bit in the mask variable is set to zero), the default text face is used. The QUALITY parameter indicates the type of text required (0 = hardware, 1 = stroked). Stroked text is automatically scaled to the specified HEIGHT and WIDTH as described below. Hardware text is matched to the height and width in that order. The HEIGHT parameter is used to indicate the desired text height in window units (real number) of the selected text font. The WIDTH parameter is used to indicate the desired text width in window units (real number). In the case of hardware generated text, if the HEIGHT parameter is set, the hardware-supported text font (of the specified text face) whose

height comes closest to the specified height without actually exceeding it is selected. In this case the WIDTH parameter is ignored. If the HEIGHT parameter is not set, then the text font whose width comes closest to the specified width without exceeding it is selected.

Since most hardware character generators cannot generate characters of arbitrary size, it is likely that for hardware text the actual character height and width used will differ from the specified height and width. In order to find the actual character height and width that result from this call, an ENQUIRE call (see Section 7) must be issued.

The actual text font selected for a given TEXTFACE command depends upon both the parameters of the TEXTFACE command and the actual hardware and stroked fonts that are supported. An attempt is first made to find a matching font of the specified quality and the specified face name. If this attempt fails, then an attempt is made to find a font of any quality of the specified face. If this attempt also fails, then an attempt is made to find a font of the specified quality but of any face. Finally if the previous attempts have failed, a search is made for a font of any face name and any quality that satisfies the text size constraints.

If the VERTICAL- and/or HORIZONTAL-SPACING parameters are set, characters are individually positioned according to the values of these parameters. The VERTICAL-SPACING parameter indicates the value in window units (real number) by which a character is to be displaced in the vertical direction from the previous character. If this value is positive, then subsequent characters will be displaced in an upward direction from the preceding character. If the value is negative, then subsequent characters will be displaced in a downward direction from the preceding character. The value of the HORIZONTAL-SPACING parameter indicates the amount in window units (real number) by which a character is to be displaced from the right-hand edge of the previous character. If the value is positive, then the displacement will be to the right, if the value is negative, then the displacement will be to the left. For example, if the VERTICAL-SPACING value is the negative of the character height and the HORIZONTAL-SPACING value is the negative of the character width, then a descending vertical string of characters will be generated.

The initialization values for text size and spacing are the device default values.

Possible errors:	ERR-02	System table overflow.
	ERR-03	No open segment.
	ERR-11	No font small enough.

Vector Specification (Absolute Coordinate Form)

The following commands specify graphic primitives in terms of the absolute window coordinates involved.

DOT(X,Y)

Display a dot at the specified position (in the user's coordinate system). X and Y are given as real

numbers. The beam position is left at the specified point. The dot is displayed only if it is inside the window. It is displayed according to the current intensity/color setting.

Possible errors:	ERR-02	System table overflow.
	ERR-03	No open segment.

DRAW (X, Y)

Draw a line from the current beam position to the specified position. X and Y are given as real numbers. The beam position is left at the specified point. The line is clipped, if needed. It is displayed according to the current intensity/color setting. If the current beam position is undefined, then the results of this call are undefined.

Possible errors:	ERR-02	System table overflow.
	ERR-03	No open segment.

MOVE (X, Y)

Position the beam position at the specified point. X and Y are given as real numbers.

Possible errors:	ERR-02	System table overflow.
	ERR-03	No open segment.

LINE (X, Y, I)

MOVE or DRAW. If I = 0, then this call is equivalent to MOVE(X,Y); if I = 1, then it is equivalent to DRAW(X,Y). As in MOVE(X,Y) and DRAW(X,Y), X and Y are specified as real, absolute coordinates, I as an integer value. The display characteristics of the line is affected by the intensity/color setting. The line is clipped, if necessary.

Possible errors:	ERR-02	System table overflow.
	ERR-03	No open segment.

ARC (X, Y, R, SA, EA)

Draw a circular arc with radius R and center-point (X,Y), starting at angle SA and proceeding in a counter-clockwise direction through angle EA. All the arguments are given as real numbers. X, Y, and R are specified in the user's (window) coordinates; SA and EA specify the starting and ending angles in radians, with 0 corresponding to "right" and PI/2 to "up". The number of straight lines used to approximate the arc may vary according to the system resolution and the value of R. The arc is drawn according to the current intensity/color setting and is clipped, as necessary.

Note that if the aspect-ratios of the WINDOW and the VIEWPORT differ, then circular arcs in the user's coordinates are elliptic arcs in the VIEWPORT coordinates, and this call is not necessarily supported.

Possible errors:		
	ERR-02	System table overflow.
	ERR-03	No open segment.
	ERR-07	R<0 or R = 0.
	ERR-16	The arc is not circular (but elliptic).

Vector Specification (Relative Coordinate Form)

The following commands specify graphic primitives in terms of the X and Y displacement in window coordinates from the current beam location. They are most commonly used in creating subroutines for repeating the same set of graphic primitives independent of the current beam location. In such cases the routine would be called to generate the graphic object after the main program had positioned the beam at the desired origin. If the beam position for the currently open segment is undefined, the result of these calls is undefined.

RELATIVE-DOT (DX, DY)

Display a dot in the position specified relative to the current beam position. DX and DY are given as real numbers. The current beam position becomes the specified point. If the beam position immediately prior to this call being issued is (X,Y), then after this call it is (X + DX,Y + DY). The dot is displayed according to the current intensity/color setting, but only if it is inside the window.

Possible errors:		
	ERR-02	System table overflow.
	ERR-03	No open segment.

RELATIVE-DRAW (DX, DY)

Draw a line from the current beam position to the point (X + DX,Y + DY) where (X,Y) is the beam position before the call. DX and DY are given as real numbers. The line is displayed according to the current intensity/color setting, and is clipped, as necessary.

Possible errors:		
	ERR-02	System table overflow.
	ERR-03	No open segment.

RELATIVE-MOVE (DX, DY)

Displace the beam from the current beam position (X,Y) to (X + DX,Y + DY). DX and DY are given as real numbers.

Possible errors:	ERR-02	System table overflow.
	ERR-03	No open segment.

RELATIVE-LINE (DX, DY, I)

RELATIVE-MOVE or RELATIVE-DRAW. If I = 0, then this call is equivalent to RELATIVE-MOVE(DX,DY); otherwise it is equivalent to RELATIVE-DRAW(DX,DY). As in RELATIVE-MOVE(DX,DY) and RELATIVE-DRAW(DX,DY), DX and DY are supplied as real values in relative coordinates. I is given as an integer value. It is displayed according to the current intensity/color setting and is clipped, as necessary.

Possible errors:	ERR-02	System table overflow.
	ERR-03	No open segment.

RELATIVE-ARC (R, SA, EA)

Draw a circular arc relative to the current beam position. This call is identical to ARC(X,Y,R,SA,EA) where (X,Y) is the current beam position.

Text Specification

The following routine allows the user to specify a text string to appear on the display surface. The text string is clipped according to the window specification and the location of characters with regard to the window. The size of the characters actually generated depends upon the results of the TEXTFACE command issued by the user or the default TEXTFACE command issued by the system.

TEXT (string)

Show a text string starting at the current beam position. The current beam position becomes the lower left corner of the first character. The beam position is left at the position computed by adding the vertical and horizontal spacing values to the lower right corner of the last character (equivalent to the lower left of the next character position).

The text is displayed according to the current intensity/color and textface settings. Text clipping eliminates any character not totally within the window. If the current beam position is undefined, then the results of this call are undefined.

Possible errors:	ERR-02	System table overflow.
	ERR-03	No open segment.

Area Specification

The following commands allow the application programmer to specify areas in his coordinate system to be filled according to specified parameters. The visible result of the area fill request depends on the graphics system and the display device. In particular, solid filling may not be supported on certain types of devices and may be defaulted to single or double hatching.

FILL-SECTOR (X, Y, R, SA, EA, MODE, ANGLE, DIST)

Fill the specified sector. X, Y and R are specified as real numbers in the user's coordinate system and correspond to the center and radius of the sector. SA and EA represent the starting angle and ending angle (in radians) for the sector where SA = 0.0 corresponds to a ray emanating from the center of the circle and oriented to the right. The sector is constructed by proceeding in a counter-clockwise fashion from SA until the value EA is reached. The sector described by X, Y, R, SA and EA is filled according to the mode, angle and distance values as follows:

MODE = 0 implies no fill pattern,
MODE = 1 implies single hatching,
MODE = 2 implies cross-hatching,
MODE = 3 implies solid filling.

ANGLE is the angle in radians for subsequent hatching marks for MODE = 1 and MODE = 2. DIST is the perpendicular distance between consecutive hatching marks in window units.

This operation is performed according to the current intensity and color settings. Clipping is applied as needed. The beam position for the currently opened segment remains unchanged.

Note that if the aspect ratios of the WINDOW and the VIEWPORT differ, then this call does not specify a circle and is not necessarily supported.

Possible errors:	ERR-02	System table overflow.
	ERR-03	No open segment.
	ERR-07	R<0 or R = 0.
	ERR-16	The specified area is not circular (but elliptic).

FILL-POLYGON

VERTEX (X, Y)

TERMINATE-POLYGON (MODE, ANGLE, DIST)

Fill the following N-vertex polygon. FILL-POLYGON indicates the start of a polygon vertex specification sequence. The polygon is defined by its N vertices (in user coordinates), specified by N consecutive VERTEX calls progressing from an arbitrary vertex around the polygon in either a clockwise or counter-clockwise direction. Polygon specification is terminated by a TERMINATE-

POLYGON call, which also specifies the filling parameters in identical fashion to the FILL-SECTOR call. The polygon must be a simple closed curve (i.e., does not intersect itself). The Nth vertex is connected to the first.

This operation is performed according to the current intensity and color settings. Clipping of the polygon is performed as necessary. The beam position for the currently opened segment remains unchanged.

Possible errors:	ERR-02	System table overflow.
	ERR-03	No open segment.
	ERR-07	N<3.
	ERR-16	Bad polygon specification.

[4] SEGMENT CONTROL

The following set of GL calls manipulate segments that have been created via the SEGMENT SPECIFICATION calls. Calls are provided to merge two existing segments, rename a segment to a new name, make segments visible or invisible, highlight segments, make the visible portions of segments sensitive to touch detection, and delete segments from the system.

MERGE (N, M)

Merge segment N into segment M. Display entities in N are catenated to M under the display attributes of M. Segment N no longer exists after this operation. If segment M is not defined, this call renames segment N to M.

Segment N and M must have identical scope attributes. Any queued input for segment N is subsequently associated with segment M.

Possible errors:	ERR-02	System table overflow.
	ERR-04	Segment N does not exist, N = 0, or M = 0.
	ERR-16	Dissimilar scopes.

POST (N)

Make segment N visible. If segment N is already visible, i.e., POSTed, no further action is taken. POST(0) results in POSTing all the segments.

Possible errors:	ERR-02	System table overflow.
	ERR-04	Segment N does not exist (for N>0).

UNPOST (N)

Make segment N invisible. If segment N is not currently visible, i.e., POSTed, no action is taken. UNPOST(0) results in UNPOSTing all the segments.

Possible errors:	ERR-02	System table overflow.
	ERR-04	Segment N does not exist (for N>0).

