The graPHIGS Programming Interface: Technical Reference

SC33-8193-11

The graPHIGS Programming Interface: Technical Reference

SC33-8193-11

Note
Before using this information and the product it supports, read the information in Appendix F, “Notices,” on page 407.

Twelfth Edition (October 2000)

This edition applies to the GDDM/graPHIGS Programming Interface, Version 2, Release 2.5, program number 5688-093, AIXwindows Environment/6000 (1.3) AIXwindows/3D feature, Program Number 5601-257, and to all subsequent releases of this product until otherwise indicated in new editions.

A reader’s comment form is provided at the back of this publication. If the form has been removed, address comments to Information Development, Department 04XA-905-6C006, 11501 Burnet Road, Austin, Texas 78758-3493. To send comments electronically, use this commercial Internet address: aix6kpub@austin.ibm.com. Any information that you supply may be used without incurring any obligation to you.

Contents

About This Book

This book provides technical information about the functions and limitations of the graPHIGS API and its supported workstations. It also contains reference information, both general and specific, about particular aspects of writing applications, namely on Character Set Facilities and on Defaults and Nicknames. The purpose of this book is to provide a comprehensive volume of technical information needed to accurately code or modify applications using the graPHIGS API.

Who Should Use This Book

This book is intended for application programmers.

Highlighting

The following highlighting conventions are used in this book:

Bold	Identifies commands, subroutines, keywords, files, structures, directories, and other items whose names are predefined by the system. Also identifies graphical objects such as buttons, labels, and icons that the user selects.
Italics	Identifies parameters whose actual names or values are to be supplied by the user.
`Monospace`	Identifies examples of specific data values, examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, messages from the system, or information you should actually type.

ISO 9000

ISO 9000 registered quality systems were used in the development and manufacturing of this product.

Related Publications

The following books contain information on graPHIGS API products:

- *The graPHIGS Programming Interface: ISO PHIGS Subroutine Reference*
- *The graPHIGS Programming Interface: Understanding Concepts*

Part 1. Workstations

Chapter 1. General Information for Workstations

This chapter contains two areas pertaining to all workstations supported by the graPHIGS API. The first section, Accessing a Workstation, describes the workstation types and how connection identifiers are used by the graPHIGS API when opening a workstation. The second section, Description Tables in the graPHIGS API, provides lists of the initial default values in the graPHIGS API Traversal State List and view table entries in the Workstation View Table Data at initialization.

Accessing a Workstation

To display graphical data from an application program, the graPHIGS API requires information about the workstation. The graPHIGS API must know the capabilities and characteristics of the workstation as well as the information to access the workstation using the operating system. This information is identified to the graPHIGS API with the workstation type and connection identifier parameters of the Open Workstation (**GPOPWS**) and Create Workstation (**GPCRWS**) subroutines.

The final workstation type and connection identifier used by the graPHIGS API are the result of processing the nicknames in the External Defaults File (EDF) or the Application Defaults Interface Block (ADIB) and may differ from the parameters specified on **GPOPWS** and **GPCRWS**. See Controlling the Environment with Defaults and Nicknames.

Workstation Types

Workstations of a certain class that share specific characteristics are referred to as a workstation type. For example, 5080 workstations are of the same type, although each may or may not have certain optional features (such as color capability). However, 5080 workstations are of a different workstation type than NATIVE workstations because of different capabilities.

The workstation category of the user's physical workstation is identified by the workstation type parameter on the Open Workstation (**GPOPWS**) or Create Workstation (**GPCRWS**) subroutine call.

The graPHIGS API supports the following workstation types:

Table 1. Workstation Category, Type, and Environment

Category	Workstation Type Parameter	Environment
6090	6090	VM/CMS MVS
5080	5080	VM/CMS MVS
GDDM Devices	GDDM	VM/CMS MVS
GDF Files	GDF	VM/CMS MVS AIX RS/6000
CGM	CGM	VM/CMS MVS AIX RS/6000
X	X	AIX RS/6000
X	XSOFT	AIX RS/6000
X	XDWA	AIX RS/6000
IMAGE files	IMAGE	AIX RS/6000

When you specify the workstation type parameter on the **GPOPWS** or **GPCRWS** subroutine calls, the workstation type is passed as an 8-byte character string, left-justified with blanks padding on the right of the type (example, `'NATIVE '`).

Refer to Workstation Description Tables for information about the supported workstation types.

For information about using nickname processing to pass workstation types to the graPHIGS API, see Controlling the Environment with Defaults and Nicknames.

The data values of the Workstation Description Table (WDT) describe the capabilities and characteristics of a workstation type. The data values of the WDT are available to your application through the WDT group of inquiries. Refer to WDT Inquiries for additional information. The workstation type parameter of the WDT inquiries identifies the category of workstation whose data is requested. (The WDT group of inquiries return only data that does not change during the processing of your application. The WSL group of inquiries return the current state of workstation data values that your application can change.)

When either the Open Workstation (**GPOPWS**) or Create Workstation (**GPCRWS**) subroutine calls create a workstation resource, the graPHIGS API creates a new unique workstation type. This created (″realized″) workstation type precisely describes the actual capabilities of the created workstation. For example, 5080 workstations may, in general, support the dial input devices. However, a particular 5080 may not have the dials present. The actual WDT will not have an entry for the valuator devices, where the general (″generic″) WDT will have such an entry. The actual workstation type is available to your application using the Inquire Realized Workstation Category and Type (**GPQRCT**) subroutine. You can then use the returned realized workstation type as the wstype parameter on the WDT group of inquiries.

Connection Identifiers

The connection identifier specified on Open Workstation (**GPOPWS**) and Create Workstation (**GPCRWS**) subroutines is used by the graPHIGS API to access a specific device for processing. The identifier is passed to the operating system, typically for allocation of the workstation.

For the 6090, the connection identifier is an ″*″ (asterisk).

For the 5080 in the S/390 environment, the connection identifier must be the DDNAME (MVS) or FILEDEF name (VM) which defines the device you wish to access. On VM, for example, issue the following command prior to running your application:

```
FILEDEF IBM5080 GRAF 120
```

You would specify `IBM5080` on **GPOPWS** or **GPCRWS** as the connection identifier parameter and the graPHIGS API would attempt to open a 5080 at virtual address 120.

For GDDM supported devices, the connection identifier is used as the ″name-list″ by the graPHIGS API when it issues the GDDM call DSOPEN. See *GDDM Base: Programming Reference* for details of the DSOPEN call and the name-list parameter. For example, specifying a connection identifier of * will indicate to GDDM (by passing a name-list of ″*″) that the user console is to be used as the workstation.

GDDM nickname processing may be performed on the data passed on the DSOPEN call (see *GDDM Base: Programming Reference*). The DSOPEN call issued by the graPHIGS API takes the form:

```
DSOPEN (DEVID, FAM, DEVTOK, 0, 0, NAMEC, NAMEL)
        |      |     |              |      |
        |      |     |              |      name list, the first entry
        |      |     |              |          of which is set equal to
        |      |     |              |          the connection identifier
        |      |     |             name count and is always set to 1
        |      |     GDDM device token (always '*')
        |      GDDM family (always 1)
       [default] encoded device identifier
```

The connection identifier defines the workstation you wish to access.

- The connection identifier for a NATIVE workstation is the device name for the workstation you are using. The device names are:

`'hft/`*n*`'`	Lower case; causes graPHIGS API to open /dev/hft/*n*

'*'	Causes graPHIGS API to open /dev/hft (which will open dev/hft/*n* where *n* is the next available number)

- The connection identifier for an X workstation specifies which X server the graPHIGS API workstation window will be associated with.

'*'	the graPHIGS API will connect to the server specified in the operating system environment DISPLAY variable.
'*host*:*server*screen	This is a standard X server specification, where *host* is the host name where an X server is running, *server* is the server ID, and *screen* is the screen number.

For information about using nickname processing to pass connection identifiers to the graPHIGS API, see Controlling the Environment with Defaults and Nicknames.

For further information about the connection identifier passed when generating GDF files, CGM files, or IMAGE files, see The GDF Workstation, The CGM Workstation, or The IMAGE Workstation.

Description Tables in the graPHIGS API

This section contains two tables:

- The graPHIGS API Traversal State List
- Workstation View Table Data

The data types of the returned values are identified by the following codes:

Data Type	Definition
I Integer	A whole number
R Real	A floating-point number
S String	A character string
E Enumeration	A data type comprised of a set of values. The set is defined by enumerating the identifiers denoting the values.
n **Quantity**	This specifies an undesignated quantity of data.

Note: The notation of *n* (number) [default] *t* (data type) indicates a collection of data of that type. This can be indicated in one of two ways:

- By using notation such as 3[default]R (three real numbers), which could specify something like the *x*, *y*, and *z* coordinates of a three-dimensional point or RGB values
- By using a variable number such as *n*[default]I, which specifies a collection of *n* integers.

The values identified with the symbol * reflect the default value of a workstation configuration variable; that is, this may not be the actual workstation value after the workstation is opened.

The graPHIGS API Traversal State List

Table 2. The graPHIGS API Traversal Defaults Table

Description	Data Type	Default Value
Annotation height scale factor	R	1.0
Annotation path (1=RIGHT, 2=LEFT, 3=UP, 4=DOWN)	E	1=RIGHT

Table 2. The graPHIGS API Traversal Defaults Table (continued)

Description	Data Type	Default Value
Annotation Text Alignment		
Horizontal (1=NORMAL, 2=LEFT_ALIGN, 3=CENTER, 4=RIGHT_ALIGN)	E	1=NORMAL
Vertical (1=NORMAL, 2=TOP, 3=CAP, 4=BASE, 5=BOTTOM)	E	1=NORMAL
Annotation up vector	2[default]R	0.0, 1.0
Antialiasing identifier (1=NONE, 2=PERFORM)	I	1=NONE
Back data modification matrix	3[default]3[default]R	1.0, 0.0, 0.0 0.0, 1.0, 0.0 0.0, 0.0, 1.0
Back interior color index	3[default]R	See note*
Back interior color source type	E	VERTEX
Back specular color	3[default]R	See note*
Back Surface Properties		
Ambient reflection coefficient	R	1.0
Diffuse reflection coefficient	R	1.0
Specular reflection coefficient	R	1.0
Specular reflection exponential	R	0.0
Transparency coefficient	R	0.0
Character base vector	2[default]R	1.0, 0.0
Character expansion factor	R	1.0
Character height	R	0.01
Character line scale factor	R	1.0
Character positioning mode (1=CONSTANT, 2=PROPORTIONAL)	I	1=CONSTANT
Color processing mode index	I	0
Character spacing	R	0.0
Character up vector	2[default]R	0.0, 1.0
Class set names, number of	I	0
Curve Approximation Criteria		
Criteria: (1=WS_DEPENDENT, 2=CONSTANT_SUBDIVISION_BETWEEN_KNOTS, 3=VARIABLE_SUBDIVISION_BETWEEN_KNOTS)	I	1=WS_DEPENDENT
Control value	R	1.0

Table 2. The graPHIGS API Traversal Defaults Table (continued)

Description	Data Type	Default Value
Data Filtering for Data Mapping		
Magnification data filtering method (1=SAMPLE_IN_BASE, 2=INTERP_IN_BASE)	E	1=SAMPLE_IN_BASE
Minification data filtering method (1=SAMPLE_IN_BASE, 2=INTERP_IN_BASE, 3=SAMPLE_IN_SQUARE_MM, 4=SAMPLE_IN_AND_INTERP_BTWN_SQUARE_MM 5=INTERP_IN_SQUARE_MM, 6=INTERP_IN_AND_BTWN_SQUARE_MM, 7=SAMPLE_IN_RECT_MM, 8=SAMPLE_IN_AND_INTERP_BTWN_RECT_MM, 9=INTERP_IN_RECT_MM)	E	1=SAMPLE_IN_BASE
u-dimension bounding method (1=CLAMP, 2=REPEAT)	E	1=CLAMP
v-dimension bounding method (1=CLAMP, 2=REPEAT)	E	1=CLAMP
Data mapping method (-1=IMAGE_ARRAY, 1=DM_METHOD_COLOR, 2=SINGLE_VALUE_UNIFORM, 4=BI_VALUE_UNIFORM)	E	1=DM_METHOD_COLOR
Data mapping table index	I	0
Data modification matrix	3[default]3[default]R	1.0, 0.0, 0.0 0.0, 1.0, 0.0 0.0, 0.0, 1.0
Data Morphing		
Data morphing scale factors	n[default]R	{1.0}
Data morphing scale factors, number of	I	1
Default attribute source flag items:		
Polyline line type (1=BUNDLED, 2=INDIVIDUAL)	E	2=INDIVIDUAL
Polyline line width scale factor (1=BUNDLED, 2=INDIVIDUAL)	E	2=INDIVIDUAL
Polyline color (1=BUNDLED, 2=INDIVIDUAL)	E	2=INDIVIDUAL
Marker type (1=BUNDLED, 2=INDIVIDUAL)	E	2=INDIVIDUAL
Marker size scale factor (1=BUNDLED, 2=INDIVIDUAL)	E	2=INDIVIDUAL

Table 2. The graPHIGS API Traversal Defaults Table (continued)

Description	Data Type	Default Value
Polymarker color (1=BUNDLED, 2=INDIVIDUAL)	E	2=INDIVIDUAL
Text font (1=BUNDLED, 2=INDIVIDUAL)	E	2=INDIVIDUAL
Text precision (1=BUNDLED, 2=INDIVIDUAL)	E	2=INDIVIDUAL
Text color (1=BUNDLED, 2=INDIVIDUAL)	E	2=INDIVIDUAL
Character expansion factor (1=BUNDLED, 2=INDIVIDUAL)	E	2=INDIVIDUAL
Character spacing (1=BUNDLED, 2=INDIVIDUAL)	E	2=INDIVIDUAL
Interior style (1=BUNDLED, 2=INDIVIDUAL)	E	2=INDIVIDUAL
Interior style index (1=BUNDLED, 2=INDIVIDUAL)	E	2=INDIVIDUAL
Interior color (1=BUNDLED, 2=INDIVIDUAL)	E	2=INDIVIDUAL
Edge flag (1=BUNDLED, 2=INDIVIDUAL)	E	2=INDIVIDUAL
Edge line type (1=BUNDLED, 2=INDIVIDUAL)	E	2=INDIVIDUAL
Edge line width scale factor (1=BUNDLED, 2=INDIVIDUAL)	E	2=INDIVIDUAL
Edge color (1=BUNDLED, 2=INDIVIDUAL)	E	2=INDIVIDUAL
Depth cue index	I	0
Destination blending function (1=DSTBF_ZERO, 2=DSTBF_ONE, 3=DSTBF_SRC_ALPHA, 4=DSTBF_ONE_MINUS_SRC_ALPHA, 5=DSTBF_DST_ALPHA, 6=DSTBF_ONE_MINUS_DST_ALPHA, 7=DSTBF_DST_COLOR, 8=DSTBF_ONE_MINUS_SRC_COLOR)	E	4=DSTBF_ONE_MINUS_SRC_ALPHA
Edge bundle table index	I	1

Table 2. The graPHIGS API Traversal Defaults Table (continued)

Description	Data Type	Default Value
Edge color	3[default]R	See note*
Edge flag (1=OFF, 2=ON, 3=GEOMETRY_ONLY)	E	1=OFF
Edge line type	I	1=SOLID
Edge line width scale factor	R	1.0
Face distinguish mode (1=NONE, 2=COLOR_SURFACE_PROPERTIES)	I	1=NONE
Face lighting method (1=FACE_INDEPENDENT, 2=FACE_DEPENDENT)	E	1=FACE_INDEPENDENT
Frame buffer comparison:		
Operation (1=NO_OPERATION, 2=WRITE_WHEN_EQUAL, 3=WRITE_WHEN_NOT_EQUAL)	E	1=NO_OPERATION
Mask	I	N/A
Comparison value	I	N/A
Frame buffer write protect mask	I	0
Geometric Text Alignment		
Horizontal (1=NORMAL, 2=LEFT_ALIGN, 3=CENTER, 4=RIGHT_ALIGN)	E	1=NORMAL
Vertical (1=NORMAL, 2=TOP, 3=CAP, 4=BASE, 5=BOTTOM)	E	1=NORMAL
Geometric text path (1=RIGHT, 2=LEFT, 3=UP, 4=DOWN)	E	1=RIGHT
Global modeling transformations	4[default]4[default]R	1.0, 0.0, 0.0, 0.0 0.0, 1.0, 0.0, 0.0 0.0, 0.0, 1.0, 0.0 0.0, 0.0, 0.0, 1.0
Highlighting color	3[default]R	See note*

Table 2. The graPHIGS API Traversal Defaults Table (continued)

Description	Data Type	Default Value
HLHSR (hidden line, hidden surface removal identifier) (1=VISUALIZE_IF_NOT_HIDDEN, 2=VISUALIZE_IF_HIDDEN, 3=VISUALIZE_ALWAYS, 4=NOT_VISUALIZE, 5=FACE_DEPENDENT_VISUALIZATION, 6=NO_UPDATE, 7=GREATER_THAN, 8=EQUAL_TO, 9=LESS_THAN, 10=NOT_EQUAL, 11=LESS_THAN_OR_EQUAL_TO)	I	1=VISUALIZE_IF_NOT_HIDDEN
Interior bundle table index	I	1
Interior color	3[default]R	See note*
Interior color source type	E	VERTEX
Interior shading method (1=SHADING_NONE, 2=SHADING_COLOR, 3=SHADING_DATA)	E	2=SHADING_COLOR
Interior style (1=HOLLOW, 2=SOLID, 3=HATCH, 4=PATTERN, 5=EMPTY)	E	1=HOLLOW
Interior style index	I	1
Label identifier	I	0
Light color source type	E	VERTEX
Light Source State		
Number of light sources	I	0
List of active light sources	E	N/A
Lighting calculation mode (1=NONE, 2=PER_AREA, 3=PER_VERTEX)	E	1=NONE
Line type	I	1=SOLID_LINE
Line width scale factor	R	1.0
Local modeling transformations	4[default]4[default]R	1.0, 0.0, 0.0, 0.0 0.0, 1.0, 0.0, 0.0 0.0, 0.0, 1.0, 0.0 0.0, 0.0, 0.0, 1.0
Marker size scale factor	R	1.0
Marker type	I	3=ASTERISK
Modeling clipping indicator (1=NOCLIP, 2=CLIP)	E	1=NOCLIP

Table 2. The graPHIGS API Traversal Defaults Table (continued)

Description	Data Type	Default Value
Parametric surface characteristics (1=NONE, 2=ISOPARAMETRIC_LINES)	E	1=NONE
Pick identifier	I	0
Polygon culling (1=NONE, 2=BACK, 3=FRONT)	I	1=NONE
Polyhedron edge culling mode (1=NONE, 2=BOTH_BACK, 3=BOTH_FRONT, 4=BOTH_BACK_OR_BOTH_FRONT, 5=BACK_AND_FRONT, 6=LEAST_ONE_BACK, 7=LEAST_ONE_FRONT)	I	1=NONE
Polyline bundle table index	I	1
Polyline color	3[default]R	See note*
Polyline end type (1=FLAT, 2=ROUND, 3=SQUARE)	E	1=FLAT
Polyline shading method (1=POLYLINE_SHADING_NONE, 2=POLYLINE_SHADING_COLOR)	E	1=POLYLINE_SHADING_NONE
Polymarker bundle table index	I	1
Polymarker color	3[default]R	See note*
Reflectance model (1=REFLECTANCE_NONE, 2=AMB, 3=AMB_DIFF, 4=AMB_DIFF_SPEC)	E	1=REFLECTANCE_NONE
Source blending function (1=SRCBF_ZERO, 2=SRCBF_ONE, 3=SRCBF_SRC_ALPHA, 4=SRCBF_ONE_MINUS_SRC_ALPHA, 5=SRCBF_DST_ALPHA, 6=SRCBF_ONE_MINUS_DST_ALPHA, 7=SRCBF_DST_COLOR, 8=SRCBF_ONE_MINUS_DST_COLOR, 9=SRCBF_MIN_SRC_ALPHA_ONE_MINUS_DST_ALPHA)	E	3=SRCBF_SRC_ALPHA
Specular color	3[default]R	See note*
Surface Approximation Criteria		
Criteria (1=WS_DEPENDENT, 2=CONSTANT_SUBDIVISION_BETWEEN_KNOTS, 3=VARIABLE_SUBDIVISION_BETWEEN_KNOTS)	I	1=WS_DEPENDENT
Control value	2[default]R	1.0, 1.0
Surface Properties		

Table 2. The graPHIGS API Traversal Defaults Table (continued)

Description	Data Type	Default Value
Ambient reflection coefficient	R	1.0
Diffuse reflection coefficient	R	1.0
Specular reflection coefficient	R	1.0
Specular reflection exponential	R	0.0
Transparency coefficient	R	0.0
Text bundle table index	I	1
Text Color	3[default]R	See note*
Text font	I	1
Text precision (1=STRING_PREC, 2=CHAR_PREC, 3=STROKE_PREC)	E	1=STRING_PREC
Transparency coefficient	R	0.0
Trimming curve approximation criteria:		
Criteria (1=WS_DEPENDENT, 2=CONSTANT_SUBDIVISION_BETWEEN_KNOTS, 3=VARIABLE_SUBDIVISION_BETWEEN_KNOTS)	I	1=WS_DEPENDENT
Control value	3[default]R	1.0, 1.0, 1.0
Vertex Morphing		
Vertex morphing scale factors	n[default]R	{1.0}
Vertex morphing scale factors, number of	I	1
View index	I	0
Z-buffer protect mask	I	0
Note: The default color is the color value contained in entry 1 of the rendering color table.		

Workstation View Table Data

When a workstation is opened with Open Workstation (**GPOPWS**) or Create Workstation (**GPCRWS**), a Workstation State List (WSL) is created. The WSL is initialized from information in the actual Workstation Description Table (WDT) that is modified to reflect the capabilities of the specific workstation to be used. The workstation transformation values and the view table in the WSL are initialized to the following values. The right-hand column lists the appropriate inquiry subroutine and parameter.

Table 3. WSL View Parameters at Initialization

Description	Data Type	Default Value	Inquiry
Workstation window	6[default]R	0.0, 1.0, 0.0, 1.0, 0.0, 1.0	
Workstation viewport	6[default]R	0.0, *WVX*, 0.0, *WVY*, 0.0, *WVZ*	
Note: For a square display surface, *WVX*, *WVY*, and *WVZ* are the maximum device coordinate values in the *x*, *y*, and *z* directions. For a non-square display surface, the largest square portion of the display surface in the lower left corner is used.			
View Table Entry Values			

Table 3. WSL View Parameters at Initialization (continued)

Description	Data Type	Default Value	Inquiry
Viewing transformation matrix	16[default]R	1.0, 0.0, 0.0, 0.0 0.0, 1.0, 0.0, 0.0 0.0, 0.0, 1.0, 0.0 0.0, 0.0, 0.0, 1.0	**GPQCVR** *[Group 18, 19]*
View Mapping Matrix	4[default]4[default]R	1.0, 0.0, 0.0, 0.0 0.0, 1.0, 0.0, 0.0 0.0, 0.0, 1.0, 0.0 0.0, 0.0, 0.0, 1.0	**GPQCVR** *[Group 22, 23]*
Window	4[default]R	-1.0, 1.0, -1.0, 1.0	**GPQCVR** *[Group 16, 17]*
Viewport	6[default]R	0.0, 1.0, 0.0, 1.0, 0.0, 1.0	**GPQCVR** *[Group 14, 15]*
Projection reference point	3[default]R	0.0, 0.0, 1.0	**GPQCVR** *[Group 16, 17]*
View plane distance	R	0.0	**GPQCVR** *[Group 16, 17]*
Near plane distance	R	1.0	**GPQCVR** *[Group 16, 17]*
Far plane distance	R	0.0	**GPQCVR** *[Group 16, 17]*
Projection type (1=PARALLEL, 2=PERSPECTIVE)	E	1=PARALLEL	
Window clipping indicator (1=OFF, 1=ON)	E	2=ON	**GPQCVR** *[Group 1]*
Far clipping indicator (1=OFF, 2=ON)	E	2=ON	**GPQCVR** *[Group 3]*
Near clipping indicator (1=OFF, 2=ON)	E	2=ON	**GPQCVR** *[Group 2]*
Shielding color type (1=INDEXED, 2=DIRECT)	E	1=INDEXED	**GPQCVR** *[Group 3]*
Shielding color	I or 3[default]R	0	**GPQCVR** *[Group 5]*
Shielding indicator (1=OFF, 2=ON)	E	1=OFF	**GPQCVR** *[Group 4]*
View border color type (1=INDEXED, 2=DIRECT)	E	1=INDEXED	**GPQCVR** *[Group 7]*
View border color	I or 3[default]R	1	**GPQCVR** *[Group 7]*
View border indicator (1=OFF, 2=ON)	E	1=OFF	**GPQCVR** *[Group 6]*
View Active Flag for Input			

Table 3. WSL View Parameters at Initialization (continued)

Description	Data Type	Default Value	Inquiry
For view 0 (1=INACTIVE, 2=ACTIVE)	E	1=ACTIVE	**GPQCVR** *[Group 20]*
For all other views (1=INACTIVE, 2=ACTIVE)	E	1=INACTIVE	**GPQCVR** *[Group 20]*
View Active Flag for Output			
For view 0 (1=INACTIVE, 2=ACTIVE)	E	2=ACTIVE	**GPQCVR** *[Group 21]*
For all other views (1=INACTIVE, 2=ACTIVE)	E	1=INACTIVE	**GPQCVR** *[Group 21]*
Temporary view indicator (1=OFF, 2=ON)	E	1=OFF	**GPQCVR** *[Group 9]*
HLHSR (hidden line, hidden surface removal) mode (1=OFF, 2=ON_THE_FLY)	E	1=OFF	**GPQCVR** *[Group 10]*
Transparency mode (1=OFF, 2=PARTIAL_TRANSPARENT, 3=BLEND, 4=BLEND_ALL)	E	1=OFF	**GPQCVR** *[Group 11]*
Antialiasing mode (1=OFF, 2=SUBPIXEL_ON_THE_FLY, 3=NON_SUBPIXEL_ON_THE_FLY)	E	1=OFF	**GPQCVR** *[Group 24]*
Initial color processing index	I	0	**GPQCVR** *[Group 12]*
Initial frame buffer write protect mask	I	0	**GPQCVR** *[Group 13]*
Shield alpha value	I	255	**GPQCVR** *[Group 25]*

Chapter 2. Supported Workstations

This chapter contains general information about the workstations supported by the graPHIGS API. It describes functions and limitations of particular workstations when running graPHIGS API applications. Consider these when writing your application programs.

You should obtain and use the latest level of microcode for your workstation when applicable. This ensures that you have fixes for microcode problems and possible performance enhancements. Ask the person responsible for installing the graPHIGS API on your system to refer to the Program Directory supplied with the latest release for information about the latest microcode release(s).

The X Workstation Family

The X Workstation represents the family of X workstations supported by the graPHIGS API.

The graPHIGS API in the X11 Windowing Environment

This section discusses the interaction between a graPHIGS API application and the X Version 11 Window System. It is not within the scope of this book to explain X or graPHIGS API concepts. Background for the standard X and graPHIGS API terminology in this chapter may be found in *The graPHIGS Programming Interface: Understanding Concepts*.

The X Window System supports multiple applications running in overlapping and hidden windows on one or more screens. X applications, called clients, share common resources such as the keyboard, mouse, display surface, and hardware colormaps. The functions provided in X for window creation, window deletion, window re-sizing, colormap allocation, and event handling make this environment very different from a user running a single full-screen application.

However, X is but one of many workstations supported by the graPHIGS API. This implies that a graPHIGS API application does not need to recognize that it is running in an X window, and that special support is not required to run under X. A graPHIGS API application is not developed specifically for the X environment, and is expected to perform in an X window as if it were running full-screen.

All processing that is unique to the X environment will be handled by the graPHIGS API. For example, when an overlapping window is moved, the application must maintain the contents of the window beneath. X generates events in order to notify the client that the window must be re-drawn. It is the responsibility of the graPHIGS API to manage all X resources and to ensure that the window contents accurately reflect the graphic contents. When the user changes the size of the window, the graPHIGS API manages the graphics and echo areas according to the window mapping method. (Refer to Window Mapping and Resize for more information). Such events are handled by the graPHIGS API on behalf of the application. So in fact, the graPHIGS API ensures that upward compatibility will be maintained and existing applications will run unchanged.

Graphic output for a logical graPHIGS API workstation will be displayed in one corresponding window. For example, if an application opens four X workstations, four windows will be created with a workstation associated to each window.

For the graPHIGS API to create X resources such as windows, it must communicate with an X server by opening a connection to the server. The process by which the graPHIGS API chooses the server is described in Opening the X Workstation. This connection is private and the application cannot access the connection. This allows an application to access both graPHIGS API and X11 functionality without confusion; it can mix graPHIGS API calls with X calls to use both the advanced graphic capabilities provided by the graPHIGS API and the user interface routines provided by X11 toolkits. Having opened a workstation, an application can access X resources by opening its own connection to a server. An application that is making X calls has access to all X windows on the screen because X has very

generous rules for sharing resources. An application could draw into the same window that the graPHIGS API accesses. However, this is strongly discouraged because it would be impossible to synchronize updates. An application mixing X subroutine calls with graPHIGS API subroutine calls should not use X to access the graPHIGS API window.

Both an application coding to the X interface and the graPHIGS API should behave as well-behaved X clients. The graPHIGS API has been designed to minimize its use of server resources.

Since both the graPHIGS API application and the graPHIGS API are considered to be a single X client, it is necessary to understand how the API creates a window, maps the graphics to a window, and handles X events. These topics will be covered in detail in the following sections:

Temporary Views

Temporary views are supported with the following limitations:

Direct Access Capabilities on the RS/6000:

- To perform the view drawing optimizations through use of temporary views:
  - The graPHIGS API must obtain the system resources needed to save the graphic data under a temporary view.
  - The graphic data under a temporary view must *not* be changed.

  Otherwise the temporary view is treated as a normal view and the most efficient uses of the temporary view are not performed.
- The graphic data under a temporary view is preserved across one workstation update only.

View Mapping

If the projection reference point is between the near and far clip planes, the projection type is changed to `PARALLEL` and an error is generated.

Supported Hardware for the X Workstation

In general, the graPHIGS API is intended to operate with an application on an IBM workstation. The graphics generated by the graPHIGS API, when using the X workstation, may be displayed on any equipment that supports a complete X server (X Version 11, Release 4 or later for the RS/6000 platform). The graPHIGS API is designed to exploit the distributed, network transparent, and device independent qualities of X. Because of these capabilities, the user may run a graPHIGS API application on the same machine as the X server or may run a graPHIGS API application on another machine that is connected to the machine on which the X server is being used for display.

IBM does not explicitly support the use of non-IBM equipment running an X server to display the output of the graPHIGS API. However, if the X server supports a full implementation of the X protocol, then there should be little difficulty in using this equipment in this way. The graPHIGS API requires that the target X server support any of the StaticGray, GrayScale, StaticColor, or PseudoColor visual classes. The DirectColor and TrueColor visual classes cannot be used for the X workstation type in XLIB mode.

On some IBM equipment, the graPHIGS API supports advanced rendering capabilities which are available through a method called **Direct Window Access (DWA)**. These capabilities are assisted through hardware on the X workstation when in DWA mode. For a list of these DWA capabilities, see Additional Capabilities Available on RS/6000. DWA capabilities are only available when the graPHIGS API nucleus runs on the same machine as the X server. In order to use the DWA capabilities on a remote machine, the application must connect to a remote graPHIGS API nucleus running on that remote machine.

Brief listings of configurations supporting the X workstation for the graPHIGS API follow:

Table 4. Configurations Supporting X Workstation for graPHIGS API Running on the RS/6000

DISPLAY ADAPTER	FRAME BUFFER DEPTH	BEST AVAILABLE BUFFER CONFIGURATION (Single or Double)[1]	SUPPORTED X WORKSTATION CAPABILITIES	VISUAL CLASS
POWER GXT6500P	24 bit	Double	DWA/XSOFT	TrueColor
	24 bit	Double	DWA/XSOFT	DirectColor
	8 bit	Single	XSOFT/XLIB	PseudoColor
POWER GXT4500P	24 bit	Double	DWA/XSOFT	TrueColor
	24 bit	Double	DWA/XSOFT	DirectColor
	8 bit	Single	XSOFT/XLIB	PseudoColor
POWER GXT6000P	24 bit	Double	DWA / XSOFT	TrueColor
	24 bit	Double	DWA / XSOFT	DirectColor
	8 bit	Single	XSOFT / XLIB	PseudoColor
POWER GXT4000P	24 bit	Double	DWA / XSOFT	TrueColor
	24 bit	Double	DWA / XSOFT	DirectColor
	8 bit	Single	XSOFT / XLIB	PseudoColor
POWER GXT300P	24 bit	Single	XSOFT	TrueColor
	24 bit	Single	XSOFT	DirectColor
	8 bit	Single	XSOFT / XLIB	PseudoColor
POWER GXT2000P	24 bit	Double	DWA / XSOFT	TrueColor
	24 bit	Double	DWA / XSOFT	DirectColor
	8 bit	Single	XSOFT / XLIB	PseudoColor
POWER GXT3000P	24 bit	Double	DWA / XSOFT	TrueColor
	24 bit	Double	DWA / XSOFT	DirectColor
	8 bit	Single	XSOFT / XLIB	PseudoColor
POWER GXT1000 and POWER GXT800P	24 bit	Double	DWA / XSOFT	TrueColor
	24 bit	Double	DWA / XSOFT	DirectColor
	8 bit	Single	XSOFT / XLIB	PseudoColor
POWER GXT550P	24 bit	Double	DWA / XSOFT	TrueColor
	24 bit	Double	DWA / XSOFT	DirectColor
	8 bit	Double	DWA / XSOFT / XLIB	PseudoColor
POWER GXT500P	24 bit	Single	XSOFT	TrueColor
	24 bit	Single	XSOFT	DirectColor
	8 bit	Double	DWA / XSOFT / XLIB	PseudoColor
	12 bit	Double	DWA	DirectColor
POWER Gt4 (24 bit), POWER Gt4x (24 bit), and POWER Gt4xi (24 bit)	24 bit	Double	DWA / XSOFT	DirectColor
	8 bit	Single	XSOFT / XLIB	PseudoColor

Table 4. Configurations Supporting X Workstation for graPHIGS API Running on the RS/6000 (continued)

DISPLAY ADAPTER	FRAME BUFFER DEPTH	BEST AVAILABLE BUFFER CONFIGURATION (Single or Double)[1]	SUPPORTED X WORKSTATION CAPABILITIES	VISUAL CLASS
POWER GXT255P	8 bit	Double	DWA / XSOFT / XLIB	PseudoColor
	24 bit	Single	XSOFT	TrueColor
	24 bit	Single	XSOFT	DirectColor
POWER GXT250P	8 bit	Double[2]	DWA / XSOFT / XLIB	PseudoColor
	8 bit	Single	XSOFT / XLIB	PseudoColor
POWER GXT500D	24 bit	Double	DWA / XSOFT	TrueColor
	24 bit	Double	DWA / XSOFT	DirectColor
	8 bit	Double	DWA / XSOFT / XLIB	PseudoColor
POWER GXT500	24 bit	Single	DWA / XSOFT	TrueColor
	24 bit	Single	DWA / XSOFT	DirectColor
	8 bit	Double	DWA / XSOFT / XLIB	PseudoColor
	12 bit	Double	DWA	DirectColor
POWER Gt1x	8 bit	Single	XSOFT / XLIB	PseudoColor
POWER GXT100 (8 bit)	8 bit	Single	XSOFT / XLIB	PseudoColor
POWER Gt4x (8 bit), POWER Gt4xi (8 bit)	8 bit	Double	DWA / XSOFT	PseudoColor
	8 bit	Single	XSOFT / XLIB	PseudoColor
POWER Gt4e (8 bit)	8 bit	Double	DWA / XSOFT	PseudoColor
	8 bit	Single	XSOFT / XLIB	PseudoColor
POWER Gt3i	8 bit	Single	XSOFT / XLIB	PseudoColor
POWER GTO (24 bit)	24 bit	Double	DWA / XSOFT	DirectColor
	8 bit	Single	XSOFT / XLIB	PseudoColor
POWER GTO (8 bit)	8 bit	Double	DWA / XSOFT	PseudoColor
	8 bit	Single	XSOFT / XLIB	PseudoColor
Color Graphics Display Adapter	8 bit	Single	XSOFT / XLIB	PseudoColor
GrayScale Graphics Display Adapter	4 bit	Single	XLIB	GrayScale

Note: [1] DWA capabilities support the double buffer configuration only. XSOFT and XLIB capabilities support the single buffer configuration only.

[2] Maximum screen size is 1024x768 for DWA capabilities.

Opening the X Workstation

When a graPHIGS API application wants to open a logical workstation, it must specify a workstation type and connection identifier. The workstation type for the X workstation support must be 'X' and the connection ID can either be '*' or the standard X windows server specification 'host:server.screen'. If you specify an asterisk, then the graPHIGS API connects to the server listed in the operating system variable `DISPLAY`.

The value of the operating system variable LANG determines the graPHIGS API primary character set.

- If the LANG variable is set to:
  - ja_JP
  - Ja_JP
  - En_JP or
  - Jp_JP

  then the character set identifier 6 (Katakana) is used as the primary character set.
- If the LANG variable is set to:
  - ko_KR

  then the character set identifier 9 (single-byte Korean) is used as the primary character set.
- If the LANG variable is set to:
  - zh_TW or
  - zh_CN

  then the character set identifier 8 (Multi-Language) is used as the primary character set.
- If the LANG variable is
  - any lower case letter other than ″j″, ″k″, or ″z″,

  then the character set identifier 10 (ISO 8859-1) is used.
- If the LANG variable is set to any value other than those listed above, then the character set identifier 8 (Multi-Language) is used as the primary character set.

To check the operating system variable LANG, enter the ″echo $LANG″ command. To set the operating system variable LANG, for example, to use Katakana (character set 6) as the primary character set identifier, enter the commands ″LANG=ja_JP″ and ″export LANG″.

The workstation type and the connection identifier can be specified through the defaults processing (EDF file or ADIB) and as parameters on the Open Workstation (**GPOPWS**) or the Create Workstation (**GPCRWS**) subroutines. The following examples illustrate two ways of specifying the connection identifier and workstation type:

1. Sample line from the External Defaults File (EDF) called PROFILE

```
AFMMNICK  TOWSTYPE=X,
          TOCONNID=unix:0
```

2. Sample 'C' language call to Create the Workstation:

```
GPCRWS(wsid, ncid,1,"*","X       ",0) ;
```

   - The *ncid* should be 1 if you do not issue the Connect to Nucleus (**GPCNC**) subroutine call.
   - The third parameter is the length (1) of the connection ID (″*″).
   - The fourth parameter is the workstation type and must be 8 characters.

Note: See the description under Additional notes for DWA Adapters for more information on opening graPHIGS Direct Window Access(DWA) Workstations.

Window Creation

The window associated with a workstation can be created by either the application or the graPHIGS API. This has been designed to provide maximum flexibility. The application can create the window and pass the window identifier to the API via the XWINDID PROCOPT. By default, the graPHIGS API will create the window as a child of the root window (top-level window), using the same visual, depth and screen as the root window. It will not create the window if the user specifies the XWINDID PROCOPT.

If the API creates the window on behalf of the application, the user may specify some information about the window that will define the initial position, initial size and window appearance. This information is

communicated to X through two mechanisms, the *XCreateWindow* subroutine call and *properties*. The following will explain how the graPHIGS API will use these mechanisms to create a window:

Properties

X allows the graPHIGS API to associate information, called properties, with a window that other clients can access. Properties that are processed by a window manager are called *hints*. A hint is appropriately named because a window manager has the authority to interpret or ignore any property. This is to say that it impossible to state exactly how the window will ultimately appear because each window manager will interpret hints differently. The graPHIGS API has chosen a standard set of properties to define on a window. These properties can be used by a window manager to define the initial position, initial size, window title, icon name, icon bitmap, minimum aspect ratio, maximum aspect ratio, window border color and the window border width.

The graPHIGS API will define these properties based on default information provided by the user. The graPHIGS API will use the standard methods for collecting user preferences, namely the **.Xdefaults** file. The user can specify a string, which is usually the application name, via a graPHIGS API `PROCOPT` called `XNAME` (see XNAME). The graPHIGS API will use this string to identify defaults specified in the **.Xdefaults** file. Table 5 describes the properties that the user can specify and the graPHIGS API system default action if the user has not specified the attribute.

Table 5. Window Creation Defaults for .Xdefaults File

Description	Format	graPHIGS API Default Action
Initial Window Geometry	*name*.geometry: WidthxHeight+(-)X+(-)Y	See discussion below
Minimum Window Size	*name*.minSize: WidthxHeight	100x100
Window Title	*name*.title: MyTitle	Blank Title
Icon Name	*name*.MyIcon: MyIcon	Default to the Window Name
Icon Bitmap Filename	*name*.iconBitmap: BitmapFilename	No icon bitmap
Minimum Aspect Ratio	*name*.aspectMinimum: WidthxHeight	None specified
Maximum Aspect Ratio	*name*.aspectMaximum: WidthxHeight	None specified
Window Border color (Pixel Value)	*name*.borderColor: color	Zero
Window Border width	*name*.borderWidth: width	Zero
Note: The value of the `XNAME PROCOPT`(see XNAME) is substituted here as *name*		

Each of the defaults may be prefixed by a *name* or a wildcard. This name can be passed in via the graPHIGS API `XNAME PROCOPT` (see XNAME), otherwise, the graPHIGS API will default the name to ″graPHIGS″.

If the user does not specify the initial window geometry, the graPHIGS API will create a window half the width of the screen and with the same aspect ratio. This is to insure that the window is large enough for the user to easily locate. Your window manager may prompt you to size or move the initial window if the graPHIGS API specifies the initial window geometry. The values that the graPHIGS API supplies will be used for the outline of the rubber band box that the window manager will display.

The window border color is specified as a pixel value and not as a named color. This is due to the fact that the graPHIGS API may associate a colormap to the window and the named color will not necessarily correspond to the pixel value.

A sample **.Xdefaults** file with the above defaults is supplied when you install the graPHIGS API diskettes. For the graPHIGS API to find your defaults, you will have to place the default information in your **.Xdefaults** file in your $HOME directory.

Converting Coordinates

The Convert Coordinate Values (**GPCCV**) subroutine can be used by the application to convert coordinate units among the NPC, DC, and window units (WU) ranges.

Inquiring Window Size

The Inquire Mapped Display Surface Size (**GPQMDS**) subroutine returns the size of the area that displays the DC range. In the `1=MAPPED` method, the display is constrained to an area with the same aspect ratio as the display surface[1]. In the `2=DIRECT` method, the value returned is the current size of the window, constrained to the same area as the root window. The application can use this value as the current size of the displayable part of DC.

Note: [1] This constraint disappears when you use the XWINDASP PROCOPT to alter the aspect ratio of the root window.

XCreateWindow

Information derived from the **.Xdefaults** file will be used to change some of the X window attributes via the XCreateWindow subroutine call. The .Xdefaults information is only processed when the window is created by the graPHIGS API and therefore the corresponding window attributes will only be changed at this time. Window attributes are also changed by the graPHIGS API in order to process events and color. The following table briefly illustrates which attributes will be used by the graPHIGS API. The column on the left applies when the API creates the window and the column on the right applies when the application creates the window.

Table 6. Attribute Table

Attribute	graPHIGS API	Application
1) background_pixmap	*	
2) background_pixel	*	
3) border_pixmap	*	
4) border_pixel	*	
5) bit_gravity	+	+
6) win_gravity		
7) backing_store	+	+
8) backing_planes	+	+
9) backing_pixel	+	+
10) override_redirect		
11) save_under	+	+
12) event_mask	*	*
13) do_not_propogate		
14) colormap	*	* [1]
15) cursor	*	*
Key: * = the graPHIGS API will change this attribute + = the graPHIGS API may use this attribute in the future		
Note: [1] This is dependent on the `XNOCLRMP PROCOPT` (see XNOCLRMP (Do Not Create an X Color Map)).		

When the graPHIGS API has created and associated a colormap to the window, the API will require that a window manager be installed. This ensures that the graPHIGS API colormap will be installed by the window manager when the pointing icon is in the graPHIGS API window. The client should avoid grabbing the input focus and installing the colormap and allow the window manager to perform these actions.

Additional Capabilities Available on RS/6000

Under specific circumstances described in the list below, the graPHIGS API provides access to graphics processor capabilities not normally available through the X Protocol. These additional capabilities allow exploitation of hardware features of a graphics processor. The availability of these capabilities is indicated in the actual WDT of the workstation. Specific capabilities are described in Workstation Description Tables as direct access features of the X workstation type.

See Table 4. Configuration supporting X Workstation for graPHIGS API Running on the RS/6000 to determine which adapters support **DWA** mode. To obtain direct access to any of these adapters, the following conditions are necessary:

- The application connects to a graPHIGS API nucleus running on a RS/6000.
- The X workstation connection *id* specifies an X server running on the same physical RS/6000 as the graPHIGS API nucleus to which the **GPOPWS** or **GPCRWS** procedure is directed.

PROCOPTs Supported by the X Workstation

XWINDID

The application can pass the graPHIGS API a window ID and the API will use this for the workstation display window. If the application creates the window, then it cannot change the window ID without closing the workstation and re-opening the workstation with a new ID. It is the responsibility of the application to map the window before opening the workstation.

A potential problem with an application creating its own window is that it cannot know for certain whether the graPHIGS nucleus on which the X workstation will be created is local to the X server where the window was created. For example, if the application intended to open an XDWA workstation but the application's window was created on an X server that is remote to the graPHIGS nucleus, then an XDWA workstation will fail during its initialization. This situation can happen readily if the graPHIGS nucleus that is connected to has been changed via a TONUC or DEFNUC default in an EDF. The application could use the **GPQNCE** (Inquire Nucleus Environment) subroutine to get the Internet address and the hostname of the system that the nucleus is running in order to avoid this situation. Also, you can see the GPES subroutine (escape 1018) for information on getting a list of supported visuals in order to guarantee that the application's window will be usable by the graPHIGS nucleus.

It is important to remember when defining your window that the client should avoid installing the colormap, and allow the window manager to perform these actions. The window manager will install the colormap automatically for top level windows. For descendents of the top level windows which have different colormaps, there is no current convention as to how their colormaps will be made active. Typically, descendents of a top level window will share the colormap associated with the top level window. See XWINDID (X Window Identifier) for additional information.

The following code illustrates the steps necessary to create a window and pass the window identifier to the graPHIGS API:

```
/* Declare the defaults data structure */
struct
{
  int   adib_len ;

  struct
  {
    int len ;
    int code ; /* code for following nicknames */

    /* nickname - must be in adib */
    char  fromws[8] ;
    char  fromconn[8] ;
    char  tows[8] ;
    char  toconn[8] ;

    /* PROCOPT for window id*/
    struct
    {
       int len ;
       int code ;
       int window_id ;
    } ads1 ;

  } nicknames ;

} adib ;

  /* Open a connection to the server */
  if (!(dpy = XOpenDisplay(NULL)))
  {
       printf("Cannot open display \n");
       exit(0);
  }
  /* Create an X window using a simple version of XCreateWindow */
  win = XCreateSimpleWindow(dpy,
                            RootWindow(dpy,0),
                            DisplayWidth(dpy,0)/4,
                            DisplayHeight(dpy,0)/4,
                            DisplayWidth(dpy,0)/2,
                            DisplayHeight(dpy,0)/2,
                            0,0,0);
   /* Map the window  */
   XMapWindow(dpy, win);
   XSync(dpy,FALSE);

  /* initialize the adib */
  adib.adib_len                = sizeof(adib) ;
  adib.nicknames.code          = 2001 ;
  adib.nicknames.len           = sizeof(adib.nicknames) ;
  strncpy(adib.nicknames.fromws,"        ",8) ;
  strncpy(adib.nicknames.fromconn,"        ",8) ;
  strncpy(adib.nicknames.tows    ,"X       ",8) ;
  strncpy(adib.nicknames.toconn  ,"*       ",8) ;
  adib.nicknames.ads1.len       = sizeof(adib.nicknames.ads1) ;
  adib.nicknames.ads1.code      = 25 ;
  adib.nicknames.ads1.window_id = win ;

  /* pass the defaults to the graPHIGS API */
GPOPPH(erfile,&adib) ;
```

XNAME

This name will be used to resolve the defaults in the **.Xdefaults** file. The following example illustrates the format of XNAME and the corresponding format of the defaults in the **.Xdefaults** file:

```
AFMMNICK  TOWSTYPE=X,
          TOCONNID=unix:0,
          PROCOPT=((XNAME,MyName))
```

.Xdefaults file in your $HOME directory

```
MyName.geometry: 500x500+0+0
MyName.title   : MyTitle
```

If the XNAME PROCOPT (see XNAME (X Default String)) is not specified, then the graPHIGS API will use the string ″graPHIGS″ to resolve the defaults. The **.Xdefaults** file would look like the following:

```
graPHIGS.geometry: 500x500+0+0
graPHIGS.title   : MyTitle
```

XNOCLRMP

This option is only processed by the graPHIGS API if the application has also specified the XWINDID PROCOPT (see XWINDID (X Window Identifier)). The API will not create a colormap if this PROCOPT has been defined. This implies that the application cannot access the colormap via the graPHIGS API. The API will initialize the display colormap as non-modifiable by the application. This allows the application to have full control over the colormap without graPHIGS API intervention. Refer to Interaction of X and graPHIGS API Color Resources and XNOCLRMP (Do Not Create an X Color Map) for more detailed information.

XWINDASP

When the workstation is created, the graPHIGS API X-Windows device driver establishes the display surface size in the Workstation Description Table (WDT). The graPHIGS API PROCOPT XWINDASP (see XWINDASP (Window Aspect Ratio)) is provided to allow the application or user to specify the aspect ratio of the display surface that is mapped to the window. If the window already exists (identified using the XWINDID PROCOPT (see XWINDASP (Window Aspect Ratio)), then the graPHIGS API X-Windows device driver uses the largest subarea of the window that has the aspect ratio specified in the XWINDASP PROCOPT. If the window does not exist, then a window is created, using hints in the **X defaults** file (or workstation defaults if the **X defaults** file hints do not exist). The graPHIGS API X-Windows device driver uses the largest subarea of the created window that has the aspect ratio specified in the XWINDASP PROCOPT. If the XWINDASP PROCOPT is not specified, then the aspect ratio of the root window is used. Refer to Controlling the Environment with Defaults and Nicknames for the format of the PROCOPT in the External Defaults File (EDF) or the Applications Interface Defaults Block (ADIB).

Other Supported PROCOPTs

Refer to Controlling the Environment with Defaults and Nicknames for information relating to the other supported PROCOPTS.

X Events

SYNCPROC mode

On the operating system, the graPHIGS API has been using an IBM extension to Xlib (X Asynchronous Event Handling) in order to receive X events. Using this extension, the graPHIGS API defines an event handler and receives X events (that pertain to the windows for any graPHIGS X workstations that are open) via an X signal handler. This technique allows the graPHIGS X workstations to handle events regardless of whether the application process is executing graPHIGS API code or not.

Unfortunately, the use of this asynchronous event extension has caused problems for some graPHIGS applications. Because of these problems, another method of handling X events was needed, one which used a synchronous method to receive X events. To use this method, a graPHIGS application has to:

- Turn on the graPHIGS SYNCPROC default (see Controlling the Environment with Defaults and Nicknames for more information on using graPHIGS defaults).

- Call **GPQSID** (Inquire the list of Socket Identifiers) to get a list of socket identifiers currently used by the graPHIGS API. This routine should be called every time a graPHIGS resource (for example, a graPHIGS nucleus or any nucleus resource) is created or destroyed.
- The socket identifiers have to be used by a routine that will tell the application when one or more of the sockets is active. The **XtAppAddInput** subroutine is one way of giving the X Toolkit the socket information (along with a callback subroutine) so that an **XtAppPending** subroutine (or an **XtAppMainLoop** subroutine) could be used to check on the sockets. The application could also set the socket identifiers into a read mask and then use a select subroutine to wait for one or more of the sockets to become active. The method used to discover whether one or more of the sockets is active depends on the application being used.
- When one or more of the sockets is active, then the **GPRDEV** (Redrive Events) subroutine must be called so that the graPHIGS API can handle the events.
- **GPRDEV** should also be called when the application uses graPHIGS workstation subroutines that cause an update to the graPHIGS workstation window.
- If the application uses graPHIGS events, then it must continue to either use an event handler or call **GPAWEV** (Await Event) to retrieve these graPHIGS events.

Examples of running with the graPHIGS SYNCPROC mode can be found in the following sample programs:

- /usr/lpp/graPHIGS/samples/samp/sampc.c
- /usr/lpp/graPHIGS/samples/widgets/lib/gPWorkstation.c and
- /usr/lpp/graPHIGS/samples/widgets/samples/viewers/viewers.c

Unfortunately, using a synchronous event handling method is complicated and depends heavily on the design of the graPHIGS application. Also, there are cases where using synchronous event handling cannot work without code changes (for example, when input devices are used in request mode).

For this reason, many graPHIGS applications would not want to use this method and, in general, most applications either handle all events themselves (which they can do if the applications create their own X windows and pass the window identifiers in a procopt in the Create Workstation (**GPCRWS**) subroutine or the Open graPHIGS (**GPOPPH**) subroutine and they do not use graPHIGS input devices), or the applications let the graPHIGS workstations handle the X events.

In the AIX 4.3, the default Xlib subroutine libraries moved from X11R5 to X11R6, and the Asynchronous X Event Handling extension which exists in X11R5 on the operating system was not ported to X11R6. Thus, the graPHIGS API, by default, must handle X events synchronously. The graPHIGS workstations cannot use a signal handler to handle X events because the X11R6 libraries are not reentrant. A separate thread cannot be used to handle X events because most applications are not linked such that the pthreads library can be used. So, to allow graPHIGS applications to run without modification when using X11R6, the graPHIGS shell creates a separate process that contains the graPHIGS nucleus: a graPHIGS child nucleus. See Advanced Concepts for more information about the concepts of the graPHIGS shell and nucleus.

With the graPHIGS nucleus in a separate process, the graPHIGS workstations are readily able to monitor their X display connections for X events. graPHIGS applications that are using the SYNCPROC default will continue to have the shell and nucleus in the same process (since they are already handling X events in a synchronous manner).

Most graPHIGS applications should not be affected by the use of the graPHIGS child nucleus. However, some applications may see performance problems— especially applications that use a lot of graPHIGS inquiry subroutines. This is due to the overhead involved in process context swapping that occurs when switching between the graPHIGS shell and the graPHIGS nucleus. If a graPHIGS application finds that it has a performance problem when running in X11R6 and the application does not use the SYNCPROC default, as long as the application is not dependent on any X11R6 function (for example, if the application

was created on an earlier version of the operating system and was not rewritten to make use of new functionality of X11R6), then the application can run with the X11R5 libraries and the graPHIGS API will run as in previous releases.

To use the X11R5 libraries, the operating system variable **LIBPATH** must be set up to point to the **/usr/lpp/X11/R5** directory before it points to **/usr/lib** or **/lib**. For example, using a **ksh**, the command to set up the LIBPATH would be:

```
export LIBPATH=/usr/lpp/X11/R5:$LIBPATH
```

If the application sets up an implicit **LIBPATH** (that is, a path specified when using a load subroutine), then it must make sure to have **/usr/lpp/X11/R5** in the path before **/usr/lib** or **/lib**.

Window Deletion

Many current window managers provide a means for the end-user to ″close″ or delete a window (via a pull-down menu, a special key sequence, or a similar method). The action that a window manager takes in this case is specific to that window manager. Often, the default case is for a window manager to issue `XKillClient()`, which simply closes the connection to the X Windows display. The client receives an X I/O Error and is expected to terminate immediately. However, this is unacceptable for an application that may have resources open or active that should be closed or saved.

The graPHIGS API is notified of the window deletion via the `WM_DELETE_WINDOW` protocol as defined by the X11.4 ICCCM. The application may request a `WINDOW_DELETE` notification event (107) through the `WINDOW_DELETE_NOTIFY` escape (1012).

There are three ways in which the application can interact with the `WINDOW_DELETE` function:

- The application enables the event
- The application disables (or does not enable) the event (the default case)
- The application uses the `XWINDID PROCOPT` (see XWINDID (X Window Identifier)).

The Application Enables the Event

If an application enables the event through the use of the escape call, the graPHIGS API puts a new event on the event queue when the end user initiates a window close (via some window manager specific action). This event is `WINDOW_DELETE` (107). The graPHIGS API takes no other actions. If the application chooses to ignore the event, the window stays on the screen, and normal processing may continue. The intent, however, is to allow the application to conduct some ″close down″ confirmation dialogue with the end user. The application may then, at its option, close the workstation or close graPHIGS API.

The Application Disables the Event

With this default case, or when the application explicitly disables the event, the graPHIGS API generates an error when the end user initiates a window close. The error number is 930, message number 2045. At this point, the graPHIGS API also unmaps (removes) the window.

The application may define an error handler to trap this error and then somehow notify the mainline application code that the window has gone away. In this way, the application may at least do some ″close down″ of its own.

The Application Uses the XWINDID PROCOPT

The method by which a client of an X window receives the `WM_DELETE_WINDOW` message is to request it via XSetVMProtocols. When an application uses the `XWINDID PROCOPT` (see XWINDID (X Window Identifier)), the graPHIGS API nucleus is not the client of the window and is unable to request the `VM_DELETE_WINDOW` protocol messages. In this case, the application should enable the `WM_DELETE_WINDOW` protocol to avoid having the window manager issue `XKillClient` on the window. If the application chooses to *not* enable the `WM_DELETE_WINDOW` protocol itself, then the results are unpredictable, and an abend or a hang could occur. The following sample code illustrates how an application may enable the `WM_DELETE_WINDOW` protocol:

```
int npcol;
Status xrc;
Atom *newpcols,*pcols;
int i;

/*-------------------------------------------------------*/
/* Enable Delete_Window window manager function          */
/*-------------------------------------------------------*/
  if ( (wmproto =
        XInternAtom(dpy,
"WM_PROTOCOLS",True))
       == None)
    goto DW_SKIP ;

  if ( (wm_delwin =
        XInternAtom(dpy,
"WM_DELETE_WINDOW",True))
       == None)
    goto DW_SKIP ;

  /* Both atoms exist at the server, continue...           */
  npcol = 0;
  if ( XGetWMProtocols(dpy,win,&pcols,&npcol) )
    /* non-zero return code ==> call failed                */
    /* ...just skip...                                     */
    goto DW_SKIP ;

  /* scan the list of protocols -- see if WM_DELETE_WINDOW  */
  /* is already there                                      */
  for (i = 0; i < npcol; i++)
    if (pcols[i] == wm_delwin)
      break;
                    /* if already there, just skip       */
  if (i < npcol) goto DW_FREE ;

  /* Allocate storage for old list plus one more           */
  newpcols = (Atom *)
      malloc( (npcol+1)*sizeof(Atom) ) ;
  if ( newpcols == (Atom *)0 )
  {
    printf(
"malloc failed!\n");
/* Output error message */
    goto DW_FREE ;
  }

  /* if there was an old list, copy it in                  */
  if (npcol) memcpy(newpcols,pcols, npcol * sizeof(Atom));

  /* Append WM_DELETE_WINDOW protocol atom to list         */
  newpcols[npcol] = wm_delwin;

  /* Set new list of protocols for this window             */
  XSetWMProtocols(dpy,win,newpcols,npcol + 1);

  free( newpcols );

DW_FREE:
  XFree( pcols );

DW_SKIP: ;
```

Window Mapping and Resize

The PHIGS static model for the display surface is maintained by the graPHIGS API and the device coordinates are defined as the size of the root window on the default screen. Typically, these are the maximum extents of the display surface. X allows windows to be larger than the display surface, but the larger windows will not be completely visible.

The Set Device Coordinate Mapping Method (**GPDCMM**) subroutine allows your application to select either `1=MAPPED` or `2=DIRECT` as the window mapping method. The `1=MAPPED` method of display is the default. When using the `2=DIRECT` display method, the graPHIGS API displays the device coordinate (DC) range directly in the X-Window with no scaling. This method of display is analogous to a ″porthole″ rather than the ″rubber sheet″ behavior exhibited by the `1=MAPPED` method of display.

Mapped Display Method

If the window mapping method is set to `1=MAPPED`, the graPHIGS API will scale all the data for a workstation to the current window size, maintaining the aspect ratio of the device coordinates. A workstation with square device coordinates will be mapped to the largest square region in the window and a workstation with rectangular device coordinates will be mapped to the largest rectangular region in the window. The display surface will also be centered in the window.

With the exception of the pixel primitive, all primitives, including annotation text and polymarkers, are scaled to the window. The pixel primitive position is transformed to the new window size but the size of the pixel primitive represents a fixed number of pixels on the screen. In addition to the primitives, all the input echo areas, echoes, **GPMSG** and pick aperture are scaled.

Direct Display Method

When a workstation is opened, the DC values in the workstation description table are initialized using the size of the root window for the DC limits. When graPHIGS API data is displayed in an X-Window, the lower-left corner of the DC volume is aligned with the lower-left corner of the window. If the window is smaller than the DC range addressed in the data, then the window clips the data. If the window is larger than the display data, the area of the window beyond the DC range of the data is unused.

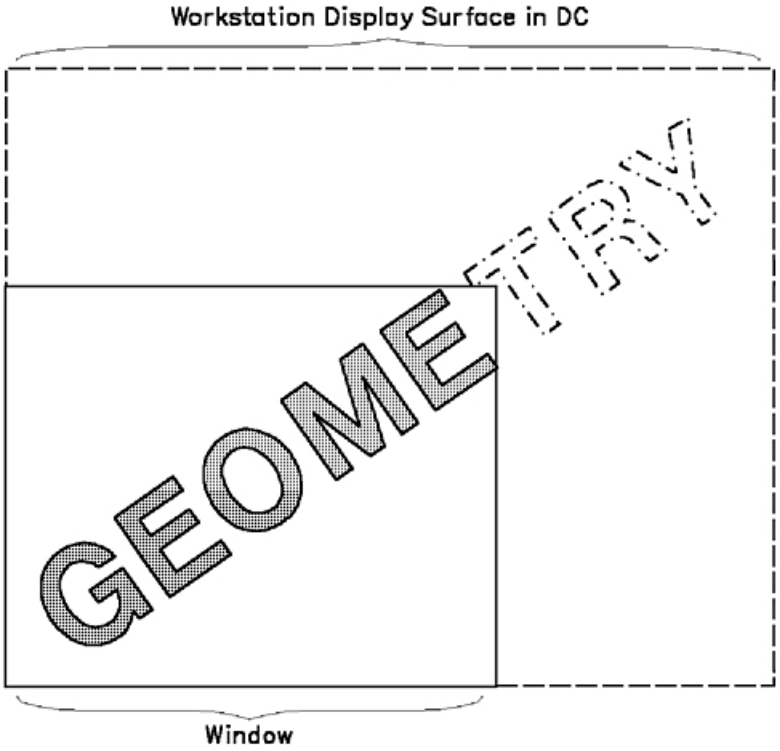


Figure 1. Direct Method of Display to X-Window. This illustration depicts the clipping of text by a window. The illustration shows a rectangular region (the window) within a larger region (the display surface). The smaller region contains the first five letters of the word geometry (geome). A ghost image of the last three letters of geometry (try) is shown in the larger region.

When the `2=DIRECT` method is used, neither the geometry nor the rendered size (line width, for example), is scaled. The area available for display is the only thing that changes. The application can, however, use the transformation pipeline to cause the geometry to grow or shrink. A control variable is available to scale the primitive nominal DC sizes, allowing the application to stretch, shrink, or leave them unchanged. The scaled nominal primitive size is used with various scale factor attributes, such as the line width scale factor, to render the primitive. This control allows applications to globally scale primitive sizes without changing scale factor attributes. This scale factor affects the following DC values:

- Nominal line width
- Nominal marker size
- Nominal edge width
- Nominal annotation height

The `2=DIRECT` display method, used with other graPHIGS API functions, allows an application to fill an X-window when the user resizes the window. The following sequence is an example:

1. The transformation pipeline transforms a region of the Normalized Projection Coordinate system (NPC) to a viewport in device coordinates (DC), filling the X-window.
2. The application uses the Window Resize Notification Control Escape to enable event notification when a resize occurs.
3. The application receives the resize event and uses the Get Window (**GPGWIN**) subroutine to determine the new window size.
4. The application changes the view table and workstation transformation to the new values so that the specified area of World Coordinates fills the new X-window area.
5. The application issues update workstation, causing the window to be redrawn with the new transformation values.

While this processing takes place, no WDT or WSL values associated with device coordinates are changed. The maximum DC value that the application can specify is the size of the root window. Specifying larger values results in an error. DC values smaller than the WDT DC are clipped to the current window size. This adjustment is not an application error nor is any warning returned to the application.

For any window mapping method, the application can request notification when a window resize occurs. The graPHIGS API enables (or disables) the notification of window resize events through an escape function (1009: Window Resize Notification Control). When this notification is enabled, all window resize events are sent to the graPHIGS API application by using the graPHIGS event queue. When the application gets control back from the Await Event (**GPAWEV**) subroutine, an event code indicates that the window size has changed.

Be aware that resizing the graPHIGS API window causes an implicit update of the display. Any deferred actions on the display surface will occur with a resize. The application can request, through escape 1009 (Window Resize Notification Control), whether or not it wants the graPHIGS API to redraw the contents of the window when a resize occurs. By default, the graPHIGS API redraws the contents of the window when a resize occurs.

The inquiry subroutine, Inquire Mapped Display Surface Size (**GPQMDS**) allows the application to obtain the display size on an X workstation. This size is referred to as the mapped display surface size. The **GPQMDS** subroutine returns the size of the window in device coordinates (i.e. meters) and in address units.

The Geometric Text Culling Escape, which accepts a size in device coordinates, is interpreted as the size on the physical screen and not as a size on the workstation display surface, and thus the cull size is not scaled to the window. Geometric Text Culling is an optimization used by the X device driver to replace geometric text with a box, or completely clip it, when the geometric text is too small to read. When the window size is increased and the text is large enough to be read, the cull size is not scaled to allow the text to be drawn.

Exposure Events

The application can request notification when a window exposure occurs. The graPHIGS API enables (or disables) the notification of window exposure events through the escape function (1011: Window Exposure Notification Control). When notification is enabled, all window expose events are sent to the graPHIGS API application using the graPHIGS API event queue as event class 106. The graPHIGS API exposure event includes data which must be retrieved using the Get Window (**GPGWIN**) subroutine, for event class 106 (Window Exposure Event). **GPGWIN** returns a bit field of flags indicating which views have been affected by the exposure.

Additionally, the application can use the Window Exposure Notification Control Escape to specify if the graPHIGS API should update the currently displayed screen when a window exposure occurs. If the application chooses to update the display itself, the graPHIGS API clears the exposed rectangular region on the visible rendering target at the earliest possible moment. For DWA clients, the rectangles are not cleared while Immediate Elements are being rendered within a Begin Structure - End Structure sequence. When the graPHIGS API clears exposed regions, only the visible rendering target is cleared. No other rendering targets or rendering resources are affected by the clear. (Refer to Explicit Traversal Control for additional information.)

> Note: An implicit update may occur to reflect the current contents of `STRUCTURE STORE` on DWA Adapters. See Table 4. Configurations Supporting X Workstation for graPHIGS API Running on the RS/6000 to determine which adapters support **DWA** mode.

Collapsing Events

The graPHIGS API attempts to collapse multiple X Windows exposure and configure notify events into a single graPHIGS API event class. For multiple exposure events, the graPHIGS API returns a single Window Exposure Event (event class 106) with a list of views affected by all the exposed regions of the

window. For combinations of window exposure and configure notify, only the configure notify is returned to the application as a Window Resize notification, and no Window Exposure events are returned in this case.

Interaction of X and graPHIGS API Color Resources

Color management is one of the most difficult topics that an X windows application must deal with. Therefore, the graPHIGS API has been designed so that applications do not have to be aware that they are operating in an X window. For those applications that wish to have a closer integration with other clients, the graPHIGS API provides the flexibility to manage the interaction of color with other X clients.

The major difficulty in managing color resources is that the physical resources of a display are limited and must be shared by all clients using that display. The X server isolates the client from these limitations by virtualizing the color resources so that each client can use as many colors as it needs. Each window has a colormap that defines the mapping between the pixel values used in the window and the color that will appear on the monitor. Different windows can share colormaps or have unique ones. The colormap is associated to the window through the colormap attribute of the window.

When virtual color allocations exceed the available physical resources, only a subset of the clients can have their requested colors active at one time. For top level windows which are children of the root, the decision as to which colormaps should be active is left up to the window manager. For descendents of the top level windows which have different colormaps, there is no current convention as to how their colormaps should be made active. Typically, the descendents of a top level window will share the colormap associated with the top level window.

When the number of virtual colormaps exceeds the number of physical colormaps, the window manager will enforce some policy as to which windows have their virtual colormaps loaded into the physical ones. The window manager will typically ensure that the top level window that contains the pointing device has its colormap loaded. This implies that other windows may not have their colormap installed and therefore will be displayed with the wrong colors. This produces what has become known as the ″false color effect″: as the input focus moves from window to window, the colors in some windows may change. Most window managers install the colormap of the window that has the input focus. In the worst case, some windows may become invisible or incomprehensible because the pixel values used in those windows correspond to approximately the same intensities of black or white in the currently installed colormap. To avoid this ″technicolor effect″, either the hardware must provide additional physical colormaps or the clients must be programmed to share colormap entries.

X also defines six different techniques for mapping pixel values into a color or intensity on a monitor. These are referred to as visual classes. The visual class of a window is defined when it is created and must be one that is supported by the server for the target screen. The six visual classes are:

- StaticGray
- GrayScale
- StaticColor
- PseudoColor
- TrueColor
- DirectColor

If you are unfamiliar with these concepts, you should refer to the X documentation since these are key to understanding how a graPHIGS API workstation interacts with the resources of an X server.

One of the important attributes of visual classes StaticGray, StaticColor, and TrueColor is that their colormaps are read only while the other three visual classes have colormaps that can be modified as well as read.

When the graPHIGS API creates a window, the visual class will default to that of the root window on the target screen. If the application creates the window, the graPHIGS API will use the visual class of the window that is passed in. The visual class will be used to determine some characteristics of the graPHIGS API workstation that is created. (See the **GPES** subroutine for information on the Inquire X Visual List Information escape.) These characteristics are summarized in the following table:

Table 7. WDT Content For Each Visual Class

	StaticGray	**GrayScale**	**StaticColor**	**Pseudo-Color**	**TrueColor (Note 1)**	**Direct-Color (Note 1)**
Frame Buffer Type	Indexed	Indexed	Indexed	Indexed	Component	Component
No. Frame Buffer Components	1	1	1	1	3	3
Color Available	No	No	Yes	Yes	Yes	Yes
Display Color Table Size	See Note 2	See Note 2	See Note 2	See Note 2	See Note 2	See Note 2
Is the Display Color Table modifiable?	No	See Note 3	No	See Note 3	No	See Note 3
Available Echo Methods	XOR	XOR and Bit Plane	XOR	XOR and Bit Plane	XOR	XOR and Bit Plane
Rendering Color Table Default Content (Note 4)	Identity Index Map	Identity Index Map	Identity Index Map	Identity Index Map	Identity Index Map	Identity Index Map
Default Color Table for **GPCR**	0	See Note 5	0	See Note 5	0	See Note 5
Default Color Processing Method	Bitwise	Bitwise	Bitwise	Bitwise	Bitwise	Bitwise
Available Color Processing Methods	Workstation_ Dependent, Bitwise	Workstation_ Dependent, Bitwise	Workstation_ Dependent, Bitwise	Workstation_ Dependent, Bitwise	Workstation_ Dependent, Bitwise	Workstation_ Dependent, Bitwise
Available Rendering Color Models	RGB	RGB	RGB	RGB	RGB	RGB

Notes:

1. This visual class is supported only on DWA and XSOFT workstations.
2. The display color table will have the same number of entries as the X colormap if XOR echo method is used or an overlay bit plane is available. If bit plane echo is used, the display color table will have half the number of entries as the X colormap.
3. If the application specifies the XNOCLRMP PROCOPT (see PROCOPT (Processing Options)) to suppress the creation of a colormap by the graPHIGS API, the display color table is not modifiable through the graPHIGS API. Its content must be modified through the X programming interface.
4. *Identity Index Map* means that the pixel values produced by the graPHIGS API rendering pipeline will be equal to the color index in the WSL or structure elements as specified by the application.
5. The default will be the display color table if it is modifiable, 0 otherwise.

The following discussion about colormap allocation assumes that the visual class of the window has a corresponding colormap that is modifiable (GrayScale, PseudoColor or DirectColor). The application programmer has a choice as to whether the application or the graPHIGS API allocates the colormap that is to be used for the window that the graPHIGS API workstation will use. If the application does not pass in a window identifier through a PROCOPT then the graPHIGS API automatically allocates a colormap as well as the window. The colormap attribute of the window is then set to that of the allocated colormap. When the application passes in the window identifier through a PROCOPT, the graPHIGS API will not allocate and assign a colormap if the `XNOCLRMP PROCOPT` is specified (see XNOCLRMP (Do Not Create an X Color Map)). The colormap attribute of the window that was passed in will not be modified.

Only when the graPHIGS API allocates the colormap can it be modified through the **GPCR** and **GPXCR** subroutine calls. The allocated colormap always corresponds to the display color table. When the graPHIGS API does not allocate the colormap, the application must set the colormap through the X programming interface. It cannot be modified through the graPHIGS API programming interface.

The bit plane echo method may require special treatment by the application. When a bit plane echo method is provided, the graPHIGS API will draw all echoes in the most significant bit plane of each frame buffer component. The echoes will be erased and drawn independently of the content of the other bit planes. To produce a constant echo color, the upper half of the X colormap will be loaded with the echo color and only the lower half of the colormap will be accessible to the graPHIGS API application. Bit plane echo will not be granted by the graPHIGS API if the X visual class is StaticGray, StaticColor or TrueColor since the colormap is not modifiable. In this case, the echo method will default to XOR or uses an overlay bit plane if available. After successful creation of a graPHIGS API workstation, the application should check whether bit plane echo method is being used or not. If it is, and the application has created the colormap, then the application is responsible for loading the upper half of the colormap with the echo color.

So far, the X resources that the graPHIGS API uses have been discussed as well as how the application can affect the allocation of these resources. In the following paragraphs, we will discuss differences between the X concepts and PHIGS concepts and how the two might coexist using the mechanisms described above.

One of the fundamental differences between X and PHIGS is that X is primarily concerned with independent pixel values that the application will use. In contrast, PHIGS, PHIGS PLUS and graPHIGS API applications are more concerned with color or ranges of colors. In the latter case, it is left entirely up to the implementation as to what pixel values are generated and how the physical colormap is used. Ranges of color are introduced in PHIGS PLUS and the graPHIGS API to support depth cueing, lighting, shading, and direct color specification. In many cases, the implementation is most efficient if it can map ranges of color to ranges of pixel values. In the graPHIGS API, additional functionality has been introduced to give the application direct control over the pixel values that are generated. This was done to support applications that need to create special effects, such as the simulation of overlay planes or to implement some form of display priority that is independent of the traversal order. Even the applications that use the direct control over the pixel values still need color ranges for lighting, shading, etc.

A large percentage of PHIGS and PHIGS PLUS applications could be supported if the implementation maps the specified color values to the closest available on the device. Since the relative intensity of color can be approximated on the gray visual classes, an application will be most portable across different devices and visual classes if it relies on the implementation to map the specified colors to the closest available. However, the quality of the display image will vary depending on the capabilities of the hardware. This class of applications, which will be termed *true color*, does not need to manipulate the graPHIGS API color processing parameters or the content of the display color table once they are initialized. The TrueColor or DirectColor visual classes would produce the best results for this type of usage.

Note: The graPHIGS API does not support the closest color approximation on StaticGray and StaticColor visual classes.

Some applications may require more accurate color approximation on some visual classes than can be achieved through closest color approximation since the accuracy or quality decreases as the number of simultaneously displayable colors decreases. For example, on an 8 bit plane system (256 colors), closest color approximation is only acceptable if dithering is supported. Otherwise, the application needs more direct control over the generation of pixel values to optimize the usage of color. The application could choose to allow only 4 object colors. In this case, it could use 64 color table entries to represent different intensities of each of the 4 colors.

If an application requires better color fidelity or any of the special effects described above, it must take a more active role in how the pixel values are generated and how the pixel values get mapped to a color on the monitor. To do this, it must manipulate the color processing parameters and content of the display color table. This class of application will be termed *direct pixel control*. For visual classes that have read only colormaps, it is almost impossible to support this class of application since the mapping between pixel values and the resulting color is fixed. Fortunately, these visuals are becoming less common.

In order to share colors and to avoid the ″technicolor effect″, either the colormap must be static (automatic sharing) or the application must explicitly control the pixel values that are produced by the graPHIGS API. In the former case, it has already been stated that it is difficult to support applications that require the special effects described above as well as to optimize the fidelity of the color approximation. Therefore, the best way to share the X color resources is for the application to take an active role in controlling the generation of pixel values.

Since the graPHIGS API does not understand how the application will use the color facilities, it will allocate an entire colormap instead of attempting to allocate specific colormap entries from X. This implies that the ″technicolor effect″ will most likely result when the graPHIGS API allocates the colormap. If the application wishes to minimize this effect, then it should suppress the creation of the colormap by specifying the `XNOCLRMP PROCOPT` (see XNOCLRMP (Do Not Create an X Color Map)) and explicitly manipulate the color processing parameters to control the pixel values that get generated.

To facilitate this usage, the default color processing parameters and content of the rendering color table will be set to map color indexes directly to pixel values. The resulting pixel value will be the same as the specified index. This will make it easier to share pixel values and colors with X. Notice that this initial setup is not appropriate for using direct color specification unless the application calculates the color components based on the desired pixel value and color processing parameters. If depth cueing, lighting, or shading are enabled, the pixel values generated will not necessarily match the specified color index since the rendering pipeline will modify them.

The following list summarizes the guidelines for applications that fall into the *direct pixel control* class. This may be due to the special effects which are desired or because the application has limited color requirements and wants to avoid the ″technicolor effect.″

- If the application does not need color ranges, such as for depth cueing or shading, then it can implement the following to minimize the technicolor effect:
  1. Create the window through X, specifying the default visual class.
  2. Create the graPHIGS API workstation, passing in the window identifier and suppressing colormap creation.
  3. As colors are needed, perform calls to the X programming interface to find the closest color available in the default map or allocate a colormap entry from the default colormap.
  4. Specify all attribute colors through the graPHIGS API as indices using the pixel values allocated through X.

  This technique works well for all visual classes including StaticGray and StaticColor.

  Notice that for bit plane echo, an entire colormap probably needs to be allocated since half of the colormap should be loaded with the echo color. This technique works well only when echo is not required or when XOR echo method is used or when overlay bit plane method is used.

- The previous technique could also be used if limited ranges of color are required. The steps to accomplish this would be:
  1. Create the window through X, specifying the default visual class.
  2. Create the graPHIGS API workstation, passing in the window identifier and suppressing colormap creation.
  3. Using the XAllocColorCells function in the X programming interface, attempt to allocate a range of color cells from the X server. If the allocation from the default colormap fails, allocate a new colormap and try again. If a new colormap is allocated, the colormap attribute of the window would have to be modified.
  4. Set the graPHIGS API color processing parameters to generate pixels in the range that were allocated. For example, if four contiguous planes are requested from XAllocColorCells, it might return a pixel value of 0x80 and a mask of 0x01, 0x02, 0x04, and 0x08. In this case, setting a color processing representation to bitwise (0,4,0) with a pad of 0x80, would provide 16 pixel values corresponding to 16 quantization levels of the green color component.
  5. Colors could then be specified as direct pixel values through color indices if the rendering color table has not been modified. The content of the rendering color table could be changed and direct color used as long as the color processing parameters are set to generate pixels in the allocated ranges.

  Notice that for bit plane echo, an entire colormap probably needs to be allocated since half of the colormap should be loaded with the echo color. This technique works well only when echo is not required or when XOR echo method is used or when overlay bit plane method is used.
- This technique is a slight modification of the previous two. Instead of using the default map, the application would allocate a new colormap and copy the default colormap to it. The application could then set or use any entry without allocating it from X resulting in more flexibility. If the application then started using colormap entries at the top first, it would minimize the impact on other windows which typically share entries at the bottom of the colormap.
- The simplest technique is to let the graPHIGS API allocate a colormap, realizing that users may encounter the ″technicolor effect.″

The following list summarizes the guidelines for applications that fall into the *true color* class and do not want to worry about the pixel values that get generated:

- If the visual class is TrueColor, the application can specify the desired color through either the rendering color table or direct color elements freely.
- For visual class GrayScale, PseudoColor, and DirectColor, the application could allocate the colormap or allocate a subset of a colormap from X depending on its needs. Whether the application allocated the colormap entries or not, it would set the graPHIGS API color processing parameters to include a few bits from each color component in the resulting pixel value, and it would load the colormap with corresponding color ramps (graduated color components). The application could then specify the desired color through either the rendering color table or direct color elements freely. For a direct color visual class, the colormap is usually loaded with ramps of red, green, and blue. This colormap content is used frequently and should be sharable with other X clients. There are several standard colormaps defined by X that might be appropriate for this class of application.
- The visuals StaticGray, StaticColor cannot be used for this class of application since the graPHIGS API does not currently provide an appropriate color approximation method.

A few final notes on color:

- Color table animation cannot be performed on a visual class with a read only color table.
- Pixel primitives cannot be displayed on visual classes with read only colormaps without modifying the pixel values prior to display since the colormap cannot be changed.

Additional Notes for DWA Adapters

Note: The following information applies to all DWA Adapters *except* the POWER GT4 Family and the POWER GTO.

Using the visual associated with the window, the graPHIGS API supports creating graPHIGS windows as 8 bit Indexed, 24 bit TrueColor, or 24 bit DirectColor. Additionally, the graPHIGS window MUST be created in the color planes and for the best performance, it is recommended that the X window (root window when X is started) be created in the overlay planes. In support of echoes, the graPHIGS API will create a child window in the overlay planes.

The graPHIGS window may be created as follows:

- By the application who then passes the window id to graPHIGS via the XWINDID procopt (see XWINDID (X Window Identifier) for additional information).
- By the graPHIGS API on behalf of the application when the workstation is created.

The visual associated with the graPHIGS window being created is selected as follows:

- Specified by the application from the supported visuals for the color planes via the XGetVisualInfo function. It is then passed to the XCreateWindow function to create the graPHIGS window in the color planes.

  In this case, start X in the overlay planes as follows:

  ```
  For POWER GXT255P and POWER GXT250P:
          xinit -- -x dbe
  For all other DWA Adapters (except the POWER GT4 Family and the POWER GTO):
          xinit -- -x dbe -x abx
  ```

  and within your application, select the desired visual and pass it to the XCreateWindow function.

  This method allows you to start X in the overlay planes while the graPHIGS API is running in the color planes, giving you the best performance. Windows in different planes will cause fewer graPHIGS redraws, since there will be fewer exposure events. If your application is NOT currently written to select a visual, this will require a change to your application.

  If your system administrator has installed the sample programs, there will be a sample program and README file in the **/usr/lpp/graPHIGS/samples/windows** directory showing how an application selects the desired visual and creates a graPHIGS window.
- Defaults to using the visual associated with the root window (the window created when X was started). This will occur if you do not pass a selected visual to the XCreateWindow function.

  In this case, you start X in the color planes and select one of the following three graPHIGS API supported frame buffer configurations for the root window:

  – The 8 bit visual:

  ```
  For POWER GXT255P and POWER GXT250P:
            xinit -- -x dbe -layer 0

  For all other DWA Adapters (except the POWER GT4 Family and the POWER GTO):
            xinit -- -x dbe -x abx -layer 0
  ```

  – The 24 bit DirectColor visual:

  ```
  For POWER GXT255P:
         xinit -- -x dbe -d 24 -cc DirectColor

  For all other DWA Adapters (except the POWER GT4 Family and the POWER GTO):
         xinit -- -x dbe -x abx -layer 0 -d 24 -cc DirectColor
  ```

  – The 24 bit TrueColor visual:

```
For POWER GXT255P:
       xinit -- -x dbe -d 24 -cc TrueColor

For all other DWA Adapters (except the POWER GT4 Family and the POWER GTO):
       xinit -- -x dbe -x abx -layer 0 -d 24 -cc TrueColor
```

This method produces the desired results and requires no change to your application, but it does not give you the best performance. If the X window is manipulated, causing an exposure event, more graPHIGS API redraws may occur since X and graPHIGS windows are both created in the color planes.

Additionally, the graPHIGS API uses an overlay window for echoes that it creates as a child of the graPHIGS window. Since this overlay window has a transparent background pixel, the graPHIGS window passed in MUST be in the base planes. Furthermore, if the graPHIGS window is passed in from the application, and is NOT the top level window, the application must add a Window Manager Colormap Install property to the application's top level window for the graPHIGS created overlay window in order for the overlay window's colormap to be installed when the graPHIGS window gets focus.

The graPHIGS API and X Input Relationship

The graPHIGS API has been designed to be consistent with the behavior of other applications sharing the same input devices, namely the keyboard and the mouse. The graPHIGS API will never grab these devices but it will expect the server to direct input to the window when the window has the focus. The keys on the keyboard will be interpreted according to the current keycode to keysym mapping. X maintains a device independent mapping between the scancodes generated from the keyboard and the meaning of a key. For example, the key top with the number '1' will generate a keycode that will be mapped to the keysym for number one. The graPHIGS API will interpret the keyboard events via keysyms. Therefore, if you change the keycode to keysym mapping via the `xmodmap` utility, the graPHIGS API will automatically interpret the new mapping. Typically, the character on the key will generate the identical keysym. This area gets a little more difficult when you consider the control keys. The following list describes some of the behavior that a particular server may display:

- A two button mouse works the following way under X windows. You will notice that button two is not the right mouse button. Remember this when you run your application, otherwise you are likely to think that there is a problem.
  - Press Button 1 - generates an event indicating Button 1 pressed
  - Press Button 2 - generates an event indicating Button 3 pressed
  - Press Button 1 and 2 together - generates an event indicating Button 2 pressed.
- The available button and PFKey counts in the actual WDT will be for the maximum number of buttons and PFKeys that the X workstation will support. There may be fewer PFKeys on your keyboard or buttons on your mouse.

(Ref #1.) On platforms that support the Lighted Program Function Keys (LPFKs) and Dial X server extensions, the graPHIGS API uses these extensions to access the lighted keys and dials (see *AIX 5L Version 5.3 AIXwindows Programming Guide*). When the graPHIGS API window receives input focus, graPHIGS API assumes itself to be the owner of the LPFKs and Dials and attempts to set the attributes of the devices (the lights mask and dial resolution) as needed.

graPHIGS applications can no longer be run on an operating system using X11R4 and displayed on an operating system using X11R5 because of X input extension compatibility issues. (Please see the /usr/lpp/X11/README for further explanation.) The recommended way to avert this problem is to open a graPHIGS remote nucleus on the same system where it is desired to have the graphical output displayed.

For applications that also use these input device extensions independent of the graPHIGS API, a contention problem can result when the applications also attempt to set the attributes of the devices. To avoid this contention, the application can issue the Set Physical Device Mode (**GPPDMO**) graPHIGS API subroutine to disable the physical button device #1 (LPFKs) and all scalar devices (Dials). When the physical devices are disabled, the graPHIGS API will not attempt to set the device attributes.

When using physical device emulation on the X workstation, you may find it useful to translate the X windows coordinate system to the physical vector device value ranges. The transformation requires the use of the mapped display surface (see the Inquire Mapped Display Surface [**GPQMDS**] subroutine), the value ranges for the vector device (see the Inquire Physical Device Characteristics [**GPQPDC**] subroutine), and the size of the X window (obtained from the X windows interface). The elements of this mapping are described in Window Mapping and Resize.

The algorithm to achieve a mapping from *x* coordinate position data to vector device value ranges is as follows:

W	X Window Geometry.
MDS	Mapped Display Surface Components from the Inquire Mapped Display Surface (**GPQMDS**) subroutine.
VR	Value Range Descriptor from the Inquire Physical Device characteristics (**GPQPDC**) subroutine.
V	Value passed to the Emulate Physical Device (**GPEPD**) subroutine.
P	Position data in *x* coordinates.

1. Clip to the Mapped Display Surface, the display surface is centered in the X Window. The clipping rectangle is computed as follows:

   ```
   X components: (Wwidth - MDSwidth / 2)
                 (Wwidth - MDSwidth / 2 +MDSwidth)

   Y components: (Wheight - MDSheight / 2)
                 (Wheight - MDSheight / 2 +MDSwidth)
   ```

2. Scale from the Mapped Display Surface (in address units) to the vector device value ranges:

   ```
   Scale Factors: (VRxhigh - VRxlow) / MDSwidth
   ```

   Note: Only one scale factor is needed since the value ranges are maintained in the same aspect ratio as the display surface.

3. Compute a value range from an X position data:

   ```
   Vx= (Px - (Wheight - MDSwidth / 2)) * Scale Factor

   Vy= (Wheight - (Wheight - MDSheight / 2))  * Scale Factor
   ```

How the graPHIGS API Uses X Window System Cursors

Whenever a graPHIGS API ″pointing″ input device is active (pick, locator, or stroke), the graPHIGS API changes the shape of the X pointing cursor when it enters the graPHIGS API window, and restores the shape of the cursor to its previous shape when it leaves. The shape of the cursor in the graPHIGS API window depends on the echo area where it is positioned, and which input devices are active.

Normally, in non-X graPHIGS API environments, when no pointing input device is active, *no* pointing cursor is displayed. However, in the X environment, the pointing cursor should never be hidden from the user, who should always be able to locate the pointing cursor as he moves it from window to window. This practice is part of being a well-behaved X client program. The shape of the pointing cursor in a graPHIGS API window with no pointing input devices active is the shape of the graPHIGS API window's parent's cursor. In applications where the graPHIGS API window is the top level window (direct descendant of the root, or background window), the parent's cursor will usually be the root window cursor. The root window's cursor will be displayed in the graPHIGS API window when no pointing input devices are active. Note that some window managers, such as the OSF/Motif window manager (mwm), will re-parent a window and supply a different parent cursor.

Fixed cursor type -1 (cross hair) extends to the limits of the graPHIGS API window. If the hardware cross hair cursor is used (either by defining the gPHWCURS environment variable or via the HWCURS PROCOPT), the cursor extends to the limit of the display. This is currently a limitation that exists in the X cursor extension.

How the graPHIGS API Handles X Window System Errors

The X Window System handles error conditions by generating an X error event, which is queued back to the X client program. The X error event contains information about the X request that caused the error condition. The default action for most X clients is to simply print the error information and then terminate. The graPHIGS API overrides this default action by intercepting this error event and signaling a graPHIGS API error to the graPHIGS API application. This method allows the graPHIGS API application to detect the error and close down in an orderly fashion, preserving application status and data, if desired. By preventing the termination of the X client program, which is the graPHIGS API nucleus in this case, a remote nucleus may continue to execute if one of the sessions using the nucleus experiences an X terminating error condition.

The graPHIGS API nucleus may experience X error conditions for three reasons: resource shortages, internal programming errors, and communication errors. All requests for X resources made by the graPHIGS API nucleus are made during graPHIGS API Open Workstation processing. X resources are simply graphic objects that the X server manipulates, such as Pixmaps or Cursors. Therefore, any resource shortage conditions will be detected during Open Workstation processing and will result in a failure of the opening of the workstation. Resource shortage conditions may be caused by:

1. Using an X server that has a very limited set of resources. The graPHIGS API is not especially resource intensive, so all but the most limited servers should have enough resources.
2. Running a large number of applications that use up the X resources. This type of shortage can be corrected by removing some of the applications that are holding the resources.

X errors may also be caused by a problem internal to the graPHIGS API nucleus. The error information returned by X is formatted into a graPHIGS API error message. This error information will give more details about the error condition.

X communication errors are most likely to happen in a networked environment where the X client (graPHIGS API nucleus) and the X server are running on different network nodes. If there is a break or other problem with the network connection, the link between client and server fails, causing the communication error. X communication errors may also occur if a client window is terminated abnormally, such as by using a window manager to close a window. The graPHIGS API traps these errors and queues an error notification to the graPHIGS API application. This is the default behavior, which may be modified (see Window Deletion for more details).

Editing in Quick Update Mode

Quick update operations fall into two categories, insertions and deletions. Even when editing structures in replace mode, the operation consists of a deletion followed by an insertion.

Insertion

Insertions are done by adding structure elements to a structure in either insert or replace mode, or by using the Copy Structure function.

The inserted primitives are drawn on the screen using the attribute and the traversal state that is in effect at the point of insertion. The traversal state is the collection of all the current values of the various attributes and transforms that are used to draw primitives on the screen. This traversal state is achieved by *pseudo-traversing* the structures up to the point of insertion. Pseudo-traversal processes the structure elements as if to draw them but does not send them to the screen. The resultant traversal state reflects the correct data although the display contents remain unchanged except for the inserted primitives.

For example, a polyline primitive inserted after a polyline color index attribute, is drawn with the color specified in the polyline color index structure element. More importantly, it is drawn in the correct position dictated by any preceding modeling transforms. In general, inserting an attribute structure element affects certain primitives that follow the inserted attribute. This can be an expensive operation because the redraw may include many primitives and must continue to the end of the structure, or until the same attribute structure element is encountered in the structure. This redraw of affected primitives is called *attribute propagation*.

Inserting an attribute structure element in a structure can cause a large part of the structure to be redrawn. This can be very time-consuming and defeat the purpose of quick update. Therefore, no attributes are propagated except color. Color is a far more common insertion attribute than line type or line width, either of which could have undesired results such as wide holes or a cluttered screen, if inserted in quick update mode.

The following structure provides an example:

```
GPOPST(1);
GPPLCI(3);
GPPL3(...);
GPPLCI(2);
GPPL3(...)
GPCLST();
```

If the first **GPPLCI** were replaced in quick update mode, then the first **GPPL3** primitive would have to be drawn, but not the second.

If an execute structure element is encountered in the attribute propagation block, it is executed and drawn normally. Even if a color attribute structure element is contained in the called structure, attribute propagation continues after the execute structure because the return from the called structure cancels any effect of any color attributes in the called structure.

If any attributes other than color are inserted, then quick update mode is aborted. But the primitives that would be affected by these attributes are not always redrawn. The result may be incorrect display update.

The following two examples illustrate how attribute propagation can affect display in unexpected ways, depending on implementation:

1. Primitives in the attribute propagation range that are NOT affected by inserted colors may or may not be drawn. Theoretically, they can be skipped, but the implementation may choose to draw them.
2. Non-color attributes that are inserted following a color attribute in the same insertion operation may be processed differently. One implementation may choose to propagate these attributes while another may not.

Note that the first effect can influence the second. Inserting structure elements that have a global effect, such as modeling transforms, and class names, causes quick update to be aborted and the screen to be redrawn.

Deletion

Deleted primitives are simply *undrawn* in the background color. The background color is defined as the shield color if the view has a shield, or black if there is no shield. Other primitives that overlap the deleted primitives may be left with holes that are not repaired.

The deletion of an attribute does not affect the display. Deleted attributes are not propagated because the workstation would have to backtrack to find the previous usage of the attribute in order to determine its value prior to the deletion point, or pseudo-traverse from the start to the deletion point. It would then have to forward propagate, drawing the affected primitives with the previous attribute value. This operation is

considered too expensive to be quick. Therefore, deleted attributes do not cancel quick update but do not change the contents of the screen. No special provision is made for the deletion of color attributes as is done for the insertion of color attributes.

In many cases, the deletion of an attribute is immediately followed by an insertion of the same attribute. This can happen, for example, if a structure edit is drawn in replace mode to change the value of a color attribute. In this case, it is appropriate to propagate only inserted attributes to avoid propagating attributes twice.

The treatment of deleted structure elements other than primitives is extremely implementation dependent.

The graPHIGS Programming Interface: Writing Applications contains information on how to select modification modes.

The XSOFT Workstation

Overview

The graPHIGS XSOFT workstation is a complete implementation of the graPHIGS API in software. It can replace the graphics sub-system by performing all graphics operations on the main CPU or workstation.

Traditionally, interactive computer graphics implementations such as graPHIGS have been implemented with the power of hardware assist. This hardware assist was usually made available in the form of a graphics sub-system consisting of general purpose processors, custom or semi-custom VLSI rasterizers and a frame buffer. For example, the IBM 5080 and IBM POWER GTO are graphics subsystems that connect to a mainframe or workstation respectively. The graphics subsystem is attached to the main CPU or workstation and was necessary because the general purpose processors of the main CPU were not capable of driving the graphics performance at interactive speeds. However, RISC processors have evolved to the point of being able to partly or completely replace the graphics subsystems and drive the computer graphics at interactive speeds.

Understanding XSOFT

At initialization **GPCRWS**/**GPOPWS**, the workstation allocates, in virtual memory, the virtual frame buffer (rendering targets) and virtual Z-buffer (rendering resources) based on the initial size of the workstation display surface. The **GPQMDS** call returns the size of the display surface as it is mapped into the X-window. Should the display surface change size (via a window resize operation), the rendering targets and the rendering resources are reallocated based on the new size. These virtual memory areas are freed when the workstation is closed (**GPCLWS**).

During an implicit (**GPUPWS**), explicit (ETC operation), or simulated (quick update) update, the affected structure elements undergo geometry processing and rasterization into these virtual resources. At the end of the update, the displayed rendering target in virtual memory will be transferred to the X-window.

The XSOFT workstation uses the `GP-MIT-SHM` extension to X to make this process more efficient. The X Windowing System has a rather small limit on the protocol buffer size. This restriction means that the XSOFT rendering target would be transferred to the X server in small "chunks", which impacts the visual quality of the update as well as the interactive performance. The `GP-MIT-SHM` extension bypasses the client-server protocol by transferring the rendering target in one piece through shared memory.

General Information

The XSOFT workstation includes the following:

- Full functionality

The graPHIGS XSOFT workstation supports the full functionality of the graPHIGS API. This includes HLHSR, lighting and shading, depth-cueing, transparency, blending and anti-aliasing in addition to basic graphic functions. Previously, this functionality was not available across the varied domain of IBM workstations, processors, and graphics adapters.

- Processor independence

  The graPHIGS XSOFT workstation runs on all IBM workstations. This includes the IBM RS/6000 processor family.