HIGHLIGHT (N, C)

Set HIGHLIGHT mode for segment N. Highlighting may be implemented in different ways, such as blinking, bold width, a change in color or intensity, underlining (for text), etc. C is specified as an integer value as follows:

C = 1 Highlight Segment N

C = 0 Normal mode

N = 0 specifies all segments. If segment N does not exist, this call results in an error.

Possible errors:	ERR-02	System table overflow.
	ERR-04	Segment N does not exist (for N>0).

TOUCH (N, K)

Activate/deactivate segment N for detection by the "touching" device (F = 3). K is specified as an integer value.

K = 1: Activate segment N for "touching"

K = 0: Deactivate segment N for "touching"

N = 0 specifies all currently defined segments. Newly created segments are not touch sensitive until specifically made so. Similarly, when an existing segment is replaced by a new one with the same ID, the new one is not touch sensitive until explicitly made so. Upon killing a segment, touch-sensitivity for that segment terminates.

When HOLD mode is on, segments that are UNPOSTed or KILLed might still be displayed. Only segments that are POSTed, and not UNPOST- or KILL-pending, and TOUCH-sensitive may be detected by the touching device (F = 3).

Possible errors:	ERR-02	System table overflow.
	ERR-04	Segment does not exist.
	ERR-08	No pointing device is available.

KILL (N)

Remove segment N from the display system. UNPOST the segment if currently POSTed, and reclaim its storage and ID. This is an irreversible operation, which makes the system "forget" about segment N. KILL(0) results in KILLing all currently defined segments.

Possible errors:	ERR-02	System table overflow.
	ERR-04	Segment N does not exist (for N>0).

[5] UPDATE CONTROL

The following two GL calls allow the application to control undesirable intermediate updating of the display. If these calls are not used, the display will be updated and regenerated for each segment changed in a series of operations, with the result that intermediate or incomplete pictures may be displayed.

HOLD (K)

Hold/Don't hold subsequent display erasures. Use of HOLD mode allows the user's application program to group display erasures such that they occur simultaneously. (This feature is often employed to minimize "erasure flash" on storage-tube display terminals.)

When in HOLD mode, the graphics system suspends display erasures until a DONE is issued. When not in this mode, display erasures are performed as encountered.

K = 0: HOLD mode not in effect.

K = 1: HOLD mode in effect.

HOLD(0) is the system initialization default.

Possible errors: None.

DONE

Perform all pending display erasures. This does not modify the HOLD mode.

Possible errors: None.

[6] GRAPHICS FILES

The following GL calls allow the application programmer to create a file containing a device-independent description of one or more user-named segments and to incorporate the contents of such a file in a graphic application. Calls are also provided so that the user can determine file specific information.

READ-FILE (FILENAME, U, H)

Read the specified graphics file. FILENAME contains the name of the file as a text string. U and H contain the names of the segments by which the unhighlighted and highlighted portions of the file can subsequently be identified. Segments previously named either U or H are deleted (KILLed). Segments U and H are left in the invisible (UNPOSTed), nonhighlighted state. The file is positioned within the current viewport.

Possible errors:	ERR-13	File does not exist, File could not be opened or read, File was not a graphic file, File protocol was incompatible, System table overflow, or No scope selected.

OPEN-OUTPUT-FILE (FILENAME, COMMENT)

Open graphics file for output. FILENAME contains the name of the file as a text string; COMMENT contains a user-specified text string. The comment string is added to the file and may be retrieved by READ-FILE-ATTRIBUTES.

Possible errors:	ERR-13	File already open, or File could not be opened.

WRITE-SEGMENT (N)

Write segment N in the currently opened output file. The segment is placed in either the unhighlighted or highlighted portion of the file depending on its highlight state. WRITE-SEGMENT (0) results in writing all visible, i.e., POSTed, segments in the file.

Possible errors:	ERR-04	Segment N does not exist.
	ERR-13	No output file open.

CLOSE-OUTPUT-FILE

Close the currently opened graphics output file, making it available for graphics input.

Possible errors: ERR-13 No output file open.

READ-FILE-ATTRIBUTES (FILENAME, X, Y, MAX-CHARS, NO-CHARS, COMMENT)

Obtain the precision and comment information from a graphics file. FILENAME contains the name of the file as a text string, while MAX-CHARS contains the maximum number of characters that can be returned as the file comment. The precision of the file is returned as integers in X and Y, and is generally used in conjunction with the VIEWPORT command for controlling placement of the file input on the display surface. The file comment is returned in COMMENT as a text string. NO-CHARS is set to the number of comment characters returned. If the number of characters of the file comment exceeds MAX-CHARS, the text string is truncated to MAX-CHARS and NO-CHARS is set to MAX-CHARS.

Possible errors: ERR-13 File does not exist,
File could not be opened or read,
File was not a graphic file, or
File protocol was incompatible.

[7] INFORMATION ENQUIRY

GL maintains and makes available to the user's application program status information regarding the current state of the graphics system as well as the characteristics of the device to which the application is connected. This permits the application program to save significant information about the current state of the graphics system, issue GL calls that alter its state, and subsequently restore the graphics system to its previous state (within certain limits). It also allows the application program to determine the type and nature of any errors that have occurred during program execution since the last time the information was accessed. Finally, it allows the application program to determine specific features about the device to which it is connected in order to optimize visual presentation. The graphics system maps application program graphic calls into features supported by the display device to which it is connected. This mapping may not always be optimal for a given application. By retrieving device-specific and connection-specific information the application program can determine the features available and tailor its calls to achieve its own feature mapping.

ENQUIRE (K, VALUE)

Enquire about the status of the system and current values of parameters. The system retains state information that is accessible via the ENQUIRE call. The enquiry call retrieves the requested state data item (K) and stores it according to the location(s) indicated by VALUE. State values can be modified by the application program only via GL calls that result in parameter changes and by the ERROR call.

The following list specifies the parameters, their type, and the side effects (if any) of the ENQUIRE call.

ERROR INFORMATION

The following three words reflect any error information logged since the previous access to this information. As each word is accessed, the corresponding information is returned to the user. Subsequent access to the same word or issuance of another GL call that accesses any other enquiry information or invokes any other graphic function results in the error information being updated and the new data made available.

K = 0: Error criticality. (integer)

The returned status element indicates the criticality of errors that have occurred since the last access. The value ranges from 0 to 15 with individual bit-positions taking the following meaning:

1. Warning: Program request not supported due to nonexistent feature.

2. Minor Error: Possible picture distortion or minor data loss (e.g., bad

parameter value).

3. Serious Error: Major data loss (e.g., system table overflow).

4. Fatal Error: System cannot continue (e.g., device connection failure).

K = 1: Error code. (integer)

The returned status information indicates the cumulative effect of errors occurring since the last access to the error information and takes a value from 0 to 65535. The definition of individual bit-positions are contained in Appendix G.

K = 2: Error subcode. (integer)

If less than 32000, this identifies the segment in error for the most severe error criticality reported (if any segment was involved). Otherwise, it represents an implementation-dependent code giving specific information about the most severe error.

DEVICE CHARACTERISTICS

The following details provides specific information about the device to which the application is connected.

K = 3: Highlighting availability. (integer)

A 16-bit field indicating the availability of highlighting on individual scopes. The low-order bit position indicates highlighting availability on scope 1 with progressively higher bit positions indicating highlighting availability on scopes 2 through 16.

K = 4: Keyboard availability. (integer)

A word indicating whether or not a keyboard is available to the application for input. If the value is nonzero, then a keyboard is available.

K = 5: Positioning availability. (integer)

An integer value indicating whether or not pointing or positioning input is available. If the value is nonzero, pointing input is available.

K = 6: Function key availability. (integer)

A word containing a count of the number of function keys available for function key input.

K = 7: Touching availability. (integer)

A word indicating whether or not segment touching is available. The value 0 indicates

that segment touching input is unavailable; 1 indicates that segment touching is supported but that only the segment id is returned; 2 indicates that both the segment id and touch location are returned.

K = 8: Analog device availability. (integer)

A word indicating whether or not analog input capability is available to the application program. If the value is nonzero, then an analog input device is available.

K = 9: Cursor availability. (integer)

An integer value indicating whether or not cursor positioning capability is available. If the value is nonzero, cursor positioning is supported.

K = 10: X resolution. (integer)

The resolution along the X-axis of the aliocation-viewport.

K = 11: Y resolution. (integer)

The resolution along the Y-axis of the allocation-viewport.

K = 12: X dimension. (real)

The dimension in centimeters along the X-axis of the allocation-viewport.

K = 13: Y dimension. (real)

The dimension in centimeters along the Y-axis of the allocation-viewport.

K = 14: X port max. (real >= 1.0)

The maximum X viewport value for the allocation-viewport.

K = 15: Y port max. (real >= 1.0)

The maximum Y viewport value for the allocation-viewport.

K = 16: Number of scopes. (integer)

A 16-bit scope mask indicating the scopes available. The low-order bit corresponds to scope # 1. Successive bits correspond to scopes 2 through 16.

K = 17: Device code. (integer)

A word indicating the graphics device type (e.g., TEKTRONIX, GENISCO).

K = 18: Device type. (integer)

A word indicating specific display properties of the connected device.

K = 19 - 20: Unused.

COLOR INFORMATION

Color Data Request Word

K = 21: Color information request control word.

A trigger word that when accessed causes requested color information to be stored in the area indicated by the return location parameter for the enquiry request. The type of color information request is indicated by specific data values in the user-specified area as follows:

Word1 = 1 : Enquire on scope data.

Word2 = Scope number.

Returns in word 3 the number of entries in the color table for this scope. (Words 1 and 2 left unchanged.)

Word1 = 2 : Enquire on color entry for specified scope.

Word2 = Scope number.

Word3 = Color entry number: ranging from 1 to the value returned in word 3 by an enquiry on scope data request.

Returns in words 4 through 8 the following (see COLOR call).

Word4: Intensity value. (0. < = real number < = 1.)

Word5: Red value. (0. < = real number < = 1.)

Word6: Green value. (0. < = real number < = 1.)

Word7: Blue value. (0. < = real number < = 1.)

Word8: Highlight indicator. (integer) 1 implies highlighting, 0 implies no highlighting.

Color Default Values

These values are used by OPEN to initialize a segment. They are set at system initialization time (see COLOR) and may be reset to user-specified values via the DEFAULT-COLOR call.

K = 22: Intensity default. (0. < = real number < = 1.)

A real number from 0. to 1. indicating the OPEN default intensity setting.

K = 23: Red default. (0. < = real number < = 1.)

A real number from 0. to 1. indicating the red color default value for OPEN processing.

K = 24: Green default. (0. < = real number < = 1.)

A real number from 0. to 1. indicating the green color default value for OPEN processing.

K = 25: Blue default. (0. < = real number < = 1.)

A real number from 0. to 1. indicating the blue default color value for OPEN processing.

Color Status

These values indicate the current color settings in effect from the last COLOR call.

K = 26: Intensity requested. (0. < = real number < = 1.)

The intensity setting to be used for subsequent graphic primitives specified for the currently open segment.

K = 27: Red value. (0. < = real number < = 1.)

The red color value setting to be used for subsequent graphic primitives specified for the currently open segment.

K = 28: Green value. (0. < = real number < = 1.)

The green color value setting to be used for subsequent graphic primitives specified for the currently open segment.

K = 29: Blue value. (0. < = real number < = 1.)