- Adapter independence

  The graPHIGS XSOFT workstation will run on any 2-D or 3-D 8-bit or 24-bit adapter that supports its own graphics sub-system or does not have one available.

- Performance
  - Scalability

    The graPHIGS XSOFT workstation graphics performance can be directly correlated to the IBM RS/6000. As the workstation's processor specifications improve, the performance of the graPHIGS XSOFT workstation will improve.

Configuring a graphics workstation for XSOFT

There are several items that need to be evaluated when considering the graPHIGS XSOFT workstation and recommending specific configurations.

- Processor requirements
- Memory requirements. The memory requirements generally fall into several areas:
  - Disk space requirements for the actual graphics XSOFT library. The graPHIGS XSOFT workstation shared library is approximately 12 Mbytes in size. This memory is essential for obtaining the high performance of the graPHIGS XSOFT workstation.
  - Paging space requirements
  - Virtual memory
- Graphics adapters

  The graPHIGS XSOFT workstation will run on any 8-bit or 24-bit graphics adapter. However, since the graPHIGS XSOFT workstation is dependent on the blt performance of the workstation, the faster the blt, the better. The performance of bit blt operation on the High Speed Graphics Subsystem (GTO) adversely effects the performance of the XSOFT workstation. This platform may not provide the interactive performance necessary for production use. If high performance graphics at a low cost is a concern, we recommend the use of 8-bit graphics adapters.

  For assistance in configuring a system for a specific need, contact an IBM Customer Representative.

Starting the X Server

The XSOFT workstation device driver uses (when available) the graPHIGS Shared Memory Image (`GP-MIT-SHM`) extension to X to provide a very efficient means of moving the image of the XSOFT workstation to the X server display. The `GP-MIT-SHM` extension is very similar to the sample Shared Memory Image extension (`MIT-SHM`) which comes from MIT.

The `GP-MIT-SHM` extension is only available to the XSOFT workstation device driver when the graPHIGS nucleus is executing on the same machine as the X server and the X server has the `GP-MIT-SHM` extension loaded.

To load the `GP-MIT-SHM` extension, start the X server with the `-x gpshm` command line option:

- For a system with X11R5 installed: `xinit – -x gpshm`
- For a system with X11R4 installed: `xinit -x gpshm`

Alternately, to automatically load the `GP-MIT-SHM` extension, add the following line to the **static_ext** file in the **/usr/lpp/X11/bin** directory:

```
gpshm /usr/lpp/graPHIGS/bin/loadgpshm
```

X Stations And Distributed X-Windows

Whenever the graPHIGS nucleus and the X server are not running on the same machine, for example X Stations and other distributed X-windows environments, the XSOFT workstation cannot take advantage of the `GP-MIT-SHM` extension. Running without this extension limits the interactive performance of the XSOFT workstation. The `GP-MIT-SHM` extension can be used in the distributed graPHIGS configuration when the graPHIGS shell and nucleus are distributed but the X server is executing on the same machine as the nucleus. Distributing the graPHIGS API in this manner does not limit the performance of the XSOFT workstation.

The XSOFT workstation supports lighting and interpolated features not supported on the X Stations by the X workstation type. Occasional or ″view only″ users of an application may find the performance acceptable. Applications with low frame rates, minimal user interactions, or user interactions implemented entirely independent of the graPHIGS API may also find this configuration acceptable.

Applications can use the X workstation type to provide interactive performance in the distributed environment and a second XSOFT workstation to provide a more advanced rendering. This is possible through the graPHIGS ability to share Structure Store among more than one workstation. For more information on this capability, see Advanced Concepts.

Special Notes about Color

3D graphics applications have special needs for color processing. The XSOFT workstation creates a private color table for use in the workstation's X-window. The use of a private color table can cause the ″false color″ effect on devices that support only one simultaneous colormap. The effect is caused by the fact that the device can only display one colormap at a time. Therefore, when the focus is on the graPHIGS window, all other windows are displayed with different colors. Although this problem is not limited to the XSOFT workstation, the additional color demands of lighting and interpolated shading may make this problem more noticeable. For more information about this, see Interaction of X and graPHIGS API Color Resources.

Lighting and interpolated shading techniques often require many colors to achieve the desired effect. Since 8-bit devices can only display 256 colors, the XSOFT workstation will dither colors on these devices. Dithering is a technique where pixels of different color are placed adjacent to one another to give the appearance of a third color. Image quality is vastly improved using this technique, although the individual colors used in the dither can sometimes be noticed as a slight pattern in filled areas. Dithering is applied to fill area primitives (i.e. polygons and triangles, etc.) and to lines when the color along the line is interpolated (i.e. Polyline set with data or depth cued lines). Dithering is not applied to constant color lines, text, markers, view shields, and view borders.

The 6090 Workstation

Shading (SHP) and Expanded Pixel Memory (EPM) are optional features on the 6090 workstation. Your use of these optional features determines whether some functions are supported by the 6090 workstation.

If you go in and out of setup while your application is running, an implicit update of the screen will occur.

The actual primary character set is determined by the language setup as follows:

- If you specify character set 1-5 or 7 in setup, then the primary character set is 8 (Multi-Language).
- If you specify character set 6 in setup, then the primary character set is 6 (Katakana).
- Specifying character set 8 in setup associates U.S. English with a Kanji keyboard. The primary character set is 1 and the available input character sets are 1, 8, and 128.

Workstation Configuration

The pre-select highlight, line on line, cursor shape, and color setup options are ignored.

Transformation Matrixes

You must put all transformation matrixes in your application in the following format:

```
| R11 R12 R13 | 0.0 |
| R21 R22 R23 | 0.0 |
| R31 R32 R33 | 0.0 |
|-------------|-----|
|  S1  S2  S3 | 1.0 |
```

The values in the fourth column, (0.0, 0.0, 0.0, and 1.0) are always used regardless of what you specify in your application program.

Temporary Views

Temporary views are not supported.

View Mapping

If the projection reference point is between the near and far clip planes, the projection type is changed to `PARALLEL` and an error is generated.

The 5080 Workstation

General Information

The IBM 5080 Graphics System uses 16-bit integers for the coordinates of figures to be drawn. Since graPHIGS API applications pass 32-bit floating-point coordinate parameters, a mapping must be done to represent the coordinates received from the application in the correct format for the 5080. This mapping ("normalization") can result in distortion when the floating-point format cannot be mapped well into the available integer range. For example, the extents of the data might be very small when compared to the distance of the data from the origin of the coordinate system, or one extent of the data might be very small when compared to other extents of the same data. To minimize these effects of normalization, center your data about the coordinate system origin whenever possible.

Note: The maximum number of structure elements in a single structure that the 5080 can display is 32,767. When an update to the workstation is processed and the resultant element count exceeds this limit, the update will be ignored and will result in an error.

Workstation Configuration

The pre-select highlighting and line on line setup options must be set to off. Cursor shape and color setup options are ignored.

Display Models

The `DISPLMOD PROCOPT` is used to identify the use of different 5081 displays. The default is the 19-inch display (5080-19). If you are using the 16-inch display (5081-16), specify `5081-16` as the `PROCOPT` value (see PROCOPT (Processing Options)). If you are using the 23-inch display (6091-23), specify `5081-23` on the `PROCOPT` value. All other values that start with the characters `5081-` are treated as the 19-inch display.

Class Set

The class names which you can specify are limited to the range 0 through 255. Any class name encountered which is greater than 255 or less than 0 is ignored.

Transformation Matrixes

All transformation matrixes specified by an application must have the following format:

```
| R11 R12 R13 | 0.0 |
| R21 R22 R23 | 0.0 |
| R31 R32 R33 | 0.0 |
|-------------|-----|
|  S1  S2  S3 | 1.0 |
```

The values in the fourth column, (0.0, 0.0, 0.0, and 1.0), are always used regardless of what was specified by the application.

Temporary views

Temporary views are not supported.

View Mapping

If the projection reference point is between the near and far clip planes, the projection type is changed to `PARALLEL` and an error is generated.

The GDDM Workstation

Class Set

The class names which you may specify are limited to the range 0 through 255. Any class name encountered which is greater than 255 or less than 0 is ignored.

View Mapping

If the projection reference point is between the near and far clip planes, the projection type is changed to `PARALLEL` and an error is generated.

The GDF Workstation

General Information

The Graphics Data Format (GDF) workstation provides a means of capturing and storing data produced by applications in a form which can be processed by other programs. This form consists of a sequence of graphics orders and their parameters.

The GDF workstation is an output-only workstation which is not associated with a physical graphics device. The GDF support assumes that the target display device has the characteristics of the 3270-PC/GX. When an application program interacts with a GDF workstation, images defined by the application are converted into GDF display lists and stored in files. Subsequently, these files can be processed by a program such as IBM Color Plotter Support for GDDM Graphics Data Format (CPS), which can plot the file to the IBM family of plotters.

Note: The CPS programs are supplied with the graPHIGS API

The application is free to select a connection identifier which is used to derive the names of the files created by the GDF workstation.

In the VM and MVS environments, a valid connection identifier consists of letters, numbers or underscores. The file name is created by taking the first five characters of the connection identifier. If any of these characters are blanks, they are replaced with a fill character. Lower case characters are converted to upper case. An 'X' replaces a blank first-position character, and a '0' replaces any other blanks between the second and fifth positions. The GDF workstation then concatenates to the end of the file name a three-digit update number in the range 001-999.

In the VM and MVS environments, the logical record length of the file must be 400 bytes and it must be fixed record format.

On VM/SP, the resulting file name is used as the file name of the generated file; the file type is always ADMGDF. On MVS, the resulting file name is used as the member name in a partitioned data set that has been allocated using the DDNAME of ADMGDF.

In the operating system, the connection identifier must be a valid file name which may include a full or partial path name. The GDF workstation strips the optional path name from the connection identifier and takes the first five characters from the resulting file name. Upper and lower case characters are valid. If any of these characters are blanks, they are replaced with a fill character. An 'X' replaces a blank first-position character, and a '0' replaces any other blanks between the second and fifth positions. The GDF workstation then concatenates to the end of the file name a three-digit update number in the range 001-999, and suffixes the extension. The file is then created in the current directory or in the directory specified by the path name in the connection identifier.

A new file is generated each time the GDF workstation is updated. When a new file is created, the number represented by the last three characters of the file name is increased by one. If this number exceeds 999, it is reset to 001. A new file will overwrite an existing file which has the same name. At this point then, the first file created during the session will be overwritten. Therefore, a maximum of 999 different files can be generated during a session.

If, for example, an application specifies 'ABC' as the connection identifier for a GDF workstation, the files created on successive workstation updates will be:

Table 8. Filename Examples

UPDATE NUMBER	FILENAME
1	ABC00001
2	ABC00002
3	ABC00003
.	.
.	.
999	ABC00999
1000	ABC00001
1001	ABC00002

Class Set

The class names which you may specify are limited to the range 0 through 255. Any class name encountered which is greater than 255 or less than 0 is ignored.

GDF Conversion Utility

The CVTGDF utility converts a file which is in GDF format (a file generated by a GDF workstation running on an operating system platform) to ADMGDF format, which is the format produced by a GDF workstation running in the VM/SP or MVS environment. This utility resides in the directory **/usr/bin**.

To run the utility the syntax is:

```
cvtgdf /dir/fn.gdf
        |  |   |
        |  |   must be the file extension
```

```
|  the file name
specifies the directory in which the file resides
(if the file resides in the current
directory, this information may be omitted)
```

The converted output is placed in the */dir/fn.*cfgdf file.

When uploading the file from the operating system to the VM or MVS host (using, for example, the 3278 emulation program), a record length of 400 and a fixed record format must be specified. On MVS, the file must be placed in a dataset which has been allocated with the DDNAME 'GDF'.

View Mapping

If the projection reference point is between the near and far clip planes, the projection type is changed to PARALLEL and an error is generated.

The CGM Workstation

General Information

The Computer Graphics Metafile (CGM) workstation provides a means for the application to store graphical information about a picture in a file. The file format consists of a set of elements encoded in CGM binary format according to ANSI standards. When an application program interacts with a CGM workstation, images defined by the application are converted into a list of CGM elements and are stored in a single file. The CGM workstation is an output-only workstation which is not associated with a physical graphics device. The graPHIGS API CGM support assumes that the target display device has the characteristics of the 3270-PC/GX. See General Output Facilities for details.

The filename is derived from the connection identifier.

On the operating system, the filetype is CONNID.cgm. For example, if an application specified TSTALL as the connection identifier on a CGM Workstation, the application creates the file: TSTALL.cgm. Upper and lower case characters are allowed. If the connection identifier consists of all blanks, the output CGM file is IBMCGM.cgm.

On MVS, the user must have allocated a *sequential* dataset with the DDNAME 'CGM'.

On VM, the filename is always converted to uppercase and the filetype is always CGM. For example, if an application specified 'ABC ' as the connection identifier on a CGM Workstation, the application creates the file: ABC CGM. This file includes all the graphics. If the connection identifier consists of all blanks, the filename is IBMCGM. The CGM workstation recognizes both upper and lowercase.

The CGM file is opened when the workstation is opened. Failure to open the file is a failure to open the workstation. Any data written to the CGM file with an Escape (**GPES**) subroutine (1014) before the first update to the workstation results in a non-conforming file.

Each update workstation operation creates a new picture in the file. The first update workstation operation creates the header. The end metafile element is not generated until close workstation is issued, therefore the CGM file does not conform unless you explicitly close the workstation. The output format is a fixed record length of 400 bytes. The output is a single CGM metafile with each update workstation generating a separate picture within the metafile.

Class Set

The class names which you may specify are limited to the range 0 through 255. Any class name encountered which is greater than 255 or less than 0 is ignored.

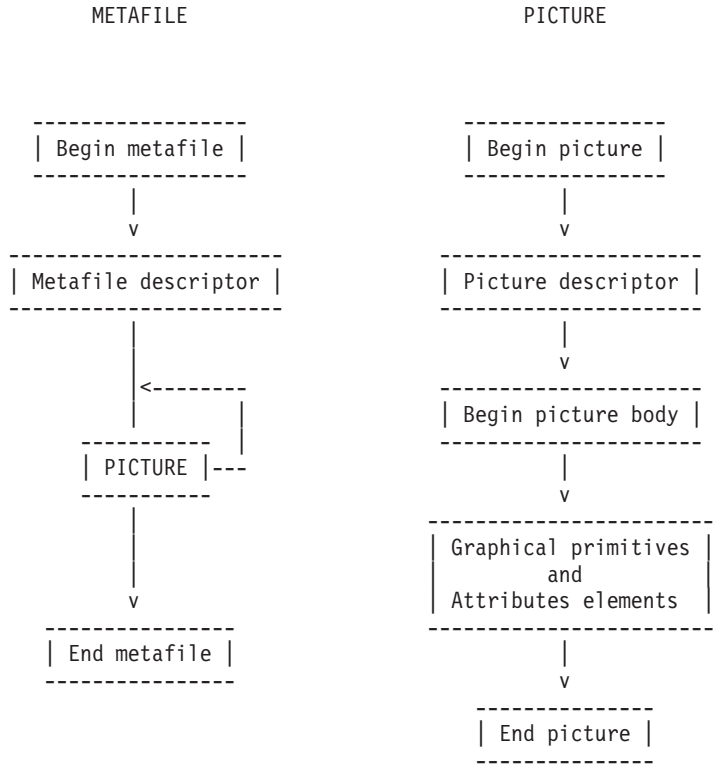
View Mapping

If the projection reference point is between the near and far clip planes, the projection type is changed to PARALLEL and an error is generated.

CGM File Structure

For general information about CGM workstations, see The CGM Workstation.

A CGM file represents a snapshot of a picture created by a program. The file holds an ordered set of elements used to describe the picture in a completely device-independent way. As shown in the example below, the structure of a CGM file accommodates more than one picture.



There are three standard ways of encoding all CGM elements:

Binary	Stores all elements as a bit stream.
Character	Stores the data in a compressed manner.
Clear text	Represents all elements in readable text

The graPHIGS API CGM device driver always uses the binary encoding method.

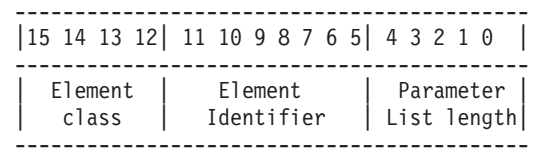
Binary encoding

All elements comprise an element header and the element data.

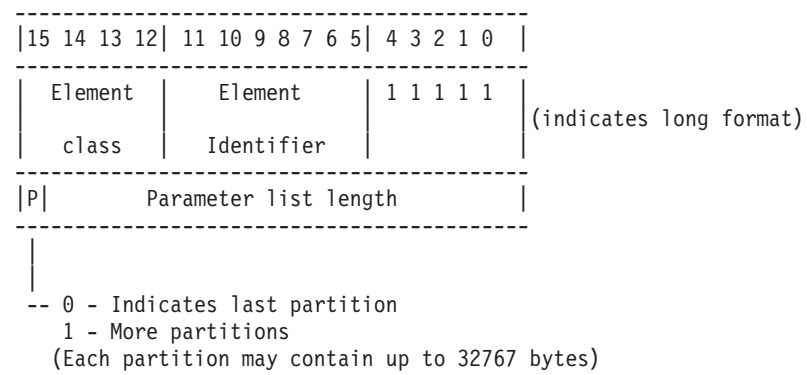
Element Header

The element header is made of an element class, an element identifier, and the parameter length. This information can be stored in two formats.

- Short form (Only accommodates up to 30 bytes of data)



- Long form (Accommodates up to 32,767 bytes of data)



Element Data

CGM defines different formats for integers and real values. The graPHIGS API CGM device driver stores integer values as two-byte integers. Negative values are represented as two's complement. All real values are represented as four-byte IEEE floating-point numbers.

Refer to the ANSI CGM documentation for detailed information about element data.

Delimiter Elements

In CGM, the following metafile will appear in the following order:

Begin Metafile Begins the file. This element has a single parameter which is a character string that identifies the metafile and has a single parameter. For metafiles produced by the graPHIGS API, this character string is `graPHIGS API Metafile./TD>`

Begin Picture Delimits the beginning of a picture description and forces all attributes to be reset to the default values. It contains a character string parameter representing the name of the picture. For metafiles produced by the graPHIGS API, the name of the picture is `PICT` *n* where *n* is incremented from 1 to *n*. All pictures in a CGM file are independent from each other and must start with this element. Each update workstation generates a new picture in the output metafile.

Begin Picture Body Ends the picture description and starts the definition of the picture. This includes a list of graphical primitives and attribute elements. There are no parameters for this element.

End Picture Ends the picture. It has no parameters.

End Metafile Ends the metafile. It has no parameters.

Metafile Descriptor Elements

These elements inform the interpreter of the capabilities needed to interpret the CGM file successfully. Included in these elements are:

Metafile Version Specifies the version of the CGM standard. The CGM device driver uses version 1.

VDC Type Specifies the type of VDC values used. The CGM device driver uses only real VDC (virtual device coordinates).

Integer Precision Specifies the precision for integers. The CGM device driver uses 16 bit integers to represent integers.

Real Precision Specifies the precision for the real numbers. The CGM device driver stores reals as 32-bit floating-point numbers with 9 bits for the exponent and 23 bits for the fraction. This is stored in IEEE floating-point format. The VDC real precision parameter value is 0,9,23.

Index Precision Specifies the precision for indexes. Integers are used to represent indexes. The CGM device driver uses 16 bit index precision.

Color Precision Specifies the precision for each color component contained in the color table. The CGM device driver uses 16 bit integers to represent each color component.

Color Index Precision Specifies the precision for the color index. The color indexes are stored in the file as two-byte integers. The CGM device driver uses 16 bits to represent color indexes

Maximum Color Index Specifies the largest color index stored in the file. This value is the number of color entries for the application output color table. The maximum color index is 255.

Color Value Extent Specifies the range of RGB values contained in the color table. These are multiplied by 1000 before stored as integers in the file. All color values in the CGM file are integers in the range from 0 to 1000. The parameters are a minimum direct color value (0,0,0) and a maximum direct color value (1000,1000,1000).

Metafile Element List Specifies all of the elements that are used in the CGM file. This list enables the CGM interpreter to check the elements before translating the file. All elements contained in this list are described in the following sections:

Picture Descriptor Elements

Picture descriptor elements describe the use of other picture elements.

Scaling Mode Defines the meaning of the virtual device coordinate (VDC) system of the CGM metafile. Two parameters are required. The first specifies the mode, which is either abstract or metric. The graPHIGS API generates metafiles with a scale mode of metric. The second parameter is the metric scale factor which specifies the distance in millimeters in the displayed picture corresponding to one VDC.

If you have not altered the plot size by using escape 1003 (GDF/CGM plot size), then the graPHIGS API computes the scale factor by dividing the maximum display area in meters (0.2582728) by the maximum display area in address units (960). Thus, the default metric scale factor is 0.269034. If you have modified the plot size, then the plot size you supplied is used in place of the maximum display area in meters.

Color Selection Mode Specifies the color selection mode. The graPHIGS API uses indexed color selection mode. The application output color table is stored in the CGM file and then indexed with a 16-bit integer.

Line Width Specification Mode Specifies the line width mode. The CGM device driver uses only an absolute line width.

Marker Size Specification Mode Specifies the marker size mode. This parameter is included in the CGM metafile element list but is irrelevant because the marker primitive is not used by the CGM device driver.

Edge Width Specification Mode Specifies the edge width mode. The CGM device driver uses only an absolute edge width specification mode.

VDC Extent Defines the range of virtual device coordinates (VDC). The CGM device driver stores device coordinates in IEEE floating-point format in the range of 0.0 to 960.0.

Background Color Specifies the background color. The CGM device driver sets the value to (0,0,0).

Control Elements

The following control elements are used by the CGM device driver:

Virtual Device Coordinate (VDC) Real Precision
Specifies the precision for the real numbers. The CGM device driver stores device coordinates as 32-bit floating-point numbers with 9 bits for the exponent and 23 bits for the fraction. This is stored in IEEE floating-point format. The VDC real precision parameter value is 0,9,23.

Clip Rectangle
The clip rectangle defines a rectangular area. The CGM interpreter should clip primitives within this rectangular area. The clip rectangle primitive appears once for each view represented in the picture. For each view represented that has view clipping set to `2=ON`, the clip rectangle parameters are at the lower right and upper left corner of the view boundaries and are translated to the output metafile device coordinate system. For each view with clipping set to `1=OFF`, the clip rectangle parameters are the extent of the virtual device coordinate system (0,0), (960,960).

Clip Indicator
The clip indicator is used to indicate to the interpreter that it should perform clipping to the inside of the rectangular area specified by the clip rectangle. Clipping is always set to `2=ON`.

CGM Elements for Graphical Primitives

The following CGM elements are used by the CGM device driver:

Polyline
The polyline primitive is used to represent polylines, disjoint polylines, nurbs, view borders, geometric text and polymarkers, and in some cases circles, circular arcs, ellipses, and elliptical arcs (see under **ELLIPSE** and **ELLIPTICAL ARC** below).

Polymarker
The polymarker primitive is not used by the device driver to represent any picture information, but is included in the CGM metafile elements list.

Text
The text element is used to represent annotation text in the CGM file. Geometric text is stroked using the **POLYLINE** element.

Polygon Set
Polygons are represented by using polygon set. Some interpreters do not support this element.

Cell Array
The device driver uses this element to support pixel primitives (**GPPXL2** and **GPPXL3**). However, ISO PHIGS cell arrays (**pcell_array**) are represented as polygon sets.

Rectangle
In previous releases this element was used for view shields and view borders. It is no longer used. Polygon set is used for view shields and polylines are used for view borders.

Ellipse
The ellipse primitive is used to represent circles and ellipses whenever possible in the CGM file. However, circles and ellipses are stroked with the polyline element if any of the following conditions are true:

- The workstation is not a CGM workstation.
- The projection type is `2=PERSPECTIVE`.
- The line type rendering is `2=SCALED_TO_FIT_RENDERING`.
- The line type is not a supported CGM line type. `5=LONG_DASH` and `6=DOUBLE_DOT` must be stroked.
- The line representation table has been modified by the application.
- The view is obscured by a higher priority view with shielding `2=ON`.

Elliptical Arc

The ellipse primitive is used to represent circular arcs and elliptical arcs whenever possible in the CGM file. However, circular arcs and elliptical arcs are stroked with the polyline element if any of the following conditions are true:

- The workstation is not a CGM workstation.
- The projection type is 2=PERSPECTIVE.
- The line type rendering is 2=SCALED_TO_FIT_RENDERING.
- The line type is not a supported CGM line type. 5=LONG_DASH and 6=DOUBLE_DOT must be stroked.
- The line representation table has been modified by the application.
- The view is obscured by a higher priority view with shielding 2=ON.

Attribute Elements

CGM Line Attributes

The CGM device driver represents line attributes as follows:

CGM Line Type

The graPHIGS API CGM device driver represents the graPHIGS API line types in the CGM metafile as follows:

Table 9. CGM Line Types

graPHIGS API line types	CGM representation	Result
1	1	SOLID_LINE
2	2	DASHED
3	3	DOTTED
4	4	DASHED_DOT
7	5	DASH_DOUBLE_DOT
Note: If the line representation table is modified or accurate line types (scaled-to-fit) are selected, then the line types are stroked. Lines types 5=LONG_DASH and 6=DOUBLE_DOT are always stroked.		

Line Width

The graPHIGS API CGM device driver stores the graPHIGS API line widths in IEEE floating-point format. The nominal line width is equivalent to 1/1000 of the default display surface area (0.96 in virtual device coordinates if the plot size has not been modified).

Line Color

The graPHIGS API CGM device driver represents line color as integers in the range of 0 to 255 (the maximum color index value). This integer is used by the interpreter as an index into the color table.

CGM Marker Attributes

Marker Type, Marker Size, Marker Color

The graPHIGS API CGM device driver represents the graPHIGS API markers in the output CGM metafile as a series of polylines. Therefore, these attribute elements are not used.

CGM Interior Attributes

The CGM device driver represents interior attributes as follows:

Interior Style

The graPHIGS API CGM device driver represents the graPHIGS API interior styles in the CGM metafile as follows:

Table 10. Interior Style

graPHIGS API interior style	CGM representation	Result
1	0	HOLLOW
2	1	SOLID
3	2	PATTERN
4	3	HATCH
5	4	EMPTY

Fill Color

The graPHIGS API CGM device driver represents interior as an integer in the range of 0 to 255 (the maximum color index value). This integer is used by the interpreter as an index into the color table.

Fill Reference Point

The graPHIGS API CGM device driver always uses the default fill reference point. This element is included in the CGM metafile element list but is not used.

Hatch Style

The graPHIGS API CGM device driver represents the graPHIGS API hatch styles in the CGM metafile as follows:

Table 11. Hatch Styles

graPHIGS API hatch style	CGM representation	Result
1	2	Vertical lines
2	1	Horizontal lines
4	3	Diagonal lines (positive slope; lower left to upper right 45[default], medium spacing)
6	4	Diagonal lines (negative slope; lower right to upper left 135[default], medium spacing)
Note: No other hatch styles are supported. Other hatch style values default to the graPHIGS API hatch style 1 (vertical lines) and CGM hatch style 2 (vertical lines).		

Pattern Table

When you open a workstation, the CGM workstation stores the default pattern tables in the CGM file.

Pattern Index

The graPHIGS API CGM device driver represents pattern index as 16-bit integers in the range of 0 and 10.

Pattern Size

The graPHIGS API CGM device driver always uses the default pattern size. This element is included in the CGM metafile element list but is not used.

CGM Text Attributes

The CGM device driver represents annotation text attributes as follows:

Text Path

The graPHIGS API CGM device driver uses the following text path `0=RIGHT, 1=LEFT, 2=UP, 3=DOWN`.

Text Alignment The graPHIGS API CGM device driver always uses the default text alignment. This element is included in the CGM metafile element list but is not used.

Text Precision The graPHIGS API CGM device driver represents text precision as integers. Text precision affects annotation text only.

Text Font Index The graPHIGS API CGM device driver always uses the default text font index. This element is included in the CGM metafile element list but is not used.

Text Color The graPHIGS API CGM device driver represents text color as integers in the range of 0 to 255 (the maximum color index value). Text color affects annotation text only. This integer is used by the interpreter as an index into the color table.

Character Set Index The graPHIGS API CGM device driver always uses the default character set index. This element is included in the CGM metafile element list but is not used. Character set index affects annotation text only.

Character Orientation The graPHIGS API CGM device driver represents the base vector always at a right angle to the up vector. Character orientation affects annotation text only.

Character Spacing The graPHIGS API CGM device driver represents the graPHIGS API character spacing in IEEE floating-point format. Character spacing affects annotation text only.

Character Expansion Factor The graPHIGS API CGM device driver represents the graPHIGS API character expansion factor in IEEE floating-point format. Character expansion affects annotation text only.

Character Height The graPHIGS API CGM device driver represents the graPHIGS API character spacing in IEEE floating-point format. Character height affects annotation text only.

CGM Edge Attributes

The CGM device driver represents edge attributes as follows:

Edge Type

The graPHIGS API CGM device driver represents the graPHIGS API edge types in the CGM metafile as follows:

Table 12. Edge Types

graPHIGS API edge type	CGM representation	Result
1	1	SOLID_LINE
2	2	DASHED
3	3	DOTTED
4	4	DASHED_DOT
7	5	DASH_DOUBLE_DOT
Note: No other edge types are supported. They default to edge type 1=SOLID_LINE.		

Edge Width

The graPHIGS API CGM device driver represents the graPHIGS API edge widths in IEEE floating-point format. The nominal line width is equivalent to 1/1000 of the display surface area which is 0.96 of the virtual device coordinates if the plot size has not been modified.

Edge Color

The graPHIGS API CGM device driver represents edge color as an integer in the range of 0 to 255 (the maximum color index value). This integer is used by the interpreter as an index into the color table.

Edge Visibility
The graPHIGS API CGM device driver uses this attribute to indicate whether or not the polygon edges are drawn. It can be either `0=OFF` or `1=ON`.

CGM Color Attributes
The CGM device driver represents color attributes as indexes into the color table.

Color Table
The first parameter is the starting color table index. It is always zero.

The second parameter is a list of 255 direct color values where each color value consists of 3 color components (red, green, and blue). Each component is a 16-bit integer in the range 0-1000.

Conformance
If an application places a CGM element in a file and that element is not included in the Metafile Element List, then the CGM file is no longer considered to conform to standard.

Full Conformance
If an application does not use **GPES** escape (1014) or **GPWDO**, and the application explicitly closes the CGM workstation, then the result is a CGM metafile fully conforming to ANSI X3.122-1986 as defined in section 7.3 of that document.

Metafile Interpreters
ANSI X3.122 does not define conformance for a metafile interpreter. Not all CGM interpreters support all CGM elements defined by the specification. For example, the graPHIGS API can generate polygon sets with multiple partitions, however, many commercial interpreters do not support polygon set. The graPHIGS API does not generate any CGM elements that are not included as part of the Drawing Plus control Set defined in section 4.3.2.2 of the ANSI Specification. See General Output Facilities for a list of metafile elements that the graPHIGS API does generate.

Workstation Dependent Output
The Escape (**GPES**) subroutine and the Workstation Dependent Output (**GPWDO**) subroutine allow an application to directly pass binary data to be included in the resultant CGM file. The escape allows the inclusion of control or other non-graphical data outside of a picture such as messages, application data, and escapes.

The **GPWDO** subroutine allows the insertion of binary data into a structure for inclusion into a CGM file along with other graPHIGS API generated CGM elements. Both the **GPES** (1014) escape and the **GPWDO** subroutines require that you specify the total length of the data being supplied to the graPHIGS API, independently of the encoded length field contained within the CGM element header.

Interpretation of CGM Data
All data is assumed to be a valid CGM binary encoded element. Both the long form and short form of the headers are valid. There is a 64K limit on the amount of data that can be passed via the nucleus connection. The output is always a single partitioned CGM element. No attempt is made to validate the class and element identifier within the CGM header, however, the length check is performed to insure that an interpreter can continue processing after encountering an unsupported or unrecognized CGM element in a metafile.

How the length is checked
Two length values are processed by the graPHIGS API during a **GPWDO** subroutine or a **GPES** escape 1014 for the CGM workstation. One is the DATALENGTH, the total length of the data supplied by the application. For **GPWDO**, this is the value of the length parameter supplied by the application. For **GPES**,

this is equal to the *lidr* parameter (length of data record) minus 16, the fixed size of the *idr* (input data record). Additionally, the CGM element must contain an ENCODED PARAMETER LIST LENGTH (see Element Header and Element Data).

If the DATALENGTH is less than the minimum two-byte CGM element size, then error 2050 is returned and no output is written. Error 2050 is also returned when the DATALENGTH is less than four bytes and the ENCODED PARAMETER LIST LENGTH indicates a long form header, because in this case insufficient data is available for a valid long form CGM header.

If the DATALENGTH is greater than 32767+4, which is the largest amount of data that can fit in a single partition, plus the size of a long form header, then error 2051 is returned and no output data is written. Otherwise, the short form ENCODED PARAMETER LIST LENGTH is checked. If this is a hex 0x1F value, then the element has a long form header and the long form ENCODED PARAMETER LIST LENGTH is used. If the sum of the ENCODED PARAMETER LIST LENGTH, added to the size of the header (two bytes for short form and four bytes for long form,) is not equal to the DATALENGTH, then error 2052 is returned and processing continues. The amount of data written to the file is equal to the size of the element header plus the ENCODED PARAMETER LIST LENGTH. Excess Data supplied by the application is ignored.

If the DATALENGTH is less than the sum of the ENCODED PARAMETER LIST LENGTH plus the size of the CGM element header, then application-supplied data equal to the DATALENGTH is written to output and the element is padded with 0's until the amount of data written is equal to the ENCODED PARAMETER LIST LENGTH plus the size of the CGM element header.

CGM Summary

- No verification other than encoded length checking is performed.
- Elements with a data length not long enough to include a valid length are rejected.
- Elements with a data length longer than a single partition length are rejected.
- Maximum data length is 32771 based on maximum single partition size and size of largest header (32767+4).
- Multiple partitions are not supported.
- Elements with data length unequal to encoded length plus header size are padded/truncated to encoded length.

Other Considerations for CGM

Partitioned Data Less than 32K Not Supported

Although the graPHIGS API does not support partitioned parameter data, partitions can be any size smaller than 32K. It is possible for an element with multiple partitions to pass the length test. The graPHIGS API *always* sets the partition bit to zero (last partition). This means that only the first of multiple partitions would be written to the CGM file. The rest would be truncated.

Clipping

Because the graPHIGS API is unable to determine the nature of the graphical objects created through these subroutines, no attempt is made to clip any of the objects.

Using GPWDO to Modify Attributes

If you intend to use **GPWDO** to modify attributes in the CGM file, it is important to understand how attributes are actually placed in the file. You should be familiar with the ANSI standard for CGM. The graPHIGS API CGM file does not necessarily resemble the contents of graPHIGS API structure store. For instance, attributes are not placed into the file by the device driver until a primitive dependent on them is written. They are not placed into the file unless they have been changed since the last graPHIGS API output using that attribute. Additionally, several graPHIGS API primitives may map to a single CGM primitive. For example, geometric text and polylines are both drawn as lines. Both rely on the line attributes in the CGM file.

For example, if you want to use **GPWDO** to change the line width used to stroke geometric text but the last line width output to the CGM file (for an arc, a polyline, etc.) is not equal to the nominal line width (scale factor of 1) used to stroke geometric text, then the device driver places another line width attribute into the CGM file, overriding the **GPWDO**. To ensure that the line width is not output with the text, it is necessary to insure that the line width set into the file is equal to the device drivers internal text line width scale factor setting (1.0). This can be accomplished by inserting a **GPLWSC** with a scale factor of 1.0 into the structure, followed by a **GPPL2**. The **GPLWSC** subroutine must be followed by a line primitive because the line width attribute is not output until a line primitive that is dependent on it is output. The graPHIGS API's internal representation of the current line width is not updated until the attribute is output. The line can be specified as zero length or in background color to prevent its display. Likewise, if a line width is changed in the CGM file it does not necessarily reset to its previous value simply by inserting **GPLWSC** into the structure. If the linewidth provided to **GPLWSC** is the same as th e last value the device driver wrote to the file, then it does not output it again.

Restrictions

- Data must be at least two bytes long.
- Data must have a valid CGM parameter list length.
- Only single partition elements are allowed. If the continuation bit in the length field is set it is cleared to zero.
- A length greater than a single partition maximum is not allowed.
- If an application makes an escape call before the first update workstation, the CGM file does not begin with a Begin Metafile Order and is not in conformance. This file may fail to work with many interpreters currently available.
- The metafile elements list is incorrect if any elements that are not normally used by the graPHIGS API are inserted into the file.
- Applications are responsible for ASCII/EBCDIC conversion and IEEE/S370 floating-point conversion. The Convert Data (**GPCVD**) subroutine may be used to facilitate this process.

The IMAGE Workstation

Overview

The IMAGE workstation is an output only workstation that provides a means for capturing and storing image data in a form which can be processed by other applications. These images are in a raster format and are stored as a color bitmap. This allows pictures, generated using advanced rendering functions such as lighting, shading and hidden line and hidden surface removal, to be stored on a disk file. The advanced rendering functions exceed the line drawing capability of vector image formats such as CGM and GDF, which store pictures as lines and filled areas.

> **Note:** The IMAGE workstation is available on the operating system only.

The graPHIGS API generates the images by rendering into a ″virtual frame buffer″, or memory map. All images are rendered using 24-bit color. The application can control the image resolution and the defined display size and can select from several IMAGE formats. The graPHIGS API then renders into the frame buffer at the appropriate size and resolution and creates a disk file according to the graphics image format specified.

Output Formats

Three image formats can be produced by the graPHIGS API.

- Adobe Encapsulated PostScript Format at 12-bits per pixel (4-bits per color component using an RGB color model). This is the default format.
- Adobe Encapsulated PostScript Format at 24-bits per pixel (8-bits per color component using an RGB color model).

- IBM's Image Object Content Architecture (IOCA) Function Set 10 (FS10).

Use the IMAGEFMT procopt, IMAGEFMT (Image Output Format), to set the image output format to one of the following:

```
PSl1_4BIT   4-bits per color component Adobe PostScript Language Level 1 (Default)
PSl1_8BIT   8-bits per color component Adobe PostScript Language Level 1
IOCA_FS10   1-bit per pixel Image Object Content Architecture.
```

Adobe PostScript Page Description Language

The Adobe PostScript Page Description Language is a page description language that includes some advanced programming constructs. Pictures can be described in this language and stored in a file. This file may be parsed and displayed by an application or device that interprets the Adobe PostScript Page Description Language. For the purposes of this documentation, Adobe PostScript output devices will refer to any device which includes a PostScript interpreter licensed from Adobe, or the Adobe Display PostScript System. These devices, printers, displays and other applications, convert or post-process the image.

Language Levels

The Adobe PostScript language exists in different levels with extensions.

Table 13. Adobe PostScript Language Levels

Level	Description
Level 1	Level 1 is the smallest set of Adobe PostScript language operators that a device, including PostScript software from Adobe Systems, will support. All Adobe PostScript devices should support all the level one operators.
Extensions	Extensions represent capabilities that have been added to Adobe PostScript Level 1. For example, the PostScript language color operators.
Level 2	Level 2 function includes all of Level 1 and its extensions, as well as additional functions.
Adobe Display PostScript System	Adobe Display PostScript System is an additional set of operators that may be included on top of Level 1, Level 1 plus extensions, or Level 2 support. These operators are generally used in an interactive environment.

The graPHIGS API IMAGE output will only contain operators that are defined within PostScript Language Level 1 with color extensions. The PostScript produced will contain Adobe PostScript language code that emulates the color extensions on Level 1 devices when the extensions are not available. These files are printable on any Adobe PostScript Level 1 device. See Printing Color PostScript Files on Black and White Printers.

Adobe Encapsulated PostScript

Adobe defines several formats of PostScript Language programs. The graPHIGS API will create an Adobe PostScript output in Encapsulated PostScript file format (ESPF). Each update workstation will store a single picture or image into a unique file name (See Output Filenames). These pictures can be sent directly to a Adobe PostScript output device or included into another Adobe PostScript language page description. If the image is sent directly to an Adobe PostScript output device, it will be placed on the device's coordinate systems origin. This is usually the lower left hand corner for a PostScript device.

Color Representation

Adobe PostScript Level 1 language with color extensions represents color pixel data as 3 integer values. These values represent the intensity of red, green, and blue in the pixel. This data is character encoded as hexadecimal data (`0F` would be the value 16). Each 8-bits of pixel data require 2 bytes of character data to be encoded. If a full 24-bit color image is to be stored in the output file, the image will require 6 bytes per pixel. The graPHIGS API can store 8-bits for each color component into the image file or truncate it and store the high order 4-bits of each component. This reduces the image storage requirements to 3 bytes per pixel and reduces the number of shades of each color band from 255 to 16. If the advanced lighting

and shading capabilities of graPHIGS are not being used and only flat shaded colors are displayed, we recommend that you use the PSl1_4BIT option on the IMAGEFMT procopt described in IMAGEFMT (Image Output Format) to reduce the size of the output PostScript file. However, if any type of continuous tone shading is used, the available shades of color will be reduced and may result in the shaded portions of the image appearing ″banded″.

Printing Color PostScript Files on Black and White Printers

Adobe PostScript language files produced by the graPHIGS API include Adobe PostScript language code that inquires whether the output device supports the colorimage operator.

For devices that don't support the colorimage operator, it is defined within the graPHIGS Adobe PostScript Output language file as a function that combines the red, green, and blue values into a single gray value according to the following ratio:

```
.299 * Red  + .586 * Green + .114 * Blue
```

The grayscale image is then displayed using the PostScript language image operator. If the PostScript image was generated as 8-bits per color component, the resultant gray value will be 8-bits and have a range of 255 shades of gray. If the PostScript image was generated as 4-bits per component, the resultant gray value will be 4-bits and have a range of 16 shades of gray. The conversion to grayscale occurs at the output device. The conversion involves a series of floating point operations and may take longer on devices with only Level 1 support.

Color Model

The Adobe PostScript language supports several color models, including RGB and CMYK. Most printers generate output using a CMYK model and halftoning techniques. The graPHIGS API rendering is done in RGB color space. Accurate mapping to the output device's color space is device dependent due to differences in inking and other printer technologies. The output images are represented in RGB color space and no assumptions are made about the output device. Adobe PostScript provides for the definition of mapping functions that can be modified by the end user or replaced by the application developer to account for differences in printer technology. If continuous tone is supported by the output device, color reproduction may occur directly. Otherwise, shades of color are produced using halftoning techniques. The Adobe PostScript language also provides functions to modify the halftoning. The effects of these functions are device specific.

Default Coordinate System

The default coordinate system origin for the Adobe PostScript language is in the lower left hand corner of the display area. Each device coordinate represents 1/72 of an inch. The X axis increments towards the right and the Y axis increments towards the top of the display area. The graPHIGS API renders Adobe PostScript language files using this default system. To modify this, see Processing graPHIGS EPS Files Using sed.

Contents of graPHIGS PostScript Images

ESPF files created by the graPHIGS API are ASCII text Adobe PostScript page description language programs. In addition to executable programs, functions and data, they contain comments to identify the file and to facilitate modifying the displayed image. The program is organized into the following four sections:

- Header
- Prolog
- Body
- Trailer

Header
The header consists mainly of PostScript comments. Some of these comments may be processed as data by other applications such as print spoolers or desk top publishing software. Most of these comments are recommended or defined as part of the Adobe encapsulated PostScript format.

```
%!PS-Adobe-2.0 EPSF-2.0
%%BoundingBox: 0 0 612 792
%%Creator: IBM graPHIGS API V2R2.4.0
%%CreationDate: Thu Jul  8 15:22:07 1993
%Bits Per Component: 4
%%Title: Samp0001.ps
%%EndComments
save
gsave
```

%!PS-Adobe-2.0 EPSF-2.0
Line 1 identifies the file as conforming to the Version 2 Adobe Encapsulated Postscript Format.

%%BoundingBox: 0 0 612 790
Line 2 defines a bounding box that encloses the entire image. The units used are printer points or 1/72ths of an inch. The values represent the lower left and upper right corners of the bounding box. Since the entire display area is initialized to color table entry 0, the graPHIGS API defines the bounding box to be the entire workstation display area. The workstation display area can be modified using the DCMETERS procopt described in DCMETERS (Device Coordinate Meters).

save gsave
These two commands save the state of the Adobe PostScript processing device prior to execution of the file so that it can be restored after the EPS file has been processed.

Prolog
The prolog is bracketed by the %%Begin Prolog and %%End Prolog comments. It consists of code that tests for the existence of the colorimage operator and emulates this operator if it doesn't exit.

Body
The body consists of Adobe PostScript language code that defines the location and orientation of the image on the output device, code to read and display the image, and the image data. In addition, %GPMODxxxx comments have been added, where xxx is a particular function. This allows you to use `sed` to post-process the image if you are going to print it directly. See Processing graPHIGS EPS Files Using sed for details.

Trailer
The `trailer` consists of:

- a **grestore** command that restores the graphics state
- a %GPMODTRAILER comment allows `sed` to be used to insert some postprocessing if desired
- a **showpage** command that displays the image when sent to an output device
- a **restore** command that resets global memory.

Processing graPHIGS EPS Files Using sed
If an application sends graPHIGS EPS files directly to an output device, it may be desirable to modify the files to change the orientation of the picture with regard to location and scale. graPHIGS EPS files contain ASCII characters and can be edited using any text editor. Due to the large size of most images, this is generally not practical. To facilitate using the **sed** command as a filter for modifying these files, comments in the form %GPMODxxxx have been placed in the file at appropriate locations to provide targets for the **sed** command. Here are some examples of using `sed` to modify the file:

- Trimming the image:

  This example uses **sed** to insert a clipping command in place of the %GPMODCLIP comment.

```
sed s/%GPMODCLIP/"newpath 100 100 moveto\
                           100 200 lineto\
                           200 200 lineto\
                           200 100 lineto\
                           closepath clip"\
```

This causes the image to be clipped to a rectangle, bounded by the points (100,100), (100,200), (200,200), (200,100), when displayed.

- Translating the image to a new origin:

  This example uses **sed** to insert a translate command in place of the %GPMODTRANSLATE comment.

  ```
  sed s/%GPMODTRANSLATE/"100 100 translate"/
  ```

  This causes the lower left hand corner of the image to be displayed at location (100,100) on the output device.

- Scaling the image:

  This example uses **sed** to insert a scale command in place of the %GPMODSCALE comment.

  ```
  sed s/%GPMODSCALE/".5 .5 scale"/
  ```

  This causes the image to display at 1/2 the size specified by the DCMETERS procopt described in DCMETERS (Device Coordinate Meters).

- Adding other objects to the image:

  This example uses `sed` to insert graphical objects in place of the %GPMODTRAILER comment.

  ```
  sed s/%GPMODTRAILER/"\.5 setgray          \
                      \/Helvetica findfont 15 scalefont setfont \
                       100 100 moveto (graPHIGS IMAGE) show"/
  ```

  This displays the text ″graPHIGS IMAGE″ at location (100,100) along with the image.

- Converting an image to reverse video:

  This example uses **sed** to replace the %GPGSMOD comment with the command which will covert an image to reverse video. The comment will only appear in 8-bit per component files. With 8-bit drawings, you can modify the grayscale values to convert to reverse video on black and white printers.

  ```
  sed s/%GPGSMOD/255 exch sub/
  ```

IOCA Function Set 10 (FS10)

The graPHIGS API can generate files that conform to IBM's Image Object Content Architecture FS10 specification. The rendered image will be converted to a 64 level grayscale according to the NSTC video standard. IOCA FS10 describes bi-level (black and white only) images. In order to support 64 shades of gray, each pixel will be halftoned in the output file using an 8 by 8 block of black or white dots. The order in which the dots in the halftone cell are blacked is represented by the following matrix. If the value of the pixel, after converting to grayscale, is less than or equal to the value in the matrix, the corresponding dot in the halftone cell is printed black, otherwise it is printed white.

```
 0,  48,  12,  60,   3,  51,  15,  63,
32,  16,  44,  28,  35,  19,  47,  31,
 8,  56,   4,  52,  11,  59,   7,  55,
40,  24,  36,  20,  43,  27,  39,  23,
 2,  50,  14,  62,   1,  49,  13,  61,
34,  18,  46,  30,  33,  17,  45,  29,
10,  58,   6,  54,   9,  57,   5,  53,
42,  26,  38,  22,  41,  25,  37,  21
```

Output Filenames

The IMAGE workstation uses the connection identifier, which the application can select, to derive the output filename. The connection identifier must be a valid file name which may include a full or relative pathname. Upper and lower case characters are valid. If the optional pathname is not specified, the file will be created in the current directory. The optional pathname is stripped from the connection identifier and the first five characters from the resulting file name are concatenated with a three digit number in the range 001-999. If any of these characters are blank, they are replaced with a fill character. An ″X″ replaces

a blank first position character, and a ″0″ replaces any other blanks between the second and fifth positions. A filename extension, which indicates the format of the output file, is added to the filename.

A new file is generated each time the IMAGE workstation is updated. When a new file is created, the number represented by the last three characters in the file is increased by one. If this number exceeds 999, it is reset to 001. A new file will overwrite an existing file of the same name. Therefore, a maximum of 999 different files can be generated during a session.

If an application specifies `'ABC '` as the connection identifier for an IMAGE workstation with `PS11` format, the files created on successive workstation updates will be:

Table 14. Example Filenames

UPDATE NUMBER	FILENAME
1	ABC00001.PS
2	ABC00002.PS
3	ABC00003.PS
.	.
.	.
999	ABC00999.PS
1000	ABC00001.PS
1001	ABC00002.PS

Image Size and Resolution

The graPHIGS API provides procopts to modify the display area and the resolution of the IMAGE workstation. The best combination of resolution and size depends on the target output device.

Modifying the Defaults

Use the DCMETERS procopt described in DCMETERS (Device Coordinate Meters) to modify the display size. The size must be greater than 0 along both axes. The graPHIGS API does not impose maximum values for the display areas. Actual limits are device dependent. The default display area is 0.2157 meters in width by 0.2794 meters in length. This translates to 8.493 inches by 11 inches which will fit within the 8.5 x 11 inches of a letter sized page. This may require modification if the output device imposes boundaries.

Use the DCUNITS procopt described in DCUNITS (Device Coordinate Address Units), to specify an image resolution in total number of pels along the X and Y axes. The minimum value for each axis is 8 pels. The maximum value is 4096 pels. The default is 637 pels along the X axis and 825 pels along the Y axis, which translates into a pel density of 75 pels per inch when the default value DCMETERS (Device Coordinate Meters) is used.

Selecting Resolution

The choice of image resolution may dramatically affect the appearance of the displayed image. Appropriate values will vary depending upon the target output device. If the image resolution is too low, the picture will display with a tile appearance. If the resolution exceeds that of the output device, the output file will be very large. Also, since the output device will have to combine several image pixels into a single device pixel, small objects in the image may appear distorted, and thin lines may appear dashed. Nominal values, such as line width and polymarker size, are defined as functions of pixel size. A resolution that is higher than that of the output device may cause some lines and objects to be rendered in a sub-pixel scale and these objects may not display.

Printer resolutions vary, but typical values are 240, 300, 480 and 600 dots per inch (dpi). Many IBM devices will be 240 or 480 dpi while most others will be 300 or 600 dpi. Since most printers will halftone to

produce shades and blends of color, the ideal image resolution will generally be lower than that of the output device. The default of 75 dpi was chosen to allow a 4x4 dot halftone cell on a 300 dpi printer or a 8x8 dot halftone cell on a 600 dpi printer.

Chapter 3. Workstation Description Tables

The following tables contain default values for the graphic workstations and adapters based on a:

- 23 inch display for the X workstation
- 23 inch display for the 6090 workstation
- 19 inch display for the 5080 workstation

(The size of the displays is determined by the diagonal measurement of the screen.)

The right-hand column in the tables lists the subroutine that you can use when you want your application to know the value for the facility.

The data types of the returned values are identified by the following codes:

Data Type	Definition
I Integer	A whole number
R Real	A floating-point number
S String	A character string
E Enumeration	A data type comprised of a set of values. The set is defined by enumerating the identifiers denoting the values.
n **Quantity**	This specifies an undesignated quantity of data.

Note: The notation of *n* (number) [default] *t* (data type) indicates a collection of data of that type. This can be indicated in one of two ways:

1. By using notation such as 3[default]R (three real numbers), which could specify something like the *x*, *y*, and *z* coordinates of a three-dimensional point or RGB values
2. By using a variable number such as *n*[default]I, which specifies a collection of *n* integers.

The values identified with the symbol * reflect the default value of a workstation configuration variable; that is, this may not be the value of the variable in the actual workstation description table after the workstation is opened. See Advanced Concepts for a discussion of this concept.

Some tables are preceded by workstation-dependent and/or adapter-specific information.

The tables in this section are arranged in this order:

General Workstation Facilities
General Output Facilities
Polyline Facilities
Polymarker Facilities
Text Facilities
Interior Facilities
Edge Facilities
Color Facilities
Generalized Drawing Primitive (GDP) Facilities
Generalized Structure Element (GSE) Facilities
Escape Facilities
Image Facilities
Advanced Output Facilities
Curve and Surface Facilities
Advanced Attribute Facilities
General Input Facilities
Available Triggers

Locator Devices
Stroke Devices
Valuator Devices
Choice Devices
Pick Devices
String Devices
Button Devices
Scalar Devices
Vector Devices
Break Action

General Workstation Facilities

All Workstations

- The maximum display surface size changes with various display hardware. Use the Inquire Maximum Display Surface Size (**GPQDS**) subroutine to obtain the maximum display surface of your workstation (in device coordinate units and address units).
- The default number of views is 16. The number of views can be increased via the View Table Entries (`VWTBLSZ`) procopt. See VWTBLSZ (View Table Entries). Use the Inquire Workstation Configuration Variability (**GPQWCV**) subroutine to obtain the maximum number of definable view table entries.
- If the Inquire Rendering Target (**GPQART**) subroutine returns a value of zero for the number of rendering targets, then the specified workstation does not support explicit traversal.

X

General Information Applying to All Adapters

- The `LANG` environment variable determines the primary character set. For more information on primary character sets, see Opening the X Workstation. Use the Inquire Primary Character Set (**GPQCS**) subroutine to obtain the primary character set identifier for the specified workstation type.
- Explicit traversal control is supported. If the workstation is in single buffer mode, then the Inquire Rendering Target (**GPQART**) subroutine returns a value of one for the number of available rendering targets. If the workstation is in double buffer mode, the **GPQART** returns a value of two for the number of available rendering targets.

Direct Window Access(DWA) Adapters

- DWA Adapters include the POWER GXT6500P, POWER GXT4500P, POWER GXT6000P, POWER GXT4000P, POWER GXT3000P, POWER GXT2000P, POWER GXT1000P, POWER GXT1000, POWER GXT800P, POWER GXT800M, POWER GXT550P, POWER GXT500P, POWER GXT500D, and POWER GXT500.

  The POWER GT4 Family and POWER GTO are also DWA Adapters but their general capabilities may differ from the others listed above.

  The POWER GXT250P can only support DWA in 8-bit color mode with a maximum 1024x768 display resolution. The POWER GXT255P can only support DWA in 8-bit color mode.

Softgraphics graPHIGS (XSOFT) Adapters

XSOFT Adapters include the POWER GXT300P, POWER Gt1x, POWER GXT100, POWER Gt3i, and the Color Graphics Display Adapter. The POWER GXT255P and POWER GXT250P also support XSOFT in all color modes greater than 8-bit, and the POWER GXT250P in all display resolutions greater than 1024x768.

XLIB (non-DWA) Adapters

The graPHIGS API may be available on additional 2D adapters via the XLIB interface.

6090 and 5080

- Primary character sets depend on the hardware configuration of your workstation. For information on how to customize your workstation, see *The GDDM/graPHIGS Programming Interface: Installation and Problem Diagnosis*.

IMAGE

- You can set the width and height of the display device in Device Coordinate address units via the Device Coordinate Address Units (DCUNITS) procopt.
- You can set the width and height of the display device in Device Coordinate meters via the Device Coordinate Meters (DCMETERS) procopt.
- For more information on setting the width and height with the display device, see The IMAGE Workstation.

GDF

- Characteristics of the 3270-PC/GX are assumed for the target display device. For more information, see The GDF Workstation.
- You can change the size of the display area using the Escape (**GPES**) subroutine (Escape 1003: GDF/CGM Plot Size). However, the Inquire Maximum Display Surface Size (**GPQDS**) does not reflect this change.

CGM

- Characteristics of the 3270-PC/GX are assumed for the target display device. For more information, see The CGM Workstation.
- You can change the size of the display area using the Escape (**GPES**) subroutine (Escape 1003: GDF/CGM Plot Size). However, the Inquire Maximum Display Surface Size (**GPQDS**) does not reflect this change.

Table 15. General Workstation Facilities - X Workstation Default Values

General Workstation Facilities	Data Type	IMAGE	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Workstation category (1=OUTPUT, 2=INPUT, 3=OUTIN)	E	OUTPUT	OUTIN	OUTIN	OUTIN	OUTIN	**GPQWC** [*type*]
Device coordinate unit (1=METERS, 2=OTHER)	E	METERS	METERS	METERS	METERS	METERS	**GPQDS** [*units*]
Maximum display surface size in device coordinate units	3[default]R	0.215, 0.279, 0.215*	0.425, 0.340, 0.425*	0.425, 0.340, 0.425*	0.425, 0.340, 0.425*	0.425, 0.340, 0.425*	**GPQDS** [*csize*]

Table 15. General Workstation Facilities - X Workstation Default Values (continued)

General Workstation Facilities	Data Type	IMAGE	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Maximum display surface size in address units	3[default]I	637, 825, 637*	1280, 1024, 1280*	1280, 1024, 1280* Exceptions **POWER GXT6500P, POWER GXT4500P, POWER GXT6000P, POWER GXT4000P, POWER GXT2000P**: 1600, 1200, 1280* **POWER GXT250P:** 1024, 768, 1024*	1280, 1024, 1280* or 1024, 768, 1024*	1280, 1024, 1280*	**GPQDS** [*asize*]
Number of definable view table entries[2]	I	63*	63*	63*	63*	63*	**GPQWCV** [*number*]
Primary character set	I	8*	8*	8*	8*	8*	**GPQPCS** [*csid*]
Number of available rendering targets	I	1	2*	2*	1 or 2*	1	**GPQART** [*totnum*]

Notes:

1. [1]See the text prior to General Workstation Facilities for a list of DWA and XSOFT Adapters.
2. [2] Entry 0 cannot be modified.

* See the text prior to this table for more information.

Table 16. General Workstation Facilities

General Workstation Facilities	Data Type	6090	5080	GDDM	GDF	CGM	Inquiry
Workstation category (1=OUTPUT, 2=INPUT, 3=OUTIN)	E	OUTIN	OUTIN	OUTIN	OUTPUT	OUTPUT	**GPQWC** [*type*]
Device coordinate unit (1=METERS, 2=OTHER)	E	METERS	METERS	METERS	METERS	METERS	**GPQDS** [*units*]
Maximum display surface size in device coordinate units	3[default]R	0.425, 0.340, 0.425*	0.28448, 0.28448, 0.28448*	0.24682, 0.17574, 0.24682*	0.2582728, 0.2582728, 0.2582728*	0.2582728, 0.2582728, 0.2582728*	**GPQDS** [*csize*]
Display surface size in address units	3[default]I	1280, 1024, 1280*	1024, 1024, 1024*	720, 384, 720	960, 960, 960	960, 960, 960	**GPQDS** [*asize*]
Number of definable view table entries[1]	I	63*	63*	63*	63*	63*	**GPQWCV** [*number*]
Primary character set	I	1*	1*	1	1	1	**GPQPCS** [*csid*]
Number of available rendering targets	I	0	0	0	0	0	**GPQART** [*totnum*]

Note:

[1]Entry 0 cannot be modified.

* See the text prior to this table for more information.

General Output Facilities

X

General Information Applying to All Adapters

- Antialiasing is supported only for polyline primitives with nominal line width.
- BNIL (Before Next Input Locally) deferral mode is treated as BNIG (Before Next Input Globally) deferral mode.
- Polygon with Data Primitives: The primitive supports optional data that indicates that the application determined the convexity of the polygon. Specifying this optional data with the primitive definition enables better performance because the system rendering code does not have to determine the convexity of the polygon each time the polygon is rendered. To determine the convexity of a set of polygons, the graPHIGS API on the RS/6000 contains a sample program under the operating system directory:

  `/usr/lpp/graPHIGS/samples/convexcheck`
- Component frame buffers have three components. Indexed frame buffers have one component.
- Pixel Primitives: The pixel primitive is not part of the PHIGS definition and needs special consideration when using it on devices with frame buffers that are not eight bits deep. Because the pixel primitive only specifies eight bits of pixel data, there is no clear method map those eight bits to frame buffers that have other than eight bits of memory.

  For frame buffers with component organizations (such as the 24-bit 3D adapter), the eight bits are replicated into each component by the graPHIGS API. This normally produces grey-scale images, since the same bits are put into each of the red, green, and blue frame buffer components.

  To replicate the pixel data to each frame buffer component, use the Set Frame Buffer Write Protect Mask (**GPFBM**) subroutine and the Pixel 2 (**GPPXL2**) subroutine three times, specifying that a different component of the frame buffer be unprotected each time.

  If, for example, you have a 24-bit component frame buffer and image data arranged in three groups for red, green, and blue,

  To write the red data:

  ```
  GPFBM(0xff00ffff);
  GPPXL2(red data);
  ```

  To write the green data:

  ```
  GPFBM(0xffff00ff);
  GPPXL2(green data);
  ```

  To write the blue data:

  ```
  GPFBM(0xffffff00);
  GPPXL2(blue data);
  ```

  Restore write protect mask:

  ```
  GPFBM(0x00000000);
  ```

Direct Window Access (DWA) Capabilities on the RS/6000 ONLY

In addition to the general capabilities applying to all adapters:

- Lighting, depth cueing and HLHSR (hidden line, hidden surface removal) are available.
- HLHSR (hidden line, hidden surface removal) can be applied to primitives by setting HLHSR to `2=ON_THE_FLY` if z-buffer is installed.
- There is no polysphere tessellation limitation.
- POWER GTO (8 bit or 24 bit):
  - HLHSR (hidden line, hidden surface removal) identifier 10 (`NOT_EQUAL`) is not supported. HLHSR identifier 2 (`VISUALIZE_IF_HIDDEN`) applies only to line primitives.

– Shading is enabled by setting depth cue mode to 2=ALLOWED or lighting calculation mode to 2=PER_AREA or 3=PER_VERTEX
– Transparency modes of 2=PARTIAL_TRANSPARENT and 3=BLEND are supported, however:
  - When using 2=PARTIAL_TRANSPARENT mode, you must also have the lighting calculation mode set to 2=PER_AREA or 3=PER_VERTEX (transparency mode 3=BLEND does not have this requirement).
  - Concave Polygons, Composite Fill Areas, Trimmed NURB Surfaces, Polygons with edge flag set to 2=ON, and Polygon with Data may be rendered using multiple passes. This multiple rendering includes the 3=BLEND transparency processing. Therefore, some parts (such as shared edges) show the results of this double-blending.
  - Transparency processing is not face-dependent. The transparency coefficient from the front-face surface properties (set by the Set Surface Properties (**GPSPR**) subroutine) is always used.
  - Transparency processing is applied during the rendering of lines and geometric text, as well as area primitives. To prevent lines and text from having transparency applied, the application can reset the transparency coefficient before the line and/or text primitives.
– Antialiasing is supported, however:
  - Antialiasing is not performed when any of the following facilities are also used to render the polylines:
    • HLHSR, depth cueing, transparency, or lighting
    • Wide lines (line width scale factor other than 1.0)
    • Line type other than 1=SOLID_LINE
    • Scaled-to-fit line type rendering
    • Polyline shading method of 2=POLYLINE_SHADING_COLOR
  - Refer to the directory

    /usr/lpp/graPHIGS/samples/antialiasing

    for a sample of antialiasing.
  - Antialiasing may degrade performance due to the additional processing involved. To control degradation, two antialiasing modes are available: 2=SUBPIXEL_ON_THE_FLY and 3=NON_SUBPIXEL_ON_THE_FLY. 2=SUBPIXEL_ON_THE_FLY results in the highest quality rendering but 3=NON_SUBPIXEL_ON_THE_FLY is faster. For correct visual results, your application must use values in the display color table that vary from 0.0 to 1.0 for each color component. For best results, your application should use gamma-corrected color component values. (If you wish to initialize the display color table with a linear ramp of color component values, you can use the Direct Color (DIRCOLOR) procopt. See DIRCOLOR (Direct Color). The antialiasing facility is supported with 8-bit and 24-bit pixel memory. Special consideration is needed for 8-bit pixel memory.
– Pixel data in obstructed window areas can be returned with unexpected results when read from the frame buffer.

• POWER Gt4x (8 bit or 24 bit):
  – Only transparency modes 1=NONE and 2=PARTIAL_TRANSPARENT are supported.
  – Antialiasing is supported, however:
    - Only antialiasing mode 2=SUBPIXEL_ON_THE_FLY is supported.
    - Antialiasing is not performed when any of the following facilities are also used to render the polylines:
      • Wide lines (line width scale factor other than 1.0)
      • Line type has depth cueing set to 2=ALLOWED (this limitation applies to POWER Gt4x 8 bit only).
    - Refer to the directory

      /usr/lpp/graPHIGS/samples/antialiasing

      for a sample of antialiasing.

- Antialiasing may degrade performance due to the additional processing involved. For correct visual results, your application must use values in the display color table that vary from 0.0 to 1.0 for each color component. For best results, your application should use gamma-corrected color component values. (If you wish to initialize the display color table with a linear ramp of color component values, you can use the Direct Color (DIRCOLOR) procopt. See DIRCOLOR (Direct Color). The antialiasing facility is supported with 8-bit and 24-bit pixel memory. Special consideration is needed for 8-bit pixel memory.

XLIB (non-DWA) Capabilities on the RS/6000 ONLY

In addition to the general capabilities supported by all adapters:

- Polysphere Primitives:
  - The data generated by tessellation of polyspheres is limited to 65,000 bytes. If the generated data exceeds this amount, then part of your polysphere will not be displayed.

XSOFT

- Antialiasing is supported only for polyline primitives with nominal line width.
- Lighting, depth cueing and HLHSR (hidden line, hidden surface removal) are available.
- There is no polysphere tessellation limitation.

6090

- BNIL (Before Next Input Locally) deferral mode is treated as BNIG (Before Next Input Globally) deferral mode.
- Only HLHSR (hidden line, hidden surface removal) identifiers 1=VISUALIZE_IF_NOT_HIDDEN, 2=VISUALIZE_IF_HIDDEN, 3=VISUALIZE_ALWAYS, and 4=NOT_VISUALIZE are supported. HLHSR identifier 2 (VISUALIZE_IF_HIDDEN) applies only to line primitives.
- If you are using the optional shading feature:
  - You can process geometrical data if the HLHSR (hidden line, hidden surface removal) is set to 2=ON_THE_FLY, or depth cue mode is set to 2=ALLOWED, or lighting calculation mode is set to 2=PER_AREA or 3=PER_VERTEX.
  - Line width is not supported.
- If you are *not* using the shading feature (HLHSR is set to 1=OFF, depth cue mode is set to 1=SUPPRESSED, and lighting calculation mode is set to 1=NONE), then:
  - line style defaults to 2=SOLID and no shading is performed, if polyline end type is set to 2=ROUND or 3=SQUARE.
  - the boundary flags defined in some primitives are ignored.
- Transparency mode is only applied when transparency mode is set to 2=PARTIAL_TRANSPARENT in the view table entry and lighting calculation mode is set to 2=PER_AREA or 3=PER_VERTEX.

5080

- Only the nominal line width scale for circle, arc, and ellipse GDPs are supported.
- Pixel Primitives and Attributes: Pixel elements cannot be clipped to viewport or window boundaries. When a pixel element extends beyond a boundary, the element is replaced by a rectangle of the same size that is filled with the highlighting color.
- The number of pixel color indexes in one pixel primitive is limited to approximately 32,000.

IMAGE

- Lighting, depth cueing and HLHSR (hidden line, hidden surface removal) are available.
- There is no polysphere tessellation limitation.

GDF

- BNIL (Before Next Input Locally) deferral mode is treated as BNIG (Before Next Input Globally) deferral mode.
- Polysphere Primitives: The data generated by tessellation of polyspheres is limited to 65,000 bytes. If the generated data exceeds this amount, then part of your polysphere will not be displayed.
- Pixel Primitives: Pixel primitives are not supported.

CGM

- BNIL (Before Next Input Locally) deferral mode is treated as BNIG (Before Next Input Globally) deferral mode.
- Polysphere Primitives: The data generated by tessellation of polyspheres is limited to 65,000 bytes. If the generated data exceeds this amount, then part of your polysphere will not be displayed.
- For CGM metafile and picture descriptor element information, see Delimiter Elements. For graphical primitives information, see CGM Elements for Graphical Primitives.

Table 17. General Output Facilities - X Workstation Default Values

General Output Facilities	Data Type	IMAGE	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Display type (1=VECTOR, 2=RASTER, 3=OTHERS)	E	RASTER	RASTER	RASTER	RASTER	**GPQWD** [*type*]
Deferral mode (1=ASAP, 2=BNIG, 3=BNIL, 4=ASTI, 5=WAIT)	E	WAIT	WAIT	WAIT	WAIT	**GPQDDV** [*defer*]
Modification mode (1=NO_IMMEDIATE_VISUAL_EFFECT, 2=UPDATE_WITHOUT_REGENERATION, 3=QUICK_UPDATE)	E	NO_ IMMEDIATE_ VISUAL_ EFFECT	NO_ IMMEDIATE_ VISUAL_ EFFECT	NO_ IMMEDIATE_ VISUAL_ EFFECT	NO_ IMMEDIATE_ VISUAL_ EFFECT	**GPQDDV** [*modif*]
Number of structure priorities supported	I	Cont. range supported[2]	Cont. range supported[2]	Cont. range supported[2]	Cont. range supported[2]	**GPQNSP** [*npri*]
Maximum hierarchical depth	I	32	32	32	16	**GPQHD** [*depth*]
Number of available class names	I	1024	1024	1024	256	**GPQNCN** [*number*]
Shielding support (NO, YES)	E	YES	YES	YES	YES	**GPQVF** [*shield*]
Number of available HLHSR modes	I	2	2	2	1	**GPQHMO** [*totnum*]
HLHSR modes (1=OFF, 2=ON_THE_FLY)	E	OFF, ON_THE_FLY	OFF, ON_THE_FLY	OFF, ON_THE_FLY	OFF	**GPQHMO** [*mode*]
Number of available antialiasing modes	I	3	2* Exception **POWER GTO**: 3*	2*	1	**GPQAMO** [*totnum*]
Available antialiasing modes (1=OFF, 2=SUBPIXEL_ON_THE_FLY, 3=NON_SUBPIXEL_ON_THE_FLY)	E	OFF, SUBPIXEL_ ON_THE_ FLY, NON_ SUBPIXEL_ ON_THE_ FLY*	OFF, SUBPIXEL_ ON_THE_ FLY* Exception **POWER GTO:** OFF, SUBPIXEL_ ON_THE_ FLY, NON_ SUBPIXEL_ ON_THE_ FLY*	OFF, SUBPIXEL_ ON_THE_ FLY, NON_ SUBPIXEL_ ON_THE_ FLY*	OFF	**GPQAMO** [*mode*]

Table 17. General Output Facilities - X Workstation Default Values (continued)

General Output Facilities	Data Type	IMAGE	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Number of available transparency modes	I	4*	4* Exceptions **GT4 Family**: 2* **POWER GTO**: 3*	4*	1	**GPQTMO** [*totnum*]
Transparency modes (1=NONE, 2=PARTIAL_TRANSPARENT, 3=BLEND, 4=BLEND_ALL)	E	NONE, PARTIAL_ TRANS- PARENT, BLEND, BLEND_ALL*	NONE, PARTIAL_ TRANS- PARENT, BLEND, BLEND_ALL* Exceptions **GT4 Family** NONE, PARTIAL_ TRANS- PARENT* **POWER GTO** NONE, PARTIAL_ TRANS- PARENT, BLEND	NONE, PARTIAL_ TRANS- PARENT, BLEND, BLEND_ALL*	NONE	**GPQTMO** [*mode*]
Frame buffer organization (1=COMPONENT, 2=INDEXED)	E	COMPONENT	**24 bit:** COMPONENT	**8 bit:** INDEXED* **24 bit:** COMPONENT	INDEXED	**GPQFBC** [*org*]
Number of available frame buffer components	I	3	**24 bit:** 3	**8 bit:** 1 **24 bit:** 3	1	**GPQFBC** [*n*]
List of bit depths for each frame buffer component	*n*[default]I	8,8,8	**24 bit:** 8,8,8	**8 bit:** 8 **12 bit:** 4,4,4 **24 bit:** 8,8,8	8	**GPQFBC** [*depth*]

Note:

[1] See the text prior to General Workstation Facilities for a list of DWA and XSOFT Adapters.

[2] Continuous range is supported but you will get a 0 back on inquiry.

* See the text prior to this table for more information.

Table 18. General Output Facilities Default Values

General Output Facilities	Data Type	6090	5080	GDDM	GDF	CGM	Inquiry
Display type (1=VECTOR, 2=RASTER, 3=OTHERS)	E	RASTER	RASTER	RASTER	OTHERS	OTHERS	**GPQWD** [*type*]
Deferral mode (1=ASAP, 2=BNIG, 3=BNIL, 4=ASTI, 5=WAIT)	E	WAIT	WAIT	WAIT	WAIT	WAIT	**GPQDDV** [*defer*]
Modification mode (1=NO_ IMMEDIATE_ VISUAL_ EFFECT, 2=UPDATE_ WITHOUT_ REGENERATION, 3=QUICK_ UPDATE)	E	NO_ IMMEDIATE_ VISUAL_ EFFECT	NO_ IMMEDIATE_ VISUAL_ EFFECT	NO_ IMMEDIATE_ VISUAL_ EFFECT	NO_ IMMEDIATE_ VISUAL_ EFFECT	NO_ IMMEDIATE_ VISUAL_ EFFECT	**GPQDDV** [*modif*]
Number of structure priorities supported	I	Cont. range supported[1]	Cont. range supported[1]	Cont. range supported[1]	Cont. range supported[1]	Cont. range supported[1]	**GPQNSP** [*npri*]

Table 18. General Output Facilities Default Values (continued)

General Output Facilities	Data Type	6090	5080	GDDM	GDF	CGM	Inquiry
Maximum hierarchical depth	I	32	16	16	16	16	**GPQHD** [*depth*]
Number of available class names	I	1024	256	256	256	256	**GPQNCN** [*number*]
Shielding support (NO, YES)	E	YES	YES	YES	YES	YES	**GPQVF** [*shield*]
Number of available HLHSR modes	I	2	1	1	1	1	**GPQHMO** [*totnum*]
HLHSR modes (1=OFF, 2=ON_THE_FLY)	E	OFF, ON_ THE_ FLY	OFF	OFF	OFF	OFF	**GPQHMO** [*mode*]
Number of available antialiasing modes	I	1	1	1	1	1	**GPQAMO** [*totnum*]
Available antialiasing modes (1=OFF, 2=SUBPIXEL_ ON_ THE_ FLY, 3=NON_ SUBPIXEL_ ON_ THE_ FLY)	E	OFF	OFF	OFF	OFF	OFF	**GPQAMO** [*mode*]
Number of available transparency modes	I	2*	1	1	1	1	**GPQTMO** [*totnum*]
Transparency modes (1=NONE, 2=PARTIAL_ TRANSPARENT, 3=BLEND, 4=BLEND_ ALL)	E	NONE, PARTIAL_ TRANSPARENT	NONE	NONE	NONE	NONE	**GPQTMO** [*mode*]
Frame buffer organization (1=COMPONENT, 2=INDEXED)	E	COMPONENT*	INDEXED	INDEXED	INDEXED	INDEXED	**GPQFBC** [*org*]
Number of available frame buffer components	I	3*	1	1	1	1	**GPQFBC** [*n*]
List of bit depths for each frame buffer component	*n*[default]I	8,8,8*	7*	4*	4	8	**GPQFBC** [*depth*]

Note:

[1]Continuous range is supported but you will get a 0 back on inquiry.

* See text prior to this table for more information.

Polyline Facilities

X

General Information Applying to All Adapters

- Nominal, minimum, and maximum line widths and the size of the line pattern unit depend on the hardware configuration of your workstation. Use the Inquire Polyline Facilities (**GPQPLF**) subroutine to obtain the values supported on your workstation.
- You can set the number of polyline table entries that can be active at any one time up to 128 via the Polyline Bundle Table (PLBTES) procopt. See PLBTES (Polyline Bundle Table).

Direct Window Access (DWA) Capabilities on the RS/6000 ONLY

In addition to the general capabilities supported by all adapters:

- When drawing a wide line with a line type other than 1=SOLID_LINE, the line end type is always 1=FLAT.
- All DWA Adapters except the POWER GT4 Family and the POWER GTO:
  - The line pattern is restarted at each vertex for wide lines that are greater than one pixel.

XLIB (non-DWA) Capabilities on the RS/6000 ONLY

In addition to the general capabilities supported by all adapters:

- Color Graphics Display Adapter:
  - The line pattern unit is the same as the nominal line width.

– No line widths other than the nominal line width are supported.

XSOFT

- Nominal, minimum, and maximum line widths and the size of the line pattern unit depend on the hardware configuration of your workstation. Use the Inquire Polyline Facilities (**GPQPLF**) subroutine to obtain the values supported on your workstation.
- You can set the number of polyline table entries that can be active at any one time up to 128 via the Polyline Bundle Table (`PLBTES`) procopt. See PLBTES (Polyline Bundle Table).
- The line pattern is restarted at each vertex for wide lines that are greater than one pixel.
- When drawing a wide line with a line type other than `1=SOLID_LINE`, the line end type is always `1=FLAT`.

6090

- You can set the number of polyline table entries that can be active at any one time up to 128 via the Polyline Bundle Table (`PLBTES`) procopt. See PLBTES (Polyline Bundle Table).
- Nominal, minimum, and maximum line widths depend on the hardware configuration of your workstation. Use the Inquire Polyline Facilities (**GPQPLF**) subroutine to obtain the values supported on your workstation.
- When an end type of `2=ROUND` or `3=SQUARE` is applied to wide lines, only a line type of `1=SOLID_LINE` is supported.

5080

- Nominal, minimum, and maximum line widths depend on the hardware configuration of your workstation. Use the Inquire Polyline Facilities (**GPQPLF**) subroutine to obtain the values supported on your workstation.
- The number of vertexes in one polyline primitive is limited to approximately 5,000 for two-dimensional and 4,000 for three-dimensional polyline primitives.

GDDM*

Nominal, minimum, and maximum line widths depend on the hardware configuration of your workstation. Use the Inquire Polyline Facilities (**GPQPLF**) subroutine to obtain the values supported on your workstation.

IMAGE

Nominal, minimum, and maximum line widths and the size of the line pattern unit depend on the hardware configuration of your workstation. The nominal line is the width of the pixel which can be modified by using the DCUNITS or DCMETERS procopts.

GDF

The nominal line width is 0.000269 meters regardless of how the GDF file is scaled (i.e., the nominal line width for the graPHIGS API is independent of the metric scale factor of the resultant GDF file). This results in a line width of 0.96 Virtual Device Coordinates (VDC). The effective maximum line width is the size of the workstation display area (0.2582728 meters). However, the size of the display area can be modified using the Escape (**GPES**) subroutine.

Wide lines are clipped to their center.

CGM

The nominal line width is 0.000258 meters regardless of how the CGM file is scaled (i.e., the nominal line width for the graPHIGS API is independent of the metric scale factor of the resultant CGM file). This results in a line width of 0.96 Virtual Device Coordinates (VDC). See Picture Descriptor Elements for more

information about VDCs. The effective maximum line width is the size of the workstation display area (0.2582728 meters). However, the size of the display area can be modified using the Escape (**GPES**) subroutine.

Wide lines are clipped to their center.

For CGM picture descriptor element information on line width specification mode, see Picture Descriptor Elements.

For CGM attribute information on line types, see CGM Line Attributes.

Table 19. Polyline Facilities - X Workstation Default Values

Polyline Facilities	Data Type	IMAGE	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Availability of line type representation (1=NOT_AVAILABLE, 2=BOTH_AVAILABLE, 3=INQUIRE_ONLY_ AVAILABLE, 4=SET_ONLY_AVAILABLE)	E	BOTH_ AVAILABLE	BOTH_ AVAILABLE	BOTH_ AVAILABLE	BOTH_ AVAILABLE	**GPQLTF** [*available*]
Size of line pattern unit (in meters)	R	0.00135466*	0.001328125*	0.001328125*	0.00033*	**GPQLTF** [*unit*]
Maximum number of line sections	I	8	8	8	8	**GPQLTF** [*sections*]
Maximum length of line pattern	I	256	256	256	256	**GPQLTF** [*maxlen*]
Number of available line rendering styles	I	2	2	2	2	**GPQLNR** [*totnum*]
Available line rendering styles (1=WORKSTATION_ DEPENDENT_RENDERING, 2=SCALED_TO_FIT_ RENDERING)	E	WORK- STATION_ DEPENDENT_ RENDERING, SCALED_TO_ FIT_ RENDERING	WORK- STATION_ DEPENDENT_ RENDERING, SCALED_TO_ FIT_ RENDERING	WORK- STATION_ DEPENDENT_ RENDERING, SCALED_TO_ FIT_ RENDERING	WORK- STATION_ DEPENDENT_ RENDERING, SCALED_TO_ FIT_ RENDERING	**GPQLNR** [*rstyle*]
Number of available line types	I	16	16	16	16	**GPQPLF** [*ntype*]
Available line types (see Table 15)	E	1-16	1-16	1-16	1-16	**GPQPLF** [*ltype*]
Number of available line widths	I	Cont. range supported[2] *	Cont. range supported[2] *	Cont. range supported[2] *	1	**GPQPLF** [*nlwidth*]
Nominal line width (in meters)	R	0.0003386*	0.000332031*	0.000332031*	0.00033*	**GPQPLF** [*lwidth*]
Minimum line width (in meters)	R	0.0003386*	0.000332031*	0.000332031*	0.00033*	**GPQPLF** [*minlw*]
Maximum line width (in meters)	R	0.215*	0.425*	0.425*	0.00033*	**GPQPLF** [*maxlw*]
Maximum number of polyline bundle table entries	I	128*	128*	128*	128*	**GPQLW** [*ltable*]
Number of predefined polyline bundle table entries	I	6	6	6	6	**GPQPLF** [*npred*]

Note:

[1]See the text prior to General Workstation Facilities for a list of DWA and XSOFT Adapters.

[2] Continuous range is supported but you will get a 0 back on inquiry.

* See the text prior to this table for more information.

Table 20. Polyline Facilities Default Values

Polyline Facilities	Data Type	6090	5080	GDDM	GDF	CGM	Inquiry
Availability of line type representation (NONE, BOTH, INQUIRE ONLY, SET ONLY)	E	BOTH	BOTH	NONE	BOTH	BOTH	**GPQLTF** [*available*]
Size of line pattern unit (in meters)	R	0.001328125*	0.00111121	N/A	0.001328125	0.001328125*	**GPQLTF** [*unit*]

Table 20. Polyline Facilities Default Values (continued)

Polyline Facilities	Data Type	6090	5080	GDDM	GDF	CGM	Inquiry
Maximum number of line sections	I	8	8	N/A	8	8	**GPQLTF** [*sections*]
Maximum length of line pattern	I	256	256	N/A	256	256	**GPQLTF** [*maxlen*]
Number of available line rendering styles	I	2	1	1	2	2	**GPQLNR** [*totnum*]
Available line rendering styles (1=WORKSTATION_ DEPENDENT_ RENDERING, 2=SCALED_ TO_ FIT_ RENDERING)	E	WORK-STATION_ DEPENDENT_ RENDERING, SCALED_TO_ FIT_ RENDERING	WORK-STATION_ DEPENDENT_ RENDERING	WORK-STATION_ DEPENDENT_ RENDERING	WORK-STATION_ DEPENDENT_ RENDERING, SCALED_TO_ FIT_ RENDERING	WORK STATION_ DEPENDENT_ RENDERING, SCALED_TO_ FIT_ RENDERING	**GPQLNR** [*rstyle*]
Number of available line types	I	16*	13	7	16	16	**GPQPLF** [*ntype*]
Available line types (see Table 15)	E	1-16*	1-13	1-7	1-16	1-16*	**GPQPLF** [*ltype*]
Number of available line widths	I	1024*	1024*	2	Cont. range supported[1]	Cont. range supported[1]	**GPQPLF** [*nlwidth*]
Nominal line width (in meters)	R	0.000332031*	0.0002778*	0.0003428*	0.000269*	0.000258*	**GPQPLF** [*lwidth*]
Minimum line width (in meters)	R	0.000332031*	0.0002778*	0.0003428*	0.000269*	0.000258*	**GPQPLF** [*minlw*]
Maximum line width (in meters)	R	0.425*	0.28448*	0.0006856*	0.2582728*	0.2582728*	**GPQPLF** [*maxlw*]
Maximum number of polyline bundle table entries	I	128*	20	20	20	20	**GPQLW** [*ltable*]
Number of predefined polyline bundle table entries	I	6	6	6	6	6	**GPQPLF** [*npred*]

Note:

[1] Continuous range is supported but you will get a 0 back on inquiry.

* See the text prior to this table for more information.

Table 21. Predefined Polyline Bundle Table

Entry	Line Type	Line Width Scale Factor	Color Type	Color Index
1	SOLID_LINE	1.0	INDEXED	1
2	DASHED	1.0	INDEXED	2
3	DOTTED	1.0	INDEXED	3
4	DASH_DOT	1.0	INDEXED	4
5	SOLID_LINE	1.0	INDEXED	5
6	DASHED	1.0	INDEXED	6

Table 22. Predefined Line Types

Entry	Line Type
1	SOLID_LINE
2	DASHED
3	DOTTED
4	DASH_DOT
5	LONG_DASH
6	DOUBLE_DOT

Table 22. Predefined Line Types (continued)

Entry	Line Type
7	DASH_DOUBLE_DOT
8- *n*	SOLID_LINE

Polymarker Facilities

X

General Information Applying to All Adapters

- Nominal, minimum, and maximum marker sizes depend on the hardware configuration of your workstation. Use the Inquire Polymarker Facilities (**GPQPMF**) subroutine to obtain the values supported on your workstation.
- You can set the number of polymarker bundle table entries that may be active at any one time up to 128 via the Polymarker Bundle Table (PMBTES) procopt. See PMBTES (Polymarker Bundle Table).

6090

Nominal, minimum, and maximum marker sizes depend on the hardware configuration of your workstation. Use the Inquire Polymarker Facilities (**GPQPMF**) subroutine to obtain the values supported on your workstation.

You can set the number of polymarker bundle table entries that may be active at any one time up to 128 via the Polymarker Bundle Table (PMBTES) procopt. See PMBTES (Polymarker Bundle Table).

5080

Nominal, minimum, and maximum marker sizes depend on the hardware configuration of your workstation. Use the Inquire Polymarker Facilities (**GPQPMF**) subroutine to obtain the values supported on your workstation.

The number of polymarkers in one polymarker primitive is limited to approximately 8,000 for the 2D primitive and 5,000 for the 3D primitive.

The marker size scale factors supported are:

- 1.50 (5080 large markers)
- 1.25 (5080 medium markers)
- 1.00 (5080 basic markers)
- 0.75 (5080 small markers)

Nominal, minimum, and maximum marker sizes depend on the hardware configuration of your workstation. Use the Inquire Polymarker Facilities (**GPQPMF**) subroutine to obtain the values supported on your workstation.

CGM

The nominal marker width is 0.00258 meters regardless of how the CGM file is scaled (i.e., the nominal marker width for the graPHIGS API is independent of the metric scale factor of the resultant CGM file). The effective maximum marker width is the size of the workstation display area (0.2582728 meters). However, the size of the display area can be modified using the Escape (**GPES**) subroutine.

Polymarkers are represented as a series of CGM polylines. For more information, see line attributes CGM Line Attributes.

Table 23. Polymarker Facilities - X Workstation Default Values

Polymarker Facilities	Data Type	IMAGE	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Availability of marker type representation (1=NOT_AVAILABLE, 2=BOTH_AVAILABLE, 3=INQUIRE_ONLY_ AVAILABLE, 4=SET_ONLY_AVAILABLE)	E	BOTH_ AVAILABLE	BOTH_ AVAILABLE	BOTH_ AVAILABLE	BOTH_ AVAILABLE	**GPQMTF** [*available*]
Marker definition format (VECTOR)	E	VECTOR	VECTOR	VECTOR	VECTOR	**GPQMTF** [*format*]
Maximum length of marker definition data (number of strokes)	I	64	64	64	64	**GPQMTF** [*maxlen*]
Number of available marker types	I	16	16	16	16	**GPQPMF** [*ntype*]
Available marker types (see Table 19)	E	1-16	1-16	1-16	1-16	**GPQPMF** [*mtype*]
Number of available marker sizes	I	Continuous range supported[2]	Continuous range supported[2]	Continuous range supported[2]	Continuous range supported[2]	**GPQPMF** [*nsize*]
Nominal marker size (in meters)	R	0.004064*	0.006640625*	0.006640625*	0.006640625*	**GPQPMF** [*size*]
Minimum marker size (in meters)	R	0.00033866*	0.000332031*	0.000332031*	0.000332031*	**GPQPMF** [*minms*]
Maximum marker size (in meters)	R	0.215*	0.425*	0.425*	0.425*	**GPQPMF** [*maxms*]
Maximum number of polymarker bundle table entries	I	128*	128*	128*	128*	**GPQLW** [*mtable*]
Number of predefined polymarker bundle table entries	I	6	6	6	6	**GPQPMF** [*npred*]

Note:

[1]See the text prior to General Workstation Facilities table for a list of DWA and XSOFT Adapters.

[2] Continuous range is supported but you will get a 0 back on inquiry.

* See the text prior to this table for more information.

Table 24. Polymarker Facilities Default Values

Polymarker Facilities	Data Type	6090	5080	GDDM	GDF	CGM	Inquiry
Availability of marker type representation (1=NOT_ AVAILABLE, 2=BOTH_ AVAILABLE, 3=INQUIRE_ ONLY_ AVAILABLE, 4=SET_ ONLY_ AVAILABLE)	E	BOTH_ AVAILABLE	NOT_ AVAILABLE	NOT_ AVAILABLE	BOTH_ AVAILABLE	BOTH_ AVAILABLE	**GPQMTF** [*available*]
Marker definition format (1=VECTOR)	E	VECTOR	N/A	N/A	VECTOR	VECTOR	**GPQMTF** [*format*]
Maximum length of marker definition data (number of strokes)	I	64	N/A	N/A	64	64	**GPQMTF** [*maxlen*]
Number of available marker types	I	16	5	5	16	16	**GPQPMF** [*mtype*]
Available marker types (see Table 19)	E	1-16	1-5	1-5	1-16	1-16*	**GPQPMF** [*mtype*]
Number of available marker sizes	I	Continuous range supported[1]	4	1	Continuous range supported[1]	Continuous range supported[1]	**GPQPMF** [*nsize*]
Nominal marker size (in meters)	R	0.006640625*	0.0055562*	0.0030852*	0.0053806	0.00258*	**GPQPMF** [*size*]
Minimum marker size (in meters)	R	0.000332031*	0.0041671*	0.0030852*	0.000538	0.000258*	**GPQPMF** [*minms*]

Table 24. Polymarker Facilities Default Values (continued)

Polymarker Facilities	Data Type	6090	5080	GDDM	GDF	CGM	Inquiry
Maximum marker size (in meters)	R	0.425*	0.0083343*	0.0030852*	0.2582728	0.2582728*	**GPQPMF** [*maxms*]
Maximum number of polymarker bundle table entries	I	128*	20	20	20	20	**GPQLW** [*mtable*]
Number of predefined polymarker bundle table entries	I	6	6	6	6	6	**GPQPMF** [*npred*]

Note:

[1] Continuous range is supported but you will get a 0 back on inquiry.

* See the text prior to this table for more information.

Table 25. Predefined Polymarker Bundle Table

Table Entry	Marker Type	Marker Size Scale Factor	Color Type	Color Index
1	DOT	1.0	INDEXED	1
2	PLUS_SIGN	1.0	INDEXED	2
3	ASTERISK	1.0	INDEXED	3
4	CIRCLE	1.0	INDEXED	4
5	DIAGONAL_CROSS	1.0	INDEXED	5
6	DOT	1.0	INDEXED	6

Table 26. Predefined Marker Types

Table Entry	Marker Type
1	DOT
2	PLUS_SIGN
3	ASTERISK
4	CIRCLE
5	DIAGONAL_CROSS
6- *n*	ASTERISK

Text Facilities

X

General Information Applying to All Adapters

You can set the number of text bundle entries that may be active at any one time up to 128 via the Text Bundle Table (TXBTES) procopt. See TXBTES (Text Bundle Table).

Annotation Text Primitives and Attributes:

- Annotation up vector, annotation alignment, and character positioning mode are not supported.
- Character precision (CHAR_PREC) is supported the same as string precision (STRING_PREC) with the additional attribute support for annotation text path, character spacing, and character expansion factor.
- Only font 1 of the primary character set is available. This may be character set 6, 8, 9, 10, 11 or 12. For a discussion of how to set the primary character set, see Opening the X Workstation.

- Kanji (character set identifiers 128 and 134), Hangul (character set identifier 129), Traditional Chinese (character set identifier 130), Simplified Chinese (character set identifier 132) and Unicode (character set identifier 131) are not supported.

Geometric Text Primitives and Attributes:

- Geometric proportional fonts are supported.

Direct Window Access (DWA) Capabilities on the RS/6000 ONLY

In addition to the general capabilities supported by all adapters:

- All annotation text attributes are supported.
- Geometric text culling is supported for 2D and 3D primitives.
- All DWA Adapters except the POWER GTO:
  - Character precision (CHAR_PREC) produces the same result as stroke precision (STROKE_PREC).

XLIB (non-DWA) Capabilities on the RS/6000 ONLY

In addition to the general capabilities supported by all adapters:

- Geometric Text Primitives and Attributes:
  - Geometric text culling is supported for 2D primitives.

XSOFT

Character precision (CHAR_PREC) produces the same result as stroke precision (STROKE_PREC).

6090

Annotation Text Primitives and Attributes:

- Character precision (CHAR_PREC) produces the same result as (STROKE_PREC).
- Only font 1 of the primary character set is available. This may be character set 6 or 8.
- Kanji (character set identifiers 128 and 134), Hangul (character set identifier 129), Traditional Chinese (character set identifier 130), Simplified Chinese (character set identifier 132), and Unicode (character set identifier 131) are not supported.

Geometric Text Primitives and Attributes:

- You can set the number of text bundle entries that may be active at any one time up to 128 via the Text Bundle Table (TXBTES) procopt. See TXBTES (Text Bundle Table).
- All geometric text attributes are supported.
- Character precision (CHAR_PREC) produces the same result as stroke precision (STROKE_PREC).
- Hangul (character set identifier 129), Traditional Chinese (character set identifier 130), Simplified Chinese (character set identifier 132), Kanji (character set identifier 134 only), and Unicode (character set identifier 131) are not supported.

5080

Annotation Text Primitives and Attributes:

- The number of characters in one annotation text primitive is limited to approximately 32,000.
- Annotation text relative is not supported.
- Character positioning mode is ignored; the 5080 supports only constant-sized annotation characters. Character expansion factor, annotation up vector, and annotation alignment are not supported.
- Character precision (CHAR_PREC) is the highest annotation precision supported. If you specify stroke precision (STROKE_PREC), then CHAR_PREC is used.
- Only font 1 of the primary character set is available. This may be character set 1 through 7, or 9 depending on the current customized language feature of the 5080.

- Kanji (character set identifiers 128 and 134), Hangul (character set identifier 129), Traditional Chinese (character set identifier 130), Simplified Chinese (character set identifier 132), and Unicode (character set identifier 131) are not supported.
- The annotation height scale factors supported by the 5080 are:
  - 1.50 (5080 large characters)
  - 1.25 (5080 medium characters)
  - 1.00 (5080 basic characters)
  - 0.75 (5080 small characters)

Geometric Text Primitives and Attributes:

- The default number of geometric character sets that may be active at any one time is three. You may set this number to any value up to ten via the Font Pool Size (FONTPSIZ) procopt. See FONTPSIZ (Font Pool Size).
- The number of characters in one geometric text primitive is limited to approximately 32,000.
- Character precision (CHAR_PREC) produces the same result as stroke precision (STROKE_PREC).
- Character positioning mode is not supported.
- For user-defined character sets, the default character is ignored; it is always the hyphen. For additional information on limitations for user-defined character sets, see IBM 5080 Character Set Restrictions.
- Kanji: If you use Kanji (character set identifier 128), it must be available in the workstation prior to invocation of the graPHIGS API using the 5080 Japanese Language Feature. For information on how to customize your 5080 to support Kanji, see *The GDDM/graPHIGS Programming Interface: Installation and Problem Diagnosis*.

  On opening the workstation, if the graPHIGS API finds a valid memory area for output Kanji then output Kanji is made available for output to the workstation (although the application must still activate it). If the Kanji output and input memory areas are detected, and Katakana is the primary character set, then input is made available for input from the workstation. After opening the workstation, the Kanji font is never deleted.

  Character set identifier 134, the IBM-943 encoded Kanji support, is not supported.
- Hangul: If you use Hangul (character set identifier 129), it must be available in the workstation prior to invocation of the graPHIGS API using the 5080 Korean Language Feature. For information on how to customize your 5086 to support Hangul, see *The GDDM/graPHIGS Programming Interface: Installation and Problem Diagnosis*.

  On opening the workstation, if the graPHIGS API finds a valid memory area for output Hangul then output Hangul is made available for output to the workstation (although the application must still activate it). If the Hangul output and input memory areas are detected, and csid 9 is the primary character set, then Hangul is made available for input from the workstation. After opening the workstation, the Hangul font is never deleted.
- Traditional Chinese: Traditional Chinese (character set identifier 130) is not supported.
- Simplified Chinese: Simplified Chinese (character set identifier 132) is not supported.
- Unicode: Unicode (character set identifier 131) is not supported.

GDDM

Annotation Text Primitives and Attributes:

- Character expansion factor, annotation up vector, annotation alignment and character positioning mode are not supported.
- Annotation text relative is not supported.
- Character precision (CHAR_PREC) is the highest precision supported. If you specify stroke precision (STROKE_PREC), then CHAR_PREC is used.
- Only font 1 of the primary character set is available.
- The number of available annotation height size scale factors is one.

Geometric Text Primitives and Attributes:

- The number of characters in one geometric text primitive is limited to approximately 32,000.
- In CHAR_PREC and STRING_PREC precision, characters are clipped on a character basis. If any part of a character is clipped, the entire character is not visible.
- Traditional Chinese (character set identifier 130) is not supported.
- Simplified Chinese (character set identifier 132) is not supported.
- The Kanji function using the IBM-943 encoding (character set identifier 134) is not supported.
- Unicode (character set identifier 131) is not supported.

IMAGE

Character precision (CHAR_PREC) produces the same result as stroke precision (STROKE_PREC).

GDF

Annotation Text Primitives and Attributes:

- Character expansion factor, annotation up vector, annotation alignment, and character positioning mode are not supported.
- Character precision (CHAR_PREC) produces the same results as stroke precision (STROKE_PREC).
- Only font 1 of the primary character set is available.
- Character code points are written directly into the CGM file. The characters drawn depend on your interpreter. The output metafile does not specify a character set or font. It uses the default of the interpreter.
- The number of available annotation height scale factors is one. The size is determined by the output system default.
- Pixel primitives and annotation text are not clipped to the exterior of obscuring views, even if the obscuring view has view shielding set to 2=ON.

Geometric Text Primitives and Attributes:

- Traditional Chinese (character set identifier 130) is not supported.
- Unicode (character set identifier 131) is not supported.
- Simplified Chinese (character set identifier 132) is not supported.
- The Kanji function using the IBM-943 encoding (character set identifier 134) is not supported.

CGM

Annotation Text Primitives and Attributes:

- Character precision (CHAR_PREC) produces the same results as stroke precision (STROKE_PREC).
- Available annotation text character sets supported depend on the configuration of your workstation. Use the Inquire Primary Character Set (**GPQPCS**) subroutine to obtain the values supported on your workstation.
- Only font 1 of the primary character set is available.
- Character code points are written directly into the CGM file. If the nucleus is running in an EBCDIC environment, then code points are first translated to ASCII using the translation table for character set 1. The characters drawn depend on your interpreter. The output metafile does not specify a character set or font. It uses the default of the interpreter.
- Kanji (character set identifiers 128 and 134), Hangul (character set identifier 129), Traditional Chinese (character set identifier 130), Simplified Chinese (character set identifier 132) and Unicode (character set identifier 131) are not supported.
- The nominal height scale factor is 1/100 of the default display surface size. Use the Escape Subroutine (1010: Inquire Mapped Display Surface Size) to determine the size of your display. An annotation height

scale factor of zero would result in 0.0000001* nominal height which is minute in size (i.e., a point). For more information on scaling, see scaling mode Picture Descriptor Elements.

- Pixel primitives and annotation text are not clipped to the exterior of obscuring views, even if the obscuring view has view shielding set to 2=ON.
- For CGM attribute information on text, see CGM Text Attributes.

Geometric Text Primitives and Attributes:

- Traditional Chinese (character set identifier 130) is not supported.
- Unicode (character set identifier 131) is not supported.
- Simplified Chinese (character set identifier 132) is not supported.
- The Kanji function using the IBM-943 encoding (character set identifier 134) is not supported.

Table 27. Text Facilities - X Workstation Default Values

Text Facilities	Data Type	IMAGE	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Maximum number of geometric character set IDs/fonts that may be activated to a workstation	I	20	20	20	20	20	**GPQFP** [*poolsize*]
Geometric filled font support (1=YES, 2=NO)	E	NO	YES	YES	YES	NO	**GPQXTX** [*filled*]
Geometric proportional font support (1=YES, 2=NO)	E	YES	YES	YES	YES	YES	**GPQXTX** [*proportional*]
Number of annotation styles	I	2	2	2	2	2	**GPQANF** [*totnum*]
Available annotation styles (1=UNCONNECTED, 2=LEAD_LINE)	E	UNCONNECTED, LEAD_LINE	UNCONNECTED, LEAD_LINE	UNCONNECTED, LEAD_LINE	UNCONNECTED, LEAD_LINE	UNCONNECTED, LEAD_LINE	**GPQANF** [*styles*]
Maximum number of text bundle table entries	I	128*	128*	128*	128*	128*	**GPQLW** [*ttable*]
Number of predefined text entries	I	6	6	6	6	6	**GPQXTX** [*npred*]
Available annotation text character sets	E	6, 8, 9, 10*	6, 8, 9, 10*	6, 8, 9, 10, 11, 12 [2,*]	6, 8, 9, 10, 11,12*	6, 8, 9, 10, 11,12*	**GPQPCS** [*csid*]
Available font of primary character set for annotation text	E	Font 1 only	Font 1 only	Font 1 only	Font 1 only	Font 1 only	**GPQFO** [*font*]

Note:

[1]See General Workstation Facilities for a list of DWA and XSOFT Adapters.

[2] Limited support may be available for additional fonts.

*See text prior to this table for more information.

Table 28. Text Facilities Default Values

Text Facilities	Data Type	6090	5080	GDDM	GDF	CGM	Inquiry
Maximum number of geometric character set IDs/fonts that may be activated to a workstation	I	No limit[1]	10*	20	20	20	**GPQFP** [*poolsize*]
Geometric filled font support (YES, NO)	E	YES	NO	NO	NO	NO	**GPQXTX** [*filled*]
Geometric proportional font support (YES, NO)	E	YES	NO	NO	NO	NO	**GPQXTX** [*proportional*]
Number of annotation styles	I	2	1	1	2	2	**GPQANF** [*totnum*]
Available annotation styles (1=UNCONNECTED, 2=LEAD_LINE)	E	UNCONNECTED, LEAD_LINE	UNCONNECTED	UNCONNECTED	UNCONNECTED, LEAD_LINE	UNCONNECTED, LEAD_LINE	**GPQANF** [*styles*]

Table 28. Text Facilities Default Values (continued)

Text Facilities	Data Type	6090	5080	GDDM	GDF	CGM	Inquiry
Maximum number of text bundle table entries	I	128*	20	20	20	20	**GPQLW** [*ttable*]
Number of predefined text entries	I	6	6	6	6	6	**GPQXTX** [*npred*]
Available annotation text character sets	E	6, 8	1-7, 9*	1	1	1,8*	**GPQPCS** [*csid*]
Available font of primary character set for annotation text	E	Font 1 only	Font 1 only	Font 1 only	Font 1 only	Font 1 only	**GPQFO** [*font*]
Note: [1] "No limit" returns 0 on the inquiry. * See text prior to this table for more information.							

Table 29. Predefined Text Bundle

Entry	Font	Precision	Expansion	Spacing	Color Type	Color
1	1	STRING_PREC	1.0	0.0	INDEXED	1
2	1	STRING_PREC	1.0	0.0	INDEXED	2
3	1	STRING_PREC	1.0	0.0	INDEXED	3
4	1	STRING_PREC	1.0	0.0	INDEXED	4
5	1	STRING_PREC	1.0	0.0	INDEXED	5
6	1	STRING_PREC	1.0	0.0	INDEXED	6

Interior Facilities

For performance reasons, some graPHIGS API workstations choose not to draw the boundary of a polygon if the edge flag is set to 2=ON. This may avoid drawing the polygon outline twice. If you use a non-solid edge type, then you may get different output on different devices, since the boundary color may or may not be visible between the edge segments.

X

General Information Applying to All Adapters

You can set the number of interior bundle table entries that may be active at any one time up to 128 via the Interior Bundle Table (IBTES) procopt. See IBTES (Interior Bundle Table).

XLIB (non-DWA) Capabilities on the RS/6000 ONLY

In addition to the general capabilities supported by all adapters:

- Availability of hatch representation is NONE.
- There are six available hatch styles.
- Available hatch styles are 1-6.
- There is no hatch definition format.
- There is no pattern definition format.
- There are no pattern indexes.
- There are a total of four interior styles.
- The available interior styles are 1=HOLLOW, 2=SOLID, 4=HATCH, 5=EMPTY.
- Hatch styles 21 and 22 are predefined as vertical lines. These hatch styles are similar in appearance to hatch style 1.

6090

Patterns are not transformable.

If you are using the shading feature (HLHSR is set to 2=ON_THE_FLY, or depth cue mode is set to 2=ALLOWED, or lighting calculation is set to 2=PER_AREA or 3=PER_AREA), then interior styles 1=HOLLOW, 3=PATTERN and 4=HATCH are not supported. Interior style 1=HOLLOW defaults to 5=EMPTY. Interior styles 3=PATTERN and 4=HATCH default to 2=SOLID.

You can set the number of interior bundle table entries that may be active at any one time up to 128 via the Interior Bundle Table (IBTES) procopt. See IBTES (Interior Bundle Table).

5080

The number of vertices in one polygon primitive is variable and depends on factors such as the coordinate distances when converted to 5080 coordinate units and the number of subareas. When subareas are used, the number of vertexes supported decreases by the number of subareas. The area fill is limited to 350 polygon points; if this maximum is exceeded, then the graPHIGS API does not fill the polygon.

Hatch styles 21 and 22 are predefined as vertical lines. These hatch styles are similar in appearance to hatch style 1.

For an interior style of 4=HATCH, the polygon is first filled using color index 0 and then the hatch pattern is overlaid in the specified color. In other words, 4=HATCH interior style is similar to 3=PATTERN in that you cannot see primitives behind the hatching pattern.

Patterns are not transformable.

When the projection type is set to 2=PERSPECTIVE, then interior styles 2=SOLID, 3=PATTERN, and 4=HATCH remain in parallel projection. They are always displayed using 1=PARALLEL projection type. For example, your application defines a polygon with an interior style, with edge flag set to 2=ON and with projection type set to 1=PARALLEL. If you change the projection type from 1=PARALLEL to 2=PERSPECTIVE, then the graPHIGS API displays only the edge in perspective. The filled interior style does not change.

The graPHIGS API clips polygon edges in the *z* direction; however, the graPHIGS API does not clip polygon interiors.

GDDM

Patterns are not transformable.

Interior style 3=PATTERN is not available on plotter devices; 1=HOLLOW is used instead. Only a monochrome pattern is available on 3270 PC/G, GX devices.

The maximum pattern size depends on the device cells. The Set Pattern Representation (**GPPAR**) subroutine function repeats or truncates the specified pattern to achieve a maximum pattern.

GDF

All hatch styles are supported. However, the hatch displayed is dependent on the post-processing routine and device.

Many interpreters do not support patterns, especially when the output device is a post-processing device, such as a pen plotter. When patterns are supported, the pattern definition is not preserved in the file. The pattern displayed is dependent on the post-processing routine and device.

Device	Size
3270-PC/G	8 by 15

Device	Size
3270-PC/GX	8 by 15
3279-Display	9 by 16

CGM

There is no limit on the maximum pattern size (*x* size, *y* size). A logical limit would be 960[default]960, but this may exceed memory constraints.

Many interpreters do not support patterns, especially when the output device is a post-processing device, such as a pen plotter.

For CGM attribute information on hatch and interior styles, see CGM Interior Attributes.

Table 30. Interior Facilities - X Workstation Default Values

Interior Facilities	Data Type	IMAGE	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Availability of hatch representation (1=NOT_AVAILABLE, 2=BOTH_AVAILABLE, 3=INQUIRE_ONLY_AVAILABLE, 4=SET_ONLY_AVAILABLE)	E	BOTH_ AVAILABLE	BOTH_ AVAILABLE	BOTH_ AVAILABLE	BOTH_ AVAILABLE	NOT_ AVAILABLE	**GPQHF** [*available*]
Number of available hatch styles	I	24	24	24	24	6	**GPQIF** [*hatnum*]
Available hatch styles	I	1-24	1-24*	1-24	1-24	1-6	**GPQIF** [*hatch*]
Hatch definition format (1=BIT_ARRAY)	E	BIT_ARRAY	BIT_ARRAY	BIT_ARRAY	BIT_ARRAY	N/A	**GPQHF** [*format*]
Maximum length hatch definition data	E	136	136	136	136	N/A	**GPQHF** [*maxlen*]
Pattern definition format (1-byte integer array)	E	1-byte	1-byte	1-byte	1-byte	N/A	
Maximum pattern size (*x* size, *y* size)	2xI	32x32	32x32	32x32	32x32	N/A	**GPQPAF** [*maxrow, maxcol*]
Maximum number of pattern indexes	I	4	4	4	4	N/A	**GPQLW** [*pttable*]
Number of predefined pattern indexes	I	2	2	2	2	0	**GPQPAF** [*indexes*]
Number of available interior styles	I	5	5	5	5	4	**GPQIF** [*intnum*]
Available interior styles (1=HOLLOW, 2=SOLID, 3=PATTERN, 4=HATCH, 5=EMPTY)	E	HOLLOW, SOLID, PATTERN, HATCH, EMPTY	HOLLOW, SOLID, PATTERN, HATCH, EMPTY	HOLLOW, SOLID, PATTERN, HATCH, EMPTY	HOLLOW, SOLID, PATTERN, HATCH, EMPTY	HOLLOW, SOLID, HATCH, EMPTY	**GPQIF** [*interiors*]
Maximum number of interior bundle table entries	I	128*	128*	128*	128*	128*	**GPQLW** [*itable*]
Number of predefined interior bundle table entries	I	6	6	6	6	6	**GPQIF** [*npred*]

Note:

[1]See General Workstation Facilities for a list of DWA and XSOFT Adapters.

* See the text prior to this table for more information.

Table 31. Interior Facilities Default Values

Interior Facilities	Data Type	6090	5080	GDDM	GDF	CGM	Inquiry
Availability of hatch representation (1=NOT_AVAILABLE, 2=BOTH_AVAILABLE, 3=INQUIRE_ONLY_AVAILABLE, 4=SET_ONLY_AVAILABLE)	E	BOTH_ AVAILABLE	BOTH_ AVAILABLE	NOT_ AVAILABLE	NOT_ AVAILABLE	NOT_ AVAILABLE	**GPQHF** [*available*]
Number of available hatch styles	I	24	24	14	14	6	**GPQIF** [*hatnum*]
For available hatch	E	1-24	1-24*	1-14	1-14	1, 2, 4, 6*	**GPQIF** [*hatch*]
Hatch definition format (1=BIT_ARRAY)	E	BIT_ARRAY	BIT_ARRAY	N/A	N/A	N/A	**GPQHF** [*format*]
Maximum length hatch definition data	I	136	32	N/A	N/A	N/A	**GPQHF** [*maxlen*]
Pattern definition format (1-byte integer array)	E	1-byte	1-byte	1-byte	1-byte	1-byte	
Maximum pattern size (*x* size, *y* size)	2 [default] I	32 [default] 32	16 [default] 16	9 [default] 12	12 [default] 20	960 [default] 960*	**GPQPAF** [*maxrow, maxcol*]
Maximum number of pattern indexes	I	4	10	10	10	10	**GPQLW** [*pttable*]
Number of predefined pattern indexes	I	2	2	2	1	2	**GPQPAF** [*indexes*]
Number of available interior styles	I	5	4	5	5	5	**GPQIF** [*intnum*]
Available interior styles (1=HOLLOW, 2=SOLID, 3=PATTERN, 4=HATCH, 5=EMPTY)	E	HOLLOW, SOLID, PATTERN, HATCH, EMPTY	HOLLOW, SOLID, PATTERN, HATCH	HOLLOW, SOLID, PATTERN, HATCH, EMPTY	HOLLOW, SOLID, PATTERN, HATCH, EMPTY	HOLLOW, SOLID, PATTERN, HATCH, EMPTY	**GPQIF** [*interiors*]
Maximum number of interior bundle table entries	I	128*	20	20	20	20	**GPQLW** [*itable*]
Number of predefined interior bundle table entries	I	6	6	6	6	6	**GPQIF** [*npred*]
Note: See the text prior to this table for more information.							

Table 32. Predefined Interior Bundle Tables

Table Entry	Interior Style	Style Index	Color Type	Color Index
1	HOLLOW	1	INDEXED	1
2	SOLID	1	INDEXED	4
3	SOLID	1	INDEXED	5
4	SOLID	1	INDEXED	6
5	PATTERN	1	INDEXED	1
6	PATTERN	1	INDEXED	1

Table 33. Default Hatch Table

Table Entry	Interior Style
1	Vertical lines
2	Horizontal lines
3	Diagonal lines (lower left to upper right 45[default], wide spacing)
4	Diagonal lines (lower left to upper right 45[default], medium spacing)
5	Diagonal lines (lower right to upper left 135[default], wide spacing)
6	Diagonal lines (lower right to upper left 135[default], medium spacing)
7	Raster pattern 1

Table 33. Default Hatch Table (continued)

Table Entry	Interior Style
8	Raster pattern 2
9	Raster pattern 3
10	Raster pattern 4
11	Raster pattern 5
12	Raster pattern 6
13	Raster pattern 7
14	Raster pattern 8
15	Cross-hatched (horizontal and vertical lines), Spacing 1
16	Cross-hatched (diagonal lines), Spacing 1
17	Cross-hatched (horizontal and vertical lines), Spacing 2
18	Cross-hatched (diagonal lines), Spacing 2
19	Cross-hatched (horizontal and vertical lines), Spacing 3
20	Cross-hatched (diagonal lines), Spacing 3
21	Cross-hatched (horizontal and vertical lines), Spacing 4
22	Cross-hatched (diagonal lines), Spacing 4
23	Brick pattern, horizontal
24	Brick pattern, diagonal

Table 34. Predefined Pattern Table

Pattern number	Pattern
Pattern 1	1 1 0 0 1 1 0 0 0 0 1 1 0 0 1 1
Pattern 2	1 2 3 0 0 1 2 3 3 0 1 2 2 3 0 1

Edge Facilities

X

General Information Applying to All Adapters

Nominal, minimum, and maximum edge widths supported depend on the hardware configuration of your workstation. Use the Inquire Edge Facilities (**GPQEF**) subroutine to obtain the values supported on your workstation.

You can set the number of edge bundle table entries that may be active at any one time up to 128 via the Edge Bundle Table (EBTES) procopt. See EBTES (Edge Bundle Table).

Direct Window Access (DWA) Capabilities on the RS/6000 ONLY

In addition to the general capabilities supported by all adapters:

- All DWA Adapters except the POWER GT4x(8 bit or 24 bit):
  - Only nominal edge line width is supported.

- POWER GTO (8 bit or 24 bit):
  - Edge line style is not applied in the following cases:
    - when rendering the Triangle Strip 3 primitive
    - when rendering trimmed surface primitives
    - when rendering Quadrilateral Mesh 3
- POWER Gt4x (8 bit or 24 bit):
  - If edge line style other than `SOLID_LINE` is specified, then the edge line width defaults to a value of 1.0.

XLIB (non-DWA) Capabilities on the RS/6000

In addition to the general capabilities supported by all adapters:

- Color Graphics Display Adapter
  - Only nominal edge line width is supported.

XSOFT

Only nominal edge line width is supported.

6090

- Surface primitives do not support edge attributes.
- Only nominal edge line width is supported.
- Nominal, minimum, and maximum edge widths supported depend on the hardware configuration of your workstation. Use the Inquire Edge Facilities (**GPQEF**) subroutine to obtain the values supported on your workstation.
- You can set the number of edge bundle table entries that may be active at any one time up to 128 via the Edge Bundle Table (`EBTES`) procopt. See EBTES (Edge Bundle Table).

5080

Only nominal edge line width is supported.

GDDM

Nominal, minimum, and maximum edge widths supported depend on the hardware configuration of your workstation. Use the Inquire Edge Facilities (**GPQEF**) subroutine to obtain the values supported on your workstation.

IMAGE

Only nominal edge line width is supported.

CGM

The nominal edge width is 0.000258 meters regardless of how the CGM file is scaled (i.e., the nominal edge width for the graPHIGS API is independent of the metric scale factor of the resultant CGM file). The effective maximum edge width is the size of the workstation display area (0.2582728 meters). However, the size of the display area can be modified using the Escape (**GPES**) subroutine.

For CGM attribute information on edge types, see CGM Edge Attributes.

Table 35. Edge Facilities - X Workstation Default Values

Edge Facilities	Data Type	IMAGE	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Number of available edge line types	I	16	16	16	16	16	**GPQEF** [*netype*]

Table 35. Edge Facilities - X Workstation Default Values (continued)

Edge Facilities	Data Type	IMAGE	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Available edge line types (see Table 15)	E	1-16	1-16	1-16	1-16	1-16	**GPQEF** [*eltype*]
Number of available edge line widths	I	1	Cont. range supported[2]*	1	1	1	**GPQEF** [*nelwidth*]
Nominal edge width (in meters)	R	0.00033866*	0.000332031*	0.000332031*	0.000332031*	0.000332031*	**GPQEF** [*elwidth*]
Minimum edge width (in meters)	R	0.00033866*	0.000332031*	0.000332031*	0.000332031*	0.000332031*	**GPQEF** [*minelw*]
Maximum edge width (in meters)	R	0.00033866*	0.425*	0.000332031*	0.000332031*	0.000332031*	**GPQEF** [*maxelw*]
Maximum number of edge bundle tables entries	I	128*	128*	128*	128*	128*	**GPQLW** [*etable*]
Number of predefined edge bundle tables entries	I	6	6	6	6	6	**GPQPER** [*index*]

Note:

[1]See the text prior to General Workstation Facilities for a list of DWA and XSOFT Adapters.

[2] Continuous range is supported but you will get a 0 back on inquiry.

* See the text prior to this table for more information.

Table 36. Edge Facilities Default Values

Edge Facilities	Data Type	6090	5080	GDDM	GDF	CGM	Inquiry
Number of available edge line types	I	16	13	7	7	5	**GPQEF** [*netype*]
Available edge line types (see Table)	E	1-16	1-13	1-7	1-7	1-4, 7	**GPQEF** [*eltype*]
Number of available edge line widths	I	1	1	2	Cont. range supported[1]	Cont. range supported[1]	**GPQEF** [*nelwidth*]
Nominal edge width (in meters)	R	0.000332031*	0.0002778*	0.0003428*	0.000269	0.00258*	**GPQEF** [*elwidth*]
Minimum edge width (in meters)	R	0.000332031*	0.0002778*	0.0003428*	0.000269	0.00258*	**GPQEF** [*minelw*]
Maximum edge width (in meters)	R	0.000332031*	0.0002778*	0.0003428*	0.2582728	0.2582728	**GPQEF** [*maxelw*]
Maximum number of edge bundle tables entries	I	128*	20	20	20	20	**GPQLW** [*etable*]
Number of predefined edge bundle tables entries	I	6	6	6	6	6	**GPQPER** [*index*]

Note:

[1] Continuous range is supported but you will get a 0 back on inquiry.

* See the text prior to this table for more information.

Table 37. Predefined Edge Bundle

Table Entry	Edge Flag	Edge Line Type	Edge Scale Factor	Color Type	Color Index
1	ON	SOLID_LINE	1.0	INDEXED	1
2	ON	SOLID_LINE	1.0	INDEXED	2
3	ON	SOLID_LINE	1.0	INDEXED	3
4	ON	SOLID_LINE	1.0	INDEXED	4
5	ON	SOLID_LINE	1.0	INDEXED	5
6	ON	SOLID_LINE	1.0	INDEXED	6

Color Facilities

X

General Information Applying to All Adapters

- The number of available colors or intensities supported depend on the hardware configuration of your workstation. Use the Inquire Color Facilities (**GPQCF**) subroutine to obtain the color facilities supported on your workstation.
- The default color table identifier is `-1=DISPLAY_COLOR_TABLE`, with the following exceptions:
  - When the Direct Color (`DIRCOLOR`) procopt or the Do Not Create an X Color Map (`XNOCLRMP`) procopt is specified with the X Window Identifier (`XWINDID`) procopt, the default color table identifier is `0=RENDERING_COLOR_TABLE`. See the PROCOPTS Section in Chapter 7 for details on the DIRCOLOR, XNOCLRMP, and XWINDID procopts.

Direct Window Access (DWA) Capabilities on the RS/6000 ONLY

In addition to the general capabilities supported by all adapters:

POWER GXT550P, POWER GXT500, POWER GXT255P, or POWER GXT250P

- The list of colors for each frame buffer component is dependent on the visual class (8 bit versus 24 bit).

XLIB (non-DWA) Capabilities on the RS/6000 ONLY

In addition to the general capabilities supported by all adapters:

- Color availability depends on the hardware configuration of your workstation.
- The maximum number of display color table entries you can define is 256; however, 16 are supported for a 4 bit display device.
- Color Graphics Display Adapter (8-bit)
  - The list of colors for each frame buffer component is 3,3,2; however, the list is 1,2,1 for 4 bit double-buffer mode.

XSOFT

The list of colors for each frame buffer component is dependent on the visual class.

6090

The number of definable color processing table entries supported depends on the hardware configuration of your workstation. Use the Inquire Color Facilities (**GPQCF**) subroutine to obtain the color facilities supported on your workstation.

The list of colors for each frame buffer component is 8,8,8 for 24 bit single-buffer mode and 3,3,2 for 8 bit double-buffer mode.

5080

The number of available colors or intensities, color availability, and the maximum number of predefined display color table entries supported depend on the hardware configuration of your workstation. Use the Inquire Color Facilities (**GPQCF**) subroutine to obtain the color facilities supported on your workstation.

The use of any of the 5086 windowing features or selective pick through SETUP panel 01-01 will reserve color table entries 120-127 for hardware use. Applications using these color table entries with this feature will produce unexpected display results.

GDDM

The number of available colors or intensities, and color availability supported depend on the hardware configuration of your workstation. Use the Inquire Color Facilities (**GPQCF**) subroutine to obtain the color facilities supported on your workstation.

IMAGE

The maximum number of available colors or intensities that can be represented in the output file is 255^3. displayed is dependent on the output device and the image output format. You can set the image output format via the Image Output Format (IMAGEFMT) procopt.

For more information on color and the IMAGE workstation, see The IMAGE Workstation.

GDF

The maximum number of available colors or intensities that can be represented in the output metafile is 16. The actual number displayed is dependent on the output device and the metafile interpreter.

CGM

The maximum number of available colors or intensities that can be represented in the output metafile is 10^9. The actual number displayed is dependent on the output device and the metafile interpreter.

The line color, text color (for annotation text only), fill color, and edge color each has a value between 0 and the maximum color index.

For CGM metafile and picture descriptor element information on color, see Delimiter Elements and Picture Descriptor Elements.

Table 38. Color Facilities - X Workstation Default Values

Color Facilities	Data Type	IMAGE	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Number of available colors or intensities	I	16,777,216*	16,777,216*	16,777,216*	16,777,216*	16,777,216*	**GPQCF** [*ncolor*]
Color availability (1=MONOCHROME, 2=COLOR)	E	COLOR	COLOR	COLOR	COLOR	COLOR*	**GPQCF** [*avcolor*]
Number of available rendering color models	I	2	2	2	2	1	**GPQRCM** [*totnum*]
Rendering color models (1=RGB_NORMAL, 2=RGB_B_ONLY)	E	RGB_ NORMAL, RGB_ B_ ONLY	RGB_ NORMAL, RGB_ B_ ONLY	RGB_ NORMAL, RGB_ B_ ONLY	RGB_ NORMAL, RGB_ B_ ONLY	RGB_ NORMAL	**GPQRCM** [*model*]
Number of available color quantization methods	I	1	1	1	1	1	**GPQCQM** [*totum*]
Color quantization methods (1=WORKSTATION_ DEPENDENT, 2=BITWISE)	E	BITWISE	BITWISE	BITWISE	BITWISE	BITWISE	**GPQCQM** [*method*]
Number of definable color processing table entries[2]	I	15	15	15	15	15	**GPQCPF** [*entry*]
Default color table identifier (-1=DISPLAY_COLOR_TABLE, 0=RENDERING_COLOR_TABLE)	E	RENDER-ING_ COLOR_ TABLE*	DISPLAY_ COLOR_ TABLE*	DISPLAY_ COLOR_ TABLE*	DISPLAY_ COLOR_ TABLE*	DISPLAY_ COLOR_ TABLE*	**GPQCID** [*ctid*]
Color table characteristics (1=NEITHER_MODIFIABLE, 2=ONLY_DISPLAY_MODIFIABLE, 3=ONLY_RENDERING_MODIFIABLE, 4=BOTH_MODIFIABLE)	E	ONLY_ RENDERING_ MODIFIABLE	BOTH_ MODIFIABLE	BOTH_ MODIFIABLE	BOTH_ MODIFIABLE	BOTH_ MODIFIABLE	**GPQXCF** [*charact*]
Maximum number of display color table entries	I	**8 bit:** 256 **24 bit:** N/A	256	**8 bit:** 256* **24 bit:** 256	**8 bit:** 256 **12 bit:** 64 **24 bit:** 256	256*	**GPQCCH** [*length*]
Number of predefined display color table entries	I	8	8	8	8	8	**GPQCF** [*npred*]
Maximum number of rendering color table entries	I	256	256	256	256	256	**GPQCCH** [*length*]
Number of predefined rendering color table entries	I	0	0	0	0	0	**GPQCF** [*npred*]

Table 38. Color Facilities - X Workstation Default Values (continued)

Color Facilities	Data Type	IMAGE	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Color processing mode (1=RGB, 2=BGR)	E	RGB	RGB	RGB	RGB	RGB	**GPQLCF** [*data*]
List of colors for each frame buffer component	*n*[default]1	8,8,8	**8 bit:** 3,3,2* **24 bit:** 8,8,8	Where supported **8 bit:** 3,3,2* **24 bit:** 8,8,8*	Where supported **8 bit:** 3,3,2* **12 bit:** 4,4,4* **24 bit:** 8,8,8*	3,2,2	

Note:

[1] See the text prior to General Workstation Facilities for a list of DWA and XSOFT Adapters.

[2] Entry 0 cannot be modified.

* See the text prior to this table for more information.

Table 39. Color Facilities Default Values

Color Facilities	Data Type	6090	5080	GDDM	GDF	CGM	Inquiry
Number of available colors or intensities	I	16,777,216	32,768*	16*	16*	10^9 .*	**GPQCF** [*ncolor*]
Color availability (1=MONOCHROME, 2=COLOR)	E	COLOR	COLOR*	COLOR*	COLOR	COLOR	**GPQCF** [*avcolor*]
Number of available rendering color models	I	2	1	1	1	1	**GPQRCM** [*totnum*]
Rendering color models (1= RGB_ NORMAL, 2= RGB_ B_ ONLY)	E	RGB_ NORMAL, RGB_ B_ ONLY	RGB_ NORMAL	RGB_ NORMAL	RGB_ NORMAL	RGB_ NORMAL	**GPQRCM** [*model*]
Number of available color quantization methods	I	1	1	1	1	1	**GPQCQM** [*totum*]
Color quantization methods (1=WORKSTATION_ DEPENDENT, 2=BITWISE)	E	BITWISE	BITWISE	WORKSTATION DEPENDENT	WORKSTATION DEPENDENT	BITWISE	**GPQCQM** [*method*]
Number of definable color processing table entries[1]	I	15*	0	0	0	15	**GPQCPF** [*entry*]
Default color table identifier (-1= DISPLAY_ COLOR_ TABLE, 0= RENDERING_ COLOR_ TABLE)	E	DISPLAY_ COLOR_ TABLE	DISPLAY_ COLOR_ TABLE	RENDERING_ COLOR_ TABLE	RENDERING_ COLOR_ TABLE	RENDERING_ COLOR_ TABLE	**GPQCID** [*ctid*]
Color table characteristics (1=NEITHER_ MODIFIABLE, 2=ONLY_ DISPLAY_ MODIFIABLE, 3=ONLY_ RENDERING_ MODIFIABLE, 4=BOTH_ MODIFIABLE)	E	BOTH_ MODIFIABLE	BOTH_ MODIFIABLE	ONLY_ RENDERING_ MODIFIABLE	ONLY_ RENDERING_ MODIFIABLE	BOTH_ MODIFIABLE	**GPQXCF** [*charact*]
Maximum number of display color table entries	I	256	128*	0	0	256	**GPQCCH** [*length*]
Number of predefined display color table entries	I	8	8	0	0	8	**GPQCF** [*npred*]
Maximum number of rendering color table entries	I	256	256	256	256	256	**GPQCCH** [*length*]
Number of predefined rendering color table entries	I	0	0	8	8	0	**GPQCF** [*npred*]
Color processing mode (1=RGB, 2=BGR)	E	RGB	RGB	RGB	RGB	RGB	**GPQLCF** [*data*]
List of colors for each frame buffer component	*n*[default]1	8,8,8*	3,2,2	N/A	N/A	N/A	

Note:

[1] Entry 0 cannot be modified.

* See the text prior to this table for more information.

Table 40. Default Color Tables

Color Tables Entry	Red	Green	Blue	Color
0	0.0	0.0	0.0	Black
1	1.0	1.0	1.0	White
2	1.0	0.0	0.0	Red
3	0.0	1.0	0.0	Green
4	0.0	0.0	1.0	Blue
5	1.0	1.0	0.0	Yellow
6	1.0	0.0	1.0	Magenta
7	0.0	1.0	1.0	Cyan

Generalized Drawing Primitive (GDP) Facilities

X

General Information Applying to All Adapters

Polygon with Data Primitives (GDPs 1016 and 1017): The primitive supports optional data that indicates that the application determined the convexity of the polygon. Specifying this optional data with the primitive definition enables better performance because the system rendering code does not have to determine the convexity of the polygon each time the polygon is rendered. To determine the convexity of a set of polygons, the graPHIGS API on the RS/6000 contains a sample program under the operating system directory:

`/usr/lpp/graPHIGS/samples/convexcheck`

Direct Window Access (DWA) Capabilities on the RS/6000 ONLY

In addition to the general capabilities supported by all adapters:

- The list of Generalized Drawing Primitives (GDPs) including the advanced Drawing Primitives are supported.
- All DWA Adapters except the POWER GTO:
  - Composite Fill Areas (GDP 1027) are not supported.
- POWER GTO:
  - Composite Fill Areas with edge flag set to `1=OFF`, will display edges.
- POWER Gt4x (8 bit or 24 bit):
  - Polyline Set 3 With Data (GDP 1014) is not supported.

6090

If you are using the shading feature (HLHSR mode is set to `2=ON_THE_FLY`, or depth cue mode is set to `2=ALLOWED`, or lighting calculation mode is set to `2=PER_AREA` or `3=PER_VERTEX`) with Polygon 2/3 with Data, the boundary of the polygon that is defined by the vertex data is assumed to be non-intersecting. If the boundary intersects itself, the visual effect is indeterminate. See *The graPHIGS Programming Interface: Understanding Concepts* for more details.

Composite Fill Areas with edge flag set to `1=OFF`, will display edges.

5080

The number of available generalized drawing primitives supported depends on the hardware configuration of your workstation. Use the Inquire List of Generalized Drawing Primitives (**GPQGD**) subroutine to obtain the generalized drawing primitives supported on your workstation.

Table 41. Generalized Drawing Primitives (GDP) Facilities - X Workstation Defaults

Generalized Drawing Primitive Facilities	Data Type	IMAGE	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Number of available generalized drawing primitives	I	26	27*	26	26	18	**GPQGD** [*totnum*]
Available generalized drawing primitives (see Table 36)	E	1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1014, 1016,* 1017, 1020, 1021, 1022, 1023, 1029, 1031, 1033, 1034, 1035, 1036, 1037, 1039, 1046	1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1014,* 1016,* 1017,* 1020, 1021, 1022, 1023, 1027,* 1029, 1031, 1033, 1034, 1035, 1036, 1037, 1039, 1046	1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1014, 1016,* 1017,* 1020, 1021, 1022, 1023, 1029, 1031, 1033, 1034, 1035, 1036, 1037, 1039, 1046	1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1014, 1016,* 1017,* 1020, 1021, 1022, 1023, 1029, 1031, 1033, 1034, 1035, 1036, 1037, 1039, 1046	1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1022, 1023, 1033, 1034, 1035, 1036, 1039, 1046	**GPQGD** [*gdpid*]

Note:

[1] See the text prior to Table 8, General Workstation Facilities, for a list of DWA and XSOFT Adapters.

* See the text prior to this table for more information.

Table 42. Generalized Drawing Primitives (GDP) Facilities Default Values

Generalized Drawing Primitive Facilities	Data Type	6090	5080	GDDM	GDF	CGM	Inquiry
Number of available generalized drawing primitives	I	25	10*	10	15	15	**GPQGD** [*totnum*]
Available generalized drawing primitives see table	E	1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1016,* 1017,* 1020, 1021, 1022, 1023, 1027, 1029, 1033, 1034, 1035, 1036, 1037, 1039, 1046	1001, 1002, 1003, 1004, 1005,* 1006,* 1007,* 1008,* 1009,* 1010*	1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010	1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1033, 1034, 1035, 1036, 1046	1001, 1002 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1033, 1034, 1035, 1036, 1046	**GPQGD** [*gdpid*]

Note: See the text prior to this table for more information.

Table 43. Available Generalized Drawing Primitives

Decimal Value	Description
1001	Pixel 3
1002	Pixel 2
1003	Disjoint polyline 3
1004	Disjoint polyline 2
1005	Circle 2
1006	Circular arc 2
1007	Ellipse 2
1008	Ellipse 3
1009	Elliptical arc 2
1010	Elliptical arc 3
1014	Polyline set 3 with data

Table 43. Available Generalized Drawing Primitives (continued)

Decimal Value	Description
1016	Polygon 3 with data
1017	Polygon 2 with data
1020	Marker grid 3
1021	Marker grid 2
1022	Line grid 3
1023	Line grid 2
1027	Composite fill area 2
1029	Triangle strip 3
1031	Quadrilateral mesh 3
1033	Non-uniform B-spline curve 3
1034	Non-uniform B-spline curve 2
1035	Non-uniform B-spline surface
1036	Trimmed non-uniform B-spline surface
1037	Polyhedron edge
1039	Character line 2
1046	Polysphere

Generalized Structure Element (GSE) Facilities

X

Direct Window Access (DWA) Capabilities on the RS/6000 ONLY

All DWA Adapters except the POWER GTO:

- The z-buffer protect mask is supported.

POWER Gt4x (8 bit or 24 bit):

- Frame Buffer Comparison: The *mask* parameter specified on this structure element is not supported.
- The z-buffer protect mask is supported.

POWER GTO:

- The z-buffer mask is *not* supported.

XLIB (non-DWA) Capabilities on the RS/6000 ONLY:

- The z-buffer protect mask is *not* supported.
- Only the set frame buffer protect mask is supported.

XSOFT

The z-buffer protect mask is supported.

IMAGE

The z-buffer protect mask is supported.

Table 44. Generalized Structure Element (GSE) Facilities - X Default Values

Generalized Structure Element Facilities	Data Type	IMAGE	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Number of available generalized structure elements	I	11*	9*	11*	11*	2	**GPQGSE** [*totnum*]
Available generalized structure elements	E	1001, 1002,* 1003, 1004, 1005, 1006, 1007, 1008, 1009*, 1011, 1012	1001, 1002,* 1003, 1004, 1005, 1006, 1007, 1008, 1009*	1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1011, 1012	1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1011, 1012	1001, 1008	**GPQGSE** [*gseid*]

Note:

[1]See the text prior to General Workstation Facilities for a list of DWA and XSOFT Adapters.

* See the text prior to this table for more information.

Table 45. Generalized Structure Element (GSE) Facilities Default Values

Generalized Structure Element Facilities	Data Type	6090	5080	GDDM	GDF	CGM	Inquiry
Number of available generalized structure elements	I	8	0	0	1	2	**GPQGSE** [*totnum*]
Available generalized structure elements	E	1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008	N/A	N/A	1008, 1013	1008, 1010, 1013	**GPQGSE** [*gseid*]

Note: See the text prior to this table for more information.

Table 46. GSE Values

Decimal Value	Description
1001	Set frame buffer protect mask
1002	Set frame buffer comparison
1003	Set condition
1004	Conditional execute structure
1005	Conditional return
1006	Text extent 3
1007	Text extent 2
1008	Parametric surface characteristics
1009	Z-buffer protect mask
1010	Workstation-dependent output
1011	Line-on-line color direct
1012	Line-on-line color index
1013	Text line width

Escape Facilities

5080

- When a user switches the 5080 between S/390 host interactive mode and PC mode, the workstation state (display storage, color look-up table, memory management control, etc.) is not preserved.

Your application should ensure that no functions are performed while the user is in non-host or PC mode. By closing and re-opening the workstation when a user switches, you avoid unpredictable results and I/O errors.

Your application can monitor the link switch status by using Escape identifier 1002, Enable/Disable (**GPES**). When a user switches to PC mode, keep the workstation open but allow no update or I/O operations. When the user switches back to host mode, close and re-open the workstation immediately.

GDF/CGM

- Escape identifier 1003 (GDF/CGM Plot Size) allows your application to directly specify the width of the plotted output. The plotting utility provided with the graPHIGS API uses this information to correctly plot the GDF/CGM contents.

CGM

- Escape identifier 1014 (Workstation-Dependent Output) allows your application to directly render data to the workstation. It is the application's responsibility to ensure that the data is valid (proper length(s), identifiers, padding, etc.). For more information, see Workstation Dependent Output or the Escape (**GPES**) subroutine.

Table 47. Escape Facilities - X Workstation Default Values

Escape Facilities	Data Type	IMAGE	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Number of available escapes	I	1	10	10	10	10	**GPQES** [*number*]
Available escapes	E	1008	1001, 1004, 1005, 1007, 1008, 1009, 1010, 1011, 1012, 1015	1001, 1004, 1005, 1007, 1008, 1009, 1010, 1011, 1012, 1015	1001, 1004, 1005, 1007, 1008, 1009, 1010, 1011, 1012, 1015	1001, 1004, 1005, 1007, 1008, 1009, 1010, 1011, 1012, 1015	**GPQES** [*idlist*]
Note: [1] See the text prior to General Workstation Facilities for a list of DWA and XSOFT Adapters.							

Table 48. Escape Facilities Default Values

Escape Facilities	Data Type	6090	5080	GDDM	GDF	CGM	Inquiry
Number of available escapes	I	6	2	1	1	2	**GPQES** [*number*]
Available escapes	E	1001, 1004, 1005, 1006, 1007, 1008	1001, 1002	1001	1003	1003, 1014	**GPQES** [*idlist*]

Table 49. Escape Values

Decimal Value	Description
1001	Sound alarm
1002	Enable/disable link switch
1003	GDF/CGM plot size
1004	Initialize pick correlation state
1005	Set pick selection criteria
1006	Set input echo color
1007	Read frame buffer
1008	Geometric text culling
1009	Window resize notification control
1010	Inquire mapped display surface
1011	Window exposure notification control

Table 49. Escape Values (continued)

Decimal Value	Description
1012	Window deletion notification control
1014	Workstation-Dependent output
1015	Convert coordinate values

Image Facilities

For image facilities that may be supported by the graPHIGS API, see the Image Board Facilities in .

Table 50. Image Facilities - X Workstation Default Values

Image Facilities	Data Type	IMAGE	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Number of available image connections	I	3	3	3	3	3	**GPQIDF** [*totnum*]
Available image connections (-1=FRAME_ BUFFER_ COMPATIBLE, 2=COMPONENT, 3=INDEXED)	E	FRAME_ BUFFER_ COMPATIBLE, COMPONENT, INDEXED	FRAME_ BUFFER_ COMPATIBLE, COMPONENT, INDEXED	FRAME_ BUFFER_ COMPATIBLE, COMPONENT, INDEXED	FRAME_ BUFFER_ COMPATIBLE, COMPONENT, INDEXED	FRAME_ BUFFER_ COMPATIBLE, COMPONENT, INDEXED	**GPQIDF** [*conn*]
Number of available image mapping methods	I	1	1	1	1	1	**GPQIMF** [*totnum*]
Available image mapping methods (1=PIXEL_ BY_ PIXEL)	E	PIXEL_BY_ PIXEL	PIXEL_BY_ PIXEL	PIXEL_BY_ PIXEL	PIXEL_BY_ PIXEL	PIXEL_BY_ PIXEL	**GPQIMF** [*method*]
Number of image mapping priorities supported	I	Cont. range supported[2]	Cont. range supported[2]	Cont. range supported[2]	Cont. range supported[2]	Cont. range supported[2]	**GPQIMF** [*nprio*]
Maximum number of definable images	I	64	64	64	64	64	**GPQIDF** [*nimage*]

Note:

[1]See the text prior to General Workstation Facilities for a list of DWA and XSOFT Adapters.

[2] Continuous range is supported but you will get a 0 back on inquiry.

Table 51. Image Facilities

Image Facilities Default Values	Data Type	6090	5080	GDDM	GDF	CGM	Inquiry
Number of available image connections	I	1	0	0	0	0	**GPQIDF** [*totnum*]
Available image connections (1=FRAME_BUFFER_COMPATIBLE, 2=COMPONENT, 3=INDEXED)	E	FRAME_ BUFFER_ COMPATIBLE	N/A	N/A	N/A	N/A	GPQIDF [*conn*]
Number of available image mapping methods	I	1	0	0	0	0	**GPQIMF** [*totnum*]
Available image mapping methods (1=PIXEL_BY_PIXEL)	E	PIXEL_ BY_ PIXEL	N/A	N/A	N/A	N/A	**GPQIMF** [*method*]
Number of image mapping priorities supported	I	Cont. range supported[1]	N/A	N/A	N/A	N/A	**GPQIMF** [*nprio*]
Maximum number of definable images	I	64	0	0	0	0	**GPQIDF** [*nimage*]

Note: [1] Continuous range is supported but you will get a 0 back on inquiry.

Advanced Output Facilities

X

Direct Window Access (DWA) Capabilities on the RS/6000 ONLY

- The number of light source table entries can be increased via the Light Source Table (LSTES) procopt. See LSTES (Light Source Table).
- The number of definable depth cue table entries can be increased via the Depth Cue Table (DCTES) procopt. See DCTES (Depth Cue Table).
- POWER GTO (8 bit or 24 bit) or POWER Gt4x (8 bit or 24 bit):
  - An ambient light source plus eight other light sources are available. Up to four of the eight light sources can be spotlight sources.
  - The POWER GTO is limited to 64 control points in the *u* direction for a NURB or trimmed NURB surface. A NURB surface primitive with more than 64 control points in the *u* direction is ignored at traversal.
  - The POWER GTO supports spot light concentration exponent values that are a power of 2. Any value specified will be mapped to the closest power of 2 supported.
  - The POWER GTO performs lighting calculations in View Coordinates rather than World Coordinates. These calculations are adjusted to account for the difference in the coordinates. However, if the view transformation is not isotropic (i.e. is not a uniform mapping in X, Y, and Z), then the adjustment is not sufficient and the lighting results may differ from other workstations that perform lighting in World Coordinates.

XSOFT

- The number of light source table entries can be increased via the Light Source Table (LSTES) procopt. See LSTES (Light Source Table).
- The number of definable depth cue table entries can be increased via the Depth Cue Table (DCTES) procopt. See DCTES (Depth Cue Table).

6090

- The number of light source table entries can be increased via the Light Source Table (LSTES) procopt. See LSTES (Light Source Table).
- The number of definable depth cue table entries can be increased via the Depth Cue Table (DCTES) procopt. See DCTES (Depth Cue Table).
- An ambient light source plus eight other light sources are available. Up to four of the eight light sources can be spotlight sources.
- The specular reflection exponent is mapped to a power of 2.
- The Set Face Lighting Method (**GPFLM**) subroutine is not supported. Lighting processing is always face lighting method 2=FACE_DEPENDENT.

Table 52. Advanced Output Facilities - X Workstation Default Values

Advanced Output Facilities	Data Type	IMAGE	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Number of available light source types	I	4	**8 bit:** 4 **24 bit:** 0	4	4	0	**GPQLSF** [*totnum*]
Available light source types (1=AMBIENT, 2=DIRECTIONAL, 3=POSITIONAL, 4=SPOT)	E	AMBIENT, DIRECTIONAL, POSITIONAL, SPOT	AMBIENT, DIRECTIONAL, POSITIONAL, SPOT	AMBIENT, DIRECTIONAL, POSITIONAL, SPOT	AMBIENT, DIRECTIONAL, POSITIONAL, SPOT	N/A	**GPQLSF** [*ltype*]

Table 52. Advanced Output Facilities - X Workstation Default Values (continued)

Advanced Output Facilities	Data Type	IMAGE	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Maximum number of simultaneously active non-ambient light sources	I	8	8	8	8	0	**GPQLSF** [*maxa*]
Maximum number of light source table entries	I	32	32*	32*	32*	0	**GPQLSF** [*maxe*]
Number of predefined light source table entries	I	0	0	0	0	0	**GPQLSF** [*npred*]
Maximum number of definable depth cue table entries[2]	I	15	15*	15*	15*	0	**GPQDCF** [*entry*]
Number of predefined depth cue table entries	I	1	1	1	1	1	**GPQDCF** [*npred*]
Maximum number of definable cull size table entries	I	16	16	16	16	0	**GPQCSF** [*entry*]
Number of predefined cull size table entries	I	0	0	0	0	0	**GPQCSF** [*npred*]

Note:

[1]See the text prior to General Workstation Facilities for a list of DWA and XSOFT Adapters.

[2] Entry 0 cannot be modified.

* See the text prior to this table for more information.

Table 53. Advanced Output Facilities Default Values

Advanced Output Facilities	Data Type	6090	5080	GDDM	GDF	CGM	Inquiry
Number of available light source types	I	4	0	0	0	0	**GPQLSF** [*totnum*]
Available light source types (1=AMBIENT, 2=DIRECTIONAL, 3=POSITIONAL, 4=SPOT)	E	AMBIENT, DIRECTIONAL, POSITIONAL, SPOT	N/A	N/A	N/A	N/A	**GPQLSF** [*ltype*]
Maximum number of simultaneously active non-ambient light sources	I	8*	0	0	0	0	**GPQLSF** [*maxa*]
Maximum number of light source table entries	I	32*	0	0	0	0	**GPQLSF** [*maxe*]
Number of predefined light source table entries	I	0	0	0	0	0	**GPQLSF** [*npred*]
Maximum number of definable depth cue table entries[1]	I	15*	0	0	0	0	**GPQDCF** [*entry*]
Number of predefined depth cue table entries	I	1	1	1	1	1	**GPQDCF** [*npred*]
Maximum number of definable cull size table entries	I	16	0	0	0	0	**GPQCSF** [*entry*]
Number of predefined cull size table entries	I	0	0	0	0	0	**GPQCSF** [*npred*]

Note:

[1] Entry 0 cannot be modified.

* See the text prior to this table for more information.

Curve and Surface Facilities

Table 54. Curve and Surface Facilities - X Default Values

Curve and Surface Facilities	Data Type	IMAGE	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Number of available curve approximation criteria	I	3	3	3	3	3	**GPQCDF** [*totnum*]
Available curve approximation criteria (1=WORKSTATION_DEPENDENT, 3=CONSTANT_SUBDIVISION_ BETWEEN_KNOTS, 8=VARIABLE_SUBDIVISION_ BETWEEN_KNOTS)	E	WORK- STATION_ DEPENDENT, CONSTANT_ SUB- DIVISION_ BETWEEN_ KNOTS, VARIABLE_ SUB- DIVISION_ BETWEEN_ KNOTS	WORK- STATION_ DEPENDENT, CONSTANT_ SUB- DIVISION_ BETWEEN_ KNOTS, VARIABLE_ SUB- DIVISION_ BETWEEN_ KNOTS	WORK- STATION_ DEPENDENT, CONSTANT_ SUB- DIVISION_ BETWEEN_ KNOTS, VARIABLE_ SUB- DIVISION_ BETWEEN_ KNOTS	WORK- STATION_ DEPENDENT, CONSTANT_ SUB- DIVISION_ BETWEEN_ KNOTS, VARIABLE_ SUB- DIVISION_ BETWEEN_ KNOTS	WORK- STATION_ DEPENDENT, CONSTANT_ SUB- DIVISION_ BETWEEN_ KNOTS, VARIABLE_ SUB- DIVISION_ BETWEEN_ KNOTS	**GPQCDF** [*criteria*]
Maximum order of trimming curve for trimmed B-Spline surface	I	26	26	26	26	26	**GPQCDF** [*maxo*]
Number of available surface approximation criteria	I	3	3	3	3	3	**GPQSDF** [*totnum*]
Available surface approximation criteria (1=WORKSTATION_DEPENDENT, 3=CONSTANT_SUBDIVISION_ BETWEEN_KNOTS, 8=VARIABLE_SUBDIVISION_ BETWEEN_KNOTS)	E	WORK- STATION_ DEPENDENT, CONSTANT_ SUB- DIVISION_ BETWEEN_ KNOTS, VARIABLE_ SUB- DIVISION_ BETWEEN_ KNOTS	WORK- STATION_ DEPENDENT, CONSTANT_ SUB- DIVISION_ BETWEEN_ KNOTS, VARIABLE_ SUB- DIVISION_ BETWEEN_ KNOTS	WORK- STATION_ DEPENDENT, CONSTANT_ SUB- DIVISION_ BETWEEN_ KNOTS, VARIABLE_ SUB- DIVISION_ BETWEEN_ KNOTS	WORK- STATION_ DEPENDENT, CONSTANT_ SUB- DIVISION_ BETWEEN_ KNOTS, VARIABLE_ SUB- DIVISION_ BETWEEN_ KNOTS	WORK- STATION_ DEPENDENT, CONSTANT_ SUB- DIVISION_ BETWEEN_ KNOTS, VARIABLE_ SUB- DIVISION _ BETWEEN_ KNOTS	**GPQSDF** [*criteria*]
Maximum order for non-uniform B-Spline surface	I	26	26	26	26	26	**GPQSDF** [*maxo*]
Number of available trimmed curve criteria	I	3	3	3	3	3	**GPQTDF** [*totnum*]
Available trimmed curve criteria (1=WORKSTATION_DEPENDENT, 3=CONSTANT_SUBDIVISION_ BETWEEN_KNOTS, 8=VARIABLE_SUBDIVISION_ BETWEEN_KNOTS)	E	WORK- STATION_ DEPENDENT, CONSTANT_ SUB- DIVISION_ BETWEEN_ KNOTS, VARIABLE_ SUB- DIVISION_ BETWEEN_ KNOTS	WORK- STATION_ DEPENDENT, CONSTANT_ SUB- DIVISION_ BETWEEN_ KNOTS, VARIABLE_ SUB- DIVISION_ BETWEEN_ KNOTS	WORK- STATION_ DEPENDENT, CONSTANT_ SUB- DIVISION_ BETWEEN_ KNOTS, VARIABLE_ SUB- DIVISION_ BETWEEN_ KNOTS	WORK- STATION_ DEPENDENT, CONSTANT_ SUB- DIVISION_ BETWEEN_ KNOTS, VARIABLE_ SUB- DIVISION_ BETWEEN_ KNOTS	WORK- STATION_ DEPENDENT, CONSTANT_ SUB- DIVISION_ BETWEEN_ KNOTS, VARIABLE_ SUB- DIVISION_ BETWEEN_ KNOTS	**GPQTDF** [*criteria*]
Maximum order for trimmed B-spline surface	I	26	26	26	26	26	**GPQTDF** [*maxo*]
Note: [1]See the text prior to General Workstation Facilities for a list of DWA and XSOFT Adapters.							

Table 55. Curve and Surface Facilities Default Values

Curve and Surface Facilities	Data Type	6090	5080	GDDM	GDF	CGM	Inquiry
Number of available curve approximation criteria	I	3	0	0	3	3	**GPQCDF** [*totnum*]

Table 55. Curve and Surface Facilities Default Values (continued)

Curve and Surface Facilities	Data Type	6090	5080	GDDM	GDF	CGM	Inquiry
Available curve approximation criteria (1=WORKSTATION_DEPENDENT, 3=CONSTANT_SUBDIVISION_BETWEEN_KNOTS, 8=VARIABLE_SUBDIVISION_BETWEEN_KNOTS)	E	WORK-STATION_DEPENDENT, CONSTANT_SUB-DIVISION_BETWEEN_KNOTS, VARIABLE_SUB-DIVISION_BETWEEN_KNOTS	N/A	N/A	WORK-STATION_DEPENDENT, CONSTANT_SUB-DIVISION_BETWEEN_KNOTS, VARIABLE_SUB-DIVISION_BETWEEN_KNOTS	WORK-STATION_DEPENDENT, CONSTANT_SUB-DIVISION_BETWEEN_KNOTS, VARIABLE_SUB-DIVISION_BETWEEN_KNOTS	GPQCDF **GPQCDF** [*criteria*]
Maximum order of trimming curve for trimmed B-Spline surface	I	26	N/A	N/A	26	26	**GPQCDF** [*maxo*]
Number of available surface approximation criteria	I	3	0	0	3	3	**GPQSDF** [*totnum*]
Available surface approximation criteria (1=WORKSTATION_DEPENDENT, 3=CONSTANT_SUBDIVISION_BETWEEN_KNOTS, 8=VARIABLE_SUBDIVISION_BETWEEN_KNOTS)	E	WORK-STATION_DEPENDENT, CONSTANT_SUB-DIVISION_BETWEEN_KNOTS, VARIABLE_SUB-DIVISION_BETWEEN_KNOTS	N/A	N/A	WORK-STATION_DEPENDENT, CONSTANT_SUB-DIVISION_BETWEEN_KNOTS, VARIABLE_SUB-DIVISION_BETWEEN_KNOTS	WORK-STATION_DEPENDENT, CONSTANT_SUB-DIVISION_BETWEEN_KNOTS, VARIABLE_SUB-DIVISION_BETWEEN_KNOTS	**GPQSDF** [*criteria*]
Maximum order for non-uniform B-Spline surface	I	26	N/A	N/A	26	26	**GPQSDF** [*maxo*]
Number of available trimmed curve criteria	I	3	0	0	3	3	**GPQTDF** [*totnum*]
Available trimmed curve criteria (1=WORKSTATION_DEPENDENT, 3=CONSTANT_SUBDIVISION_BETWEEN_KNOTS, 8=VARIABLE_SUBDIVISION_BETWEEN_KNOTS)	E	WORK-STATION_DEPENDENT, CONSTANT_SUB-DIVISION_BETWEEN_KNOTS, VARIABLE_SUB-DIVISION_BETWEEN_KNOTS	N/A	N/A	WORK-STATION_DEPENDENT, CONSTANT_SUB-DIVISION_BETWEEN_KNOTS, VARIABLE_SUB-DIVISION_BETWEEN_KNOTS	WORK-STATION_DEPENDENT, CONSTANT_SUB-DIVISION_BETWEEN_KNOTS, VARIABLE_SUB-DIVISION_BETWEEN_KNOTS	**GPQTDF** [*criteria*]
Maximum order for trimmed B-spline surface	I	26	N/A	N/A	26	26	**GPQTDF** [*maxo*]

Advanced Attribute Facilities

X

Direct Window Access (DWA) Capabilities on the RS/6000 ONLY

- The advanced rendering attributes are supported.
- When drawing a wide line with a line type other than 1=SOLID_LINE, the line end type is always 1=FLAT.
- POWER GTO (8 bit or 24 bit) or POWER Gt4x (8 bit or 24 bit):
  - On the POWER Gt4x, the specified end type is ignored for lines of width <=5 pixels. These lines are drawn ″stacked″ on top of each other, so that the lines resemble a parallelogram with the ends always being vertical.

XLIB (non-DWA) Capabilities on the RS/6000 ONLY

- The advanced rendering attributes are not supported.
- Color Graphics Display Adapter:
  – Only polyhedron edge culling mode 1=NONE is supported.

XSOFT

- The advanced rendering attributes are supported.
- When drawing a wide line with a line type other than 1=SOLID_LINE, the line end type is always 1=FLAT.

6090

- The lighting calculation modes supported depend on the hardware configuration of your workstation. Use the Inquire Advanced Attribute Facilities (**GPQAAF**) subroutine to obtain the lighting calculation modes supported on your workstation.
- When an end type of 2=ROUND or 3=SQUARE is applied to wide lines, only a line type of 1=SOLID_LINE is supported.

Table 56. Advanced Attribute Facilities - X Workstation Default Values

Advanced Attribute Facilities	Data Type	IMAGE	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Supported edge flag enumeration (1=OFF, 2=ON, 3=GEOMETRY_ ONLY)	E	OFF, ON, GEOMETRY_ ONLY	OFF, ON, GEOMETRY_ ONLY	OFF, ON, GEOMETRY_ ONLY	OFF, ON, GEOMETRY_ ONLY	OFF, ON	**GPQAAF** [*enum*]
Supported face distinguish mode enumerations (1=NONE, 2=COLOR_ SURFACE_ PROPERTIES)	E	NONE, COLOR_ SURFACE_ PROPERTIES	NONE, COLOR_ SURFACE_ PROPERTIES	NONE, COLOR_ SURFACE_ PROPERTIES	NONE, COLOR_ SURFACE_ PROPERTIES	NONE	**GPQAAF** [*enum*]
Supported lighting calculation mode enumeration (1=NONE, 2=PER_ AREA, 3=PER_ VERTEX)	E	NONE, PER_AREA, PER_VERTEX	NONE, PER_AREA, PER_VERTEX	NONE, PER_AREA, PER_VERTEX	NONE, PER_AREA, PER_VERTEX	NONE	**GPQAAF** [*enum*]
Supported reflectance modes (1=REFLECTANCE_ NONE, 2=AMB, 3=AMB_ DIFF, 4=AMB_ DIFF_ SPEC)	E	REFLECTANCE_ NONE, AMB, AMB_ DIFF, AMB_ DIFF_ SPEC	N/A	REFLECTANCE_ NONE, AMB, AMB_ DIFF, AMB_ DIFF_ SPEC	REFLECTANCE_ NONE, AMB, AMB_ DIFF, AMB_ DIFF_ SPEC	N/A	**GPQAAF** [*enum*]
Supported interior shading methods (1=SHADING_ NONE, 2=SHADING_ COLOR, 3=SHADING_ DATA)	E	SHADING_ NONE, SHADING_ COLOR, SHADING_ DATA	N/A	SHADING_ NONE, SHADING_ COLOR	SHADING_ NONE, SHADING_ COLOR, SHADING_ DATA	N/A	**GPQAAF** [*enum*]
Supported polygon culling enumeration (1=NONE, 2=BACK, 3=FRONT)	E	NONE, BACK, FRONT	NONE, BACK, FRONT	NONE, BACK, FRONT	NONE, BACK, FRONT	NONE	**GPQAAF** [*enum*]
Supported polyhedron edge culling enumeration (1=NONE, 2=BOTH_ BACK, 3=BOTH_ FRONT, 4=BOTH_ BACK_ OR_ BOTH_ FRONT, 5=BACK_ AND_ FRONT, 6=LEAST_ ONE_ BACK, 7=LEAST_ ONE_ FRONT)	E	NONE, BOTH_ BACK, BOTH_ FRONT, BOTH_ BACK_ OR_ BOTH_ FRONT, BACK_ AND_ FRONT, LEAST_ ONE_ BACK, LEAST_ ONE_ FRONT	NONE, BOTH_ BACK, BOTH_ FRONT, BOTH_ BACK_ OR_ BOTH_ FRONT, BACK_ AND_ FRONT, LEAST_ ONE_ BACK, LEAST_ ONE_ FRONT	NONE, BOTH_ BACK, BOTH_ FRONT, BOTH_ BACK_ OR_ BOTH_ FRONT, BACK_ AND_ FRONT, LEAST_ ONE_ BACK, LEAST_ ONE_ FRONT	NONE, BOTH_ BACK, BOTH_ FRONT, BOTH_ BACK_ OR_ BOTH_ FRONT, BACK_ AND_ FRONT, LEAST_ ONE_ BACK, LEAST_ ONE_ FRONT	NONE	**GPQAAF** [*enum*]

Table 56. Advanced Attribute Facilities - X Workstation Default Values (continued)

Advanced Attribute Facilities	Data Type	IMAGE	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Supported polyline end type enumeration (1=FLAT, 2=ROUND, 3=SQUARE)	E	FLAT, ROUND, SQUARE*	FLAT, ROUND, SQUARE*	FLAT, ROUND, SQUARE*	FLAT, ROUND, SQUARE*	FLAT	**GPQAAF** [*enum*]
Maximum number of modeling clipping half-spaces	I	6	N/A	6	6	N/A	**GPQWDT** [*odata*]
Supported modeling clipping operators (1=REPLACE_ VOLUME, 2=INTERSECT_ VOLUME)	E	REPLACE_ VOLUME, INTERSECT_ VOLUME	N/A	REPLACE_ VOLUME, INTERSECT_ VOLUME	REPLACE_ VOLUME, INTERSECT_ VOLUME	N/A	**GPQWDT** [*odata*]
Supported transparency facilities (1=ALPHA BUFFER AVAILABLE)	E	ALPHA BUFFER AVAILABLE	N/A	ALPHA BUFFER AVAILABLE	ALPHA BUFFER AVAILABLE	N/A	**GPQWDT** [*odata*]
Number of partial transparency levels supported	I	17	N/A	17	17	N/A	**GPQWDT** [*odata*]
Supported source blending functions (1=SRCBF_ ZERO, 2=SRCBF_ ONE, 3=SRCBF_ SRC_ ALPHA, 4=SRCBF_ ONE_ MINUS_ SRC_ ALPHA, 5=SRCBF_ DST_ ALPHA, 6=SRCBF_ ONE_ MINUS_ DST_ ALPHA, 7=SRCBF_ DST_ COLOR, 8=SRCBF_ ONE_ MINUS_ DST_ COLOR, 9=SRCBF_ MIN_ SRC_ ALPHA_ ONE_ MINUS_ DST_ ALPHA)	E	SRCBF_ ZERO, SRCBF_ ONE, SRCBF_ SRC_ ALPHA, SRCBF_ ONE_ MINUS_ SRC_ ALPHA, SRCBF_ DST_ ALPHA, SRCBF_ ONE_ MINUS_ DST_ ALPHA, SRCBF_ DST_ COLOR, SRCBF_ ONE_ MINUS_ DST_ COLOR, SRCBF_ MIN_ SRC_ ALPHA_ ONE_ MINUS_ DST_ ALPHA	N/A	SRCBF_ ZERO, SRCBF_ ONE, SRCBF_ SRC_ ALPHA, SRCBF_ ONE_ MINUS_ SRC_ ALPHA, SRCBF_ DST_ ALPHA, SRCBF_ ONE_ MINUS_ DST_ ALPHA, SRCBF_ DST_ COLOR, SRCBF_ ONE_ MINUS_ DST_ COLOR, SRCBF_ MIN_ SRC_ ALPHA_ ONE_ MINUS_ DST_ ALPHA	SRCBF_ ZERO, SRCBF_ ONE, SRCBF_ SRC_ ALPHA, SRCBF_ ONE_ MINUS_ SRC_ ALPHA, SRCBF_ DST_ ALPHA, SRCBF_ ONE_ MINUS_ DST_ ALPHA, SRCBF_ DST_ COLOR, SRCBF_ ONE_ MINUS_ DST_ COLOR, SRCBF_ MIN_ SRC_ ALPHA_ ONE_ MINUS_ DST_ ALPHA	N/A	**GPQWDT** [*odata*]
Supported destination blending functions (1=DSTBF_ZERO, 2=DSTBF_ ONE, 3=DSTBF_ SRC_ ALPHA, 4=DSTBF_ ONE_ MINUS_ SRC_ ALPHA, 5=DSTBF_ DST_ ALPHA, 6=DSTBF_ ONE_ MINUS_ DST_ ALPHA, 7=DSTBF_ SRC_ COLOR, 8=DSTBF_ ONE_ MINUS_ SRC_ COLOR)	E	DSTBF_ ZERO, DSTBF_ ONE, DSTBF_ SRC_ ALPHA, DSTBF_ ONE_ MINUS_ SRC_ ALPHA, DSTBF_ DST_ ALPHA, DSTBF_ ONE_ MINUS_ DST_ ALPHA, DSTBF_ SRC_ COLOR, DSTBF_ ONE_ MINUS_ SRC_ COLOR	N/A	DSTBF_ ZERO, DSTBF_ ONE, DSTBF_ SRC_ ALPHA, DSTBF_ ONE_ MINUS_ SRC_ ALPHA, DSTBF_ DST_ ALPHA, DSTBF_ ONE_ MINUS_ DST_ ALPHA, DSTBF_ SRC_ COLOR, DSTBF_ ONE_ MINUS_ SRC_ COLOR	DSTBF_ ZERO, DSTBF_ ONE, DSTBF_ SRC_ ALPHA, DSTBF_ ONE_ MINUS_ SRC_ ALPHA, DSTBF_ DST_ ALPHA, DSTBF_ ONE_ MINUS_ DST_ ALPHA, DSTBF_ SRC_ COLOR, DSTBF_ ONE_ MINUS_ SRC_ COLOR	N/A	**GPQWDT** [*odata*]
Number of morphing vectors supported	I	4	N/A	4	4	N/A	**GPQWDT** [*odata*]
Maximum data mapping table index	I	8	N/A	8	8	N/A	**GPQWDT** [*odata*]
Supported data mapping methods (-1=IMAGE_ ARRAY, 1=DM_ METHOD_ COLOR 2=SINGLE_ VALUE_ UNIFORM 4=BI_ VALUE_ UNIFORM)	E	IMAGE_ ARRAY, DM_ METHOD_ COLOR, SINGLE_ VALUE_ UNIFORM, BI_ VALUE_ UNIFORM	N/A	DM_ METHOD_ COLOR	IMAGE_ ARRAY, DM_ METHOD_ COLOR, SINGLE_ VALUE_ UNIFORM, BI_ VALUE_ UNIFORM	N/A	**GPQWDT** [*odata*]

Table 56. Advanced Attribute Facilities - X Workstation Default Values (continued)

Advanced Attribute Facilities	Data Type	IMAGE	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Supported data mapping data color types (1=TYPE_ COLOR, 2=TYPE_ PACKED_ RGB, 3=TYPE_ COLOR_ TRANS 4=TYPE_ PACKED_ RGB_ ALPHA)	E	TYPE_ COLOR, TYPE_ PACKED_ RGB, TYPE_ COLOR_ TRANS, TYPE_ PACKED_ RGB_ ALPHA	N/A	N/A	TYPE_ COLOR, TYPE_ PACKED_ RGB, TYPE_ COLOR_ TRANS, TYPE_ PACKED_ RGB_ ALPHA	N/A	**GPQWDT** [*odata*]
Supported text encoding methods (1=UNICODE)	E	1=UNICODE	1=UNICODE	1=UNICODE	1=UNICODE	1=UNICODE	**GPQWDT** [*odata*]

Note:

[1]See the text prior to General Workstation Facilities for a list of DWA and XSOFT Adapters.

* See the text prior to this table for more information.

Table 57. Advanced Attribute Facilities Default Values

Advanced Attribute Facilities	Data Type	6090	5080	GDDM	GDF	CGM	Inquiry
Supported edge flag enumeration (1=OFF, 2=ON, 3=GEOMETRY_ONLY)	E	OFF, ON, GEOMETRY_ ONLY	OFF, ON	OFF, ON	OFF, ON	OFF, ON	**GPQAAF** [*enum*]
Supported face distinguish mode enumerations (1=NONE, 2=COLOR_SURFACE_ PROPERTIES)	E	NONE, COLOR_ SURFACE_ PROPERTIES	NONE	NONE	NONE	NONE	**GPQAAF** [*enum*]
Supported lighting calculation mode enumeration (1=NONE, 2=PER_AREA, 3=PER_VERTEX)	E	NONE, PER_AREA, PER_VERTEX *	NONE	NONE	NONE	NONE	**GPQAAF** [*enum*]
Supported reflectance modes (1=RELECTANCE_NONE, 2=AMB, 3=AMB_DIFF, 4=AMB_DIFF_SPEC)	E	N/A	N/A	N/A	N/A	N/A	**GPQWDT** [*odata*]
Supported interior shading methods (1=SHADING_NONE, 2=SHADING_COLOR, 3=SHADING_DATA)	E	N/A	N/A	N/A	N/A	N/A	**GPQWDT** [*odata*]
Supported polygon culling enumeration (1=NONE, 2=BACK, 3=FRONT)	E	NONE, BACK, FRONT	NONE	NONE	NONE	NONE	**GPQAAF** [*enum*]
Supported polyhedron edge culling enumeration (1=NONE, 2=BOTH_BACK, 3=BOTH_FRONT, 4=BOTH_BACK_OR_ BOTH_FRONT, 5=BACK_AND_FRONT, 6=LEAST_ONE_BACK, 7=LEAST_ONE_FRONT)	E	NONE, BOTH_BACK, BOTH_ FRONT, BOTH_BACK_ OR_BOTH FRONT, BACK_AND_ FRONT, LEAST_ONE_ BACK, LEAST_ONE_ FRONT	NONE	NONE	NONE	NONE	**GPQAAF** [*enum*]
Supported polyline end type enumeration (1=FLAT, 2=ROUND, 3=SQUARE)	E	FLAT, ROUND, SQUARE*	FLAT, ROUND, SQUARE	FLAT	FLAT	FLAT	**GPQAAF** [*enum*]
Maximum number of modeling clipping half-spaces	I	N/A	N/A	N/A	N/A	N/A	**GPQWDT** [*odata*]
Supported modeling clipping operators (1=REPLACE_VOLUME, 2=INTERSECT_VOLUME)	E	N/A	N/A	N/A	N/A	N/A	**GPQWDT** [*odata*]
Supported transparency facilities (1=ALPHA BUFFER AVAILABLE)	E	N/A	N/A	N/A	N/A	N/A	**GPQWDT** [*odata*]

Table 57. Advanced Attribute Facilities Default Values (continued)

Advanced Attribute Facilities	Data Type	6090	5080	GDDM	GDF	CGM	Inquiry
Number of transparency levels supported	E	N/A	N/A	N/A	N/A	N/A	**GPQWDT** [*odata*]
Supported source blending functions (1=SRCBF_ZERO, 2=SRCBF_ONE, 3=SRCBF_SRC_ALPHA, 4=SRCBF_ONE_MINUS_ DST_ALPHA, 5=SRCBF_DST_ALPHA, 6=SRCBF_ONE_MINUS_ DST_ALPHA, 7=SRCBF_DST_COLOR, 8=SRCBF_ONE_MINUS_ DST_COLOR, 9=SRCBF_MIN_SRC_ALPHA_ ONE_MINUS_DST_ ALPHA)	E	N/A	N/A	N/A	N/A	N/A	**GPQWDT** [*odata*]
Supported destination blending functions (1=DSTBF_ZERO, 2=DSTBF_ONE, 3=DSTBF_SRC_ALPHA, 4=DSTBF_ONE_MINUS_ SRC_ALPHA, 5=DSTBF_DST_ALPHA, 6=DSTBF_ONE_MINUS_DST_ ALPHA, 7=DSTBF_SRC_COLOR, 8=DSTBF_ONE_MINUS_SRC_ COLOR)	E	N/A	N/A	N/A	N/A	N/A	**GPQWDT** [*odata*]
Number of morphing vectors supported	I	N/A	N/A	N/A	N/A	N/A	**GPQWDT** [*odata*]
Maximum data mapping table index	I	N/A	N/A	N/A	N/A	N/A	**GPQWDT** [*odata*]
Supported data mapping methods (-1=IMAGE_ARRAY, 1=DM_METHOD_COLOR 2=SINGLE_VALUE_UNIFORM 4=BI_VALUE_UNIFORM)	E	N/A	N/A	N/A	N/A	N/A	**GPQWDT** [*odata*]
Supported data mapping data color types (1=TYPE_COLOR, 2=TYPE_PACKED_RGB, 3=TYPE_COLOR_TRANS 4=TYPE_PACKED_RGB_ALPHA)	E	N/A	N/A	N/A	N/A	N/A	**GPQWDT** [*odata*]
Supported text encoding methods (1=UNICODE)	E	N/A	N/A	N/A	UNICODE	UNICODE	**GPQWDT** [*odata*]
Note: See the text prior to this table for more information.							

General Input Facilities

This section provides a description of the input device classes and the associated input trigger capabilities and echo characteristics. For each supported device, the triggers are listed in the order the workstation processes them (from the highest number secondary trigger proceeding toward and ending with the primary trigger).

If the cursor controller is not in the echo area of an active device, an asterisk '*' indicates the position of the input device. However, if a user-defined cursor from the cursor shape table is in use, there is no change in the appearance of the cursor when it leaves the echo area.

The default echo color on all workstations is white. The default prompt/echo on all workstations is type=1.

X and XSOFT

General Information Applying to All Adapters

- The number of available triggers, the range of qualifiers of your cursor controller or stylus, and the number of valuator logical, choice logical, button physical and scalar physical input devices supported depend on the hardware configuration of your workstation. If you do not have the lighted program function keys (LPFKs) installed, then trigger type 1 is not available.
- Fixed cursor type -1 (cross hair) extends to the limits of the graPHIGS API window.
- The cursor shape table has two predefined shapes that are defined as follows:
  - Entry 1 contains a cursor shape of a pointing hand.
  - Entry 2 contains a cursor shape of a pointing arrow.
- Input character sets 1-5 and 7 are supported through character set 8 (multi-language).
- You can set the number of locator devices via the Locator Devices (`LOCDEVS`) procopt. See LOCDEVS (Locator Devices). To set the number of string devices, use the String Devices (`STRDEVS`) procopt. See STRDEVS (String Devices).
- The IBM 6094 Dial Model 10 and the IBM 6094 LPFKs Model 20 are supported through the Graphic Input Device Adapter (2810) and its attachment cable (4015). These dials and LPFKs are also supported through the serial port adapter (4060) and the direct attachment (4061).
- Engineering symbols are defined in the graPHIGS API font files for character sets 6 (Japanese Katakana), 8 (Multi-Language), and 9 (Single-byte Korean), although no keyboard engineering symbols are engraved on the RS/6000 keyboard. These engineering symbols are available for input through the Alt + key sequence for English, Japanese, and Korean language environments, and the Alt-Gr + key sequence for European language environments.

  The graPHIGS API determines your language environment and keyboard from the variable `LANG`. The following table shows how the engineering symbols map to the keyboard for a given set of `LANG` variable values and X keysym values. For `LANG` variable values not listed, the `En_US` (U.S. English) mapping is the default.

Table 58. Available Keysyms for Completion of Engineering Symbol Sequence

LANG	¢	Φ	○	Ω	≤	≥	±	µ	·	√	∪	∫
En_US	q	w	e	r	t	y	u	i	o	a	s	d
En_GB	q	w	e	r	t	y	u	i	o	a	s	d
De_DE	f	w	engr	r	t	z	u	m	o	a	s	d
Fr_FR	a	z	engr	r	t	y	u	engr	o	q	s	d
It_IT	q	w	engr	r	t	y	u	i	o	a	s	d
En_JP	q	w	e	r	t	y	u	i	o	a	s	d
Sv_SE	q	w	e	r	t	y	u	i	o	a	s	d
ko_KR	q	w	e	r	t	y	u	i	o	a	s	d
Nl_BE Fr_BE	a	z	e,engr	r	t	y	u	i,engr	o	q	s	d
De_CH Fr_CH	q	w	e,engr	r	t	z	u	i	o	a	s	d
zh_TW zh_CN	q	w	e	r	t	y	u	i	o	a	s	d

Note: When an engineering symbol is engraved on the keyboard (indicated by *engr*), the engineering symbol measure follows the key sequence as indicated on the key top, not necessarily the Alt keysym or Alt+Gr keysym sequence.

The keysyms *degree, mu,* and *plusminus* exist in the X11 keysym definitions and can be considered to correspond to the engineering symbols they best represent. Since you can map these keysyms to any key position you want, the engineering symbols follow these keysym definitions to their mapped position.

To map the keys on a keyboard to a specific language keyboard engraving, use the X utility, `xmodmap`. Language key map files can be found in the directory

```
/usr/lpp/X11/defaults/xmodmap/<=LANG>=
```

The file, **keyboard**, corresponds to the RS/6000 default mapping. The file, **keyboard.alt**, if it exists, corresponds to the 5080 default mapping.

- For character set 6 (Katakana) the engineering symbols defined by keysyms may not be remapped. Also, if the keyboard is mapped to a language in which the engineering symbols correspond only to engraved positions on a keyboard (such as *degree* for French keyboards), then the engineering symbols are not available in a Katakana character set.

6090

- The qualifiers supported by your cursor controller or stylus depends on the hardware configuration of your workstation.
- Fixed cursor type -1 (cross hair) extends to the limits of the screen.
- The cursor shape table has two predefined shapes that are defined as follows:
  - Entry 1 contains a cursor shape of a pointing hand.
  - Entry 2 contains a cursor shape of a pointing arrow.
- Input character sets 1-5 and 7 are supported through character set 8 (multi-language).
- For input Kanji, the primary character set must be set to Katakana through the customization panel.

5080

- The number of available triggers, the range of qualifiers of your cursor controller or stylus, and the number of valuator logical, choice logical, button physical and scalar physical input devices supported depend on the hardware configuration of your workstation. If you do not have the lighted program function keys (LPFKs) installed, then trigger type 1 is not available.
- The number of available input character sets depend on the configuration of your workstation.
- Use the IBM 5080 Japanese Language Feature to assign input to either Kanji (character set identifier 128) or Katakana (character set identifier 6). Use the IBM 5080 Korean Language Feature to assign input to Hangul (character set identifier 129).
- Traditional Chinese (character set identifier 130) is not supported.
- Simplified Chinese (character set identifier 132) is not supported.

GDDM

- The number of stroke logical, choice logical, string logical, and button physical input devices supported depend on the hardware configuration of your workstation.
- `3=EVENT` mode input from GDDM-supported workstations must be handled differently by the graPHIGS API from `EVENT` mode input from asynchronous workstations.

  A synchronous interface is used to obtain input from a workstation. GDDM issues a `WAIT` for input, and no other processing can be done until the I/O has been received from the workstation. If an application has several workstations open simultaneously, some of which are asynchronous and some of which are GDDM-supported, then a call to Await Event (**GPAWEV**) only waits for event input from the asynchronous workstations.

  If only GDDM-supported workstations are open and have input devices in `EVENT` mode, then GDDM is called to process the `AWAIT` I/O processing. In this case, the time-out value in the await event is ignored

as GDDM waits until an I/O operation is completed. If several GDDM-supported workstations are open with input devices in EVENT mode, event input is solicited from the workstations in the order that they were opened by the application.

- The input device echo is not clipped to the echo area on GDDM workstations.

Table 59. General Input Facilities - X Workstation Default Values

General Input Facilities	Data Type	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Number of available trigger types for the break action	I	4*	4*	4*	4*	**GPQBK** [*ntrigs*]
Available trigger types for the break action	*n*[default]I	1, 2, 3, 4*	1, 2, 3, 4*	1, 2, 3, 4*	1, 2, 3, 4*	**GPQBK** [*ltrigs*]
Default break trigger type	I	4	4	4	4	**GPQDBK** [*trigger*]
Default break trigger qualifier	I	65539	65539	65539	65539	**GPQDBK** [*trigger*]
Number of locator logical input devices	I	1*	1*	1*	1*	**GPQLI** [*ndev*]
Number of stroke logical input devices	I	2	2	2	2	**GPQLI** [*ndev*]
Number of valuator logical input devices	I	8*	8*	8*	8*	**GPQLI** [*ndev*]
Number of choice logical input devices	I	4*	4*	4*	4*	**GPQLI** [*ndev*]
Number of pick logical input devices	I	1	1	1	1	**GPQLI** [*ndev*]
Number of string logical input devices	I	1*	1*	1*	1*	**GPQLI** [*ndev*]
Number of button physical input devices	I	4*	4*	4*	4*	
Number of scalar physical input devices	I	8*	8*	8*	8*	
Number of vector physical input devices	I	1	1	1	1	
Logical input device interrupt type (1=ASYNCHRONOUS, 2=SYNCHRONOUS)	E	ASYNCHRONOUS	ASYNCHRONOUS	ASYNCHRONOUS	ASYNCHRONOUS	
Number of available input character sets	I	8	11	11	11	**GPQISF** [*ncsid*]
Available input character sets	E	6, 8, 9, 10, 128, 129, 130, 132	6, 8, 9, 10, 11, 12, 128, 129, 130, 132, 134	6, 8, 9, 10, 11, 12, 128, 129, 130, 132, 134	6, 8, 9, 10, 11, 12, 128, 129, 130, 132, 134	**GPQISF** [*csid*]
Maximum number of cursor shape table entries	I	2	2	2	2	**GPQCUF** [*maxent*]
Number of predefined cursor shape table entries	I	2	2	2	2	**GPQCUF** [*npred*]
Number of available cursor definition formats	I	1	1	1	1	**GPQCUF** [*totnum1*]
Number of available fixed cursor types	I	3	3	3	3	**GPQCUF** [*totnum2*]
Available fixed cursor types (-1=full screen cross-hair cursor, -2=none, -3=two color cursor logical input)	E	-1, -2, -3	-1, -2, -3	-1, -2, -3	-1, -2, -3	**GPQCUF** [*lcursor*]

Note:

[1]See the text prior to General Workstation Facilities for a list of DWA and XSOFT Adapters.

* See the text prior to this table for more information.

Table 60. General Input Facilities Default Values

General Input Facilities	Data Type	6090	5080	GDDM	Inquiry
Number of available trigger types for the break action	I	4	4*	0	**GPQBK** [*ntrigs*]
Available trigger types for the break action	*n*[default]I	1, 2, 3, 4	1, 2, 3, 4*	N/A	**GPQBK** [*ltrigs*]
Default break trigger type	I	4	4	4	**GPQDBK** [*trigger*]
Default break trigger qualifier	I	65539	65539	65539	**GPQDBK** [*trigger*]
Number of locator logical input devices	I	1*	1*	1	**GPQLI** [*ndev*]
Number of stroke logical input devices	I	2	2	2*	**GPQLI** [*ndev*]
Number of valuator logical input devices	I	8	8*	0	**GPQLI** [*ndev*]
Number of choice logical input devices	I	4	4*	2*	**GPQLI** [*ndev*]
Number of pick logical input devices	I	1	1	1	**GPQLI** [*ndev*]
Number of string logical input devices	I	1*	1*	1*	**GPQLI** [*ndev*]
Number of button physical input devices	I	4	4*	3*	
Number of scalar physical input devices	I	8	8*	0	
Number of vector physical input devices	I	1	1	1	
Logical input device interrupt type (ASYNCHRONOUS, SYNCHRONOUS)	E	ASYNCH- RONOUS	ASYNCH- RONOUS	SYNCH- RONOUS	
Number of available input character sets	I	8	9*	1	**GPQISF** [*ncsid*]
Available input character sets	E	1-7, 128	1-7, 128, 129*	1	**GPQISF** [*csid*]
Maximum number of cursor shape table entries	I	2	0	0	**GPQCUF** [*maxent*]
Number of predefined cursor shape table entries	I	2	0	0	**GPQCUF** [*npred*]
Number of available cursor definition formats	I	1	0	0	**GPQCUF** [*totnum1*]
Number of available fixed cursor types	I	1	0	0	**GPQCUF** [*totnum2*]
Available fixed cursor types (-1=full screen cross-hair cursor, -2=none, -3=two color cursor logical input)	E	-1	N/A	N/A	**GPQCUF** [*lcursor*]
Note: See the text prior to this table for more information.					

Table 61. Available Cursor Definition Formats for X

format	parm1	parm2
Fixed size bit array	64	64

Table 62. Available Cursor Definition Formats for the 6090

format	parm1	parm2
Fixed size bit array	64	64

Available Triggers

Table 63. Available Triggers for X

Trigger Type	Description	Qualifiers	Description
-2	Trigger when primary fires	0	The secondary trigger fires when the primary fires
-1	Change in measure	Trigger threshold[1]	Change in the physical device's measure
1	Lighted program function keyboard	1-32*	The 32 LPF keys
2	Cursor controller or stylus	1-8*	1) Cursor controller button #1 is released or stylus tip switch is released 2) Cursor controller button #2 is released 3) Cursor controller button #3 is released 4) Cursor controller button #4 is released 5) Cursor controller button #1 is pressed down or stylus tip switch down 6) Cursor controller button #2 is pressed down 7) Cursor controller button #3 is pressed down 8) Cursor controller button #4 is pressed down
3	PF keys on keyboard	1 to n^2	The PF keys on the keyboard
4	Alphanumeric keyboard	See Choice Devices (choice device #4)	See Choice Devices

Note:

[1] A low qualifier specifies that the threshold must be crossed before the device is fired.

[2] *n* can be >=32 depending on the X server and the keyboard being used with your workstation.

* See General Input Facilities for more information.

Table 64. Available Triggers for the 6090

Trigger Type	Description	Qualifiers	Description
-1	Change in measure	0	Change in the physical device's measure
1	Lighted program function keyboard	1-32	The 32 LPF keys

Table 64. Available Triggers for the 6090 (continued)

Trigger Type	Description	Qualifiers	Description
2	Cursor controller or stylus	1-8*	1) Cursor controller button #1 is released or stylus tip switch is released 2) Cursor controller button #2 is released 3) Cursor controller button #3 is released 4) Cursor controller button #4 is released 5) Cursor controller button #1 is pressed down or stylus tip switch down 6) Cursor controller button #2 is pressed down 7) Cursor controller button #3 is pressed down 8) Cursor controller button #4 is pressed down
3	PF keys on keyboard	1-32	The PF keys on the keyboard
4	Alphanumeric keyboard	See Choice Devices (choice device #4)	See Choice Devices
Note: See the text prior to this table for more information.			

Table 65. Available Triggers for the 5080

Trigger Type	Description	Qualifiers	Description
1	Lighted program function keyboard	1-32*	The 32 LPF keys
2	Cursor controller or stylus	1-8*	1) Cursor controller button #1 is released or stylus tip switch is released 2) Cursor controller button #2 is released 3) Cursor controller button #3 is released 4) Cursor controller button #4 is released 5) Cursor controller button #1 is pressed down or stylus tip switch down 6) Cursor controller button #2 is pressed down 7) Cursor controller button #3 is pressed down 8) Cursor controller button #4 is pressed down
3	PF keys on keyboard	1 to *n*[1]	The PF keys on the keyboard
4	Alphanumeric keyboard	See Choice Devices (choice device #4)	See Choice Devices *n* can be <=41 depending on the

Table 65. Available Triggers for the 5080 (continued)

Trigger Type	Description	Qualifiers	Description
Note: [1] Keyboard being used with your workstation. * See General Input Facilities for more information.			

Table 66. Available Triggers for GDDM

Trigger Type	Description	Qualifiers	Description
1	PF keys	1 - 24	The PF keys
2	Mouse (and tablet)	1 - 3*	The mouse buttons
4[1]	Keyboard	65537, 65539	The Enter key, the Cancel key
Note: [1] The button device 4 is used as a trigger but cannot be used as a separate choice device. * See the text prior to this table for more information.			

Locator Devices

Only one locator device is provided by default. For those workstations that support up to two locator devices, use the Locator Devices (LOCDEVS) procopt to modify the number of locators. See LOCDEVS (Locator Devices).

Locator devices do not have secondary triggers. For most workstations, the primary trigger defaults to the release of the buttons on the puck.

The default echo area supported depends on the maximum display surface of your workstation. The maximum display surface changes with various display hardware. Use the Inquire Default Locator Device Data (**GPQDLC**) subroutine to obtain the default echo area of your workstation.

X and XSOFT

General Information Applying to All Adapters

- Attributes specified in the data record are ignored for locator input echo attributes (**GPINLC**).
- Locator Echo Type 7 (structure drag) ignores color and interior attributes specified in the structure network.
- The default high qualifier and available trigger types depend on the hardware configuration of your workstation. If you do not have lighted program function keys (LPFKs) installed, then trigger type 1 is not available.

6090

- The color attribute is ignored for locator input echo attributes (**GPINLC**).

5080

- All attributes are ignored for locator input echo attributes (**GPINLC**).
- The default high qualifier and available trigger types depend on the hardware configuration of your workstation. If you do not have lighted program function keys (LPFKs) installed, then trigger type 1 is not available.

- Locator Echo Type 7 (structure drag) drags pixel primitives by setting all of the pixels to the echo color.

GDDM

- If a mouse or tablet is configured (they are mutually exclusive), this is the locator device. If no mouse or tablet is configured, the cursor keys are used. The locator device trigger is not programmable. The Enter key and the PF keys act as the triggers.
- The locator echo area may not be changed and defaults to full screen.
- All attributes are ignored for location input echo attributes (**GPINLC**).

The following table provides the default triggers for locator devices, which include releasing any of the four cursor controller buttons or releasing the stylus tip switch.

Table 67. Locator Logical Devices - X Default Values

Locator Logical Devices	**Data Type**	**POWER GT4 Family and POWER GTO**	**DWA Adapters[1]**	**XSOFT Adapters[1]**	**XLIB Adapters**	**Inquiry**
Locator device number	I	1, 2*	1, 2*	1, 2*	1, 2*	**GPQLI** [*dev*]
Maximum number of locator devices	I	2	2	2	2	
Number of prompt/echo types	I	6	6	6	6	**GPQDLC** [*necho*]
Available prompt/echo types (1-5, 7)	E	1-5, 7	1-5, 7	1-5, 7	1-5, 7	**GPQDLC** [*echo*]
Default echo area	6[default]R	0.0-0.425, 0.0-0.340, 0.0-0.425*	0.0-0.425, 0.0-0.340, 0.0-0.425*	0.0-0.425, 0.0-0.340, 0.0-0.425*	0.0-0.240, 0.0-0.170, 0.0-0.240*	**GPQDLC** [*area*]
Available supported input character sets (1=PRIMARY, 2=ALL)	E	PRIMARY	PRIMARY	PRIMARY	PRIMARY	**GPQPCS** [*csid*]
Physical input device type for the measure (1=BUTTON, 2=SCALAR, 3=2D_VECTOR)	E	2D_VECTOR	2D_VECTOR	2D_VECTOR	2D_VECTOR	**GPQSPD** [*category*]
Physical input device number for the measure	I	1 (tablet or mouse)	1 (tablet or mouse)	1 (tablet or mouse)	1 (tablet or mouse)	GPQSPD **GPQSPD** [*pdevice*]
Default view index	I	0	0	0	0	
Default initial locator position	3[default]R	0.0, 0.0, 0.0	0.0, 0.0, 0.0	0.0, 0.0, 0.0	0.0, 0.0, 0.0	**GPQDLC** [*pos*]

Note:

[1]See the text prior to General Workstation Facilities for a list of DWA and XSOFT Adapters.

* See the text prior to this table for more information.

Table 68. Locator Logical Devices Default Values

Locator Logical Devices	**Data Type**	**6090**	**5080**	**GDDM**	**Inquiry**
Locator device number	I	1, 2*	1, 2*	1	**GPQLI** [*dev*]
Maximum number of locator devices	I	2	2	1	
Number of prompt/echo types	I	6	6	5	**GPQDLC** [*necho*]
Available prompt/echo types (1-5, 7)	E	1-5, 7	1-5, 7	1-5*	**GPQDLC** [*echo*]
Default echo area	6[default]R	0.0-0.425, 0.0-0.340, 0.0-0.425*	0.0-0.28448, 0.0-0.28448, 0.0-0.28448*	0.0-0.24682, 0.0-0.17574, 0.0-0.24682*	**GPQDLC** [*area*]

Table 68. Locator Logical Devices Default Values (continued)

Locator Logical Devices	Data Type	6090	5080	GDDM	Inquiry
Available supported input character sets (1=PRIMARY, 2=ALL)	E	PRIMARY	PRIMARY	PRIMARY	**GPQPCS** [*csid*]
Physical input device type for the measure (1=BUTTON, 2=SCALAR, 3=2D_VECTOR)	E	2D_VECTOR	2D_VECTOR	2D_VECTOR	**GPQSPD** [*category*]
Physical input device number for the measure	I	1 (tablet)	1 (tablet)	1 (tablet, keyboard, or mouse)	GPQSPD **GPQSPD** [*pdevice*]
Default view index	I	0	0	0	
Default initial locator position	3[default]R	0.0, 0.0, 0.0	0.0, 0.0, 0.0	0.0, 0.0, 0.0	**GPQDLC** [*pos*]
Note: See the text prior to this table for more information.					

Table 69. Locator Trigger Types

Device Number	Trigger Level	Default Trigger Type	Default Low Qualifier	Default High Qualifier	Available Trigger Types
X:					
Locator 1	0	2	1	4*	-1, 1, 2, 3, 4*
Locator 2	0	2	1	4*	-1, 1, 2, 3, 4*
6090:					
Locator 1	0	2	1	4	-1, 1, 2, 3, 4*
Locator 2	0	2	1	4	-1, 1, 2, 3, 4*
5080:					
Locator 1	0	2	1	4*	1, 2, 3, 4*
Locator 2	0	2	1	4*	1, 2, 3, 4*
GDDM:					
Locator 1	0	2	1	3	None
	0	1	1	24	None
	0	4	65537	65537	None
Note: The values identified with the * reflect the default value, but not necessarily the actual value. It depends on the hardware configuration of the workstation.					

Stroke Devices

There are two types of stroke devices:

- Stroke device #1 is the streaming stroke device and has two secondary triggers.
  - Secondary trigger #2 initiates the accumulation of stroke points into the stroke buffer.
  - Secondary trigger #1 ends the accumulation of points into the stroke buffer.

By default, the primary trigger corresponds to the same event as the secondary trigger #1. Therefore, upon the release of a cursor control button, accumulation of input stops and input is fired to the application.

- Stroke device #2 is a discrete stroke device and has four secondary triggers that act upon the point at the edit position.

  The edit position can be from 0 to *n*+1 where 0 refers to the position before the first stroke point and *n*+1 refers to the position after the last stroke point. The four secondary triggers act as follows:

  – Secondary trigger #4 deletes the point at the edit position and moves the edit position backward one point (that is, toward the first point in the list).

    If the list is not empty and the edit position is neither before the first point nor after the last point, this trigger deletes the point at the edit position. If the edit position is not before the first point, this trigger moves it back one.

- Secondary trigger #3 inserts the point identified by the current cursor location into the list after the edit position and moves the edit position forward one point (that is, toward the last point in the list).

  If there are fewer points in the list than the application will accept, it inserts the new point into the list. Otherwise, this trigger sounds the alarm and does not change either the buffer or the edit position.

  If the edit position is not beyond the end of the list, this trigger moves all points above the current point up one, opening the slot above the current point. It inserts the new point in that slot and advances the edit position to the new point.

  If the edit position is beyond the end of the list, this trigger appends the new point to the list and leaves the edit position unchanged (that is, pointing at the new point).

- Secondary trigger #2 moves the edit position forward one point (that is, toward the last point in the list).

  If the list is not empty and the edit position is not after the last point, this trigger moves the edit position.

- Secondary trigger #1 moves the edit position backward one point (that is, toward the first point in the list).

  If the list is not empty and the edit position is not before the first point, this trigger moves the edit position.

When the echo method uses polylines or polymarkers and the edit position is within the range 1-*n*, the current point is indicated with a diamond around the point.