The blue color value setting to be used for subsequent graphic primitives specified for the currently open segment.

K = 30: Unused.

TEXT DATA INFORMATION

Text Data Request Word

K = 31: Text information request control word.

A trigger word that when accessed causes requested text font information to be stored in the area indicated by the return location parameter for the enquiry request. The type of text font information desired is indicated by specific data values in the user-specified area as follows:

Word1 = 1 : Enquire on text face data.

Word2 = Face entry index: a value ranging from 0 (default text face) to the value of status entry 32 (the number of text faces supported by the device.

Returns in words 3 and 4 the following:

Word3 · the name code designation (e.g., BODONI, TIMES ROMAN) for this text face (See Appendix E).

Word4 · the number of fonts supported for this text face.

(The values in words 1 and 2 are left unchanged.)

Word1 = 2 : Enquire on text font data.

Word2 = Face entry index (see above).

Word3 = ignored.

Word4 = Font index value: a value ranging from 0 (the default text font for this text face) to the value returned in the Enquire on Face Data request.

Returns in words 5 through 26:

Word5 · font quality code.

Word6 · text character height in window units (real).

Word7 · text character width in window units (real).

Word8 · default inter-character vertical spacing for this text font in window units (real).

Word9 · default inter-character horizontal spacing for this text font in window units (real).

Word10 - character base line offset in window units from the base of the character envelope (real).

Word11 through Word26 -
16 consecutive 16-bit words that when catenated represent a 256-bit string indicating the availability of individual characters for the specified face and font.

Text Face Count Word

K = 32: Number of text faces. (integer)

The number of text faces supported by this device. This is the maximum value that may be specified for the text face index in the previous text information request mechanism.

Default Text Face/Font Parameters

The current OPEN default values for text attributes as specified during system initialization or via the DEFAULT-TEXT command.

K = 33: Face name default. (integer)

The text face name default value for OPEN processing.

K = 34: Text quality default. (integer)

The text quality default value for OPEN processing.

K = 35: Text height default. (real)

The character height default value for OPEN processing in window units.

K = 36: Text width default. (real)

The character width default value for OPEN processing in window units.

K = 37: Text vertical spacing default. (real)

The inter-character vertical spacing default value for OPEN processing in window units.

K = 38: Text horizontal spacing default. (real)

The inter-character horizontal spacing default value for OPEN processing in window units.

K = 39: Unused.

User-requested Text Face/Font Parameters

The most recently requested text font values as specified via the TEXTFACE command or as a consequence of OPEN default processing. These values are used for text clipping and intercharacter positioning. They specify the desired text font values with the height and width values indicating the maximum acceptable for a selected text font. The actual size of the text selected (as indicated in words 52 - 56 of the enquiry information) may differ from these values.

K = 40: Face name. (integer)

The name of the text face requested for the currently open segment via the TEXTFACE command or via OPEN default processing.

K = 41: Text quality value. (integer)

The requested quality, e.g., hardware or software.

K = 42: Text height value. (real)

The requested character height in window units.

K = 43: Text width value. (real)

The requested character width in window units.

K = 44: Text vertical-spacing value. (real)

The requested inter-character vertical spacing in window units.

K = 45: Text horizontal-spacing value. (real)

The requested inter-character horizontal spacing in window units.

K = 46 - 47: Unused.

Currently Selected Text Face/Font Information

The following apply to the text font selected by the system in response to a user- or system-issued TEXTFACE command.

K = 48: Text face index. (integer)

The index value for the currently selected text face. (See status entry 31.)

K = 49: Text face name. (integer)

A code indicating the name of the currently selected text face (e.g., BODONI).

K = 50: Text font index. (integer)

The font index value for the currently selected text font. (See status entry 31.)

K = 51: Text quality.

A word indicating the type of the text that has been selected, e.g., hardware, software.

K = 52: Character height. (real)

The character height in window units for the currently selected text font.

K = 53: Character width. (real)

The character width in window units for the currently selected text font.

K = 54: Character vertical spacing. (real)

The inter-character vertical spacing default value in window units for the currently selected text font.

K = 55: Character horizontal spacing. (real)

The inter-character horizontal spacing default value in window units for the currently selected text font.

K = 56: Character base line offset. (real)

The distance in window units above the bottom of the character envelope of the base line for the character-set.

K = 57: Unused.

ASCII Character Availability

Sixteen 16-bit words that, when catenated, represent a 256-bit string indicating the availability of individual characters for the currently selected text font.

K = 58: Character availability. (integer)

K = 59: Character availability. (integer)

K = 60: Character availability. (integer)

K = 61: Character availability. (integer)

K = 62: Character availability. (integer)

K = 63: Character availability. (integer)

K = 64: Character availability. (integer)

K = 65: Character availability. (integer)

K = 66: Character availability. (integer)

K = 67: Character availability. (integer)

K = 68: Character availability. (integer)

K = 69: Character availability. (integer)

K = 70: Character availability. (integer)

K = 71: Character availability. (integer)

K = 72: Character availability. (integer)

K = 73: Character availability. (integer)

K = 74 - 78: Unused.

FACILITIES ENABLED INFORMATION WORD

K = 79: Facilities enabled. (integer)

A 16-bit word indicating enabled facilities. A 1-bit in the i-th bit from the right indicates the i-th facility is enabled.

CURRENT SCOPE DATA

K = 80: Scopes used. (integer)

A word indicating the scopes currently selected. The low-order 16 bits correspond to the individual scopes with the low-order bit indicating scope # 1 is selected and successive bits indicating scopes 2 through 16. A "1" value for the corresponding bit position indicates that the scope is selected.

CURRENT WINDOW VALUES

The following four values (in user coordinates) define the current clipping window. They are set via the WINDOW call or defaulted by the system initialization process.

K = 81: Window XL. (real)

The coordinate of the left-hand edge (X-left) of the clipping window.

K = 82: Window YB. (real)

The coordinate of the bottom edge (Y-bottom) of the clipping window.

K = 83: Window XR. (real)

The coordinate of the right-hand edge (X-right) of the clipping window.

K = 84: Window YT. (real)

The coordinate of the top edge (Y-top) of the clipping window.

CURRENT VIEWPORT VALUES

The following four values identify the current viewport to which the clipping window will be mapped. They are specified via the VIEWPORT call or are defaulted via the system initialization process. (See VIEWPORT command.)

K = 85: Port XL. (real)

The left-hand edge (X-left) of the selected viewport in viewport units.

K = 86: Port YB. (real)

The bottom edge (Y-bottom) of the selected viewport in viewport units.

K = 87: Port XR. (real)

The right-hand edge (X-right) of the selected viewport in viewport units.

K = 88: Port YT. (real)

The top edge (Y-top) of the selected viewport in viewport units.

CURRENT BEAM VALUES

K = 89: X beam value. (real)

The X coordinate of the current beam location in window units.

K = 90: Y beam value. (real)

The Y coordinate of the current beam location in window units.

K = 91 - 97: Unused.

MISCELLANEOUS

K = 98: Hold mode setting. (integer)

0 indicates hold mode not in effect, 1 indicates hold mode active.

K = 99: Sense Distance. (real)

The sense distance in centimeters as specified via the SENSE call.

[8] GRAPHIC INPUT

The system has 4 main input facilities.

(1) Positioning
(2) Function keys
(3) Segment Touching
(4) Analog device

Each facility can be either ENABLEd or DISABLEd (i.e., not-ENABLEd). The initialization default disables all the input facilities.

All user actions at the display terminal regarding a disabled facility are ignored. All user actions regarding any enabled facility are queued until requested by the application program. In addition, the application program can check for availability of any input.

ENABLE(F)

Enable input facility "F", where:

F = 1 means pointing.

F = 2 means function key.

F = 3 means segment touching.

F = 4 means analog device.

"F" is always specified as an integer. Other nonzero values of "F" may be added, but the "standard" display device is not guaranteed to support them. Values of "F" above 16 may be used for private devices, specific to a system. ENABLE(F) takes no action if the facility "F" is already enabled.

For ENABLE(3), the user must also tell the system which segments should be checked and within which sensitivity aperture (the maximum distance between the cursor and any portion of a display segment, for detection of "touching"). For more details see SENSE and TOUCH. A segment is checked for being touched only if it is both (i) touch sensitive (by the TOUCH call) and (ii) POSTed.

Possible errors: ERR-08 The facility "F" is not available (for 0<F<4).

DISABLE(F)

Disable the input facility "F" as defined above. DISABLE(0) disables all input facilities. The system initialization default is DISABLE(0). When a facility is disabled, all queued input from that facility is cleared. DISABLE(F) takes no action if the facility "F" is not enabled.

Possible errors: ERR-08 The facility "F" is undefined (for 0<F<4).

SENSE (D)

Define the sensitivity distance for the device used for the touching facility. D specifies this distance in centimeters (as a real number). The system default value is SENSE(.3), or about 1/8 of an inch. Implementations are allowed to deviate from the value of D up to 50 percent (to allow simple distance computation, without the need to evaluate square roots, etc.). For some particular input devices (e.g., hardware lightpens), this call may have no effect.

Possible errors: ERR-07 D<0.
ERR-08 No pointing device is available.

INPUT (W, F, K, X, Y)

Get input from the device. When this call is issued, a nonzero value of "W" specifies that the system should wait for the next available input. A zero value for "W" specifies that the system should not wait, and if no input is available, this should be indicated (by the system to the calling program) by setting "F" to zero.

Upon return, "F" is set by the system either to a nonzero value identifying the input facility, or to zero if it was called with W = 0 and no input was queued.

FOR POSITION INPUT (F = 1), X and Y (real numbers), are set to the appropriate values in the user's (WINDOW) coordinate system. K is undefined.

FOR FUNCTION KEY INPUT (F = 2), K (integer) is set by the system to the integer corresponding to the key. X and Y are undefined.

FOR SEGMENT TOUCHING INPUT (F = 3 or F = -3), K is set by the system to the ID of the segment that was touched, and X and Y (real numbers) are set to the position at which the touch was detected. Devices that cannot supply the X and Y of the touch do not modify the values of X and Y. The ENQUIRE call is used to determine if the device is capable of supplying the X and Y of the touch. Only the first touch of any segment is reported. If several segments are touched, each is reported, in an arbitrary order. In order to retrieve all of them, multiple INPUT calls must be issued. When multiple touching occurs, all touches are reported with F = -3, except the last one, which is reported with F = 3.

FOR ANALOG INPUT (F = 4), X is set to the normalized value of the device between 0. and 1., Y and K are undefined.

When a facility is disabled, all its queued inputs are cleared.

Possible errors: ERR-10 No facility enabled or only touch facility enabled but no segments touch enabled.

[9] DEFAULT MODIFICATION

The following calls permit the application program to modify the standard OPEN default settings for color and text face. By setting the default values to his most commonly used values, the application programmer can reduce the amount of work the system must perform as well as reduce the number of GL calls his program must issue.

DEFAULT-COLOR (I, R, G, B)

Reset the default color values for open processing from the current default values to the specified values. The value range for individual parameters is the same as for the COLOR call.

Possible errors: None.

DEFAULT-TEXT (MASK, NAME, QUAL, HEIGHT, WIDTH, VERTICAL-SPACING, HORIZONTAL-SPACING)

Set the default text face values for open processing to the indicated values. This routine changes the open default values for the text face from the system default values to the specified values. The value range for individual parameters is the same as for the TEXTFACE call. The individual fields are identical to those defined for the TEXTFACE command. Fields for which the corresponding mask bit is not set assume the default values for the most closely matching text font.