When the echo method uses polylines and the edit position is 0 or *n*+1, a rubberband line is displayed from the first or last stroke point to the current cursor position.

The time interval parameter in the Initialize Stroke (**GPINSK**) data record is ignored.

The default echo area supported depends on the maximum display surface of your workstation. The maximum display surface changes with various display hardware. Use the Inquire Default Stroke Device Data (**GPQDSK**) subroutine to obtain the default echo area of your workstation.

The default high qualifier and available trigger types depend on the hardware configuration of your workstation. If you do not have lighted program function keys (LPFKs) installed, then trigger type 1 is not available. It also depends on the number of mouse buttons provided.

X and XSOFT

General Information Applying to All Adapters

The color attribute is ignored for stroke input echo attributes (**GPINSK**).

6090, 5080

The color attribute is ignored for stroke input echo attributes (**GPINSK**).

- Stroke device #1 is only available if a mouse or tablet is configured. When a mouse or tablet button is pressed, the stroke device is activated and points are sampled at fixed intervals. Pressing a mouse or puck button or stylus switch a second time deactivates the device and suspends stream sampling. You can only trigger this device when it is not accumulating points.

  The data record, the editing position, and the echo area are not evaluated. Only the first point of the initial stroke is displayed.
- Stroke device #2 is only available if a mouse or tablet is configured. One stroke point is sampled each time a mouse or puck button or stylus switch is pressed.

  The data record, the editing position, and the echo area are not evaluated. Only the first point of the initial stroke is displayed.

Table 70. Stroke Logical Devices - X Default Values (Stroke 1 and 2)

Stroke Logical Devices	**Data Type**	**POWER GT4 Family and POWER GTO**	**DWA Adapters[1]**	**XSOFT Adapters[1]**	**XLIB Adapters**	**Inquiry**
Stroke device number	I	1, 2	1, 2	1, 2	1, 2	**GPQLI** [*dev*]
Number of prompt/echo types	I	3	3	3	3	**GPQDSK** [*necho*]
Available prompt/echo types (1, 3, 4)	E	1, 3, 4	1, 3, 4	1, 3, 4	1, 3, 4	**GPQDSK** [*echo*]
Default echo area	6[default]R	0.0-0.425, 0.0-0.340, 0.0-0.425*	0.0-0.425, 0.0-0.340, 0.0-0.425*	0.0-0.425, 0.0-0.340, 0.0-0.425*	0.0-0.240, 0.0-0.170, 0.0-0.240*	**GPQDSK** [*area*]
Available supported input character sets (PRIMARY, ALL)	E	PRIMARY	PRIMARY	PRIMARY	PRIMARY	**GPQPCS** [*csid*]
Physical input device type for the measure (1=BUTTON, 2=SCALAR, 3=2D_VECTOR)	E	2D_VECTOR	2D_VECTOR	2D_VECTOR	2D_VECTOR	**GPQSPD** [*category*]
Physical input device number for the measure	I	1 (tablet or mouse)	1 (tablet or mouse)	1 (tablet or mouse)	1 (tablet or mouse)	**GPQSPD** [*pdevice*]
Maximum input buffer size (in points)	I	337	337	337	337	**GPQSK** [*buflen*]
Default initial stroke input buffer size (in points)	I	337	337	337	337	**GPQDSK** [*buflen*]
Default view index	I	0	0	0	0	
Initial number of points	I	0	0	0	0	**GPQSK** [*npoint*]
Editing position	I	1	1	1	1	**GPQSK** [*editpos*]

Note: [1]See the text prior to General Workstation Facilities for a list of DWA and XSOFT Adapters.

* See the text prior to this table for more information.

Table 71. Stroke Logical Devices (Stroke 1 and 2) Default Values

Stroke Logical Devices	**Data Type**	**6090**	**5080**	**GDDM**	**Inquiry**
Stroke device number	I	1, 2	1, 2	1, 2	**GPQLI** [*dev*]
Number of prompt/echo types	I	3	3	2, 3	**GPQDSK** [*necho*]
Available prompt/echo types (1, 3, 4)	E	1, 3, 4	1, 3, 4	Stroke 1: 1, 3, 4	**GPQDSK** [*echo*]
Default echo area	6[default]R	0.0-0.425, 0.0-0.340, 0.0-0.425*	0.0-0.28448, 0.0-0.28448, 0.0-0.28448*	0.0-0.24682, 0.0-0.17574, 0.0-0.24682*	**GPQDSK** [*area*]
Available supported input character sets (PRIMARY, ALL)	E	PRIMARY	PRIMARY	PRIMARY	**GPQPCS** [*csid*]

Table 71. Stroke Logical Devices (Stroke 1 and 2) Default Values (continued)

Stroke Logical Devices	Data Type	6090	5080	GDDM	Inquiry
Physical input device type for the measure (1=BUTTON, 2=SCALAR, 3=2D_VECTOR)	E	2D_VECTOR	2D_VECTOR	2D_VECTOR	**GPQSPD** [*category*]
Physical input device number for the measure	I	1 (tablet)	1 (tablet)	1 (tablet or mouse)	**GPQSPD** [*pdevice*]
Maximum input buffer size (in points)	I	337	337	64	**GPQSK** [*buflen*]
Default initial stroke input buffer size (in points)	I	337	337	64	**GPQDSK** [*buflen*]
Default view index	I	0	0	0	
Initial number of points	I	0	0	0	**GPQSK** [*npoint*]
Editing position	I	1	1	1	**GPQSK** [*editpos*]
Note: See the text prior to this table for more information.					

Table 72. Stroke Trigger Types

Device Number	Trigger Level	Default Trigger Type	Default Low Qualifier	Default High Qualifier	Available Trigger Types
Stroke 1	0	2	1	4*	1, 2, 3, 4*
	1[1]	2	1	4*	1, 2, 3, 4*
	2[1]	2	5	8*	1, 2, 3, 4*
Stroke 2	0	4	65537	65537	1, 2, 3, 4*
	1[1]	none	none	none	1, 2, 3, 4*
	2[1]	2	1	4*	1, 2, 3, 4*
	3[1]	2	1	4*	1, 2, 3, 4*
	4[1]	2	1	4*	1, 2, 3, 4*
Note: [1] There are no available trigger types for GDDM. * See the text prior to this table for more information.					

Valuator Devices

Valuator devices do not have secondary triggers.

By default, the trigger type is workstation-dependent. When the device is in REQUEST mode, it is triggered by the Enter key on the keyboard. When it is in EVENT mode, it is triggered by any movement of the dial that results in a change to the measure.

The default echo area supported depends on the maximum display surface of your workstation. The maximum display surface changes with various display hardware. Use the Inquire Default Valuator Device Data (**GPQDDV**) subroutine to obtain the default echo area of your workstation.

The available trigger types depend on the hardware configuration of your workstation. If you do not have lighted program function keys (LPFKs) installed, then trigger type 1 is not available.

X AND XSOFT

General Information Applying to All Adapters

You can set the primary trigger type to one that is fired whenever the measure of the device changes by an amount greater than an application-specified delta. This is known as trigger by change in measure and is most useful when the device is in EVENT mode.

You can also set the primary trigger type to any one of the button device types, using any of the alternatives or any combination of the alternatives on the selected device. This trigger type is most useful when the device is in REQUEST mode.

Use **GPIT** to change the defaults, but once you've changed the trigger type from the default, you cannot change it back to the workstation-dependent type, since this type is not selectable through programming.

The application can use XChangeDeviceControl (a function in the Enhanced X-Windows Input Extension Library) to set the granularity of the valuators before opening the workstation. This value can be used to set the dial granularity to a user-defined value.

6090, 5080

The valuator device number depends on the hardware configuration of your workstation. Use the Inquire List of Logical Input Devices (**GPQLI**) subroutine to obtain the valuator device number defaults supported by your workstation.

Table 73. Valuator Logical Devices (Values 1-8) - X Default Values

Valuator Logical Devices	Data Type	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Valuator device number	E	1-8*	1-8*	1-8*	1-8*	**GPQLI** [*dev*]
Number of prompt/echo types	I	3	3	3	3	**GPQDVL** [*necho*]
Available prompt/echo types (1, 3, 4)	E	1, 3, 4	1, 3, 4	1, 3, 4	1, 3, 4	**GPQDVL** [*echo*]
Default echo area	6[default]R	1) 0.0-0.425, 0.3267-0.340, 0.0-0.425* 2) 0.0-0.425, 0.3134-0.340, 0.0-0.425* 3) 0.0-0.425, 0.3001-0.340, 0.0-0.425* 4) 0.0-0.425, 0.2868-0.340, 0.0-0.425* 5) 0.0-0.425, 0.2735-0.340, 0.0-0.425* 6) 0.0-0.425, 0.2602-0.340, 0.0-0.425* 7) 0.0-0.425, 0.2469-0.340, 0.0-0.425* 8) 0.0-0.425, 0.2336-0.340, 0.0-0.425*	1) 0.0-0.425, 0.3267-0.340, 0.0-0.425* 2) 0.0-0.425, 0.3134-0.340, 0.0-0.425* 3) 0.0-0.425, 0.3001-0.340, 0.0-0.425* 4) 0.0-0.425, 0.2868-0.340, 0.0-0.425* 5) 0.0-0.425, 0.2735-0.340, 0.0-0.425* 6) 0.0-0.425, 0.2602-0.340, 0.0-0.425* 7) 0.0-0.425, 0.2469-0.340, 0.0-0.425* 8) 0.0-0.425, 0.2336-0.340, 0.0-0.425*	1) 0.0-0.425, 0.3267-0.340, 0.0-0.425* 2) 0.0-0.425, 0.3134-0.340, 0.0-0.425* 3) 0.0-0.425, 0.3001-0.340, 0.0-0.425* 4) 0.0-0.425, 0.2868-0.340, 0.0-0.425* 5) 0.0-0.425, 0.2735-0.340, 0.0-0.425* 6) 0.0-0.425, 0.2602-0.340, 0.0-0.425* 7) 0.0-0.425, 0.2469-0.340, 0.0-0.425* 8) 0.0-0.425, 0.2336-0.340, 0.0-0.425*	1) 0.0-0.28448, 0.27448-0.28448, 0.0-0.28448* 2) 0.0-0.28448, 0.26448-0.28448, 0.0-0.28448* 3) 0.0-0.28448, 0.25448-0.28448, 0.0-0.28448* 4) 0.0-0.28448, 0.24448-0.28448, 0.0-0.28448* 5) 0.0-0.28448, 0.23448-0.28448, 0.0-0.28448* 6) 0.0-0.28448, 0.22448-0.28448, 0.0-0.28448* 7) 0.0-0.28448, 0.21448-0.28448, 0.0-0.28448* 8) 0.0-0.28448, 0.20448-0.28448, 0.0-0.28448*	**GPQDVL** [*area*]
Physical input device type for the measure (1=BUTTON, 2=SCALAR, 3=2D_VECTOR)	E	SCALAR	SCALAR	SCALAR	SCALAR	**GPQSPD** [*category*]

Table 73. Valuator Logical Devices (Values 1-8) - X Default Values (continued)

Valuator Logical Devices	Data Type	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Physical input device number for the measure	I	1) 1 (dial 1) 2) 2 (dial 2) 3) 3 (dial 3) 4) 4 (dial 4) 5) 5 (dial 5) 6) 6 (dial 6) 7) 7 (dial 7) 8) 8 (dial 8)	1) 1 (dial 1) 2) 2 (dial 2) 3) 3 (dial 3) 4) 4 (dial 4) 5) 5 (dial 5) 6) 6 (dial 6) 7) 7 (dial 7) 8) 8 (dial 8)	1) 1 (dial 1) 2) 2 (dial 2) 3) 3 (dial 3) 4) 4 (dial 4) 5) 5 (dial 5) 6) 6 (dial 6) 7) 7 (dial 7) 8) 8 (dial 8)	1) 1 (dial 1) 2) 2 (dial 2) 3) 3 (dial 3) 4) 4 (dial 4) 5) 5 (dial 5) 6) 6 (dial 6) 7) 7 (dial 7) 8) 8 (dial 8)	**GPQSPD** [*pdevice*]
Default initial value	R	0.0	0.0	0.0	0.0	**GPQDVL** [*ivalue*]
Default low value	R	0.0	0.0	0.0	0.0	**GPQDVL** [*lovalue*]
Default high value	R	1.0	1.0	1.0	1.0	**GPQDVL** [*hivalue*]

Note:

[1]See the text prior to General Workstation Facilities for a list of DWA and XSOFT Adapters.

* See the text prior to this table for more information.

Table 74. Valuator Logical Devices (Values 1-8) Default Values

Valuator Logical Devices	Data Type	6090	5080	Inquiry
Valuator device number	E	1-8*	1-8*	**GPQLI** [*dev*]
Number of prompt/echo types	I	3	3	**GPQDVL** [*necho*]
Available prompt/echo types (1, 3, 4)	E	1, 3, 4	1, 3, 4	**GPQDVL** [*echo*]

Table 74. Valuator Logical Devices (Values 1-8) Default Values (continued)

Valuator Logical Devices	Data Type	6090	5080	Inquiry
Default echo area	6[default]R	1) 0.0-0.425, 0.3267-0.340, 0.0-0.425* 2) 0.0-0.425, 0.3134-0.340, 0.0-0.425* 3) 0.0-0.425, 0.3001-0.340, 0.0-0.425* 4) 0.0-0.425, 0.2868-0.340, 0.0-0.425* 5) 0.0-0.425, 0.2735-0.340, 0.0-0.425* 6) 0.0-0.425, 0.2602-0.340, 0.0-0.425* 7) 0.0-0.425, 0.2469-0.340, 0.0-0.425* 8) 0.0-0.425, 0.2336-0.340, 0.0-0.425*	1) 0.0-0.28448, 0.27448-0.28448, 0.0-0.28448* 2) 0.0-0.28448, 0.26448-0.28448, 0.0-0.28448* 3) 0.0-0.28448, 0.25448-0.28448, 0.0-0.28448* 4) 0.0-0.28448, 0.24448-0.28448, 0.0-0.28448* 5) 0.0-0.28448, 0.23448-0.28448, 0.0-0.28448* 6) 0.0-0.28448, 0.22448-0.28448, 0.0-0.28448* 7) 0.0-0.28448, 0.21448-0.28448, 0.0-0.28448* 8) 0.0-0.28448, 0.20448-0.28448, 0.0-0.28448*	**GPQDVL** [*area*]
Physical input device type for the measure (1=BUTTON, 2=SCALAR, 3=2D_VECTOR)	E	SCALAR	SCALAR	**GPQSPD** [*category*]
Physical input device number for the measure	I	1) 1 (dial 1) 2) 2 (dial 2) 3) 3 (dial 3) 4) 4 (dial 4) 5) 5 (dial 5) 6) 6 (dial 6) 7) 7 (dial 7) 8) 8 (dial 8)	1) 1 (dial 1) 2) 2 (dial 2) 3) 3 (dial 3) 4) 4 (dial 4) 5) 5 (dial 5) 6) 6 (dial 6) 7) 7 (dial 7) 8) 8 (dial 8)	**GPQSPD** [*pdevice*]
Default initial value	R	0.0	0.0	**GPQDVL** [*ivalue*]
Default low value	R	0.0	0.0	**GPQDVL** [*lovalue*]
Default high value	R	1.0	1.0	**GPQDVL** [*hivalue*]
Note: See the text prior to this table for more information.				

Table 75. Valuator Triggers

Device Number	Trigger Level	Default Trigger Type	Default Low Qualifier	Default High Qualifier	Available Trigger Types
For X					
Valuator 1-8	0	0	0	0	-1, 1, 2, 3, 4*
For the 6090					
Valuator 1-8	0	0	0	0	-1, 1, 2, 3, 4*
For the 5080					

Table 75. Valuator Triggers (continued)

Device Number	Trigger Level	Default Trigger Type	Default Low Qualifier	Default High Qualifier	Available Trigger Types
Valuator 1-8	0	0	0	0	None
Note: See the text prior to this table for more information.					

Choice Devices

All choice devices possess only primary triggers. In general, the default corresponds to all choice alternatives on the given choice device. The alternatives for choice device triggers are limited to the alternatives on the given choice device; therefore, a device cannot be fired by input actions on other input devices. Choice alternatives are deactivated by setting the choice device triggers to the subset of choice alternatives that you would like to activate. Echo is not supported on choice devices.

A choice device in sample mode always returns ″no choice.″

The four choice devices are:

- Choice device #1 is typically the lighted program function keys.
- Choice device #2 corresponds to the buttons on a pointing device. There are multiple pointing devices, such as:
  - Cursor controller, which has eight trigger alternatives.

    The default triggers for the four-button cursor controller are alternatives 1 through 4, which correspond to the release of a cursor button. By default, alternatives 5 through 8 are disabled. These alternatives correspond to the depression of buttons 1 through 4, respectively.
  - Stylus, which has two trigger alternatives.

    The default trigger for the stylus is alternative 1, which corresponds to the release of the tip switch. By default, alternative 5 is disabled. This alternative corresponds to the depression of the tip switch.
  - Mouse buttons; there are two trigger alternatives, button pressed and button released, for each button on the mouse.
- Choice device #3 is the function keys on the keyboard. The number of alternatives is dependent on the number of function keys on your keyboard.
- Choice device #4 corresponds to the data and control keys on the keyboard.

  For the data keys, this choice device returns a choice's value which corresponds to the character of the key pressed. The values returned are dependent on the keyboard that is attached to your workstation and to the input device's primary character set. Some workstations may support several language keyboards which have different keys and corresponding code points. For the available keys, the tables found in Character Sets and Fonts Provided by the API illustrate the corresponding code points for each available language. (Use Font 1 of the primary character set to determine the available choice alternatives.) The returned choice values for the data keys are EBCDIC code points for the GDDM/graPHIGS API Programming Interface and ASCII code points for the Personal graPHIGS API.

  For the control keys (for which no character code point exists), the following lists the supported keys and their corresponding choice values:

Table 76. Supporting keys and choice values

Key	Choice Value	
Enter	X'10001'	= 65537
New line	X'10002'	= 65538
Cancel	X'10003'	= 65539
Up arrow	X'10004'	= 65540

Table 76. Supporting keys and choice values (continued)

Key	Choice Value	
Down arrow	X'10005'	= 65541
Left arrow	X'10006'	= 65542
Right arrow	X'10007'	= 65543
Tab forward	X'10008'	= 65544
Tab back	X'10009'	= 65545
Insert	X'1000A'	= 65546
Delete	X'1000B'	= 65547
Backspace	X'1000C'	= 65548
Up arrow + Shift	X'1000D'	= 65549
Down arrow + Shift	X'1000E'	= 65550
Left arrow + Shift	X'1000F'	= 65551
Right arrow + Shift	X'10010'	= 65552
Up arrow + Alt	X'10011'	= 65553
Down arrow + Alt	X'10012'	= 65554
Left arrow + Alt	X'10013'	= 65555
Right arrow + Alt	X'10014'	= 65556
Home	X'10015'	= 65557
Home + Shift	X'10016'	= 65558
Home + Alt	X'10017'	= 65559
PA1	X'10018'	= 65560
EOF	X'10019'	= 65561
EOF + Shift	X'1001A'	= 65562
EOF + Alt	X'1001B'	= 65563
PA2	X'1001C'	= 65564
CLEAR	X'1001D'	= 65565
+ on numeric pad + Alt	X'1001E'	= 65566
- on numeric pad + Alt	X'1001F'	= 65567

The choice device number depends on the hardware configuration of your workstation. Use the Inquire List of Logical Input Devices (**GPQLI**) subroutine to obtain the choice device number defaults supported by your workstation.

The default echo area supported depends on the maximum display surface of your workstation. The maximum display surface changes with various display hardware. Use the Inquire Default Choice Device Data (**GPQDCH**) subroutine to obtain the default echo area of your workstation.

The default high qualifier depends on the hardware configuration of your workstation.

X and XSOFT

General Information Applying to All Adapters

Choice device #3 has the following values:

1-255	The keyboard program function keys F1 to F*x* (where *x* is the highest PF key on your keyboard).
256-511	The keyboard program function keys F1 to F*x* plus the SHIFT key (where *x* is the highest PF key on your keyboard).

Only 24 choice device #3 triggers are supported with an IBM keyboard. 12 of which are the unshifted PF keys, and the other 12 are the shifted PF keys.

To support additional choice device #3 triggers, the current keyboard may be remapped using the X utility, `xmodmap`.

You must edit the file **/usr/lpp/X11/defaults/xmodmap/$LANG/keyboard** with superuser authority, where `$LANG` is the result you obtain when entering the **echo $LANG** command.

Each entry is listed as a keycode, base identifier, shift modifier, and alternate graphics modifier. You may wish to change the states of the base identifier and shift modifier to return a new pfkey measure. If a base identifier already returns a pfkey measure, you may wish to change the state of the shift modifier to return an additional pfkey measure.

As an example, with a current entry in the table that appears as follows:

```
keycode 120 = F1        NoSymbol     NoSymbol
```

you may change the shift identifier (the second column) to be `F16`. The new edited entry then appears as follows:

```
keycode 120 = F1        F16          NoSymbol
```

You may wish to change the print key measures. Currently, the print key returns `Print` when it is pressed. If you wish to change the print key to return `Function 13` when the print key is pressed and `Function 28` when the shift and the print key are pressed, you would change the entry:

```
keycode 132 = Print     NoSymbol     NoSymbol
```

to

```
keycode 132 = F13       F28           NoSymbol
```

When you have finished editing the keyboard file, type the command **xmodmap keyboard** to have the changes take effect as the current keyboard mapping.

Some keys that you may wish to remap may be first used by the window manager, so, the measure is not given to your application. An example of this is the `SHIFT-ESCAPE` key while running the Motif window manager. Such keys under the Motif window manager may be changed in the key binding description file. This file name appears as **/usr/lib/X11/system.mwmrc** on your system. In the DefaultKeyBindings structure, you may comment any lines which describe a key that you wish the window manager *not* to control. For example, if an entry in the file appears as follows:

```
Shift<Key>Escape      icon|window      f.post_wmenu
```

you may comment this line to obtain the measure for the `SHIFT-ESCAPE` key through the graPHIGS API. Then the Motif window manager allows the `SHIFT-ESCAPE` key to pass through to your application for your use.

The number of mouse buttons supported for choice device #2 may differ from the number of physical mouse buttons because many X servers map the mouse's physical mouse buttons into a larger number of logical mouse buttons. For example, the PS/2 mouse has two buttons, but the X server recognizes the mouse as having three buttons: *left*, *middle* (both left and right depressed), and *right*. Since the graPHIGS API uses the X server's physical-to-logical mapping of the buttons, the number of logical buttons will be consistent between graPHIGS API and X applications.

6090

Choice device #3 has the following values:

1-16 The keyboard program function keys F0 to F15
17-32 The keyboard program function keys F0 to F15 plus the SHIFT key
33-41 The keyboard program function keys F0 to F8 plus the ALT key

5080

Choice device #3 has either 24 or 41 alternatives, depending on the type of keyboard present on the 5080. When a 5085 Model 1 keyboard is used, 24 choice alternatives are present. Otherwise, 41 choice alternatives are present. These choice values correspond to the following:

1-16 The keyboard program function keys F0 to F15
17-32 The keyboard program function keys F0 to F15 plus the SHIFT key
33-41 The keyboard program function keys F0 to F8 plus the ALT key

The following list itemizes differences that exist between the geometric text code pages found in Character Sets and Fonts Provided by the API and the available choice alternatives for the French and Italian language features:

- French (CSID 4) on the 5080 Models 1, 1A
  - X'42', X'52', X'56', X'57', X'63', X'CA', X'CB', X'DB', X'EA', X'FA' are not available as choice alternatives.
- French (CSID 4) on the 5080 Model 2
  - X'CA' is not available as a choice alternative.
- Italian (CSID 5)
  - X'DB' is not available as a choice alternative.

Table 77. Choice Logical Devices (Values 1-4) - X Workstation Default Values

Choice Logical Devices	Data Type	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Choice device number	I	1) LPFKs 2) mouse or tablet 3) PF keys 4) keyboard*	1) LPFKs 2) mouse or tablet 3) PF keys 4) keyboard*	1) LPFKs 2) mouse or tablet 3) PF keys 4) keyboard*	1) LPFKs 2) mouse or tablet 3) PF keys 4) keyboard*	**GPQLI** [*dev*]
Number of prompt/echo types	I	1) 2 2) 1 3) 1 4) 1	1) 2 2) 1 3) 1 4) 1	1) 2 2) 1 3) 1 4) 1	1) 2 2) 1 3) 1 4) 1	**GPQDCH** [*necho*]
Available prompt/echo types (1, 2)	E	1) 1, 2 2) 1 3) 1 4) 1	1) 1, 2 2) 1 3) 1 4) 1	1) 1, 2 2) 1 3) 1 4) 1	1) 1, 2 2) 1 3) 1 4) 1	**GPQDCH** [*echo*]

Table 77. Choice Logical Devices (Values 1-4) - X Workstation Default Values (continued)

Choice Logical Devices	Data Type	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Default echo area	6[default]R	0.0-0.425, 0.0-0.340, 0.0-0.425*	0.0-0.425, 0.0-0.340, 0.0-0.425*	0.0-0.425, 0.0-0.340, 0.0-0.425*	0.0-0.240, 0.0-0.170, 0.0-0.240*	**GPQDCH** [*area*]
Available supported input character sets (PRIMARY, ALL)	E	PRIMARY	PRIMARY	PRIMARY	PRIMARY	**GPQPCS** [*csid*]
Physical input device type for the measure (1=BUTTON, 2=SCALAR, 3=2D_VECTOR)	E	BUTTON	BUTTON	BUTTON	BUTTON	**GPQSPD** [*category*]
Physical input device number for the measure	I	1=LPFKs 2=mouse buttons 3=PF keys 4=keyboard	1=LPFKs 2=mouse buttons 3=PF keys 4=keyboard	1=LPFKs 2=mouse buttons 3=PF keys 4=keyboard	1=LPFKs 2=mouse buttons 3=PF keys 4=keyboard	**GPQSPD** [*pdevice*]
Maximum choice alternatives	I	1) 32 2) 8 3) 1 to n^2 4) 210	1) 32 2) 8 3) 1 to n^2 4) 210	1) 32 2) 8 3) 1 to n^2 4) 210	1) 32 2) 8 3) 1 to n^2 4) 210	**GPQDCH** [*choice*]
Default choice alternative	I	1) 0 2) 0 3) 0 4) 0	1) 0 2) 0 3) 0 4) 0	1) 0 2) 0 3) 0 4) 0	1) 0 2) 0 3) 0 4) 0	

Note:

[1]See the text prior to General Workstation Facilities for a list of DWA and XSOFT Adapters.

[2] *n* can be >=32 depending on the X server and the keyboard being used with your workstation.

* See the text prior to this table for more information.

Table 78. Choice Logical Devices (Values 1-4) Default Values

Choice Logical Devices	Data Type	6090	5080	GDDM	Inquiry
Choice device number	I	1) LPFKs 2) 4-button cursor controller 3) PF keys 4) keyboard*	1) LPFKs 2) 4-button cursor controller 3) PF keys 4) keyboard*	1) PF keys 2) mouse or tablet*	**GPQLI** [*dev*]
Number of prompt/echo types	I	1) 2 2) 1 3) 1 4) 1	1) 2 2) 1 3) 1 4) 1	1) 2 2) 1	**GPQDCH** [*necho*]
Available prompt/echo types (1, 2)	E	1) 1, 2 2) 1 3) 1 4) 1	1) 1, 2 2) 1 3) 1 4) 1	1) 1, 2 2) 1	**GPQDCH** [*echo*]
Default echo area	6[default]R	0.0-0.425, 0.0-0.340, 0.0-0.425*	0.0-0.28448, 0.0-0.28448, 0.0-0.28448*	0.0-0.24682, 0.0-0.17574, 0.0-0.424682*	**GPQDCH** [*area*]

Table 78. Choice Logical Devices (Values 1-4) Default Values (continued)

Choice Logical Devices	Data Type	6090	5080	GDDM	Inquiry
Available supported input character sets (PRIMARY, ALL)	E	PRIMARY	PRIMARY	PRIMARY	**GPQPCS** [*csid*]
Physical input device type for the measure (1=BUTTON, 2=SCALAR, 3=2D_VECTOR)	E	BUTTON	BUTTON	BUTTON	**GPQSPD** [*category*]
Physical input device number for the measure	I	1=LPFKs 2=cursor keys 3=PF keys 4=key-board	1=LPFKs 2=cursor keys 3=PF keys 4=key-board	1=LPFKs 2=mouse buttons *	**GPQSPD** [*pdevice*]
Maximum choice alternatives	I	1) 32 2) 8 3) 41 4) 172	1) 32 2) 8 3) 1 to *n* [1] 4) 138	1) 24 2) 3	**GPQDCH** [*choice*]
Default choice alternative	I	1) 0 2) 0 3) 0 4) 0	1) 0 2) 0 3) 0 4) 0	1) 0 2) 0	

Note:

[1] *n* can be <=41 depending on the keyboard being used with your workstation.

* See the text prior to this table for more information.

Table 79. Choice Triggers

Device Number	Trigger Level	Default Trigger Type	Default Low Qualifier	Default High Qualifier	Available Trigger Types
For X					
Choice 1	0	1	1	32	1
Choice 2	0	2	1	4*	2
Choice 3	0	3	See Table 65	See Table 65	3
Choice 4	0	4	See Table 65	See Table 65	4
For 6090					
Choice 1	0	1	1	32	1
Choice 2	0	2	1	4*	2
Choice 3	0	3	1	41	3

Table 79. Choice Triggers (continued)

Device Number	Trigger Level	Default Trigger Type	Default Low Qualifier	Default High Qualifier	Available Trigger Types
Choice 4	0	4	See Table 65	See Table 65	4
For 5080					
Choice 1	0	1	1	32*	1
Choice 2	0	2	1	4*	2
Choice 3	0	3	1	41*	3
Choice 4	0	4	See Table 65	See Table 65	4
For GDDM					
Choice 1	0	1	1	24*	None
Choice 2	0	2	1	3*	None
Note: See the text prior to this table for more information.					

Pick Devices

The pick device supports two secondary triggers. Secondary trigger #2 starts the pick correlation process. Secondary trigger #1 ends the pick correlation process. For most workstations, these triggers default to the pressing and releasing of buttons on a mouse or tablet device.

The device determines the minimum and maximum pick apertures. If the value you set in Set Pick Aperture (**GPPKAP**) is less or greater than these limits, then it is automatically adjusted to the minimum or maximum value.

A pick device in sample mode always returns a pick path length of zero.

The default pick aperture for all workstations is 2% of the smaller of the display surface *x*, *y* values.

The default echo area supported depends on the maximum display surface of your workstation. The maximum display surface changes with various display hardware. Use the Inquire Default Pick Device Data (**GPQDPK**) subroutine to obtain the default echo area of your workstation.

The available trigger types depend on the hardware configuration of your workstation. If you do not have lighted program function keys (LPFKs) installed, then trigger type 1 is not available. The default high qualifier of the pick trigger also also depends on the number of mouse buttons provided.

X and XSOFT

General Information Applying to All Adapters

- Pick Echo Type 1 (primitive highlighting) is not supported with exclusive-or echo.
- Pick triggers #1 and #2 are only available if a mouse or tablet is configured.
- The default echo area is defined by the maximum display surface. The maximum display surface changes with various display hardware.
- When trigger by change in measure is active, the logical device is triggered by a change in the physical device measure that results in a change in the logical device measure. The device can be triggered by buttons regardless of the correlation state. A change in physical device measure only causes device correlation if correlation is set to `ON`. Correlation can be set to `ON` by a button secondary trigger or through the **GPIPKC** subroutine.

  When trigger by change in measure is active, a `NO-PICK` event is reported (pick with depth of 0) when the pick device leaves a pickable object and enters a non-pickable object or background.

The graPHIGS API accumulates the distance from one event to the next and correlates only when the low qualifier threshold is exceeded, except for the last event occurring in a chain of events. The last in a chain of events is always correlated, regardless of the low qualifier. Events accumulated prior to the last event are also correlated when they exceed the low qualifier threshold.

- Trigger when primary fires (trigger type -2) is valid only for secondary triggers. Low and high qualifiers are not used and should be set to zero.
- POWER Gt4x (8 bit or 24 bit):
  - Pick selection criteria, 4=FIRST_VISIBLE, 5=LAST_VISIBLE, 6=ALL_VISIBLE, are not supported.

6090

When the pick criteria is ALL, the application gets up to 20 items that pass through the pick aperture.

5080

Pick aperture size cannot be defined as less than two pixels.

GDDM

If a mouse or tablet is configured (they are mutually exclusive), this is the pick device. If no mouse or tablet is configured, the cursor keys are used. The echo area is not evaluated. The Enter key and the PF keys are triggers.

Picking on GDDM workstations is performed by using the . position of the cross hair. The picked primitive is determined by examining the central structure storage based on this location. Your application may use a deferral mode in which the screen image does not always match the central storage structure. In this case the pick information may not match what the user is visually selecting. To avoid this situation, the application should always update the workstation before the user interaction occurs.

Table 80. Pick Logical Devices - X Workstation Default Values

Pick Logical Devices	Data Type	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Pick device number	I	1	1	1	1	**GPQLI** [*dev*]
Number of prompt/echo types	I	1	1	1	1	**GPDPK** [*necho*]
Available prompt/echo types (1)	E	1	1	1	1	**GPQDPK** [*echo*]
Default echo area	6[default]R	0.0-0.425, 0.0-0.340, 0.0-0.425*	0.0-0.425, 0.0-0.340, 0.0-0.425*	0.0-0.425, 0.0-0.340, 0.0-0.425*	0.0-0.240, 0.0-0.170, 0.0-0.240*	**GPQDPK** [*area*]
Available supported input character sets (PRIMARY, ALL)	E	PRIMARY	PRIMARY	PRIMARY	PRIMARY	**GPQPCS** [*csid*]
Physical input device type for the measure (1=BUTTON, 2=SCALAR, 3=2D_VECTOR)	E	2D_VECTOR	2D_VECTOR	2D_VECTOR	2D_VECTOR	**GPQSPD** [*category*]
Physical input device number for the measure	I	1 (tablet or mouse)	1 (tablet or mouse)	1 (tablet or mouse)	1 (tablet or mouse)	**GPQSPD** [*pdevice*]
Pick measure type (1=NORMAL, 2=EXTENDED)	E	EXTENDED	EXTENDED	EXTENDED	EXTENDED	**GPQPKT** [*type*]
Maximum pick path depth	I	32	32	32	16	
Number of available pick selection criteria	I	6*	6*	6*	3	
Available pick selection criteria (1=FIRST, 2=LAST, 3=ALL, 4=FIRST_VISIBLE, 5=LAST_VISIBLE, 6=ALL_VISIBLE)	E	FIRST, LAST, ALL*	FIRST, LAST, ALL, FIRST_ VISIBLE, LAST_ VISIBLE, ALL_ VISIBLE*	FIRST, LAST, ALL, FIRST_ VISIBLE, LAST_ VISIBLE, ALL_ VISIBLE*	FIRST, LAST, ALL	**GPQIDD** [*odata*]
Initial pick path depth	I	0	0	0	0	

Table 80. Pick Logical Devices - X Workstation Default Values (continued)

Pick Logical Devices	Data Type	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Default pick correlation state (1=OFF, 2=ON)	E	OFF	OFF	OFF	OFF	
Default pick path order (1=TOP_FIRST, 2=BOTTOM_FIRST)	E	TOP_FIRST	TOP_FIRST	TOP_FIRST	TOP_FIRST	

Note:

[1]See the text prior to General Workstation Facilities for a list of DWA and XSOFT Adapters.

* See the text prior to this table for more information.

Table 81. Pick Logical Devices Default Values

Pick Logical Devices	Data Type	6090	5080	GDDM	Inquiry
Pick device number	I	1	1	1	**GPQLI** [*dev*]
Number of prompt/echo types	I	1	1	1	**GPQDPK** [*necho*]
Available prompt/echo types (1)	E	1	1	1	**GPQDPK.htm** [*echo*]
Default echo area	6[default]R	0.0-0.425, 0.0-0.340, 0.0-0.425*	0.0-0.28448, 0.0-0.28448, 0.0-0.28448*	0.0-0.24682, 0.0-0.17574, 0.0-0.24682*	**GPQDPK** [*area*]
Available supported input character sets (PRIMARY, ALL)	E	PRIMARY	PRIMARY	PRIMARY	**GPQPCS** [*csid*]
Physical input device type for the measure (1=BUTTON, 2=SCALAR, 3=2D_VECTOR)	E	2D_VECTOR	2D_VECTOR	2D_VECTOR	**GPQSPD** [*category*]
Physical input device number for the measure	I	1 (tablet)	1 (tablet)	1 (tablet or mouse)	**GPQSPD** [*pdevice*]
Pick measure type (1=NORMAL, 2=EXTENDED)	E	EXTENDED	NORMAL	NORMAL	**GPQPKT** [*type*]
Maximum pick path depth	I	32	16	16	
Number of available pick selection criteria	I	6	1	1	
Available pick selection criteria (1=FIRST, 2=LAST, 3=ALL, 4=FIRST_VISIBLE, 5=LAST_VISIBLE, 6=ALL_VISIBLE)	E	FIRST, LAST, ALL, FIRST_ VISIBLE, LAST_ VISIBLE, ALL_ VISIBLE	LAST	LAST	**GPQIDD** [*odata*]
Initial pick path depth	I	0	0	0	
Default pick correlation state (1=OFF, 2=ON)	E	OFF	OFF	OFF	
Default pick path order (1=TOP_FIRST, 2=BOTTOM_FIRST)	E	TOP_FIRST	TOP_FIRST	TOP_FIRST	

Note: See the text prior to this table for more information.

Table 82. Pick Triggers for X Workstation

Device Number	Trigger Level	Default Trigger Type	Default Low Qualifier	Default High Qualifier	Available Trigger Types
Pick 1	0	2	1	4*	-1, 1, 2, 3, 4*
Pick 1	1	2	1	4*	-2, 1, 2, 3, 4*
Pick 1	2	2	5	8*	-2, 1, 2, 3, 4*

Note: See the text prior to this table for more information.

Table 83. Pick Triggers for 6090 Workstation

Device Number	Trigger Level	Default Trigger Type	Default Low Qualifier	Default High Qualifier	Available Trigger Types
Pick 1	0	2	1	4*	1, 2, 3, 4*
Pick 1	1	2	1	4*	1, 2, 3, 4*
Pick 1	2	2	5	8*	1, 2, 3, 4*
Note: See the text prior to this table for more information.					

Table 84. Pick Triggers for 5080 Workstation

Device Number	Trigger Level	Default Trigger Type	Default Low Qualifier	Default High Qualifier	Available Trigger Types
Pick 1	0	2	1	4*	1, 2, 3, 4*
Pick 1	1	2	1	4*	1, 2, 3, 4*
Pick 1	2	2	5	8*	1, 2, 3, 4*
Note: See the text prior to this table for more information.					

Table 85. Pick Triggers for GDDM Workstation

Device Number	Trigger Level	Default Trigger Type	Default Low Qualifier	Default High Qualifier	Available Trigger Types
Pick 1	0	2	1	3	None
Pick 1	0	1	1	24	None
Pick 1	0	4	65537	65537	None

String Devices

Each workstation supports up to 16 string devices. By default, the graPHIGS API only provides one. To modify the number of string devices, use the String Devices (`STRDEVS`) procopt. See STRDEVS (String Devices).

The string is echoed in the lower left corner of each string device's echo area.

The default echo area is defined by the maximum display surface size. The maximum display surface size changes with various display hardware. Use the Inquire Default String Device Data (**GPQDST**) subroutine to obtain the default echo area. Use the Inquire Maximum Display Surface Size (**GPQDS**) subroutine to obtain the maximum display surface of your workstation.

You cannot trigger a national language string device by any trigger if the string device is in pre-editing mode or if it has auxiliary translation windows present.

Only one prompt is visible on the screen at any given time. However, each string device maintains its own current cursor position. When the application activates the device, the graPHIGS API positions the cursor as specified by the Initialize String (**GPINST**) subroutine call or, if your application does not specify **GPINST**, then the graPHIGS API positions the cursor at position 1. As you type, the graPHIGS API updates the current cursor position appropriately. If you advance the cursor beyond the end of the current string device's input buffer, then it advances to the first position of the next string device's input buffer. If only one string device is active, then the cursor wraps around in the input buffer. If you are in insert mode and you are typing in characters, then characters do *not* shift from one string device into the next and the cursor does *not* advance to the next string device's input buffer.

The string device operates as follows:

Key	Action
Tab Forward key Ind	Moves the cursor from its current position to the current cursor position in the next active string device.
Tab Back key	Moves the cursor from its current position back one active string device to that device's current cursor position.
Insert key	Toggles between insert mode and replace mode.
Delete key	Deletes the current character and shifts the succeeding characters left.
Left Arrow key	Moves the cursor to the previous character or to the last position in the previous active string device.
Right Arrow key	Moves the cursor to the next character or to the first position in the next active string device.
Double Left Arrow key	Moves the cursor as if the Left Arrow key were pressed twice.
Double Right Arrow key	Moves the cursor as if the Right Arrow key were pressed twice.
Backspace key	Moves the cursor to the previous character, deletes it, and then shifts the succeeding characters left. If the cursor is in the first position, this key has no effect.
Erase EOF key	Clears the current string from the cursor to the end of the string.
Erase Field key	Moves the cursor to the first position in the current string and then clears from there to the end of the string.
Erase Input key	Clears all string device input areas and places the cursor at the first position in the active string device with the lowest device number.
Clear key	Resets all string devices to their initial states and places the cursor at the initial position in the active string device with the lowest device number.
Newline key	Moves the cursor from its current position to the first position of the next string device's input buffer.
Home key	Moves the cursor to the current position in the active string device with the lowest device number.
Reset key	Resets string device to replace mode and also resets any dead key sequence.

X and XSOFT

General Information Applying to All Adapters

This workstation supports all the above string editing keys except Double Left Arrow, Double Right Arrow, Erase Input, Erase Field, Clear, Home, and Reset.

Each string device has only a primary trigger. This trigger is programmable. By default, a string device's trigger is the Enter key.

Unicode: (character set identifier 131) is not supported.

Kanji: String editing within a Kanji (character set identifiers 128 and 134) string device involves a conversion between the characters that are actually typed on the keyboard, and the final Kanji characters. Characters that are first typed in the string device, may appear in the reverse video area or they may be underlined, or they may appear in the reverse video area and be underlined. This mode is the pre-editing mode. During pre-editing, no graPHIGS API input events are triggered by the physical button device 4 (the keyboard). Pressing the Enter key ends pre-editing mode. Then, when you press the string device trigger (typically when you press the Enter key again), the graPHIGS API triggers the string device.

Indicator:

The graPHIGS API displays the indicators of a graPHIGS API string device at the rightmost position of the string device echo area. Two indicators are displayed in the indicator boxes:

- DBCS (double-byte character set) shift indicator
- Alphanum/Hiragana/Katakana shift indicator

The Romaji/Kana Conversion (RKC) shift indicator is displayed as a small filled rectangle at the inside upper right of the Alphanum/Hiragana/Katakana shift indicator box. If the state is `RKC on` and *MAX* is maximum byte length of the Kanji string device, you can input *MAX-2* byte length text in it. To input *MAX* byte length, you have to change the RKC state to `off`.

DBCS/SBCS shift state:

The graPHIGS API for the AIXwindows Environment/6000 string device supports the double-byte character set (DBCS) state only. The DBCS/SBCS shift toggle key has no meaning. The maximum input buffer size and default initial string input buffer size is 120 bytes.

Hangul:

String editing within a Hangul (character set identifier 129) string device involves a conversion between the characters actually typed on the keyboard, and the final Hangul characters. Characters that are typed in the string device while in Hangul mode may be underlined. While composing Hangul characters, the graPHIGS API string device may not be triggered until the composed character is complete.

IBM Personal graPHIGS API differs from Hangul in the following ways:

- Indicator

  The graPHIGS API displays the indicators of a graPHIGS API string device at the rightmost position of the string device echo area. Two indicators are displayed in the indicator boxes:

  – DBCS (double-byte character set) shift indicator

    The graPHIGS API for the AIXwindows Environment/6000 string device supports the double-byte character set (DBCS) state only. The DBCS/SBCS shift toggle key has no meaning.

  – English/Hangul/Jamo shift indicator

-

- Simplified Chinese: String editing within a Simplified Chinese (character set identifier 132) string device involves a conversion between the characters actually typed on the keyboard, and the final Simplified Chinese characters. Characters that are first typed in the string device may appear in reverse video. This mode is the pre-editing mode. During pre-editing, no graPHIGS API input events are triggered by the physical button device 4 (the keyboard). Pressing the convert, non-convert, reset (Escape) key ends pre-editing mode. Then, when you press the string device trigger (typically when you press the Enter key), the graPHIGS API triggers the string device.

  IBM Personal graPHIGS API differs from Simplified Chinese in the following ways:

  – Indicator

    The graPHIGS API displays the indicators of a graPHIGS API string device at the rightmost position of the string device echo area. Up to four indicators are displayed in the indicator boxes:

    - Chinese/English shift indicator
    - Half/Full-width shift indicator

      The graPHIGS API supports the double-byte character set (DBCS) state only. The Half/Full-width shift toggle key has no meaning.

    - Row-Column/ Pinyin / English / Intelligent ABC conversion shift indicator
    - Association shift indicator (used only with Pinyin)

  – No error messages are displayed

- Traditional Chinese: String editing within a Traditional Chinese (character set identifier 130) string device involves a conversion between the characters that are actually typed on the keyboard, and the final Traditional Chinese characters. Characters that are first typed in the string device, may appear in the reverse video or be underlined, or may appear both in the reverse video and be underlined. This mode is the pre-editing mode. During pre-editing, no graPHIGS API input events are triggered by the physical

button device 4 (the keyboard). Pressing the convert, non-covert or reset (Escape) key ends pre-editing mode. Then, when you press the string device trigger (typically when you press the Enter key), the graPHIGS API triggers the string device.

The graPHIGS API string device returns the Traditional Chinese codepoints using the IBM-927 character encoding (double-bytes only).

IBM Personal graPHIGS API differs from Traditional Chinese in the following ways:

– Indicator

  The graPHIGS API displays the indicators of a graPHIGS API string device at the rightmost position of the string device echo area. Three indicators are displayed in the indicator boxes:

  - Chinese/English shift indicator
  - Half/Full-width shift indicator

    The graPHIGS API supports the double-byte character set (DBCS) state only. The Half/Full-width shift toggle key has no meaning.
  - Tsang-Jye/Phonetic/Internal-Code shift indicator

– No error messages are displayed

  The graPHIGS API does not display error messages for messages such as `No Word` or `Wrong Key`.

6090

- In general, it is not possible to change the input character set for string input; the character set is determined by the 6090 Language Customization Feature. The only exception is when using the IBM 6090 Japanese Language Feature; then the input character set can be assigned to either Kanji (character set identifier 128), or Katakana (character set identifier 6), which is the default.
- Hangul (character set identifier 129), Traditional Chinese (character set identifier 130), Simplified Chinese (character set identifier 132), Unicode (character set identifier 131), and Kanji supporting the IBM-943 encoding (character set identifier 134) are not supported.
- This workstation supports all the above string editing keys except Double Left Arrow, Double Right Arrow, Erase Input, Erase Field, Clear, Home, and Reset.
- Each string device has only a primary trigger. This trigger is programmable. By default, a string device's trigger is the Action key.

5080

- In general, it is not possible to change the input character set for string input; the character set is determined by the 5085 Language Customization Feature. The only exception is when using either the IBM 5080 Japanese Language Feature or the IBM 5080 Korean Language Feature. Only one is supported at a time, not both. With the Japanese Language Feature, you can assign the input character set to either Kanji (character set identifier 128) or Katakana (character set identifier 6), which is the default. With the Korean Language Feature, you can assign the input character set to either Hangul (character set identifier 129) or Single-byte Korean (character set identifier 9), which is the default.
- Traditional Chinese (character set identifier 130) is not supported.
- Simplified Chinese (character set identifier 132) is not supported.
- Unicode (character set identifier 131) is not supported.
- Kanji (character set identifier 128): If you want to provide the additional characters for the echoing of string device input, you need to update the IBM 5080 Japanese Language Feature.
- Kanji support for the IBM-943 encoding (character set 134), is not supported.
- Each string device has only a primary trigger. This trigger is programmable. By default, a string device's trigger is the Action key.

GDDM

- The display area is divided into cells. The cells are rectangular in shape, arranged in rows and columns, and each can display one character. Therefore, because the echo area of a string device is mapped to the nearest row/column, the actual echo area may not exactly correspond to the one specified.
- The action of each key is dependent on the type of GDDM device being used. Therefore, the string device may not operate as described above. Refer to a GDDM keyboard manual for a list of key actions.
- For prompt/echo type 2, the GDDM prompt string and input buffer share the same screen field. The length of this field is limited to the maximum input buffer size in bytes. One byte of this field is reserved for GDDM use. The total length, in bytes, of the prompt string and the length of the input buffer plus the extra byte required for GDDM must be less than or equal to the maximum input buffer size.
- Kanji (character set identifiers 128 and 134), Hangul (character set identifier 129), Traditional Chinese (character set identifier 130), and Simplified Chinese (character set identifier 132), and Unicode (character set identifier 131) are not supported.

Table 86. String Logical Devices - X Workstation Default Values

String Logical Devices	Data Type	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
String device number	I	1-16*	1-16*	1-16*	1-16*	**GPQLI** [*dev*]
Number of prompt/echo types	I	2	2	2	2	**GPQDST** [*necho*]
Available prompt/echo types (1, 2)	E	1, 2	1, 2	1, 2	1, 2	**GPQDST** [*echo*]
Default echo area	6[default]R	0.0-0.425, 0.0-0.0133, 0.0-0.425*	0.0-0.425, 0.0-0.0133, 0.0-0.425*	0.0-0.425, 0.0-0.0133, 0.0-0.425*	0.0-0.28442, 0.0-0.00711, 0.0-0.28448 *	**GPQDST** [*area*]
Available supported input character sets (1=PRIMARY, 2=ALL)	E	ALL	ALL	ALL	ALL	
Physical input device type for the measure (1=BUTTON, 2=SCALAR, 3=2D_VECTOR)	I	BUTTON	BUTTON	BUTTON	BUTTON	**GPQSPD** [*category*]
Physical input device number for the measure	I	4 (keyboard)	4 (keyboard)	4 (keyboard)	4 (keyboard)	**GPQSPD** [*pdevice*]
Maximum input buffer size	I	120	120	120	120	**GPQDST** [*size*]
Default initial string input buffer	I	120	120	120	120	**GPQDST** [*buflen*]
Length of default initial string	I	0	0	0	0	**GPQPCS** [*strlen*]
Default editing mode	I	1	1	1	1	**GPQDST** [*editpos*]

Note:

[1]See the text prior to General Workstation Facilities for a list of DWA and XSOFT Adapters.

* See the text prior to this table for more information.

Table 87. String Logical Devices Default Values

String Logical Devices	Data Type	6090	5080	GDDM	Inquiry
String device number	I	1-16*	1-16*	1-16*	**GPQLI** [*dev*]
Number of prompt/echo types	I	2	2	2	**GPQDST** [*necho*]
Available prompt/echo types (1, 2)	E	1, 2	1, 2	1, 2	**GPQDST** [*echo*]

Table 87. String Logical Devices Default Values (continued)

String Logical Devices	Data Type	6090	5080	GDDM	Inquiry
Default echo area	6[default]R	0.0-0.425, 0.0-0.0133, 0.0-0.425*	0.0-0.28448, 0.0-0.00711, 0.0-0.28448*	0.0-0.24682, 0.0-0.17574, 0.0-0.24682*	**GPQDST** [*area*]
Available supported input character sets (1=PRIMARY, 2=ALL)	E	ALL	ALL	PRIMARY	
Physical input device type for the measure (1=BUTTON, 2=SCALAR, 3=2D_VECTOR)	E	BUTTON	BUTTON	BUTTON	**GPQSPD** [*category*]
Physical input device number for the measure	I	4 (keyboard)	4 (keyboard)	4 (keyboard)	**GPQSPD** [*pdevice*]
Maximum input buffer size	I	160	80	80*	**GPQDST** [*size*]
Default initial string input buffer size	I	160	80	80	**GPQDST** [*buflen*]
Length of default initial string	I	0	0	0	**GPQST** [*strlen*]
Default editing position	I	1	1	1	**GPQDST** [*editpos*]
Note: See the text prior to this table for more information.					

Table 88. String Triggers

Device Number	Trigger Level	Default Trigger Type	Default Low Qualifier	Default High Qualifier	Available Trigger Types
For the X workstation					
String 1-16	0	4	65538	65538	1, 2, 3, 4[1] *
For the 6090 and 5080					
String 1-16	0	4	65537	65537	1, 2, 3, 4[1] *
Note: [1] GDDM has no available trigger types. * See the text prior to this table for more information.					

Button Devices

A button device reports a button number when a button is pressed. Some devices also report the release of a button. In general, workstations support the following button devices:

- Lighted Program Function Keys (LPFKs)
- Pointing devices

  - Four-button cursor controller
  - Stylus
  - Mouse
- Program Function (PF) keys on the keyboard
- Keyboard.

See The graPHIGS API and X Input Relationship section more information concerning (LPFKs).

The effect of button physical device 4 (keyboard) as a trigger is limited when processing national languages. During pre-editing mode or when auxiliary translation windows are present, the keyboard does not provide input to any measure or trigger process. For example, during pre-editing mode, an application does not receive any choice device 4 input events.

The presence of flags and number of value range descriptors depend on the hardware configuration of your workstation.

The flag attribute `2=NOT_PRESENT_CAN_BE_EMULATED` is available via the Create Input Device (`CLDEVS`) procopt. See CLDEVS (Create Input Device).

Table 89. Button Physical Devices (Values 1 - 4) - X Default Values

Button Physical Devices	**Data Type**	**POWER GT4 Family and POWER GTO**	**DWA Adapters[1]**	**XSOFT Adapters[1]**	**XLIB Adapters**	**Inquiry**
Button device number	I	1) LPFKs 2) pointing device 3) PF keys 4) keyboard	1) LPFKs 2) pointing device 3) PF keys 4) keyboard	1) LPFKs 2) pointing device 3) PF keys 4) keyboard	1) LPFKs 2) pointing device 3) PF keys 4) keyboard	**GPQPDC** [*device*]
Flags (`1=NOT_PRESENT_CANNOT_BE_ EMULATED, 2=NOT_PRESENT_CAN_BE_ EMULATED, 3=PRESENT_CANNOT_BE_EMULATED, 4=PRESENT_CAN_BE_EMULATED`)	E	`PRESENT_CAN_BE_EMULATED*`	`PRESENT_CAN_BE_EMULATED*`	`PRESENT_CAN_BE_EMULATED*`	`PRESENT_CAN_BE_EMULATED*`	**GPQPDC** [*flags*]
Device sub-type (`1=KEYBOARD, 2=OTHER`)	E	1) `OTHER` 2) `OTHER` 3) `OTHER` 4) `KEYBOARD`	1) `OTHER` 2) `OTHER` 3) `OTHER` 4) `KEYBOARD`	1) `OTHER` 2) `OTHER` 3) `OTHER` 4) `KEYBOARD`	1) `OTHER` 2) `OTHER` 3) `OTHER` 4) `KEYBOARD`	**GPQPDC** [*type*]
Number of value range descriptors	I	1) 1 2) 1* 3) 1 4) 19*	1) 1 2) 1* 3) 1 4) 19*	1) 1 2) 1* 3) 1 4) 19*	1) 1 2) 1* 3) 1 4) 19*	**GPQPDC** [*totnum*]
Value range descriptors	I	1) 1-32 2) 1-8* 3) 1 to n^2	1) 1-32 2) 1-8* 3) 1 to n^2	1) 1-32 2) 1-8* 3) 1 to n^2	1) 1-32 2) 1-8* 3) 1 to n^2	**GPQPDC** [*vrange*]

Table 89. Button Physical Devices (Values 1 - 4) - X Default Values (continued)

Button Physical Devices	Data Type	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Note: [1]See the text prior to General Workstation Facilities for a list of DWA and XSOFT Adapters. [2]*n* can be >=32 depending on the X server and the keyboard being used with your workstation. * See the text prior to this table for more information.						

Table 90. Button Physical Devices Default Values

Button Logical Devices	Data Type	6090	5080	GDDM	Inquiry
Button device number	I	1) LPFKs 2) pointing device 3) PF keys 4) keyboard	1) LPFKs 2) pointing device 3) PF keys 4) keyboard	1) PF keys 2) mouse 4) keyboard	**GPQPDC** [*device*]
Flags (1= NOT_PRESENT_CANNOT_BE_EMULATED, 2= NOT_PRESENT_CAN_BE_EMULATED, 3= PRESENT_CANNOT_BE_EMULATED, 4= PRESENT_CAN_BE_EMULATED)	E	PRESENT_ CAN_ BE_ EMULATED*	PRESENT_ CANNOT_ BE_ EMULATED	PRESENT_ CANNOT_ BE_ EMULATED	**GPQPDC** [*flags*]
Device sub-type (1=KEYBOARD, 2=OTHER)	E	1) OTHER 2) OTHER 3) OTHER 4) KEYBOARD	1) OTHER 2) OTHER 3) OTHER 4) KEYBOARD	1) OTHER 2) OTHER 4) KEYBOARD	**GPQPDC** [*type*]
Number of value range descriptors	I	1) 1 2) 1 3) 1 4) 19*	1) 1 2) 1* 3) 1 4) 19	1) 1 2) 1 4) 2	**GPQPDC** [*totnum*]
Value range descriptors	I	1) 1-32 2) 1-8 3) 1-41	1) 1-32 2) 1-8* 3) 1- *n* [1]	1) 1-24 2) 1-3 4) 65537 & 65539	**GPQPDC** [*vrange*]
Note: [1] *n* can be <=41 depending on the keyboard being used with your workstation. * See the text prior to this table for more information.					

Scalar Devices

A scalar device reports a value within some continuous range. The graPHIGS API supports dial scalar devices which report values as relative values.

See Ref #1. under "The graPHIGS API and X Input Relationship" section in Chapter 2 for information concerning scalar devices (dials).

The presence of flags depend on the hardware configuration of your workstation. The flag attribute 2=NOT_PRESENT_CAN_BE_EMULATED is available via the Create Input Device (CLDEVS) procopt. See CLDEVS (Create Input Device).

Table 91. Scalar Devices - X Workstation Default Values

Scalar Physical Devices	Data Type	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Scalar device number	I	1-8 (dials)	1-8 (dials)	1-8 (dials)	1-8 (dials)	**GPQPDC** [*pdevice*]
Flags (1= NOT_PRESENT_CANNOT_BE_EMULATED, 2= NOT_PRESENT_CAN_BE_EMULATED, 3= PRESENT_CANNOT_BE_EMULATED, 4= PRESENT_CAN_BE_EMULATED)	E	PRESENT_ CAN_ BE_ EMULATED*	PRESENT_ CAN_ BE_ EMULATED*	PRESENT_ CAN_ BE_ EMULATED*	PRESENT_ CAN_ BE_ EMULATED*	**GPQPDC** [*flags*]
Device sub-type (1=ABSOLUTE, 2=RELATIVE)	E	RELATIVE	RELATIVE	RELATIVE	RELATIVE	**GPQPDC** [*type*]
Number of value range descriptors	I	1	1	1	1	**GPQPDC** [*totnum*]
Value range descriptors	I	0-256	0-256	0-256	0-256	**GPQPDC** [*vrange*]
Note: [1]See the text prior to General Workstation Facilities, for a list of DWA and XSOFT Adapters. * See the text prior to this table for more information.						

Table 92. Scalar Devices Default Values

Scalar Physical Devices	Data Type	6090	5080	Inquiry
Scalar device number	I	1-8 (dials)	1-8 (dials)	**GPQPDC** [*pdevice*]
Flags (1= NOT_PRESENT_CANNOT_BE_EMULATED, 2= NOT_PRESENT_CAN_BE_EMULATED, 3= PRESENT_CANNOT_BE_EMULATED, 4= PRESENT_CAN_BE_EMULATED)	E	PRESENT_ CAN_ BE_ EMULATED*	PRESENT_ CAN_ BE_ EMULATED	**GPQPDC** [*flags*]
Device sub-type (1=ABSOLUTE, 2=RELATIVE)	E	RELATIVE	RELATIVE	**GPQPDC** [*type*]
Number of value range descriptors	I	1	1	**GPQPDC** [*totnum*]
Value range descriptors	I	0-256	0-256	**GPQPDC** [*vrange*]
Note: See the text prior to this table for more information.				

Vector Devices

A vector device reports an *x*, *y* value within some continuous range. The device can provide either absolute or relative values.

The graPHIGS API supports 2D vector devices which include the tablet and the mouse. The tablet reports absolute values and the mouse reports relative values.

The value range descriptors depend on the hardware configuration of your workstation.

See The graPHIGS API and X Input Relationship section for more information concerning vector devices.

Table 93. Vector Physical Devices - X Workstation Default Values

Vector Physical Devices	Data Type	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Vector device number	I	1 (tablet or mouse)	1 (tablet or mouse)	1 (tablet or mouse)	1 (tablet or mouse)	**GPQSPD** [*pdevice*]
Flags (1= NOT_PRESENT_CANNOT_BE_EMULATED, 2= NOT_PRESENT_CAN_BE_EMULATED, 3= PRESENT_CANNOT_BE_EMULATED, 4= PRESENT_CAN_BE_EMULATED)	E	PRESENT_ CAN_ BE_ EMULATED	PRESENT_ CAN_ BE_ EMULATED	PRESENT_ CAN_ BE_ EMULATED	PRESENT_ CAN_ BE_ EMULATED	**GPQPDC** [*flags*]
Type of physical input device (1=ABSOLUTE, 2=RELATIVE)	E	ABSOLUTE	ABSOLUTE	ABSOLUTE	ABSOLUTE	**GPQPDC** [*type*]
Number of value range descriptors	I	2	2	2	2	**GPQPDC** [*totnum*]

Table 93. Vector Physical Devices - X Workstation Default Values (continued)

Vector Physical Devices	Data Type	POWER GT4 Family and POWER GTO	DWA Adapters[1]	XSOFT Adapters[1]	XLIB Adapters	Inquiry
Value range descriptors	I[default]2 I[default]2	(0,65535), (0,65535)*	(0,65535), (0,65535)*	(0,65535), (0,65535)*	(0,65535), (0,65535)*	**GPQPDC** [*vrange*]
Note: [1]See the text prior to General Workstation Facilities for a list of DWA and XSOFT Adapters. * See the text prior to this table for more information.						

Table 94. Vector Physical Devices Default Values

Vector Logical Devices	Data Type	6090	5080	GDDM	Inquiry
Vector device number	I	1 (tablet)	1 (tablet)	1 (mouse, keyboard, or tablet)	**GPQSPD** [*pdevice*]
Flags (1= NOT_PRESENT_CANNOT_BE_EMULATED, 2=NOT_PRESENT_CAN_BE_EMULATED, PRESENT_CANNOT_BE_EMULATED, 4=PRESENT_CAN_BE_EMULATED)	E	PRESENT_ CAN_ BE_ EMULATED	PRESENT_ CANNOT_ BE_ EMULATED	PRESENT_ CANNOT_ BE_ EMULATED	**GPQPDC** [*flags*]
Type of physical input device (1=ABSOLUTE, 1=RELATIVE)	E	ABSOLUTE	ABSOLUTE	ABSOLUTE or RELATIVE	**GPQPDC** [*type*]
Number of value range descriptors	I	2	2	2	**GPQPDC** [*totnum*]
Value range descriptors	I[default]2 I[default]2	(0,1279), (0,1023)	(0,1023), (0,1023)	(0,1023), (0,1023)	**GPQPDC** [*vrange*]
Note: See the text prior to this table for more information.					

Break Action

The break action for most workstations is the Cancel key. The break action applies to all device classes when a device is in 1=REQUEST mode.

The available trigger types depend on the hardware configuration of your workstation. If you do not have lighted program function keys (LPFKs) installed, then trigger type 1 is not available.

X and XSOFT The break action is programmable. The Scroll Lock key is the default break action.

6090, 5080 The break action is programmable.

GDDM Press the Clear key on the keyboard. The break action is not programmable.

The following table describes the available break triggers and the default break action.

Table 95. Break Action Triggers

Default Trigger Type	Default Trigger Qualifier	Available Trigger Types
4	65539	1, 2, 3, 4*
Note: See the text prior to this table for more information.		

Part 2. Distributed graPHIGS API

Chapter 4. The graPHIGS API Nucleus

Connecting to the Nucleus

Version 2 of the graPHIGS API separates the application interface functions from the graphic management functions by introducing the concept of a shell and a nucleus. The shell is a subroutine library that is part of the application process. It accepts the call from the application, checks the parameters, and transmits the request to the nucleus.

The nucleus receives and processes requests from the application process. A graPHIGS API nucleus controls resources that are created by the application process. These resources include workstations (represented by a workstation state list), structure stores (represented by a structure store state list), image boards, and font directories. The requests sent to the nucleus are directed to one of these resources controlled by the nucleus.

All releases of the graPHIGS API follow the Version 2 architecture. For Version 2 of the graPHIGS API, a nucleus may reside in any of three environments:

- S/390 environment (VM or MVS)
- AIX environment
- 6090 environment

Before your application's request can be sent to a specific nucleus for processing, the application process must connect to that nucleus. Connecting to a nucleus involves selecting a connection method supported by the nucleus and a corresponding nucleus connection specification. Connection methods and connection specifications are selected by explicitly issuing a Connect to Nucleus subroutine call (**GPCNC**) and/or by specifying a default option (DEFNUC) in either the External Defaults File (EDF) or the Application Default Interface Block (ADIB). The specified connection method and specification can also be controlled through use of the TONUC default option. See Advanced Concepts for more information about connecting to a nucleus.

Managing the graPHIGS API Nucleus in AIX

In the AIX environment, a remote graPHIGS API nucleus must have been explicitly started *before* the application attempts connection to it. The graPHIGS API install procedures provide commands to start the nucleus as a process, manage access to it, and to terminate the process and free up the resources at the close of the application. Errors from these procedures do not go to the standard error path but rather return through the application shells as graPHIGS API messages.

Each command has an associated shell script which supplies default values to it. You may modify the scripts to add additional parameters, tailoring them to start customized environments.

The Remote graPHIGS API Nucleus's TCP/IP Port Number (RS/6000 only)

In order for an application to communicate via the **SOCKETS** connection method (TCP/IP) to a remote graPHIGS API nucleus, the graPHIGS API shell, which is linked with the application, must know the name of the host on which the graPHIGS API nucleus is running and the TCP/IP port number, on which the graPHIGS API nucleus is ″listening″ for connections. The hostname is provided as the first part of the *connection specification* when the host connects to the nucleus. The TCP/IP port number must be determined at run time.

The remote graPHIGS API nucleus, when started (with the **gPinit** command), needs to know on which TCP/IP port number to ″listen″ for connections from graPHIGS API shells.

The TCP/IP port number is determined by adding the nucleus connection number of the target remote graPHIGS API nucleus and the base port number as defined in the system file **/etc/services**. The default base port number is 8000.

Notes:

1. The remote graPHIGS API nucleus uses the **/etc/services** file on the same host on which it is started. The graPHIGS API shell uses the **/etc/services** file on which the application is running. Therefore, it is mandatory that the base port number, as defined in the **/etc/services** file on each host, is the same for all hosts which use the remote graPHIGS API nuclei.
2. The use of a hostname alias to direct routing over redundant networks may result in refused connections because the graPHIGS API remote nucleus and **gPgated** authorize access by hostname. When the **gPhost** or the **chgPcon** command is issued, the target hostname must match the hostname configured on the target host.
3. The user should be aware of potential conflicts with other services defined in **/etc/services**. Remote graPHIGS API nuclei with identification numbers of 0 through 255 will attempt to use TCP/IP port numbers 8000 through 8255, respectively, although only the base port number of 8000 will be reserved in **/etc/services**.

The Nucleus Description Table

The Nucleus Description Table contains configuration information for a graPHIGS API nucleus. The values in the NDT are provided by and initialized by the system. Inquiries to an NDT return information about a nucleus' default values and general characteristics.

Table 96. Nucleus Facilities

Nucleus Facilities	Data Type	6090 Values	S/390 Values	AIX Values	Inquiry
Nucleus storage size	I	*n* bytes	No limit[1]	No limit[1]	**GPQNCS** *[size]*
Note: [1] "No limit" returns a 0 on the inquiry.					

Table 97. Nucleus Description Table for Workstation

Workstation Facilities	Data Type	6090 Values	S/390 Values	AIX Values	Inquiry
Maximum number of simultaneously opened workstations	I	1	No limit[1]	No limit[1]	N/A
Number of available generic workstation types	I	1	3	9	**GPQWTN** *[totnum]*
Available generic workstation types	S	'6090 '	'5080 ' 'GDDM ' 'GDF '	'X ' 'GDF ' 'CGM ' 'XLIB ' 'XDWA ' 'NATIVE ' 'XPEX ' 'XSOFT ' 'IMAGE '	**GPQWTN** *[wstype]*
Note: [1] "No limit" returns a 0 on the inquiry.					

Table 98. Nucleus Description Table for Structure Store

Structure Store Facilities	Data Type	6090 Values	S/390 Values	AIX Values	Inquiry
Maximum number of simultaneously existing structure stores	I	1	No limit[1]	No limit[1]	N/A
Maximum number of simultaneously associated workstations	I	1	4	32[2]	**GPQWTN** *[maxa]*
Number of available structure store types	I	1	1	1	N/A
Available structure store types (NORMAL)	E	NORMAL	NORMAL	NORMAL	N/A
Note: [1] ″No limit″ returns a 0 on the inquiry. [2] See MAXWKS (Maximum Workstation Support) for more details.					

Table 99. Nucleus Description Table for Image Board

Image Board Facilities	Data Type	6090 Values	S/390 Values	AIX Values	Inquiry
Maximum number of simultaneously existing image boards	I	No limit[1]	N/A	No limit[1]	N/A
Maximum number of simultaneously associated workstations	I	1	N/A	No limit[1]	N/A
Number of available image board types	I	1	N/A	1	N/A
Available image board types (NORMAL)	E	NORMAL	N/A	NORMAL	N/A
Maximum image board size	2[default]I	No limit[1]	N/A	No limit[1]	**GPQIBF** *[n,v]*
Number of available image board bit depths	I	5	N/A	5	**GPQIBF** *[totnum]*
Available image board bit depths (bit depth 1, 2, 4, 8, 12)	E	1, 2, 4, 8, 12	N/A	1, 2, 4, 8, 12	**GPQIBF** *[depth]*
Number of available two-operand pixel operations	I	1	N/A	1	**GPQPO** *[totnum]*

Table 99. Nucleus Description Table for Image Board (continued)

Image Board Facilities	Data Type	6090 Values	S/390 Values	AIX Values	Inquiry
List of available two-operand pixel operations (REFLECTION)	E	REFLECTION	N/A	REFLECTION	**GPQPO** *[op]*
Number of available three-operand pixel operations	I	2	N/A	2	**GPQPO** *[totnum]*
List of available three-operand pixel operations (LOGICAL OPERATION, ARITHMETIC OPERATION)	E	LOGICAL OPERATION, ARITHMETIC OPERATION	N/A	LOGICAL OPERATION, ARITHMETIC OPERATION	**GPQPO** *[op]*
Number of available pixel data formats	I	1	N/A	1	N/A
Available pixel data formats (PIXEL ARRAY)	E	PIXEL ARRAY	N/A	PIXEL ARRAY	N/A
Note: [1] "No limit" returns a 0 on the inquiry.					

Table 100. Nucleus Description Table for Font Directory

Font Directory Facilities	Data Type	6090 Values	S/390 Values	AIX Values	Inquiry
Maximum number of simultaneously existing font directories	I	1	No limit[1]	No limit[1]	N/A
Maximum number of simultaneously associated workstations	I	1	No limit[1]	No limit[1]	N/A
Number of available font directory types	I	1	1	1	N/A
Available font directory types (NORMAL)	E	NORMAL	NORMAL	NORMAL	N/A
Note: [1] "No limit" returns a 0 on the inquiry.					

Table 101. Nucleus Description Table for Application Region

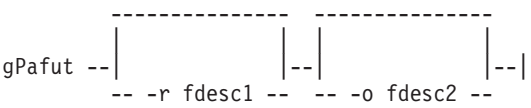
Application Region Facilities	Data Type	6090 Values	S/390 Values	AIX Values	Inquiry
Maximum number of simultaneously defined application regions[1]	I	No limit[2]	N/A	N/A	N/A
Note: [1] The 6090 workstation supports as many application regions as storage is available. However, only up to three of the distributed applications may connect to the 6090 nucleus at any one time. [2] ″No limit″ returns a 0 on the inquiry.					

gPafut Command

Purpose

Recover an archive file that has been left in an incomplete state.

Syntax



Description

The **gPafut** command recovers the data in a specified archive file so that the archive file is usable by the graPHIGS API. An archive file can be left in an unusable state when the application updating it (by archiving structures to the file or deleting structures from the file) terminates abnormally. If the archive file is incomplete, then the application receives one of two messages: `1110 CONCURRENT USAGE OF FILE` *a2* `NOT ALLOWED` (because it appears that another application has already opened the file) or `1205 FILE IS NOT A VALID graPHIGS ARCHIVE FILE` (because various fields are set incorrectly).

Two options are available for use with the **gPafut** command:

-r Specifies the recover option. The data in the archive file identified by *fdesc1* is recovered.

-o Specifies the output option. The data in the archive file identified by *fdesc1* is placed in the file identified by *fdesc2*.

fdesc1 and *fdesc2* identify archive files with the following information:

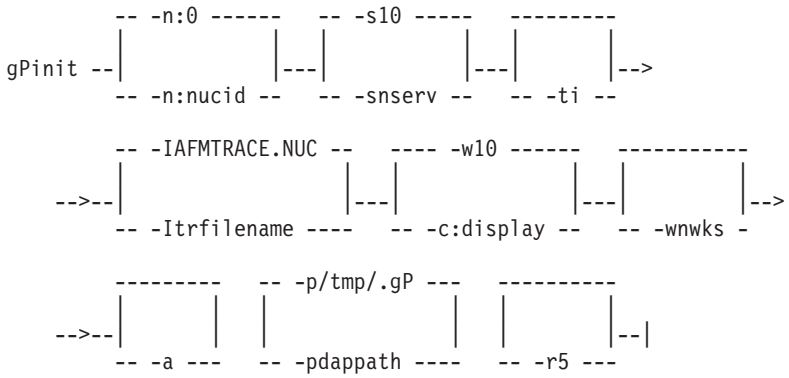
- For the AIX, `[/path/]fn[screen`
- For MVS, `ddname`
- For VM, `fn ft [fm]`

gPinit Command

Purpose

Starts a remote graPHIGS API nucleus.

Syntax



Description

The **gPinit** command starts a remote graPHIGS API nucleus as a separate process named gP. The nucleus is initialized with the default state list and resource stores. It is then ready to accept connections from applications/graPHIGS API shells which connect to a remote nucleus and use this nucleus as the connection specification.

The connection specification that should be used to connect to this nucleus is:

[*hostname*]:*nucid*

- *hostname* is the name of the host on which the **gPinit** command was issued. If not specified, the remote graPHIGS API nucleus is assumed to be running on the same host as the application/graPHIGS API shell.
- *nucid* is the number specified with the -n option of the **gPinit** command (default is 0).

If **gPinit** is unsuccessful, the following message will appear at the terminal:

```
AFM605 gP IS UNABLE TO START A REMOTE NUCLEUS
```

If unable to easily determine the cause for the failure, invoke **gPq** with the nucleus ID that you intended to use:

```
gPq -n:nucid
```

If **gPq** returns the process ID of an active nucleus, you have tried to start a nucleus with an identifier that is already in use. You must either terminate that active nucleus, or simply have your application connect to the active nucleus.

If **gPq** does not return the process ID of an active nucleus, the previous nucleus with the same identifier may not have terminated normally. To allow starting up another nucleus with the same identifier, issue **gPterm** with the same identifier:

```
gPterm -n:nucid
```

gPinit should now be able to start a nucleus with that identifier.

Flags

-n:*nucid*
: Specifies the numeric identification of this remote graPHIGS API nucleus. This number is used in the connection specification used by the application when connecting to this nucleus. The default is 0.

-s*nserv*
: Defines the maximum number of X workstation servers to which this nucleus will be able to connect. The default is 10.

-ti
Enables internal trace. See *The Personal graPHIGS Programming Interface: Installation and Problem Diagnosis*.

-l*trfilename*
Specifies the internal trace path name. The default is AFMTRACE.NUC.

-c:*display*
Forces the nucleus to act in conjunction with the specified X workstation server operating on the same host as the nucleus. See the **gPhost** command.

-w:*nwks*
Defines the maximum number of graPHIGS API workstations that may be associated to a single structure store. The default is ten. See Controlling the Environment with Defaults and Nicknames for more information.

-a
Enables the nucleus for DAP execution when the nucleus is started. The default is that the nucleus is not enabled for DAP execution.

-p*dappath*
Specifies the file path for the temporary storage of downloaded DAP files. The default file path is **/tmp/.gP**.

-r5
Specifies that the DAP to be executed should run as an X11R5 client. The default is that the DAP would run as an X11R6 client. This new option is especially useful for DAPs that depend on X11R5 functionality which no longer exists in X11R6; that is, XAsyncInput().

If the following message is observed while running an X11R5 DAP, then restart the graPHIGS remote nucleus by adding the -r5 option to the **gPinit** command:

> **Note:** The XAsyncInput API is no longer implemented in X11R6.

Examples

1. To start a remote graPHIGS API nucleus running with an identification number of 0, with the current directory as its working directory:

   ```
   gPinit
   ```

2. To start a remote graPHIGS API nucleus running with an identification number of 10, running in conjunction with an X server number 0, with the current directory as its working directory:

   ```
   gPinit -n:10 -c:0
   ```

3. To start a remote graPHIGS API nucleus running with an identification number of 0, with the current directory as its working directory, and increase the number of servers to which it is able to connect to 20:

   ```
   gPinit -s20
   ```

Files

/usr/bin/gPinit
/usr/lpp/graPHIGS/bin/gP

Related Information

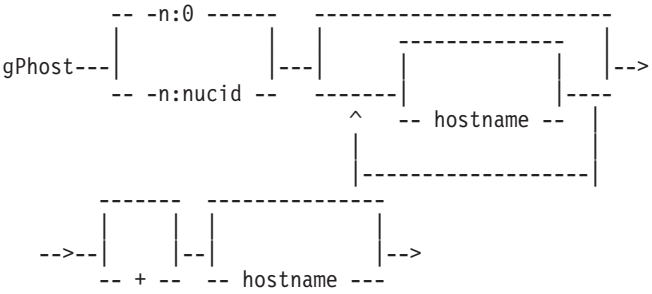
The **gPhost** command, **gPq** command, **gPterm** command, **makegP** command.

gPhost Command

Purpose

Changes the access security handling characteristics for a remote nucleus.

Syntax



Description

This command adds or deletes hosts from the access list maintained by the specified nucleus, or displays the current access list for the specified nucleus. **gPhost** may also be used to disable or enable host access checking. The nucleus must reside on the same host as where the command is issued.

Entering **gPhost** with no arguments, other than a -n, displays the current host access list for the specified nucleus.

If the specified nucleus was started and defined to run in conjunction with an X server (via **gPinit** with the -c flag), the server's security characteristics will be used in addition to whatever characteristics are defined for the nucleus. In this case, any host which is permitted to access the server will be allowed to access the nucleus.

The default access list for a remote nucleus is only the current host. To allow an application running on another host to connect to the specified nucleus, the other host must be on the access list of the nucleus, or access checking must be disabled.

Additional hosts may be added to the default host access list by creating the file **/etc/gP?.hosts** (the *?* is the identifier of the nucleus for which to enable access).

Entering **gPhost** with a + or a - as a flag by itself will disable or enable, respectively, host access checking for the specified nucleus.

If an application attempts to connect to a remote nucleus from a host which is not on the nucleus' host access list, the application will receive the graPHIGS API error number 208.

Flags

-n:*nucid*
Specifies the numeric identification of the remote graPHIGS API nucleus. The default is 0.

-*hostname*
Deletes the hostname from the access list of the specified nucleus. If no host name follows the -, host access checking is disabled.

+*hostname*
Adds the hostname to the access list of the specified nucleus. If no host name follows the +, host access checking is enabled. The + is optional; if a hostname is specified alone, it will be added to the access list.

Examples

1. To display the current host access list for the nucleus whose identification number is 2:
   ```
   gPhost -n:2
   ```
2. To disable access checking for the nucleus whose identification is 1:
   ```
   gPhost -n:1 +
   ```
3. To add the hosts named 'xyz' and 'abc' to the access list for the nucleus whose identification is 0:
   ```
   gPhost +xyz +abc
   ```

Files

/usr/bin/gPhost
/usr/lpp/graPHIGS/bin/gphost
/etc/gP?.hosts

Related Information

The **gPinit** command.

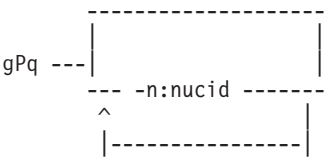
---

gPq Command

Purpose

Inquire information about remote graPHIGS API nuclei.

Syntax



Description

This command is intended to provide the user with a means of determining the state of remote graPHIGS API nuclei.

Issuing **gPq** with no arguments will display a list of all currently active nuclei running on the same host where the command is issued. The list consists of the nucleus' name followed by the process ID of the nucleus.

If one or more nucleus identifiers are specified, a list containing each specified nucleus ID and its process ID is displayed. If information for a nucleus associated with a particular identifier could not be found, message ″AFM0604″ will be displayed instead of the process ID. If a specified nucleus that was thought to be active was deemed ″Not Active″, try **gPq** again. If the nucleus is still ″Not Active″, it may be in some error state and should probably be terminated.

Flags

-n:*nucid* — Specifies the numeric identification of the remote graPHIGS API nucleus to query.

Examples

1. To display a list of all active remote graPHIGS API nuclei on the user's system:

   ```
   gPq
   ```

2. To display information about nuclei with identification numbers 0, 1, and 10:

   ```
   gPq -n:0 -n:1 -n:10
   ```

Files

/usr/bin/gPq
/usr/lpp/graPHIGS/bin/gpq

Related Information

The **gPinit** command, **gPterm** command.

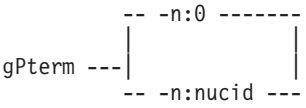
---

gPterm Command

Purpose

Terminates the specified remote graPHIGS API nucleus.

Syntax



Description

This command will stop the nucleus and release resources back to the operating system. The command must be issued from the same host as where the nucleus was started.

This command will also release resources held by a nucleus which terminated abnormally.

Flags

-n:*nucid* — Specifies the numeric identification of the remote graPHIGS API nucleus. The default is 0.

Files

/usr/bin/gPterm
/usr/lpp/graPHIGS/bin/gpterm

Related Information

The **gPinit** command.

makegP Command(AIX PS/2 only)

Purpose

Relinks the remote graPHIGS API nucleus. This command is not provided in graPHIGS API on the RS/6000.

Syntax

```
makegP
```

Description

This command relinks the remote graPHIGS API nucleus, allowing the end user to change which workstations the nucleus will support. To use **makegP**, you must be a member of the system group or have superuser authority. Be sure there are no remote graPHIGS nuclei active.

makegP is an interactive, menu-driven program.

Files

/usr/bin/makegP
/usr/lpp/graPHIGS/bin/gP
/usr/lpp/graPHIGS/bin/gP.o
/usr/lpp/graPHIGS/bin/gPAPA.o
/usr/lpp/graPHIGS/bin/gPAPAGDF.o
/usr/lpp/graPHIGS/bin/gPNOWS.o
/usr/lpp/graPHIGS/bin/gPX.o
/usr/lpp/graPHIGS/bin/gPXGDF.o
/usr/lib/libgP.a

Chapter 5. graPHIGS API Host and Workstation Connectivity

This chapter describes the following sections:

- The graPHIGS API Gateway Daemon
- The SOCKETS Connection Method
- graPHIGS/GAM Direct Connection

The graPHIGS API Gateway Daemon

Overview

The graPHIGS API gateway daemon, **gPgated**, is a process which enables a GDDM/graPHIGS API application using GAM to access a graPHIGS API remote nucleus, and graPHIGS API workstations managed by it. **gPgated** uses one or more IBM RS/6000 Host Interface Attachment (HIA) features connected to an IBM 5088 or 6098 communications controller to communicate to graPHIGS API nuclei running locally, or to graPHIGS API nuclei running remotely via TCP/IP over a LAN. The following figure is a high-level diagram that represents the components of this configuration:

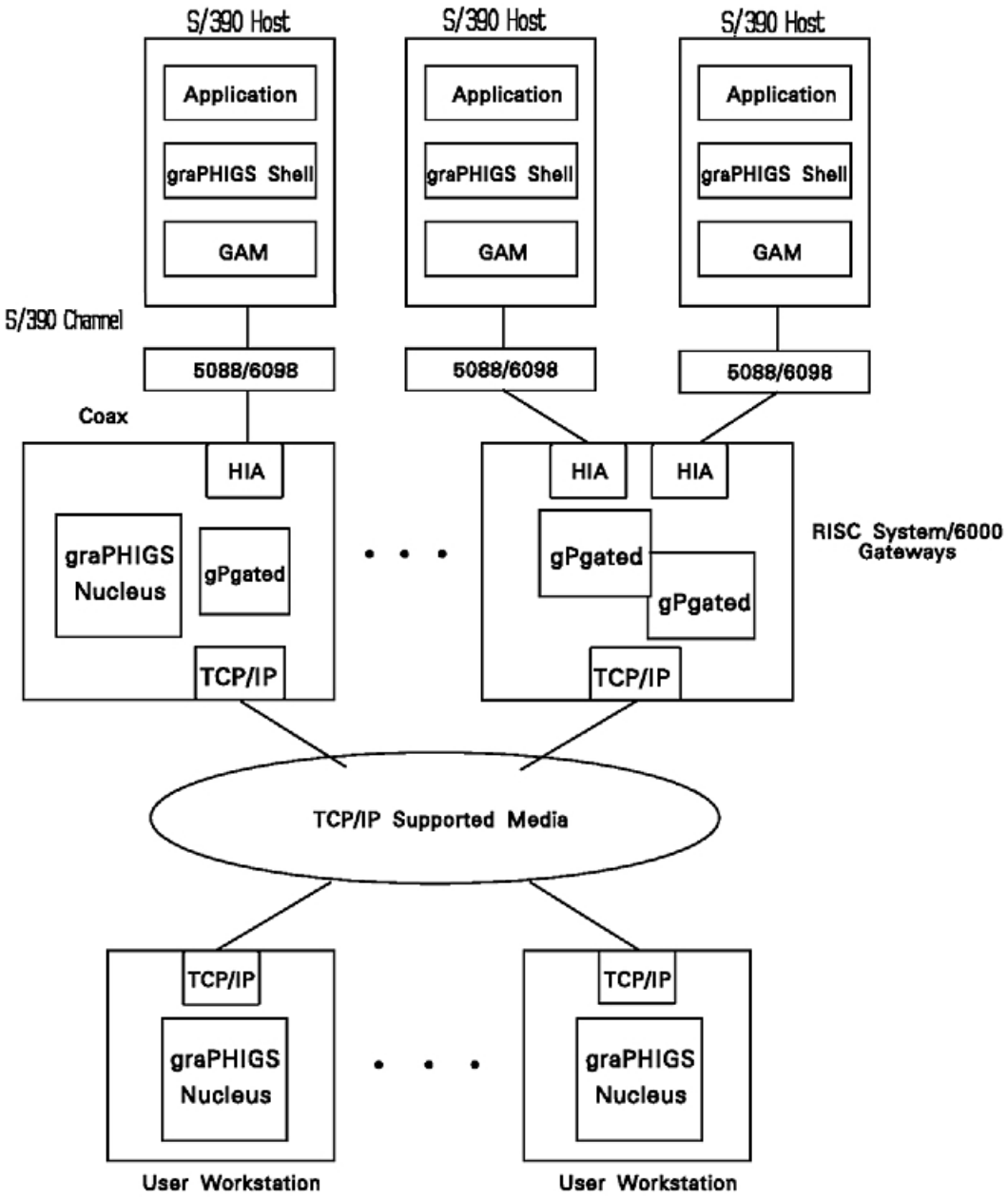


Figure 2. GDDM/graPHIGS API Application to graPHIGS API Remote Nucleus Flow. This diagram depicts a distributed graPHIGS application running on various hardware configurations which are attached through a central TCP/IP network. Several S/390 Host systems are shown running instances of the graPHIGS application and Shell, using the GAM to communicate across an IBM 5088 or 6098 communications controller to network-connected RS/6000 gateway systems.

The configuration is flexible.A few of the options are:

- Multiple graPHIGS API gateway daemons can exist on a LAN.
- Multiple gateway daemons can run on a RS/6000.

- The graPHIGS API gateway daemon can communicate with a remote nucleus running locally or to a remote nucleus running on any user workstation able to be reached via TCP/IP.
- A remote nucleus can communicate with multiple graPHIGS API gateway daemons and with other graPHIGS API shells.
- A remote nucleus can support multiple graphics adapters and remote X servers.
- Multiple remote nuclei may run in a user workstation.
- One application process can connect to multiple graPHIGS API remote nuclei.

With this many options, some tuning (i.e., configuration management) may be required to provide optimal performance and/or resource utilization. In this environment, items such as network capacity planning become very important.

Also, ensure that you have met the hardware and software requirements.

Customizing the graPHIGS API Gateway Daemon

After the hardware and software prerequisites are satisfied, the system administrator should use the **smit 5085** command to perform some HIA configuration at the gateway node. If this is a new adapter, you must add it as an *hia* and define:

- The link speed
- The number of 5080 addresses
- The 5080 base offset
- The 5080 channel address.

If you have or are going to install the 3270 Host Connection Program 2.1 and 1.3.3 for AIX: Guide and Reference, you should customize the HIA-attached 3270 sessions at this time as well.

This information defines the range of 5080 addresses that the gateway administers. The 5080 channel address is an arbitrary three- or four-digit hex number representing the first IBM S/390 device address assigned to the range. No checking is done on this data, but it is used in assigning connections and is passed to the user when the connection is granted.

The administrator may also want to customize the TCP/IP port number of the **gPgated** service in **/etc/services**. There is a default assignment of 7999 at the time the graPHIGS API API is installed. See The gPgated TCP/IP Port Number (RS/6000 only) for more information on the TCP/IP port number. The procedure for customizing the TCP/IP port number is identical to customizing for the graPHIGS API remote nucleus. See Problem Determination for gPgated, the graPHIGS API Gateway Daemon for information on customizing for a graPHIGS API remote nucleus.

Activating the graPHIGS API Gateway Daemon

Once the HIA is configured, the **gPgated** command may be invoked to start the graPHIGS API gateway daemon.

For those administrators who wish to have a fixed GAM address to workstation connection bindings, use the **-p** option to predefine those connections. Note that a **gPhost** command must be issued at the user workstation to allow the daemon to communicate with the remote nucleus. You could include this in a shell script that is used to start the graPHIGS API remote nucleus. Under some conditions, a *not-ready-to-ready* interrupt may be required at the host to invoke a ″startup″ program. The **chgPcon** command allows you to do this.

Activating the User Workstation

The graPHIGS API remote nucleus must be running on the user workstation before communication may be initiated. Use the **gPinit** command to start a remote nucleus.

If you establish the connection profile entry for this workstation/nucleus configuration before the nucleus is initialized, then you must issue a **gPhost** command to permit the gateway daemon to transmit data to the nucleus. If you do not issue the **gPhost** command, then when the application attempts to open the connection, the following message appears:

```
AFM0208 CONNECTION NOT CURRENTLY PERMITTED FROM THIS HOST TO NUCLEUS 'hostname:0'
```

Using the graPHIGS API Gateway Daemon

To run a GDDM/graPHIGS API application to a Personal graPHIGS API remote nucleus, ensure that the following items have been completed:

1. The installation prerequisites need to be met and the configuration complete.
2. The graPHIGS API remote nucleus must be running at the user workstation as explained in Activating the User Workstation.
3. The graPHIGS API gateway daemon must be running as explained in Activating the graPHIGS API Gateway Daemon.
4. A runtime connection profile entry which associates an IBM S/390 device address with the target remote nucleus must be established. Refer to the section on **chgPcon**.
5. You may have to customize your application environment as explained in Customizing the Application Environment.
6. Start your GDDM/graPHIGS API application.

If you encounter problems with this procedure, refer to Problem Determination for gPgated, the graPHIGS API Gateway Daemon.

Customizing the Application Environment

To override application nucleus connection defaults, use the graPHIGS API default facilities as described in Controlling the Environment with Defaults and Nicknames. An example of a graPHIGS API *PROFILE* entry is:

```
AFMMDFT DEFNUC=(2, ibmagc)
   AFMMNICK TOCONNID=connid,TOWSTYPE=nucwstype
```

Where *ibmagc* is the `DDNAME` of the device, *connid* is the connection identifier which may be `*` or others, as defined in Controlling the Environment with Defaults and Nicknames, and *nucwstype* is the remote nucleus workstation type desired, e.g., X or CGM. Other parameters are allowed as desired. For example, you can use the `XNAME PROCOPT` (see XNAME (X Default String)) for installation specific information. In the default case where there are no matches in the **.Xdefaults** file, use `XNAME` to set the title information for the X-window opened for a created workstation. For example:

```
PROCOPT=((XNAME,MVS3_YourApplication_YourGRAFaddr))
```

Note: This example puts the string: `MVS3_Your application_YourGRAFaddr` in the X-window title bar:

Table 102. Notice to 6095 Users

6095 Users
If you are using GDDM/graPHIGS API on the 6095, at this time there are known limitations to function or functional differences that you may encounter. See Workstation Description Tables for these limitations.

Memory Configuration and Application Performance

For each connection where a gateway daemon runs, a certain amount of real memory and virtual memory (paging space) is required. A minimum of 128 Kb of real memory and 1024 Kb of paging space should be allocated for each connection. Therefore, a machine with a minimum of 16 Mb of real memory and 32 Mb of paging space is able to support eight connections. In order to support more than eight connections, more paging space is needed and more real memory is needed for optimum performance.

There is an option to allocate more memory to each connection to help buffer outbound data. Depending on the application and IBM S/390 loading, allocating more memory can have a profound effect on elapsed time for large model downloads. This buffer can be considered a speed matching device, or a balloon in the pipeline that allows data to be transmitted to the graPHIGS API remote nucleus as fast as can be accepted. The system administrator may want to use this option as a tuning parameter and trade off reduced model download time against increased real memory and virtual memory requirements.

The **lsgPcon** command enables the administrator to monitor the progress of data flow through the system. The **lsgPcon -S** command returns data that is useful in the tuning process. In particular, if paging is invoked on `buffers exceeded` a large number of times, the administrator might want to use the **-b** option to increase the buffer size.

The gPgated TCP/IP Port Number (RS/6000 only)

gPgated, **chgPcon**, and **lsgPcon** must use the same TCP/IP port number when communicating across a TCP/IP network.

The graPHIGS API gateway daemon needs to know on which TCP/IP port number to ″listen″ for information from **chgPcon** and **lsgPcon** commands. Likewise, these commands must know on which TCP/IP port number to send information. The **gPgated** TCP/IP port number is determined when the graPHIGS API gateway daemon is started by an inquiry to the **services** database (most commonly represented by the system file **/etc/services**) for the port number of the **gPgated** service. If any command fails to find the **gPgated** service in the services database, a default base port of 7999 is used.

Notes:

- The graPHIGS API gateway daemon, **gPgated**, uses the **services** database on the same host on which it is started. The **chgPcon** and **lsgPcon** commands use the **services** database on the host on which they are issued. Therefore, it is mandatory that the port number, as defined in the **services** database on each host, is the same for all hosts on which the **gPgated**, **chgPcon**, and **lsgPcon** commands are to be issued.
- The use of a hostname alias to direct routing over redundant networks may result in refused connections because the graPHIGS API remote nucleus and **gPgated** authorize access by hostname. When the **gPhost** or the **chgPcon** command is issued, the target hostname must match the hostname configured on the target host.

The SOCKETS Connection Method

Overview

The GDDM graPHIGS API provides SOCKETS communication so that an application executing in a VM or MVS environment can connect to a remote graPHIGS nucleus on a supported workstation, such as a RS/6000.

Note:

- Although the SOCKETS connection method is easier to use than the GAM connection method and provides comparable or better performance, it often consumes more mainframe CPU resources for the same application.
- We do not recommend using the SOCKETS connection method on MVS for production purposes. The implementation of TCP/IP Version 2 Release 2 on current MVS systems forces the address space of any application using the SOCKETS connection method to become *non-swappable*. are used then system wide performance will be negatively impacted. The impact will depend on the sizes of the applications and the amount of system memory. This problem does not exist in the VM environment. Furthermore, on current MVS systems, i.e. MVS Versions 2, 3, and 4, use of TCP/IP connections with Version 2 of TCP/IP forces the application address space to become *non-swappable*. are used then system wide performance is negatively

impacted. For this reason, customers are not encouraged to use the SOCKETS connection method on MVS for production purposes. This is not a problem for applications run in the VM environment.

Prerequisites

The SOCKETS connection method for the graPHIGS API requires that the IBM TCP/IP licensed program product be installed and operational on your CPU. The required versions are:

- On VM, TCP/IP Version 2 Release 1, or higher, program number 5735-FAL
- On MVS, TCP/IP Version 2 Release 2, or higher, program number 5735-HAL

If you want to take advantage of the host name resolution capability, then you need the run time libraries for the IBM C/370 program product. The required version is:

- C/370 Version 2 Release 1, or higher, program number 5688-039

Host name resolution is the ability to specify a host name instead of an IP address when using the Connect to Nucleus **(GPCNC)** subroutine or specifying the DEFNUC or TONUC options in either the External Defaults File (EDF) or the Application Default Interface Block (ADIB).

Specifying the Target Nucleus

To connect to a remote graPHIGS nucleus, an application must supply a nucleus connection specification using the Connect to Nucleus **(GPCNC)** subroutine specifying the DEFNUC or TONUC options in either the External Defaults File (EDF) or the Application Default Interface Block (ADIB).

For information about **GPCNC**, see GPCNC - Connect Nucleus. For more information about overriding application nucleus connection defaults, see Controlling the Environment with Defaults and Nicknames.

The specification is defined as *<hostname>:<nucleus_id>*. There are two ways this can be done, depending on whether you have host name resolution enabled. Generally, enabling host name resolution makes it easier to use. For example, a host called ″host1″, with internet address ″129.40.17.9″ is running a remote graPHIGS nucleus with id ″0″.