Possible errors: None.

[10] SCOPE SELECTION

The following GL calls allow the application program to specify the scope(s) on which subsequently defined segments will appear and to control the activation/deactivation of those scopes. Individual scopes may support different color sets and/or highlighting. Depending upon the device type, individual scopes may be physically separate from display devices or may be overlayed bitmaps, permitting capabilities such as transparent overlays and overlay precedence.

DEFAULT-SCOPE (L)

Display subsequently OPENed segments on the identified scopes. L is a scope mask; the i-th bit from the right identifies the i-th scope. The system initialization default is to enable all available scopes.

Possible errors: ERR-07 Invalid scope mask L.

ACTIVE-SCOPE (L)

Activate/deactivate the identified scopes. L is a scope mask; the i-th bit from the right identifies the i-th scope. A 1 activates the scope; 0 deactivates the scope. The system initialization default is to activate all available scopes.

Possible errors: ERR-07 Invalid scope mask L.

[11] MISCELLANEOUS

The following GL calls provide additional capabilities to the application programmer that may be useful in some circumstances, but are generally not required for the development of graphic application programs.

CURSOR (X, Y)

Position the cursor, if possible, at the specified point. If a cursor is displayed (i.e., for touching or positioning) this call allows the user to move the cursor to the point specified by (X,Y), given the in user's (WINDOW) coordinates (real numbers).

If the specified point is outside of the window, the cursor is positioned at the window edge. If the above devices are not enabled, or if the hardware does not support cursor positioning, no action is taken.

Possible errors: None.

ESCAPE (CODE, ARGUMENT-BIT-STRING, RESULT-BIT-STRING)

Bypass standard graphics system processing to send raw data directly to the specified escape function. This GL call provides a handle by which installation-specific options may be introduced into the graphics system.

CODE (an integer) is the name of the function to be executed. ARGUMENT-BIT-STRING is the location of the bits to be transmitted to the specified escape function. RESULT-BIT-STRING is the location at which any response from the specified ESCAPE function is to be returned. The number of bits required for any given ESCAPE function and the interpretation of the bit stream data is dependent upon the particular function involved. This call may or may not be implemented in any given system, and the function CODEs are device/system dependent.

Possible errors:	ERR-05	The CODE is unknown to the display system.
		Any other error (e.g., system table overflow).

ERROR (ERROR, SUBCODE, CRITICALITY)

Set the enquiry array error information according to the indicated values. (See ENQUIRE discussion.) ERROR and CRITICALITY are CRed into their respective fields in the enquiry array. SUBCODE replaces the equivalent field in the enquiry array only if CRITICALITY is greater than the criticality for all errors that have occurred since the last time error information was reported to the user.

Possible errors: None.

SYNCHRONIZE

Interpret all previously issued Graphic Language commands and complete all pending Graphics System tasks.

Possible errors: None.

APPENDIX A

FORTRAN LANGUAGE INTERFACE

The following describes the programmer's interface for FORTRAN users of Graphics Language.

DEVICE CONNECTION INITIATION AND TERMINATION

DINIT(CONFIGURATION-STRING, ASPECT-RATIO)
CONFIGURATION-STRING is a string, ASPECT-RATIO a real number. See Appendix F for possible values for CONFIGURATION-STRING.

DREL
Release the display device being used.

VIEWING AREA AND COORDINATE SYSTEM SELECTION

DPORT (XL, YB, XR, YT)
Define the area of the CRT to be used by the system. All the arguments are real numbers.

DWINDO (XL, YB, XR, YT)
Define the user coordinate system. All the arguments are real numbers.

SEGMENT SPECIFICATION

DOPEN (N)
Initiate specification of a segment N. N, is an integer between 1 and 32000.

DCLOSE
Terminate specification of the currently open segment.

DCOLOR (I, R, G, B)
Set the intensity and chromaticity (hue and saturation) for the remainder of this segment. I, R, G, and B are real numbers.

DINT (I)

Set the intensity level for the remainder of this segment. I is specified as a real number.

DTFACE (M, F, Q, H, W, VS ,HS)

Set text face/font for the remainder of this segment. H, W, VS and HS are specified in window units (real numbers) and indicate the desired height, width, vertical-spacing and horizontal-spacing values for the character font selected. Q, an integer, specifies the quality.

DDOT (X, Y)

Display a dot at the specified position (in the user's coordinate system). X and Y are given as real numbers.

DDRAW (X, Y)

Draw a line from the current beam position to the specified position. X and Y are given as real numbers.

DMOVE (X, Y)

Move the beam position to the specified point. X and Y are given as real numbers.

DLINE (X, Y, I)

DMOVE or DDRAW. If I = 0, then this call is equivalent to DMOVE(X,Y), otherwise it is equivalent to DDRAW(X,Y). As in DMOVE(X,Y) and DDRAW(X,Y), X and Y are supplied as real numbers in absolute coordinates. I is given as an integer.

DARC (X, Y, R, SA, EA)

Draw a circular arc with radius R around the point (X,Y) starting at the angle SA, counter-clockwise ending at angle EA. All arguments are real numbers.

DRDOT (DX, DY)

Display a dot in the position specified relative to the current beam position. DX and DY are given as real numbers.

DRDRAW (DX, DY)

Draw a line from the current beam position to the point (X + DX,Y + DY) where (X,Y) is the beam position before the call. DX and DY are given as real numbers.

DRMOVE (DX, DY)

Move the beam position from the current beam position (X,Y) to (X + DX,Y + DY). DX and DY are given as real numbers.

DRLINE (DX, DY, I)

DRMOVE or DRDRAW. If I = 0 then this call is equivalent to DRMOVE(DX,DY), otherwise it is equivalent to DRDRAW(DX,DY). As in DRMOVE(DX,DY) and DRDRAW(DX,DY), DX and DY are given as real numbers in relative coordinates. I is given as an integer.

DRARC (R, SA, EA)

Draw a circular arc with radius R around the current beam position starting at the angle SA, counter-clockwise ending at angle EA. All arguments are real numbers.

DTEXT (N, STRING-POINTER)

Show a text string starting at the current beam position. N is an integer value indicating the number of characters in the text string pointed to by a string-pointer. The string-pointer may be established either via an array reference to an array containing the text string or via specification of the text string as a literal in the DTEXT call.

DFILLS (X, Y, R, SA, EA, MODE, ANGLE, DIST)

Fill the specified sector of a circle. MODE is an integer and the remaining arguments are real numbers.

DFILLP

Begin polygon definition.

DVERTX (X, Y)

Define polygon vertex. X and Y are real values.

DFILLX (MODE, ANGLE, DIST)

Terminate polygon specification and fill according to indicated parameters. MODE is an integer indicating the type of filling desired. ANGLE and DIST are real numbers indicating the hatch angle and inter-hatch-mark spacing.

SEGMENT CONTROL

DMERGE (M, N)

Merge segment M into segment N. M and N, must be integer values between 1 and 32000.

DPOST (N)

Display segment N on the CRT. N, is an integer between 0 and 32000.

DNPOST (N)

Stop displaying segment N on the CRT. N, is an integer between 0 and 32000.

DHLGHT (N, C)

Set HIGHLIGHT mode for segment N. N, is an integer between 0 and 32000. C is specified as an integer.

DTOUCH (N, K)

Activate/deactivate the existing segment N for "touching" by the device F = 3. N, is an integer between 0 and 32000. K is an integer.

DKILL (N)

Remove segment N from the display system. N, is an integer between 0 and 32000.

UPDATE CONTROL

DHOLD (K)

Hold/don't hold subsequent display erasures. K is an integer value.

DDONE

Update screen to reflect current segment status.

GRAPHIC FILES

DFILEI (FILENAME-POINTER, U, H)

Read the specified graphics file. FILENAME-POINTER points to a text string containing the filename. U and H are the segment IDs for the UNHIGHLIGHTED and HIGHLIGHTED portions of the file. Both are integers between 1 and 32000.

DFOPEN (FILENAME-POINTER, N, COMMENT-POINTER)

Open graphics file for output. FILENAME-POINTER and COMMENT-POINTER point to text strings containing the filename and the file comment. N, an integer, specifies the number of characters in the file comment.

DFWRTE (N)

Write segment N in the currently opened output file. The segment name N is an integer between 0 and 32000.

DFCLSE

Close the output file.

DFATTR (FILENAME-POINTER, X, Y, MAX-CHARS, NO-CHARS, COMMENT-POINTER)

Obtain the precision and comment information from a graphics file. FILENAME-POINTER points to a text string containing the file name. COMMENT-POINTER points to a text area in which the comment is returned. MAX-CHARS is an integer specifying the maximum size set by the system to the number of comment characters returned. Integers X and Y are file precision.

INFORMATION ENQUIRY

DENQ (K, VALUE)

Enquire about the status of the system and current values of parameters. K is an integer value specifying the desired status item. VALUE is in the format as specified for the indicated entry.

GRAPHIC INPUT

DENABL (F)

Enable input facility "F". F is an integer.

DDSABL (F)

Disable the input facility "F". F is an integer.

DSENSE (D)

Define the sensitivity distance for the device used for the touching facility. D specifies this distance in centimeters (as a real number).

DINPUT (W, F, K, X, Y)

Get input from the device. F, W, and K are integers; X and Y are real values. For keyboard terminals such as the Tektronix 4012, 4014, and 4027, or the HP-2648A, keys 1-9 are function keys and return the corresponding integer; key 0 is used for pointing and touching input.

DEFAULT MODIFICATION

DDFCLR (I, R, G, B)

Set color default values for DOPEN processing. The parameters are as described under DCOLOR.

DDFTFA (M, F, Q, H, W, VS, HS)

Set text face/font DOPEN default values. The parameters are as described under DTFACE.

SCOPE SELECTION

DDFSCP (N)

Display the following on scope N. N is an integer. This command is currently used in multi-plane bitmap terminals such as the AED512 for identifying groups of memory planes.

DASCOP (N)

Activate/deactivate the named scopes. N is an integer. This command is currently used in multi-plane bitmap terminals such as the AED512 for controlling groups of memory planes.

MISCELLANEOUS

DCURSR (X, Y)

Move the cursor to the specified point. X and Y are given as real numbers.

DERROR (ERROR, SUBCODE, CRITICALITY)

Set the enquiry array error bits. ERROR is an integer value from 1 to 65535, SUBCODE an integer from 0 to 65535, and CRITICALITY an integer from 0 to 15.

DSYNCH

Interpret all previously issued Graphics Language commands.

APPENDIX B

BLISS LANGUAGE INTERFACE

The following describes the programmer's interface for BLISS users of Graphics Language.

DEVICE CONNECTION INITIATION AND TERMINATION

BINIT (CONFIGURATION-STRING, ASPECT-RATIO)
CONFIGURATION-STRING is a string, ASPECT-RATIO a real number. See Appendix F for possible values for CONFIGURATION-STRING.

BREL ( )
Release the display device being used.

VIEWING AREA AND COORDINATE SYSTEM SELECTION

BPORT (XL, YB, XR, YT)
Define the area of the CRT to be used by the system. All the arguments are real numbers.