If host name resolution is enabled, you can use the following specifications:

```
AFMMDFT  DEFNUC=(3,host1:0)
AFMMNICK TOCONNID=host1:0
```

If host name resolution is NOT enabled, you need to use the following specifications:

```
AFMMDFT  DEFNUC=(3,129.40.17.9:0)
AFMMNICK TOCONNID=129.40.17.9:0
```

Application Customization When Using Host Name Resolution on MVS

The following libraries need to be concatenated to the STEPLIB of the job which runs the application to enable host name resolution:

```
//          DD   DSN=EDC.SEDCLINK,DISP=SHR
//          DD   DSN=PLI.SIBMLINK,DISP=SHR
```

These lines should be modified to specify the actual names of your IBM C/370 SEDCLINK and SIBMLINK datasets. Further information on link-editing and running C/370 for MVS applications can be found in the *IBM C/370: Programming Guide*, SC09-1384.

Application Customization When Using Host Name Resolution on VM

The AFMZRES MODULE, which was generated by your system programmer as part of the GDDM graPHIGS installation process, must be available (on an accessed disk) for your application while it is executing.

Also, the following libraries need to be GLOBALed when the application is executed (in addition to the graPHIGS API run-time libraries) to enable host name resolution:

```
GLOBAL TXTLIB COMMTXT IBMLIB EDCBASE
GLOBAL LOADLIB EDCLINK
```

Further information on link-editing and running C/370 for VM applications may be found in the *IBM C/370: Programming Guide*, SC09-1384.

Note: You may be required to use the **gPhost** command on the target workstation to allow the target nucleus to receive data from the application process.

Run-Time Errors

If an error related directly to the TCP/IP SOCKETS support for the graPHIGS API occurs, the following error will be reported by the graPHIGS API:

```
AFM0593 COMMUNICATION ERROR: MAJOR 7, MINOR xxx
```

The MINOR code will generally equate to a TCP/IP ″errno″ value, as defined in the TCPERRNO file shipped with the TCP/IP program products.

Error codes which are not defined in the TCPERRNO file are generally configuration problems, and should be report to system support personnel, or IBM Support personnel.

Configuration Details

See the GDDM graPHIGS program directory for more information on the customization and configuration that must be performed by your system programmer in order to use the SOCKETS connection method.

graPHIGS/GAM Direct Connection

Overview

The graPHIGS/GAM direct connection of the graPHIGS nucleus to 6098 with FDDI feature enables a host graPHIGS API application using GAM to access a workstation and graPHIGS API workstations managed by it. The direct connection function provides a high bandwidth, low-cost connection from the GDDM/graPHIGS API shell to a remote nucleus.

The following figure is a high-level diagram that represents the components of this configuration using a graPHIGS/GAM direct connection:

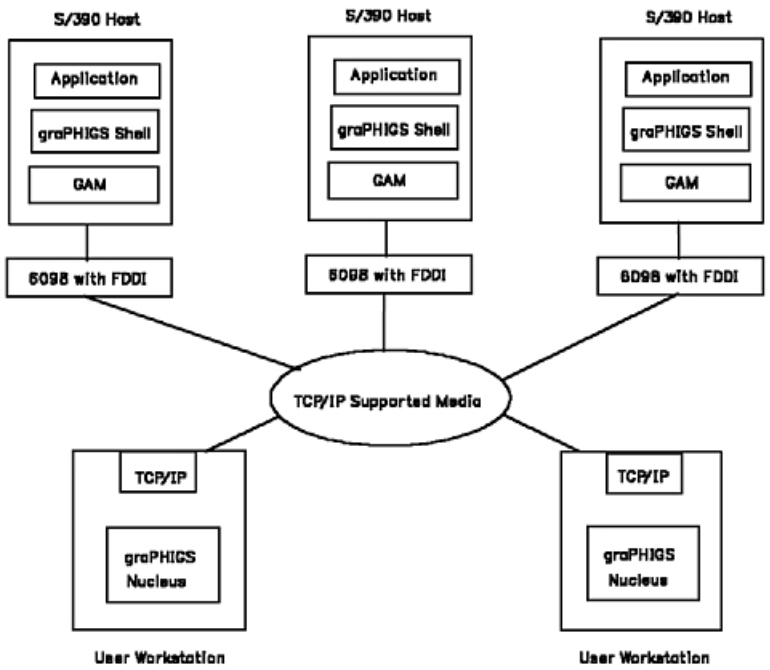


Figure 3. graPHIGS/GAM Direct Connection. This diagram depicts a distributed graPHIGS application in which the S/390 Host systems are using the GAM to communicate to network-connected 6098 systems with the FDDI feature.

The configuration is flexible. A few of the options are:

- Multiple 6098s with FDDI feature can exist on a LAN.
- Multiple 6098s with FDDI feature can run to a RS/6000.
- A remote nucleus can communicate with multiple 6098s with FDDI feature and with other graPHIGS API shells.
- A remote nucleus can support multiple graphics adapters and remote X servers.
- Multiple remote nuclei may run in a user workstation.
- One application process can connect to multiple graPHIGS API remote nuclei.

With this many options, some tuning (i.e., configuration management) may be required to provide optimal performance and/or resource utilization. In this environment, items such as network capacity planning become very important.

Customizing the 6098 with FDDI Feature

After the hardware and software prerequisites are satisfied, the system administrator should customize the 6098 with FDDI. Specifically, for users of the **mkgPcon** command, the system administrator may set up a 6098 configuration parameter called the environment descriptor to help in associating an IBM S/390 device address and a 6098 port number. See **mkgPcon** and **ls6098** for more information on the environment descriptor.

Activating the User Workstation

The graPHIGS API remote nucleus must be running on the user workstation before communication may be initiated. Use the **gPinit** command to start a remote nucleus.

Using the graPHIGS/GAM Direct Connection

To run a GDDM/graPHIGS API application to a Personal graPHIGS API remote nucleus, ensure that the following items have been completed:

1. The installation prerequisites need to be met and the configuration complete.

2. The graPHIGS API remote nucleus must be running at the user workstation as explained in Activating the User Workstation.
3. A connection which associates an IBM S/390 device address with the target remote nucleus must be established. Refer to the section on **mkgPcon**.
4. You may have to customize your application environment as explained in Customizing the Application Environment.
5. Start your GDDM/graPHIGS API application.

If you encounter problems with this procedure, see Problem Determination for gPgated, the graPHIGS API Gateway Daemon for more information.

Customizing the Application Environment

To override application nucleus connection defaults, use the graPHIGS API default facilities as described in Controlling the Environment with Defaults and Nicknames. An example of a graPHIGS API *PROFILE* entry is:

```
AFMMDFT DEFNUC=(2, ibmagc)
   AFMMNICK TOCONNID=connid,TOWSTYPE=nucwstype
```

Where *ibmagc* is the `DDNAME` of the device, *connid* is the connection identifier which may be `*` or others, as defined in Controlling the Environment with Defaults and Nicknames, and *nucwstype* is the remote nucleus workstation type desired, e.g., X or CGM. Other parameters are allowed as desired. For example, you can use the `XNAME PROCOPT` in Controlling the Environment with Defaults and Nicknames. (See XNAME (X Default String) for installation specific information).

In the default case where there are no matches in the **.Xdefaults** file, use `XNAME` to set the title information for the X-window opened for a created workstation. For example:

```
PROCOPT=((XNAME,MVS3_YourApplication_YourGRAFaddr))
```

Note: This example puts the string: `MVS3_Your application_YourGRAFaddr` in the X-window title bar.

Table 103. Notice to 6095 Users

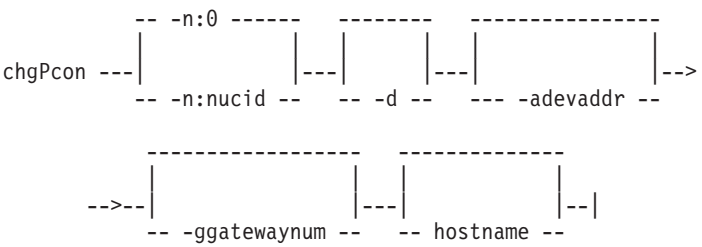
6095 Users
If you are using GDDM/graPHIGS API on the 6095, at this time there are known limitations to function or functional differences that you may encounter. See Workstation Description Tables for these limitations.

chgPcon Command

Purpose

Add/change/delete a runtime connection profile entry for a graPHIGS API gateway daemon.

Syntax



Description

This command modifies runtime connection profile entries for the graPHIGS API gateway daemon running on the specified host. If there is an entry for the specified IBM S/390 device address in the runtime connection profile, the entry is changed or deleted. Otherwise, the entry is added.

The command may be used to add or modify an entry such that a connection to an IBM S/390 device address allocated to the gateway may connect to the specified graPHIGS API remote nucleus. The nucleus must reside on the host where the command is issued. At the time the runtime connection profile entry is added, the host on which the graPHIGS API gateway daemon is running is granted access to the local nucleus, so that the **gPhost** command need not be used.

The command may also be used to delete a runtime connection profile entry.

If **chgPcon** is successful, the following message appears at the terminal:

```
gPgated: GAM address devaddr allocated to hostname:nucid
```

Where *devaddr* is the IBM S/390 device address used by the GDDM/graPHIGS API application, *hostname* is the host name of the local workstation and *nucid* is the nucleus identifier of the remote graPHIGS API nucleus to which the connection will be established.

If **chgPcon** is unsuccessful, one of the following messages appear at the terminal:

```
AFM0604  NUCLEUS n1 NOT STARTED OR NOT RESPONDING
AFM0616  NO DEVICE ADDRESSES AVAILABLE
AFM0617  DEVICE ADDRESS devaddr ALREADY ALLOCATED OR UNAVAILABLE
AFM0593  COMMUNICATION ERROR: MAJOR 7 MINOR 73
AFM1101  NOT ENOUGH STORAGE TO PERFORM REQUESTED FUNCTION
```

If you are having problems running the application, refer to Problem Determination for gPgated, the graPHIGS API Gateway Daemon.

Flags

-g:*gatewaynum*
: Specifies the numeric suffix to be appended to the HIA device name that is opened and managed. The default is zero, for example, `/dev/hia0`.

-n:*nucid*
: Specifies the nucleus identifier of the graPHIGS API remote nucleus. The default is zero.

-d
: Deletes the runtime connection profile entry which defines a connection between the specified remote nucleus and the IBM S/390 device address. A new runtime connection profile entry for the IBM S/390 device address may then be added. If the **-d** flag is not specified then an entry is changed or added instead of deleted.

  When the **-d** flag *is* specified but the **-a** flag is *not* specified, then the connection that is dropped is the first one encountered that conforms to the specified parameters. If there are multiple connections that match the specified parameters, then the user should consider using the **-a** flag unless the specifics of the connection are unimportant.

-a*devaddr*
: Requests that a specific GAM address *devaddr* be reserved or dropped according to the flags specified. If *hostname:nucid* already has allocated the address and the **-d** option is not specified, then a reset function is performed. This reset causes a *not-ready-to-ready* interrupt to be received at the IBM S/390 and all internal state information to be reset for that connection. The following message occurs:

  ```
  gPgated:  GAM address devaddr reset for hostname: nucid
  ```

hostname
: Specifies the name of the host where the graPHIGS API gateway daemon is running.

  If a *hostname* is not specified, the default is the local host.

Examples

1. To request a connection from a graPHIGS API gateway daemon running on the local host to the default nucleus (which has an identification number of zero):

   ```
   chgPcon
   ```

2. To request a connection from a graPHIGS API gateway daemon running on the local host to the graPHIGS API remote nucleus which has an identification number of two:

   ```
   chgPcon -n:2
   ```

3. To request a connection from a graPHIGS API gateway daemon running on the host *gatenode* to the default nucleus (which has an identification number of zero):

   ```
   chgPcon gatenode
   ```

4. To request a connection from a graPHIGS API gateway daemon running on the host *gatenode* to the remote nucleus which has an identification number of two:

   ```
   chgPcon -n:2 gatenode
   ```

5. To request a connection on a specific IBM S/390 device address from a graPHIGS API gateway daemon running on the host *gatenode* to the remote nucleus which has an identification number of two:

   ```
   chgPcon -a13cb -n:2 gatenode
   ```

6. To request a connection on a specific GAM address from a graPHIGS API gateway daemon that is managing the device **/dev/hia3** running on the host *gatenode*, to the remote nucleus which has an identification number of two:

   ```
   chgPcon -a13cb -n:2 -g3 gatenode
   ```

7. To drop a connection between the default nucleus and the graPHIGS API gateway daemon running on the host, *gatenode*:

   ```
   chgPcon -d gatenode
   ```

8. To drop a connection between the remote nucleus which has an identification number of two and the graPHIGS API gateway daemon running on the host *gatenode*:

   ```
   chgPcon -d -n:2 gatenode
   ```

9. To drop a connection between the nucleus which has an identification number of two, and the gateway daemon that is managing the device **/dev/hia1** running on the host, *gatenode*:

   ```
   chgPcon -d -n:2 -g1 gatenode
   ```

10. To drop a connection between the default nucleus and a graPHIGS API gateway daemon running on the local host:

    ```
    chgPcon -d
    ```

11. To drop a connection on a specific IBM S/390 address between the default nucleus and a graPHIGS API gateway daemon running on the local host:

    ```
    chgPcon -d -a13cb
    ```

Files

/usr/bin/chgPcon

Related Information

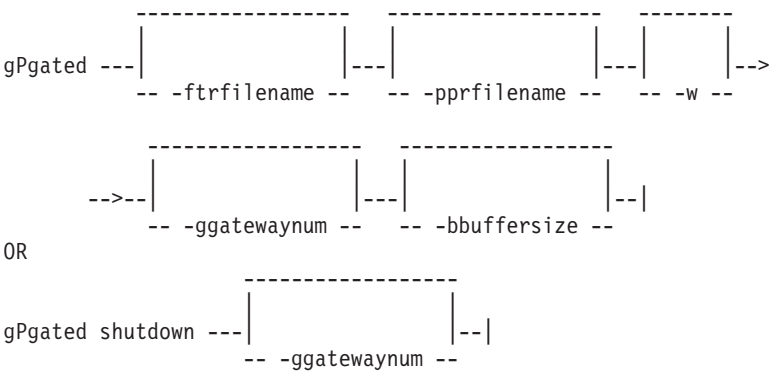
The **gPgated** command, **lsgPcon** command, **gPhost** command.

gPgated Command

Purpose

Start a graPHIGS API gateway daemon to allow a GDDM/graPHIGS API shell to communicate with Personal graPHIGS API remote nuclei.

Syntax



Description

The **gPgated** command starts a graPHIGS API gateway daemon. This daemon allows a GDDM/graPHIGS API application, using a 5088 or 6098 communications control unit and the GAM connection method, to communicate with Personal graPHIGS API remote nuclei. The daemon process runs on a RS/6000 that has a **5080 Host Interface Adapter** (HIA) installed and configured with 5080-type devices enabled. One IBM S/390 device address is required to access each connection to a remote nucleus. Each graPHIGS API gateway daemon can support up to sixteen concurrent connections.

The graPHIGS API gateway daemon is initialized with the specified options, which may assign some connection profile entries. The daemon is then ready to accept changes to the runtime connection profile via the **chgPcon** command or configuration inquiries via the **lsgPcon** command.

The runtime connection profile is a table of entries containing the following information:

- IBM S/390 device address for the connection
- Hostname and nucleus identifier of the graPHIGS API remote nucleus that receives the connection
- Connection state information.

When an entry exists in the runtime connection profile, whether it has been added due to the processing of a **chgPcon** command, or was defined in the initial configuration file specified by the **-p** option, the IBM S/390 device address state associated with that entry makes a *not-ready-to-ready* transition, i.e. comes *online*. When execution of the **chgPcon** command causes an entry to be deleted from the runtime connection profile, the IBM S/390 device address state associated with that entry is set to *not-ready*, or *offline*. If an entry is deleted while an application's device is `OPEN`, then an asynchronous error is generated before the state changes to *not-ready* to allow the application to terminate.

The runtime connection profile information is recorded in a ″warmstart″ data file. This data file may be used in the event of a system failure to recover connection profile information. For details, see the **-w** option described below.

Once the graPHIGS API gateway daemon is running and a runtime connection profile entry is established, a GDDM/graPHIGS API application may attempt to connect to a graPHIGS API remote nucleus by using the GAM connection method.

If **gPgated** is successful, a series of messages such as the following appear in the transaction file for each configuration file entry:

```
 gPgated: Device address
devaddr allocated to
hostname:nucid
```

Where *devaddr* is an address in the range defined in the HIA configuration, using the **smit 5085** command, and *hostname:nucid* is the hostname and nucleus identifier of the target graPHIGS API remote nucleus.

If the **-p** option is not specified, no messages appear.

If **gPgated** is unsuccessful, one of the following messages may appear at the terminal:

```
 AFM1101  NOT ENOUGH STORAGE TO PERFORM REQUESTED FUNCTION
 AFM1203  FILE SERVICE open ERROR RETURN CODE = 13 ON FILE xxxxx
 AFM1107  FILE /usr/profile NOT FOUND
 gPgated: Illegal option option
 gPgated already started
```

In these cases, the graPHIGS API gateway daemon terminates.

gPgated is intended to run as a daemon which provides service to users on the network continually. In the event that a reconfiguration needs to take place, or **gPgated** needs to be stopped, the **gPgated shutdown** command option causes it to gracefully exit. You must use either root privileges or the same userid that initiated **gPgated** to run this shutdown command, otherwise, the command is unsuccessful and the following error message occurs:

```
  AFM1201  SYSTEM SERVICE shmct1 ERROR RETURN CODE = 1
```

Note: If the gateway daemon terminates or is abnormally ended, you should use the **gPgated shutdown** command to perform the cleanup that normally occurs upon exit. This ensures normal startup when you next use the gateway daemon.

If you are having problems starting a gateway daemon, refer to Problem Determination for gPgated, the graPHIGS API Gateway Daemon.

Flags

-f*trfilename*
: Specifies the filename of the gateway transaction file. This file records all transactions processed by **gPgated**. It lists the time and date of each event, the requesting host and nucleus identifier, and the action performed. The default is standard out. Entries may appear as follows:

```
// Tue May 21 10:23:31 1991
  gPgated: GAM address e63 allocated to :0
//
```

Once initialization is completed, the **-f***trfilename* flag also causes **gPgated** to run as a background process.

-g*gatewaynum*
: Specifies the numeric suffix to be appended to the HIA device name that is opened and managed. The default is zero, for example, `/dev/hia0`

-p*prfilename*

Specifies the filename of the initial configuration file. This file allows connections to be predefined by a system administrator. The file has the following format:

- Entries are in the format *hexaddr hostname:nucid* [**-b***buffersize*]

  where *hexaddr* is the three or four digit hex GAM address to be assigned to the connection, *hostname* is the internet name of the target host, *nucid* is the graPHIGS API remote nucleus identifier to connect at the node *hostname*, and the **-b** option applies to each entry allowing the system administrator to tailor buffering to individual applications for specific users. See -b*buffersize* for the definition of the **-b** option.
- Comments are allowed and begin with the # symbol
- Blank lines are allowed.

Some sample file entries:

```
abc5 rover:0 #joes current workstation
abc6 rover:0 #joes current workstation
#This line is a comment.
abc7 :0 #administrators session on local node
abc8 stranger:255 #interesting setup
abc9 cad:0 -b1024 #1 Meg buffer on this connection
abca cad:1 -b0 #But no buffering allowed here
```

For each target remote nucleus in the file, the graPHIGS API gateway daemon attempts to issue a **gPhost** command to allow itself access to that nucleus. If the target remote nucleus is not running, the user needs to issue the **gPhost** command manually after the remote nucleus is started. This option is mutually exclusive with the **-w** option.

-w

Specifies that the initial configuration file saved during the last execution of **gPgated** be used to establish new connections, i.e. a″warmstart″ screen. It retrieves this data from the file **/tmp/.gP/gPgated.warmstart.data**, which is the file that **gPgated** uses to record the current runtime connection profile.

This option is mutually exclusive with the **-p** option.

-b*buffersize*

Specifies the size of the internal buffer pool to be used for each connection. This buffer is used when the graPHIGS API remote nucleus consumes data more slowly than the IBM S/390 produces it. *buffersize* is a positive decimal number indicating the number of 1024 byte blocks to use before invocation of the pacing mechanism. The default is 256 1024-byte blocks per connection.

Note: You may need to adjust the paging space when using large buffers.

shutdown

Requests that the gateway daemon be gracefully terminated. The session connection profile is saved to be used for a later ″warmstart″ and applications are notified that open connections are being broken as required. This option is mutually exclusive with all options other than the **-g** option.

Examples

1. To start a graPHIGS API gateway daemon:

   ```
   gPgated
   ```
2. To stop a graPHIGS API gateway daemon:

   ```
   gPgated shutdown
   ```
3. To start a graPHIGS API gateway daemon and assign connections as predefined in the file `myconfig`:

   ```
   gPgated -pmyconfig
   ```
4. To start a graPHIGS API gateway daemon and assign connections as predefined in the file `myconnections` and record the transactions in the file `mylog`:

   ```
   gPgated -pmyconnections -fmylog
   ```

5. To restart a graPHIGS API gateway daemon after a system crash and record the transactions in `mylog`:
   ```
   gPgated -w -fmylog
   ```
6. To restart a graPHIGS API gateway daemon after a system crash and use a buffer size of 65536:
   ```
   gPgated -w -b64
   ```
7. To start a graPHIGS API gateway daemon and use the adapter configured as **/dev/hia4**:
   ```
   gPgated -g4
   ```
8. To stop the graPHIGS API gateway daemon managing the adapter configured as **/dev/hia4**:
   ```
   gPgated shutdown -g4
   ```

Files

/usr/bin/gPgated
/usr/lpp/graPHIGS/bin/gPgated
/tmp/.gP/gPgated.warmstart.data0
/tmp/.gP/gPgated.warmstart.data1
/tmp/.gP/gPgated.warmstart.data2
/tmp/.gP/gPgated.warmstart.data3
/tmp/.gP/gPgated.warmstart.data4
/tmp/.gP/gPgated.warmstart.data5
/tmp/.gP/gPgated.warmstart.data0.bak
/tmp/.gP/gPgated.warmstart.data1.bak
/tmp/.gP/gPgated.warmstart.data2.bak
/tmp/.gP/gPgated.warmstart.data3.bak
/tmp/.gP/gPgated.warmstart.data4.bak
/tmp/.gP/gPgated.warmstart.data5.bak

Related Information

The **chgPcon** command, **lsgPcon** command, **gPhost** command.

---

ls6098 Command

Purpose

Inquire connection information for the 6098 with FDDI feature.

Syntax

```
ls6098 hostname
```

Description

This command inquires connection information of a 6098 with FDDI feature called *hostname*.

The following is an example of input for making an inquiry about connection information for a 6098 with FDDI feature:

```
ls6098 w20
```

where *w20* is the hostname.

If **ls6098** is successful, messages like the following appear at the terminal:

```
IBM 6098 named 'w20' returns descriptor 'KGNVMP: LAB: CHPID 19: w20'
Application Name        Starting Port,Number of Ports,State
FPGP TESTING USAGE      0008          0008            NOTBUSY
```

where *KGNVMP: LAB: CHPID 19: w20* is the environment descriptor or location of the 6098 with FDDI feature.

If **ls6098** is unsuccessful, one of the following messages appear at the terminal:

```
AFM0593  COMMUNICATION ERROR: MAJOR 7 MINOR 73
AFM0640  UNKNOWN HOST 'w20'
AFM0641  HOST 'w20' NOT RESPONDING
AFM0642  HOST 'w20' IS NOT AN IBM 6098 W/FDDI FEATURE
AFM1201  SYSTEM SERVICE read ERROR RETURN CODE = 73
```

Flags

hostname Specifies the hostname or IP address of the 6098 with FDDI feature to which the inquiry will be sent.

Examples

1. To inquire from the 6098 with FDDI feature called *hostname*, the connection information about the local host:

   ```
   ls6098 hostname
   ```

Files

/usr/bin/ls6098

Related Information

The **mkgPcon** command

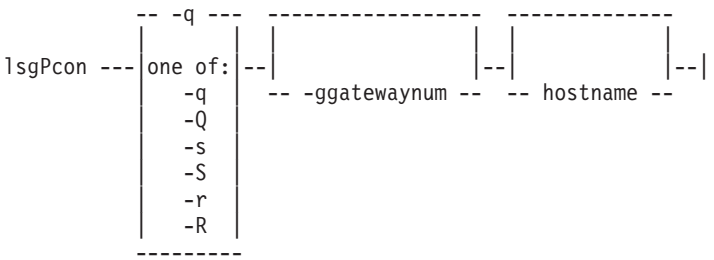
---

lsgPcon Command

Purpose

Inquire runtime connection profile information from a graPHIGS API gateway daemon.

Syntax



Description

This command inquires runtime connection profile information from a graPHIGS API gateway daemon running on the specified host.

If **lsgPcon** is successful, one or more of the following messages appear at the terminal:

```
devaddr  hostname:nucid  stateinfo
```

Where *devaddr* is the IBM S/390 device address used by the GDDM/graPHIGS API application, and *hostname:nucid* is the hostname and nucleus identifier of the host where the graPHIGS API gateway daemon is running. *stateinfo* may be one of the following:

- DEFINED

  There exists a runtime connection profile entry which defines the connection between the IBM S/390 device address and the graPHIGS API remote nucleus.

- OPEN

  The runtime connection profile entry is defined and there is an active connection between the IBM S/390 device address and the graPHIGS API remote nucleus.

If **lsgPcon** is unsuccessful, one of the following messages appear at the terminal:

```
AFM0604  NUCLEUS n1 NOT STARTED OR NOT RESPONDING
AFM0593  COMMUNICATION ERROR: MAJOR 7 MINOR 73
```

If you are having problems running the application, refer to Problem Determination for gPgated, the graPHIGS API Gateway Daemon.

Flags

hostname	Specifies the hostname of the gateway to which the inquiry will be sent.
-g:*gatewaynum*	Specifies the numeric suffix to be appended to the HIA device name that is opened and managed. The default is zero, for example, /dev/hia0.
-q	Requests the return of the status of those connections allocated to the requesters node at the specified gateway. This is the default option. This option is mutually exclusive with all other options.
-Q	Requests the return of the status of *all* connections at the specified gateway. The default is for the return of only those connections allocated to the requesters node at the specified gateway. This option is mutually exclusive with all other options.
-s	Requests the return of the status of those connections allocated to the requesters node at the host where the specified gateway daemon is running (as with the **-q** option). In addition, requests the return of the I/O statistics associated with those connections. This option is mutually exclusive with all other options.
-S	Requests the return of the status of *all* connections at the host where the specified gateway daemon is running (as with the **-Q** option). In addition, requests the return of the I/O statistics associated with those connections. This option is mutually exclusive with all other options.
-r	Requests the reset of the I/O statistics of those connections allocated to the requesters node at the host where the specified gateway daemon is running. This option is mutually exclusive with all other options.
-R	Requests the reset of I/O statistics of *all* connections at the host where the specified gateway daemon is running. This option is mutually exclusive with all other options.

Examples

1. To inquire, from the graPHIGS API gateway daemon running on the local host, the runtime connection profile information about the local host:

   ```
   lsgPcon
   ```

2. To inquire the runtime connection profile information for all connections from a gateway daemon running on the host, *gatenode*:

   ```
   lsgPcon -Q gatenode
   ```

3. To inquire, from the graPHIGS API gateway daemon running on the local host, the runtime connection profile information and statistics for the local host:

   ```
   lsgPcon -s
   ```

4. To reset the statistics for the gateway daemon running on the local host:

```
lsgPcon -r
```

5. To inquire the runtime connection profile information and statistics for *all* connections managed by a gateway daemon running on the host called *gatenode*:

```
lsgPcon -S gatenode
```

6. To inquire the runtime connection profile information and statistics for all connections to the gateway daemon that is managing the device **/dev/hia/** running on the a host called *gatenode*:

```
lsgPcon -S -g1 gatenode
```

7. To reset the statistics for *all* connections managed by a gateway daemon running on the host called *gatenode*:

```
lsgPcon -R gatenode
```

Files

/usr/bin/lsgPcon

Related Information

The **gPgated** command, **chgPcon** command.

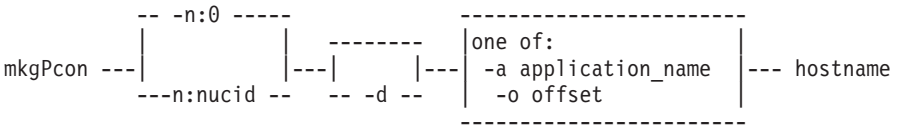
---

mkgPcon Command

Purpose

Make a connection to or break a connection from a 6098 with FDDI feature.

Syntax



Description

This command makes a connection to or breaks a connection from the 6098 with FDDI feature and provides verification and identification of the requested action.

Making a connection with the 6098 with FDDI feature causes an IBM S/390 device address to be allocated to the specified graPHIGS API remote nucleus. The nucleus must reside on the workstation where the command is issued and be running at the time the **mkgPcon** command is used. At the time the connection is made, the 6098 with FDDI feature is granted access to the local nucleus, so that the **gPhost** command need not be used.

This command may also be used to break a single connection. The **gPterm** command may be used to break all connections to the nucleus.

The IBM S/390 device address that is assigned when a connection is made is a function of:

1. The channel or channel path to which the 6098 is attached
2. The control unit offset that is configured at the 6098
3. The 6098 port selected

The first two are controlled by your system programmer and planner. The last is a function of the parameters used on the **mkgPcon** command and 6098 port availability. There is a one-to-one correlation between 6098 port and channel address. There are up to 256 ports (device addresses) available, and they are referred to by their *offset* from a configured base port. Thus, the value range of port offsets is 0 to 255 (or 0x0 to 0xff).

The 6098 provides two mechanisms for port assignments:

- Explicit port assignment - invoked by using the **-o** parameter
- Get next available port - invoked by using the **-a** parameter or no parameters

When you use the **mkgPcon** command, the *hostname* parameter determines the 6098 and thus the channel and control unit offset. To determine the S/390 device address, you must add the offset returned by the **mkgPcon** command to the data provided to you by your system programmer on the channel and control unit offset. There is a 6098 configuration parameter, known as the *channel path id*, which is a text string that may be set by your administrator to provide that information. It is recommended that the channel path id data be in the form:

```
CPUname channel_path other_useful_data <base_hex_address>
```

If it is, then **mkgPcon** parses *base_hex_address*, performs the addition, and returns the IBM S/390 device address desired. So the manual operation described above is only required if the administrator has not customized the 6098 in the prescribed manner.

The following is an example of input for making a connection:

```
mkgPcon -a'FPGP TESTING USAGE' w20
```

If **mkgPcon** is successful, the following message appears at the terminal:

```
mkgPcon: Attempting connection on port 8
```

Then one of the following messages appears:

EITHER

```
mkgPcon: Host 'w20' accepted connection on port 8
```

OR

```
mkgPcon: Host 'w20' accepted connection on device address 19A2
```

If **mkgPcon** is unsuccessful, one of the following messages appear at the terminal:

```
AFM0604  NUCLEUS n1 NOT STARTED OR NOT RESPONDING
AFM0593  COMMUNICATION ERROR: MAJOR 7 MINOR 73
AFM0640  UNKNOWN HOST 'w20'
AFM0641  HOST 'w20' NOT RESPONDING
AFM0642  HOST 'w20' IS NOT AN IBM 6098 W/FDDI FEATURE
AFM0643  HOST 'w20' HAS NO FREE 'FPGP TESTING USAGE' PORTS
AFM0644  HOST 'w20' DID NOT ACCEPT THE CONNECTION AS SPECIFIED
AFM0646  HOST 'w20' HAS NO CONNECTION AS SPECIFIED TO BE DELETED
AFM0645  RPC FUNCTION get base port FAILED FOR APPLICATION 'TEST123' ON PORT 36
AFM1201  SYSTEM SERVICE read ERROR RETURN CODE = 73
```

If you are having problems running the application, refer to Problem Determination for gPgated, the graPHIGS API Gateway Daemon.

Flags

-n:*nucid* Specifies the nucleus identifier of the graPHIGS API remote nucleus. The default is zero.

-d Deletes the connection between the specified remote nucleus and the IBM S/390 device address. A new connection for the IBM S/390 device address may then be added.

-a Defines the port number selection by *application name*. The application name is an arbitrary string of up to 31 characters. When using blank characters within an application name, enclose the application name in quotes, for example:

```
'FPGP TESTING USAGE'
```

-o Defines the port number selection by offset from base. This parameter may be entered in any of three forms: octal, hexidecimal, or decimal. Using the standard notation:

- A leading 0 indicates an octal number (e.g. 077)
- A leading 0x or 0X indicates a hexadecimal number (e.g. 0xA0)
- All other entries will be assumed to be decimal

hostname A required parameter that specifies the name or IP address of the 6098 with FDDI feature.

Examples

1. To request a connection from a 6098 with FDDI feature called *hostname* to the default nucleus (which has an identification number of zero):

   ```
   mkgPcon hostname
   ```

2. To request a connection from a 6098 with FDDI feature called *hostname* to the remote nucleus which has an identification number of two:

   ```
   mkgPcon -n:2 hostname
   ```

3. To request a connection on a specific GAM address from a 6098 with FDDI feature called *hostname* to the remote nucleus which has an identification number of two:

   ```
   mkgPcon -o0xfe -n:2 hostname
   ```

4. To request a connection from a 6098 with FDDI feature called *hostname* to the default nucleus (which has an identification number of 0) using the application name *testing123* as a port selector:

   ```
   mkgPcon -atesting123 hostname
   ```

5. To drop a connection between the default nucleus and the 6098 with FDDI feature called *hostname*:

   ```
   mkgPcon -d hostname
   ```

6. To drop a connection between the remote nucleus which has an identification number of two and the 6098 with FDDI feature called *hostname*:

   ```
   mkgPcon -d -n:2 hostname
   ```

7. To drop a connection on a specific IBM S/390 address between the default nucleus and a 6098 with FDDI feature called *hostname*:

   ```
   mkgPcon -d -o0xfe hostname
   ```

8. To drop a connection from a 6098 with FDDI feature called *hostname* to the default nucleus (which has an identification number of 0) using the application name *testing123* as a port selector:

   ```
   mkgPcon -d -atesting123 hostname
   ```

Files

/usr/bin/mkgPcon

Related Information

The **ls6098** command

The graPHIGS API Nucleus

Chapter 6. Enabling User Exits for Conferencing

The graPHIGS API provides a set of user exits through which your application display data can be distributed and managed by a conference utility. Conference utilities allow multiple users on separate workstations to participate in a single graPHIGS API application. This is very useful when a team of designers, for example, who work in different locations need to discuss changes to a model they are all using. New users of a graPHIGS API application may also benefit when getting assistance from the help desk through a conference utility.

The graPHIGS API can accommodate a conference utility in the following way. The graPHIGS API application runs on a master workstation while a conference utility controller communicates with the participating workstations. This conference controller must be given information about all participating workstations before the application is started. One way to supply this information is through a conference session manager connected to a user interface. The graPHIGS API supports message facilities that allow this information to flow between the controller and the session manager. The session manager, conference controller, and the application must all run on the same node as the master workstation. See the following figure:

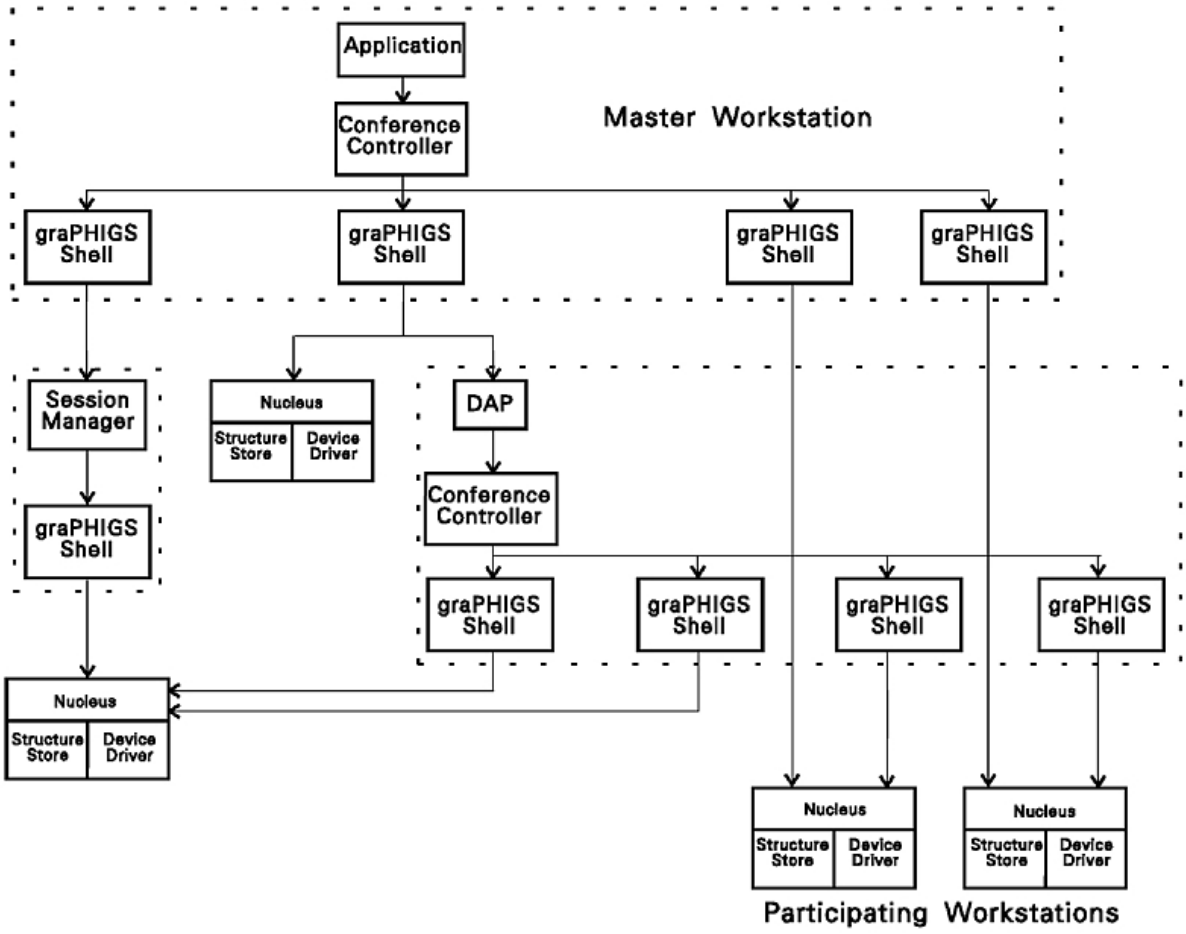


Figure 4. The graPHIGS API Running with a Typical Conference Utility. This diagram shows the master workstation, with its application, conference controller and session manager, communicating with several participating workstations.

As the application renders data to the master workstation, the conference controller renders the same data to each participating workstation, allowing all participants to view the graPHIGS API application. It can also allow control to be passed from the master workstation to any one of the participating workstations for interaction with the application.

In order to render data to the participating workstations, the conference controller must intercept each graPHIGS API function issued by the application. It must then determine which functions to issue to the participating workstations as well as to the master workstation, and which to simply pass along to the master workstation. Further decisions must be made regarding the function parameters: if the controller issues the function to the participating workstations, it may use the same parameters set by the application or it may modify them before issuing them.

The conference controller must define and allocate a separate application anchor block (AAB) for each participating workstation. In this way, the conference controller keeps track of which workstation session has input focus. The conference controller should not allow any session to remain in a state awaiting input, such as a response to a Request Choice (**GPRQCH**) subroutine. This prevents input focus from being switched among master and participating workstation sessions.

The following sequence of events summarizes the initialization and exchange of data that might be used for a graPHIGS API conference utility.

1. The user starts the conference session manager on the master workstation.
2. The user inputs information about participating workstations to the session manager.
3. The user starts the application on the master workstation.
4. The user starts the conference controller (see Starting and Stopping the Conference Utility Controller).
5. The conference controller issues the Inquire Nucleus Specification (**GPQNS**) subroutine to determine the session manager hostname.
6. The conference controller issues the Inquire Workstation Type and Options (**GPQWTO**) subroutine to determine the workstation type and options for the master workstation.
7. The conference controller uses the Create Workstation (**GPCRWS**) subroutine with NICKCHK to open participating workstations using the same type and options as the master workstation.
8. The conference controller might issue the Inquire Input Device State (**GPQID**) subroutine to determine input device state of a participating workstation.
9. Once it has determined the input device state for the participating workstation, the conference controller might then issue one or more of the following Inquire Device State subroutines to determine the state of an input device attached to a specified participating workstation.
   - Inquire Choice Device State (**GPQCH**)
   - Inquire Locator Device State (**GPQLC**)
   - Inquire Pick Device State (**GPQPK**)
   - Inquire Stroke Device State (**GPQSK**)
   - Inquire String Device State (**GPQST**)
   - Inquire Valuator Device State (**GPQVL**)
10. The conference controller issues the Set Input Device Mode (**GPIDMO**) subroutine to set the input device state in order to request input from a specific device without locking out another participating workstation from input focus.
11. The conference controller intercepts request input subroutine calls from the application. The conference controller manages input focus among participating workstations and ensures that a request to change input focus is honored.

Starting and Stopping the Conference Utility Controller

When the application issues the first graPHIGS API subroutine call, the graPHIGS API attempts to load a user exit facility routine when the environment is set up as follows:

For AIX:

- To allow for the conference controller utility to intercept graPHIGS API calls from a ksh or sh, the user sets the environment variable, **AFMEXIT** to *xxxx*, where *xxxx* is the fully qualified or relative pathname of the conference controller executable to be loaded. From a Csh, the user issues **set AFMEXIT *xxxx***, where *xxxx* is the fully qualified or relative pathname of the executable.
- To prevent the conference controller from being loaded in the shell, issue **unset AFMEXIT**.
- To display the current path, the user issues **echo $AFMEXIT**.

For MVS:

The graPHIGS API always attempts to load a module named **AFMEXIT**.

- If the application requires the exit loaded, then the **loadlib** containing **AFMEXIT** must be concatenated to **STEPLIB**.
- To prevent the conference controller from being loaded, ensure that **AFMEXIT** is not concatenated to **STEPLIB**.

For VM:

- Specify **GLOBALV SELECT GRAPHIGS SETP AFMEXIT *xxxxxx***, where *xxxxxx* is the name of the the executable to be loaded. It must be a member of a **loadlib**.
- Issue **GLOBALV SELECT GRAPHIGS PURGE** to remove the exit facility routine specification.
- Issue **GLOBALV SELECT GRAPHIGS LIST** to list any exit routines that are currently set.

The Conference Controller

The conference controller minimally includes the user exit routine and the application intercept exit routine.

The User Exit Routine

Once the user exit routine successfully loads, the address returned by the load service is invoked with the following parameters.

```
int exit_init (appl_anc, gPFuncListp, exit_anchor)
  int *appl_anc;
  void (*gPFuncListp)();
  int *exit_anchor;
{
}
```

appl_anc
: The graPHIGS API anchors the exit environment in this block. The user exit routines should not modify it. This parameter is provided to the application intercept exit routine for two purposes:
  - For use with *gPSetInterceptExit(C or F)* function
  - For use with error handlers when the application is using the reentrant or non-reentrant interfaces. The application's Application Anchor Block (AAB), not the user exit's internal AAB, must be passed to the application error handler.

gPFuncListp
: This parameter is a vector of function pointers. There are six function pointers available to the conference controller:

gPCallThru The entry point to the function that the application intercept exit routine calls to invoke the graPHIGS API subroutine.

gPSetInterceptExitC he entry point to the function used to set the application intercept exit routine address that receives control for the application's current and subsequent calls to the graPHIGS API. This function should be used if your conference controller is written in C.

gPSetInterceptExitF The entry point to the function used to set the application intercept exit routine address that receives control for the application's current and subsequent calls to the graPHIGS API. This function should be used if your conference controller is written in Fortran.

gPInvApplC The entry point to the conference controller function that must be used to invoke an application routine such as an error handler set by either the Define Error Handling (**GPEHND**) subroutine or the Specify and Error Exit and Error Threshold (**GPEXIT**) subroutine. This routine ensures that the application's environment is set up before it is invoked. This entry should be used if your conference controller is written in C.

gPInvApplF The entry point to the conference controller function that must be used to invoke an application routine such as an error handler set by either the Define Error Handling (**GPEHND**) subroutine or the Specify and Error Exit and Error Threshold (**GPEXIT**) subroutine. This routine ensures that the application's environment is set up before it is invoked. This entry should be used if your conference controller is written in Fortran.

gPSetPassthruAAB The entry point to the function to call prior to disabling the application intercept exit or prior to returning a return code of -1 from the application intercept exit. This function establishes the environment that the passthru application anchor block points to in order to replace the exit environment when it is terminated or disabled.

exit_anchor
A pointer to an 8-byte area that the application intercept exit routine uses to anchor its dynamic storage. This pointer is passed to the application intercept exit routine on each call.

exit_init is invoked immediately after the exit is loaded. It should allocate storage and initialize but make no calls to the graPHIGS API, except to *gPSetInterceptExit(C or F)* or *gPSetPassthruAAB*.

To process the application API call that caused it to be loaded, the user exit routine must make a call to *gPSetInterceptExit(C or F)* to enable an application intercept exit routine before returning. The graPHIGS API then invokes the specified intercept exit routine prior to returning to the application and on subsequent graPHIGS API subroutine calls.

If *exit_init* returns non-zero, then the graPHIGS API assumes that the initialization failed. It unloads the exit and proceeds with normal graPHIGS API initialization. If *exit_init* returns zero, then it loads the application intercept exit routine and processes the API subroutine that caused it to be loaded.

exit_init is also called at termination time. The parameter list is identical for both calls. This makes cleanup symmetrical to initialization.

Due to the possible recursion when error handlers are called, application intercept exit routines must be recursive (no writable statics). For this reason, Fortran is not recommended for a conference controller, although it may work when the application is written in Fortran as well. The only disadvantage to writing all conference controllers in the C programming language is that they require the runtime C library.

An example of how typical declarations might be specified for the initial entry:

```
/*-----------------------------------------------------------------*/
/* Overlay for function vector (2nd parm on init call).            */
/*-----------------------------------------------------------------*/
struct gPfuncv {
  void (*gPCallThru)();                 /* gPCallThru ptr          */
  void (*gPSetInterceptExitC)();        /* gPSetExit for C exits   */
  void (*gPSetInterceptExitF)();        /* gPSetExit for F exits   */
```

```
  void (*gPInvApplC)();                 /* gPInvAppl for C exits   */
  void (*gPInvApplF)();                 /* gPInvAppl for F exits   */
  void (*gPSetPassthruAAB)();           /* gPSetPassthruAAB ptr    */
};

/*-------------------------------------------------------------------*/
/* Main routine for exit.  This routine is invoked when the exit is */
/* first loaded.  It is invoked only once.                           */
/*-------------------------------------------------------------------*/
int exit_init(appl_anc, funclistp, anchor)
  int *appl_anc;                        /* Env anchor (Read/Only)  */
  struct gPfuncv *funclistp;            /* Ptr to function vectors */
  int *anchor;                          /* Anchor for exit storage */
{
}
```

The Application Intercept Exit Routine

Once the user exit routine has control, it may enable an application intercept exit routine to be given control on the current application request and on subsequent API requests by issuing the *gPSetInterceptExit(C or F)* function with the address of the intercept exit routine.

If the *gPSetInterceptExit(C or F)* function is issued with the address of zero in the *funcp* parameter, then the application intercept exit routine is disabled from being called on each API request. This establishes the environment for the passthru AAB to point to when it replaces the the disabled exit environment. Before disabling an intercept exit, the application must invoke the *gPSetPassthruAAB* function routine.

A return code of -1 from the application intercept exit routine means that the conference controller will be terminated and replaced by the environment pointed to by the passthru AAB. This allows an exit and its overhead to be removed without terminating an active graPHIGS API application.

For example, for C:

```
void gPSetInterceptExitC (appl_anc, funcp)
  int *appl_anc;
  int (*funcp)();
{
}
```

or, for Fortran:

```
void gPSetInterceptExitF (appl_anc, funcp)
  int *appl_anc;
  int (*funcp)();
{
}
```

appl_anc — A pointer to the passthru AAB. It must *not* be modified by the application intercept exit routine. It is the same as the first parameter on the *exit_init* call.

funcp — The function pointer that identifies which entry point is to serve as the application intercept exit routine. If *funcp* is null, then the graPHIGS API works in a passthru mode such that all calls are passed from the graPHIGS API using the current passthru AAB.

Invoking the Application Intercept Exit Routine

Any application intercept exit routine that is enabled by the user exit routine through *gPSetInterceptExit* in response to an API request by the application is invoked as follows:

```
int intercept_exit(exit_anchor,rcpp,plistp,scbp)
  int *exit_anchor;
  int *rcpp;
```

```
  int **plistp;
  struct scb *scbp;
{
}
```

exit_anchor A pointer to the 8-byte area that was passed to the *exit_init* that is used to anchor any storage required by the intercept exit routine.

rcpp A pointer to the rcp code of the current graPHIGS API subroutine.

plistp A pointer to the application's parameter list. For the generic binding and the ISO PHIGS Fortran binding, this is a list of pointers, one for each parameter. For the ISO PHIGS C binding this can be pointers or values, depending on the API call itself. Some parameters are passed by value for the C binding.

scbp A pointer to a stub communication block that describes some aspects of the API call. The format for *scbp* is:

```
struct scb {
  char          ssid;                   /* Sub-system id               */
                                        /* 'T' = TSO (MVS),            */
                                        /* 'V' = VM, 'X' = AIX */
  unsigned char iftype;                 /* 0=non-ren, 1=ren, 2=SPI     */
  unsigned char binding;                /* 0=generic/Fortran, 2=ANSI C */
  unsigned char rsvd[1[default]
  unsigned char rsvd2;
  void (*erhp)();                       /* ANSI default Exit           */
}
```

Application Intercept Exit Call Through to the graPHIGS API

The application intercept exit routine calls through to the graPHIGS API to process the API request as follows:

```
void gPCallThru(aab,rcpp,plistp,scbp)
int *aab
unsigned int *rcpp
int **plistp
struct scb *scbp
{
}
```

aab Defined by the graPHIGS API for the reentrant and SPI interfaces. It is an 8-byte area that must be initialized to zero before it is first used to invoke the graPHIGS API.

rcpp A pointer to the rcp code of the current graPHIGS API subroutine.

plistp A pointer to the application's parameter list. For the generic binding and the ISO PHIGS Fortran binding this is a list of pointers, one for each parameter. For the ISO PHIGS C binding this is pointers or values, depending on the API call itself. Some parameters are passed by value for the C binding.

scbp A pointer to a stub communication block that describes some aspects of the API call. The format for *scbp* is shown above.

Preparing to Disable the Intercept Exit Routine

Before the application intercept exit routine stops processing the API requests, it issues the following:

```
void gPSetPassthruAAB(appl_anc,aabp)
int *appl_anc
int *aabp
{
}
```

appl_anc A pointer to the passthru AAB. It must not be modified by the application intercept exit routine. It is the same as the first parameter on the init call.

aabp A pointer to an area defined by the application intercept exit routine that is initialized during invocation of the graPHIGS API. This function is used before the intercept is set to be disabled and before -1 is returned from the intercept to establish the AAB that is to be used for subsequent API calls. If the passthru AAB is not set to the AAB used to invoke the graPHIGS API, then a default AAB is used. This results in errors indicating that the graPHIGS API is not open because it will not yet have an AAB. The only current restriction is that passthru mode cannot be mixed with the exit that set error handlers in VM/MVS. If this restriction is not observed, the error handler may be invoked with the incorrect programming language environment, resulting in an abend. This restriction does not apply to AIX.

Passing Error Handler Calls from the graPHIGS API to the Application

The *gPInvAppl(C or F)* routines must be used by the application intercept exit routine when an error handler call from the graPHIGS API is intercepted and is to be passed on to the actual application error handler. *gPCallThru* does not apply in this case because the application, not the graPHIGS API, is being called.

For example, for C:

```
void gPInvApplC(appl_anc,routine_addr,parms)
```

or for Fortran:

```
void gPInvApplF(appl_anc,routine_addr,parms)
int *appl_anc
void(*routine_addr)()
parms
{
}
```

Where

appl_anc A pointer to the passthru AAB. It must not be modified by the application intercept exit routine. It is the same as the first parameter on the init call.

routine_addr A pointer to the application's error handler routine.

parms Any other parameters expected by the receiving error handler routine.

These special entry points are necessary because of the great differences in runtime environments among C, Fortran, Pascal, and other programming languages. In order to ensure that a Fortran application with a Fortran error handler runs correctly, graPHIGS API must invoke the error handler with the same register content as when the application last called the graPHIGS API. this process, the graPHIGS API can no longer act directly since the call that generates the error comes from the exit code and not from the application. Exit routines written in a language different from the application must define their own error handlers and pass them on to those of the application using the *gPInvAppl(C or F)* routine.

The general flow for a conference utility exit routine is shown in the following figure. (A) and (B) show the points at which the runtime environment must be the same (that is, the register contents must be very similar).

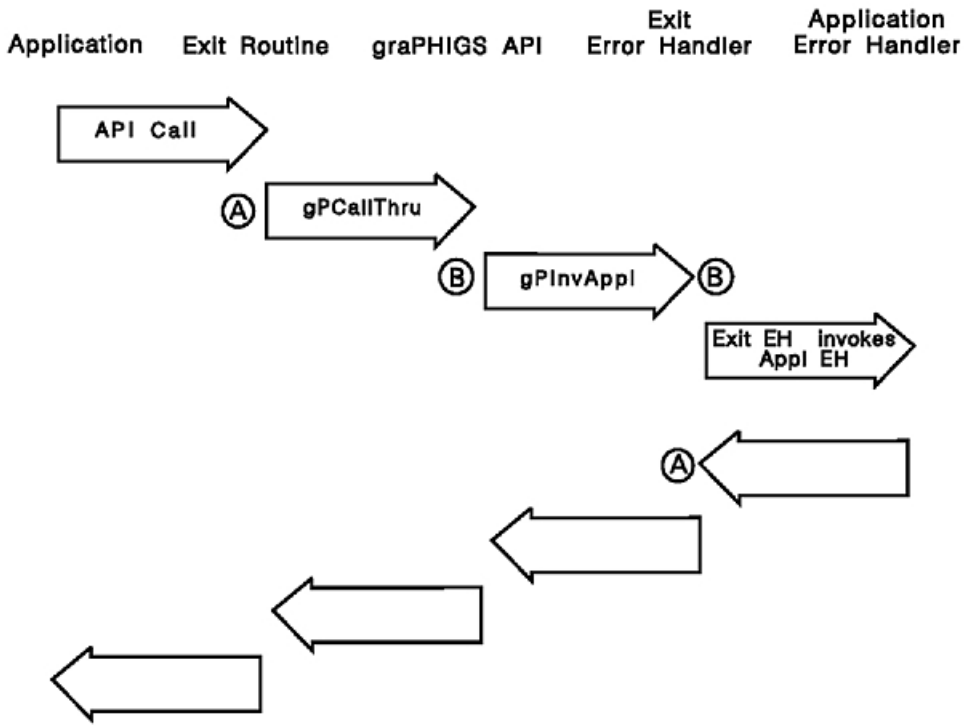


Figure 5. User Exit Routine Flow. This diagram shows the general flow for a conference utility exit routine. The application uses an API call to invoke the exit routine (A), which uses gPCallThru to invoke the graPHIGS API (B). The graPHIGS API (B) uses gPInvAppl to invoke the Exit Error Handler (B), which directly invokes the Application Error Handler. The Application Error Handler returns to the Error Handler (A), which returns to the graPHIGS API, and so on back to the top level. Points "A" denote that the program state must be the same before calling the gPCallThru routine and after returning from the Application Error Handler. Points "B" denote that program state must be equivalent before and after a call to the gPInvAppl routine.

Part 3. Defaults and Nicknames

Chapter 7. Controlling the Environment with Defaults and Nicknames

This chapter provides information about defaults and nicknames. It also provides information about processing options (PROCOPTs). Use this information to modify aspects of your environment to suit the needs of your applications.

Overview of Controlling the Environment

In the graPHIGS API system and workstation environment, certain fields (sizes, names, options, etc.) are given preset values by the graPHIGS API. In some instances, these preset values may not suit the needs of the application. The graPHIGS API allows the programmer or end user to change these preset values through defaults and nicknames. Defaults are intended to modify the system environment values, while nicknames are intended to modify values associated with a particular workstation environment.

Changes can be made in two ways:

- The application programmer and subsequently, the end user, may specify defaults and/or nicknames through an External Defaults File (EDF). The application program does not need to be recompiled or rebuilt.

  In addition, you can specify the workstation processing options (PROCOPTS) by using the PROCOPTS option (see PROCOPT (Processing Options)) on the Create Workstation (**GPCRWS**) subroutine call.
- The application programmer may specify defaults and/or nicknames through a control block specified as a parameter on the Open graPHIGS (**GPOPPH**) subroutine call in the application program. This control block is called the Application Defaults Interface Block (ADIB). If defaults or nicknames are changed, the application must be re-compiled and rebuilt.

Defaults allow you to change the system environment. You can control the trace state, direct trace output, adjust the size of the in-core trace table, set input and output buffer and queue sizes, define nucleus connection processing, and inhibit shell syntax checking.

If the same nickname or default value is specified in both an ADIB and an EDF, the EDF value will be ignored and the ADIB value will be used. Because the ADIB has higher priority than the EDF, the application programmer can control the user's environment. The only exception to this rule is the trace control word— the trace control word itself determines priorities. (Trace priorities are discussed in *The graPHIGS Programming Interface: Writing Applications*.)

Processing of the defaults and nicknames specified through an EDF and/or an ADIB occurs during Open graPHIGS processing.

Nicknames allow you to change workstation environment values. The nickname values associated with a workstation environment are referred to as processing options or PROCOPTS. Nicknames also allow you to change the workstation type *(wstype)* and connection identifier *(connid)* parameters specified on the Open Workstation (**GPOPWS**) and Create Workstation (**GPCRWS**) subroutines.

If the same nickname is specified in both an ADIB and an EDF, the API resolves the nickname using first the EDF values followed by the ADIB values. Any PROCOPTs included on the selected nicknames are merged with any PROCOPTs specified on the **GPCRWS** subroutine to produce a final PROCOPT list. See How the graPHIGS API Processes Nicknames for a discussion of this processing.

The remainder of this chapter explains how defaults and nicknames are processed, the format of an EDF specification, the format of an ADIB specification, and the explanation and syntax of each default, nickname, and PROCOPT.

The following table compares the EDF and ADIB.

Table 104. Comparison of EDF and ADIB

Name	Description	Limitations	Priority
External Defaults File (EDF)	A file for users to modify their environment	Not valid for applications running locally on a 6090 in a DAP environment.	Each default or nickname value is in effect only if no ADIB value exists.
Application Defaults Interface Block (ADIB)	A control block area passed as a parameter by the application program	To change defaults, you must re-compile or re-build your program	Overrides EDF (except for Trace setting in the EDF)

The External Defaults File (EDF)

The External Defaults File contains records which consist of User-Defined Specifications (UDSs). The UDSs in this file allow you to change user default options at run-time without re-compiling or re-building your application. The API accesses the External Defaults File as follows:

AIX The file must be named **PROFILE** or must be specified in the **gPPROFILE** environment variable. When the **GPOPPH** subroutine is called or when a remote graPHIGS API nucleus is started, the graPHIGS API searches for a **PROFILE** in this order:

1. **gPPROFILE** environmental variable

   The **gPPROFILE** environment variable allows you to specify an alternate filename or an alternate directory path containing the file, **PROFILE**, as the external defaults file.

   If the **gPPROFILE** environmental variable is defined as a valid file name, then that file is used as the External Defaults File. If the **gPPROFILE** environmental variable is defined as a valid directory name, then that directory is searched for a file named **PROFILE**. If this file is found, then it is used as the External Defaults File.

   If the **gPPROFILE** environmental variable is not defined, is defined with an invalid file name or directory name, or there is no file named **PROFILE** in the defined valid directory name, the search continues.

   For more information on setting environment variables, see the *AIX 5L Version 5.3 Commands Reference*.

2. Current directory

   The current directory is searched for a file named **PROFILE**. If there is no file named **PROFILE** in the current directory, the search continues.

3. **/usr/lpp/graPHIGS/etc** directory

   The graPHIGS API provides a sample External Defaults File as **/usr/lpp/graPHIGS/etc/PROFILE**.

MVS `AFMDEFS` must be the DDNAME used to allocate the sequential data set containing the default information. The file must be F- or V- format, with an LRECL of no greater than 256. The recommended format is F(80).

VM The file must be named **PROFILE**, have a filetype of `AFMDEFS`, and be on a currently-accessed disk when the application calls Open graPHIGS (**GPOPPH**).

Format of the User-Defined Specification (UDS)

A UDS is a string that can be up to 32,000 characters long. Use one of the following forms for a record in the graPHIGS API External Defaults File:

Table 105. UDS Format

[label]	UDS-type	UDS-value	[OPTIONAL COMMENTS]
[label]	UDS-type	UDS-value-part1,	[OPTIONAL COMMENTS]
		UDS-value-part2,	[OPTIONAL COMMENTS]

Table 105. UDS Format (continued)

		UDS-value-part3,	[OPTIONAL COMMENTS]
		UDS-value-partn,	[OPTIONAL COMMENTS]
* COMMENT TEXT			

The UDS-type parameter is one of the following:

AFMMDFT or DEFAULT	Default value
AFMMNICK or NICKNAME	Nickname value

The UDS value is specified as `keyword=specified_value`. If you specify a keyword with nothing after the equal sign (=), the current value is not changed.

Valid default and nickname keywords and values are given with a description of each default or nickname in the following sections.

Records in the External Defaults File must conform to Assembler-like coding conventions. When specifying a UDS, you must observe the following conventions:

- Labels are optional. If specified, they must start in column one and must not be longer than eight characters. They are ignored.
- The *UDS-type* must be preceded by at least one blank.
- The *UDS-type* and *UDS-value* parameters must be separated by at least one blank.
- In a *UDS-value* parameter, a comma (,) followed by a blank or end-of-record marker indicates that the *UDS-value* is continued on the *next* non-comment record. The continuation must be *preceded* by at least one blank. Any text that starts in column one is assumed to be part of a label.
- The *UDS-value* parameter must not contain any embedded blanks.
- The API assumes that any text following a blank after a *UDS-value* parameter is comment text, and ignores it.
- There is no limit on the number of continuation records permitted.
- If you specify comments, they must have an asterisk (*) in column one. The API ignores comment records in all circumstances.
- You can use mixed case to enter a UDS. In the S/390 environment, all lowercase characters are converted to uppercase before processing. In the AIX environment, all UDS types and keywords are converted to uppercase, but values are not converted.

The Application Defaults Interface Block (ADIB)

The Application Defaults Interface Block (ADIB) is the second method for the application programmer to specify defaults and nicknames. The ADIB takes priority over any defaults or nicknames specified also in an External Defaults File (EDF), except possibly the Trace default.

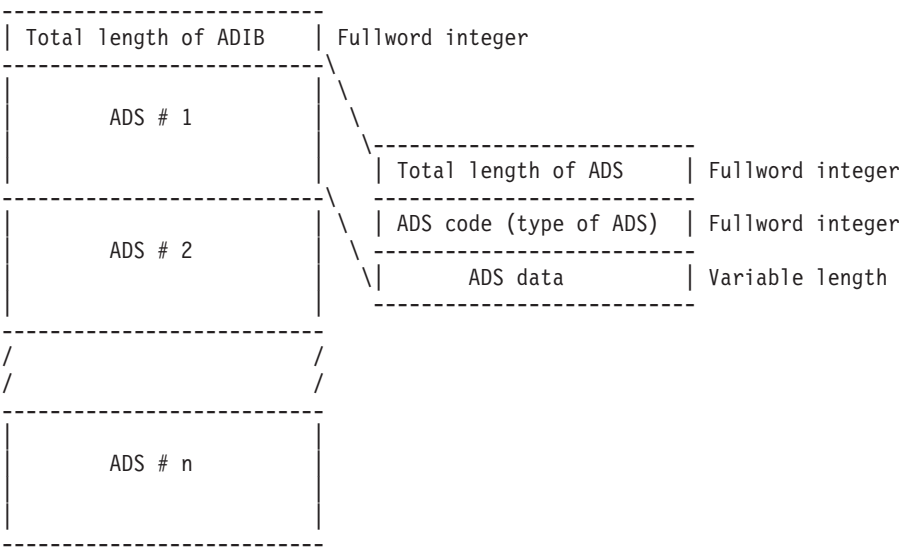
The application must specify the ADIB as the second parameter on the Open graPHIGS (**GPOPPH**) subroutine. If you don't specify any ADIB options, set the **GPOPPH** parameter to zero.

Format of the ADIB

The ADIB consists of a fullword integer specifying the ADIB length, followed by any number of Application Default Specifications (ADS). When you specify the value of the length fields in bytes, include the length fields themselves.

This is the format of an ADIB with its ADS:

Figure 8. Format of ADIB and ADS



The ADS code identifies the type of ADS. The ADS data is a variable length field and is dependent on the type of ADS.

Defaults

Defaults allow the application programmer or end user to modify the preset values of the graPHIGS API system environment. Following are the default descriptions and their syntax as set through an EDF or an ADIB.

AIXTRCE (AIX Trace Output)

This default directs trace output from the API on the operating system. It is specified as a string of up to 50 characters that indicates the file path, followed by two 8-character strings that specify a filename and file extension.

Initial Preset Value

The filename **AFMTRACE** within the current working path.

EDF

To specify this default through an EDF, the correct syntax is:

```
AFMMDFT AIXTRCE=(aaa...a,bbbbbbbb,cccccccc)
```

aaa...a is the file path, *bbbbbbbb* is the filename, and *cccccccc* is the file extension for trace output.

If you specify no values between commas, for example: `AFMMDFT AIXTRCE=,,` the values are set to nulls (file path='', file name='', file extension='').

ADIB

To specify this default through an ADIB, the correct structure is:

Total length of ADS (greater than 8, but a fullword multiple)	Fullword integer
ADS code = 603	Fullword integer
Fully-qualified filename	Variable-length character string

ARCHIVE (File Descriptors)

This default directs the output of an Open Archive File (**GPOPAR**) subroutine. A *From* file descriptor and a *To* file descriptor are specified.

EDF

To specify this default through an EDF, the correct syntax is:

```
AFMMDFT ARCHIVE=(aaa...a,bbb...b,)
```

aaa...a is the *From* archive file descriptor used in the Open Archive File (**GPOPAR**) subroutine, *bbb...b* is the *To* archive file descriptor to be used instead. Specify both file descriptors with no blanks, but include a comma before closing the parenthesis. Refer to Archiving Structures for more information on structure archive.

ADIB

To specify this default through an ADIB, the correct structure is:

```
-----------------------------------------------
| Total length of ADS (application dependent) | Fullword integer
-----------------------------------------------
| ADS code = 117                              | Fullword integer
-----------------------------------------------
| Reserved = 0                                | Fullword integer
-----------------------------------------------
| Length of From descriptor                   | Fullword integer
-----------------------------------------------
| From descriptor                             | *Variable-length character string
-----------------------------------------------
| Length of To descriptor                     | Fullword integer
-----------------------------------------------
| To descriptor                               | *Variable-length character string
-----------------------------------------------

*(padded to word boundary)
```

CMSTRCE (CMS Trace Output)

This default directs trace output from the API on the VM/CMS system. Two 8-character strings indicate the filename and filetype used by the graPHIGS API.

Initial Preset Value

A filename of **AFM00001** and a filetype of AFMTRACE.

EDF

To specify this default through an EDF, the correct syntax is:

```
AFMMDFT CMSTRCE=(aaaaaaaa,bbbbbbbb)
```

aaaaaaaa is the filename, *bbbbbbbb* is the filetype for receiving trace output in the CMS environment. The statement `AFMMDFT CMSTRCE=,` sends trace output to the printer.

ADIB

To specify this default through an ADIB, the correct structure is:

```
-------------------------------------
| Total length of ADS = 24          | Fullword integer
-------------------------------------
| ADS code = 502                    | Fullword integer
-------------------------------------
| Filename                          | 8-byte character string
| - - - - - - - - - - - - - - - - - |
| Filetype                          | 8-byte character string
-------------------------------------
```

COMBSZ (Input and Output Buffer Sizes)

This default sets the input and output buffer sizes that are to be used for all nucleus connections.

Initial Preset Value

65516 bytes.

EDF

To specify this default through an EDF, the correct syntax is:

```
AFMMDFT COMBSZ=(n[default],n[default])
```

where *n[default]* is the size of the input buffer that the shell uses to receive responses and events from the nucleus. *n[default]* is the size of the output buffer that the shell uses to send requests to the nucleus.

The size of each of the input and output buffers may have a maximum of 65516 bytes and a minimum of 4K bytes.

ADIB

To specify this default through an ADIB, the correct structure is:

```
-------------------------------------
| Total length of ADS = 16          | Fullword integer
-------------------------------------
| ADS code = 106                    | Fullword integer
-------------------------------------
| Size of input buffer (in bytes)   | Fullword integer
| - - - - - - - - - - - - - - - - - |
| Size of output buffer (in bytes)  | Fullword integer
-------------------------------------
```

COMMENT (Programming Comments)

This default lets you annotate the EDF with comments. A comment can be a list of strings of 8 or less non-blank characters. Your application can specify up to 8,000 comment strings. The API ignores comments during default processing. You can specify comments *only* in the External Defaults File.

EDF

To specify this default through an EDF, the correct syntax is:

```
AFMMDFT COMMENT=(cccccccc,cccccccc,........)
```

ADIB

Not valid.

DAPPATH (DAP Download File Path)

This default allows you to specify the file path for temporary storage of downloaded DAP files. This default must be specified as an EDF on the operating system, where the target remote nucleus resides.

Initial Preset Value

/tmp/.gP

EDF

To specify this default through an EDF, the correct syntax is:

```
AFMMDFT DAPPATH=ccccc
```

where *ccccc* is a file path up to fifty characters in length. This file path is used as the temporary directory for the storage of DAP files on the current operating system remote nucleus. The file path is used for the transfer and execute function of the Execute Application Process (**GPEXAP**) subroutine.

ADIB
Not valid.

DEFACTF (Activate Font Handling)
This default allows you to select the way the graPHIGS API handles activate font requests.

Initial Preset Value
yes

EDF
To specify this default through an EDF, the correct syntax is:

```
AFMMDFT DEFACTF=yes|no
```

If DEFACTF=yes, and an activate font request to a workstation fails because the character set/font pair cannot be found on the nucleus disk system, the graPHIGS API searches its disk for the character/font. If it finds the font definition, it will send it to the nucleus for activation to the workstation.

If DEFACTF=no, and an activate font request to a workstation fails, the graPHIGS API simply posts the error to the application.

ADIB
To specify this default through an ADIB, the correct structure is:

```
-------------------------------------
| Total length of ADS = 12          | Fullword integer
-------------------------------------
| ADS code = 109                    | Fullword integer
-------------------------------------
| Request font handling             | Fullword integer
| 0 = no, 1 = yes                   |
-------------------------------------
```

DEFNUC (Define Nucleus Connection Processing)
This default defines whether your application will explicitly issue the Connect to Nucleus subroutine (**GPCNC**) or graPHIGS API will do nucleus connection processing for your application. If this default is *not* specified and your application is *not* a distributed application process (DAP), the graPHIGS API will connect to a nucleus with identifier=1 on behalf of your application, using the CALL connection method. (It will additionally create a structure store using identifier=1 and select that structure store for editing.) If this default is *not* specified and your application is a distributed application process (DAP), the graPHIGS API will not connect to a nucleus. Your application must explicitly issue the Connect to Nucleus (**GPCNC**) subroutine. For an explanation of the connection processing, and for the valid data values, see Connecting to the Nucleus. See also NUC/TONUC (Nucleus Respecification) for information on changing the connection method and specification using defaults.

Initial Preset Value
Connection method = CALL, connection specification = NULL. (For DAPs, the initial preset value is 0, which means nucleus connection processing is suppressed and your application will explicitly issue the Connect to Nucleus (**GPCNC**) subroutine.)

EDF
- To have the graPHIGS API do nucleus connection processing on behalf of your application, specify the connection method and specification as:

  ```
  AFMMDFT DEFNUC=(n,ccc ... ccc)
  ```

  where *n* is an integer that specifies one of the connection methods supported by your nucleus and *ccc ... ccc* is a variable-length character string that specifies a nucleus connection specification in support of the method selected.

Note: As part of doing nucleus connection processing, the graPHIGS API will additionally create a structure store (**GPCRSS**) using identifier=1 and select that structure store (**GPSSS**) for editing.

- To have your application explicitly issue the Connect to Nucleus (**GPCNC**) subroutine (suppress graPHIGS API from doing nucleus connection processing), specify the UDS as follows:

```
AFMMDFT DEFNUC=(0,)
```

ADIB

To specify this default through an ADIB, the correct structure is:

Total length of ADS (is application-dependent)	Fullword integer (must be a multiple of 4)
ADS code = 104	Fullword integer
Connection method	Fullword integer
Length of nucleus connection specification	Fullword integer
Nucleus connection specification	Variable-length character string

Note: Specifying a connection method =0, a specification length =0, and an ADS length =16 in the ADIB will suppress graPHIGS API from doing nucleus connection processing.

ERREVENT (Enable Error Event)

You use the ERREVENT default to enable the queuing of errors on the graPHIGS event-queue. When enabled, errors are queued as events of class 401 with a major code equal to the error's number and a minor code of 0. No additional data is queued with the event.

Other than the queuing of an error as an error-event, the error-handling mechanisms of the graPHIGS API are unaffected by the value of the ERREVENT default. See Error Handling for an explanation of error-handling within the graPHIGS API.

The only errors that cannot be queued on the graPHIGS event-queue are errors related to the state of the graPHIGS API. For example, error 5 ″FUNCTION REQUIRES STATE STOP OR NROP (NOT STCL)″ is not to be queued on the graPHIGS event queue even if you enable error-event queuing via the ERREVENT default.

In combination with a graPHIGS event-handler, enabling error-events enables the graPHIGS API to inform your application of an error condition without the need for your application to call a graPHIGS API function.

Initial Preset Value

no

EDF

- To specify this default through an EDF, the correct syntax is:

```
AFMMDFT ERREVENT=yes|no
```

- If ERREVENT=no (the default), errors detected by the graPHIGS API Are queued on the graPHIGS error-queue only (not the graPHIGS event-queue).
- If ERREVENT=yes, all errors detected by the graPHIGS API are queued on the graPHIGS event-queue as events of class 401 with a major-code equal to the error's number, a minor-code of zero (0), and no event-data. All errors are also queued on the graPHIGS error-queue.
- For both ERREVENT=no and ERREVENT=yes, errors queued on the graPHIGS error-queue are processed by the next call to the graPHIGS API.

ADIB

To specify this default through an ADIB, the correct structure is:

```
-------------------------------------
| Total length of ADS = 12          | Fullword integer
-------------------------------------
| ADS code = 119                    | Fullword integer
-------------------------------------
| Request queuing of error-events   | Fullword integer
| 0 = no, 1 = yes                   |
-------------------------------------
```

HCHECK (Shell Syntax Checking)

This default allows you to specify if the shell will check the syntax of its input parameters.

Initial Preset Value

yes (the shell does syntax checking).

EDF

To specify this default through an EDF, the correct syntax is:

```
AFMMDFT HCHECK=yes|no
```

ADIB

To specify this default through an ADIB, the correct structure is:

```
-------------------------------------
| Total length of ADS = 12          | Fullword integer
-------------------------------------
| ADS code = 105                    | Fullword integer
-------------------------------------
| Syntax checking                   | Fullword integer
| 0 = no, 1 = yes                   |
-------------------------------------
```

IQSIZE (Input Queue Size)

With this default you can change the size of the allocated input queue.

Initial Preset Value

16K bytes.

EDF

To specify this default through an EDF, the correct syntax is:

```
AFMMDFT IQSIZE=n
```

where *n* is the queue size, the number of bytes of storage to allocate for the input queue (minimum of 4K bytes).

Note: If storage is not available for the size you request, the graPHIGS API will not open.

ADIB

To specify this default through an ADIB, the correct structure is:

```
-------------------------------------
| Total length of ADS = 12          | Fullword integer
-------------------------------------
| ADS code = 107                    | Fullword integer
-------------------------------------
| Number of bytes for the queue     | Fullword integer
-------------------------------------
```

MAXWKS (Maximum Workstation Support)

This default specifies the maximum number of workstations that may be opened and associated to the same structure store.

Initial Preset Value

Four (4).

EDF

To specify this default through an EDF, the correct syntax is:

```
AFMMDFT MAXWKS=n
```

where *n* is the number of workstations to be opened and associated. *n* may be any value from 1 to 32.

ADIB

To specify this default through an ADIB, the correct structure is:

```
-------------------------------------
| Total length of ADS = 12          |
-------------------------------------
| ADS code = 113                    |
-------------------------------------
| Number of workstations            |
-------------------------------------
```

NICKCHK (Nickname Processing Default)

This default allows you to specify if the graPHIGS API will do nickname processing for workstations opened by your application.

Initial Preset Value

yes

EDF

To specify this default through an EDF, the correct syntax is:

```
AFMMDFT NICKCHK=yes|no
```

If NICKCHK=*yes*, then the graPHIGS API library performs nickname processing to resolve the workstation type, connection identifier, and options using values specified in the EDF, ADIB, and on the Create Workstation (**GPCRWS**) or the Open Workstation (**GPOPWS**) subroutine. It sends the result of that processing to the nucleus when requesting a create workstation. This is the default.

If NICKCHK=*no*, then the graPHIGS API library does not perform nickname processing to resolve the workstation type, connection identifier, and options. It sends the workstation type, connection identifier, and options as explicitly specified in the Create Workstation (**GPCRWS**) or Open Workstation (**GPOPWS**) subroutine to the nucleus to request a create workstation.

ADIB

To specify this default through an ADIB, the correct structure is:

```
-------------------------------------
| Total length of ADS = 12          | Fullword integer
-------------------------------------
| ADS code = 116                    | Fullword integer
-------------------------------------
| Nickname processing               | Fullword integer
| 0 = no, 1 = yes                   |
-------------------------------------
```

NUC/TONUC (Nucleus Respecification)

This default allows you to change the nucleus connection processing values (the FROM connection method and connection specification) to the specified replacement values (TO). This default is used whenever a nucleus is connected (either explicitly using the **GPCNC** subroutine or implicitly by using the default nucleus connection processing).

If several NUC/TONUC defaults are specified (for example, in the ADIB and in the EDF), the list of them is searched starting with the ADIB defaults, followed by the EDF defaults. The lists are searched until a matching default is found. A match occurs when the input connection method and connection specification (specified on the **GPCNC** subroutine or from the nucleus connection processing) are both the same as the FROM connection method and connection specification in the default (i.e., the NUC values). When matched, the TO connection method and connection specification values (the TONUC values) replace the input values and are used to complete the nucleus connection processing.

See Connecting to the Nucleus for a description of the supported connection methods and connection specifications.

EDF

To specify this default through an EDF, the correct syntax is:

```
AFMMDFT NUC=(cml, csl),TONUC=(cm2,cs2)
```

where *cm1* and *cs1* are the nucleus connection method and connection specification to be matched and *cm2* and *cs2* are the replacement connection method and connection specification values.

The NUC and TONUC values must be on the same default specification. If the NUC value is omitted, the default will match any input connection method and specification.

ADIB

To specify this default through an ADIB, the correct structure is:

Total length of ADS (greater than 8, but a fullword multiple)	Fullword integer
ADS code = 110	Fullword integer
FROM connection method	Fullword integer
TO connection method	Fullword integer
FROM specification length	Fullword integer
FROM specification	(padded to word boundary)
TO specification length	Fullword integer
TO specification	(padded to word boundary)

Specify the value of zero in the FROM connection method and FROM specification length fields in order for the ADS default to match any input method and specification.

SYNCPROC (Synchronous X Event Processing)

This default tells the graPHIGS API that the application will be monitoring X events for the graPHIGS X workstations and will notify the graPHIGS API when an event occurs.

Initial Preset Value

no (the graPHIGS API will check for X events)

EDF
To specify this default through an EDF, the correct syntax is:

```
AFMMDFT SYNCPROC=yes|no
```

ADIB
To specify this default through an ADIB, the correct structure is:

```
-------------------------------------
| Total length of ADS = 12          | Fullword integer
-------------------------------------
| ADS code = 120                    | Fullword integer
-------------------------------------
| SYNCPROC active                   | Fullword integer
| 0 = no, 1 = yes                   | Fullword integer
-------------------------------------
```

TRACE (Trace Control Word)
This default indicates the state of the Trace Control Word (whether it allows tracing or not).

The trace word itself determines priority.

Initial Preset Value
0 (off).

EDF
To specify this default through an EDF, the correct syntax is:

```
AFMMDFT TRACE=n
```

where *n* is the value of the Trace Control Word. See *The graPHIGS Programming Interface: Writing Applications* or *The GDDM/graPHIGS Programming Interface: Installation and Problem Diagnosis* for more information about the function and priority of trace.

ADIB
To specify this default through an ADIB, the correct structure is:

```
-------------------------------------
| Total length of ADS = 12          | Fullword integer
-------------------------------------
| ADS code = 102                    | Fullword integer
-------------------------------------
| Trace Control Word                | Fullword integer
-------------------------------------
```

TRTABLE (Trace Table Entries)
This default lets you determine the number of trace entries that the API holds in the cyclic in-core trace table. This does not apply to AIX.

Initial Preset Value
100 entries.

EDF
To specify this default through an EDF, the correct syntax is:

```
AFMMDFT TRTABL=n
```

where *n* is an integer, in the range of 5 to 1000, that defines the number of trace entries you want in the trace table.

ADIB
To specify this default through an ADIB, the correct structure is:

```
-------------------------------------
| Total length of ADS = 12          | Fullword integer
-------------------------------------
| ADS code = 103                    | Fullword integer
-------------------------------------
| Number of trace table entries     | Fullword integer
-------------------------------------
```

TSOTRCE (TSO Trace Output)

This default is an 8-character string indicating the DDNAME used by the API to direct trace output on MVS/TSO.

Initial Preset Value

AFMTRACE

EDF

To specify this default through an EDF, the correct syntax is:

```
AFMMDFT TSOTRCE=cccccccc
```

where *cccccccc* is the DDNAME for the MVS/TSO trace output.

ADIB

To specify this default through an ADIB, the correct structure is:

```
-------------------------------------
| Total length of ADS = 16          | Fullword integer
-------------------------------------
| ADS code = 401                    | Fullword integer
-------------------------------------
| MVS/TSO DDNAME                    | 8 character bytes
-------------------------------------
```

Nicknames

Using nicknames, you can modify values associated with a particular workstation environment. You can specify a different workstation type and connection identifier as well as processing options (called PROCOPTS: see PROCOPT (Processing Options)).

Each nickname specification may contain a workstation type (WSTYPE: see WSTYPE (Workstation Type)) and connection identifier (CONNID: see CONNID (Connection Identifier)) to match the workstation type and connection identifier specified on the Open Workstation (**GPOPWS**) or Create Workstation (**GPCRWS**) subroutine calls.

The nickname may also contain to-workstation types (TOWSTYPE: see TOWSTYPE (Target Workstation Type)) and to-connection identifiers (TOCONNID: see TOCONNID (Target Connection Identifier)) to replace the matched WSTYPE and CONNID specified on the **GPOPWS** or **GPCRWS** subroutines. Both of these specifications are optional. You can specify multiple nickname specifications and/or define more than one nickname for the same WSTYPE and CONNID.

Additionally, you may specify PROCOPT workstation environment values. These are applied to the workstation environment to allow customization of certain options.

A nickname in the EDF may then appear as

```
AFMMNICK CONNID=matched_connid,
         TOCONNID=new_connid,
         WSTYPE=matched_wstype,
         TOWSTYPE=new_wstype,
         PROCOPT=((keyword1,value1),(keyword2,value2)...)
```

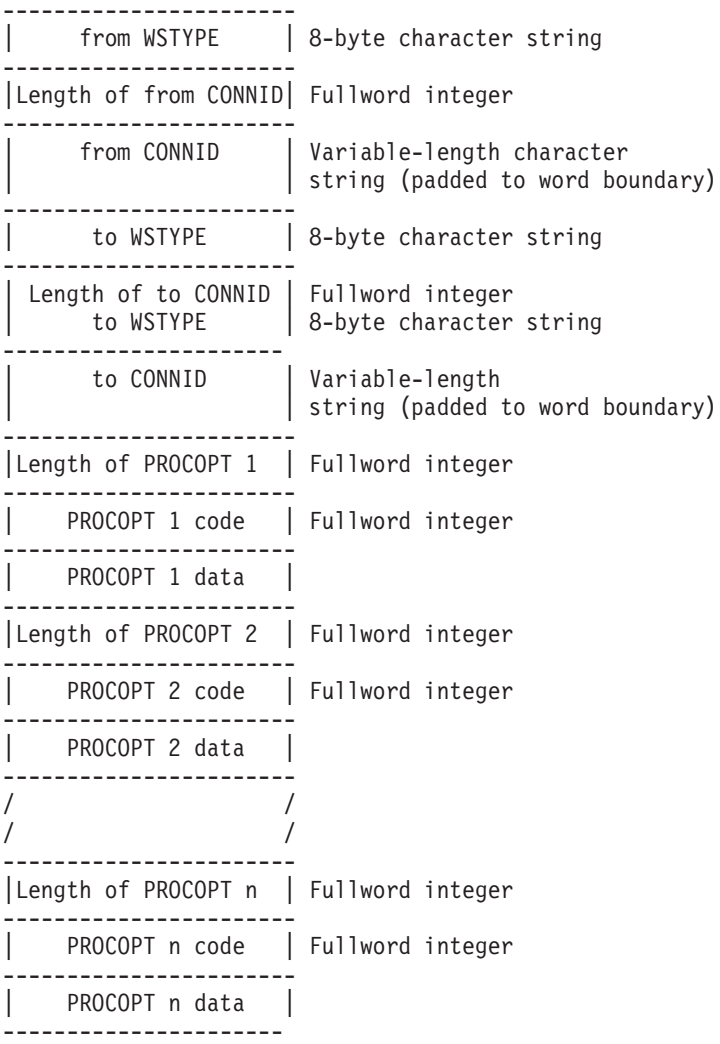
When you specify a nickname ADS in the ADIB, you specify:

- The total length of the field (such as a fullword integer)
- The nickname code
- The ADS data that includes character string fields that specify the WSTYPE (see WSTYPE (Workstation Type)) to be matched, the CONNID (see CONNID (Connection Identifier)) to be matched, a new WSTYPE, and a new CONNID you specify. If you do not want to specify a replacement WSTYPE or replacement CONNID field, specify blanks.
- Optional PROCOPT ADSs.

There are two formats for a nickname ADS. One format is for compatibility and supports connection identifiers that are specified as 8-byte character strings. A second format supports connection identifiers that are variable-length character strings.

Nickname Specification, ADS codes 2001 and 2002

```
FORMAT 1:
8-BYTE CHARACTER STRING CONNECTION IDENTIFIER
----------------------
| Total length of ADIB| Fullword integer
----------------------
| Total length of ADS | Fullword integer
----------------------
| ADS code = 2001     | Fullword integer
----------------------
|     from WSTYPE     | 8-byte character string
----------------------
|     from CONNID     | 8-byte character string
----------------------
|      to WSTYPE      | 8-byte character string
----------------------
|      to CONNID      | 8-byte character string
----------------------
|Length of PROCOPT 1  | Fullword integer
----------------------
|    PROCOPT 1 code   | Fullword integer
----------------------
|    PROCOPT 1 data   |
----------------------
|Length of PROCOPT 2  | Fullword integer
----------------------
|    PROCOPT 2 code   | Fullword integer
----------------------
|    PROCOPT 2 data   |
----------------------
/                     /
/                     /
----------------------
|Length of PROCOPT n  | Fullword integer
----------------------
|    PROCOPT n code   | Fullword integer
----------------------
|    PROCOPT n data   |
----------------------

FORMAT 2:
VARIABLE-LENGTH CHARACTER STRING CONNECTION IDENTIFIERS
----------------------
| Total length of ADIB| Fullword integer
----------------------
| Total length of ADS | Fullword integer
----------------------
| ADS code = 2002     | Fullword integer
```



See examples and explanations of PROCOPTS in PROCOPT (Processing Options). Default values are workstation-dependent.

How the graPHIGS API Processes Nicknames

The following steps explain the process the graPHIGS API uses to process nicknames.

1. When your application calls Open graPHIGS (**GPOPPH**), the API builds two lists of nicknames: one list from any nicknames found in the EDF and the other list from any nicknames found in the ADIB. These lists are used on all subsequent Open Workstation (**GPOPWS**) and Create Workstation (**GPCRWS**) subroutine calls.
2. When the Open or Create Workstation subroutine is called, the two nickname lists are scanned to determine replacement values for the workstation type and connection identifier and to obtain the associated PROCOPT values. (The criteria of the search is described below.) The workstation type and connection identifier passed on the subroutine call are used to scan the EDF file nicknames. Any replacement values for the workstation identifier and/or connection identifier are then used to scan the ADIB nicknames. (If the EDF scan did not result in replacement values, then the subroutine input values are used to scan the ADIB nicknames.) The connection identifier and workstation type nickname values from the ADIB scan are then used to open the specified device.
3. The nickname scan is a multi-pass scan that searches for nicknames which match the current workstation type and connection identifier: when matching values are found, the specified replacement values become the current workstation type and connection identifier, replacing the previous values. The scan is then resumed with the replacement values.

Each nickname scan operates as follows:

a. Before the scan is started, the API constructs a ″current″ parameter list that contains:
   - ″current workstation type″ (WSTYPE: see WSTYPE (Workstation Type))
   - ″current connection identifier″ (CONNID: see CONNID (Connection Identifier))

   These current values are those passed to each of the two scans (See step 2 above).

b. The nickname specifications are compared for any WSTYPE and CONNID values that both match the ″current″ WSTYPE and CONNID values. A blank or nulls in the nickname specification WSTYPE or CONNID always matches the ″current″ WSTYPE or CONNID value.

c. When a match is found, the API creates a ″replacement″ parameter list from the nickname TOWSTYPE and (TOWSTYPE: see TOWSTYPE (Target Workstation Type)) TOCONNID values, (TOCONNID: see TOCONNID (Target Connection Identifier)) and obtains any PROCOPT values specified by the nickname.

d. This process continues until the end of the nickname list is reached. If any matches were found, the ″current″ values are updated with the ″replacement″ values found (the TOWSTYPE and TOCONNID). The nickname list is then re-scanned using these new ″current″ values. Any previously-matched nicknames are excluded from the re-scan.

e. The API continues re-scanning (steps B through E above) the nickname list until a scan is completed without any further matches. The last ″current″ WSTYPE and CONNID values are returned from the scan process, along with any PROCOPT values from the matching nickname specification.

4. When the two nickname lists have been scanned, the final WSTYPE (see WSTYPE (Workstation Type)) and CONNID (see CONNID (Connection Identifier)) values are used to complete the Open Workstation or Create Workstation processing.
5. PROCOPTs (see PROCOPT (Processing Options)) may be specified as part of the nickname specifications and as a parameter on the Create Workstation subroutine call.

The PROCOPTs are merged together to form a combined PROCOPT list. The merge is performed as follows:

1. If the subroutine call is for Create Workstation and PROCOPTs (see PROCOPT (Processing Options)) are specified, the entire list of specified PROCOPTs is obtained.
2. If any PROCOPTs were specified on the matching nickname from the ADIB scan, then any of these PROCOPTs not already specified in the Create Workstation list are added to that list of PROCOPTs (if any).
3. If any PROCOPTs were specified on the matching nickname from the EDF scan, then any of these PROCOPTs not already specified in the merged list are added to that list of PROCOPTs (if any).

The resultant merged list of PROCOPTs is then used to complete the Open or Create workstation processing.

Nickname Syntax

Nicknames are intended to modify values associated with a particular workstation environment. The syntax explanations of a nickname set through an EDF and an ADIB follow (see ADIB).

CONNID (Connection Identifier)

The CONNID is a string of characters used to specify the connection identifier to which this nickname applies.

TOCONNID (Target Connection Identifier)

The target connection identifier is a string of characters that replace the specified connection identifier, if the *wstype* and *connid* parameters match the corresponding actual values.

WSTYPE (Workstation Type)

The workstation type is a string of 0-8 characters used to match the nickname workstation type.

TOWSTYPE (Target Workstation Type)

The actual workstation type is a string of 0-8 characters that are the *real* value substituted for a connection identifier. The *towstype* replaces the actual workstation type if the *wstype* and *connid* parameters match the corresponding actual values.

PROCOPT (Processing Options)

PROCOPTS are a list of processing options. You can use these options to change the way in which the API treats a specific device or to specify workstation-specific default values. PROCOPTS can also be specified with the **option** parameter on the Create Workstation subroutine call (**GPCRWS**). Each specification defines a specific workstation option followed by a number of arguments that are valid for that option.

PROCOPTS are listed in the form of ″PROCOPT-specifications″ (PROCOPT-specs). Each PROCOPT-spec defines a specific workstation option followed by a number of arguments that are valid for that option in the following format:

```
PROCOPT=((PROCOPT_spec),(PROCOPT_spec),...)
```

The expanded PROCOPT appears like this:

```
PROCOPT=((option_keyword,argument,argument),(option_keyword,argument),...
```

See PROCOPTS for details about PROCOPTS.

Nickname Example

The following sample EDF nickname specification changes from one workstation to another and specifies a PROCOPT:

```
AFMMNICK CONNID=*,
         TOCONNID=IBM5080,
         WSTYPE=GDDM,
         TOWSTYPE=5080,
         PROCOPT=((DISPLMOD,5080-16))
```

PROCOPTS

PROCOPTS are a list of processing options. You can use these options to change the way in which the API treats a specific device or to specify workstation-specific default values. PROCOPTS can also be specified with the **option** parameter on the Create Workstation (**GPCRWS**) subroutine. Each specification defines a specific workstation option followed by a number of arguments that are valid for that option.

Examples and explanations of PROCOPT specifications follow. Default values are workstation-dependent. See Workstation Description Tables for specific workstation values.

CLDEVS (Create Input Device)

This PROCOPT lets you create a logical input device even if a physical device is not present.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((CLDEVS),...
```

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

```
-------------------------------------
| Total length of ADS = 8           | Fullword integer
-------------------------------------
|  ADS code =  23                   | Fullword integer
-------------------------------------
```

DCMETERS (Device Coordinate Meters)

This PROCOPT enables you to define the width and height of the display device in Device Coordinate (DC) meters. Both the width and the height values must be greater than zero (0.0). The maximum value for both the width and the height is subject to the capabilities of the workstation.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((DCMETERS, float1, float2),...
```

where *float1* is the display width specified in meters and *float2* is the display height specified in meters.

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

```
-------------------------------------
|  Total length of ADS = 16         | Fullword integer
-------------------------------------
|  ADS code =  36                   | Fullword integer
-------------------------------------
|  Display width in meters          | Short floating-point value
-------------------------------------
|  Display height in meters         | Short floating-point value
-------------------------------------
```

DCTES (Depth Cue Table)

This PROCOPT lets you specify the number of depth cue table entries.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((DCTES, n),...
```

where *n* is the number of depth cue table entries.

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

```
-------------------------------------
|  Total length of ADS = 12         | Fullword integer
-------------------------------------
|  ADS code = 21                    | Fullword integer
-------------------------------------
|  Number of depth cue entries      | Fullword integer
-------------------------------------
```

DCUNITS (Device Coordinate Address Units)

This PROCOPT enables you to define the width and height of the display in Device Coordinate (DC) address units. The minimum value for either the width or the height is 8. Likewise, the maximum value for either the width or the height is 4096.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((DCUNITS, n1, n2),...
```

where *n1* is the display width in address units and *n2* is the display height in address units.

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

```
-------------------------------------
|  Total length of ADS = 16         | Fullword integer
-------------------------------------
|  ADS code =  35                   | Fullword integer
-------------------------------------
|  Display width in address units   | Fullword integer
-------------------------------------
|  Display height in address units  | Fullword integer
-------------------------------------
```

DIRCOLOR (Direct Color)

This PROCOPT lets you specify workstation color tables to be initialized for direct color rather than initialized for the default which is indexed color.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((DIRCOLOR),...
```

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

```
-------------------------------------
| Total length of ADS = 8           | Fullword integer
-------------------------------------
|  ADS code =  29                   | Fullword integer
-------------------------------------
```

DISPLMOD (Display Model)

This PROCOPT lets you specify the size of the display for the workstation.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((DISPLMOD, cccccccc),...
```

where *cccccccc* is an 8-character string specifying the display model.

See Display Models for possible values of this PROCOPT.

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

```
-------------------------------------
|  Total length of ADS = 16         | Fullword integer
-------------------------------------
|  ADS code = 9                     | Fullword integer
-------------------------------------
|  Display model                    | 8-byte character string
-------------------------------------
```

DUMPFLGS (Dump Flags)

This PROCOPT lets you indicate the enabling and disabling of the 5080 workstation diagnostic options.

For specific information on the setting of the dump flag values, see *The GDDM/graPHIGS Programming Interface: Installation and Problem Diagnosis*.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((DUMPFLGS, n),...
```

where *n* is the desired option value.

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

Total length of ADS = 12	Fullword integer
ADS code = 5	Fullword integer
5080 diagnostic options	Fullword integer

DUMPPRFX (Dump Prefix)

This PROCOPT provides the prefix of the file to which 5080 workstation diagnostic data is written.

For specific device information, see *The GDDM/graPHIGS Programming Interface: Installation and Problem Diagnosis*.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((DUMPPRFX, cccc),...
```

where *cccc* is a 4-character string specifying the file prefix.

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

Total length of ADS = 12	Fullword integer
ADS code = 6	Fullword integer
File prefix for 5080 diagnostic data	4-byte character string

EBTES (Edge Bundle Table)

This PROCOPT lets you specify the number of edge bundle table entries.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((EBTES, n),...
```

where *n* is the number of edge bundle table entries.

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

Total length of ADS = 12	Fullword integer
ADS code = 20	Fullword integer
Number of edge bundle table entries	Fullword integer

ECHOMETH (Input Echo)

This PROCOPT lets you specify the type of input echoing to be done for NATIVE or X workstations.

For displays with less than 8-bit planes, the default echo is Exclusive-Or on the color table.

With bit plane echoing, the number of available colors is halved and the echo is always white.

With XOR, echoing is implemented by exclusive-or on the color table. For example, with a 16-color, 4-bit plane device, suppose a locator echo area is filled with color 5 and a rubber-band line prompt/echo type is selected. This rubber-band line will appear in color 10 (0101 exclusive or'd with 1111 = 1010). In this instance, you should ensure that color 10 is visible with a background color of 5.

The colors of:

- String input echo
- Valuator input echo
- The locator, stroke, and pick prompts

will not be set by XOR echoing.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((ECHOMETH, n),...
```

where *n* is 1 for reserve bit plane or 2 for XOR.

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

```
-------------------------------------
|  Total length of ADS = 12         | Fullword integer
-------------------------------------
|  ADS code = 10                    | Fullword integer
-------------------------------------
|  Method of input echoing          | Fullword integer
-------------------------------------
```

FBUFFER (Frame Buffer Configuration)

This PROCOPT lets you select the number of logical buffers that comprise the physical frame buffer.

Note: This PROCOPT is only supported when running Direct Window Access using the *High Performance 3D Color Graphics Processor 8 bit or 24 bit)*.

By default, if running with **Direct Window Access:**

- *The High Performance 3D Color Graphics Processor with 8 bit planes* has a frame buffer consisting of one buffer of 8 bits.
- *The High Performance 3D Color Graphics Processor with 24 bit planes* has a frame buffer consisting of two buffers of 12 bits each.
- *The High Performance Graphics Subsystem (Model 730, 8 bit or 24 bit)*, *POWER GtO (8 bit or 24 bit)*, and *POWER Gt4x (8 bit or 24 bit)* has a frame buffer consisting of two buffers of either 8 or 24 bits each.

If not running with the Direct Window Access: all adapters have a frame buffer consisting of one buffer and this PROCOPT is ignored.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((FBUFFER, n),...
```

where *n* is the number of buffers and may be set to a value of 1 or 2.

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

```
-------------------------------------
|  Total length of ADS = 12         | Fullword integer
-------------------------------------
|  ADS code = 31                    | Fullword integer
-------------------------------------
|  Organization of frame buffer     | Fullword integer
-------------------------------------
```

FONTLIST (Character Font List)

This PROCOPT allows you to specify user-defined character sets as valid character set/font identifiers for a 5080 workstation.

Specify a list of up to 10 (the maximum allowable) filenames of the symbol definition files of the desired user-defined character sets. If a filename cannot be found, it will be ignored and no error will be generated.

Note: All the character sets that the graPHIGS API defines are automatically included and never need to be specified via the FONTLIST.

Below is an example of the PROCOPT you would use to specify Character Set 229 Font 128, Character Set 229 Font 129, and Character Set 101 Font 128:

```
PROCOPT=((FONTLIST,AFME580,AFME581,AFM6580)
```

Note: When a 5080 workstation is opened, the API also uses these names to determine the sizes of the desired character sets in order to allocate sufficient storage in the 5085 for them.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((FONTLIST, cccccccc,cccccccc,...),...
```

where *cccccccc* is a valid 8-character character set filename.

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

```
-------------------------------------
|  Total length of ADS =  n         | Fullword integer
|  (application dependent)          |
-------------------------------------
|  ADS code = 15                    | Fullword integer
-------------------------------------
|  Up to 10 font list filenames     | n 8-byte character strings
|                                   |    .
|                                   |    .
-------------------------------------
```

FONTPSIZ (Font Pool Size)

For a 5080 workstation, this PROCOPT indicates the maximum number of character sets which can be active at one time (including the primary character set). Note also the considerations in IBM 5080 Character Set Restrictions.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((FONTPSIZ, n),...
```

where *n* is an integer between 1 and 10 (maximum allowable value). If you do not specify FONTPSIZ, a value of 3 is used.

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

```
-------------------------------------
|  Total length of ADS = 12         | Fullword integer
-------------------------------------
|  ADS code =  14                   | Fullword integer
-------------------------------------
|  Number of 5080 maximum active    | Fullword integer
|  character font sets (1-10)       |
-------------------------------------
```

HWCURS (Hardware Crosshair Cursor)

This PROCOPT allows you to request the use of the full-screen hardware crosshair cursor provided by the X cursor extension code in place of the usual crosshair cursor provided by the graPHIGS API (through the use of the **GPCUS** subroutine).

The hardware cursor provides a performance improvement over the current crosshair cursor. Unfortunately, though, the hardware cursor spans the entire display. For applications that normally use full-screen windows this will not be a problem. For applications that use smaller windows, the user may or may not wish to use this cursor.

The PROCOPT will be supported on all graPHIGS X workstation types. The X server where the window will be created must support the X cursor extension in order for this PROCOPT to take effect.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((HWCURS)),...
```

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

```
-------------------------------------
|  Total length of ADS = 8          | Fullword integer
-------------------------------------
|  ADS code =  39                   | Fullword integer
-------------------------------------
```

Note: The hardware cursor may also be activated by defining an environment variable (in lieu of using a PROCOPT). If the **gPHWCURS** environment variable is defined at the time that the nucleus is started (in the case of a remote nucleus being started via the **gPinit** command) or when the application is started, then when a crosshair cursor is requested (as defined by the **GPCUS** subroutine), the full-screen hardware crosshair cursor will be used instead of the regular crosshair cursor provided by the graPHIGS workstation.

IBTES (Interior Bundle Table)

This PROCOPT lets you specify the number of interior bundle table entries.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((IBTES, n),...
```

where *n* is the number of interior bundle table entries.

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

```
-------------------------------------
|  Total length of ADS = 12         | Fullword integer
-------------------------------------
|  ADS code =  19                   | Fullword integer
-------------------------------------
|  Number of interior bundle table  | Fullword integer
|  entries                          |
-------------------------------------
```

IMAGEFMT (Image Output Format)

This PROCOPT enables you to define the image format used for output images. The valid formats are `1=PSL1_4BIT`, `2=PSL1_8BIT`, and `3=IOCA_FS10` where:

- `PSL1_4BIT` defines a 12 bit per-pixel image in PostScript Level One format
- `PSL1_8BIT` defines a 24 bit per-pixel image in PostScript Level One format
- `IOCA_FS10` defines a one-bit per-pixel image in IBM's Image Object Content Architecture (IOCA) FS10 format.

The default image format is `1=PSL1_4BIT`.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((IMAGEFMT, n),...
```

where *n* is an image output format.

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

```
-------------------------------------
|  Total length of ADS = 12         | Fullword integer
-------------------------------------
|  ADS code =  37                   | Fullword integer
-------------------------------------
|  Image output format              | Fullword integer
-------------------------------------
```

KEYBOARD (Language Keyboard)

This PROCOPT lets you specify the keyboard installed for the NATIVE workstation.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((KEYBOARD, n),...
```

where *n* is an integer from the following list to identify the keyboard:

```
1 = United States English
2 = United Kingdom English
3 = German
4 = French
5 = Italian
6 = Japanese
7 = Swedish
```

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

Total length of ADS = 12	Fullword integer
ADS code = 11	Fullword integer
Number that identifies the installed keyboard	Fullword integer

LOCDEVS (Locator Devices)

This PROCOPT lets you specify the number of locator devices for the workstation. The limit for the number of locator devices is device dependent. For specific device information, see Locator Devices.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((LOCDEVS, n),...
```

where *n* is the number of locator devices your workstation is to support.

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

Total length of ADS = 12	Fullword integer
ADS code = 3	Fullword integer
Number of locator devices	Fullword integer

LSTES (Light Source Table)

This PROCOPT lets you specify the number of light source table entries.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((LSTES, n),...
```

where *n* is the number of light source table entries.

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

Total length of ADS = 12	Fullword integer
ADS code = 22	Fullword integer
Number of light source table entries	Fullword integer

PLBTES (Polyline Bundle Table)

This PROCOPT lets you specify the number of polyline bundle table entries.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((PLBTES, n),...
```

where *n* is the number of polyline bundle table entries.

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

```
-------------------------------------
|  Total length of ADS = 12         | Fullword integer
-------------------------------------
|  ADS code = 16                    | Fullword integer
-------------------------------------
|  Number of polyline bundle table  | Fullword integer
|  entries                          |
-------------------------------------
```

PMBTES (Polymarker Bundle Table)

This PROCOPT lets you specify the number of polymarker bundle table entries.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((PMBTES, n),...
```

where *n* is the number of polymarker bundle table entries.

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

```
-------------------------------------
|  Total length of ADS = 12         | Fullword integer
-------------------------------------
|  ADS code =  17                   | Fullword integer
-------------------------------------
|  Number of polymarker bundle      | Fullword integer
|  table entries                    |
-------------------------------------
```

PNTHLHSR (Annotation Text and Marker Hidden Line Hidden Surface Removal)

This PROCOPT enables you to define the coordinate system in which HLHSR of all annotation text and polymarker primitives will occur. The valid values are `1=VIEWING_COORDINATES` and `2=DEVICE_COORDINATES` `1=VIEWING_COORDINATES`.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((PNTHLHSR, n),...
```

where *n* is the primitive's HLHSR coordinate system.

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

```
-------------------------------------
|  Total length of ADS = 12         | Fullword integer
-------------------------------------
|  ADS code =  38                   | Fullword integer
-------------------------------------
|  Primitive's HLHSR coordinate     | Fullword integer
|  system                           |
-------------------------------------
```

STRDEVS (String Devices)

This PROCOPT lets you specify the number of string devices for the workstation. For specific device information, see String Devices.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((STRDEVS, n),...
```

where *n* is the number of string devices. The limit for the number of string devices is device dependent. For specific device information, see String Devices.

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

```
-------------------------------------
|  Total length of ADS = 12         | Fullword integer
-------------------------------------
|  ADS code = 4                     | Fullword integer
-------------------------------------
|  Number of string devices         | Fullword integer
-------------------------------------
```

TXBTES (Text Bundle Table)

This PROCOPT lets you specify the number of text bundle table entries.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((TXBTES, n),...
```

where *n* is the number of text bundle table entries.

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

```
-------------------------------------
|  Total length of ADS = 12         | Fullword integer
-------------------------------------
|  ADS code = 18                    | Fullword integer
-------------------------------------
|  Number of text bundle table      | Fullword integer
|  entries                          |
-------------------------------------
```

VWTBLSZ (View Table Entries)

This PROCOPT lets you specify the number of view table entries for the workstation. For specific device information, see General Workstation Facilities.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((VWTBLSZ, n),...
```

where *n* is the number of view table entries. All workstations must have at least one definable view table entry. Therefore, you must specify VWTBLSZ greater than or equal to 2 (view 0 plus one definable entry). The maximum limit for the view table size is device dependent.

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

```
-------------------------------------
|  Total length of ADS = 12         | Fullword integer
-------------------------------------
```

```
|  ADS code =  2                    | Fullword integer
-------------------------------------
|  Number of view table entries     | Fullword integer
-------------------------------------
```

XNAME (X Default String)

This PROCOPT lets you specify a name to be used to resolve the defaults in the .Xdefaults file. If this PROCOPT is not specified, then the graPHIGS API will use the default string 'graPHIGS' to resolve the defaults.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
     AFMMNICK PROCOPT=((XNAME, cccc),...
```

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

```
-------------------------------------
|  Total length of ADS = variable   | Fullword integer
-------------------------------------
|  ADS code =  27                   | Fullword integer
-------------------------------------
|  Length of default string         | Fullword integer
-------------------------------------
|  Default String                   | Variable length character string
|                                   | padded to a word boundary
-------------------------------------
```

XNOCLRMP (Do Not Create an X Color Map)

This PROCOPT lets you specify to not have graPHIGS API create an X color map. This implies that the application cannot access the color map via graPHIGS API.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
     AFMMNICK PROCOPT=((XNOCLRMP),...
```

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

```
-------------------------------------
|  Total length of ADS = 8          | Fullword integer
-------------------------------------
|  ADS code =  28                   | Fullword integer
-------------------------------------
```

XWINDASP (Window Aspect Ratio)

This PROCOPT lets you change the aspect ratio of the window to the displayed surface of the X Workstation. Whenever the graPHIGS API window is resized, the device driver establishes a new display surface size within that window. This is the largest subarea of the resized X-Window that maintains the aspect ratio provided by this procopt (or the root window, if XWINDASP PROCOPT is not specified). The current display contents are scaled uniformly to this new mapped display surface size.

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
     AFMMNICK PROCOPT=((XWINDASP, float1,
float2),...
```

where *float1* is the value of the aspect ratio specifying the size in the x direction and *float2* is the value of the aspect ratio specifying the size in the y direction.

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

```
-------------------------------------
|  Total length of ADS = 16         | Fullword integer
-------------------------------------
|  ADS code =  30                   | Fullword integer
-------------------------------------
|  size of the window (x-direction) | Floating-point number
-------------------------------------
|  size of the window (y-direction) | Floating-point number
-------------------------------------
```

XWINDID (X Window Identifier)

This PROCOPT lets you pass a window identifier to the graPHIGS API. The window was created by your application and the graPHIGS API will use this for the X workstation display window. (See the **GPES** subroutine (escape 1018) for information on getting a list of visuals in order to guarantee that the application's window will be usable by the graPHIGS nucleus.)

EDF

To specify this PROCOPT through an EDF, the correct syntax is:

```
AFMMNICK PROCOPT=((XWINDID, n),...
```

ADIB

To specify this PROCOPT through an ADIB, the correct structure is:

```
-------------------------------------
|  Total length of ADS = 12         | Fullword integer
-------------------------------------
|  ADS code =  25                   | Fullword integer
-------------------------------------
|  Window Identifier                | Fullword integer
-------------------------------------
```

PROCOPT Parameters Table

This table summarizes the parameters you can specify with a PROCOPT.

Table 106. PROCOPT Parameters

Identifier	Length	Type	Name	Description	Workstations
2	12	Integer	VWTBLSZ	Number of view table entries	6090, 5080, GDDM, NATIVE, GDF, X, XSOFT, XPEX, IMAGE
3	12	Integer	LOCDEVS	Number of locator devices	6090, 5080, GDDM, NATIVE, X, XSOFT, XPEX
4	12	Integer	STRDEVS	Number of string devices	6090, 5080, GDDM, NATIVE, X, XSOFT, XPEX
5	12	Bit (32)	DUMPFLGS	Dump flag	5080
6	16	Char (4)	DUMPPRFX	Dump file prefix	5080
9	16	Char (8)	DISPLMOD	Monitor model	5080, NATIVE
10	12	Integer	ECHOMETH	Input device echo method	NATIVE, X, XSOFT
11	12	Integer	KEYBOARD	Keyboard language	NATIVE

Table 106. PROCOPT Parameters (continued)

Identifier	Length	Type	Name	Description	Workstations
14	12	Integer	FONTPSIZ	Font pool size	5080
15	16-88[1]	Char (8)	FONTLIST	Font list	5080
16	12	Integer	PLBTES	Number of polyline bundle table entries	6090, X, XSOFT, XPEX, IMAGE
17	12	Integer	PMBTES	Number of polymarker bundle table entries	6090, X, XSOFT, XPEX, IMAGE
18	12	Integer	TXBTES	Number of text bundle table entries	6090, X, XSOFT, XPEX, IMAGE
19	12	Integer	IBTES	Number of interior bundle table entries	6090, X, XSOFT, XPEX, IMAGE
20	12	Integer	EBTES	Number of edge bundle table entries	6090, X, XSOFT, XPEX, IMAGE
21	12	Integer	DCTES	Number of depth cue table entries	6090, X, XSOFT, XPEX, IMAGE
22	12	Integer	LSTES	Number of light source table entries	6090, X, XSOFT, XPEX, IMAGE
23	8		CLDEVS	Create logical input devices even if physical devices are not present	6090, X, XSOFT, XPEX
25	12	Integer	XWINDID	X workstation display window identifier	X, XSOFT, XPEX
27	Variable	Char	XNAME	Name used to resolve the defaults in the .Xdefaults file	X, XSOFT, XPEX
28	8		XNOCLRMP	Do not have the graPHIGS API create an X color map	X, XSOFT, XPEX
29	8		DIRCOLOR	Initialize workstation color map for direct color	X, XSOFT, XPEX
30	16	Float (2)	XWINDASP	Initial Aspect Ratio	X, XSOFT, XPEX
31	12	Integer	FBUFFER	Number of buffers that comprise the frame buffer	X, XPEX

Table 106. PROCOPT Parameters (continued)

Identifier	Length	Type	Name	Description	Workstations
35	16	Integer (2)	DCUNITS	Device Coordinate address units	IMAGE
36	16	Float (2)	DCMETERS	Device Coordinate meters	IMAGE
37	12	Integer	IMAGEFMT	Image output format	IMAGE
38	12	Integer	PNTHLHSR	Primitive's HLHSR Coordinate system	X, XSOFT, IMAGE
39	8	Integer	HWCURS	Hardware Crosshair Cursor	X, XSOFT
Note: [1] Application-dependent					

Part 4. Character Sets and Fonts

Chapter 8. Character Set Facilities of the graPHIGS API

There are several facilities available for the display of text information:

- Geometric text and annotation text that allow an application to specify a character string that is to be displayed on a workstation
- Echo of string device input that allows a user to see the input being entered on a string device (for example, a keyboard).

Your application can specify two attributes that affect the interpretation and display of text information: the character set identifier (CSID) and the font identifier. When the application specifies a character to be displayed, the CSID and font identifier together determine the symbol that is displayed to represent the character.

Identifying a Character Set

A CSID is a number that identifies a set of graphic characters (numbers, letters, and special characters) that is treated as an entity. For example, a CSID may specify characters that represent the alphabet of a language (for example, English or Kanji).

Each character within a CSID has assigned to it a code point, which is a unique bit pattern used to represent the character. For example, the letter ″A″ has been assigned the EBCDIC code point X'C1' and the ASCII code point X'41'. Character sets that use 8 bits (1 byte) to represent the character are referred to as single-byte character sets (SBCS). Character sets that use 16 bits (2 bytes) to represent the character are referred to as double-byte character sets (DBCS). The code points assigned to letters, numbers, and certain common symbols follow the EBCDIC or the ASCII encoding sequences.

Character sets can differ from each other in two ways: different sets may vary in the code points used in the set; they may also vary in the symbol displayed for a given code point.

Your application program specifies CSIDs as integer values. The API divides CSID values into the following ranges:

CSID 1 - CSID 100	Reserved for use by IBM for single-byte character sets
CSID 101 - CSID 127	Reserved for your use for single-byte character sets
CSID 128 - CSID 228	Reserved for use by IBM for double-byte character sets
CSID 229 - CSID 255	Reserved for your use for double-byte character sets

Identifying a Font

A font identifier is a number that identifies an appearance of the characters in the character set. For example, US English (CSID 1) has fonts that display characters in Italic, Script, Gothic, etc.

Fonts differ from each other only in the appearances of the symbols displayed for a given character. Different fonts within a character set all maintain the same correspondence between characters and code points.

Within each CSID, the API defines these font identifier values:

FONT 1 - FONT 127	Reserved for assignment by IBM in IBM-reserved character sets; available for your use in user-reserved character sets
FONT 128 - FONT 255	Reserved for your use

Using the Character Set Facilities

Although CSIDs and font identifiers are used with geometric text, annotation text, and input devices, these facilities have different restrictions in supporting various character sets and fonts.

You can create your own character sets and fonts for use with geometric text; you cannot define your own character sets and fonts for annotation text and input device processing.

Some workstations may support only one CSID (or one CSID at a time) for annotation text and input devices.

For input device processing, you may specify only the CSID, not the font identifier.

Chapter 9. Character Sets and Fonts Provided by the API

The graPHIGS API provides several character sets in various fonts. In addition, you can create your own character sets and fonts for use with geometric text. (See User-Definable Fonts for further information.)

This section contains charts of fonts (excluding Hangul, Kanji, Traditional Chinese, Simplified Chinese and Unicode) supported by the graPHIGS API for the S/390 and operating system. Each chart presents the actual characters that correspond to each character code in the supported fonts. When using annotation text, the appearance and, in some cases, the availability of particular characters may vary. Also, only a subset of the characters can generally be generated by the keyboard or input device for input characters.

Note: For ease of use, nondisplayable characters are represented by a blank ( ). The space or blank character is represented by the symbol SP.

Using the Unicode Character Set

The Unicode standard, modeled on the ASCII character set, is a universal set of characters. It includes characters and common technical symbols from the world's scripts. The graPHIGS API provides a Unicode character set (CSID 131), which is a subset of the characters included in the Unicode standard. The subset includes all characters and symbols supported in other graPHIGS API character sets, with the exception of a few technical drawing, or engineering, symbols.

The graPHIGS API does not support bidirectional text rendering of Unicode text elements.

The graPHIGS API Unicode text processing does not support non-spacing mark, or ″dead-key″ processing. That is, all elements described in the Unicode standard as non-spacing marks will be treated as spacing marks and occupy a spacing position by themselves. The graPHIGS application should use pre-combined character elements, when possible.

> **Note:** See *The Unicode Standard, Worldwide Character Encoding* Version 1.0, Volume 1, Addison-Wesley Publishing Company, Inc., 1991, and *The Unicode Standard, Worldwide Character Encoding* Version 1.0, Volume 2, Addison-Wesley Publishing.

Proportional character symbol positioning is not supported for Unicode character sets.

Using Kanji Character Sets in the Operating System

For the operating system, the graPHIGS API supports two unique encodings for the Kanji character set. CSID 128 supports the IBM-932 Japanese encoding, while CSID 134 supports the IBM-943 encoding. Your application should choose the character set identifier whose encoding best suits your application and operating system.

In order to accommodate applications that wish to change encodings without making coding changes to modify the CSID from 128 to 134, the graPHIGS API has provided an environment variable which will map all occurrences of CSID 128 as if it was coded to CSID 134. That is, when using this environment variable, although the application is coded as specifying 128, the graPHIGS API will treat it as CSID 134. This environment variable should be specified as **AFM_FORCE_IBM943**, if you wish to invoke this behavior. If the environment variable is not specified, all occurrences of CSID 128 will remain unaffected.

Using this environment variable has severe restrictions. It cannot be used in when the application specifies a remote nucleus configuration. Also, caution should be taken when modifying or supplementing the graPHIGS API font files. If **AFM_FORCE_IBM943** is specified, the graPHIGS API will use the CSID 134 font files, namely, afm86.sym and afm8601.sym. Since the application is coded to use CSID 128 in this

case, the application should insure that there are no user-defined font files intended to supplement the CSID 128 font files, afm80.sym and afm8001.sym, which will no longer be referenced.

Character Code Points and Symbols

The graPHIGS API provides these fonts with characters mapped to the EBCDIC or ASCII standard:

Table 107. Available Fonts

CSID 1 - US English	FONT 1 graPHIGS Default FONT 2 Complex Roman FONT 3 Complex Italian FONT 4 Complex Script FONT 5 Duplex Roman FONT 6 Gothic English FONT 7 Gothic German FONT 8 Gothic Roman FONT 9 Simplex Roman FONT 10 Triplex Italic FONT 11 Triplex Roman FONT 12 Filled FONT 13 Proportional FONT 14 Filled-Proportional
CSID 2 - UK English	FONT 1 graPHIGS Default
CSID 3 - German	FONT 1 graPHIGS Default
CSID 4 - French	FONT 1 graPHIGS Default
CSID 5 - Italian	FONT 1 graPHIGS Default
CSID 6 - Japanese Katakana	FONT 1 graPHIGS Default FONT 2
CSID 7 - Swedish	FONT 1 graPHIGS Default
CSID 8 - Multinational	FONT 1 graPHIGS Default
CSID 9 - Single-byte Korean	FONT 1 graPHIGS Default
CSID 10 - ISO 8859-1 (Latin 1)	
CSID 11 - ISO 8859-2[3]	FONT 1 graPHIGS Default
CSID 12 - ISO 8859-5 Cyrillic[4]	FONT 1 graPHIGS Default
CSID 128 - Japanese Kanji (IBM-932 encoding)[2]	FONT 1 graPHIGS Default
CSID 129 - Hangul[5]	FONT 1 graPHIGS Default
CSID 130 - Traditional Chinese[1, 6]	FONT 1 graPHIGS Default
CSID 131 - Unicode	FONT 1 graPHIGS Default
CSID 132 - Simplified Chinese[1, 7]	FONT 1 graPHIGS Default
CSID 134 - Japanese Kanji (IBM-943 encoding)[1]	FONT 1 graPHIGS Default

Notes:

1. CSIDs 130, 132, and 134 are not supported in the EBCDIC environment.
2.
   - For all 5085 feature models, the feature code is 2986 (S/390 systems).
   - For 5086 models, the feature codes are 4111 (S/390 systems), 4222 (Kanji ROS), and 4651 (graphic keyboard). Customers in Japan specify feature code 2973 for the graphic keyboard rather than feature code 4651.
   - On the NATIVE workstations, Kanji input requires the NATIVE AIX GSL Japanese Language Support and the Japanese keyboard.

3. CSID 11, iso8859-2, is only valid when you are using the Czech locale, cs_CZ.
4. CSID 12, Cyrillic is only valid when you are using the Cyrillic locale, ru_RU.
5. Hanja characters are not supported. For the 5080 Workstation, Hangul requires models with NLS microcode and the HGC utility.
6. For Traditional Chinese, the graPHIGS API provides this font with characters mapped only to ASCII code points. The code points follow the IBM-927 Corporate Specification (PS/55 double-byte only code structure). For details, see *AIX Chinese Code Book*.
7. For Simplified Chinese, the graPHIGS API provides this font with characters mapped only to ASCII code points. The encoding for the font is the IBM CH-S DBCS_EUC GB 2312 set.

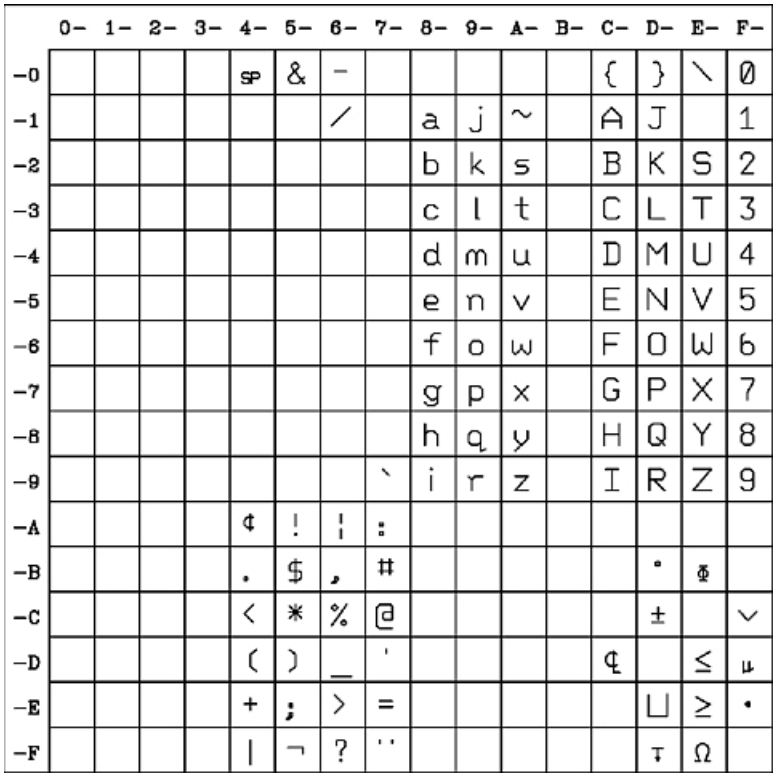
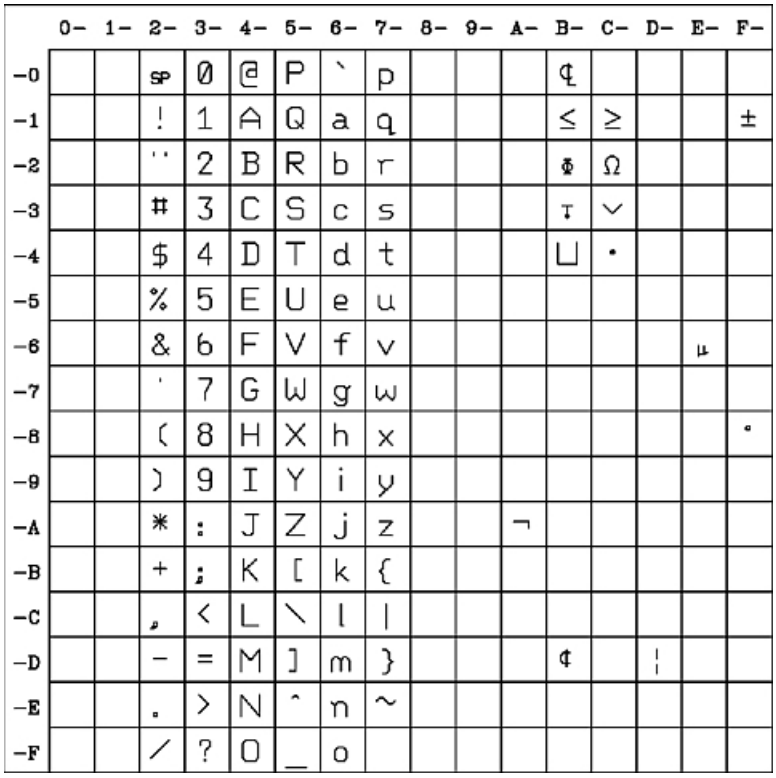
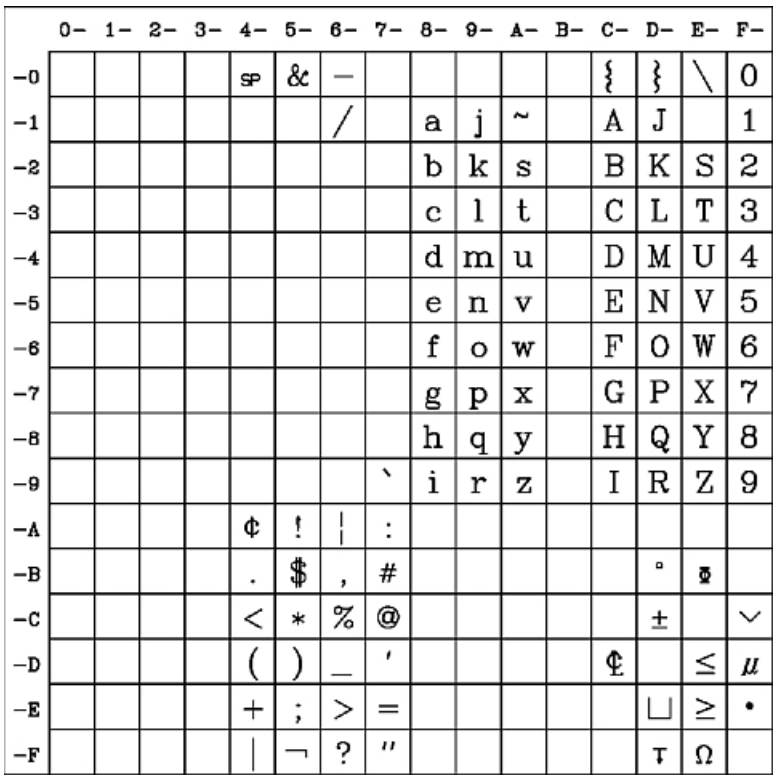


Figure 6. (EBCDIC) US English Character Set (1). Font 1 (Primary)



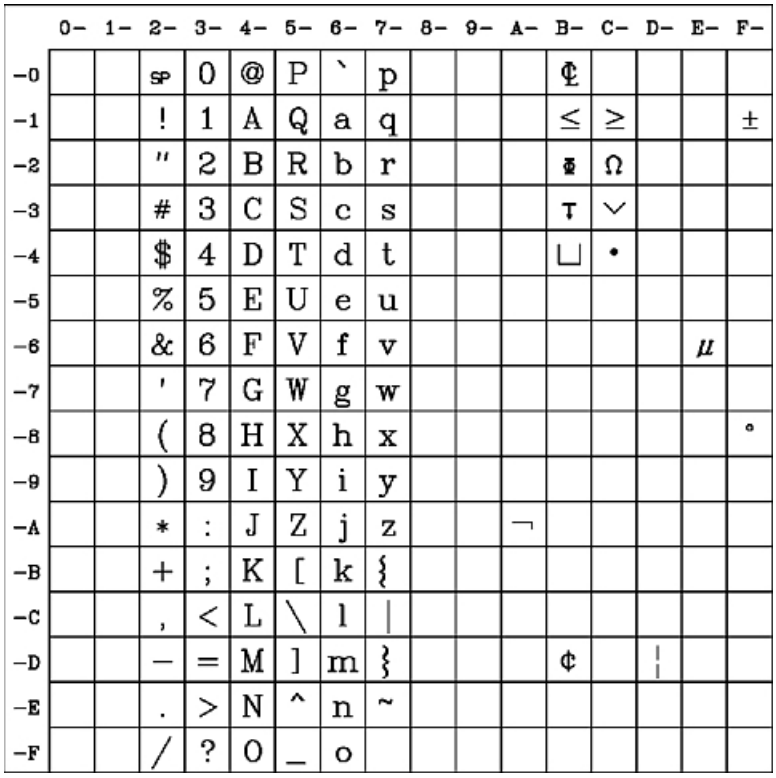
	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0			SP	0	@	P	`	p				¢				
-1			!	1	A	Q	a	q				≤	≥			±
-2			"	2	B	R	b	r				¤	Ω			
-3			#	3	C	S	c	s				¥	∨			
-4			$	4	D	T	d	t				⌴	•			
-5			%	5	E	U	e	u								
-6			&	6	F	V	f	v							µ	
-7			'	7	G	W	g	w								
-8			(	8	H	X	h	x								°
-9			)	9	I	Y	i	y								
-A			*	:	J	Z	j	z			¬					
-B			+	;	K	[	k	{								
-C			,	<	L	\	l	|								
-D			-	=	M	]	m	}				¢		¦		
-E			.	>	N	^	n	~								
-F			/	?	O	_	o									

Figure 7. (ASCII) US English Character Set (1). Font 1 (Primary)



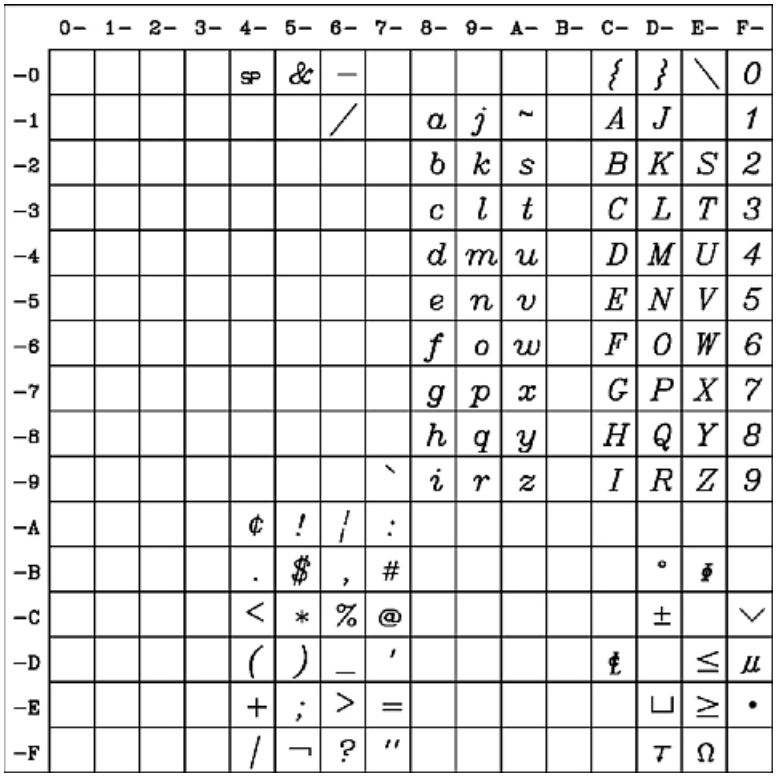
	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0					SP	&	-						{	}	\	0
-1							/		a	j	~		A	J		1
-2									b	k	s		B	K	S	2
-3									c	l	t		C	L	T	3
-4									d	m	u		D	M	U	4
-5									e	n	v		E	N	V	5
-6									f	o	w		F	O	W	6
-7									g	p	x		G	P	X	7
-8									h	q	y		H	Q	Y	8
-9								`	i	r	z		I	R	Z	9
-A					¢	!	¦	:								
-B					.	$	,	#						¤	¥	
-C					<	*	%	@						±		∨
-D					(	)	_	'					¢		≤	µ
-E					+	;	>	=						⌴	≥	•
-F					|	¬	?	"						¥	Ω	

Figure 8. (EBCDIC) US English Character Set (1). Font 2 (Complex Roman)



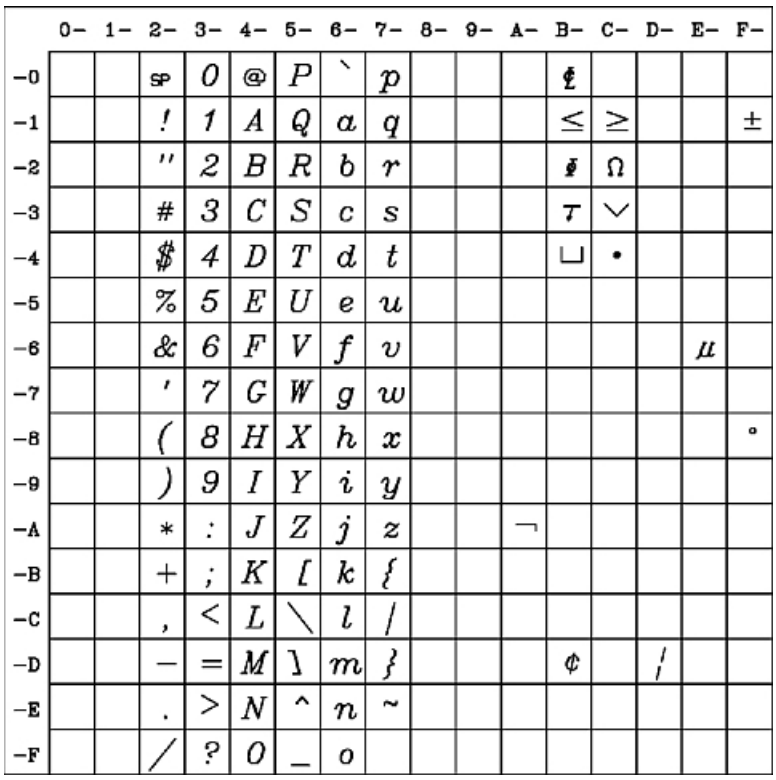
	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0			SP	0	@	P	`	p				¢				
-1			!	1	A	Q	a	q				≤	≥			±
-2			"	2	B	R	b	r				ß	Ω			
-3			#	3	C	S	c	s				┬	ˇ			
-4			$	4	D	T	d	t				␣	•			
-5			%	5	E	U	e	u								
-6			&	6	F	V	f	v							μ	
-7			'	7	G	W	g	w								
-8			(	8	H	X	h	x								°
-9			)	9	I	Y	i	y								
-A			*	:	J	Z	j	z			¬					
-B			+	;	K	[	k	{								
-C			,	<	L	\	l	|								
-D			−	=	M	]	m	}				¢		¦		
-E			.	>	N	^	n	~								
-F			/	?	O	_	o									

Figure 9. (ASCII) US English Character Set (1). Font 2 (Complex Roman)



	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0					SP	&	-						{	}	\	0
-1							/		a	j	~		A	J		1
-2									b	k	s		B	K	S	2
-3									c	l	t		C	L	T	3
-4									d	m	u		D	M	U	4
-5									e	n	v		E	N	V	5
-6									f	o	w		F	O	W	6
-7									g	p	x		G	P	X	7
-8									h	q	y		H	Q	Y	8
-9								`	i	r	z		I	R	Z	9
-A					¢	!	¦	:								
-B					.	$	,	#						°	¤	
-C					<	*	%	@						±		∨
-D					(	)	_	'					¢		≤	μ
-E					+	;	>	=						⊔	≥	•
-F					/	¬	?	"						↑	Ω	

Figure 10. (EBCDIC) US English Character Set (1). Font 3 (Complex Italic)



	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0			SP	0	@	P	`	p				¢				
-1			!	1	A	Q	a	q				≤	≥			±
-2			"	2	B	R	b	r				¤	Ω			
-3			#	3	C	S	c	s				↑	∨			
-4			$	4	D	T	d	t				⊔	•			
-5			%	5	E	U	e	u								
-6			&	6	F	V	f	v							μ	
-7			'	7	G	W	g	w								
-8			(	8	H	X	h	x								°
-9			)	9	I	Y	i	y								
-A			*	:	J	Z	j	z			¬					
-B			+	;	K	[	k	{								
-C			,	<	L	\	l	|								
-D			-	=	M	]	m	}				¦		/		
-E			.	>	N	^	n	~								
-F			/	?	O	_	o									

Figure 11. (ASCII) US English Character Set (1). Font 3 (Complex Italic)

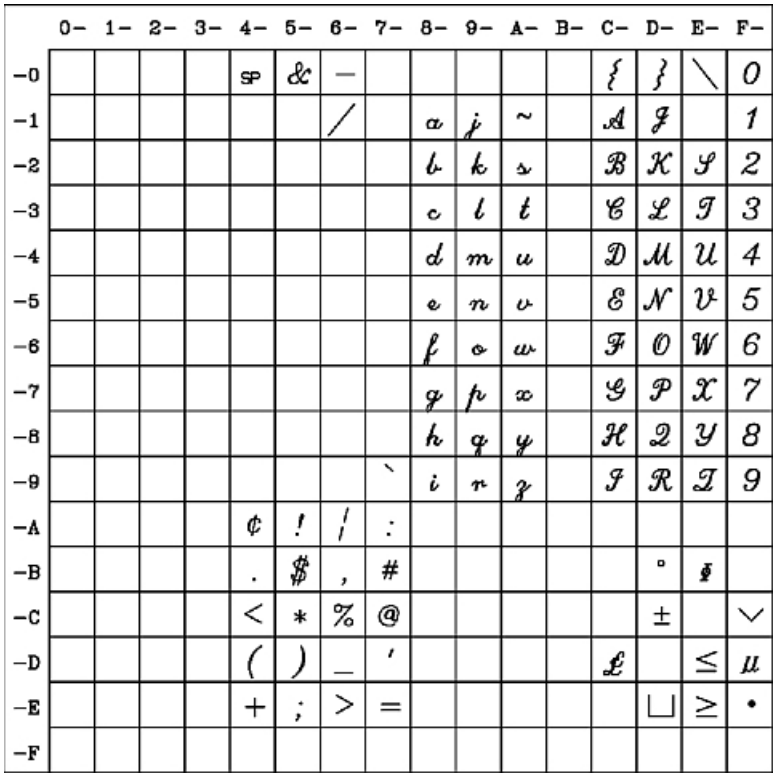


Figure 12. (EBCDIC) US English Character Set (1). Font 4 (Complex Script)

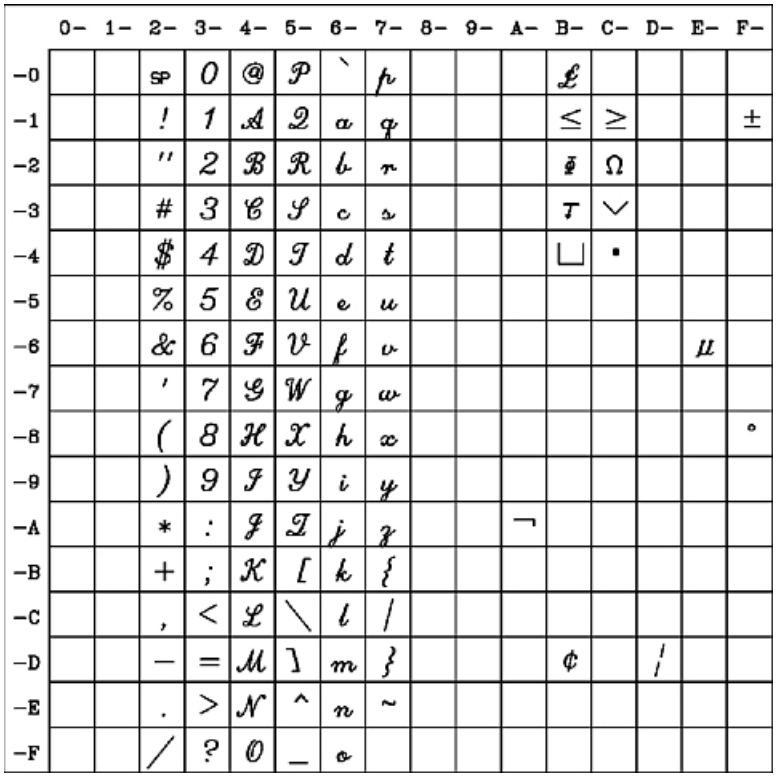


Figure 13. (ASCII) US English Character Set (1). Font 4 (Complex Script)

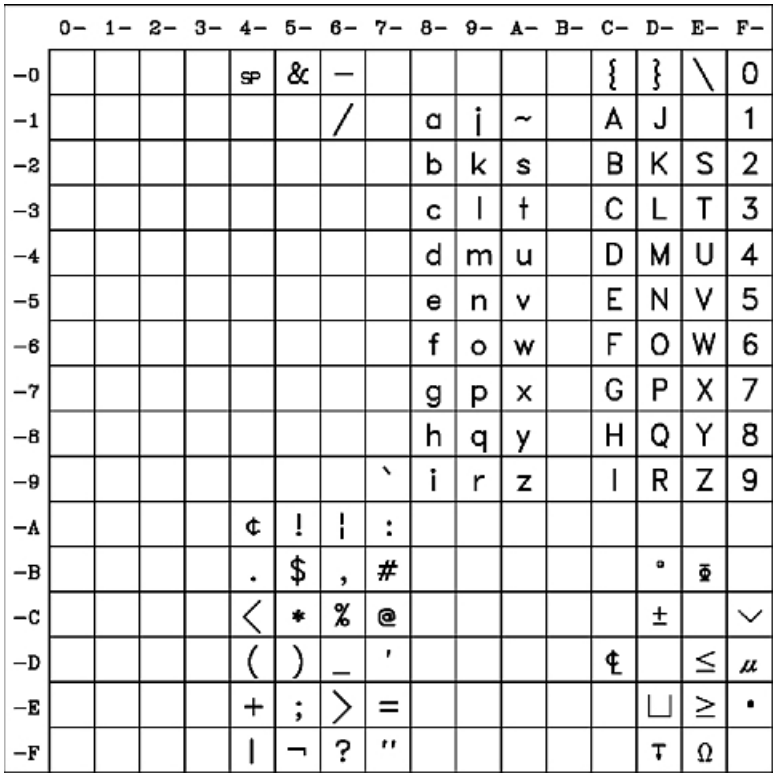


Figure 14. (EBCDIC) US English Character Set (1). Font 5 (Duplex Roman)

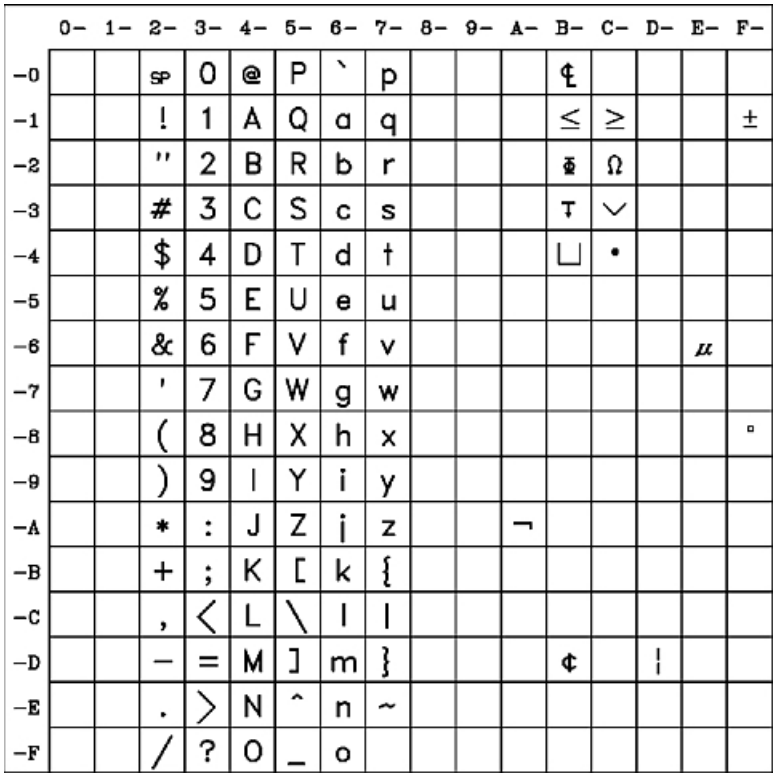


Figure 15. (ASCII) US English Character Set (1). Font 5 (Duplex Roman)

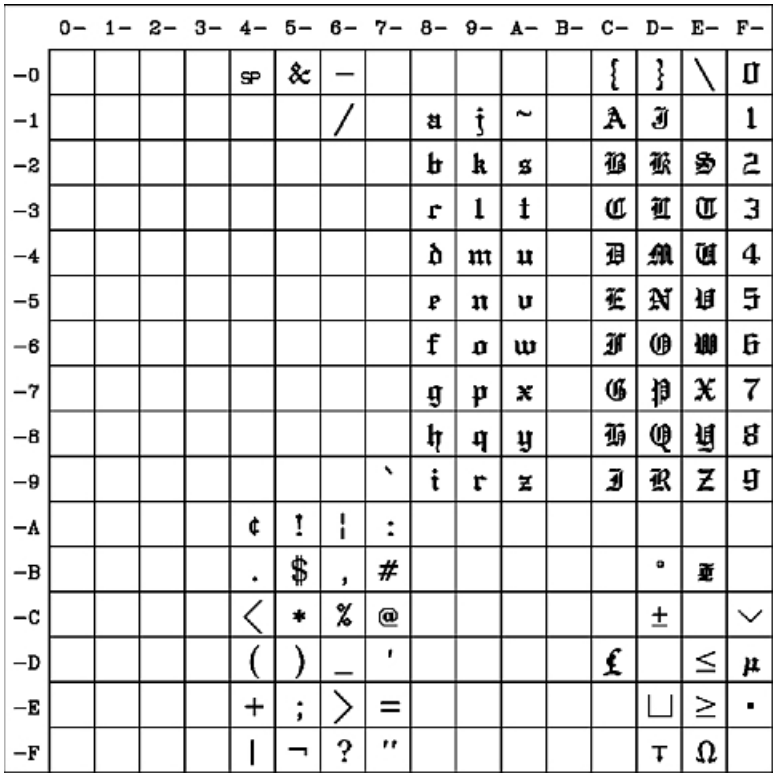


Figure 16. (EBCDIC) US English Character Set (1). Font 6 (Gothic English)

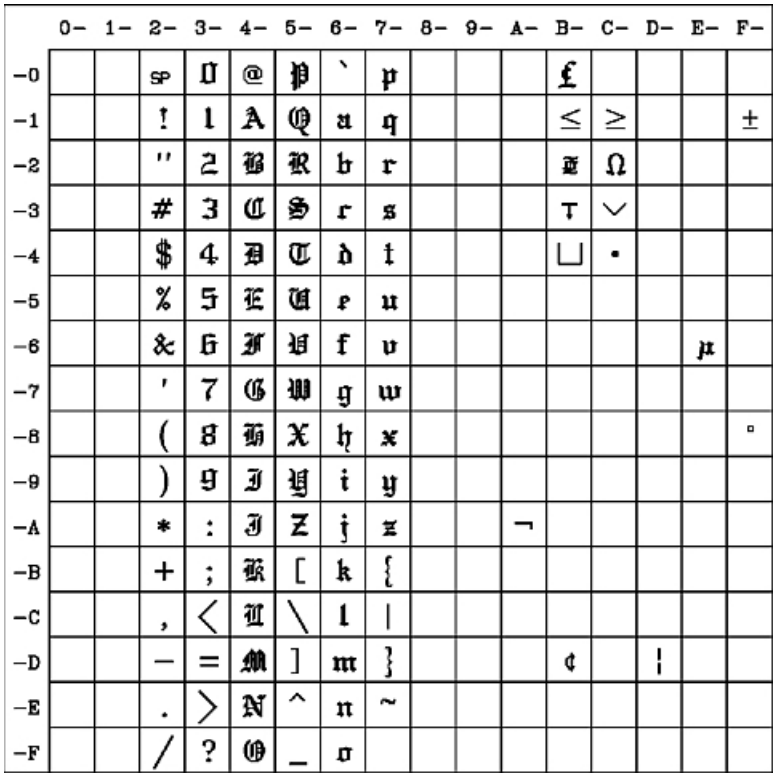


Figure 17. (ASCII) US English Character Set (1). Font 6 (Gothic English)

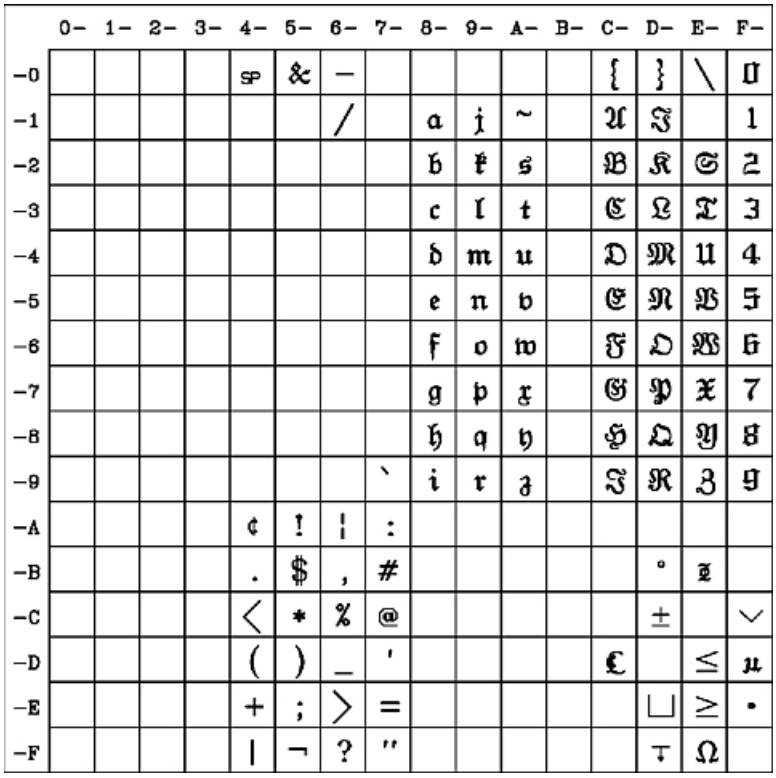


Figure 18. (EBCDIC) US English Character Set (1). Font 7 (Gothic German).

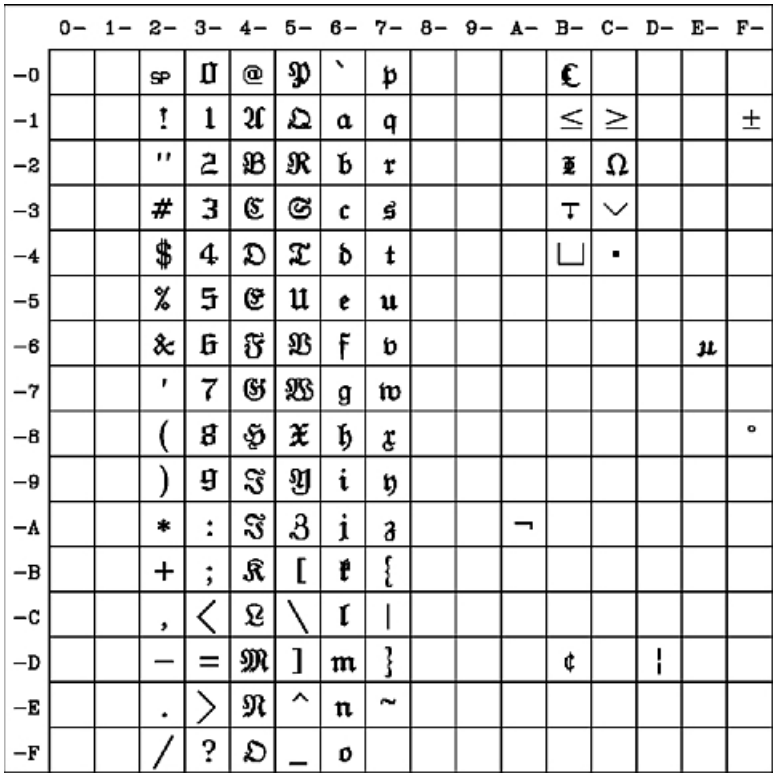


Figure 19. (ASCII) US English Character Set (1). Font 7 (Gothic German)

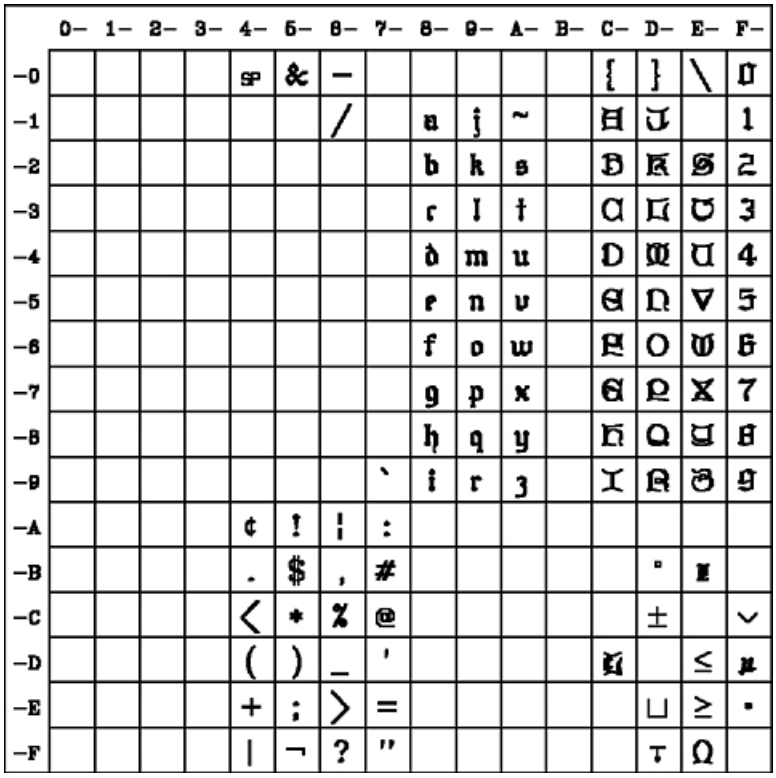
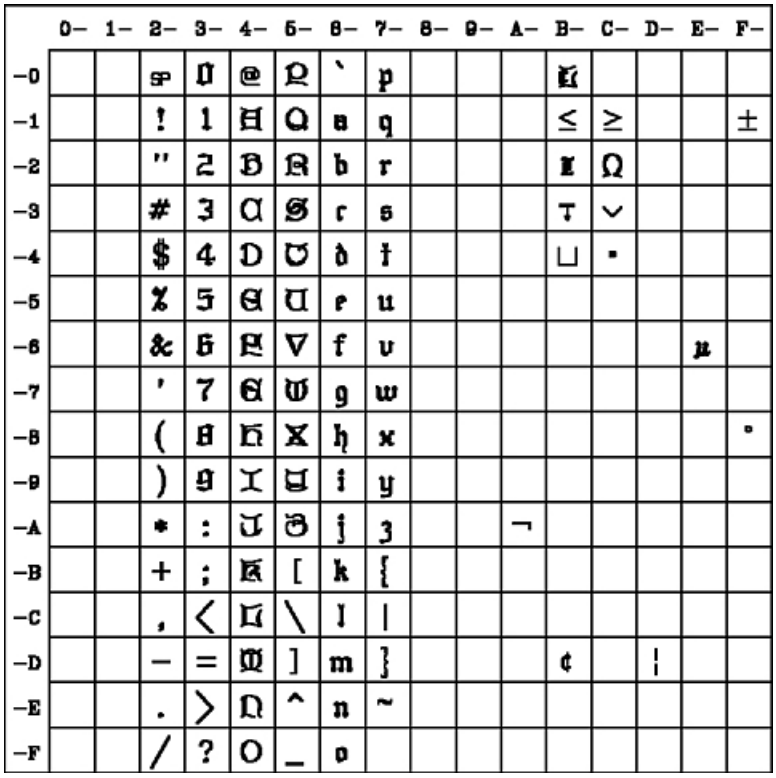
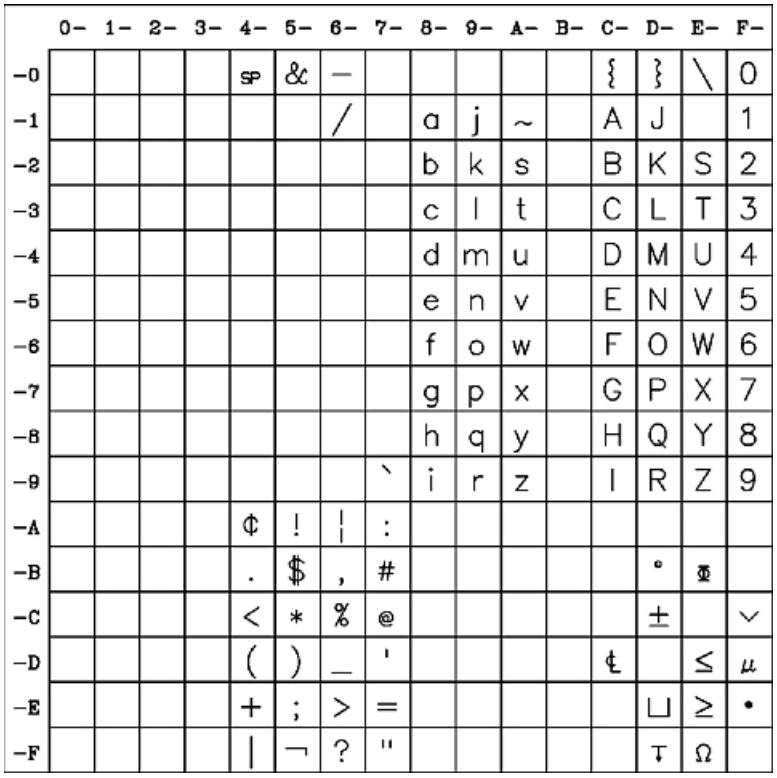


Figure 20. (EBCDIC) US English Character Set (1). Font 8 (Gothic Italic)



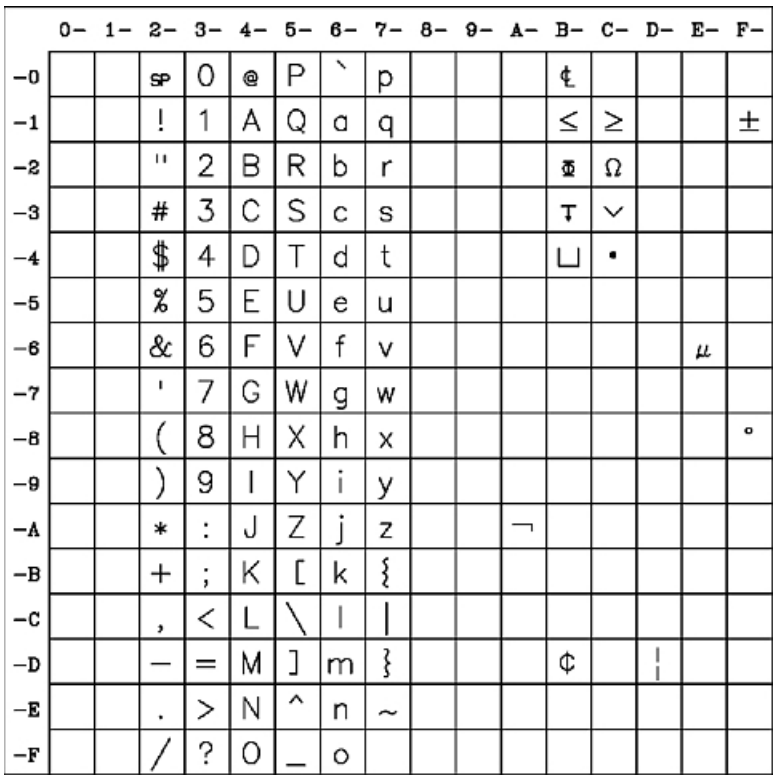
	0–	1–	2–	3–	4–	5–	6–	7–	8–	9–	A–	B–	C–	D–	E–	F–
–0																
–1			!									≤	≥			±
–2			"										Ω			
–3			#													
–4			$									⌴	▪			
–5			%													
–6			&												µ	
–7			'													
–8			(													°
–9			)													
–A			*	:							¬					
–B			+	;		[		{								
–C			,			\		|							
–D			–	=		]		}				¢		¦		
–E			.			^		~								
–F			/	?		_										

Figure 21. (ASCII) US English Character Set (1). Font 8 (Gothic Italic)



	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0					SP	&	–						{	}	\	0
-1							/		a	j	~		A	J		1
-2									b	k	s		B	K	S	2
-3									c	l	t		C	L	T	3
-4									d	m	u		D	M	U	4
-5									e	n	v		E	N	V	5
-6									f	o	w		F	O	W	6
-7									g	p	x		G	P	X	7
-8									h	q	y		H	Q	Y	8
-9								`	i	r	z		I	R	Z	9
-A					¢	!	¦	:								
-B					.	$	,	#						°	Φ	
-C					<	*	%	@						±		∨
-D					(	)	_	'					₺		≤	μ
-E					+	;	>	=						⊔	≥	•
-F					|	¬	?	"						↧	Ω	

Figure 22. (EBCDIC) US English Character Set (1). Font 9 (Simplex Roman)



	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0			SP	0	@	P	`	p				₺				
-1			!	1	A	Q	a	q				≤	≥			±
-2			"	2	B	R	b	r				Φ	Ω			
-3			#	3	C	S	c	s				↧	∨			
-4			$	4	D	T	d	t				⊔	•			
-5			%	5	E	U	e	u								
-6			&	6	F	V	f	v							μ	
-7			'	7	G	W	g	w								
-8			(	8	H	X	h	x								°
-9			)	9	I	Y	i	y								
-A			*	:	J	Z	j	z			¬					
-B			+	;	K	[	k	{								
-C			,	<	L	\	l	|								
-D			–	=	M	]	m	}				¢		¦		
-E			.	>	N	^	n	~								
-F			/	?	O	_	o									

Figure 23. (ASCII) US English Character Set (1). Font 9 (Simplex Roman)

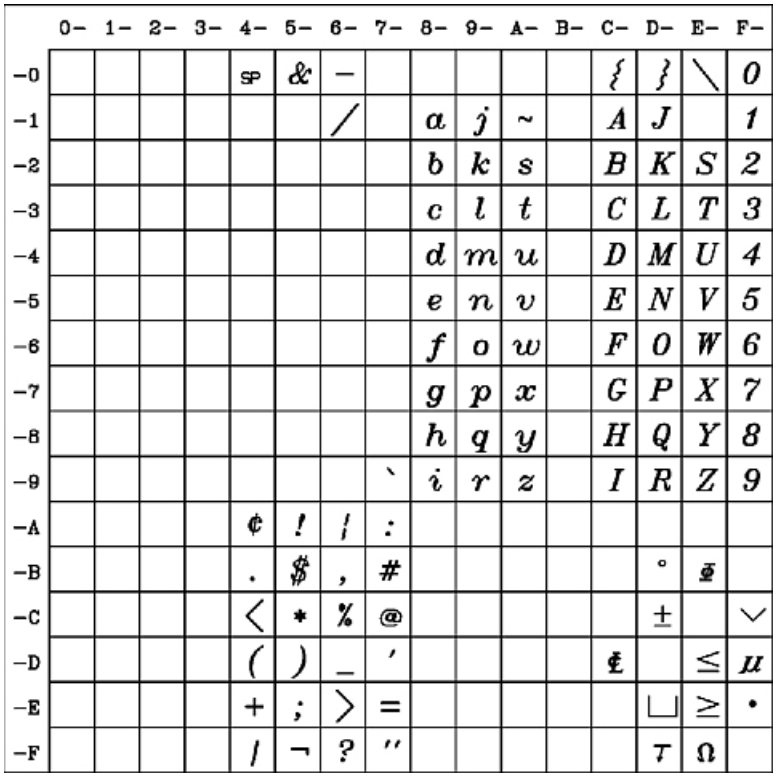
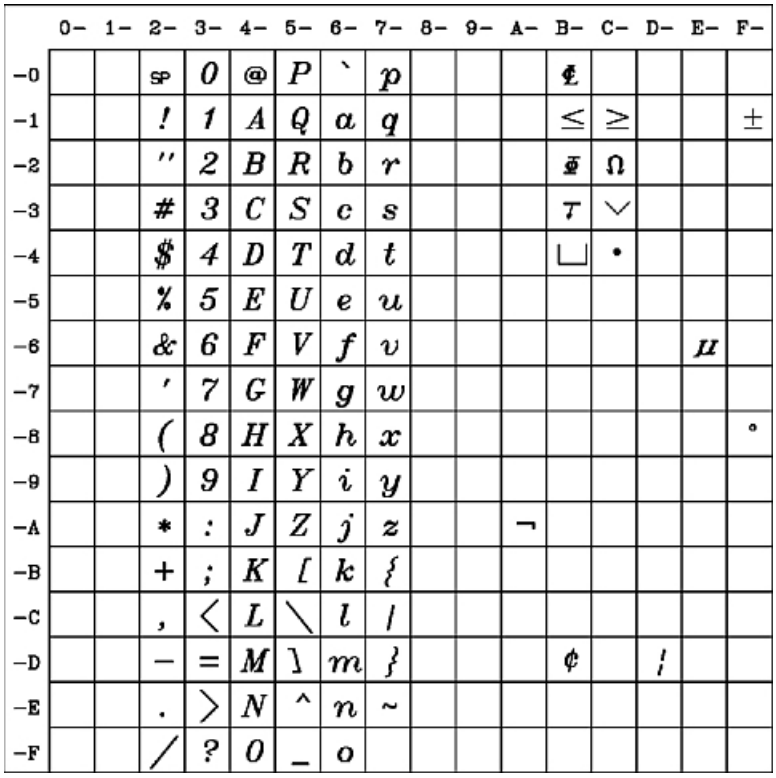
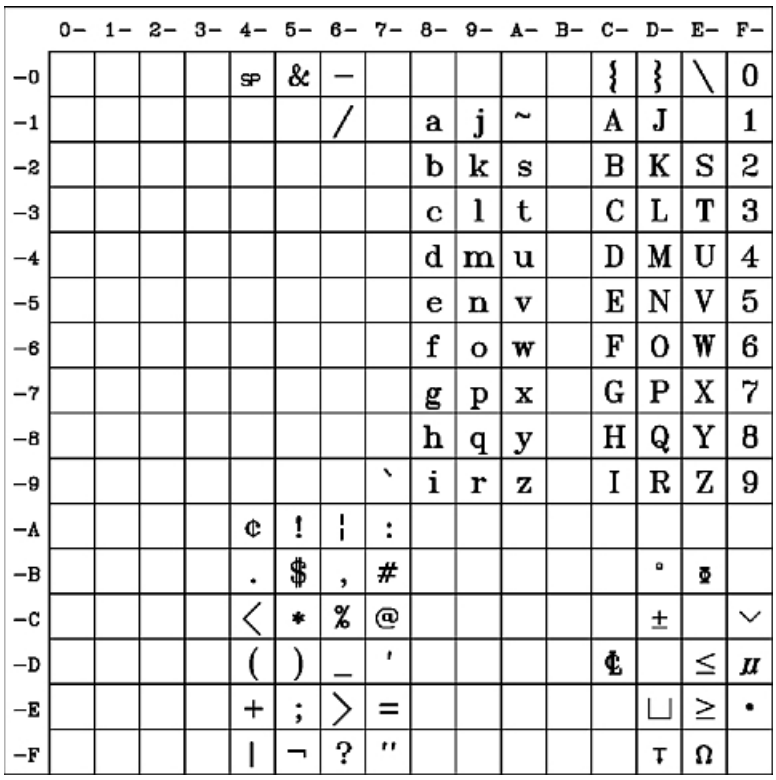


Figure 24. (EBCDIC) US English Character Set (1). Font 10 (Triplex Italic)



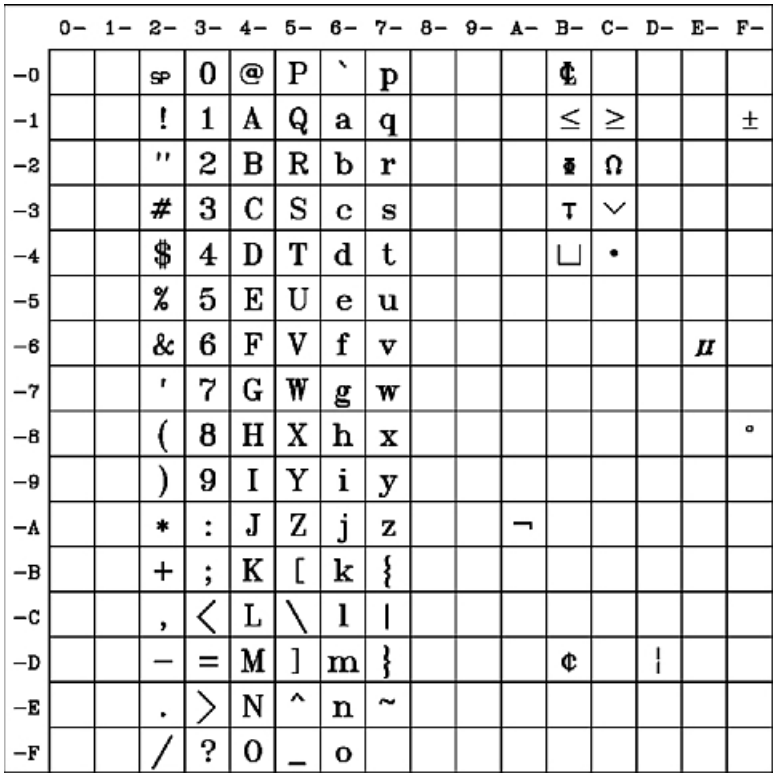
	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0			SP	0	@	P	`	p				¢				
-1			!	1	A	Q	a	q				≤	≥			±
-2			"	2	B	R	b	r				¯	Ω			
-3			#	3	C	S	c	s				⊤	∨			
-4			$	4	D	T	d	t				⌴	•			
-5			%	5	E	U	e	u								
-6			&	6	F	V	f	v							μ	
-7			'	7	G	W	g	w								
-8			(	8	H	X	h	x								°
-9			)	9	I	Y	i	y								
-A			*	:	J	Z	j	z			¬					
-B			+	;	K	[	k	{								
-C			,	<	L	\	l	|								
-D			−	=	M	]	m	}				¢		¡		
-E			.	>	N	^	n	~								
-F			/	?	O	_	o									

Figure 25. (ASCII) US English Character Set (1). Font 10 (Triplex Italic)



	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0					SP	&	−						{	}	\	0
-1							/		a	j	~		A	J		1
-2									b	k	s		B	K	S	2
-3									c	l	t		C	L	T	3
-4									d	m	u		D	M	U	4
-5									e	n	v		E	N	V	5
-6									f	o	w		F	O	W	6
-7									g	p	x		G	P	X	7
-8									h	q	y		H	Q	Y	8
-9								`	i	r	z		I	R	Z	9
-A					¢	!	¦	:								
-B					.	$	,	#						°	¯	
-C					<	*	%	@						±		∨
-D					(	)	_	'					¢		≤	μ
-E					+	;	>	=						⌴	≥	•
-F					|	¬	?	"						⊤	Ω	

Figure 26. (EBCDIC) US English Character Set (1). Font 11 (Triplex Roman)



	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0			SP	0	@	P	`	p				¢				
-1			!	1	A	Q	a	q				≤	≥			±
-2			"	2	B	R	b	r				ß	Ω			
-3			#	3	C	S	c	s				τ	ˇ			
-4			$	4	D	T	d	t				␣	•			
-5			%	5	E	U	e	u								
-6			&	6	F	V	f	v							µ	
-7			'	7	G	W	g	w								
-8			(	8	H	X	h	x								°
-9			)	9	I	Y	i	y								
-A			*	:	J	Z	j	z			¬					
-B			+	;	K	[	k	{								
-C			,	<	L	\	l	|								
-D			−	=	M	]	m	}				¢		¦		
-E			.	>	N	^	n	~								
-F			/	?	O	_	o									

Figure 27. (ASCII) US English Character Set (1). Font 11 (Triplex Roman)

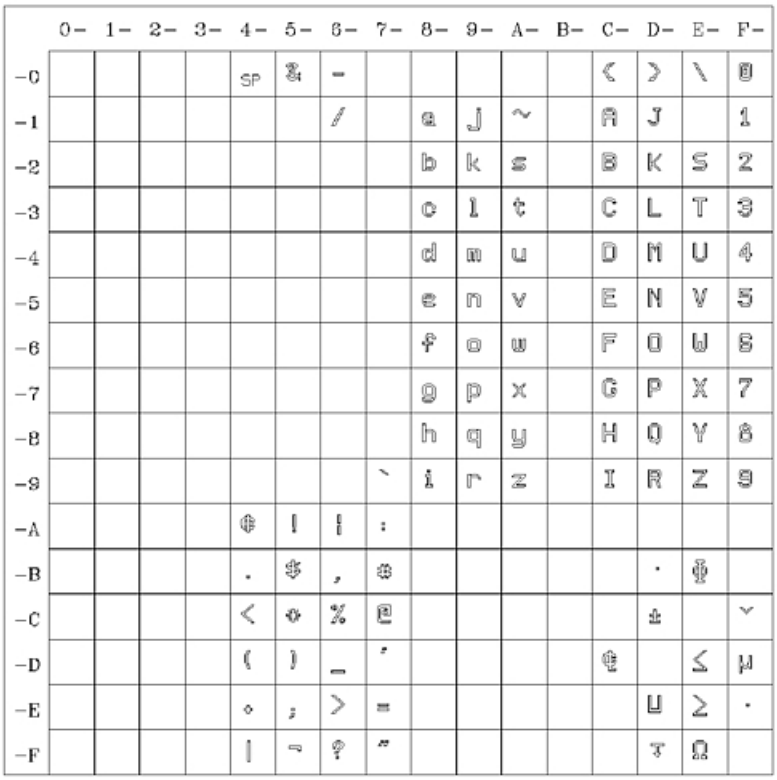


Figure 28. (EBCDIC) US English Character Set (1), Font 12 (Filled), Font 13 (Proportional), Font 14 (Filled-Proportional)

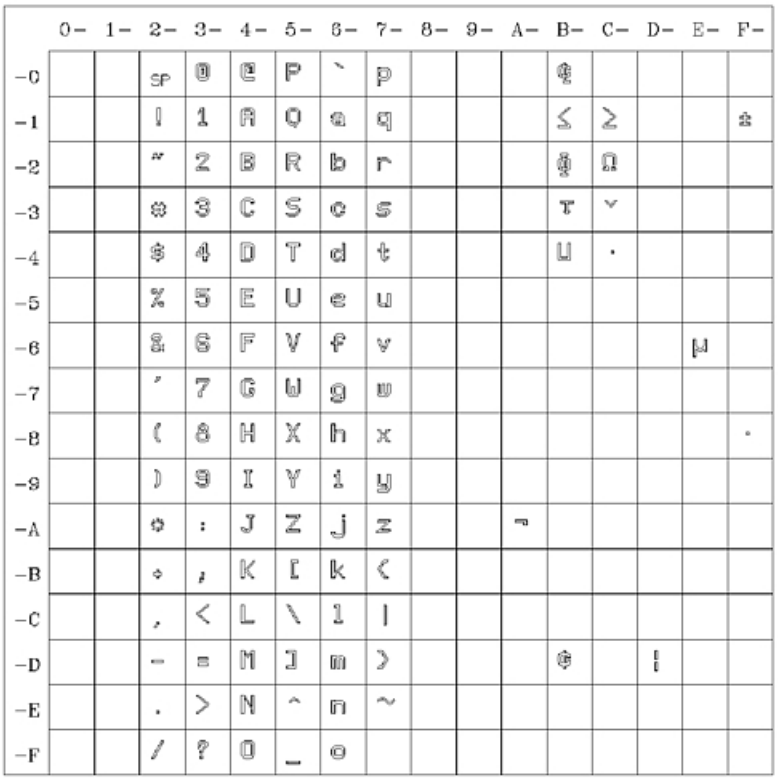


Figure 29. (ASCII) US English Character Set (1), Font 12 (Filled), Font 13 (Proportional), Font 14 (Filled-Proportional)

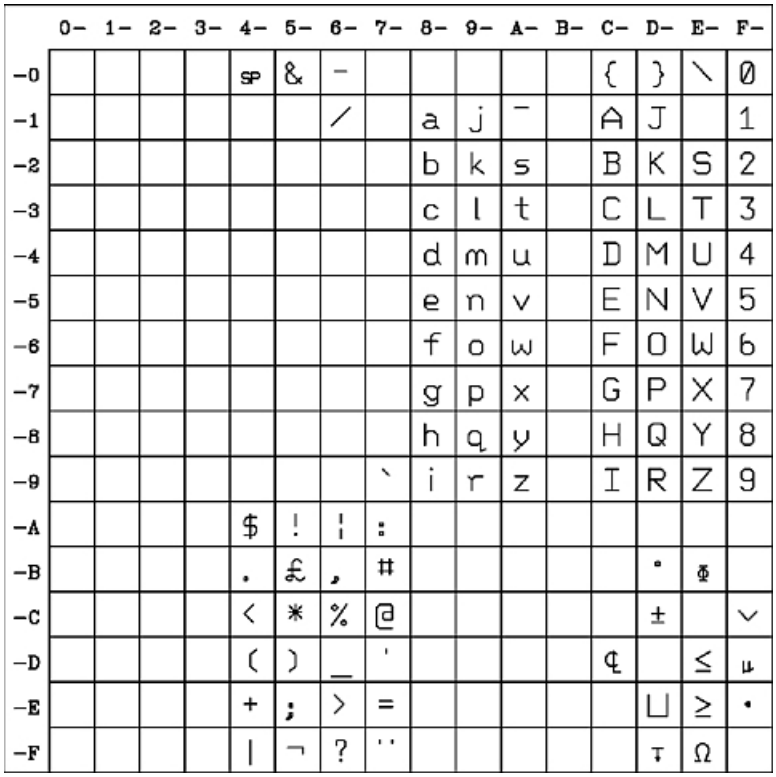


Figure 30. (EBCDIC) UK English Character Set (2). Font 1 (Primary)

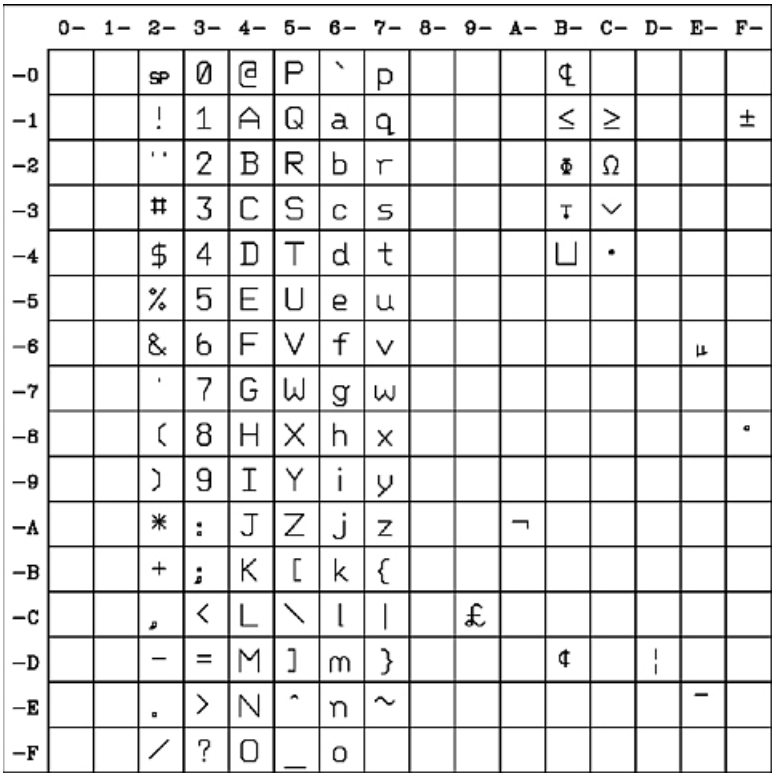
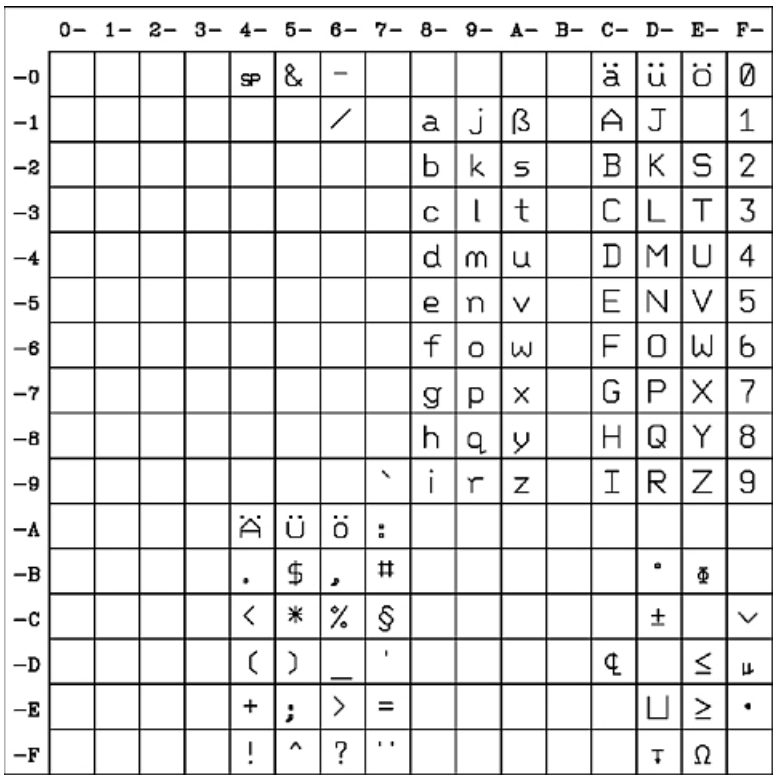
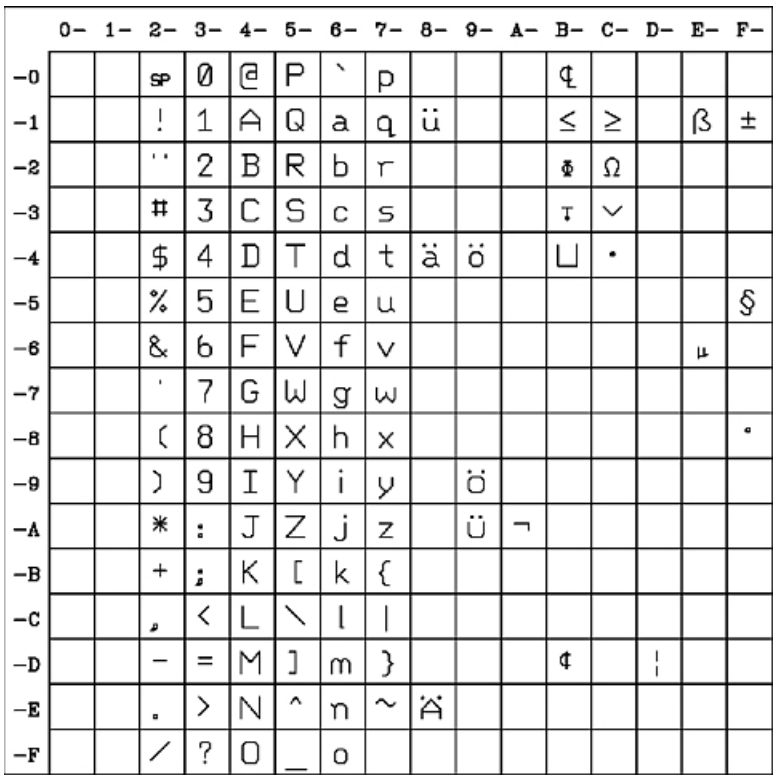


Figure 31. (ASCII) UK English Character Set (2). Font 1 (Primary).



	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0					SP	&	-						ä	ü	ö	Ø
-1							/		a	j	ß		A	J		1
-2									b	k	s		B	K	S	2
-3									c	l	t		C	L	T	3
-4									d	m	u		D	M	U	4
-5									e	n	v		E	N	V	5
-6									f	o	w		F	O	W	6
-7									g	p	x		G	P	X	7
-8									h	q	y		H	Q	Y	8
-9								`	i	r	z		I	R	Z	9
-A					Ä	Ü	ö	:								
-B					.	$	,	#						°	Φ	
-C					<	*	%	§						±		∨
-D					(	)	_	'					¢		≤	µ
-E					+	;	>	=						⊔	≥	•
-F					!	^	?	¨						↧	Ω	

Figure 32. (EBCDIC) German Character Set (3). Font 1 (Primary)



	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0			SP	0	@	P	`	p				¢				
-1			!	1	A	Q	a	q	ü			≤	≥		ß	±
-2			¨	2	B	R	b	r				Φ	Ω			
-3			#	3	C	S	c	s				↧	∨			
-4			$	4	D	T	d	t	ä	ö		⊔	•			
-5			%	5	E	U	e	u								§
-6			&	6	F	V	f	v							µ	
-7			'	7	G	W	g	w								
-8			(	8	H	X	h	x								°
-9			)	9	I	Y	i	y		ö						
-A			*	:	J	Z	j	z		ü	¬					
-B			+	;	K	[	k	{								
-C			,	<	L	\	l	|								
-D			-	=	M	]	m	}				¢		¦		
-E			.	>	N	^	n	~	Ä							
-F			/	?	O	_	o									

Figure 33. (ASCII) German Character Set (3). Font 1 (Primary)

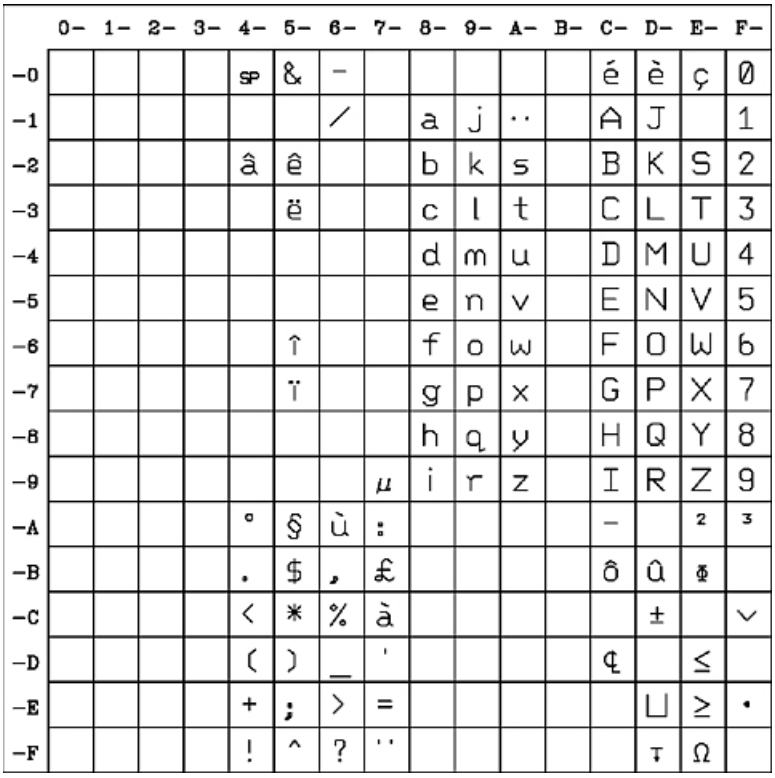
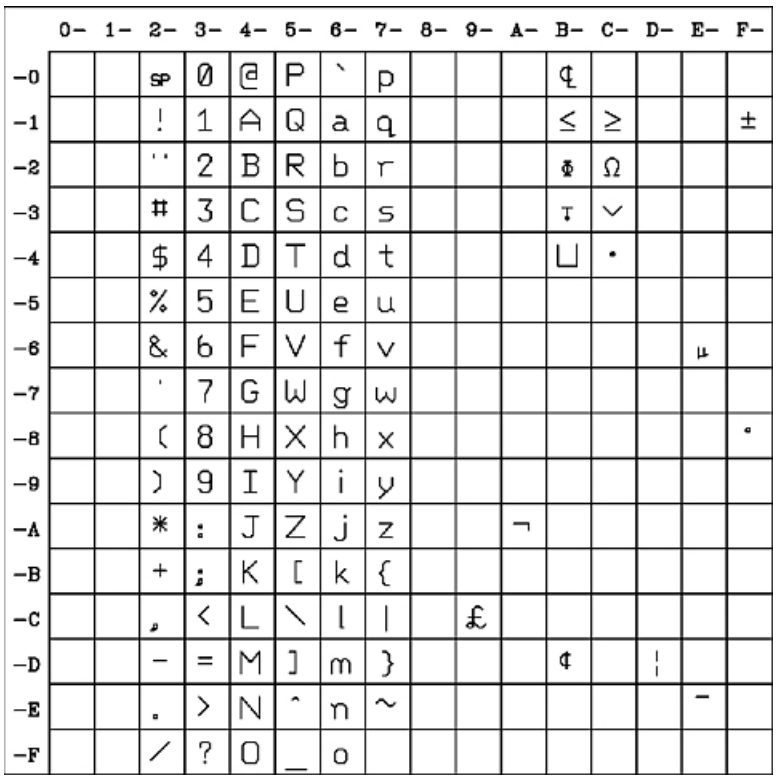


Figure 34. (EBCDIC) French Character Set (4). Font 1 (Primary)

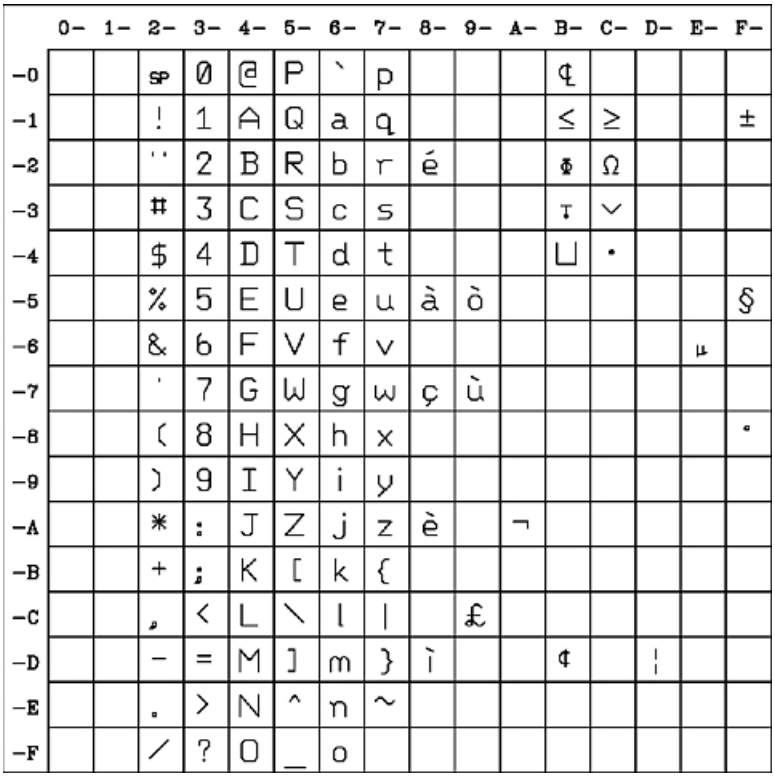


	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0			SP	0	@	P	`	p				¢				
-1			!	1	A	Q	a	q				≤	≥			±
-2			"	2	B	R	b	r				Φ	Ω			
-3			#	3	C	S	c	s				↧	∨			
-4			$	4	D	T	d	t				␣	•			
-5			%	5	E	U	e	u								
-6			&	6	F	V	f	v							µ	
-7			'	7	G	W	g	w								
-8			(	8	H	X	h	x								°
-9			)	9	I	Y	i	y								
-A			*	:	J	Z	j	z			¬					
-B			+	;	K	[	k	{								
-C			,	<	L	\	l	|		£						
-D			-	=	M	]	m	}				¢		¦		
-E			.	>	N	^	n	~							-	
-F			/	?	O	_	o									

Figure 35. (ASCII) French Character Set (4). Font 1 (Primary)

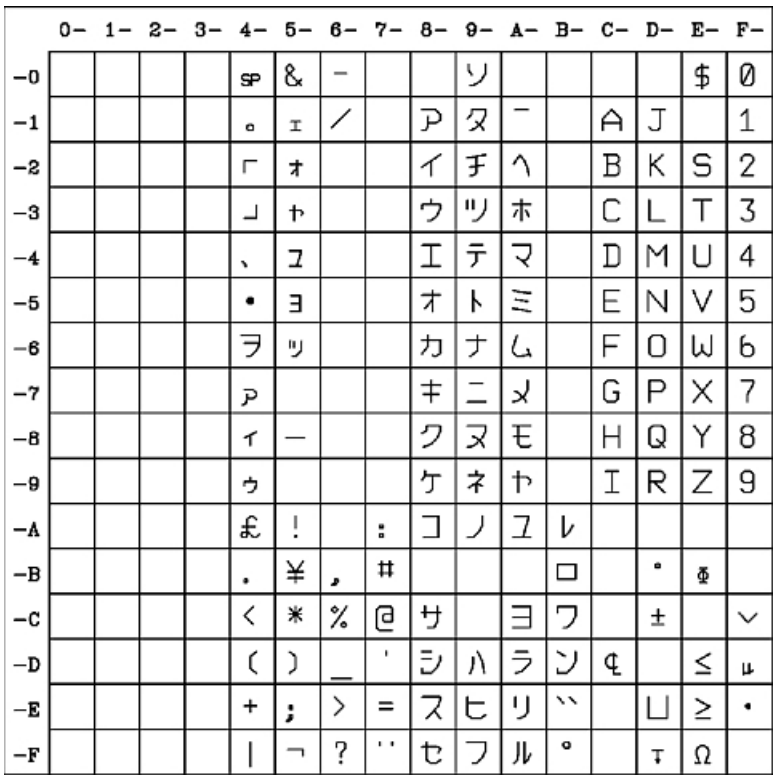
	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0					SP	&	-						à	è	ç	0
-1							/		a	j	ì		A	J		1
-2									b	k	s		B	K	S	2
-3									c	l	t		C	L	T	3
-4									d	m	u		D	M	U	4
-5									e	n	v		E	N	V	5
-6									f	o	w		F	O	W	6
-7									g	p	x		G	P	X	7
-8									h	q	y		H	Q	Y	8
-9								ù	i	r	z		I	R	Z	9
-A					°	é	ò	:								
-B					.	$	,	£						°	Φ	
-C					<	*	%	§						±		∨
-D					(	)	_	'					¢		≤	µ
-E					+	;	>	=						␣	≥	•
-F					!	^	?	"						↧	Ω	

Figure 36. (EBCDIC) Italian Character Set (5). Font 1 (Primary)



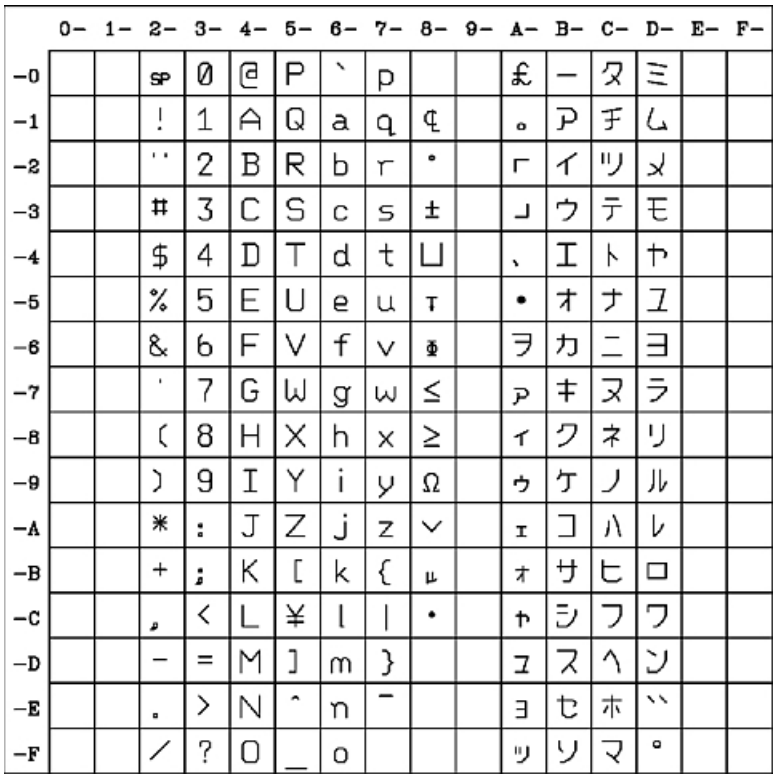
	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0			SP	0	@	P	`	p				¢				
-1			!	1	A	Q	a	q				≤	≥			±
-2			"	2	B	R	b	r	é			Φ	Ω			
-3			#	3	C	S	c	s				↧	∨			
-4			$	4	D	T	d	t				⌴	•			
-5			%	5	E	U	e	u	à	ò						§
-6			&	6	F	V	f	v							µ	
-7			'	7	G	W	g	w	ç	ù						
-8			(	8	H	X	h	x								°
-9			)	9	I	Y	i	y								
-A			*	:	J	Z	j	z	è		¬					
-B			+	;	K	[	k	{								
-C			,	<	L	\	l	|		£						
-D			-	=	M	]	m	}	ì			¢		¦		
-E			.	>	N	^	n	~								
-F			/	?	O	_	o									

Figure 37. (ASCII) Italian Character Set (5). Font 1 (Primary)



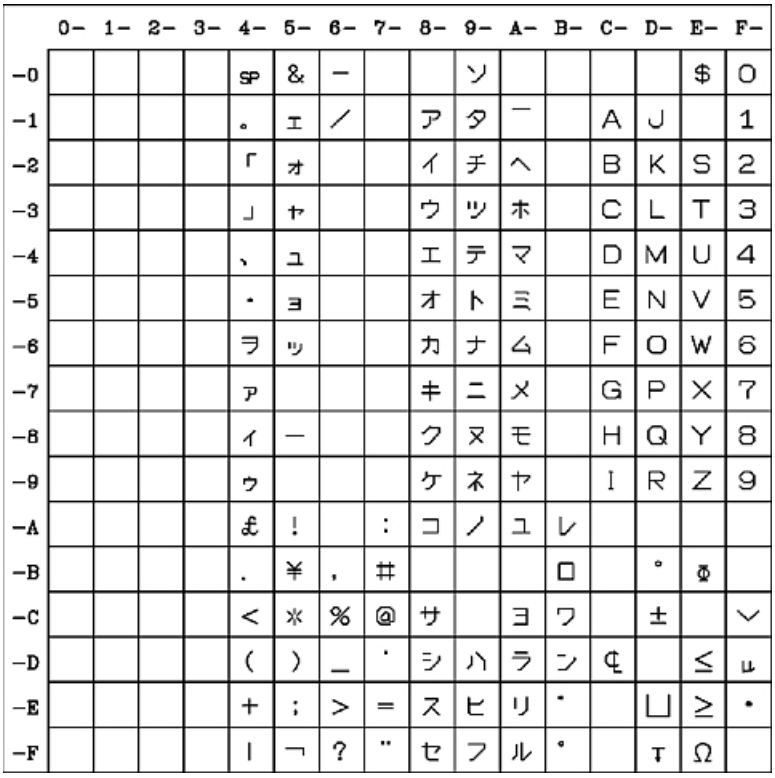
	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0					SP	&	-			ｿ					$	0
-1					｡	ｱ	/		ﾀ	ﾀﾞ	¯		ｱ	J		1
-2					｢	ｵ			ｲ	ﾁ	ﾍ		B	K	S	2
-3					｣	ｶ			ｳ	ﾂ	ﾎ		C	L	T	3
-4					、	ｭ			ｴ	ﾃ	ﾏ		D	M	U	4
-5					•	ｮ			ｵ	ﾄ	ﾐ		E	N	V	5
-6					ｦ	ｯ			ｶ	ﾅ	ﾑ		F	O	W	6
-7					ｧ				ｷ	ﾆ	ﾒ		G	P	X	7
-8					ｨ	ｰ			ｸ	ﾇ	ﾓ		H	Q	Y	8
-9					ｩ				ｹ	ﾈ	ﾔ		I	R	Z	9
-A					£	!		:	ｺ	ﾉ	ｭ	ﾚ				
-B					.	¥	,	#				ﾛ		°	Φ	
-C					<	*	%	@	ｻ		ﾖ	ﾜ		±		ﾚ
-D					(	)	_	'	ｼ	ﾊ	ﾗ	ﾝ	¢		≤	μ
-E					+	;	>	=	ｽ	ﾋ	ﾘ	ﾞ		Ш	≥	•
-F					|	¬	?	¨	ｾ	ﾌ	ﾙ	°		ｰ	Ω	

Figure 38. (EBCDIC) Katakana Character Set (6). Font 1 (Primary).