BWINDO (XL, YB, XR, YT)
Define the user coordinate system. All the arguments are real numbers.

SEGMENT SPECIFICATION

BOPEN (N)
Initiate specification of a segment with ID N. N, is an integer between 1 and 32000.

BCLOSE ( )
Terminate specification of the currently open segment.

BCOLOR (I, R, G, B)
Set the intensity and chromaticity (hue and saturation) for the remainder of this segment. I, R, G, and B are real numbers.

BINT (I)

Set the intensity level for the remainder of this segment. I is specified as a real number.

BTFACE (M, F, Q, H, W, VS ,HS)

Set text face/font for the remainder of this segment. H, W, VS and HS are specified in window units (real numbers) and indicate the desired height, width, vertical-spacing and horizontal-spacing values for the character font selected. Q, an integer, specifies the quality.

BDOT (X, Y)

Display a dot at the specified position (in the user's coordinate system). X and Y are given as real numbers.

BDRAW (X, Y)

Draw a line from the current beam position to the specified position. X and Y are given as real numbers.

BMOVE (X, Y)

Move the beam position to the specified point, without any drawing. X and Y are given as real numbers.

BLINE (X, Y, I)

BMOVE or BDRAW. If I = 0, then this call is equivalent to BMOVE(X,Y), otherwise it is equivalent to BDRAW(X,Y). As in BMOVE(X,Y) and BDRAW(X,Y), X and Y are supplied as real numbers in absolute coordinates. I is given as an integer.

BARC (X, Y, R, SA, EA)

Draw a circular arc with radius R around the point (X,Y) starting at the angle SA, counter-clockwise ending at angle EA. All arguments are real numbers.

BRDOT (DX, DY)

Display a dot in the position specified relative to the current beam position. DX and DY are given as real numbers.

BRDRAW (DX, DY)

Draw a line from the current beam position to the point (X + DX,Y + DY) where (X,Y) is the beam position before the call. DX and DY are given as real numbers.

BRMOVE (DX, DY)

Move the beam position from the current beam position (X,Y) to (X + DX,Y + DY) without any drawing. DX and DY are given as real numbers.

BRLINE (DX, DY, I)

BRMOVE or BRDRAW. If I = 0 then this call is equivalent to BRMOVE(DX,DY), otherwise it is equivalent to BRDRAW(DX,DY). As in BRMOVE(DX,DY) and BRDRAW(DX,DY), DX and DY are given as real numbers in relative coordinates. I is given as an integer.

BRARC (R, SA, EA)

Draw a circular arc with radius R around the current beam position starting at the angle SA, counter-clockwise ending at angle EA. All arguments are real numbers.

BTEXT (N, STRING-POINTER)

Show a text string starting at the current beam position. N is an integer value indicating the number of characters in the text string pointed to by a string-pointer.

BFILLS (X, Y, R, SA, EA, MODE, ANGLE, DIST)

Fill the specified sector of a circle. MODE is an integer and the remaining arguments are real numbers.

BFILLP ( )

Begin polygon definition.

BVERTX (X, Y)

Define polygon vertex. X and Y are real values.

BFILLX (MODE, ANGLE, DIST)

Terminate polygon specification and fill according to indicated parameters. MODE is an integer indicating the type of filling desired. ANGLE and DIST are real numbers indicating the hatch angle and inter-hatch-mark spacing.

SEGMENT CONTROL

BMERGE (M, N)

Merge segment M into segment N. M and N, must be integer values between 1 and 32000.

BPOST (N)

Display segment N on the CRT. N, is an integer between 0 and 32000.

BNPOST (N)
Stop displaying segment N on the CRT. N, is an integer between 0 and 32000.

BHLGHT (N, C)
Set HIGHLIGHT mode for segment N. N is an integer between 0 and 32000. C is specified as an integer.

BTOUCH (N, K)
Activate/deactivate the existing segment N for "touching" by the device F = 3. N is an integer between 0 and 32000. K is an integer.

BKILL (N)
Remove segment N from the display system. N, is an integer between 0 and 32000.

UPDATE CONTROL

BHOLD (K)
Hold/don't hold subsequent display erasures. K is an integer value.

BDONE ( )
Update screen to reflect current segment status.

GRAPHICS FILES

BFILEI (FILENAME-POINTER, U, H)
Read the specified graphics file. FILENAME-POINTER points to a text string containing the filename. U and H are the segment IDs for the UNHIGHLIGHTED and HIGHLIGHTED portions of the file. Both are integers between 1 and 32000.

BFOPEN (FILENAME-POINTER, N, COMMENT-POINTER)
Open graphics file for output. FILENAME-POINTER and COMMENT-POINTER point to text strings containing the filename and the file comment. N, an integer, specifies the number of characters in the file comment.

BFWRTE (N)
Write segment N in the currently opened output file. The segment name N is an integer

between 0 and 32000.

BFCLSE ( )
Close the output file.

BFATTR (FILENAME-POINTER, X, Y, MAX-CHARS, NO-CHARS, COMMENT-POINTER)
Obtain the precision and comment information from a graphics file. FILENAME-POINTER points to a text string containing the file name. COMMENT-POINTER points to a text area in which the comment is returned. MAX-CHARS is an integer specifying the maximum size set by the system to the number of comment characters returned. Integers X and Y are file precision.

INFORMATION ENQUIRY

BENQ (K, VALUE)
Enquire about the status of the system and current values of parameters. K is an integer value specifying the desired status item; VALUE is in the format as specified for the indicated entry.

GRAPHIC INPUT

BENABL (F)
Enable input facility "F". F is an integer.

BDSABL (F)
Disable the input facility "F". F is an integer.

BSENSE (D)
Define the sensitivity distance for the device used for the touching facility. D specifies this distance in centimeters (as a real number).

BINPUT (W, F, K, X, Y)
Get input from the device. F, W, and K are integers; X and Y are real values. For keyboard terminals such as the Tektronix 4012, 4014, and 4027, or the HP-2648A, keys 1-9 are function keys and return the corresponding integer; key 0 is used for pointing and touching input.

DEFAULT MODIFICATION

BDFCLR (I, R, G, B)

Set color default values for BOPEN processing. The parameters are as described under BCOLOR.

BDFTFA (M, F, Q, H, W, VS, HS)

Set text face/font BOPEN default values. The parameters are as described under BTFACE.

SCOPE SELECTION

BDFSCP (N)

Display the following on scope N. N is an integer. This command is currently used in multi-plane bitmap terminals such as the AED512 for identifying groups of memory planes.

BASCOP (N)

Activate/deactivate the named scopes. N is an integer. This command is currently used in multi-plane bitmap terminals such as the AED512 for controlling groups of memory planes.

MISCELLANEOUS

BCURSR (X, Y)

Move the cursor to the specified point. X and Y are given as real numbers.

BERROR (ERROR, SUBCODE, CRITICALITY)

Set the status block error bits. ERROR is an integer value from 1 to 65535, SUBCODE an integer from 0 to 65535, and CRITICALITY an integer from 0 to 15.

BSYNCH ( )

Interpret all previously issued Graphics Language commands and complete all pending graphics system tasks.

APPENDIX C

"C" LANGUAGE INTERFACE

The following describes the programmer's interface for C users of Graphics Language.

DEVICE CONNECTION INITIATION AND TERMINATION

cinit (configuration-string, aspect-ratio)

The first parameter is a pointer to a character string, the second is a real number. "configuration-string" is discussed in Appendix F.

crel ()

Release the display device being used.

VIEWING AREA AND COORDINATE SYSTEM SELECTION

cport (xl, yb, xr, yt)

Define the area of the CRT to be used by the system. All the arguments are of type float.

cwindo (xl, yb, xr, yt)

Define the user coordinate system. All the arguments are of type float.

SEGMENT SPECIFICATION

copen (n)

Initiate specification of a segment with ID n. n, is an integer between 1 and 32000 of type unsigned short int.

cclose ()

Terminate specification of the currently open segment.

ccolor (i, r, g, b)

Set the intensity and chromaticity (hue and saturation) for the remainder of this segment. i, r, g, and b are are of type float.

cint (i)

Set the intensity level for the remainder of this segment. i is of type float.

ctface (m, f, q, h, w, vs, hs)

Set text face/font for the remainder of this segment. m, with type of unsigned char, is the mask setting which indicates which of the following parameters are to be included in the font selection process. f, with type of unsigned char, indicated the desired text face. q, with type of unsigned char, specifies the quality. h, w, vs and hs are specified in window units as type float and indicate the desired height, width, vertical-spacing and horizontal-spacing values for the character font selected.

cdot (x, y)

Display a dot at the specified position (in the user's coordinate system). x and y are of type float.

cdraw (x, y)

Draw a line from the current beam position to the specified position. x and y are of type float.

cmove (x, y)

Move the beam position to the specified point. x and y are of type float.

cline (x, y, i)

cmove or cdraw. If i = 0, then this call is equivalent to cmove(x,y), otherwise it is equivalent to cdraw(x,y). As in cmove(x,y) and cdraw(x,y), x and y are specified in absolute coordinates and are of type float. i is of type unsigned short int.

carc (x, y, r, sa, ea)

Draw a circular arc with radius r around the point (x,y) starting at the angle sa, counter-clockwise ending at angle ea. All arguments are of type float

crdot (dx, dy)

Display a dot in the position specified relative to the current beam position. dx and dy are of type float.

crdraw (dx, dy)

Draw a line from the current beam position to the point (x + dx,y + dy) where (x,y) is the beam position before the call. dx and dy are of type float.

crmove (dx, dy)

Move the beam position from the current beam position (x,y) to (x + dx,y + dy). dx and dy are of

type float.

crline (dx, dy, i)
crmove or crdraw. If i = 0 then this call is equivalent to crmove(dx,dy), otherwise it is equivalent to crdraw(dx,dy). As in crmove(dx,dy) and crdraw(dx,dy), dx and dy are in relative coordinates and of type float.

crarc (r, sa, ea)
Draw a circular arc with radius r around the current beam position starting at the angle sa, proceeding counter-clockwise and ending at angle ea. All arguments are of type float.

ctext (n, string)
Show a text string starting at the current beam position. n is of type short int and indicates the number of characters in the text string. string is a pointer to a character string.

cfills (x, y, r, sa, ea, mode, angle, dist)
Fill the specified sector of a circle. mode is of type short int and the remaining arguments are of type float.

cfillp ()
Begin polygon definition.

cvertx (x, y)
Define polygon vertex. X and Y are real values.

cfillx (mode, angle, dist)
Terminate polygon specification and fill according to indicated parameters. mode indicates the type of filling desired and is of type short int. angle and dist define the hatch angle and inter-hatch-mark spacing respectively and are of type float.

SEGMENT CONTROL

cmerge (m, n)
Merge segment m into segment n. M and N are of type unsigned short int and must be values between 1 and 32000.

cpost (n)
Display segment n on the CRT. n is of type unsigned short int and must be between 0 and 32000.

cnpost (n)

Stop displaying segment n on the CRT. n is of type unsigned short int and must be between 0 and 32000.

chlght (n, c)

Set HIGHLIGHT mode for segment n. n and c are of type unsigned short int. n must be between 0 and 32000. c must be either 0 or 1.

ctouch (n, k)

Activate/deactivate the existing segment n for "touching" by the device F = 3. n and k are of type unsigned short int.

ckill (n)

Remove segment n from the display system. n must be between 0 and 32000 and is of type unsigned short int.

UPDATE CONTROL

chold (k)

Hold/don't hold subsequent display erasures. k is of type unsigned short int and must be either 0 or 1.

cdone ()

Update screen to reflect current segment status.

GRAPHICS FILES

cfilei (filename, u, h)

Read the specified graphics file. filename is a pointer to a string of characters containing the name of the desired GL file. u and h are the segment IDs for the UNHIGHLIGHTED and HIGHLIGHTED portions of the file. Both are of type unsigned short int and must be between 1 and 32000.

cfopen (filename, n, comment)

Open graphics file for output. filename and comment are pointers to character strings containing the filename and the file comment. n is of type unsigned short int and specifies the number of characters in the comment string.

cfwrte (n)

Write segment n in the currently opened output file. The segment name n is of type unsigned short int and must be between 0 and 32000.

cfclse ()

Close the output file.

cfattr (filename, x, y, max-chars, no-chars, comment)

Obtain the precision and comment information from a graphics file. filename is a pointer to a character string containing the name of the desired file. comment is a pointer to a character string containing the comment field associated with the specified file. max-chars is of type short int and limits the number of comment field characters that should be returned via this GL call. no-chars is a pointer to a short integer in which the actual number of comment characters returned is made available. x and y are pointers to unsigned short integers in which the file precision values are returned via this GL call.

INFORMATION ENQUIRY

cenq (k, structure)

Enquire about the status of the system and current values of parameters. k is an unsigned short integer specifying the desired structure element to be retrieved; structure is a pointer to a user enqary structure defined as follows:

```
struct enqary                    ENQUIRY STRUCTURE DEFINITIONS
 {
  long int uerrcrit;             error criticality
  long int uerr;                 error code
  long int uerrseg;              segment in error
  long int uhlghavl;             highlighting availability
  long int ukbrdavl;             keyboard availability
  long int upsitavl;             positioning availability
  long int ufnctavl;             function key availability
  long int utchavl;              touching availability
  long int uanalavl;             analog device availability
  long int ucursr;               cursor availability
  long int uxresol;              x resolution
  long int uyresol;              y resolution
  float uxdimsn;                 x dimension
  float uydimsn;                 y dimension
  float uxport;                  x port max
  float uyport;                  y port max
  long int uscpsmsk;             number of scopes
  long int udevcde;              device code
```

```
long int udevtype;          device type
long int;                   unused
long int;                   unused
struct enqclr *ucolctl;     color info request trigger/pntr
float uintdef;              intensity default
float ureddef;              red default
float ugrndef;              green default
float ubludef;              blue default
float uintval;              intensity requested
float uredval;              red value
float ugrnval;              green value
float ublueval;             blue value
struct enqfnt *utxtctl;     text info request trigger/pntr
long int ufacecnt;          number of text faces
long int ufcdef;            face default
long int uqldef;            text quality default
float uchdef;               box height default
float ucwdef;               box width default
float uvsdef;               box vertical spacing dflt
float uhsdef;               box horizontal spacing dflt
long int;                   unused
long int ufcval;            face value
long int uqlval;            text quality value
float uchval;               box height value
float ucwval;               box width value
float uvsval;               box vertical spacing value
float uhsval;               box horizontazl spacing value
long int;                   unused
long int;                   unused
long int utxtfndx;          text face index
long int utxtface;          text face code
long int utxtsndx;          text size index
long int utxtqlty;          text quality
float uchrhght;             character height
float uchrwdth;             character width
float uchrvspc;             character vertical spacing
float uchrhspc;             character horizontal spacing
float ucbsl;                base line offset
long int;                   unused
long int ucword1;           character availability
long int ucword2;           character availability
long int ucword3;           character availability
long int ucword4;           character availability
long int ucword5;           character availability
long int ucword6;           character availability
long int ucword7;           character availability
long int ucword8;           character availability
long int ucword9;           character availability
long int ucword10;          character availability
long int ucword11;          character availability
long int ucword12;          character availability
long int ucword13;          character availability
long int ucword14;          character availability
long int ucword15;          character availability
long int ucword16;          character availability
```

```
    long int unumltyp;          number of line types
    float ultypdef;             line type default
    float ultype;               current line type
    float ultypctl;             line type trigger trigger
    float uescctl;              unused
    long int ufenabld;          facilities enabled
    long int uscope;            scopes used
    float uwindxl;              window xl
    float uwindyb;              window yb
    float uwindxr;              window xr
    float uwindyt;              window yt
    float uportxl;              port xl
    float uportyb;              port yb
    float uportxr;              port xr
    float uportyt;              port yt
    float uxbeam;               x beam value
    float uybeam;               y beam value
    long int;                   unused
    long int;                   unused
    long int;                   unused
    long int;                   unused
    long int;                   unused
    long int;                   unused
    long int;                   unused
    long int udhold;            hold mode setting
    float udsense;              sense distance
   };
```

The requested entry is filled in by the GL system for subsequent access by the user. Two of the entries are pointers to other structures. If access to either text font information (via font id) or color table information (via color entry index) is desired, the appropriate pointer must be established prior to the enquiry request being initiated. The definition of the two structures follows:

```
Pointed at by ucolctl:

   struct enqclr                COLOR REQUEST STRUCTURE
    {
     long int uclrctl;          color enquiry control
     long int uclrscp;          color enquiry scope number
     long int uclrndx;          color enquiry number
     float uclrint;             color enquiry intensity value
     float uclrred;             color enquiry red value
     float uclrgrn;             color enquiry green value
     float uclrblu;             color enquiry blue value
     long int uclrhlgt;         color enquiry highlight value
    };


Pointed at by utxtctl:

   struct enqfnt                FONT REQUEST STRUCTURE
    {
     long int ufntctl;          font enquiry control
     long int ufntfce;          face entry index
```

```
    long int ufntnme;          face name designation
    long int ufntfnt;          font entry index
    long int ufntqlty;         font quality code
    float ufntht;              font height in window units
    float ufntwd;              font width in window units
    float ufntvs;              font vertical spacing
    float ufnths;              font horizontal spacing
    float ufntbsl;             font base line offset
    long int ufntwd1;          font character support word
    long int ufntwd2;          font character support word
    long int ufntwd3;          font character support word
    long int ufntwd4;          font character support word
    long int ufntwd5;          font character support word
    long int ufntwd6;          font character support word
    long int ufntwd7;          font character support word
    long int ufntwd8;          font character support word
    long int ufntwd9;          font character support word
    long int ufntwd10;         font character support word
    long int ufntwd11;         font character support word
    long int ufntwd12;         font character support word
    long int ufntwd13;         font character support word
    long int ufntwd14;         font character support word
    long int ufntwd15;         font character support word
    long int ufntwd16;         font character support word
};
```

The above three structure definitions are contained in the following file which may be included in the user's program:

~level 2/cenqary.h

GRAPHIC INPUT

cenabl (f)

Enable input facility "f". f is of type unsigned short int.

cdsabl (f)

Disable input facility "f". f is of type unsigned short int.

csense (d)

Define the sensitivity distance for the device used for the touching facility. d is of type float and specifies the desired distance in centimeters.

cinput (w, f, k, x, y)

Get input from the device. w is of type unsigned short int and indicates whether the system is to wait until input is available or not before returning control to the user program. f is a pointer to

a short integer and indicates the facility desired. k is a pointer to an unsigned short integer in which the system returns specific input data. x and y are pointers to objects of type float in which the system returns input coordinate data. For keyboard terminals such as the Tektronix 4012, 4014, and 4027, or the HP-2648A, keys 1-9 are function keys and return the corresponding integer; key 0 is used for pointing and touching input.

DEFAULT MODIFICATION

cdfclr (i, r, g, b)
Set color default values for copen processing. The parameters are as described under ccolor.

cdftfa (m, f, q, h, w, vs, hs)
Set text face/font copen default values. The parameters are as described under ctface.

SCOPE SELECTION

cdfscp (n)
Display the following on scope n. n is of type unsigned short int. This command is currently used in multi-plane bitmap terminals such as the AED512 for identifying groups of memory planes.

cascop (n)
Activate/deactivate the named scopes. n is an integer. This command is currently used in multi-plane bitmap terminals such as the AED512 for controlling groups of memory planes.

MISCELLANEOUS

ccursr (x, y)
Move the cursor to the specified point. x and y are of type float.

cerror (error, subcode, criticality)
Set the enquiry structure error fields. error is an integer value from 1 to 65535, subcode an integer from 0 to 65535, and criticality an integer from 0 to 15. All three are of type unsigned short int.

csynch ()

Interpret all previously issued Graphics Language commands.

APPENDIX D

INTERLISP LANGUAGE INTERFACE

The following describes the INTERLISP interface to the Graphics System. Use of the INTERLISP interface requires two files in the LEVEL2 graphics directory, L2LISP.COM and RECVR.SAV, as well as pseudo-teletype capability for internal inter-process communication between elements of the graphics system.

GENERAL CALLS

(DINIT <-devdesignation-> <-aspect-ratio->)

DINIT initializes the graphics system. <-devdesignation-> specifies the type of backend and display device desired. (See Appendix F for more information.) <-aspect-ratio-> specifies the desired aspect ratio for the allocated display surface area. An aspect ratio of -1.0 uses the default aspect-ratio for the connected display divice. It is recommended that a COND be used to test the results of the DINIT operation. DINIT returns NIL if it fails to initialize the Graphics System successfully.

example 1:
```
(SETQ STR "BACKEND = (TEKTRONIX),DEVICE = (D,TTY:)")
(DINIT STR 1.0)    (* initialize c2g and use TTY:
                      as the device and it understands
                      TEKTRONIX display codes)
```

example 2:
```
(COND ((DINIT STR 1.0) 'WIN)
(T (ERROR "DINIT has failed")))
```

Note: DINIT is interlocked so DREL must be called before another another DINIT can be issued.

(DREL)

DREL terminates the graphics system and releases the connected display device.

VIEWING AREA AND COORDINATE SYSTEM SELECTION

(DPORT XL YB XR YT)

Defines the portion of the allocated display area (see DINIT) to be used for subsequent picture generation. All the arguments are real numbers.

(DWINDOW XL YB XR YT)

Define the user coordinate system to be mapped to the current viewport. All the arguments are real numbers.

SEGMENT SPECIFICATION

(DOPEN N)

Open segment with ID N. N, is an integer between 1 and 32000.

(DCLOSE)

Close the currently open segment and send it to the display system.

(DCOLOR I R G B)

Set the intensity and chromaticity (hue and saturation) for the remainder of this segment. I, R, G, and B are real numbers.

(DINT C)

Set the intensity level for the remainder of this segment. C is specified by a real number.

(DTFACE MASK NAME QUAL HEI WID VERTSP HORSP)

Selects a type face/font. The first 3 arguments are integers and the rest are real numbers.

(DDOT X Y)

Display a dot at the specified position (in the user's coordinate system). X and Y are given as real numbers.

(DDRAW X Y)

Draw a line from the current beam position to the specified position. X and Y are real numbers.

(DMOVE X Y)

Reposition the beam at the specified point without any drawing. X and Y are real numbers.

(DLINE X Y I)

If I = 0 then this call is equivalent to DMOVE(X,Y), otherwise it is equivalent to (DDRAW X Y). As in (DMOVE X Y) and (DDRAW X Y), X and Y are real numbers, and absolute coordinates are used. I is an integer.