	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0			SP	0	@	P	`	p			£	ー	ﾀ	ﾐ		
-1			!	1	A	Q	a	q	¢		｡	ｱ	ﾁ	ﾑ		
-2			¨	2	B	R	b	r	°		｢	ｲ	ﾂ	ﾒ		
-3			#	3	C	S	c	s	±		｣	ｳ	ﾃ	ﾓ		
-4			$	4	D	T	d	t	Ш		、	ｴ	ﾄ	ﾔ		
-5			%	5	E	U	e	u	ｰ		•	ｵ	ﾅ	ﾕ		
-6			&	6	F	V	f	v	Φ		ｦ	ｶ	ﾆ	ﾖ		
-7			'	7	G	W	g	w	≤		ｧ	ｷ	ﾇ	ﾗ		
-8			(	8	H	X	h	x	≥		ｨ	ｸ	ﾈ	ﾘ		
-9			)	9	I	Y	i	y	Ω		ｩ	ｹ	ﾉ	ﾙ		
-A			*	:	J	Z	j	z	ﾚ		ｪ	ｺ	ﾊ	ﾚ		
-B			+	;	K	[	k	{	μ		ｫ	ｻ	ﾋ	ﾛ		
-C			､	<	L	¥	l	|	•		ｬ	ｼ	ﾌ	ﾜ		
-D			-	=	M	]	m	}			ｭ	ｽ	ﾍ	ﾝ		
-E			.	>	N	^	n	~			ｮ	ｾ	ﾎ	ﾞ		
-F			/	?	O	_	o				ｯ	ｿ	ﾏ	°		

Figure 39. (ASCII) Katakana Character Set (6). Font 1 (Primary)



	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0					SP	&	−			ソ					$	0
-1					。	エ	／		ア	タ	¯		A	J		1
-2					「	オ			イ	チ	ヘ		B	K	S	2
-3					」	ャ			ウ	ツ	ホ		C	L	T	3
-4					、	ュ			エ	テ	マ		D	M	U	4
-5					・	ョ			オ	ト	ミ		E	N	V	5
-6					ヲ	ッ			カ	ナ	ム		F	O	W	6
-7					ァ				キ	ニ	メ		G	P	X	7
-8					ィ	ー			ク	ヌ	モ		H	Q	Y	8
-9					ゥ				ケ	ネ	ヤ		I	R	Z	9
-A					£	!		:	コ	ノ	ユ	レ				
-B					.	¥	,	#				□		°	¢	
-C					<	*	%	@	サ		ヨ	ワ		±		∨
-D					(	)	_	'	シ	ハ	ラ	ン	¢		≤	μ
-E					+	;	>	=	ス	ヒ	リ	゜		⌊⌋	≥	•
-F					|	¬	?	¨	セ	フ	ル	°		₸	Ω	

Figure 40. (EBCDIC) Katakana Character Set (6). Font 2

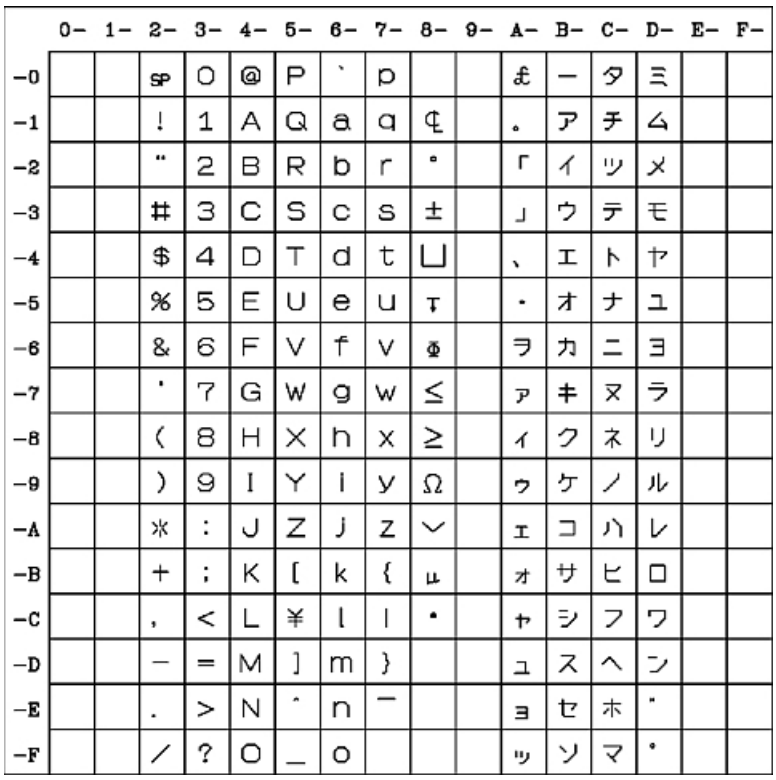


Figure 41. (ASCII) Katakana Character Set (6). Font 2

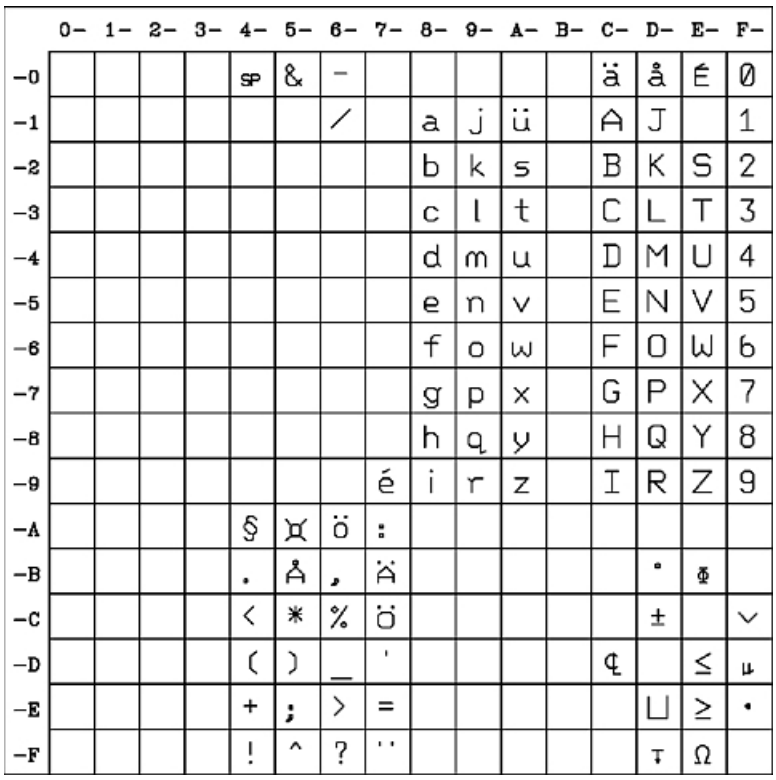
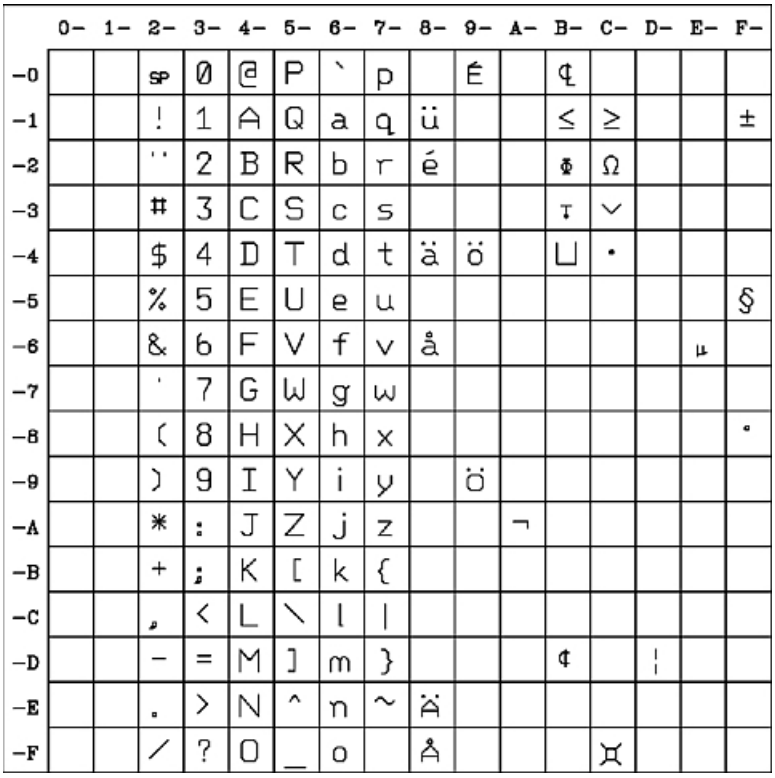


Figure 42. (EBCDIC) Swedish Character Set (7). Font 1 (Primary)



	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0			SP	0	@	P	`	p		É		¢				
-1			!	1	A	Q	a	q	ü			≤	≥			±
-2			"	2	B	R	b	r	é			Φ	Ω			
-3			#	3	C	S	c	s				↧	∨			
-4			$	4	D	T	d	t	ä	ö		⌴	•			
-5			%	5	E	U	e	u								§
-6			&	6	F	V	f	v	å						µ	
-7			'	7	G	W	g	w								
-8			(	8	H	X	h	x								°
-9			)	9	I	Y	i	y		Ö						
-A			*	:	J	Z	j	z			¬					
-B			+	;	K	[	k	{								
-C			,	<	L	\	l	|								
-D			-	=	M	]	m	}				¢		¦		
-E			.	>	N	^	n	~	Ä							
-F			/	?	O	_	o		Å				¤			

Figure 43. (ASCII) Swedish Character Set (7). Font 1 (Primary)

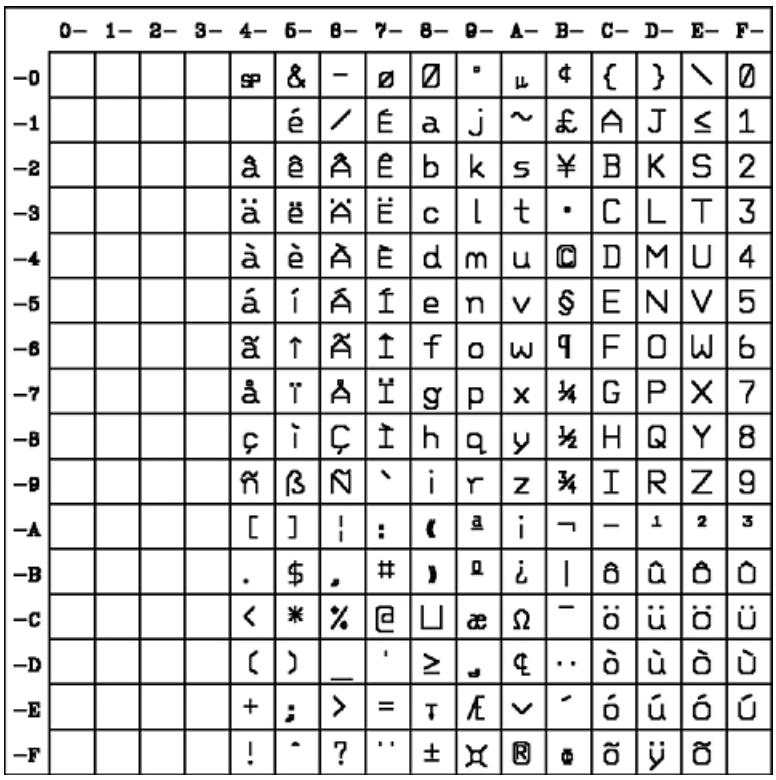


Figure 44. (EBCDIC) Multi-Language Character Set (8). Font 1 (Primary)

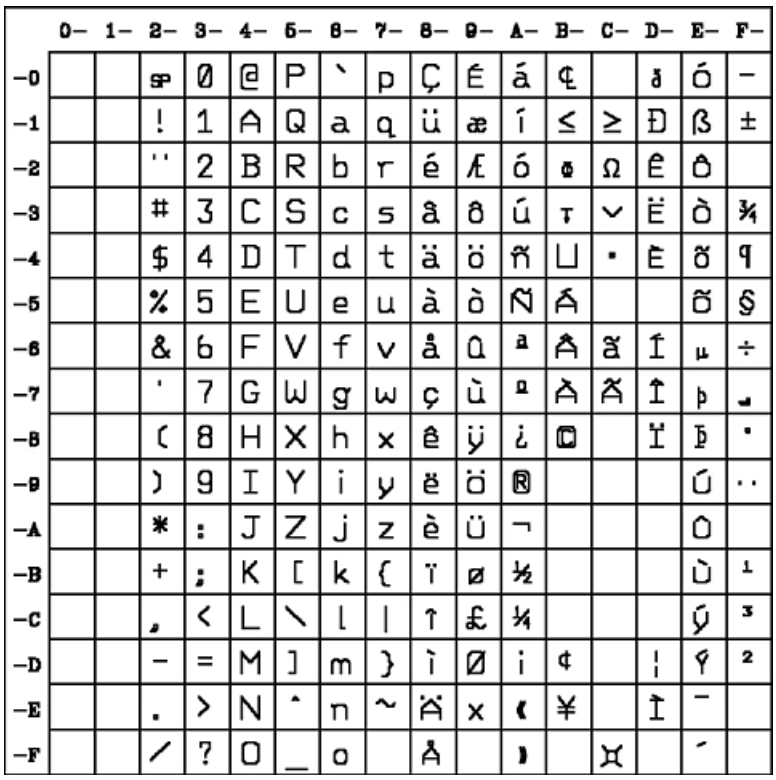


Figure 45. (ASCII) Multi-Language Character Set (8). Font 1 (Primary)

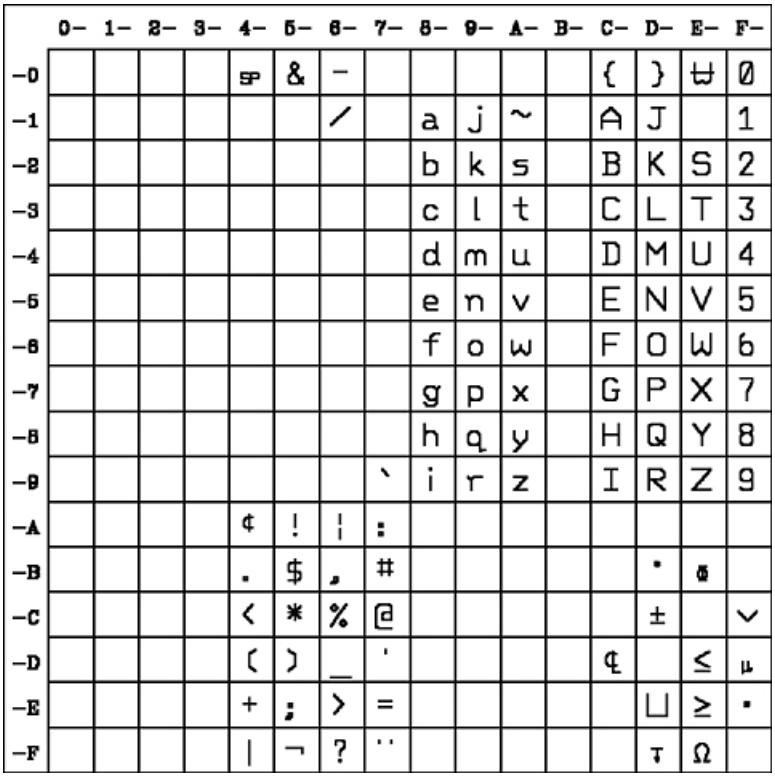


Figure 46. (EBCDIC) Single-Byte Korean (9). Font 1 (Primary)

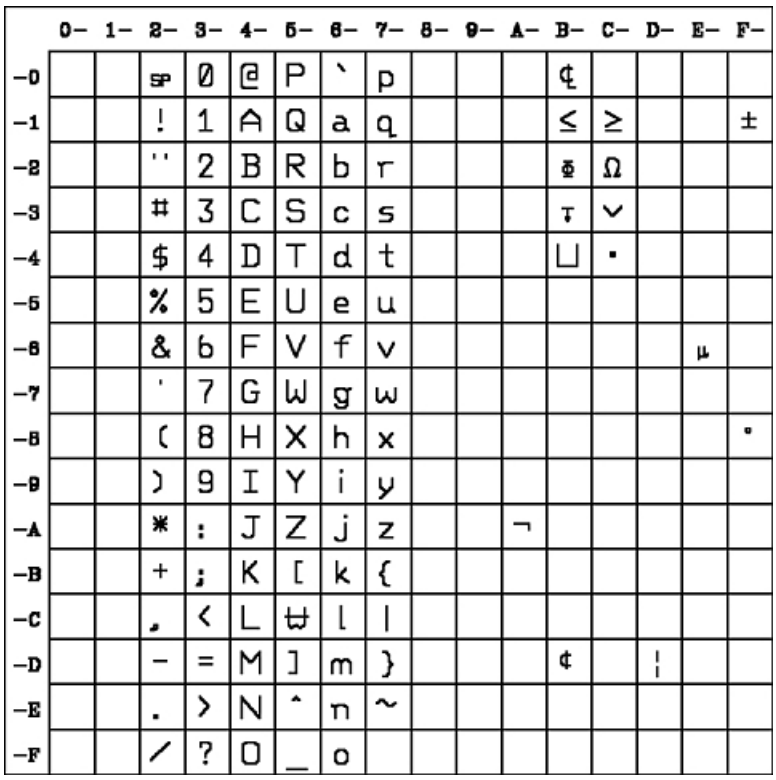


Figure 47. (ASCII) Single-Byte Korean (9). Font 1 (Primary)

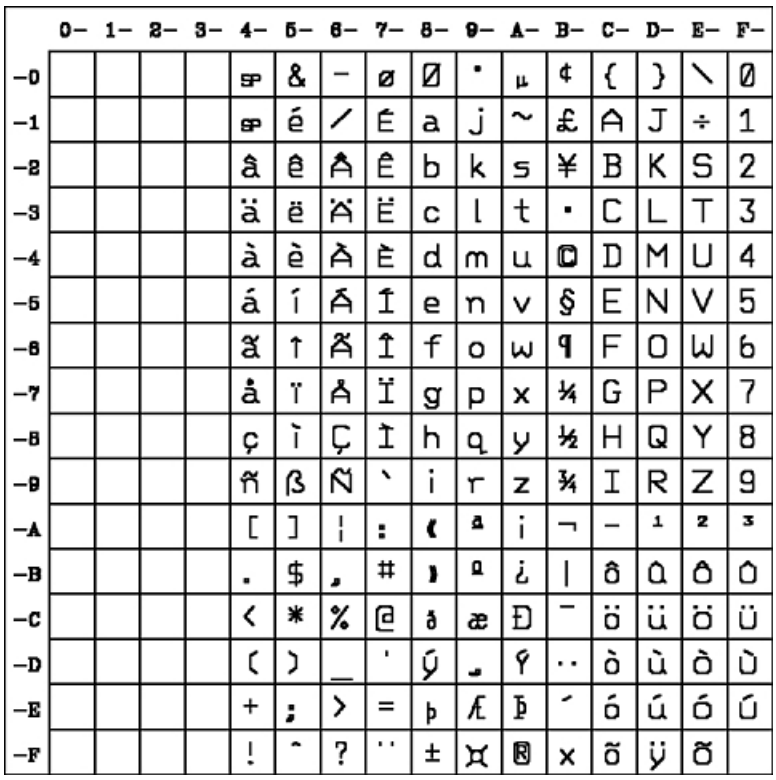
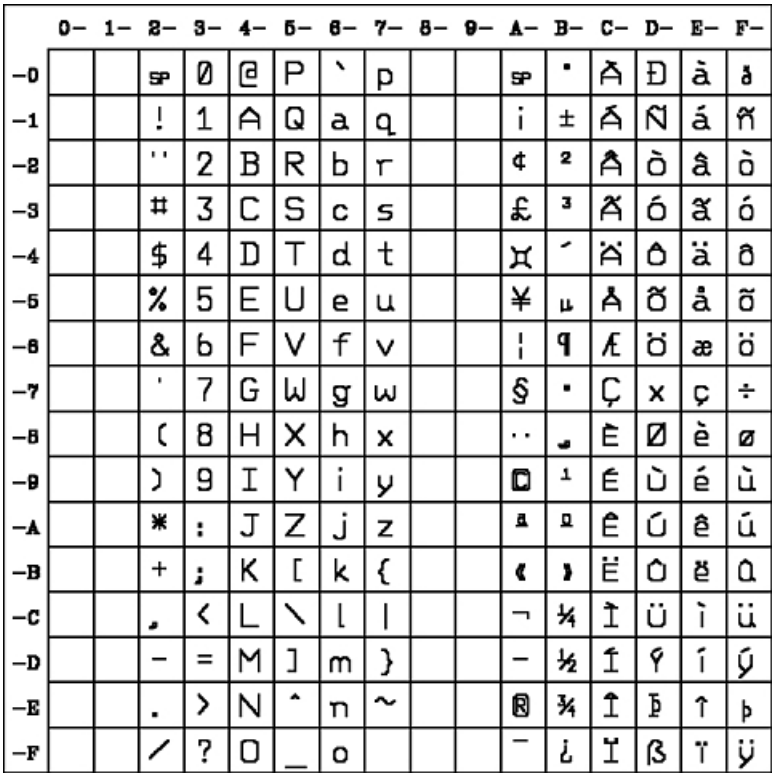
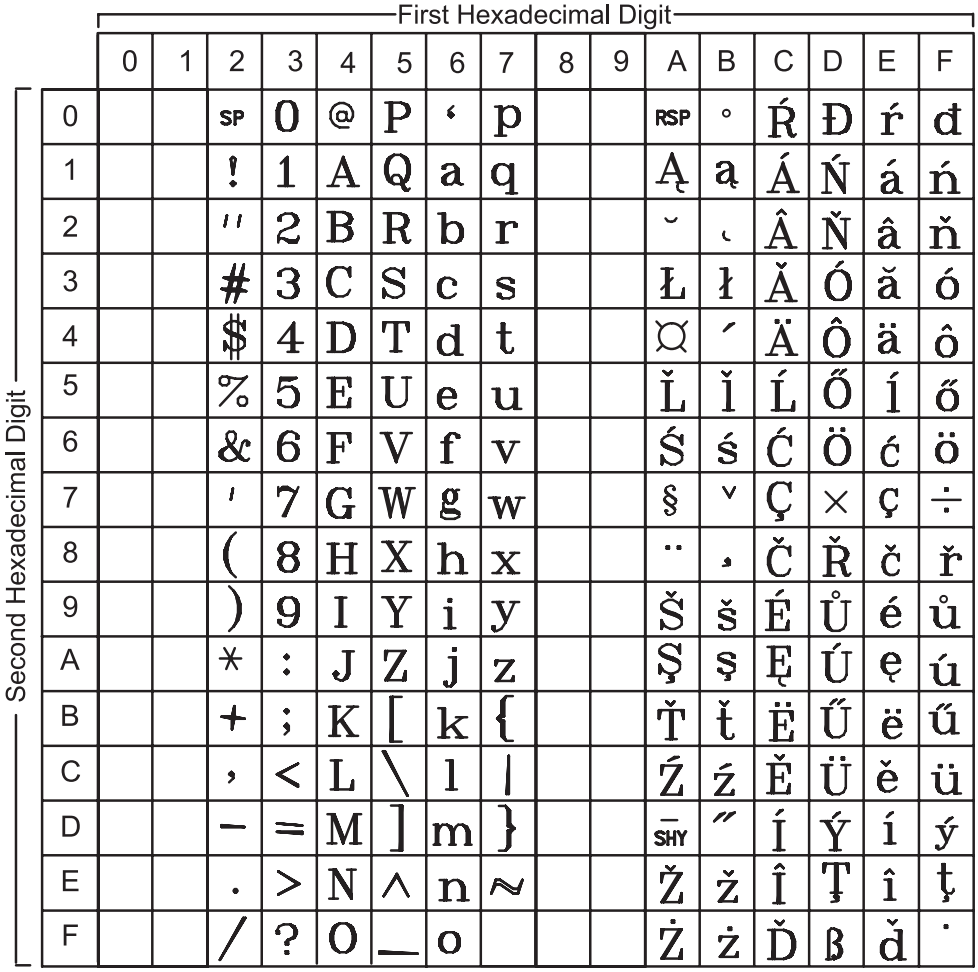


Figure 48. (EBCDIC) ISO 8859-1 (Latin 1) Character Set (10). Font 1 (Primary)



	0–	1–	2–	3–	4–	5–	6–	7–	8–	9–	A–	B–	C–	D–	E–	F–
–0			SP	0	@	P	`	p			SP	°	À	Ð	à	ð
–1			!	1	A	Q	a	q			¡	±	Á	Ñ	á	ñ
–2			"	2	B	R	b	r			¢	²	Â	Ò	â	ò
–3			#	3	C	S	c	s			£	³	Ã	Ó	ã	ó
–4			$	4	D	T	d	t			¤	´	Ä	Ô	ä	ô
–5			%	5	E	U	e	u			¥	µ	Å	Õ	å	õ
–6			&	6	F	V	f	v			¦	¶	Æ	Ö	æ	ö
–7			'	7	G	W	g	w			§	·	Ç	×	ç	÷
–8			(	8	H	X	h	x			¨	¸	È	Ø	è	ø
–9			)	9	I	Y	i	y			©	¹	É	Ù	é	ù
–A			*	:	J	Z	j	z			ª	º	Ê	Ú	ê	ú
–B			+	;	K	[	k	{			«	»	Ë	Û	ë	û
–C			,	<	L	\	l	|			¬	¼	Ì	Ü	ì	ü
–D			-	=	M	]	m	}			-	½	Í	Ý	í	ý
–E			.	>	N	^	n	~			®	¾	Î	Þ	î	þ
–F			/	?	O	_	o				¯	¿	Ï	ß	ï	ÿ

Figure 49. (ASCII) ISO 8859-1 (Latin 1) Character Set (10). Font 1 (Primary)



Second Hexadecimal Digit \ First Hexadecimal Digit	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0			SP	0	@	P	‘	p			RSP	°	Ŕ	Đ	ŕ	đ
1			!	1	A	Q	a	q			Ą	ą	Á	Ń	á	ń
2			"	2	B	R	b	r			˘	˛	Â	Ň	â	ň
3			#	3	C	S	c	s			Ł	ł	Ă	Ó	ă	ó
4			$	4	D	T	d	t			¤	´	Ä	Ô	ä	ô
5			%	5	E	U	e	u			Ľ	ľ	Ĺ	Ő	Í	ő
6			&	6	F	V	f	v			Ś	ś	Ć	Ö	ć	ö
7			'	7	G	W	g	w			§	ˇ	Ç	×	ç	÷
8			(	8	H	X	h	x			¨	¸	Č	Ř	č	ř
9			)	9	I	Y	i	y			Š	š	É	Ů	é	ů
A			*	:	J	Z	j	z			Ş	ş	Ę	Ú	ę	ú
B			+	;	K	[	k	{			Ť	ť	Ë	Ű	ë	ű
C			,	<	L	\	l	|		Ź	ź	Ě	Ü	ě	ü
D			−	=	M	]	m	}			SHY	˝	Í	Ý	í	ý
E			.	>	N	^	n	~			Ž	ž	Î	Ţ	î	ţ
F			/	?	O	_	o				Ż	ż	Ď	ß	ď	˙

Figure 50. (ASCII) ISO 8859-2 Character Set (11). Font 1 (Primary)

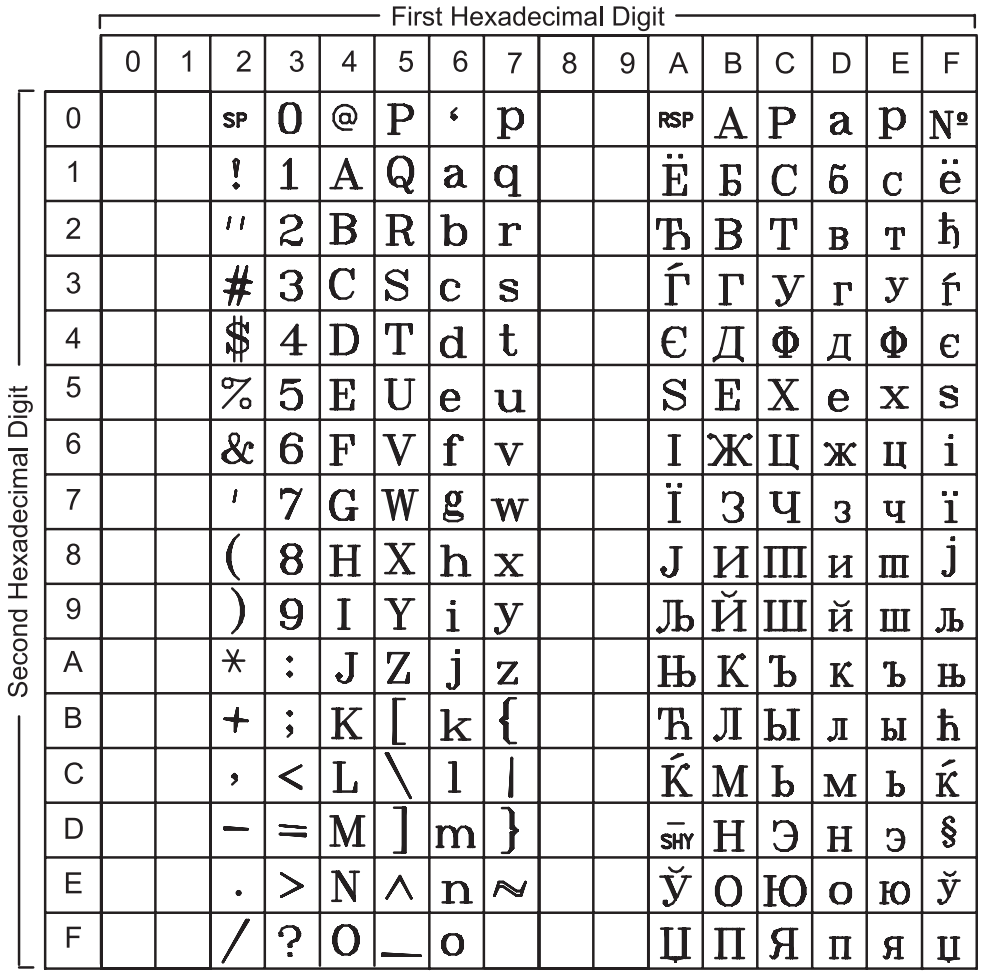


Figure 51. (ASCII) ISO 8859-5 Cyrillic Character Set (12). Font 1 (Primary)

Chapter 10. User-Definable Fonts

You can define your own symbols to display geometric text for use with the graPHIGS API. See Character Sets and Fonts Provided by the API for a detailed explanation of CSIDs and fonts.

If you are defining fonts to be used on the IBM 5080, be sure to read IBM 5080 Character Set Restrictions.

Defining Your Own Characters

Your application can specify a character string to be displayed in geometric text. The graPHIGS API, in processing that character string, accesses two files to get the information necessary to interpret and display the text:

- A character set file containing information about a particular character set. The contents of the character set file tell:
  - The available EBCDIC and ASCII code points of the character set
  - The translation between EBCDIC and ASCII
  - The default character used when a character string contains characters not defined in the file
  - The index that is used to locate the symbol in a symbol file

  There is only one character set file for a particular character set identifier.
- A symbol file containing the drawing controls for each symbol that is displayed to represent the character. There is a symbol file for each available font within a particular character set.

By creating new files, you can create your own character sets and fonts for use by the graPHIGS API. Your character sets and fonts should be assigned a unique character set identifier within the range specified for your use. See Identifying a Character Set for more information.

Your application program specifies CSIDs as integer values. The API divides CSID values into the following ranges:

CSID 1 - CSID 100	Reserved for use by IBM for single-byte character sets
CSID 101 - CSID 127	Reserved for your use for single-byte character sets
CSID 128 - CSID 228	Reserved for use by IBM for double-byte character sets
CSID 229 - CSID 255	Reserved for your use or for double-byte character sets

The graPHIGS API provides many character set files and symbol files. *You should not modify the IBM-supplied files.* The installation of new releases causes the IBM-supplied files to be loaded to your installation disk, replacing the current files. New releases may contain corrections to errors or may provide additional fonts for existing character sets. If you modify the IBM-supplied files, you will lose your changes when the next release is installed. If you wish to modify the IBM-supplied character set and symbol files, copy them to separate files and use the copies to create user-defined character sets and fonts. IBM-supplied, double-byte character sets have exceptions to this rule. See Creating New Double-Byte Code Points for more information.

Font Editor

For your convenience, the Personal graPHIGS product on the RS/6000 provides a basic font editor. The font editor allows you to create and modify characters in font files. The font editor is found in the directory:

```
/usr/lpp/graPHIGS/clients/fonteditor
```

See the README files in the directory for instructions on use of the font editor.

Assigning ASCII and EBCDIC Code Points

Assignable code points of a character set must be within the ranges shown in the following table:

Table 108. Range of Assignable Code Points

Assigned Byte	Code Point in EBCDIC	Code Point in ASCII
Single-byte of SBCS	X'40' to X'FE'	X'20' to X'FF'
Both bytes of DBCS	X'40' to X'FE'	X'20' to X'FF'
Both bytes of Unicode DBCS	X'00' to X'FF'	X'00' to X'FF'

Translation Tables

If you create your own character sets, you should provide the translation tables for EBCDIC and ASCII. Though your current application needs may be only for one encoding (such as EBCDIC), your future requirements may use the character set and font files in other environments. Specifying both the EBCDIC and ASCII organizations and the corresponding translations will eliminate rework of your files for such a conversion. If your character set doesn't follow a standard EBCDIC or ASCII definition, you could assign the same code point for both the EBCDIC and ASCII tables.

Font Considerations

If you are creating your own font within a character set, ensure that you define a symbol for every character in the set you are working with, and that the character maintains its meaning.

If you wish to add or delete characters, move characters to different code points, or change the meaning of the code point, create a new character set for the font.

Creating New Double-Byte Code Points

Double-byte character sets (DBCS) have user-definable subareas of the code point range where you may create additional characters.

The ranges of these sub-areas are shown in the following table:

Table 109. User-assignable Code Point Ranges for Kanji, Hangul, Chinese, and Unicode

Encoding	First byte	Second byte
Kanji (EBCDIC)	X'69' to X'7F'	X'40' to X'FE'
Kanji (ASCII) (CSIDs 128 and 134)	X'F0' to X'F9'	X'40' to X'FC'
Hangul (ASCII)[2]	X'8F' to X'A0'	X'20' to X'FF'
Hangul (EBCDIC)[2]	X'D4' to X'DD'	X'40' to X'FE'
Traditional Chinese (ASCII)	X'DB' to X'FB'	X'40' to X'7E' X'80' to X'FC'

Table 109. User-assignable Code Point Ranges for Kanji, Hangul, Chinese, and Unicode (continued)

Encoding	First byte	Second byte
Simplified Chinese (ASCII)	X'A2' X'A2' X'A2' X'A2' X'A4' X'A5' X'A6' X'A6' X'A7' X'A7' X'A8' X'A8' X'A9' X'A9' X'AA' to X'AF' X'D7' X'F8' to X'FD' X'FE'	X'A1' to X'B0' X'E3' to X'E4' X'EF' to X'F0' X'FD' to X'FE' X'F4' to X'FE' X'F7' to X'FE' X'B9' to X'C0' X'D9' to X'FE' X'C2' to X'D0' X'F2' to X'FE' X'BB' to X'C4' X'EA' to X'FE' X'A1' to X'A3' X'F0' to X'FE' X'A1' to X'FE' X'FA' to X'FE' X'A1' to X'FE' X'A1' to X'DF'
Unicode[1]	X'E0' to X'F7'	X'00' to X'FF'

Notes:

1. The Unicode standard defines a Private Use Area. By convention, the Private Use Area is divided into a Corporate Use Zone, starting at 0xF7FF and decreasing in values, and an End User Zone, starting at 0xE000 and increasing in values.
2. For Hangul there are 1880 user-definable characters in the S/390 environment, but only 188 are used by the graPHIGS API because the 5086 workstation is based on the KSC5601-87 code set.

The graPHIGS API supplies two methods for you to define characters for the user-definable ranges of double-byte character sets:

- Add new character definitions to the existing IBM-supplied character set and font files.
- Create a character set and font files for just the user-definable subarea. The graPHIGS API could then use these new files in conjunction with the IBM-supplied double-byte files.

Method 1

Alter the IBM-supplied character set files by adding characters in the user-definable range. You must also update the two translation tables for translating between EBCDIC and ASCII, except for the Unicode character set which has no translation table. These altered IBM files then become unique to your application.

Method 2

Create a new character set and font files for just the user-assignable code points you wish to define. These files may then be combined with the IBM-supplied files by the graPHIGS API. This method has particular advantages pertaining to the graPHIGS API shell and nucleus organization. (See *The graPHIGS Programming Interface: Understanding Concepts* for discussions of the shell-nucleus organization.) With this method, all applications running to a particular nucleus can share the common IBM-supplied files, yet each application can have a unique set of files defining some set of the user-assignable codepoints installed on the application's disk.

The IBM-defined double-byte files should be installed on the nucleus font disk. The user-defined files should be on the application's disk.

To use a double-byte character set on a workstation:

1. Create a font directory with the Create Font Directory (**GPCRFD**) subroutine

2. Associate it with the workstation specified with the Associate Font Directory with Workstation (**GPAFDW**) subroutine
3. Load the user-defined font with Load Font (**GPLDFO**) subroutine
4. Activate the font with the Activate Font (**GPACFO**) subroutine.

When receiving the activate font request, the graPHIGS API nucleus searches in the associated font directory for the font files. If the nucleus finds the user-defined files in the associated font directory, then it searches for the IBM-defined files on the nucleus' disk. It then activates the font and includes both the IBM ranges and the user-defined ranges. Therefore, each application can ensure that the activated font includes its own user-defined codepoints. The IBM-defined font is always taken from the nucleus font disk and the application need not send it from the shell side.

When creating user-defined characters for Traditional Chinese, be aware that the graPHIGS API processes code points as IBM-927 double-byte encoding, not the Traditional Chinese IBM-EUC encoding.

Displaying a Text String

This section explains the relationship between the information in a symbol file and the display of a text string. The text string appearance is defined by the font characteristics specified in the symbol file and the geometric text attributes.

For example, the alignment attribute allows your application to specify a positioning line on which to position the entire text extent rectangle relative to the text position (see Text Extent Rectangle for an explanation of the text extent rectangle). The positioning line is specified by the attribute and the value of the positioning line is in the symbol file. The graPHIGS API uses both the symbol file description and the attributes to control text position, alignment, height and width. The following sections describe the symbol file contents and illustrate how it relates to geometric text attributes.

Note: For illustration purposes, all diagrams contain proportional sized character boxes.

Font Description Coordinate System

The shape of each symbol representing a character is defined in its own local, two dimensional, Cartesian font coordinate system. The intersection of the x- and y-axes in this coordinate system is called the font coordinate origin.

Each character in a font coordinate system has a character box composed of the character body (topline, bottomline, rightline and leftline), baseline, halfline, capline and centerline. Character box characteristics are defined in relation to the font coordinate origin (the intersection of the x- and y-axes). The following diagram describes the character box about a local font coordinate system:

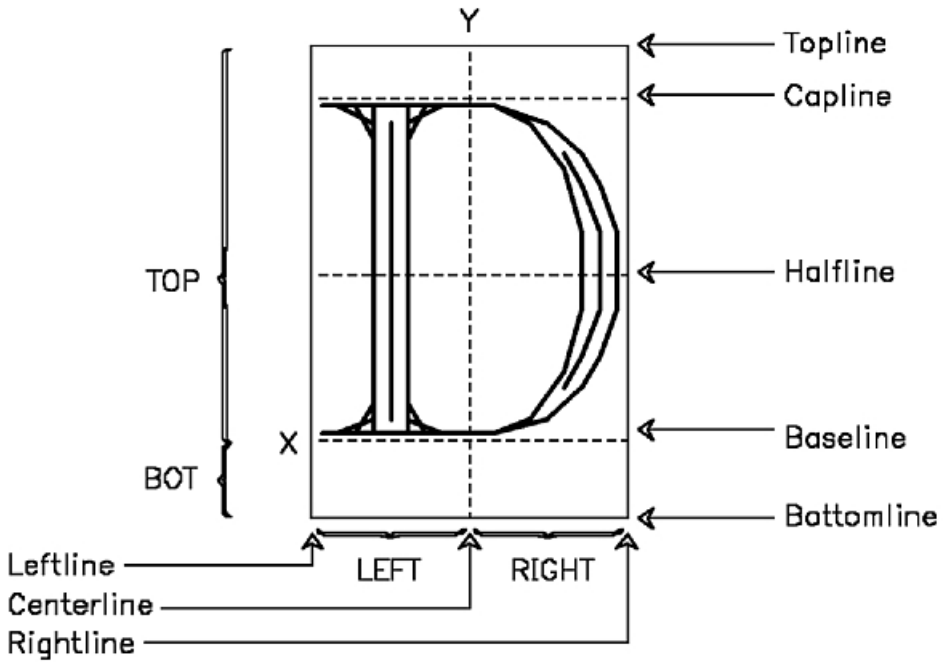


Figure 52. Character box about a local font coordinate system. This illustration shows a character box containing the upper-case letter D. The D is horizontally centered about a Y-axis. An X-axis forms the baseline of the D. Listed from top to bottom and positioned just outside and to the right of the character box are the following callouts: Topline, Capline, Halfline, Baseline, and Bottomline. The D is contained within the capline and baseline and is centered about the halfline. Along the bottom of the character box and positioned from left to right are the following callouts: Leftline, Centerline, and Rightline, which denote the vertical reference points of the character box.

The rectangle that bounds the character box, called the character body, is defined by four values named `TOP`, `BOT`, `RIGHT` and `LEFT`. Each value is a signed value relative to the local coordinate system origin.

A font contains character box definitions where the baseline, halfline, capline and centerline are fixed per font. The baseline and centerline correspond respectively to the x- and y-axes of a character's local font coordinate system. The capline is specified per font and the halfline is midway between the baseline and the capline.

Fixed-sized fonts have the character body, `TOP`, `BOT`, `RIGHT` and `LEFT` specified once for the entire font. Proportionally-spaced fonts also have a character body dimension for each character.

Symbol Position and Inter-Symbol Alignment

The character body is used to position a character relative to other characters within a text string. The `TOP`, `BOT`, `RIGHT` and `LEFT` values are used to position adjacent characters with their bodies touching. For example, in a text string with text path right and no additional space between character bodies, the right edge of each character body will be co-linear with the left edge of the character body to its immediate right.

Characters within a text string are aligned along each character's baseline or centerline. All the origins are co-linear along a line that is parallel to the text path direction.

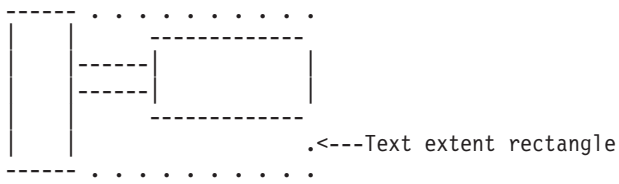
Characters are defined within the character body boundaries but they may exceed the boundaries. For example, a kern or oversized character like the integral sign may exceed the boundaries. All character positioning is performed on the character body not on the character extents, which may exceed the character body.

Text Extent Rectangle

The text extent rectangle is the minimum rectangle which completely encloses all the character extent rectangles in a string. The character extent rectangle has dimensions similar to the character body, TOP, BOT, RIGHT and LEFT. Conceptually, there is a character extent rectangle for each character. The character extent rectangle and the character body will be defined by the same parameters.

Calculation of the vertical and horizontal components of the text extent rectangle involves calculating the maximum dimensions in a text string. For example, the text extent for path right or left is the left-most leftline, right-most rightline, maximum topline and the minimum bottomline.

Location of a text extent rectangle



Text Alignment

Text alignment is calculated relative to the text extent rectangle. The horizontal component of text alignment has three values: RIGHT_ALIGN, LEFT_ALIGN and CENTER. The vertical component has five values: TOP, CAP, HALF, BASE and BOTTOM.

Notice where these values lie when you specify a horizontal alignment, as shown in the figure below:

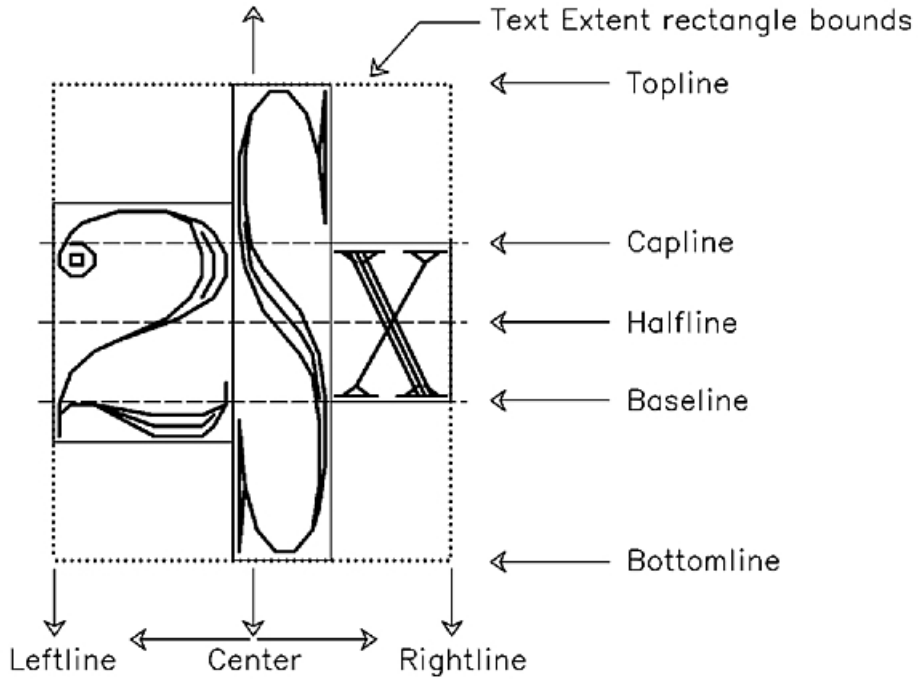


Figure 53. . This illustration shows how horizontal alignment can affect characters.

When your application specifies a vertical alignment, as shown in the following figure, notice:

- Halfline specification aligns the midpoint between the top-most character's halfline and the bottom-most character's halfline with the text position.
- Centerline specification aligns along the centerline or Y axis passing through all the character extent rectangles.

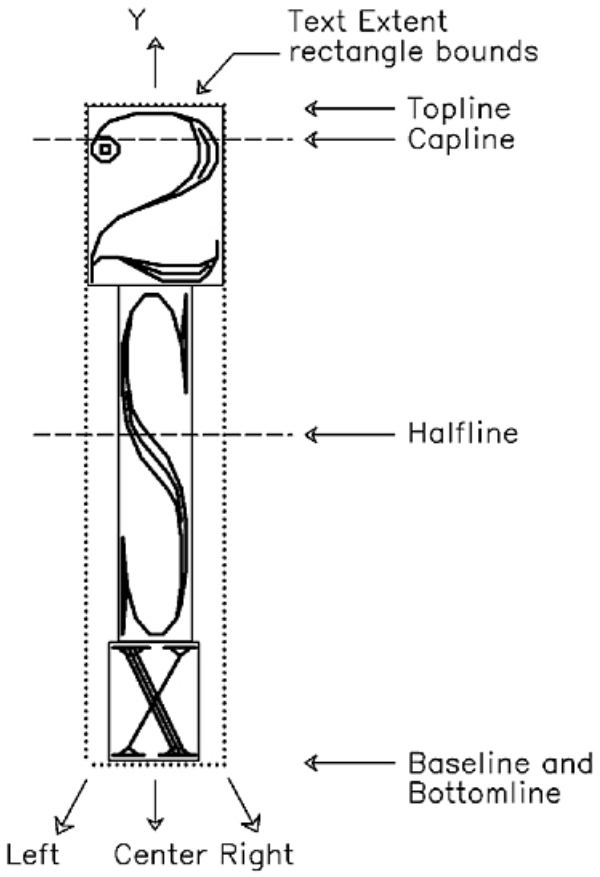


Figure 54. Alignment of a text string specified with path up or down. This illustration shows how vertical alignment can affect characters.

Mapping Font Coordinates to Modeling Coordinates

The graPHIGS API displays text strings by mapping the local font coordinates of each symbol to modeling coordinates. The application specifies the character height in modeling coordinates. The height of the font in the font coordinate system is mapped to the height specified in modeling coordinates. The ratio between the font height to the modeling height is multiplied by the expansion factor and the font width produce the width. The following diagram illustrates the relationship between local font coordinates and modeling coordinates:

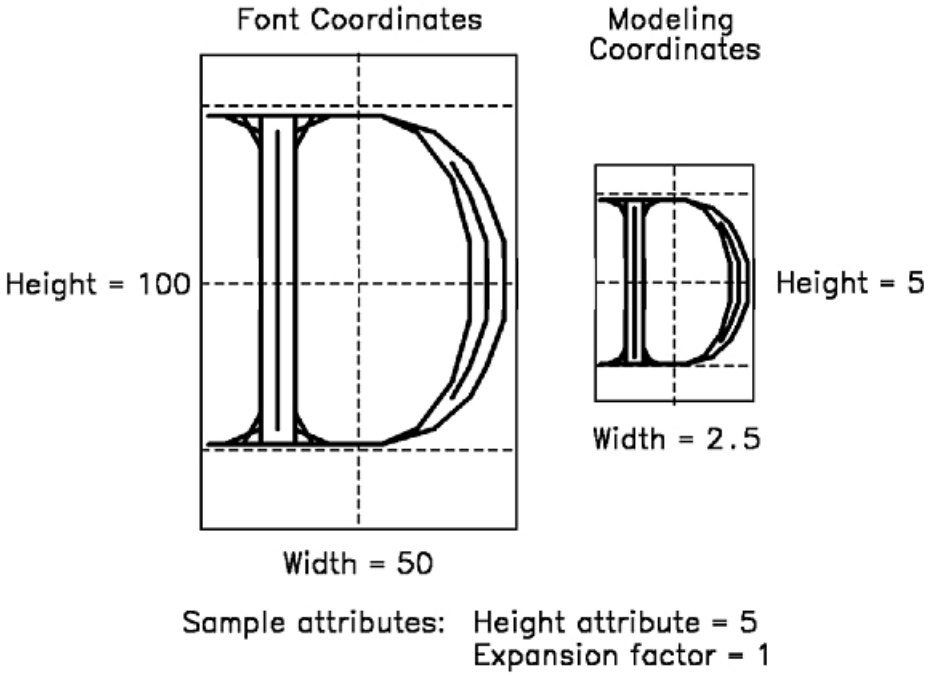


Figure 55. Relationship between Local Font Coordinates and Modeling Coordinates. This illustration shows the relationship between local font coordinates and modeling coordinates.

Font File Organization Overview

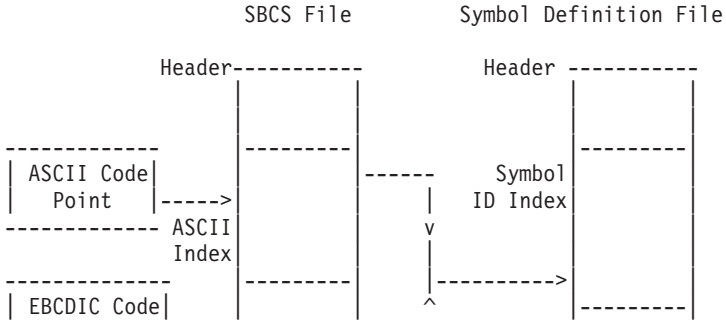
This section describes the file organization used to map a code point to its drawing controls.

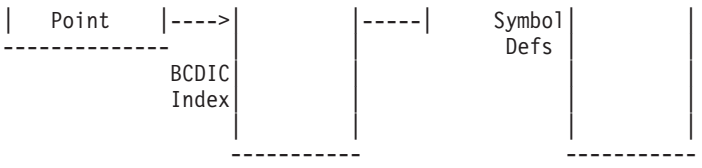
Single-Byte Character Sets

Each single-byte character set (SBCS) and font combination consists of two files, the SBCS file and the Symbol Definition file.

The SBCS file contains ASCII and EBCDIC code point indexes that map code points to symbol IDs. (The symbol ID serves as a generic name for a symbol that is independent of an encoding scheme such as ASCII or EBCDIC.) The Symbol Definition file contains a symbol ID table that maps symbol IDs to their drawing controls.

The complete process from code point to drawing controls begins with the code point, which maps to a symbol ID that in turn identifies a location where the drawing controls for that symbol ID are found. The following diagram shows this flow in relationship to the files.





Single-Byte SBCS and Symbol Definition Files
Single-byte ASCII and EBCDIC code points
having the same symbol ID will map to the same drawing controls.

Double-Byte Character Sets

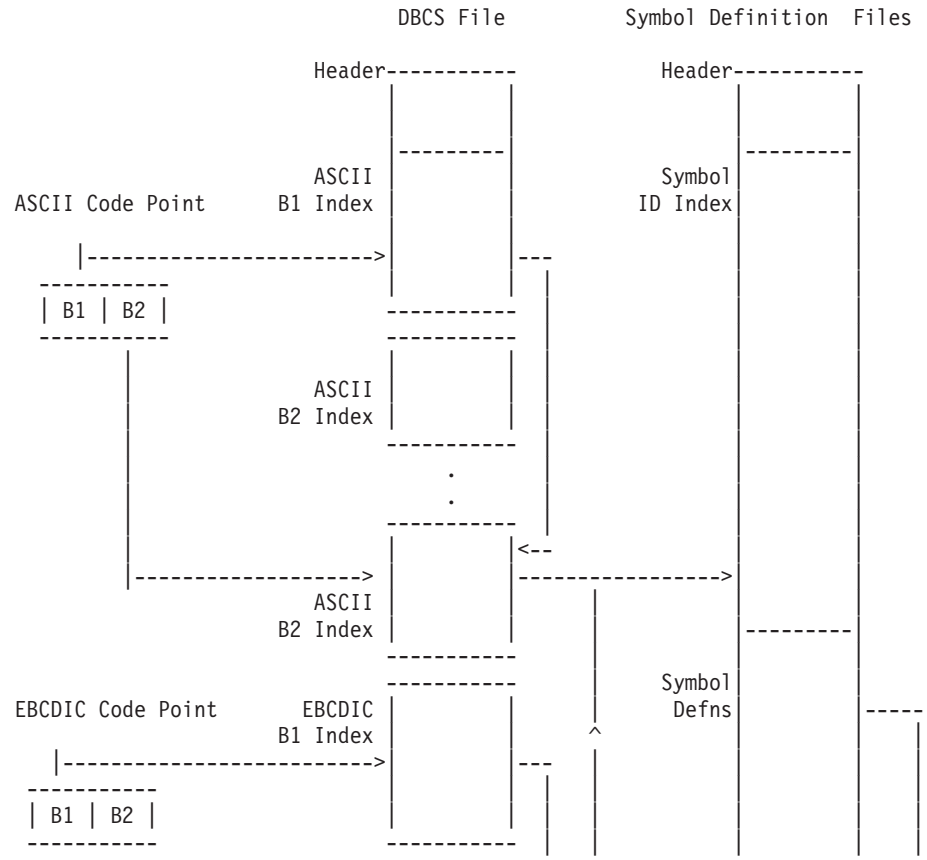
Each double-byte character set (DBCS) and font combination consists of 2 files, the DBCS file and the Symbol Definition file.

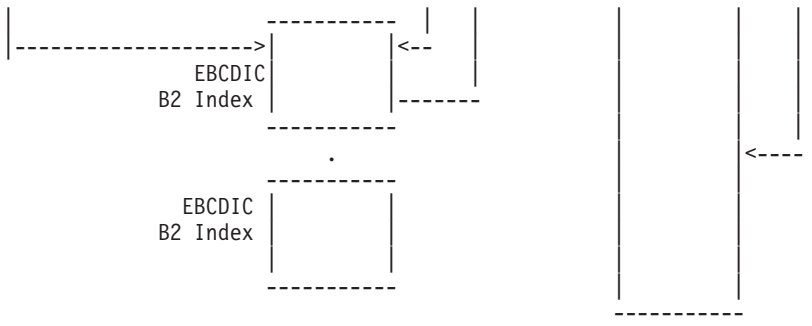
The DBCS file contains ASCII and EBCDIC code point indexes that map code points to symbol IDs. (The symbol ID serves as a generic name for a symbol that is independent of an encoding scheme such as ASCII or EBCDIC.) The Symbol Definition file contains a symbol ID table that maps symbol IDs to their drawing controls.

The double-byte character set file has two levels of in-direction. The first byte of the code point ('B1') is used as an index into a table, called the B1 table. The B1 table contains the addresses of tables indexed by the second byte of the code point ('B2'), called the B2 tables. The entry in the B2 table contains a symbol ID.

The complete process from code point to drawing controls begins with the code point, which maps to a symbol ID (using two levels of in-direction), that in turn identifies a location where the drawing controls for that symbol ID are found.

The following example shows this flow in relationship to the files.





Double-Byte DBCS and Symbol Definition Files
Double-byte ASCII and EBCDIC Code Points
having the same symbol ID maps to the same drawing controls.

Overview of Font File Contents

Character Set Files

The Single-Byte Character Set (SBCS) files and the Double-Byte Character Set (DBCS) files are unique per character set. The information in these files varies depending on whether it is for a single-byte or double-byte character set.

Header

The SBCS files and the DBCS files each have a fixed-length header which is located at the beginning of the file and contains control information. Header information includes:

- Font File Version Identifier
  - This identifies the format of the font file to the graPHIGS API.
- ASCII and EBCDIC Default characters
  - You do not need to define all characters in a character set. Each character set contains a symbol called the default character that is displayed when your application references an undefined character.
- Offsets to tables
  - These are offsets to variable-length tables which follow the header. All offsets are from the start of the variable data.
- Starting and last character codes
  - This is the range of character codepoints for a SBCS and the range of the first byte of the codepoints for a DBCS.

Variable Data

Variable data in the SBCS files and the DBCS files begins immediately after the header. The graPHIGS API accesses this part of the file by using the offsets and ranges in the header. The variable data may include:

- ASCII and EBCDIC Character Code Tables
  - These tables are in the SBCS files, and contain symbol IDs.
- ASCII and EBCDIC B1 Tables
  - These tables are in the DBCS files, and contain offsets to the B2 tables.
- ASCII and EBCDIC B2 Tables
  - These tables are in the DBCS files. The tables contain range values for the second byte of a double-byte codepoint, and symbol IDs for that range.
- Translation Tables

– Two translation tables translate character codes from ASCII to EBCDIC and EBCDIC to ASCII.

Symbol Definition File

The Symbol Definition files are unique per font within a character set.

Header

The file contains a fixed length header which includes:

- Font File Version Identifier
  – This identifies the format of the symbol definition file to the graPHIGS API.
- Character box and clipping boundaries
  – Nominal character box values are used for text alignment. Clipping box values are used for clipping and trivial accept/reject.
- Offsets
  – These are the offsets that the graPHIGS API uses to locate (in the variable data) the symbol ID index and the symbol definitions. All offsets are from the start of the variable data, which begins immediately after the header.
- Starting and last symbol ID
  – This is the range of symbol IDs for this font.

Variable Data

Variable data in the Symbol Definition files begins just after the header. The graPHIGS API accesses this part of the file by using the offsets and ranges in the header. The variable data includes:

- Symbol ID index
  – This table contains an entry for every symbol ID in the range given in the header. Each entry contains information pertaining to a symbol ID, including an offset to the drawing controls.
- Symbol Definitions
  – This is a collection of the drawing controls for each symbol.

Font File Naming Conventions

When the graPHIGS API receives a request to activate (Activate Font - **GPACFO**) or load (Load Font - **GPLDFO**) a character set and font, it attempts to load either the SBCS or DBCS file and the Symbol Definition file. The filenames that the software expects for these files differ according to your environment: AIX, MVS, or VM.

Character Set Files

Use the following tables to ensure that your filenames are correct for your environment and/or the load option for the **GPLDFO** subroutine.

Each character set identifier has one SBCS/DBCS file that contains the ASCII/EBCDIC index and the translate tables. The graPHIGS API attempts to load the SBCS/DBCS file according to the naming convention. The graPHIGS API also attempts to load the user-defined subareas of double-byte character sets, according to the naming convention.

Table 110. SBCS/DBCS File-Naming Convention on AIX, MVS, and VM

VM	MVS	AIX
1. Filename 'AFM*cc* '	**2. PDS member**'AFM*cc* '	**3. Filename** 'afm*cc*.sym'
4. Filetype 'AFMSYMBL'	**5. DDNAME** 'AFMSYMBL'	
Note: *cc* = Character set identifier in hexadecimal; see Identifying a Character Set.		

Table 111. Naming Convention on AIX, MVS, and VM for User-Defined DBCS

VM	MVS	AIX
1. Filename 'UDF*cc* '	**2. PDS member**'UDF*cc* '	**3. Filename** 'udf*cc*.sym'
4. Filetype 'AFMSYMBL'	**5. DDNAME** 'AFMSYMBL'	
Notes: 1. *cc* = Character set identifier in hexadecimal; see Identifying a Character Set 2. CSIDs 130, 132 and 134 are only supported in the operating system environment.		

Symbol Definition Files

If the SBCS/DBCS file loads successfully, then the graPHIGS API attempts to load the respective Symbol Definition file according to the naming convention in the following table. For IBM-supplied double-byte symbol files, the API also attempts to load the user-defined range, according to the naming convention in the following user-defined table.

Table 112. Symbol Definition File Naming Conventions on AIX, MVS, and VM

VM	MVS	AIX
1. Filename 'AFM*ccff* '	**2. PDS member**'AFM*ccff* '	**3. Filename** 'afm*ccff*.sym'
4. Filetype 'AFMSYMBL'	**5. DDNAME** 'AFMSYMBL'	
Note: *cc* = Character set identifier in hexadecimal; See Identifying a Character Set. *ff* = Font identifier in hexadecimal		

Table 113. Naming Convention on AIX, MVS, and VM for User-Defined Symbol Definition File

VM	MVS	AIX
1. Filename 'UDF*ccff* '	**2. PDS member**'UDF*ccff* '	**3. Filename** 'udf*ccff*.sym'
4. Filetype 'AFMSYMBL'	**5. DDNAME** 'AFMSYMBL'	
Note: *cc* = Character set identifier in hexadecimal; See Identifying a Character Set. *ff* = Font identifier in hexadecimal CSIDs 130, 132 and 134 are only supported in the operating system environment.		

Font File Format Specifications

This section provides tabular summaries of file format and notes on the contents of these three files:

- Single-Byte Character Set (SBCS) file
- Double-Byte Character Set (DBCS) file
- Symbol Definition file

The internal file format is binary on all environments. On VM and MVS, all files are four hundred byte fixed length records.

Refer to the tables for information on byte offset, field length, field content, or brief descriptions of these characteristics.

The notes following each table, field content descriptions, give you more details on the content of the fields in the file, along with special considerations, restrictions, and usage.

Notes:

1. A “V” in the length field indicates variable-length.
2. All reserved fields in the font files must be initialized to zero.

Single-Byte Character Set File Format

Table 114. Single-Byte Character Set File Format

Byte	Field Length	Field Content	Description
0	1	VERSION	Font file version identifier
1	1	RESERVED	
2	1	FLAGS	Bits are on to indicate: **0** ASCII Index present **1** EBCDIC Index present **2** ASCII to EBCDIC translation table present **3** EBCDIC to ASCII translation table present **4-7** Reserved
3	1	RESERVED	
4	4	LENGTH	Total length of file, all fields
8	2	ASCII DEF	ASCII default character code
10	1	ASCII CP0	Starting ASCII character code
11	1	ASCII CPN	Last ASCII character code
12	4	ASCII CHAR INDEX OFFSET	Offset of ASCII CHAR INDEX
16	4	ASCII —> EBCDIC OFFSET	Offset of ASCII to EBCDIC translation table
20	2	EBCDIC DEF	EBCDIC default character code
22	1	EBCDIC CP0	Starting EBCDIC character code
23	1	EBCDIC CPN	Last EBCDIC character code
24	4	EBCDIC CHAR INDEX OFFSET	Offset of EBCDIC CHAR INDEX
28	4	EBCDIC —> ASCII OFFSET	Offset of EBCDIC to ASCII translation table

Table 114. Single-Byte Character Set File Format (continued)

Byte	Field Length	Field Content	Description
32	**LENGTH DATA**		
	V	ASCII CHAR INDEX	2 byte symbol IDs
		EBCDIC CHAR INDEX	2 byte symbol IDs
		ASCII —> EBCDIC	Table to translate from ASCII to EBCDIC 1 byte character codes
		EBCDIC —> ASCII	Table to translate from EBCDIC to ASCII 1 byte character codes

Field Content Description

- VERSION
  - The version identifier for this format is X'01'.
- FLAGS
  - ASCII Index flag
    - This flag is on to indicate that an ASCII index has been specified for this character set.
  - EBCDIC Index flag
    - This flag is on to indicate that an EBCDIC index has been specified for this character set.
  - ASCII to EBCDIC translation table flag
    - This flag is on to indicate that the ASCII to EBCDIC translation table has been specified for this character set.
  - EBCDIC to ASCII translation table flag
    - This flag is on to indicate that the EBCDIC to ASCII translation table has been specified for this character set.
- DEF (ASCII and EBCDIC)
  - Character code used when a request is made for a character code outside the range CP0-CPN, or those with an index value of zero. The default must be a defined character code.
  - The default character on the 5080 will be ignored; it is always the hyphen.
  - For a single-byte character set, the first byte of the character code is zero and the second byte is the default character code, e.g., X'00cc'.
- OFFSET (All table offset fields)
  - The location of variable length tables are specified by an offset into the file that is relative to the start of the variable length data. For example, the first table after the header is at offset zero.
- CP0 and CPN (ASCII and EBCDIC)
  - The range of character code points start at CP0 and end at CPN.
  - You do not have to specify both an ASCII and EBCDIC index. If you do not specify one of the indices, make sure that the corresponding ASCII/EBCDIC flag is off to indicate that an index has not been included.
  - With the exception of the Unicode character set (CSID 131), valid character code ranges for CP0 to CPN are between X'20' and X'FF' for ASCII and X'40' and X'FE' for EBCDIC. The graPHIGS API ignores all character codes in the range X'00' to X'1F' for ASCII, and X'00' to X'3F' and X'FF' for EBCDIC. The graPHIGS API, however, recognizes the full range of character codes X'00' to X'FF' for the Unicode character set.
- CHAR INDEX (ASCII and EBCDIC)
  - Character index is an array of symbol IDs. A symbol ID is a two-byte positive integer.

- Even if you do not define all the characters, this index must be complete between CP0-CPN. You must set the character index for undefined characters to zero, in order to indicate non-supported characters to the API.

- ASCII > EBCDIC translation table
  - This table converts an ASCII character code to the corresponding EBCDIC character code.
  - The translation table is not provided for the Unicode character set (CSID 131). The ASCII and EBCDIC code points are identical.
  - The translation table is an array of one byte character codes. The following illustrates the translation table format:

Table 115. ASCII <—> EBCDIC Translation

Field Length	Field Content	Description
1	CP0	Starting character code
1	CPN	Last character code
V	CODE POINTS	1 byte code points

- EBCDIC —> ASCII Translation table
  - This table converts an EBCDIC character code to the corresponding ASCII character code.
  - The translation table is not provided for the Unicode character set (CSID 131). The ASCII and EBCDIC code points are identical.
  - The translation table is an array of one byte character codes. The format of this translation table is identical to the ASCII —> EBCDIC translation table above.

Double-Byte Character Set File Format

Table 116. Double-Byte Character Set File Format

Byte	Field Length	Field Content	Description
0	1	VERSION	Font file version identifier
1	1	RESERVED	
2	1	FLAGS	Bits are on to indicate: **0** ASCII Index **1** EBCDIC Index **2** ASCII to EBCDIC translation table **3** EBCDIC to ASCII translation table **4-7** Reserved
3	1	RESERVED	
4	4	LENGTH	Total length of file, all fields
8	2	ASCII DEF	ASCII default character code
10	1	ASCII B10	Starting character code for first byte of double-byte character code
11	1	ASCII B1N	Last character code for first byte of double-byte character code.
12	4	ASCII B1 TABLE OFFSET	Offset of B1 table

Table 116. Double-Byte Character Set File Format (continued)

Byte	Field Length	Field Content	Description
16	4	ASCII —> EBCDIC OFFSET	Offset of ASCII to EBCDIC translation table
20	2	EBCDIC DEF	EBCDIC default character code
22	1	EBCDIC B10	Starting character code for first byte of double-byte character code
23	1	EBCDIC B1N	Last character code for first byte of double-byte character code.
24	4	EBCDIC B1 TABLE OFFSET	Offset of B1 table
28	4	EBCDIC —> ASCII OFFSET	Table to translate from EBCDIC to ASCII
32	**START OF VARIABLE LENGTH DATA**		
	V	ASCII B1 TABLE	Fullword offset for each B2 table
		ASCII B2 TABLES	B2 table definition
		EBCDIC B1 TABLE	Fullword offset for each B2 table
		EBCDIC B2 TABLES	B2 table definition
		ASCII —> EBCDIC	Table to translate from ASCII to EBCDIC
		EBCDIC —> ASCII	Table to translate from EBCDIC to ASCII

Field Content Description

The first six fields of the DBCS file are similar to the SBCS file. Refer to Field Content Description, the description for SBCS, for more information.

- OFFSET (All table offset fields)
  - The location of variable length tables are specified by an offset into the file that is relative to the start of the variable length data. For example, the first table after the header is at offset zero.
- B10 and B1N (ASCII and EBCDIC)
  - The range of the first byte in the double-byte character code starts at B10 and ends at B1N.
  - You do not have to specify both an ASCII and EBCDIC index. If you do not specify one of the indexes, make sure that the corresponding ASCII/EBCDIC flag is off to indicate that an index has not been included.
  - With the exception of the Unicode character set (CSID 131), valid character code ranges for B10 to B1N are between X'20' and X'FF' for ASCII and X'40' and X'FE' for EBCDIC. The graPHIGS API ignores all character codes in the range X'00' to X'1F' for ASCII, and X'00' to X'3F' and X'FF' for EBCDIC. The graPHIGS API, however, recognizes the full range of character codes X'00' to X'FF' for the Unicode character set.
- B1 table (ASCII and EBCDIC)
  - The B1 table is an array of offsets to the B2 tables. The offsets are relative to the start of the variable length data area.

– This index must be complete between B10-B1N even if all the characters are not defined. The character index for undefined characters should be zero, indicating a non-supported character encountered.

- B2 tables (ASCII and EBCDIC)
  – B2 table format:

Table 117. B2 ASCII and EBCDIC Format

Field Length	Field Content	Description
1	B20	Starting character code for 2nd byte of double-byte character code
1	B2N	Last char code for 2nd byte of double-byte character code
V	CHAR INDEX	2-byte symbol IDs

  – B20 and B2N
    - The range of the second byte in the double-byte character code starts at B20 and ends at B2N.
    - With the exception of the Unicode character set (CSID 131), valid character code ranges for B20 to B2N are between X'20' and X'FF' for ASCII and X'40' and X'FE' for EBCDIC. The graPHIGS API restricts all character sets from using X'00' to X'1F' in ASCII, and X'00' to X'3F' and X'FF' in EBCDIC. The Unicode character set does not restrict any symbols in the range X'00' to X'FF'.
  – CHAR INDEX
    - Character index is an array of symbol IDs. A symbol ID is a two-byte positive integer.
    - Even if you do not define all the characters, this index must be complete between B20-B2N. You must set the character index for undefined characters to zero, in order to indicate non-supported characters to the API.

- Translation tables (ASCII to EBCDIC, EBCDIC to ASCII)
  – The translation tables use a two-level lookup identical to the double-byte character index. The B1 table contains the addresses of B2 tables. The B2 tables contain codepoints. The two level lookup maps from one encoding, (ASCII or EBCDIC), to the opposite encoding. The following describes the B1 and B2 tables.

Table 118. B1 (ASCII to EBCDIC) Addresses

Field Length	Field Content	Description
1	B10	Starting character code
1	B1N	Last character code
2	RESERVED	
V	OFFSET	4 byte offset to start of the B2 table relative to the start of the variable data area. The offset for undefined character codes should be zero.

Table 119. B2 (ASCII to EBCDIC) Code Points

Field Length	Field Content	Description
1	B20	Starting character code for 2nd byte of double-byte character code
1	B2N	Last char code for 2nd byte of double-byte character code
V	CODE POINTS	2-byte code points

Symbol Definition File Format

Table 120. Symbol Definition File Format

Byte	Field Length	Field Content	Description
0	1	VERSION	Font file version identifier
1	2	FLAGS	Bits are to indicate what optional information is present for each symbol: **0** Symbol positioning box is present **1-7** Reserved
3	1	CAP	CAP line, must be greater than 0
4	1	NTOP	TOP of nominal symbol box
5	1	NBOT	BOT of nominal symbol box
6	1	NRIGHT	RIGHT of nominal symbol box
7	1	NLEFT	LEFT of nominal symbol box
8	2	CTOP	TOP of symbol clip box
10	2	CBOT	BOT of symbol clip box
12	2	CRIGHT	RIGHT of symbol clip box
14	2	CLEFT	LEFT of symbol clip box
16	2	ASYM0	Starting symbol ID for range of symbol IDs within this font
18	2	ASYMN	Last symbol ID for range of symbol IDs within this font
20	4	LENGTH	Total length of structure, including field
24	4	SID INDEX OFFSET	Offset of the symbol ID index
28	4	SYMBOL DEFNS OFFSET	Offset to the start of the symbol definitions
32	**START OF VARIABLE LENGTH DATA**		
	V	SID INDEX	Symbol ID index
		SYMBOL DEFNS	Symbol definitions

Field Content Description

- VERSION
  - The version identifier for this format is X'01'.
- FLAGS
  - Symbol positioning box flag. This flag is on to indicate that each symbol index entry contains a character body: `TOP, BOT, RIGHT, LEFT`.
- CAP
  - Capline for this font. This value must be positive.
- NTOP, NBOT, NRIGHT and NLEFT

  - Values used to display the font with fixed size. The nominal symbol body dimensions (NTOP, NBOT, NRIGHT and NLEFT) are signed integers. All values may range from positive 127 to negative 128.
- CTOP, CBOT, CRIGHT and CLEFT
  - The minimum clipping boundary such that all strokes fall completely inside the boundary whenever all symbols are positioned on a common origin. The minimum clipping boundary is expressed in local font coordinates relative to the common origin.
  - The clip box dimensions (CTOP, CBOT, CRIGHT and CLEFT) are signed integers.
  - The API uses these dimensions for clipping and trivial accept/reject. If symbols exceed these dimensions then unpredictable results may occur.
- ASYM0 and ASYMN
  - The range of symbol IDs start at ASYM0 and end at ASYMN. This is the range for all the symbol IDs within this font.
  - Symbol IDs are positive integers that must be greater than or equal to one.
- OFFSET (SID INDEX and SYMBOL DEFNS)
  - The location of variable length tables are specified by an offset into the file that is relative to the start of the variable length data. For example, the first table after the header is at offset zero.
- SID INDEX
  - This index must be complete between ASYM0-ASYMN even if all the symbols are not defined. A symbol index entry for undefined symbols should have the undefined symbol flag on.
  - Each entry in the symbol ID index has the following format:

Table 121. B1 SID Index

Field Length	Field Content	Description
4	OFFSET	Offset to locate symbol. The offset is relative to the start of the symbol definitions, e.g., the first symbol definition after the index has offset zero
2	LENGTH	Number of bytes in symbol definition
1	FLAGS	Flags are on to indicate: **0** Undefined symbol **1** Symbol definition should be filled[1] **2-7** Reserved
1	RESERVED	
1	TOP	Top of character body[2]
1	BOTTOM	Bottom of character body[2]
1	RIGHT	Right of character body[2]
1	LEFT	Left of character body[2]

Note:

[1]Each character does not have to be filled. The flags field indicates whether this character is filled. If the interface does not support filled fonts, then the characters are not filled. See *The graPHIGS Programming Interface: Understanding Concepts* for more information.

[2]The fields TOP, BOTTOM, RIGHT, LEFT are only present if bit 0 of the FLAGS field of the Symbol Definition file header is set ON.

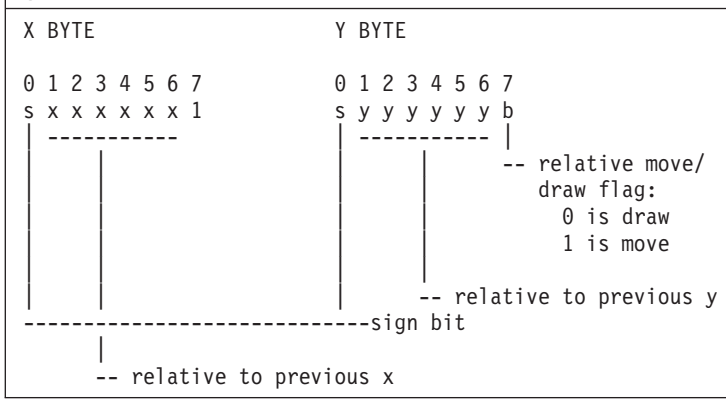
- SYMBOL DEFNS

– Each symbol is represented by a list of X and Y pairs. A move/draw flag is associated with each X-Y pair to indicate whether the graPHIGS API moves or draws to the pair. The X-Y pair are signed relative values, relative to the previous ending point, for the first character to the origin of the character box. If the first X-Y pair is a draw rather than a move then a line is drawn from the origin of the character box.

  Note: The X-Y relative values are represented in two's complement notation (in which both positive and negative numbers are represented as zeros and ones).

Table 122. Symbol Definitions

Symbol Definitions



– The maximum relative distance for X or Y is +63 or -64.

IBM 5080 Character Set Restrictions

For the 5080, the following restrictions apply:

1. The graPHIGS API must know the complete set of character set identifiers and font identifiers which will be used during the workstation session, and the sizes of these character sets.

   The `FONTLIST PROCOPT` allows the user to list the user-defined character sets which will be used for a specific workstation type or connection identifier.

   The API will determine the size of these character sets at Open Workstation time in order to allocate the needed space for them. All of the character sets supplied by IBM will be automatically available on the 5080 workstation, whether or not this `PROCOPT` is specified. See the `FONTLIST PROCOPT` description in FONTLIST (Character Font List).
2. You must indicate the maximum number of character sets which will be active at any one time. This defaults to 3, but may be set to any number up to 10 via the `PROCOPT FONTPSIZ`. See FONTPSIZ (Font Pool Size).
3. The IBM 5080 restricts the number of Programmable Character Sets (PCS) to a maximum of 48. The IBM 5080 uses a PCS for a single-byte character set OR a single first byte range of a double-byte character set. Thus, the total number of single-byte character sets plus the total number of first byte ranges of double-byte character sets cannot exceed 48 during a workstation session.

   If your application uses the Japanese Language Feature for Kanji, PCSs are allocated in the 5080 for the requested wards. These count towards the maximum of 48, but are undetectable by the API. Thus, in this case you must ensure that the number of PCSs simultaneously active in the 5080 does not exceed 48.

Part 5. Format and Content of Structure Element Records

Chapter 11. Structure Element Content as Returned by GPQED

The contents and organization of the structure element contents built by the graPHIGS API are provided in this chapter. The element codes that your application can use in the inclusion and exclusion lists for the Element Search (**GPELS**) subroutine are also shown.

When your application uses the Inquire List of Element Data (**GPQED**) subroutine, see Structure Element Content as Returned by GPQE.

Partial Table-of-Contents

- Area Primitives
- Polygon 3 (GPPG3)
- Polygon 2 (GPPG2)
- Polygon 3 With Data (GPPGD3)
- Polygon 2 With Data (GPPGD2)
- Triangle Strip 3 (GPTS3)
- Quadrilateral Mesh 3 (GPQM3)
- Non-Uniform B-Spline Surface (GPNBS)
- Composite Fill Area 2 (GPCFA2)
- Trimmed Non-uniform B-Spline Surface (GPTNBS)
- Polysphere (GPSPH)
- Pixel Primitives
- Pixel 3 (GPPXL3)
- Pixel 2 (GPPXL2)
- Attribute Setting Structure Elements
- General Attributes
- Set HLHSR Identifier (GPHID)
- Set Antialiasing Identifier (GPAID)
- Set Z-buffer Protect Mask (GPZBM)
- Set Face Lighting Method (GPFLM)
- Set Depth Cue Index (GPDCI)
- Set Color Processing Index (GPCPI)
- Set Highlighting Color Index (GPHLCI)
- Set Highlighting Color Direct (GPHLCD)
- Add Class Name to Set (GPADCN)
- Remove Class Name from Set (GPRCN)
- Set Vertex Morphing Factors (GPVMF)
- Set Transparency Coefficient (GPTCO)
- Set Blending Function (GPBLF)
- Set Line-on-Line Color Direct (GPLLCD)
- Set Line-on-Line Color Index (GPLLCI)
- Attribute Selection
- Set Attribute Source Flag (GPASF)
- Polyline Attributes
- Set Curve Approximation Criteria (GPCAC)
- Set Trimming Curve Approximation Criteria (GPTCAC)
- Set Polyhedron Edge Culling (GPPHEC)
- Set Polyline Index (GPPLI)
- Set Linetype (GPLT)
- Set Polyline End Type (GPPLET)
- Set Linewidth Scale Factor (GPLWSC)
- Set Polyline Color Index (GPPLCI)
- Set Polyline Color Direct (GPPLCD)
- Set Polyline Shading Method (GPPLSM)
- Polymarker Attributes

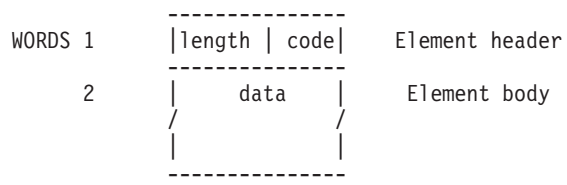
- Set Polymarker Index (GPPMI)
- Set Marker Type (GPMT)
- Set Marker Size Scale Factor (GPMSSC)
- Set Polymarker Color Index (GPPMCI)
- Set Polymarker Color Direct (GPPMCD)
- Text Attributes
- Set Character Height (GPCHH)
- Set Character Line Scale Factor (GPCHLS)
- Set Character Up Vector (GPCHUP)
- Set Character Up and Base Vectors (GPCHUB)
- Set Text Path (GPTXPT)
- Set Text Alignment (GPTXAL)
- Set Character Positioning Mode (GPCHPM)
- Set Text Index (GPTXI)
- Set Text Font (GPTXFO)
- Set Text Precision (GPTXPR)
- Set Text Linewidth Scale Factor (GPTLWS)
- Set Character Expansion Factor (GPCHXP)
- Set Character Spacing (GPCHSP)
- Set Text Color Index (GPTXCI)
- Set Text Color Direct (GPTXCD)
- Annotation Text Attributes
- Set Annotation Text Height Scale Factor (GPAHSC)
- Set Annotation Text Height (GPAH)
- Set Annotation Style (GPAS)
- Set Annotation Text Up Vector (GPAUP)
- Set Annotation Text Path (GPAPT)
- Set Annotation Text Alignment (GPAAL)
- Polygon Attributes
- Set Surface Approximation Criteria (GPSAC)
- Set Polygon Culling (GPPGC)
- Interior Attributes
- Set Face Distinguish Mode (GPFDMO)
- Set Light Source State (GPLSS)
- Set Lighting Calculation Mode (GPLMO)
- Set Interior Index (GPII)
- Set Interior Style (GPIS)
- Set Interior Style Index (GPISI)
- Set Interior Color Index (GPICI)
- Set Interior Color Direct (GPICD)
- Set Back Interior Color Index (GPBICI)
- Set Back Interior Color Direct (GPBICD)
- Set Specular Color Index (GPSCI)
- Set Specular Color Direct (GPSCD)
- Set Back Specular Color Index (GPBSCI)

- Set Back Specular Color Direct (GPBSCD)
- Set Surface Properties (GPSPR)
- Set Back Surface Properties (GPBSPR)
- Set Back Transparency Coefficient (GPBTCO)
- Set Back Blending Function (GPBBLF)
- Set Parametric Surface Characteristics (GPPSC)
- Set Data Morphing Factors (GPDMF)
- Set Back Data Morphing Factors (GPBDMF)
- Set Data Mapping Index (GPDMI)
- Set Back Data Mapping Index (GPBDMI)
- Set Data Filtering Method (GPDFM)
- Set Back Data Filtering Method (GPBDFM)
- Set Data Matrix 2 (GPDM2)
- Set Back Data Matrix 2 (GPBDM2)
- Set Reflectance Model (GPRMO)
- Set Back Reflectance Model (GPBRMO)
- Set Interior Shading Method (GPISM)
- Set Back Interior Shading Method (GPBISM)
- Edge Attributes
- Set Edge Index (GPEI)
- Set Edge Flag (GPEF)
- Set Edge Linetype (GPELT)
- Set Edge Scale Factor (GPESC)
- Set Edge Color Index (GPECI)
- Set Edge Color Direct (GPECD)
- Transformation Setting Structure Elements
- Modeling Transformation
- Set Global Transformation 3 (GPGLX3)
- Set Global Transformation 2 (GPGLX2)
- Set Modeling Transformation 3 (GPMLX3)
- Set Modeling Transformation 2 (GPMLX2)
- Set Modeling Clipping Indicator (GPMCI)
- Restore Modeling Clipping Volume (GPRMCV)
- Set Modeling Clipping Volume 3 (GPMCV3)
- Set Modeling Clipping Volume 2 (GPMCV2)
- Miscellaneous Structure Elements
- View selection
- Set View Index (GPVWI)
- Traversal Control
- Execute Structure (GPEXST)
- Test Extent 3 (GPTEX3)
- Test Extent 2 (GPTEX2)
- Set Condition (GPCOND)
- Conditional Execute Structure (GPCEXS)
- Conditional Return (GPCRET)

- Identification
- Insert Label (GPINLB)
- Set Pick Identifier (GPPKID)
- Frame Buffer Control
- Set Frame Buffer Protect Mask (GPFBM)
- Set Frame Buffer Comparison (GPFBC)
- Application-Defined Data
- Insert Application Data (GPINAD)
- Null Data (GPNULL)
- Workstation Dependent Output (GPWDO)

General Format

Every structure element as returned by the Inquire List of Element Data (**GPQED**) inquiry subroutine has the following data format:



The actual element body may contain more data than is given in the specified structure elements that follow this introduction. Any data that is undocumented is used internally by the graPHIGS API. You should always use the Inquire List of Element Headers (**GPQEHD**) subroutine to determine the actual length of the structure element. Use the length returned from the inquiry to calculate the size of your buffer (to contain the data), and to access the position of the next structure element.

length A halfword integer containing the length, in bytes, of the structure element. The length value includes this field and the ″code″ field.

code A halfword integer uniquely identifying the structure element. The following value ranges are used:

<= 0 Reserved

1-255 Structure elements other than output primitives

256-511
Output primitive structure elements

>= 512 Reserved

Within each value range, codes for individual structure elements are chosen in an arbitrary manner. The actual code values for individual elements are shown following.

data A sequence of parameters defining the structure element. Parameters specified by the application through the graPHIGS API may be reorganized but are stored in this field without changing their values except in the following cases:

- When the parameter is required to show the data format of a parameter in the API but its data format in the structure element is always fixed. A typical example of this case is the *width* parameter specifying the distance between individual elements in a list type parameter. It is required for the API but is not required for the structure element because no extra data is inserted between the elements in the structure element format. Such a parameter is discarded and is not stored in the element.
- When the parameter is used to show the length of a list type parameter. Because the list length in the structure element can usually be determined from the length of the structure element itself, the ″number of entries″ parameter is not required for the structure element. A typical example of this case is the *npoint* parameter for the Polyline primitive. Such a parameter is discarded.
- When the parameter includes a vector and its length has no significance. To increase the performance of structure traversal, such a vector may be normalized by the API. A typical example of this case is the ″character up vector″ for the text primitives.

Structure Element Codes

The following table shows the mnemonics used for the structure element codes and their actual values. The inquiry **GPQED** returns the value as part of the header.