(DARC XC YC R SA EA)

Draw a circular arc with radius R around the point (XC,YC). Starting at the angle SA, then moving counter-clockwise and ending at angle EA. All arguments are real numbers.

(DRDOT DX DY)

Display a dot at the position specified relative to the current beam position. DX and DY are given as real numbers.

(DRDRAW DX DY)

Draw a line from the current beam position to the point (X + DX,Y + DY) where (X,Y) is the beam position before the call. DX and DY are real numbers.

(DRMOVE DX DY)

Reposition the beam at the location (X + DX,Y + DY) without any drawing where (X, Y) is the current beam location. DX and DY are real numbers.

(DRLINE DX DY I)

If I = 0, this call is equivalent to (DRMOVE X Y), otherwise it is equivalent to (DRDRAW X Y). As in (DRMOVE DX DY) and (DRDRAW DX DY), DX and DY are real numbers, relative to the current beam location. I is an integer.

(DRARC R SA EA)

Draw a circular arc with radius R around the current beam position starting at the angle SA, progressing counter-clockwise and ending at angle EA. All arguments are real numbers.

(DTEXT <-string->)

Creates a text string starting at the current beam position.

example : (PROG () (DMOVE .5 .5) (DTEXT "Hi there"))

(DFILLS X Y R SA EA MODE ANGLE DIST)

Fill the sector of a circle specified via the parameter list. The first five arguments are the same as DARC, the rest are mode, angle, and distance and specify the filling criteria.

(DFILLP)

Begin polygon definition for filling. No arguments are necessary.

(DVERTEX X Y)

Add a vertex to the polygon specification initiated via the preceding DFILLP call.

(DFILLX MODE ANGLE DIST)

Terminate the current polygon definition and specify the filling parameters for that polygon. Mode is an integer value that defines the type of filling to be performed. Angle is a real number that defines the hatching angle to be used. Dist is a real number that defines the inter-hatching spacing.

SEGMENT CONTROL

(DMERGE M N)
Merges two segments together. Segment M is merged into segment N.

(DPOST N)
Display segment N on the CRT. N, is an integer between 1 and 32000.

(DNPOST N)
Stop displaying segment N on the CRT. N, is an integer between 1 and 32000.

(DHLGHT N C)
Set HIGHLIGHT mode for segment N. N, is an integer between 1 and 32000. C is specified as an integer.

(DTOUCH N K)
Enable/disable the existing Segment, N, for touching. N is an integer between 1 and 32000. K is an integer.

(DKILL N)
Remove segment N from the display system. N, is an integer between 1 and 32000.

UPDATE CONTROL

(DHOLD K)
Hold/don't hold subsequent display erasures. K is an integer value.

(DDONE)
Update screen to reflect current segment status.

GRAPHIC FILES

(DFILE! FILENAME UNHILITE HILITE)
Read the graphics file specified by the string FILENAME. UNHILITE and HILITE are segment numbers to be used for the non-hilighted and hilighted segments respectively.

(DFOPEN FILENAME COMMENT)
Open a graphics file for output. Both arguments are strings. FILENAME is the name of the file

to be created and COMMENT is an arbitrary comment of the user's choice.

(DFWRTE SEGMENT)
Write a segment to the currently open graphics file.

(DFCLSE)
Close the currently open graphics file and make it available to any GL application program.

(DFATTR FILENAME MAX.CHARS)
Return the attributes of the identified graphics file. FILENAME is a string identifying the desired graphics file and MAX.CHARS is an integer defining the maximum number of characters to be returned in the comment string. The results of this function call is three fixed numbers and a comment string.

INFORMATION ENQUIRY

(DENQ KEY COMMAND SCP.TFC V4 V5)
DENQ retrieves status information from the graphics system. KEY (the only argument required for the DENQ call) identifies the status element to be retrieved. The remaining arguments, COMMAND, SCP.TFC, V4 and V5, are only needed when the KEY is 21 (decimal), or 31 (decimal). *(See the ENQUIRE description in this manual for more details.) DENQ, as with any* well-behaved side-effectless LISP function, always returns a value whose type is dependent on the values of its formal arguments.

GRAPHIC INPUT

(DENABLE F)
Enable input facility "F". F is an integer.

(DDISABLE F)
Disable the input facility "F". F is an integer.

(DSENSE R)
Define the radius of sensitivity of the device used for the touching facility. R specifies the radius in centimeters (as a real number).

(DINPUT W)
Get input from the device where W indicates that the system should either wait for next input event (if non are already pending) or return immediately. DINPUT returns a four element list, i.e. (F K X Y). F and K are integers; X and Y are real values. For the HP2648 and the Tektronix, keys

1-9 are function keys and return the corresponding integer; key 0 is used for pointing and touching input.

DEFAULT MODIFICATION

(DDFCLR I R G B)
Set the default value of color for DOPEN processing. (See DCOLOR.)

(DDFTFA MASK NAME QUAL HEI WID VERTSP HORSP)
Select the default typeface for DOPEN processing. The arguments are the same as DTFACE.

SCOPE SELECTION

(DDFSCP N)
Use scope N for the following display. N is fixed. This command is currently used in the Genisco and AED512 to select different memory planes.

MISCELLANEOUS

(DCURSOR X Y)
Not implemented.

(DERROR ERROR SUBCODE CRITICAL)
DERROR is used by GL subroutines to set the error information in the GL status block for retrieval by invoking GL routines. (See ENQUIRE discussion). All arguments are of type fix.

(DSYNCH)
Interpret all previously issued Graphics Language commands.

APPENDIX E

The following table provides a correspondence between text face types supported by the graphics system and the face-name-codes used to designate them (see Enquary entries 31, 33, 40, 49).

TEXT FACES

FACE NAME	FACE NAME CODE
ASCII	1
CYRILLIC	2
MATH SET	3
ICON SET	4
MICRO GAMMA OUTLINE	5
ASCII CAPITALS	6

APPENDIX F

CONNECTION CONFIGURATION STRING

The following describes the CONFIGURATION STRING options specifiable by the user in the INITIATE command. The CONFIGURATION-STRING consists of one or more of the following keyword parameters separated by commas:

```
BACKEND=(TYPE[,HOST])

DEVICE=(CONNECTION,DISPLAY-ADDRESS[,DISPLAY-SUBADDRESS])

INITFILE=(S-T-CODE[,FILENAME])
```

where [...] indicates an optional parameter. Each keyword parameter is described below.

BACKEND = (TYPE[,HOST)]

This parameter tells the Graphics System the type of display code and the host on which the display code is to be generated. Acceptable values for TYPE include:

```
GENISCO     - for the GENISCO GCT-3000
TEKTRONIX   - for the Tektronix 4010, 4012, and 4014
TK4027      - for the Tektronix 4027
HP2648      - for the Hewlett-Packard HP-2648A
HP9872      - for the Hewlett-Packard HP-9872A
AED512      - for the Advanced Electronic Design 512
```

Normally, the Graphics System generates display-device orders on the same host on which the user's graphic application program runs. The user can cause the Graphics System to generate display device orders on another host (e.g., a Remote Site Module) by specifying the optional HOST parameter. To do this, HOST should be the ARPANET hostname of a host containing a Graphics System Backend Server.

DEVICE = (CONNECTION,DISPLAY-ADDRESS[,DISPLAY-SUBADDRESS)]

This parameter tells the Graphics System where the display device is located and how the device connection is to be made. CONNECTION specifies the protocol to be used to connect to the device. Four protocols are supported:

```
D    - Direct connection for local displays.
S    - Server connection for displays connected to remote hosts.
T    - TCP connection for displays connected to TACS.
```

DISPLAY-ADDRESS is the address of the display device. For local displays on TENEX/TOPS-20, this is a device designator, typically TTY: or TTYddd: where ddd is the terminal number. For local

displays on UNIX, this is a device designator, typically /dev/tty or /dev/ttyddd where ddd is the terminal number. For ARPANET "wild" TAC connected display devices, the device designator is:

NET:host-ts

"host" is the name of the TAC to which the terminal is attached, and "ts" is the octal number, (portnumber)*2↑8 + 27_8. For example, port 15 (OCTAL) on USC-TAC would be NET:USC-TAC-6427. (see the ARPANET "TAC Users Guide" or Appendix I for additional information). For ARPANET server connected display devices, the device designator is:

NET:host-fs

"host" is the ARPANET host name to which the terminal is attached, and "fs" is the server's listening socket number. (As an example, a display device on ACCAT-UNIX that had a server listening on socket 77 would be NET:ACCAT-UNIX-77.) For TCP connected display devices the display address is:

NET:host-fp

where "host" is either the name or internet number for the remote host and "fp" is the TCP foreign port number, in octal, of the connected display.

DISPLAY-SUBADDRESS is an integer that may be used to distinguish between multiple display devices at the same physical address, e.g., multiple display devices served by a server or multiple work stations on a single display device.

INITFILE = (S-T-CODE[,FILENAME)]

This parameter allows the user to include CONFIGURATION-STRING parameters from a file. The contents of the specified file replace the INITFILE = (...) in the CONFIGURATION-STRING parameter. The file may contain any CONFIGURATION-STRING parameter(s) except another INITFILE = (...) parameter. S-T-CODE can be one of the following:

```
T     - Obtain the filename for the included
        file from the user's primary TTY:
F     - Use the FILENAME parameter for the name
        of the included file.
```

FILENAME is present only for S-T-CODE = F and contains the name of the included file.

Examples:

The following are examples of INIT calls. In all examples, "-1." is used as the desired aspect-ratio value to indicate that the device default aspect-ratio is to be used.

To use a Tektronix 4014 attached either directly to a host or to a TAC both as the login device and as the graphics terminal, code:

TOPS-20: CALL DINIT ('BACKEND = (TEKTRONIX), DEVICE = (D,TTY:)', -1.)

UNIX: cinit ("BACKEND = (TEKTRONIX), DEVICE = (D, /dev/tty)", -1.)

To use a Tektronix 4027 terminal attached to terminal port 107, code:

TOPS-20: CALL DINIT ('BACKEND = (TK4027), DEVICE = (D,TTY107:)', -1.)

UNIX: cinit ("BACKEND = (TK4027), DEVICE = (D,/dev/tty107)", -1.)

To use a HP-2648A terminal attached to the NPS-TAC port 5 with that port set in "wild" mode, code:

TOPS-20: CALL DINIT ('BACKEND = (HP2648), DEVICE = (T,NET:1.NPS-TAC-2427)', -1.)

UNIX: cinit ("BACKEND = (HP2648), DEVICE = (D,NET:NPS-TAC-2427)", -1.)

To use GENISCO display 2 at ACCAT-UNIX with display code generated at the application program host, code:

TOPS-20: CALL DINIT ('BACKEND = (GENISCO), DEVICE = (S,NET:ACCAT-UNIX-77,2)', -1.)

UNIX: cinit ("BACKEND = (GENISCO), DEVICE = (S,NET:ACCAT-UNIX-77,2)", -1.)

If any of the above CONFIGURATION-STRINGs was kept in the file "foo", then code:

TOPS-20: CALL DINIT ('INITFILE = (F,foo)', -1.)

UNIX: cinit ("INITFILE = (F,foo)", -1.)

or code:

TOPS-20: CALL DINIT ('INITFILE = (T)', -1.)

UNIX: cinit ("INITFILE = (T)", -1.)

and enter foo on the primary TTY: when asked for the INITFILE filename.

Connection String Configuration Subroutine

A connection string configuration subroutine is provided to simplify the task of constructing the connection string. It is invoked as follows:

CONFIGURE(CONFIGURATION-STRING)

This routine builds the appropriate INITIATE configuration information via interactive user dialog. The configuration information is returned as a text string in CONFIGURATION-STRING.

For TOPS-20, a BLISS callable configure subroutine can be found in <LEVEL2>L2BCNF.REL. The entry name is

bcnf(configuration-string)

For UNIX, a C callable configure subroutine can be found in ~level2/l2ccnf.o. The entry name is

ccnf(configuration-string)

APPENDIX G

GL ERROR CODES

The following are the error codes. When ERR-N occurs, 2**(N-1) is ORed into status entry 1.

ERR-01 System error.
ERR-02 System table overflow.
ERR-03 No open segment.
ERR-04 Segment n does not exist.
ERR-05 The system does not support this call.
ERR-06 Device/back-end unknown, or not available.
ERR-07 Arguments to this call are out of the allowed range.
ERR-08 Undefined input facility.
ERR-09 Wrong number of parameters.
ERR-10 No facility enabled.
ERR-11 Communications error.
ERR-12 System not initialized.
ERR-13 File input/output error.
ERR-14 Not assigned yet.
ERR-15 Not assigned yet.
ERR-16 Miscellaneous

APPENDIX H

SYSTEM FILES

The TOPS-20 graphics system is distributed as five files:

L2SYS.REL	- Graphics System
L2FOR.REL	- FORTRAN Language Interface
L2BLI.REL	- BLISS Language Interface
L2FB.REL	- FORTRAN/BLISS Language Interface
L2LISP.COM	- Interlisp Language Interface

These files are contained in <LEVEL2> on ISIB, ISIC, and ISIE. Files can be retrieved using FTP. A graphics application program is formed by linkediting the application program(s) with L2SYS.REL and either L2FOR.REL, L2BLI.REL, or L2FB.REL depending on whether the application was written in FORTRAN-10, BLISS-10, or both.

The VAX graphics system is distributed as:

l2sys.o	- Graphics System with C interface

contained in <level2>.

APPENDIX I

USING A GRAPHICS TERMINAL ATTACHED TO AN ARPANET TAC

The Graphics System provides the capability for operating a graphics terminal from an ARPANET TAC port. A typical graphics application requires two terminals, one alphanumeric terminal for controlling the program and performing nongraphics input/output, and one graphics terminal for graphics input/output. The following procedure is suggested for operating in this mode:

1. Before attempting to use a Graphics Terminal on a TAC, verify that the port to be used is set in "Wild Mode" and "Quiet Mode". These settings must be authorized by the site liaison and executed by the Network Operations Center. Only if "Wild Mode" is enabled will the Graphics System be able to connect to the TAC; if not enabled, the Graphics System will simply wait half a minute during Initialization and then return an error (subcode = 64009).

   "Quiet Mode" suppresses all messages on the TAC port. This means no "TCP Trying...", "Open", or "Closed" messages should appear when attempting to use the Graphics Terminal as a regular terminal.

2. Set the baud rate of the graphics terminal and TAC port (if a baud rate over 300 baud is desired). To change the baud rate of the TAC port, enter the TAC command

   @DEVICE RATE #

   from the graphics terminal, where # is a 10-bit encoding of the input and output baud rates. Example values for # are:

#	Output baud	Input Baud
373	300	300
438	600	600
503	1200	1200
633	2400	2400
629	2400	300
693	4800	300
757	9600	300

   See the TAC User's Guide for details of the encoding for other input/output rates.

3. Determine the TAC port number to which the graphics terminal is attached by entering the TAC command

   @RESET

from the graphics terminal. The octal port number is the last number in the resulting herald.

4. Instruct the TAC not to echo characters by entering

   @ECHO HALFDUPLEX

   from the graphics terminal.

5. Use another terminal to login and run your program using the string "DEVICE = (T,TAC-NAME-xxx)" in the INIT command. "xxx" should be the TCP port number, in octal, of the graphics terminal on hte TAC. TCP port numbers may be calculated from the terminal port number in the herald after the RESET command (in the above case, the terminal port number is 3).

Terminal Port Number	TCP Port Number (Octal)
1	427
2	1027
3	1427
4	2027
5	2427
6	3027
.	.
.	.
.	.
p	256 * p + 23 (expressed in octal)

   For example, to run your program on a Tektronix connected to terminal port 4 on CCA-TAC us the initialization string:

   BACKEND = (TEKTRONIX),DEVICE = (T,CCA-TAC-2027)

6. At the completion of the graphics session, reset the TAC port by entering the TAC command

   @ECHO REMOTE

   from the graphics terminal.

7. Restore the graphics terminal and the TAC port to its previous baud rates if appropriate.

APPENDIX J

GRAPHICS LANGUAGE FORTRAN USAGE EXAMPLE

```
C
C
C
C       The following example is intended to help the novice user
C       understand how GL might be used to satisfy a simple graphics
C       output requirement.  The example utilizes the FORTRAN language
C       version of GL to invoke the desired graphics functions.  The
C       main routine builds a three-element bar chart, each element of
C       which is colored and filled in a different fashion.  The
C       routine creates and labels an axis in addition to creating
C       the filled bar chart itself.  Subroutine BOX is invoked with
C       the height and width and fill mode parameters for each of the
C       three elements of the bar chart.  The subroutine creates each
C       box such that it's lower left-most corner coincides with the
C       current beam location.  The remaining comments relate to the
C       correspondingly numbered comments in the code itself.
C
C       Comment 1:
C
C               The subroutine determines the current location of the
C               beam by enquiring on cells 89 and 90 of the enquiry
C               array which contain the current coordinates of the
C               beam in window units (floating point values).  The
C               values in cells 89 and 90 change whenever any GL call
C               is issued that affects the current beam location (e.g.,
C               MOVE, DRAW, TEXT).  Note that the values in cells 89
C               and 90 reflect the results of GL calls issued during
C               segment creation; they have nothing whatsoever to do
C               with segment drawing and, thus, are unaffected by
C               POSTing, UNPOSTing, HIGHLIGHTing, and the like.
C
C       Comment 2:
C
C               The subroutine utilizes relative draw operations to
C               create a box around the area to be filled.  (The
C               main routine positioned the beam at the box origin
C               prior to invoking the BOX subroutine.)
C
C       Comment 3:
C
C               The area to be filled is specified in absolute
C               coordinates using the polygon filling capability.  The
C               vertices of the polygon are specified in counter-
C               clockwise order (clock-wise order is also accepted) and
C               the desired filling parameters indicated.  The vertex
C               specification is bounded by the DFILLP call (indicating
C               start of polygon specification) and the DFILLX call
C               (indicating completion of polygon specification).  The
C               inter-hatching distance, DIST, is specified in window
C               units.
C
```

```
C       Comment 11:
C
C               The main routine first initializes the graphics device
C               connection and indicates the coordinate system in which
C               the user wishes to operate.  A backend and device
C               address is specified by the user at run time.  The
C               aspect ratio of the desired display surface viewing
C               area is one-to-one.  The default window values are
C               overidden to establish a coordinate range more useful
C               to the application.
C
C       Comment 12:
C
C               Specification of a segment with the name "439" is
C               begun.  Absolute MOVEs and DRAWs are used to produce a
C               single quadrant coordinate axis, the Y-component of
C               which receives a tic mark every ten units.
C
C       Comment 13:
C
C               The default text specification is overridden to yield
C               a text specification which selects stroked text and
C               results in a vertically descending label along the
C               Y-axis of the chart.  This text specification is
C               immediately overridden with a new specification which
C               results in additional intercharacter horizontal
C               spacing for the default character font.
C
C       Comment 14:
C
C               Three new segments are created, each of which
C               overrides the segment default color specification.  A
C               single new color is utilized for the contents of each
C               segment, produced by invoking the BOX subroutine.
C               Prior to each invocation of the BOX routine, the beam
C               is positioned at the desired box origin in the user's
C               coordinate system.
C
C       Comment 15:
C
C               One of the segments just created is highlighted.
C
C       Comment 16:
C
C               All of the segments created by the preceding code are
C               made visible in a single operation.  Alternatively,
C               each segment might be individually made visible.  The
C               connection with the device is terminated and the
C               program exits.
C
C
```

```
      SUBROUTINE BOX (YHGHT, XWIDTH, IFILL, ANGLE, DIST)
C1
C1 get the current x,y beam position
C1
      CALL DENQ (89,X)
      CALL DENQ (90,Y)
C2
C2 draw the box at the current beam position using relative draws
C2
      CALL DRDRAW (XWIDTH,0.)
      CALL DRDRAW (0.,YHGHT)
      CALL DRDRAW (-XWIDTH,0.)
      CALL DRDRAW (0.,-YHGHT)
C3
C3 specify the polygon vertices for filling
C3
      CALL DFILLP
      CALL DVERTX (X+XWIDTH,Y)
      CALL DVERTX (X+XWIDTH,Y+YHGHT)
      CALL DVERTX (X,Y+YHGHT)
      CALL DVERTX (X,Y)
      CALL DFILLX (IFILL,ANGLE,DIST)
      RETURN
      END
C11
C11 initialize the graphics system and set window
C11
      CALL DINIT ( 'INITFILE = (T)', 1.)
      CALL DWINDO (-20.,-20.,110.,110.)
C12
C12 draw axis with tic marks in white in segment 439
C12
      CALL DOPEN (439)
      CALL DMOVE (0.,100.)
      CALL DDRAW (0.,0.)
      CALL DDRAW (100.,0.)
      DO 8100 I=10,100,10
      TIC = I
      CALL DMOVE (-2.,TIC)
      CALL DRDRAW (2.,0.)
8100  CONTINUE
C13
C13 select text and label axis and chart
C13
      CALL DTFACE (63,1,1,5.,3.,-10.,-3.)
      CALL DMOVE (-18.,70.)
      CALL DTEXT (4,'LOAD')
      CALL DTFACE (32,0,0,0.,0.,0.,1.)
      CALL DMOVE (-5.,-10.)
      CALL DTEXT (34,'System Load for 3 Consecutive Days')
      CALL DCLOSE
C14
C14 draw three boxes in segments 1629-1631
C14
      CALL DOPEN (1629)
```

```
        CALL DCOLOR (1.,1.,.5,0.)
        CALL DMOVE (0.,0.)
        CALL BOX (45.,30.,2,3.1416/4.,2.)
        CALL DCLOSE
        CALL DOPEN (1630)
        CALL DCOLOR (1.,.5,0.,.7)
        CALL DMOVE (30.,0.)
        CALL BOX (30.,30.,1,3.1416/5.,2.)
        CALL DCLOSE
        CALL DOPEN (1631)
        CALL DCOLOR (1.,1.,0.,.8)
        CALL DMOVE (60.,0.)
        CALL BOX (88.,30.,3,0.,0.)
        CALL DCLOSE
C15
C15 highlight one of the boxes
C15
        CALL DHLGHT (1630,1)
C16
C16 make everything visible and terminate
C16
        CALL DPOST (0)
        CALL DREL
        END
```

Index