Table 123. Mnemonics

Mnemonic	Dec	Hex	Description
GPAAL	37	0025	Set Annotation Text Alignment
GPADCN	226	00E2	Add Class Name to Set
GPAH	34	0022	Set Annotation Text Height
GPAHSC	33	0021	Set Annotation Text Height Scale Factor
GPAID	82	0052	Set Antialiasing Identifier
GPAN2	266	010A	Annotation Text 2
GPAN3	265	0109	Annotation Text 3
GPANR2	270	010E	Annotation Text Relative 2
GPANR3	269	010D	Annotation Text Relative 3
GPAPT	36	0024	Set Annotation Text Path
GPAS	32	0020	Set Annotation Style
GPASF	53	0035	Set Attribute Source Flag
GPAUP	35	0023	Set Annotation Text Up Vector
GPBBLF	104	0068	Set Back Blending Function
GPBDFM	111	006F	Set Back Data Filtering Method
GPBDMF	116	0074	Set Back Data Morphing Factors
GPBDMI	109	006D	Set Back Data Mapping Index
GPBDM2	113	0071	Set Back Data Matrix 2

Table 123. Mnemonics (continued)

Mnemonic	Dec	Hex	Description
GPBICD	64	0040	Set Back Interior Color Direct
GPBICI	63	003F	Set Back Interior Color Index
GPBISM	107	006B	Set Back Interior Shading Method
GPBLF	103	0067	Set Blending Function
GPBRMO	105	0069	Set Back Reflectance Model
GPBSCD	68	0044	Set Back Specular Color Direct
GPBSCI	67	0043	Set Back Specular Color Index
GPBSPR	71	0047	Set Back Surface Properties
GPBTCO	102	0066	Set Back Transparency Coefficient
GPCAC	76	004C	Set Curve Approximation Criteria
GPCEXS	254	00FE	Conditional Execute Structure
GPCFA2	308	0134	Composite Fill Area 2
GPCHH	19	0013	Set Character Height
GPCHLS	39	0027	Set Character Line Scale Factor
GPCHL2	304	0130	Character Line 2
GPCHPM	22	0016	Set Character Positioning Mode
GPCHSP	17	0011	Set Character Spacing
GPCHUB	38	0026	Set Character Up and Base Vectors
GPCHUP	20	0014	Set Character Up Vector
GPCHXP	16	0010	Set Character Expansion Factor
GPCOND	243	00F3	Set Condition
GPCPI	7	0007	Set Color Processing Index
GPCRA2	274	0112	Circular Arc 2
GPCRET	240	00F0	Conditional Return
GPCR2	273	0111	Circle 2
GPDCI	6	0006	Set Depth Cue Index
GPDFM	110	006E	Set Data Filtering Method
GPDMF	115	0073	Set Data Morphing Factors
GPDMI	108	006C	Set Data Mapping Index
GPDM2	112	0070	Set Data Matrix 2

Table 123. Mnemonics (continued)

Mnemonic	Dec	Hex	Description
GPDPL2	260	0104	Disjoint Polyline 2
GPDPL3	259	0103	Disjoint Polyline 3
GPECD	44	002C	Set Edge Color Direct
GPECI	29	001D	Set Edge Color Index
GPEF	27	001B	Set Edge Flag
GPEI	4	0004	Set Edge Index
GPELA2	283	011B	Elliptical Arc 2
GPELA3	282	011A	Elliptical Arc 3
GPELT	28	001C	Set Edge Linetype
GPEL2	281	0119	Ellipse 2
GPEL3	280	0118	Ellipse 3
GPESC	30	001E	Set Edge Scale Factor
GPEXST	250	00FA	Execute Structure
GPFBC	50	0032	Set Frame Buffer Comparison
GPFBM	49	0031	Set Frame Buffer Protect Mask
GPFDMO	72	0048	Set Face Distinguish Mode
GPFLM	84	0054	Set Face Lighting Method
GPGLX2	211	00D3	Set Global Transformation 2
GPGLX3	210	00D2	Set Global Transformation 3
GPHID	74	004A	Set HLHSR Identifier
GPHLCD	225	00E1	Set Highlighting Color Direct
GPHLCI	224	00E0	Set Highlighting Color Index
GPICD	43	002B	Set Interior Color Direct
GPICI	26	001A	Set Interior Color Index
GPII	5	0005	Set Interior Index
GPINAD	228	00E4	Insert Application Data
GPINLB	251	00FB	Insert Label
GPIS	24	0018	Set Interior Style
GPISI	25	0019	Set Interior Style Index
GPISM	106	006A	Set Interior Shading Method
GPLG2	296	0128	Line Grid 2
GPLG3	295	0127	Line Grid 3
GPLLCD	117	0075	Set Line-on-Line Color Direct
GPLLCI	118	0076	Set Line-on-Line Color Index
GPLMO	79	004F	Set Lighting Calculation Mode
GPLSS	73	0049	Set Light Source State

Table 123. Mnemonics (continued)

Mnemonic	Dec	Hex	Description
GPLT	8	0008	Set Linetype
GPLWSC	9	0009	Set Linewidth Scale Factor
GPMCI	214	00D6	Set Modeling Clipping Indicator
GPMCV2	213	00D5	Set Modeling Clipping Volume 2
GPMCV3	212	00D4	Set Modeling Clipping Volume 3
GPMG2	294	0126	Marker Grid 2
GPMG3	293	0125	Marker Grid 3
GPMLX2	209	00D1	Set Modeling Transformation 2
GPMLX3	208	00D0	Set Modeling Transformation 3
GPMSSC	12	000C	Set Marker Size Scale Factor
GPMT	11	000B	Set Marker Type
GPNBC2	279	0117	Non-uniform B-Spline Curve 2
GPNBC3	278	0116	Non-uniform B-Spline Curve 3
GPNBS	305	0131	Non-Uniform B-Spline Surface
GPNULL	229	00E5	Null Data
GPPGC	69	0045	Set Polygon Culling
GPPGD2	300	012C	Polygon 2 With Data
GPPGD3	299	012B	Polygon 3 With Data
GPPG2	290	0122	Polygon 2
GPPG3	289	0121	Polygon 3
GPPHE	309	0135	Polyhedron Edge
GPPHEC	78	004E	Set Polyhedron Edge Culling
GPPKID	252	00FC	Set Pick Identifier
GPPLCD	40	0028	Set Polyline Color Direct
GPPLCI	10	000A	Set Polyline Color Index
GPPLD3	318	013E	Polyline Set 3 with Data
GPPLET	31	001F	Set Polyline End Type
GPPLI	1	0001	Set Polyline Index
GPPL2	258	0102	Polyline 2
GPPL3	257	0101	Polyline 3
GPPLSM	98	0062	Set Polyline Shading Method

Table 123. Mnemonics (continued)

Mnemonic	Dec	Hex	Description
GPPMCD	41	0029	Set Polymarker Color Direct (GPPMCD)
GPPMCI	13	000D	Set Polymarker Color Index
GPPMI	2	0002	Set Polymarker Index
GPPM2	262	0106	Polymarker 2
GPPM3	261	0105	Polymarker 3
GPPSC	81	0051	Set Parametric Surface Characteristics
GPPXL2	272	0110	Pixel 2
GPPXL3	271	010F	Pixel 3
GPQM3	320	0140	Quadrilateral Mesh 3
GPRCN	227	00E3	Remove Class Name from Set
GPRMCV	215	00D7	Restore Modeling Clipping Volume
GPRMO	99	0063	Set Reflectance Model
GPSAC	77	004D	Set Surface Approximation Criteria
GPSCD	66	0042	Set Specular Color Direct
GPSCI	65	0041	Set Specular Color Index
GPSPH	312	0138	Polysphere
GPSPR	70	0046	Set Surface Properties
GPTCAC	80	0050	Set Trimming Curve Approximation Criteria
GPTCO	101	0065	Set Transparency Coefficient
GPTEX2	242	00F2	Test Extent 2
GPTEX3	241	00F1	Test Extent 3
GPTNBS	306	0132	Trimmed Non-uniform B-Spline Surface
GPTS3	301	012D	Triangle Strip 3
GPTXAL	23	0017	Set Text Alignment
GPTXCD	42	002A	Set Text Color Direct
GPTXCI	18	0012	Set Text Color Index
GPTXFO	14	000E	Set Text Font
GPTXI	3	0003	Set Text Index
GPTXPR	15	000F	Set Text Precision
GPTXPT	21	0015	Set Text Path
GPTX2	264	0108	Geometric Text 2
GPTX3	263	0107	Geometric Text 3
GPVMF	114	0072	Set Vertex Morphing Factors
GPVWI	216	00D8	Set View Index

Table 123. Mnemonics (continued)

Mnemonic	Dec	Hex	Description
GPWDO	246	00F6	Workstation Dependent Output
GPZBM	85	0055	Set Z-buffer Protect Mask

Common Data Types

This section provides a generic listing of possible data types for structure element parameters. The Output Primitives section refers to the data types described here in order to describe the specific data type of a particular structure element parameter.

Point 3

```
----------------------
|       x            | Short floating-point number
----------------------
|       y            | Short floating-point number
----------------------
|       z            | Short floating-point number
----------------------
```

Point 2

```
----------------------
|       x            | Short floating-point number
----------------------
|       y            | Short floating-point number
----------------------
```

Vector 3

```
----------------------
|       x            | Short floating-point number
----------------------
|       y            | Short floating-point number
----------------------
|       z            | Short floating-point number
----------------------
```

Vector 2

```
----------------------
|       x            | Short floating-point number
----------------------
|       y            | Short floating-point number
----------------------
```

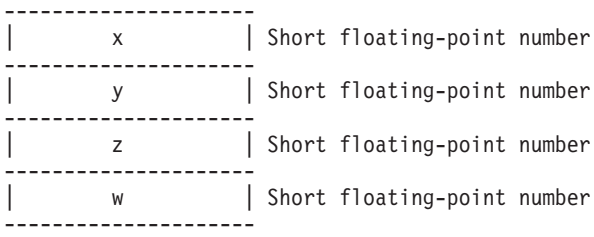
Control point 3

The format of this data type depends on the rationality flag, bit 15, of the *cflag*/*sflag* parameter within the primitive.

0 NON-RATIONAL

```
----------------------
|       x            | Short floating-point number
----------------------
|       y            | Short floating-point number
----------------------
|       z            | Short floating-point number
----------------------
```

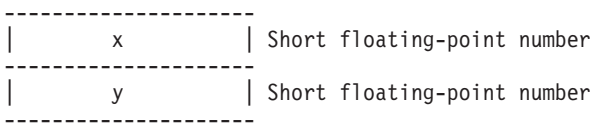
1 RATIONAL



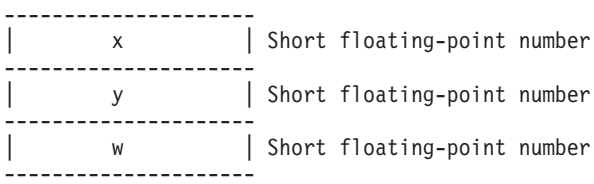
Control point 2

The format of this data type depends on the rationality flag, bit 15, of the *cflag* parameter within the primitive.

0 NON-RATIONAL



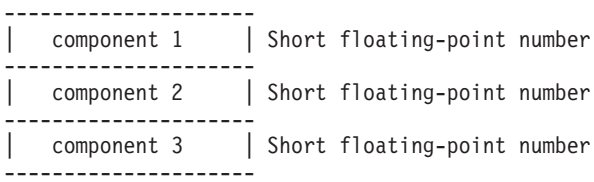
1 RATIONAL



Color

The format of this data type depends on the ″color type″ bit in the primitive's *gflag* parameter.

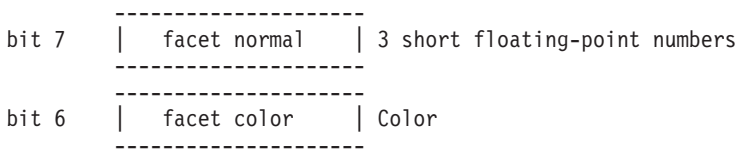
1 DIRECT



where 0.0 <= component *n* <= 1.0

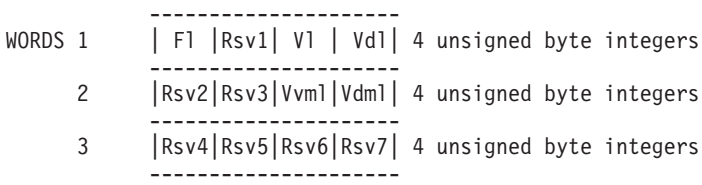
Facet 3

The format of this data type depends on the bit settings in the ″oflag″ field in the primitive. When the bit indicated to the left of each data item is 1, the data item is present. Otherwise, the corresponding data item is not present. All data items, if they are present, must be specified in the order shown below.



Data_Extension

The presence of this data type depends on a bit setting of the ″gflag″ field in the primitive. When bit 11 of the ″gflag″ is 1, then additional data is present after the ″oflag″.



Fl An unsigned byte counter that specifies the total number of words in the facet.

Rsv1 Reserved. Set to 0.

Vl An unsigned byte counter that specifies the total number of words in the vertex.

Vdl An unsigned byte counter that specifies the number of words of vertex data mapping data (Vdl <= Vl).

Rsv2 Reserved. Set to 0.

Rsv3 Reserved. Set to 0.

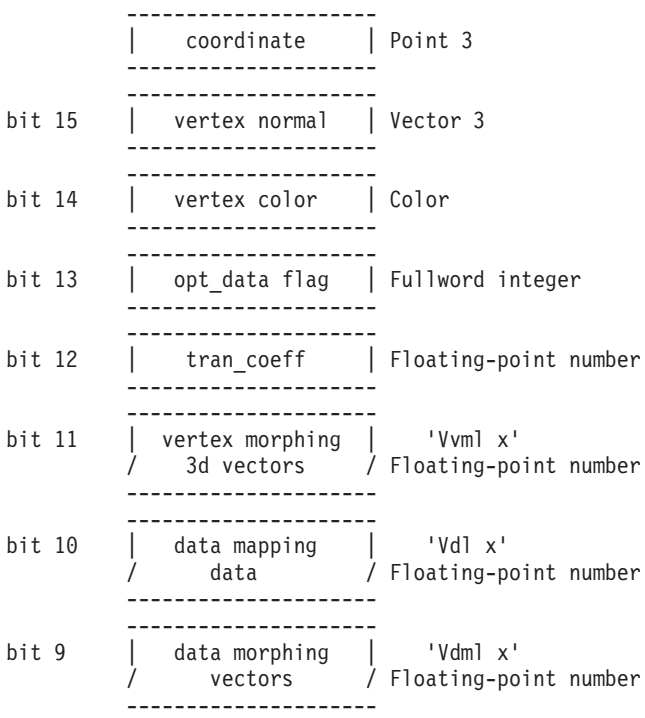
Vvml An unsigned byte counter that specifies the number of words of vertex morphing vectors (Vvml <= Vl).

Vdml An unsigned byte counter that specifies the number of words of data morphing vectors (Vvml <= Vl).

Rsv4, Rsv5, Rsv6, Rsv7 Reserved. Set to 0.

Vertex 3

The format of this data type depends on the bit settings of the ″oflag″ field in the primitive. When the bit indicated to the left of each data item is 1, the data item is present. Otherwise, the corresponding data item is not present. The data item without any corresponding bit is always present. All data items, if they are present, must be specified in the order shown below.



Vertex 2

The format of this data type depends on bit setting of the ″oflag″ field in the primitive. When the bit indicated in the left of each data item is 1, the data item is present. Otherwise, the corresponding data item is not present. The data item without any corresponding bit is always present. All data items, if they are present, must be specified in the order shown below.

```
                  ---------------------
                  |    coordinate     | Point 2
                  ---------------------
                  ---------------------
          bit 14  |    vertex color   | Color
                  ---------------------
                  ---------------------
          bit 13  |   opt_data flag   | Fullword integer
                  ---------------------
                  ---------------------
          bit 12  |    tran_coeff     | Floating-point number
                  ---------------------
                  ---------------------
          bit 11  |  vertex morhping  |     'Vvml x'
                  /   2d vectors      / Floating-point number
                  ---------------------
                  ---------------------
          bit 10  |   data mapping    |     'Vdl x'
                  /      data         / Floating-point number
                  ---------------------
                  ---------------------
          bit 9   |  data morphing    |     'Vdml x'
                  /     vectors       / Floating-point number
                  ---------------------
```

Notes:

1. The format of the ″vertex color″ field is determined by the ″color type″ bit of the *gflag* parameter.
2. The ″opt_data flag″ is a 32-bit integer which specifies additional information for this vertex depending on the area primitive defined.

bit 0-29	Reserved
bit 30	Edge indicator: 0 = NOT_AN_EDGE, 1 = IS_AN_EDGE
bit 31	Edge indicator: 0 = NOT_AN_EDGE, 1 = IS_AN_EDGE

When the value of bit 30 or bit 31 is 1, a line starting from this vertex is treated as a geometric edge. Bit 30 is used by area primitives that require only one edge indicator flag per vertex. Quadrilateral Mesh 3 and Triangle Strip 3 are the only area primitives that require two edge indicators per vertex. Therefore, for the Quadrilateral Mesh 3 primitive, bit 30 is the edge indicator from specified vertex ″V″ ($_{i, j}$) to vertex ″V″ ($_{i + 1, j}$), and bit 31 is the edge indicator from specified vertex ″V″ (sub i,j) to vertex ″V″ ($_{i, j + 1}$). For the Triangle Strip 3 primitive, bit 30 is the edge indicator from specified vertex ″V″ ($_{i}$) to vertex ″V″ ($_{i + 1}$), and bit 31 is the edge indicator from specified vertex ″V″ ($_{i}$) to vertex ″V″ ($_{i + 2}$)

Edge indicators are used only when the current global edge flag is set to `ON` or `GEOMETRY_ONLY`

Output Primitives

Line Primitives

Polyline 3 (GPPL3)

This structure element defines a series of three-dimensional points that are to be connected by straight lines. The *width* parameter is discarded when creating the structure element.

```
                  ---------------------
        WORDS 1   | length | X'0101'  | Element header
                  ---------------------
```

```
        2     |                   | Array of short
              /  X,Y,Z,X,Y,Z,X... / floating-point numbers
              /                   /
              ---------------------
```

Polyline 2 (GPPL2)

This structure element defines a series of two-dimensional points that are to be connected by straight lines. The *width* parameter is discarded when creating the structure element.

```
              ---------------------
      WORDS 1 | length | X'0102'  | Element header
              ---------------------
        2     |                   | Array of short
              /   X,Y,X,Y,X ...   / floating-point numbers
              /                   /
              ---------------------
```

Polyline Set 3 with Data (GPPLD3)

This structure element defines multiple three-dimensional polylines. The *plwidth*, *vxwidth*, and *pldata* parameters are discarded when creating the structure element. The API's *vxdata* parameter is used to build the ″seglist″ field and is then discarded. Bits in the API's *pflags, plflags,* and *vxflags* parameters are used to build the *oflag* field and are then discarded.

```
              ----------------------
      WORDS 1 | length | X'013E'   | Element header
              ----------------------
        2     | gflag  | oflag     | 2 halfword integers
              ----------------------
        3-5   |  data_extension*   | Data_extension
              ----------------------
              |     seglist        | n x Segment
              /                    /
              /                    /
              ----------------------


    Segment format

              ---------------------
              |      length       | Fullword integer
              ---------------------
              |      vlist        | m x Vertex 3
              /                   /
              /                   /
              ---------------------
```

* The data_extension field is present only if bit 11 of the gflag is 1.

Disjoint Polyline 3 (GPDPL3)

This structure element defines a series of three-dimensional points that are to be connected by straight lines in three-dimensional modeling space. The API's *npoint*, *width*, *pointlist* and *mdarray* parameters are used to build the ″seglist″ field and are then discarded.

```
              ---------------------
      WORDS 1 | length | X'0103'  | Element header
              ---------------------
        2     |                   | n x Segment
              /      seglist      /
              /                   /
              ---------------------

    Segment format
              ---------------------
              |      length       | Fullword integer
              ---------------------
```

```
                |                   | Array of short
                /  X,Y,Z,X,Y,Z,X... / floating-point numbers
                /                   /
                ---------------------
```

Disjoint Polyline 2 (GPDPL2)

This structure element defines a series of two-dimensional points that are to be connected by straight lines. The API's *npoint*, *width*, *pointlist* and *mdarray* parameters are used to build the ″seglist″ field and are then discarded.

```
                ---------------------
        WORDS 1 | length |  X'0104' | Element header
                ---------------------
              2 |      seglist      | n x segment
                /                   /
                /                   /
                ---------------------

      Segment format
                ---------------------
                |      length       | Fullword integer
                ---------------------
                |                   | Array of short
                /  X,Y,X,Y,X,Y,X... / floating-point numbers
                /                   /
                ---------------------
```

Circle 2 (GPCR2)

This structure element defines a circle in two-dimensional modeling space.

```
                ---------------------
        WORDS 1 | length | X'0111'  | Element header
                ---------------------
              2-3 |      center       | 2 short floating-point numbers
                ---------------------
              4 |      radius       | Short floating-point number
                ---------------------
```

Circular Arc 2 (GPCRA2)

This structure element defines a circular arc in two-dimensional modeling space.

```
                ---------------------
        WORDS 1 | length | X'0112'  | Element header
                ---------------------
              2-3 |      center       | 2 short floating-point numbers
                ---------------------
              4 |      radius       | Short floating-point number
                ---------------------
              5 |      startang     | Short floating-point number
                ---------------------
              6 |      endang       | Short floating-point number
                ---------------------
```

Ellipse 3 (GPEL3)

This structure element defines an ellipse in three-dimensional modeling space.

```
                ---------------------
        WORDS 1 | length | X'0118'  | Element header
                ---------------------
              2-4 |      center       | 3 short floating-point numbers
                ---------------------
              5-7 |       refv1       | 3 short floating-point numbers
                ---------------------
             8-10 |       refv2       | 3 short floating-point numbers
                ---------------------
```

Ellipse 2 (GPEL2)

This structure element defines an ellipse in two-dimensional modeling space.

```
                ---------------------
WORDS 1         | length | X'0119'  | Element header
                ---------------------
      2-3       |      center       | 2 short floating-point numbers
                ---------------------
      4-5       |       refv1       | 2 short floating-point numbers
                ---------------------
      6-7       |       refv2       | 2 short floating-point numbers
                ---------------------
```

Elliptical Arc 3 (GPELA3)

This structure element defines an elliptical arc in three-dimensional modeling space.

```
                ---------------------
WORDS 1         | length | X'011A'  | Element header
                ---------------------
      2-4       |      center       | 3 short floating-point numbers
                ---------------------
      5-7       |       refv1       | 3 short floating-point numbers
                ---------------------
      8-10      |       refv2       | 3 short floating-point numbers
                ---------------------
      11        |       startv      | Short floating-point number
                ---------------------
      12        |        endv       | Short floating-point number
                ---------------------
```

Elliptical Arc 2 (GPELA2)

This structure element defines an elliptical arc in two-dimensional modeling space.

```
                ---------------------
WORDS 1         | length | X'011B'  | Element header
                ---------------------
      2-3       |      center       | 2 short floating-point numbers
                ---------------------
      4-5       |       refv1       | 2 short floating-point numbers
                ---------------------
      6-7       |       refv2       | 2 short floating-point numbers
                ---------------------
      8         |       startv      | Short floating-point number
                ---------------------
      9         |        endv       | Short floating-point number
                ---------------------
```

Line Grid 3 (GPLG3)

This structure element defines a grid of lines in the plane defined by *point*, *refv1*, and *refv2* in three-dimensional modeling space.

```
                ---------------------
WORDS 1         | length | X'0127'  | Element header
                ---------------------
      2-4       |      point        | 3 short floating-point numbers
                ---------------------
      3-7       |      refv1        | 3 short floating-point numbers
                ---------------------
      8-10      |      refv2        | 3 short floating-point numbers
                ---------------------
      11        |      imin         | Fullword integer
                ---------------------
      12        |      imax         | Fullword integer
                ---------------------
```

```
        13    |      jmin         | Fullword integer
              ---------------------
        14    |      jmax         | Fullword integer
              ---------------------
```

Line Grid 2 (GPLG2)

This structure element defines a grid of lines in the plane defined by *point*, *refv1*, and *refv2* in two-dimensional modeling space.

```
              ---------------------
  WORDS 1     | length | X'0128'  | Element header
              ---------------------
        2-3   |      point        | 2 short floating-point numbers
              ---------------------
        4-5   |      refv1        | 2 short floating-point numbers
              ---------------------
        6-7   |      refv2        | 2 short floating-point numbers
              ---------------------
        8     |      imin         | Fullword integer
              ---------------------
        9     |      imax         | Fullword integer
              ---------------------
        10    |      jmin         | Fullword integer
              ---------------------
        11    |      jmax         | Fullword integer
              ---------------------
```

Non-uniform B-Spline Curve 3 (GPNBC3)

This structure element defines a non-uniform B-spline curve of the specified order using the specified coefficients in three-dimensional modeling space.

```
              ---------------------
  WORDS 1     | length | X'0116'  | Element header
              ---------------------
        2     | cflag  |  tflag   | 2 halfword integers
              ---------------------
        3     |      order        | Fullword integer
              ---------------------
        4     |      npoint       | Fullword integer
              ---------------------
        5     |      tmin         | Short floating-point number
              ---------------------
        6     |      tmax         | Short floating-point number
              ---------------------
        7     |      iknot        | 'order' x Iknot
              /                   /
              /                   /
              ---------------------
              |      ipoint       | ('order'-1) x Ipoint
              /                   /
              /                   /
              ---------------------
              |      spanlist     | ('npoint'-'order'+1) x span
              /                   /
              /                   /
              ---------------------

Iknot format
              ---------------------
              |      knot         | Array of short floating-point numbers
              ---------------------

Ipoint format
              ---------------------
              |      cpoint       | Control point 3
              ---------------------
              |      knot         | Array of short floating-point numbers
```

```
                    ---------------------

      Span format
                    ---------------------
                    |      cpoint       | Control point 3
                    ---------------------
```

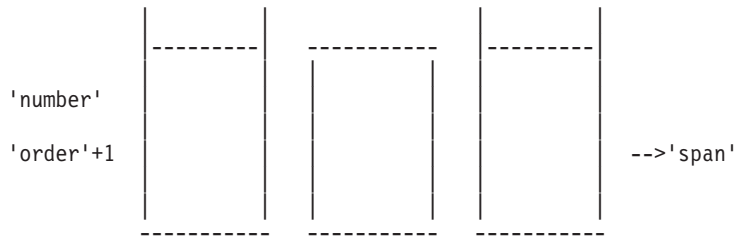
Non-uniform B-Spline Curve 2 (GPNBC2)

This structure element defines a non-uniform B-spline curve of the specified order using the specified coefficients in two-dimensional modeling space.

```
                    ---------------------
        WORDS 1     | length | X'0117'  | Element header
                    ---------------------
              2     | cflag  |  tflag   | 2 halfword integers
                    ---------------------
              3     |      order        | Fullword integer
                    ---------------------
              4     |      npoint       | Fullword integer
                    ---------------------
              5     |      tmin         | Short floating-point number
                    ---------------------
              6     |      tmax         | Short floating-point number
                    ---------------------
              7     |      iknot        | 'order' x iknot
                    /                   /
                    /                   /
                    ---------------------
                    |      ipoint       | ('order'-1) x ipoint
                    /                   /
                    /                   /
                    ---------------------
                    |      spanlist     | ('npoint'-'order'+1) x span
                    /                   /
                    /                   /
                    ---------------------

      Iknot format
                    ---------------------
                    |      knot         | Array of short floating-point numbers
                    ---------------------

      Ipoint format
                    ---------------------
                    |      cpoint       | Control point 2
                    ---------------------
                    |      knot         | Array of short floating-point numbers
                    ---------------------

      Span format
                    ---------------------
                    |      cpoint       | Control point 2
                    ---------------------
```

The relation between the API parameters and fields in this structure element is shown by the following picture.

```
          Control points Tessellation   Knot values

                                       -----------
                                       |         |
 'order'                               |         |  -->'iknot'
                                       |         |
                                       |         |
           -----------                 |---------|
           |         |                 |         |
 'order'-1 |         |                 |         |  -->'ipoint'
```



Polyhedron Edge (GPPHE)

This structure element simulates the edge of a polyhedron type object.

```
                  ---------------------
    WORDS 1       | length | X'0135'  | Element header
                  ---------------------
          2       |      edgelist     | 'nedge' x edge
                  /                   /
                  /                   /
                  ---------------------

     Edge format
                  ---------------------
                  |      normal#1     | 3 short floating-point numbers
                  ---------------------
                  |      normal#2     | 3 short floating-point numbers
                  ---------------------
                  |      vertex#1     | 3 short floating-point numbers
                  ---------------------
                  |      vertex#2     | 3 short floating-point numbers
                  ---------------------
```

Note: The API's *nedge* parameter is discarded after being used to build the ″edgelist″ field.

Marker Primitives

Polymarker 3 (GPPM3)

This structure element defines a series of points that are to be identified with markers in three-dimensional modeling space.

```
                  ---------------------
       WORDS 1    | length | X'0105'  | Element header
                  ---------------------
             2    |     pointlist     | 'npoint' x Point 3
                  /                   /
                  /                   /
                  ---------------------
```

Note: The API's *npoint*, *width* and *pointlist* parameters are used to build the ″plist″ field and are then discarded.

Polymarker 2 (GPPM2)

This structure element defines a series of points that are to be identified with markers in two-dimensional modeling space.

```
                  ---------------------
       WORDS 1    | length | X'0106'  | Element header
                  ---------------------
             2    |     pointlist     | 'npoint' x Point 2
                  /                   /
                  /                   /
                  ---------------------
```

Note: The API's *npoint*, *width* and *pointlist* parameters are used to build the ″plist″ field and are then discarded.

Marker Grid 3 (GPMG3)

This structure element defines a grid of markers in the plane defined by *point*, *refv1*, and *refv2* in three-dimensional modeling space.

WORDS 1	length	X'0125'	Element header
2-4	point		3 short floating-point numbers
3-7	refv1		3 short floating-point numbers
8-10	refv2		3 short floating-point numbers
11	imin		Fullword integer
12	imax		Fullword integer
13	jmin		Fullword integer
14	jmax		Fullword integer

Marker Grid 2 (GPMG2)

This structure element defines a grid of markers in the plane defined by *point*, *refv1*, and *refv2* in two-dimensional modeling space.

WORDS 1	length	X'0126'	Element header
2-3	point		2 short floating-point numbers
4-5	refv1		2 short floating-point numbers
6-7	refv2		2 short floating-point numbers
8	imin		Fullword integer
9	imax		Fullword integer
10	jmin		Fullword integer
11	jmax		Fullword integer

Annotation Text Primitives

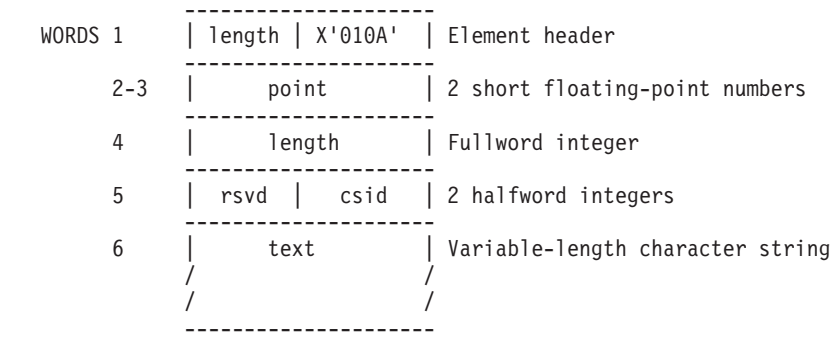
Annotation Text 3 (GPAN3)

This structure element defines an annotation text string in three-dimensional modeling space.

WORDS 1	length	X'0109'	Element header
2-4	point		3 short floating-point numbers
5	length		Fullword integer
6	rsvd	csid	2 halfword integers
7	text		Variable-length character string

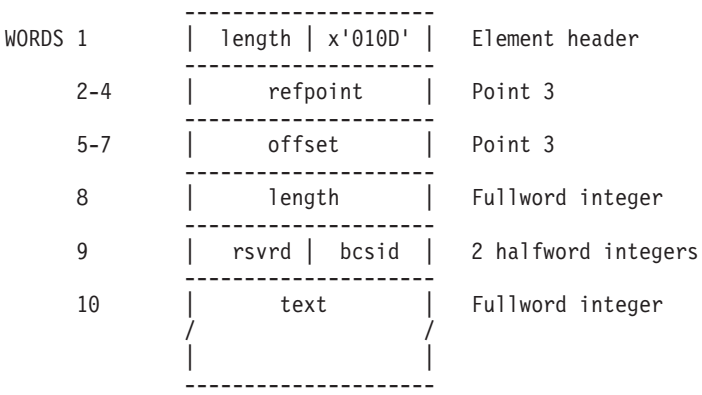
Annotation Text 2 (GPAN2)

(Ref #91.) This structure element defines an annotation text string in two-dimensional modeling space.



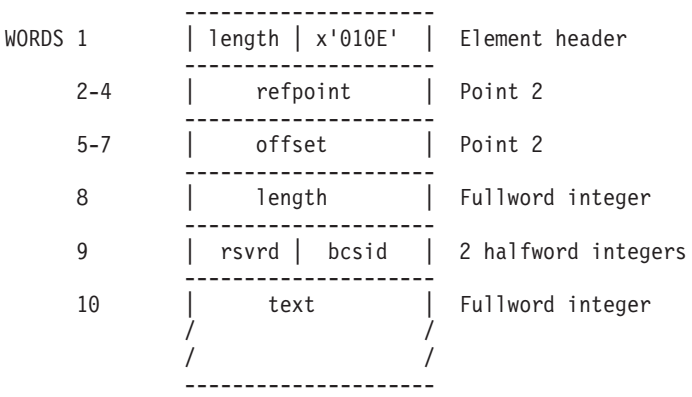
Annotation Text Relative 3 (GPANR3)

This structure element defines an annotation text string in three-dimensional normalized projection coordinates and annotates a specified reference point according to the annotation style in the traversal state list.



Annotation Text Relative 2 (GPANR2)

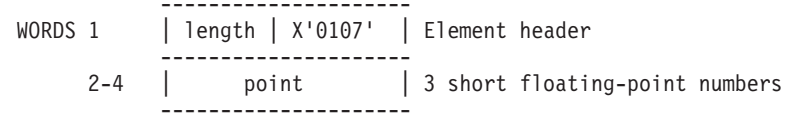
This structure element defines an annotation text string in two-dimensional normalized projection coordinates and annotates a specified reference point according to the annotation style in the traversal state list.



Geometric Text Primitives

Geometric Text 3 (GPTX3)

This structure element defines a geometric text string in three-dimensional modeling space.



```
      5-7   |      refvx        | 3 short floating-point numbers
            ---------------------
      8-10  |      refvd        | 3 short floating-point numbers
            ---------------------
      11-13 |      refvy        | 3 short floating-point numbers
            ---------------------
      14    |      length       | Fullword integers
            ---------------------
      15    |  rsvd  |  csid    | 2 halfword integers
            ---------------------
      16    |       text        | Variable-length character string
            /                   /
            /                   /
            ---------------------
```

Geometric Text 2 (GPTX2)

This structure element defines a geometric text string in two-dimensional modeling space.

```
            ---------------------
WORDS 1     | length | X'0108'  | Element header
            ---------------------
      2-3   |      point        | 2 short floating-point numbers
            ---------------------
      4     |      length       | Fullword integer
            ---------------------
      5     |  rsvd  |  csid    | 2 halfword integers
            ---------------------
      6     |      text         | Variable-length character string
            /                   /
            /                   /
            ---------------------
```

Character Line 2 (GPCHL2)

This structure element is used to define an integral number of a specific character along a line between two specified points.

```
              ---------------------
WORDS 1       | length | X'0130'  | Element header
              ---------------------
        2-3   |      startp       | 2 short floating-point numbers
              ---------------------
        4-5   |      endp         | 2 short floating-point numbers
              ---------------------
        6-7   |      refvx        | 2 short floating-point numbers
              ---------------------
        8-9   |      refvy        | 2 short floating-point numbers
              ---------------------
        10    |      distance     | Short floating-point number
              ---------------------
        11    |      nomhgt       | Short floating-point number
              ---------------------
        12    | X'0000' | csid    | 2 halfword integers
              ---------------------
        13    |char|      pad     | Character code
              ---------------------
```

Note: Character code is left-adjusted in the ″char″ field and the element is padded to a word boundary.

Area Primitives

Polygon 3 (GPPG3)

This structure element defines a polygon in three-dimensional modeling space.

```
              ---------------------
WORDS 1       | length | X'0121'  | Element header
              ---------------------
```

```
        2     |  gflag | reserved | 2 halfword integers
              ---------------------
        3     |     contlist      | 'areas' x contour
              /                   /
              /                   /
              ---------------------

Contour format
              ---------------------
              |      contlen      | Fullword integer
              ---------------------
              |      plist        | m x Point 3
              /                   /
              /                   /
              ---------------------
```

Note: The *pointlist* parameter is reorganized into the *contlist*. The *areas*, *width* and *npoint* parameter are discarded after being used.

Polygon 2 (GPPG2)

This structure element defines a polygon in two-dimensional modeling space.

```
              ---------------------
  WORDS 1     | length | X'0122'  | Element header
              ---------------------
        2     |  gflag | reserved | 2 halfword integers
              ---------------------
        3     |     contlist      | 'areas' x contour
              /                   /
              /                   /
              ---------------------

Contour format
              ---------------------
              |      contlen      | Fullword integer
              ---------------------
              |      plist        | m x Point 2
              /                   /
              /                   /
              ---------------------
```

Note: The *pointlist* parameter is reorganized into the *contlist*. The *areas*, *width* and *npoint* parameter are discarded after being used.

Polygon 3 With Data (GPPGD3)

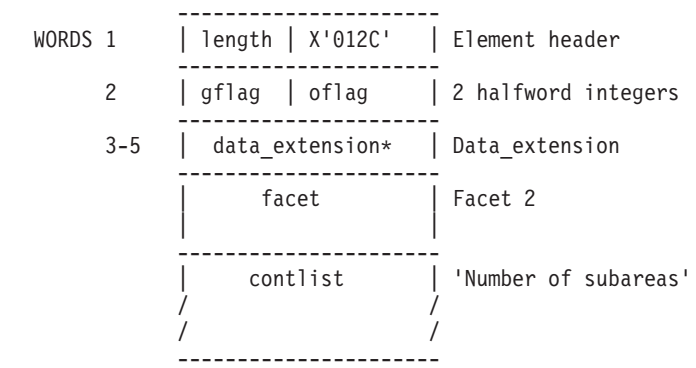
This structure element defines a polygon with the specified number of subareas in three-dimensional modeling space.

```
              ----------------------
  WORDS 1     | length | X'012B'   | Element header
              ----------------------
        2     | gflag  |  oflag    | 2 halfword integers
              ----------------------
        3-5   |  data_extension*   | Data_extension
              ----------------------
              |      facet         | Facet 3
              |                    |
              ----------------------
              |     contlist       | 'Number of subareas'
              /                    /
              /                    /
              ----------------------

Contour format
              ---------------------
              |      contlen      | Fullword integer
```

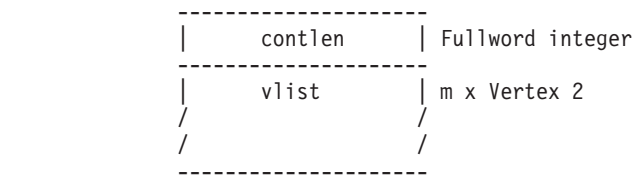
* The data_extension field is present only if bit 11 of the gflag is 1.

Polygon 2 With Data (GPPGD2)

This structure element defines a polygon with the specified number of subareas in two-dimensional modeling space.



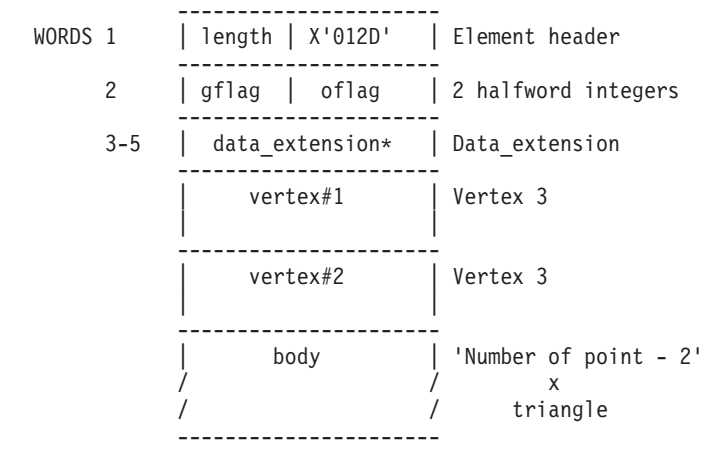
Contour format



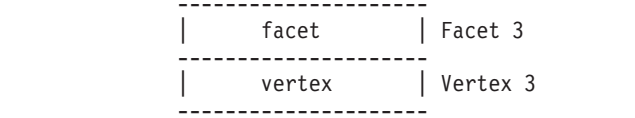
* The data_extension field is present only if bit 11 of the gflag is 1.

Triangle Strip 3 (GPTS3)

This structure element defines (*n-2*) triangles from *n* vertexes in three-dimensional modeling space.



Triangle format



* The data_extension field is present only if bit 11 of the gflag is 1.

Quadrilateral Mesh 3 (GPQM3)

This structure element defines (*m-1* [default] *n-1*) quadrilaterals from a two-dimensional array of *m* [default] *n* vertices in three-dimensional modeling space.

```
                ----------------------
  WORDS 1       | length | X'0140'   | Element header
                ----------------------
        2       | gflag  |  oflag    | 2 halfword integers
                ----------------------
        3-5     |  data_extension*   | Data_extension
                ----------------------
                |     row_dim        | Fullword integer
                ----------------------
                |     col_dim        | Fullword integer
                ----------------------
                |      vlist         | row_dim x Vertex 3
                /                    /
                /                    /
                ----------------------
                |    quad_rows       | (col_dim-1) x Quad_row
                /                    /
                /                    /
                ----------------------

Quad_row format
                ----------------------
                |      vertex        | Vertex 3
                ----------------------
                |      quads         | (row_dim-1) x Quad
                /                    /
                ----------------------

Quad format
                ----------------------
                |      facet         | Facet 3
                ----------------------
                |      vertex        | Vertex 3
                ----------------------
```

* The data_extension field is present only if bit 11 of the gflag is 1.

Non-Uniform B-Spline Surface (GPNBS)

This structure element defines a Non-Uniform B-Spline Surface of the specified *u* and *v* orders using the specified control points and knots.

```
                ----------------------
  WORDS 1       | length | X'0131'   | Element header
                ----------------------
        2       | sfflag |  stflag   | 2 halfword integers
                ----------------------
        3       |     u-order        | Fullword integer
                ----------------------
        4       |     unumber        | Fullword integer
                ----------------------
        5       |     vorder         | Fullword integer
                ----------------------
        6       |     vnumber        | Fullword integer
                ----------------------
        7       |     umin           | Short floating-point number
                ----------------------
        8       |     umax           | Short floating-point number
                ----------------------
        9       |     vmin           | Short floating-point number
                ----------------------
        10      |     vmax           | Short floating-point number
```

```
                ---------------------
          11    |     uknots        | 'u-number' + 'u-order'
                /                   /            x
                /                   / short floating-point number
                ---------------------
                |     utess         | 'u-number' - 'u-order'+1
                /                   /            x
                |                   | short floating-point number
                ---------------------
                |     vknots        | 'v-number' + 'v-order'
                /                   /            x
                |                   | short floating-point number
                ---------------------
                |     vtess         | 'v-number' - 'v-order'+1
                /                   /            x
                |                   | short floating-point number
                ---------------------
                |     clist         | 'u-number' x 'v-number'
                /                   /            x
                |                   | Control point 3
                ---------------------
```

Composite Fill Area 2 (GPCFA2)

This structure element defines the planar area geometry defined by the specified contours using polygon attributes.

```
                ---------------------
        WORDS 1 | length | X'0134'  | Element header
                ---------------------
              2 |  gflag | reserved | 2 halfword integers
                ---------------------
              3 |     contlist      | 'ncontour' x contour
                /                   /
                /                   /
                ---------------------

      Contour format
                ---------------------
                |      contlen      | Fullword integer
                ---------------------
                |      seglist      | 'ncurve' x segment
                /                   /
                /                   /
                ---------------------

      Segment format
                ---------------------
                |      seglen       | Fullword integer
                ---------------------
                | sgflag |  sgtype  | 2 halfword integers
                ---------------------
                |      segdef       | Variable data
                /                   /
                /                   /
                ---------------------
```

Trimmed Non-uniform B-Spline Surface (GPTNBS)

This structure element defines a trimmed non-uniform B-spline surface of the specified *u* and *v* orders using the specified control points and knots.

```
                ---------------------
        WORDS 1 | length | X'0132'  | Element header
                ---------------------
              2 | sflag  |  stflag  | 2 halfword integers
                ---------------------
              3 |     uorder        | Fullword integer
                ---------------------
```

```
         4     |     unumber       | Fullword integer
               ---------------------
         5     |     vorder        | Fullword integer
               ---------------------
         6     |     vnumber       | Fullword integer
               ---------------------
         7     |     reserved1     | Fullword integer
               ---------------------
         8     |     reserved2     | Fullword integer
               ---------------------
         9     |     trimming      | Variable-length field
               /                   /
               ---------------------
               |     uknots        | 'u-number' + 'u-order'
               /                   /           x
               /                   / short floating-point number
               ---------------------
               |     utess         | 'u-number' - 'u-order'+1
               /                   /           x
               /                   / short floating-point number
               ---------------------
               |     vknots        | 'v-number' + 'v-order'
               /                   /           x
               /                   / short floating-point number
               ---------------------
               |     vtess         | 'v-number' - 'v-order'+1
               /                   /           x
               /                   / short floating-point number
               ---------------------
               |     clist         | 'u-number' x 'v-number'
               /                   /          x
               /                   / Control point 3
               ---------------------

Trimming format
               ---------------------
               |      trimlen      | Fullword integer
               ---------------------
               |      contlist     | 'ncontour' x contour
               /                   /
               /                   /
               ---------------------
Contour format
               ---------------------
               |      contlen      | Fullword integer
               ---------------------
               |      seglist      | 'ncurve' x segment
               /                   /
               /                   /
               ---------------------
Segment format
               ---------------------
               |      seglen       | Fullword integer
               ---------------------
               | sgflag |  sgtype  | 2 halfword integers
               ---------------------
               |      segdef       | Variable data
               /                   /
               /                   /
               ---------------------
```

Polysphere (GPSPH)

This structure element defines a sphere or a sequence of spheres in modeling space.

```
               ---------------------
        WORDS 1     | length | X'0138'  | Element header
               ---------------------
              2     |    reserved       | Fullword integer
```

```
                ---------------------
          3     |    spherelist     |
                /                   /
                /                   /
                ---------------------
Spherelist format
                ---------------------
                |       point       | 3 short floating-point numbers
                ---------------------
                |       radius      | Short floating-point number
                ---------------------
```

Pixel Primitives

Pixel 3 (GPPXL3)

This structure element defines a three-dimensional rectangular array of pixels in modeling space.

```
                ---------------------
 WORDS 1        | length | X'010F'  | Element header
                ---------------------
       2-4      |      point        | 3 short floating-point numbers
                ---------------------
       5        |      nrows        | Fullword integer
                ---------------------
       6        |      ncols        | Fullword integer
                ---------------------
       7        |      pixels       | 'nrows' x 'ncols'
                /                   / one-byte unsigned integers
                /                   /
                ---------------------
```

Pixel 2 (GPPXL2)

This structure element defines a two-dimensional rectangular array of pixels in modeling space.

```
                ---------------------
 WORDS 1        | length | X'0110'  | Element header
                ---------------------
       2-4      |      point        | 2 short floating-point numbers
                ---------------------
       5        |      nrows        | Fullword integer
                ---------------------
       6        |      ncols        | Fullword integer
                ---------------------
       7        |      pixels       | 'nrows' x 'ncols'
                /                   / one-byte unsigned integers
                /                   /
                ---------------------
```

Attribute Setting Structure Elements

General Attributes

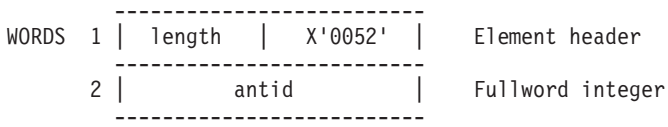
Set HLHSR Identifier (GPHID)

This structure element specifies how each geometric entity should be processed in the hidden line, hidden surface removal process.

```
                ---------------------
 WORDS 1        | length | X'004A'  | Element header
                ---------------------
       2        |      HLHSRid      | Fullword integer
                ---------------------
```

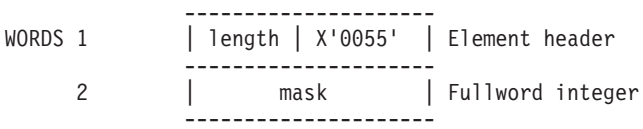
Set Antialiasing Identifier (GPAID)

This structure element sets whether antialiasing is performed for consecutive primitives within a view depending on the antialiasing mode setting for the view.



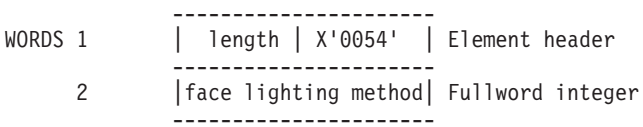
Set Z-buffer Protect Mask (GPZBM)

This structure element defines a mask used to enable or disable updates of the Z-buffer.



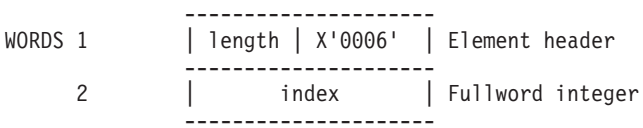
Set Face Lighting Method (GPFLM)

This structure element sets the current face lighting method. Subsequent lighting calculations are effected by this setting.



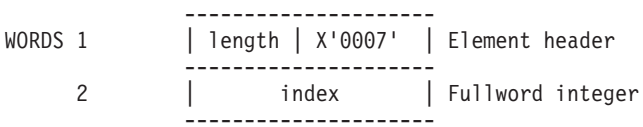
Set Depth Cue Index (GPDCI)

This structure element supplies depth cue information to the workstation. The attributes defined in the entry control the depth cueing applied to subsequent primitives.



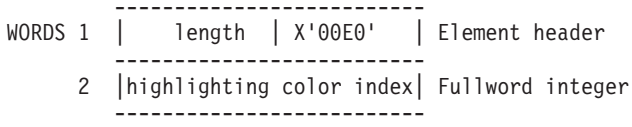
Set Color Processing Index (GPCPI)

This structure element sets an index into the color processing table on the workstation. The values in the entry are used when creating output primitives.



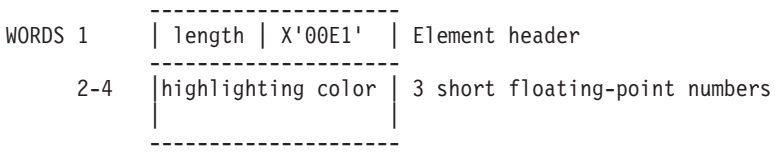
Set Highlighting Color Index (GPHLCI)

This structure element sets the index into the workstation-dependent color table, which is used for highlighted primitives.



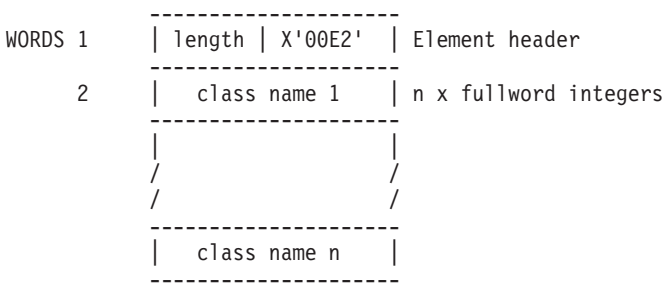
Set Highlighting Color Direct (GPHLCD)

This structure element sets the direct color values to be used to render subsequent highlighted primitives.



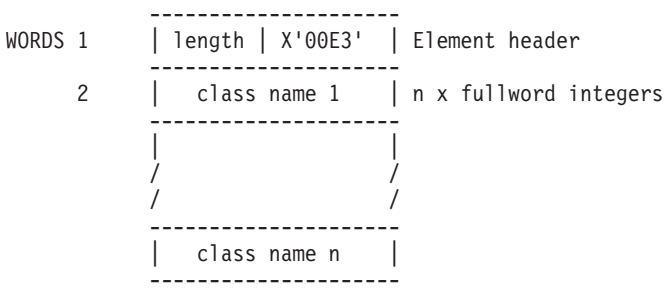
Add Class Name to Set (GPADCN)

This structure element allows an application to define the eligibility of a primitive for pickability (detectability), highlighting, and invisibility by associating it with a set of class names. During structure traversal, the specified class names are added to the current class set.



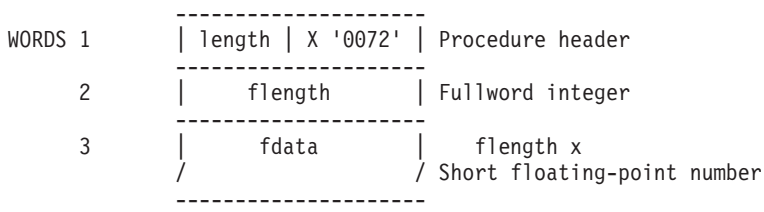
Remove Class Name from Set (GPRCN)

This structure element allows an application to define the eligibility of a primitive for pickability (detectability), highlighting, and invisibility by associating it with a set of class names. During structure traversal, the specified class names are removed from the current class set.



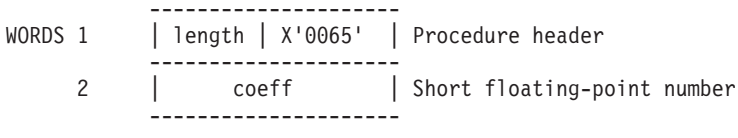
Set Vertex Morphing Factors (GPVMF)

This structure element specifies vertex morphing factors which affect morphing of subsequent primitives supplied with vertex morphing vectors.



Set Transparency Coefficient (GPTCO)

This structure element specifies the source transparency coefficient used to blend subsequent primitives with previously rendered output.



Set Blending Function (GPBLF)

This structure element specifies the source blending function and the destination blending function used to blend subsequent primitives with previously rendered output.

```
              --------------------
WORDS 1       | length | X'0067' |  Procedure header
              --------------------
      2       |      srcf        |  Fullword integer
              --------------------
      3       |     destf        |  Fullword integer
              --------------------
```

Set Line-on-Line Color Direct (GPLLCD)

This structure element specifies the direct color values to be used when highlighting using the Frame Buffer Comparison option `WRITE_WHEN_NOT_EQUAL`.

```
              ---------------------
WORDS 1       | length | X'0075'  | Element header
              ---------------------
      2       |   components      | 3 floating point numbers
              ---------------------
```

Set Line-on-Line Color Index (GPLLCI)

This structure element specifies an entry in the workstation's rendering color table that contains the color to be used when highlighting using the Frame Buffer Comparison option `WRITE_WHEN_NOT_EQUAL`.

```
              ---------------------
WORDS 1       | length | X'0076'  | Element header
              ---------------------
      2       |      index        | Fullword integer
              ---------------------
```

Attribute Selection

Set Attribute Source Flag (GPASF)

This structure element defines whether a particular attribute used for rendering a primitive should be from the `BUNDLED` or `CURRENT_INDIVIDUAL` attribute setting.

```
              ---------------------
WORDS 1       | length | X'0035'  | Element header
              ---------------------
      2       |     asflist       | n x ASFspec
              /                   /
              /                   /
              ---------------------

 ASFspec format
---------------------------------------
|attribute identifier |  ASF value    |  2 halfword integers
---------------------------------------
```

Polyline Attributes

Set Curve Approximation Criteria (GPCAC)

This structure element determines how curves are to be tessellated for subsequent curve primitives during structure traversal.

```
              ---------------------
WORDS 1       | length | X'004C'  |  Element header
              ---------------------
      2       |      criteria     |  Fullword integer
              ---------------------
      3       |      cvalue       |  Short floating-point number
              ---------------------
```

Set Trimming Curve Approximation Criteria (GPTCAC)

This structure element enables the application to control the tessellation of the trimming curve as well as the surface in the area of the curve when rendering subsequent trimming surface primitives.

WORDS 1	length	X'0050'	Element header
2	criteria		Fullword integer
3	cvalue		Short floating-point number
4	uvalue		Short floating-point number
5	vvalue		Short floating-point number

Set Polyhedron Edge Culling (GPPHEC)

This value supplies edge culling information to the workstation, and is used when drawing polyhedron edge output primitives.

WORDS 1	length	X'004E'	Element header
2	mode		Fullword integer

Set Polyline Index (GPPLI)

(Ref #92.) This structure element sets the current polyline bundle index to the specified value. All subsequent polyline primitives use the contents of the specified bundle table entry for all polyline attributes whose attribute source flag is set to BUNDLED.

WORDS 1	length	X'0001'	Element header
2	polyline bundle index		Fullword integer

Set Linetype (GPLT)

This structure element sets the current line type to the specified value. All subsequent polyline primitives use this line type for drawing the primitive if the line type attribute source flag is set to INDIVIDUAL.

WORDS 1	length	X'0008'	Element header
2	linetype		Fullword integer

Set Polyline End Type (GPPLET)

This structure element sets the polyline end type to the specified value.

WORDS 1	length	X'001F'	Element header
2	endtype		Fullword integer

Set Linewidth Scale Factor (GPLWSC)

This structure element sets the current line width scale factor to the specified value. All subsequent polyline primitives use this value to determine the line width of lines to be drawn if the attribute source flag is set to INDIVIDUAL.

WORDS 1	length	X'0009'	Element header
2	linewidth scale factor		Short floating-point number

Set Polyline Color Index (GPPLCI)

This structure element sets the current polyline color index to the specified value. All subsequent polyline primitives use this color index for drawing the primitive if the polyline color index attribute source flag is set to INDIVIDUAL.

```
              ------------------------
WORDS 1       |  length    |  X'000A' | Element header
              ------------------------
      2       | polyline color index | Fullword integer
              ------------------------
```

Set Polyline Color Direct (GPPLCD)

This structure element is used when drawing polyline output primitives. All subsequent polyline primitives use this value to determine the color of the output primitives if the polyline color attribute source flag is set to INDIVIDUAL.

```
              ---------------------
WORDS 1       | length | X'0028'  | Element header
              ---------------------
      2-4     |  polyline color   | 3 short floating-point numbers
              |                   |
              ---------------------
```

Set Polyline Shading Method (GPPLSM)

This structure element sets the current polyline shading method. This shading method is used at structure traversal time to render all subsequent Polyline with Data primitives with vertex colors defined. The vertex colors are interpolated through connecting polylines when the polyline shading method is 2=POLYLINE_SHADING_COLOR. The i% sup th vertex color is used to color the i% sup th line when the polyline shading method is 1=POLYLINE_SHADING_NONE.

```
              ---------------------
WORDS 1       | length | X'0062'  | Element header
              ---------------------
      2       |     method        | Fullword integer
              ---------------------
```

Polymarker Attributes

Set Polymarker Index (GPPMI)

This structure element sets the current polymarker bundle index to the specified value. All subsequent polymarker primitives use the contents of the specified bundle table entry for all polymarker attributes whose attribute source flags are set to BUNDLED.

```
             -------------------------
WORDS 1      |    length  | X'0002'   | Element header
             -------------------------
      2      | polymarker bundle index| Fullword integer
             -------------------------
```

Set Marker Type (GPMT)

This structure element sets the current marker type. All subsequent polymarker primitives use this marker type for identifying each point if the marker type attribute source flag is set to INDIVIDUAL.

```
              ---------------------
WORDS 1       | length | X'000B'  | Element header
              ---------------------
      2       |    marker type    | Fullword integer
              ---------------------
```

Set Marker Size Scale Factor (GPMSSC)

This structure element sets the current marker size scale factor. All subsequent polymarker primitives use this value to determine the size to draw the markers if the attribute source flag is set to INDIVIDUAL.

```
          --------------------------
 WORDS 1  |   length   |  X'000C'   | Element header
          --------------------------
       2  |marker size scale factor | Short floating-point number
          --------------------------
```

Set Polymarker Color Index (GPPMCI)

This structure element sets the current polymarker color index to the specified value. All subsequent polymarker primitives use this color index for drawing the primitive if the polymarker color index attribute source flag is set to INDIVIDUAL.

```
             --------------------------
 WORDS 1     |   length   |   X'000D' | Element header
             --------------------------
       2     | polymarker color index | Fullword integer
             --------------------------
```

Set Polymarker Color Direct (GPPMCD)

This structure element is used when drawing polymarker output primitives. All subsequent polymarker primitives use this value for drawing the primitive if the polymarker color attribute source flag is set to INDIVIDUAL.

```
              ---------------------
 WORDS 1      | length | X'0029'  | Element header
              ---------------------
       2-4    | polymarker color  | 3 short floating-point numbers
              |                   |
              ---------------------
```

Text Attributes

Set Character Height (GPCHH)

This structure element sets the current character height. All subsequent geometric text primitives will be drawn with this value for the character height.

```
              ---------------------
 WORDS 1      | length | X'0013'  | Element header
              ---------------------
       2      | character height  | Short floating-point number
              ---------------------
```

Set Character Line Scale Factor (GPCHLS)

This structure element sets the value to be used to determine the height of the characters when rendering all subsequent character line primitives.

```
              ---------------------
 WORDS 1      | length | X'0027'  | Element header
              ---------------------
       2      | line scale factor | Short floating-point number
              ---------------------
```

Set Character Up Vector (GPCHUP)

This structure element sets the current geometric text character up vector to the specified value. The base vector is set to 90[default] clockwise from the up vector. All subsequent geometric text primitives are drawn with this value for the character up vector. The character up vector specifies the direction of the font coordinate y-axis within the text reference coordinate system. The character base vector specifies the direction of the font coordinate x-axis with the text reference coordinate system.

```
             ---------------------
WORDS 1      |  length  | X'0014'| Element header
             ---------------------
      2      |    X direction    | Short floating-point number
             ---------------------
      3      |    Y direction    | Short floating-point number
             ---------------------
```

Set Character Up and Base Vectors (GPCHUB)

This structure element sets the current geometric text character up vector and base vector to the specified value. The character up vector specifies the direction of the font coordinate y-axis within the text reference coordinate system. The character base vector specifies the direction of the font coordinate x-axis with the text reference coordinate system.

```
                ---------------------
        WORDS 1 | length | X'0026'  | Element header
                ---------------------
              2 |  up X-direction   | Short floating-point number
                ---------------------
              3 |  up Y-direction   | Short floating-point number
                ---------------------
              4 | base X-direction  | Short floating-point number
                ---------------------
              5 | base Y-direction  | Short floating-point number
                ---------------------
```

Set Text Path (GPTXPT)

This structure element sets the current geometric text path to the specified value. All subsequent geometric text primitives are drawn with this value for the text path.

```
                ---------------------
      WORDS 1   | length |  X'0015' | Element header
                ---------------------
            2   |     text path     | Fullword integer
                ---------------------
```

Set Text Alignment (GPTXAL)

This structure element sets the current geometric text alignment to the specified value. All subsequent geometric text primitives are drawn with this value for text alignment.

```
                ---------------------
        WORDS 1 | length | X'0017'  | Element header
                ---------------------
              2 |    horizontal     | Fullword integer
                ---------------------
              3 |    vertical       | Fullword integer
                ---------------------
```

Set Character Positioning Mode (GPCHPM)

This structure element sets the current character positioning mode entry to the specified value. The character positioning mode determines whether the character positioning box for the specific character or the nominal positioning box for the font should be used in rendering annotation and geometric text primitives.

```
                ---------------------
        WORDS 1 | length | X'0016'  | Element header
                ---------------------
              2 |       mode        | Fullword integer
                ---------------------
```

Set Text Index (GPTXI)

This structure element sets the current text bundle index to the specified value. All subsequent annotation and geometric text primitives use the contents of the specified bundle table entry for all text attributes whose attribute source flags are set to BUNDLED.

```
                ---------------------
        WORDS 1 | length | X'0003'  | Element header
                ---------------------
              2 | text bundle index | Fullword integer
                ---------------------
```

Set Text Font (GPTXFO)

This structure element sets the current text font to the specified value. All subsequent annotation and geometric text primitives are drawn in this font if the text font attribute source flag is set to INDIVIDUAL.

```
              ---------------------
WORDS 1       | length | X'000E'  | Element header
              ---------------------
      2       |    text font      | Fullword integer
              ---------------------
```

Set Text Precision (GPTXPR)

This structure element sets the current text precision to the specified value. All subsequent annotation and geometric text primitives are drawn at this precision if the text precision attribute source flag is set to INDIVIDUAL.

```
              ---------------------
WORDS 1       | length | X'000F'  | Element header
              ---------------------
      2       |  text precision   | Fullword integer
              ---------------------
```

Set Text Linewidth Scale Factor (GPTLWS)

This structure element specifies the width of a geometric text primitive's lines (strokes) as a fraction of the nominal text width. The device support multiplies this scale factor times the nominal linewidth on the corresponding device to determine the requested width. The calculated value is then mapped to the closest width available on the device. A scale factor of 1.0 generates a nominal size text line on any workstation. At structure traversal, this scale factor is used, when the text line width factor ASF is set to INDIVIDUAL.

```
              ---------------------
WORDS 1       | length | X'0077'  | Element header
              ---------------------
      2       |     lwidth        | Floating point
              --------------------- number
```

Set Character Expansion Factor (GPCHXP)

This structure element sets the current character expansion factor. It indicates the deviation of the character's width/height ratio from the font default. All subsequent annotation and geometric text primitives are drawn with this value for the character expansion factor if its attribute source flag is set to INDIVIDUAL.

```
            ------------------------------
WORDS 1     |     length    |  X'0010'    | Element header
            ------------------------------
      2     | character expansion factor | Short floating-point number
            ------------------------------
```

Set Character Spacing (GPCHSP)

This structure element sets the current character spacing, indicating the additional amount of space to be placed between characters as a fraction of the character's design. All subsequent annotation and geometric text primitives are drawn with this value for the character spacing if its attribute source flag is set to INDIVIDUAL.

```
              ---------------------
WORDS 1       | length | X'0011'  | Element header
              ---------------------
      2       | character spacing | Short floating-point number
              ---------------------
```

Set Text Color Index (GPTXCI)

This structure element sets the text color index to the specified value. All subsequent annotation and geometric text primitives use this color index for drawing the primitive if the text color index attribute source flag is set to INDIVIDUAL.

```
                ---------------------
      WORDS 1   | length | X'0012'  | Element header
                ---------------------
            2   | text color index  | Fullword integer
                ---------------------
```

Set Text Color Direct (GPTXCD)

This structure element sets the current text color to the specified value. All subsequent annotation and geometric text primitives use the direct color values for drawing the primitive if the text color attribute source flag is set to INDIVIDUAL.

```
                ---------------------
      WORDS 1   | length | X'002A'  | Element header
                ---------------------
            2-4 |    text color     | 3 short floating-point numbers
                |                   |
                ---------------------
```

Annotation Text Attributes

Set Annotation Text Height Scale Factor (GPAHSC)

This structure element sets the current annotation height scale factor, specifying a ratio of the annotation character height to the workstation's nominal character height. All subsequent annotation text primitives are drawn with this value for the height scale factor.

```
                ---------------------
      WORDS 1   | length | X'0021'  | Element header
                ---------------------
            2   | annotation height | Short floating-point number
                |  scale factor     |
                ---------------------
```

Set Annotation Text Height (GPAH)

This structure element sets the current annotation height to the specified value.

```
                ---------------------
      WORDS 1   | length | X'0022'  | Element header
                ---------------------
            2   | annotation height | Short floating-point number
                ---------------------
```

Set Annotation Style (GPAS)

This structure element sets the text style determining how the Annotation Text Relative 2/3 primitives are to be visualized. This style value is used at the structure traversal time to render all subsequent annotation text relative primitives.

```
         -----------------------------
 WORDS 1 |  length     |   X'0020'   |  Element header
         -----------------------------
       2 |         style             |  Fullword integer
         -----------------------------
```

Set Annotation Text Up Vector (GPAUP)

This structure element sets the current annotation text character up vector to the specified value.

```
                ---------------------
      WORDS 1   | length | X'0023'  | Element header
                ---------------------
            2   |    X direction    | Short floating-point number
                ---------------------
            3   |    Y direction    | Short floating-point number
                ---------------------
```

Set Annotation Text Path (GPAPT)

This structure element sets the current annotation text path to the specified value.

```
              ---------------------
WORDS 1       | length | X'0024'  | Element header
              ---------------------
      2       | annotation path   | Fullword integer
              ---------------------
```

Set Annotation Text Alignment (GPAAL)

This structure element sets the current annotation text alignment to the specified value, affecting the manner in which the annotation text extent rectangle is positioned in relation to the text position.

```
              ---------------------
WORDS 1       | length | X'0025'  | Element header
              ---------------------
      2       |    horizontal     | Fullword integer
              ---------------------
      3       |    vertical       | Fullword integer
              ---------------------
```

Polygon Attributes

Set Surface Approximation Criteria (GPSAC)

This structure element sets the current surface approximation criteria to the specified value.

```
              ---------------------
WORDS 1       | length | X'004D'  | Element header
              ---------------------
      2       |      criteria     | Fullword integer
              ---------------------
      3       |      uvalue       | Short floating-point number
              ---------------------
      4       |      vvalue       | Short floating-point number
              ---------------------
```

Set Polygon Culling (GPPGC)

This structure element sets the current polygon culling mode to the specified value. This value is used when rendering polygon output primitives, and supplies polygon culling information to the workstation.

```
              ---------------------
WORDS 1       | length | X'0045'  | Element header
              ---------------------
      2       |       mode        | Fullword integer
              ---------------------
```

Interior Attributes

Set Face Distinguish Mode (GPFDMO)

This structure element sets the current face distinguish mode to the specified value.

```
              ---------------------
WORDS 1       | length | X'0048'  | Element header
              ---------------------
      2       |       mode        | Fullword integer
              ---------------------
```

Set Light Source State (GPLSS)

This structure element adds light source indices specified in the activation list to the current light source state and removes those in the deactivation list.

```
              ---------------------
WORDS 1       | length | X'0049'  | Element header
              ---------------------
      2       |      dnum         | Fullword integer
              ---------------------
      3       |      anum         | Fullword integer
              ---------------------
```

```
      4     |      dlist        | 'dnum' x fullword integers
            /                   /
            /                   /
            ---------------------
            |      alist        | 'anum' x fullword integers
            /                   /
            /                   /
            ---------------------
```

Set Lighting Calculation Mode (GPLMO)

This structure element sets the current lighting calculation mode to the specified value. This value is used when creating polygon output primitives and supplies lighting information to the workstation.

```
                  ---------------------
WORDS 1           | length | X'004F'  | Element header
                  ---------------------
      2           |       mode        | Fullword integer
                  ---------------------
```

Set Interior Index (GPII)

This structure element specifies an index into the interior bundle table, affecting only those attributes for which the ASF is set to BUNDLED. All subsequent polygon primitives use the contents of the specified bundle table entry for all interior attributes whose attribute source flags are set to BUNDLED.

```
                 ------------------------
WORDS 1          |  length  | X'0005'   | Element header
                 ------------------------
      2          |interior bundle index | Fullword integer
                 ------------------------
```

Set Interior Style (GPIS)

This structure element sets the current interior style to the specified value. This value is used when drawing polygon output primitives. All subsequent primitives use this attribute when drawing the interior if the corresponding attribute source flag is set to INDIVIDUAL.

```
                  ---------------------
WORDS 1           | length | X'0018'  |  Element header
                  ---------------------
      2           |      style        |  Fullword integer
                  ---------------------
```

Set Interior Style Index (GPISI)

This structure element sets the current interior style index, specifying an index into the pattern table or the workstation-dependent hatch table (depending on the setting of the interior style). All subsequent primitives use this attribute when drawing the interior if the corresponding attribute source flag is set to INDIVIDUAL and the current interior style is HATCH or PATTERN.

```
                  ---------------------
WORDS 1           | length | X'0019'  | Element header
                  ---------------------
      2           |       index       | Fullword integer
                  ---------------------
```

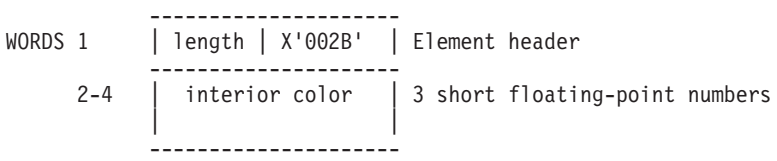
Set Interior Color Index (GPICI)

This structure element sets the current interior color index to the specified color index. All subsequent primitives use this attribute when drawing the interior if the corresponding attribute source flag is set to INDIVIDUAL and the current interior style is SOLID or HATCH.

```
                 ----------------------
WORDS 1          |  length | X'001A'  |  Element header
                 ----------------------
      2          |interior color index|  Fullword integer
                 ----------------------
```

Set Interior Color Direct (GPICD)

This structure element sets the current interior color to the specified value. It is used when creating output primitives. All subsequent primitives use this direct color value when drawing the interior if the corresponding attribute source flag is set to INDIVIDUAL and the current interior style is set to SOLID or HATCH.

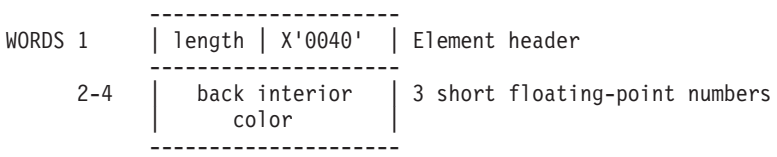


Set Back Interior Color Index (GPBICI)

This structure element sets the current back interior color index to the specified color index. All subsequent primitives use this attribute when drawing the back interior if the corresponding attribute source flag is set to INDIVIDUAL and the current interior style is set to SOLID or HATCH and face distinguish mode is set to use the back color attribute.

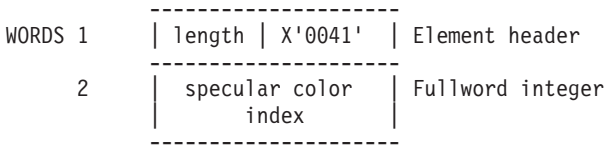
Set Back Interior Color Direct (GPBICD)

This structure element sets the current back interior color to the specified value. This value is used when drawing polygon output primitives. All subsequent primitives use this direct color value when drawing the back interior if the corresponding attribute source flag is set to INDIVIDUAL and the current interior style is set to SOLID or HATCH and face distinguish mode is set to use the back color attribute.



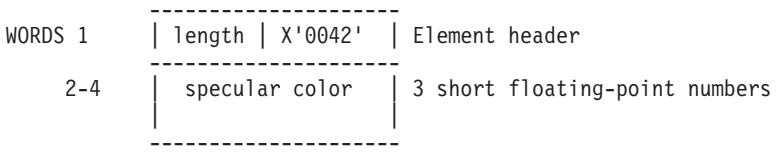
Set Specular Color Index (GPSCI)

This structure element sets the current specular color index to the specified color value for area defining geometries in lighting calculations.



Set Specular Color Direct (GPSCD)

This structure element sets the current specular color to the specified value for area defining geometries in lighting calculations.



Set Back Specular Color Index (GPBSCI)

This structure element sets the current back specular color index to the specified value for area defining geometries in lighting calculations.

```
                  ---------------------
WORDS 1           | length | X'0043'  | Element header
                  ---------------------
      2           |  back specular    | Fullword integer
                  |   color index     |
                  ---------------------
```

Set Back Specular Color Direct (GPBSCD)

This structure element sets the current back specular color to the specified value for area defining geometries in lighting calculations.

```
                  ---------------------
   WORDS 1        | length | X'0044'  | Element header
                  ---------------------
         2-4      |back specular color| 3 short floating-point numbers
                  |                   |
                  ---------------------
```

Set Surface Properties (GPSPR)

This structure element sets the current surface properties to the specified values.

```
                  ---------------------
   WORDS 1        | length | X'0046'  | Element header
                  ---------------------
         2        |      ambient      | Short floating-point number
                  ---------------------
         3        |      diffuse      | Short floating-point number
                  ---------------------
         4        |      specular     | Short floating-point number
                  ---------------------
         5        |      exponent     | Short floating-point number
                  ---------------------
         6        |      transparent  | Short floating-point number
                  ---------------------
```

Set Back Surface Properties (GPBSPR)

This structure element sets the current back surface properties to the specified values.

```
                  ---------------------
                  | length | X'0047'  | Element header
                  ---------------------
         2        |      ambient      | Short floating-point number
                  ---------------------
         3        |      diffuse      | Short floating-point number
                  ---------------------
         4        |      specular     | Short floating-point number
                  ---------------------
         5        |      exponent     | Short floating-point number
                  ---------------------
         6        |      transparent  | Short floating-point number
                  ---------------------
```

Set Back Transparency Coefficient (GPBTCO)

This structure element specifies the source transparency coefficient used to blend subsequent back facing portions of area primitives with previously rendered output.

```
                  ---------------------
   WORDS 1        | length | X'0066'  | Procedure header
                  ---------------------
         2        |      coeff        | Short floating-point number
                  ---------------------
```

Set Back Blending Function (GPBBLF)

This structure element specifies the source blending function and the destination blending function used to blend subsequent back facing portions of area primitives with previously rendered output.

```
              --------------------
WORDS 1       | length | X'0068' |  Procedure header
              --------------------
      2       |      srcf        |  Fullword integer
              --------------------
      3       |     destf        |  Fullword integer
              --------------------
```

Set Parametric Surface Characteristics (GPPSC)

This structure element sets the characteristics for rendering parametric surfaces in wireframe.

```
              --------------------
WORDS 1       | length | X'0051'  | Element header
              --------------------
      2       |        type      | Fullword integer
              --------------------

for type=ISOPARAMETRIC LINES

              --------------------
WORDS 1       |       scope      | Fullword integer
              --------------------
      2       |     number of    | Fullword integer
              |   isoparametrics |
              |   in u direction |
              --------------------
      3       |     number of    | Fullword integer
              |   isoparametrics |
              |   in v direction |
              --------------------
```

Set Data Morphing Factors (GPDMF)

This structure element supplies data morphing factors which affect morphing of subsequent primitives supplied with data morphing vectors.

```
              --------------------
WORDS 1       | length | X '0073' | Procedure header
              --------------------
      2       |     flength      | Fullword integer
              --------------------
      3       |      fdata       |    flength x
              /                  / Short floating-point number
              --------------------
```

Set Back Data Morphing Factors (GPBDMF)

This structure element supplies back data morphing factors which affect morphing of subsequent back facing portions of area primitives supplied with data morphing vectors.

```
              --------------------
WORDS 1       | length | X '0074' | Procedure header
              --------------------
      2       |     flength      | Fullword integer
              --------------------
      3       |      fdata       |    flength x
              /                  / Short floating-point number
              --------------------
```

Set Data Mapping Index (GPDMI)

This structure element specifies an entry in the workstation's data mapping table which contains values used to data map subsequent area primitives.

```
              --------------------
WORDS 1       | length | X '006C' | Procedure header
              --------------------
      2       |       index      | Fullword integer
              --------------------
```

Set Back Data Mapping Index (GPBDMI)

This structure element specifies an entry in the workstation's data mapping table which contains values used to data map subsequent back facing portions of area primitives.

WORDS 1	length	X '006D'	Procedure header
2	index		Fullword integer

Set Data Filtering Method (GPDFM)

This structure element specifies the filtering methods used when performing data mapping.

WORDS 1	length	X '006E'	Procedure header
2	minfm		Fullword integer
3	magfm		Fullword integer
4	boundu		Fullword integer
5	boundv		Fullword integer

Set Back Data Filtering Method (GPBDFM)

This structure element specifies the filtering methods used when performing data mapping. These values are used when rendering back facing portions of subsequent area primitives.

WORDS 1	length	X '006F'	Procedure header
2	minfm		Fullword integer
3	magfm		Fullword integer
4	boundu		Fullword integer
5	boundv		Fullword integer

Set Data Matrix 2 (GPDM2)

This structure element specifies a matrix used to modify the data mapping values specified in certain primitives that support data mapping.

WORDS 1	length	X'0070'	Procedure header
2-10	row 1 col 1 matrix element		Short floating-point number
	row 1 col 2 matrix element		Short floating-point number
	row 1 col 3 matrix element		Short floating-point number
	row 2 col 1 matrix element		Short floating-point number
	row 2 col 2 matrix element		Short floating-point number
	row 2 col 3 matrix element		Short floating-point number
	row 3 col 1 matrix element		Short floating-point number
	row 3 col 2 matrix element		Short floating-point number
	row 3 col 3 matrix element		Short floating-point number

Set Back Data Matrix 2 (GPBDM2)

This structure element specifies a matrix used to modify the data mapping values on back facing portions of certain primitives that support data mapping.

WORDS 1	length	X'0071'	Element header
2-10	row 1 col 1 matrix element		Short floating-point number
	row 1 col 2 matrix element		Short floating-point number
	row 1 col 3 matrix element		Short floating-point number
	row 2 col 1 matrix element		Short floating-point number
	row 2 col 2 matrix element		Short floating-point number
	row 2 col 3 matrix element		Short floating-point number
	row 3 col 1 matrix element		Short floating-point number
	row 3 col 2 matrix element		Short floating-point number
	row 3 col 3 matrix element		Short floating-point number

Set Reflectance Model (GPRMO)

This structure element specifies the method which is used to control the lighting calculations performed on subsequent area primitives.

WORDS 1	length	X '0063'	Element header
2	model		Fullword integer

Set Back Reflectance Model (GPBRMO)

This structure element specifies the method which is used to control the lighting calculations performed on back facing portions of subsequent area primitives.

WORDS 1	length	X '0069'	Element header
2	model		Fullword integer

Set Interior Shading Method (GPISM)

This structure element specifies the method to shade the interior of subsequent area primitives.

WORDS 1	length	X '006A'	Procedure header
2	method		Fullword integer

Set Back Interior Shading Method (GPBISM)

This structure element specifies the method to shade the interior of back facing portions of subsequent area primitives.

WORDS 1	length	X '006B'	Procedure header
2	method		Fullword integer

Edge Attributes

Set Edge Index (GPEI)

This structure element sets the edge bundle index to the specified value. All subsequent polygon primitives use the contents of the specified bundle table entry for all edge attributes whose attribute source flags are set to BUNDLED.

```
                ---------------------
WORDS 1         | length | X'0004'  | Element header
                ---------------------
      2         | edge bundle index | Fullword integer
                ---------------------
```

Set Edge Flag (GPEF)

This structure element sets the visibility of edges. All subsequent primitives use this attribute to determine whether the edges should be drawn if the attribute source flag is set to INDIVIDUAL.

```
                ---------------------
WORDS 1         | length | X'001B'  | Element header
                ---------------------
      2         |    edge flag      | Fullword integer
                ---------------------
```

Set Edge Linetype (GPELT)

This structure element sets the current edge line type to the specified value. All subsequent primitives use this edge line type for drawing the primitive if the edge line type attribute source flag is set to INDIVIDUAL and the edge flag is set to ON.

```
                ---------------------
WORDS 1         | length | X'001C'  | Element header
                ---------------------
      2         |  edge linetype    | Fullword integer
                ---------------------
```

Set Edge Scale Factor (GPESC)

This structure element sets the current edge scale factor to the specified value. All subsequent primitives use this value to determine the width of the edges to be drawn if the attribute source flag is set to INDIVIDUAL and the edge flag is set to ON.

```
                ---------------------
WORDS 1         | length | X'001E'  | Element header
                ---------------------
      2         | edge scale factor | Short floating-point number
                ---------------------
```

Set Edge Color Index (GPECI)

This structure element sets the current edge color index to the specified value. All subsequent primitives use this color index for drawing the edges if the edge color index attribute source flag is set to INDIVIDUAL and the edge flag is set to ON.

```
                ---------------------
WORDS 1         | length | X'001D'  | Element header
                ---------------------
      2         | edge color index  | Fullword integer
                ---------------------
```

Set Edge Color Direct (GPECD)

This structure element sets the current edge color entry to the specified value. All subsequent output primitives use the direct color values for drawing the edges if the edge color direct attribute source flag is set to INDIVIDUAL and the edge flag is set to ON.

```
                 --------------------
 WORDS 1         | length | X'002C'  | Element header
                 --------------------
       2-4       |    edge color     | 3 short floating-point numbers
                 |                   |
                 --------------------
```

Transformation Setting Structure Elements

Modeling Transformation

Set Global Transformation 3 (GPGLX3)

This structure element specifies a global modeling transformation in three-dimensional modeling space, causing the specified value to become the current global transformation for the current structure.

```
                 ----------------------------
 WORDS 1         |   X'0044'  | X'00D2'     | Element header
                 ----------------------------
       2-17      |row 1 col 1 matrix element| Short floating-point number
                 ----------------------------
                 |row 1 col 2 matrix element| Short floating-point number
                 ----------------------------
                 |row 1 col 3 matrix element| Short floating-point number
                 ----------------------------
                 |row 1 col 4 matrix element| Short floating-point number
                 ----------------------------
                 |row 2 col 1 matrix element| Short floating-point number
                 ----------------------------
                 |row 2 col 2 matrix element| Short floating-point number
                 ----------------------------
                 |row 2 col 3 matrix element| Short floating-point number
                 ----------------------------
                 |row 2 col 4 matrix element| Short floating-point number
                 ----------------------------
                 |row 3 col 1 matrix element| Short floating-point number
                 ----------------------------
                 |row 3 col 2 matrix element| Short floating-point number
                 ----------------------------
                 |row 3 col 3 matrix element| Short floating-point number
                 ----------------------------
                 |row 3 col 4 matrix element| Short floating-point number
                 ----------------------------
                 |row 4 col 1 matrix element| Short floating-point number
                 ----------------------------
                 |row 4 col 2 matrix element| Short floating-point number
                 ----------------------------
                 |row 4 col 3 matrix element| Short floating-point number
                 ----------------------------
                 |row 4 col 4 matrix element| Short floating-point number
                 ----------------------------
```

Set Global Transformation 2 (GPGLX2)

This structure element specifies a global modeling transformation in two-dimensional modeling space, causing the specified value to become the current global transformation for the current structure.

```
                 ----------------------------
 WORDS 1         |   X'0028'  | X'00D3'     | Element header
                 ----------------------------
       2-10      |row 1 col 1 matrix element| Short floating-point number
                 ----------------------------
                 |row 1 col 2 matrix element| Short floating-point number
                 ----------------------------
                 |row 1 col 3 matrix element| Short floating-point number
                 ----------------------------
                 |row 2 col 1 matrix element| Short floating-point number
```

```
            ----------------------------
            |row 2 col 2 matrix element| Short floating-point number
            ----------------------------
            |row 2 col 3 matrix element| Short floating-point number
            ----------------------------
            |row 3 col 1 matrix element| Short floating-point number
            ----------------------------
            |row 3 col 2 matrix element| Short floating-point number
            ----------------------------
            |row 3 col 3 matrix element| Short floating-point number
            ----------------------------
```

Set Modeling Transformation 3 (GPMLX3)

(Ref #93.) This structure element specifies a modification for a local modeling transformation in three-dimensional modeling space. The specified matrix either replaces, is pre-concatenated with, or is post-concatenated with the current local modeling transformation.

```
            ----------------------------
 WORDS 1    |   length    | X'00D0'    | Element header
            ----------------------------
       2    |     composition type     | Fullword integer
            ----------------------------
       3-18 |row 1 col 1 matrix element| Short floating-point number
            ----------------------------
            |row 1 col 2 matrix element| Short floating-point number
            ----------------------------
            |row 1 col 3 matrix element| Short floating-point number
            ----------------------------
            |row 1 col 4 matrix element| Short floating-point number
            ----------------------------
            |row 2 col 1 matrix element| Short floating-point number
            ----------------------------
            |row 2 col 2 matrix element| Short floating-point number
            ----------------------------
            |row 2 col 3 matrix element| Short floating-point number
            ----------------------------
            |row 2 col 4 matrix element| Short floating-point number
            ----------------------------
            |row 3 col 1 matrix element| Short floating-point number
            ----------------------------
            |row 3 col 2 matrix element| Short floating-point number
            ----------------------------
            |row 3 col 3 matrix element| Short floating-point number
            ----------------------------
            |row 3 col 4 matrix element| Short floating-point number
            ----------------------------
            |row 4 col 1 matrix element| Short floating-point number
            ----------------------------
            |row 4 col 2 matrix element| Short floating-point number
            ----------------------------
            |row 4 col 3 matrix element| Short floating-point number
            ----------------------------
            |row 4 col 4 matrix element| Short floating-point number
            ----------------------------
```

Set Modeling Transformation 2 (GPMLX2)

This structure element specifies a modification for a local modeling transformation in two-dimensional modeling space. The specified matrix either replaces, is pre-concatenated with, or is post-concatenated with the current local modeling transformation.

```
            ----------------------------
 WORDS 1    |   X'002C'  |  X'00D1'    | Element header
            ----------------------------
       2    |     composition type     | Fullword integer
            ----------------------------
       3-11 |row 1 col 1 matrix element| Short floating-point number
            ----------------------------
```

```
|row 1 col 2 matrix element| Short floating-point number
----------------------------
|row 1 col 3 matrix element| Short floating-point number
----------------------------
|row 2 col 1 matrix element| Short floating-point number
----------------------------
|row 2 col 2 matrix element| Short floating-point number
----------------------------
|row 2 col 3 matrix element| Short floating-point number
----------------------------
|row 3 col 1 matrix element| Short floating-point number
----------------------------
|row 3 col 2 matrix element| Short floating-point number
----------------------------
|row 3 col 3 matrix element| Short floating-point number
----------------------------
```

Set Modeling Clipping Indicator (GPMCI)

This structure element indicates whether or not to perform modeling clipping on subsequent primitives.

```
            ---------------------
WORDS 1     | length | X'00D6'  | Procedure header
            ---------------------
      2     |     indicator     | Fullword integer
            ---------------------
```

Restore Modeling Clipping Volume (GPRMCV)

This structure element causes the current modeling clipping volume in the traversal state list to be restored to the volume inherited by that structure.

```
            ---------------------
WORDS 1     | length | X'00D7'  | Procedure header
            ---------------------
```

Set Modeling Clipping Volume 3 (GPMCV3)

This structure element sets the current modeling clipping volume in the traversal state list.

```
            ---------------------
WORDS 1     | length | X'00D4'  | Procedure header
            ---------------------
      2     |     operator      | Fullword integer
            ---------------------
      3     |      number       | Fullword integer
            ---------------------
      4 - n |     lhspace       | number x half_space
            ---------------------
Half_space format
            ---------------------
            |      point        | Point 3
            ---------------------
            |      normal       | Vector 3
            ---------------------
```

Set Modeling Clipping Volume 2 (GPMCV2)

This structure element sets the current modeling clipping volume in the traversal state list.

```
            ---------------------
WORDS 1     | length | X'00D5'  | Procedure header
            ---------------------
      2     |     operator      | Fullword integer
            ---------------------
      3     |      number       | Fullword integer
            ---------------------
      4 - n |     lhspace       | number x half_space
            ---------------------
Half_space format
            ---------------------
```

```
        |      point        | Point 2
        ---------------------
        |      normal       | Vector 2
        ---------------------
```

Miscellaneous Structure Elements

View selection

Set View Index (GPVWI)

This structure element defines a view index to replace the current view index. The view index specifies an entry in the workstation's view table from which to select view orientation and mapping transformations.

```
              ---------------------
WORDS 1       | length | X'0008'  | Element header
              ---------------------
      2       |       index       | Fullword integer
              ---------------------
              |      reserved     | Fullword integer
              ---------------------
```

Traversal Control

Execute Structure (GPEXST)

This structure element defines a call or invocation of another structure, relating two structures.

```
              ---------------------
WORDS 1       | length | X'00FA'  | Element header
              ---------------------
      2       |   structure id    | Fullword integer
              ---------------------
      3       |     reserved      | Fullword integer
              ---------------------
      4       |     reserved      | Fullword integer
              ---------------------
```

Test Extent 3 (GPTEX3)

This structure element modifies the cull flag (30th bit) and the prune flag (31st bit) within the current set of condition flags. These flags are used when processing subsequent conditional execute structure elements and conditional return elements.

```
              ---------------------
WORDS 1       | length | X'00F1'  | Element header
              ---------------------
      2-4     |      corner1      | 3 short floating-point numbers
              |                   |
              ---------------------
      5-7     |      corner2      | 3 short floating-point numbers
              |                   |
              ---------------------
      8       |  cull table index | Fullword integer
              ---------------------
```

Test Extent 2 (GPTEX2)

This structure element modifies the cull flag (30th bit) and the prune flag (31st bit) within the current set of condition flags. These flags are used when processing subsequent conditional execute structure elements and conditional return elements.

```
              ---------------------
WORDS 1       | length | X'00F2'  | Element header
              ---------------------
      2-3     |      corner1      | 2 short floating-point numbers
              |                   |
              ---------------------
```

```
        4-5   |      corner2      | 2 short floating-point numbers
              |                   |
              ---------------------
        6     |  cull table index | Fullword integer
              ---------------------
```

Set Condition (GPCOND)

This structure element modifies the current condition flag with the specified value.

```
              ---------------------
WORDS 1       | length | X'00F3'  | Element header
              ---------------------
      2       |      onflag       | Fullword integer
              ---------------------
      3       |      offflag      | Fullword integer
              ---------------------
```

Conditional Execute Structure (GPCEXS)

This structure element specifies a conditional call or invocation of another structure. The current set of condition flags are tested against the specified *mask* and *condition*. If the condition is satisfied, the target structure is invoked.

```
              ---------------------
WORDS 1       | length | X'00FE'  | Element header
              ---------------------
      2       |      mask         | Fullword integer
              ---------------------
      3       |      condition    | Fullword integer
              ---------------------
      4       |      type         | Fullword integer
              ---------------------
      5       |   structure id    | Fullword integer
              ---------------------
      6       |      reserved     | Fullword integer
              ---------------------
      7       |      reserved     | Fullword integer
              ---------------------
```

Conditional Return (GPCRET)

This structure element specifies a conditional return to the parent structure.

```
              ---------------------
WORDS 1       | length | X'00F0'  | Element header
              ---------------------
      2       |      mask         | Fullword integer
              ---------------------
      3       |      condition    | Fullword integer
              ---------------------
```

Identification

Insert Label (GPINLB)

This structure element defines a label that the application uses to reference and modify structure elements.

```
              ---------------------
WORDS 1       | length | X'00FB'  | Element header
              ---------------------
      2       |      label        | Fullword integer
              ---------------------
      3       |      reserved     | Fullword integer
              ---------------------
```

Set Pick Identifier (GPPKID)

This structure element sets the current pick identifier to the specified value.

```
                ---------------------
WORDS 1         | length | X'00FC'  | Element header
                ---------------------
      2         |      pickid       | Fullword integer
                ---------------------
      3         |      reserved     | Fullword integer
                ---------------------
```

Frame Buffer Control

Set Frame Buffer Protect Mask (GPFBM)

This structure element sets the current frame buffer write protect mask to the specified value.

```
                ---------------------
WORDS 1         | length | X'0031'  | Element header
                ---------------------
      2         | write protect mask| Fullword integer
                ---------------------
```

Set Frame Buffer Comparison (GPFBC)

This structure element sets the current frame buffer comparison to the specified value.

```
                ---------------------
WORDS 1         | length | X'0032'  | Element header
                ---------------------
      2         |      type         | Fullword integer
                ---------------------
      3         |  comparison mask  | Fullword integer
                ---------------------
      4         |  comparison value | Fullword integer
                ---------------------
```

Application-Defined Data

Insert Application Data (GPINAD)

This structure element allows the insertion of application specific data into a structure element. The data is ignored during traversal. The element is padded to a fullword boundary following the application defined data.

```
                ---------------------
WORDS 1         | length | X'00E4'  | Element header
                ---------------------
      2         |      length       | Fullword integer
                ---------------------
      3         |   application     | Byte string
                /   defined data    /
                /                   /
                ---------------------
```

Null Data (GPNULL)

This structure element defines a null data position in a structure. The data is ignored during traversal.

```
                ---------------------
WORDS 1         | length | X'00E5'  | Element header
                ---------------------
```

Workstation Dependent Output (GPWDO)

This structure element defines data the application is sending directly to the workstation.

```
                ---------------------
WORDS 1         | length | X'00F6'  | Element header
                ---------------------
      2         |      length       | Fullword integer
                ---------------------
```

```
3     |   application     | Byte string
      /   defined data    /
      /                   /
      ---------------------
```

Chapter 12. Structure Element Content as Returned by GPQE

If your application was coded to the Version 1 graPHIGS API, it may be dependent on the format of structure elements from that version. The Inquire Element Content (**GPQE**) subroutine returns the contents of the Version 1 structure elements in a format that is compatible with the Version 1 format. The contents and organization of the structure element records built by the graPHIGS API are provided in this chapter.

When your application uses the Inquire Element Content (**GPQE**) subroutine, the size in bytes and the contents of the structure element record are returned by the API in the formats presented here. The subroutine call in parenthesis following each element shows the corresponding subroutine call used to create the element.

The **GPQE** subroutine is provided only for compatibility with Version 1. If you use any of the new subroutine calls provided in Version 2, you must convert your application to use the Inquire Element Content (**GPQED**) subroutine. of structure elements returned by **GPQED**.

The notes preceding each structure element record format tell how the element is defined, and for some, processing considerations of which you should be aware.

Structure elements are organized in this chapter as follows:

- Output Primitives
- Attributes
- Modeling and Viewing
- Miscellaneous Structure Elements

Output Primitives

Annotation 2 (GPAN2)

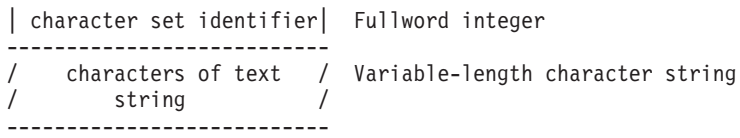
This structure element defines an annotation text string in modeling space. It is drawn at the location specified in a plane parallel to the view plane.

```
--------------------------
|      x position        |  Short floating-point number
--------------------------
|      y position        |  Short floating-point number
--------------------------
|  length of text string |  Fullword integer
--------------------------
| character set identifier|  Fullword integer
--------------------------
/    characters of text   /  Variable-length character string
/        string           /
--------------------------
```

Annotation 3 (GPAN3)

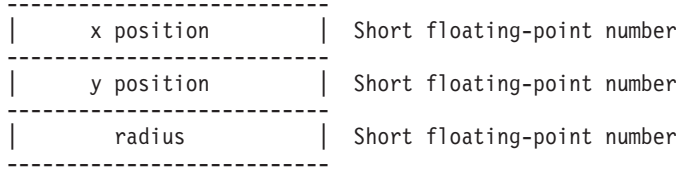
This structure element defines an annotation text string and its position in modeling space. It is drawn at the location specified in a plane parallel to the view plane.

```
--------------------------
|      x position        |  Short floating-point number
--------------------------
|      y position        |  Short floating-point number
--------------------------
|      z position        |  Short floating-point number
--------------------------
|  length of text string |  Fullword integer
--------------------------
```



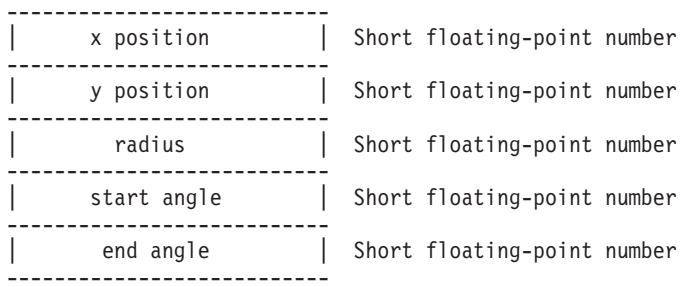
Circle 2 (GPCR2)

This structure element defines a two-dimensional circle primitive.



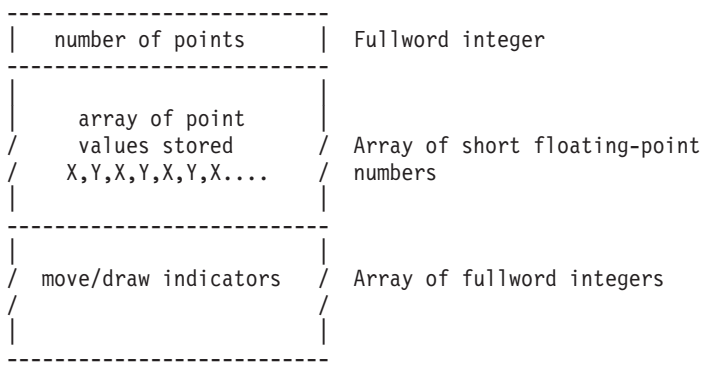
Circular Arc 2 (GPCRA2)

This structure element defines a two-dimensional circular arc primitive.



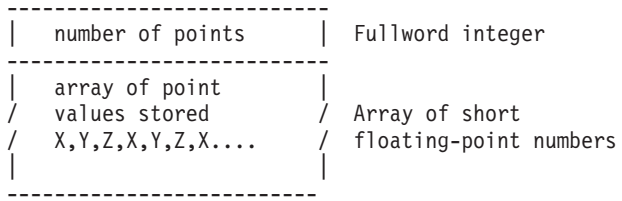
Disjoint Polyline 2 (GPDPL2)

This structure element defines a series of two-dimensional points that may or may not be connected by straight lines. When processing this element during traversal, the *z* coordinate defaults to 0.0. The `WIDTH` parameter is discarded when creating the structure element.



Disjoint Polyline 3 (GPDPL3)

This structure element defines a series of three-dimensional points that may or may not be connected by straight lines. The `WIDTH` parameter is discarded when creating the structure element.



```
|                          |
/  move/draw indicators    /  Array of fullword integers
/                          /
|                          |
---------------------------
```

Ellipse 2 (GPEL2)

This structure element defines a two-dimensional ellipse primitive.

```
---------------------------
|      x position         |  Short floating-point number
---------------------------
|      y position         |  Short floating-point number
---------------------------
|  major axis component 1 |  Short floating-point number
---------------------------
|  major axis component 2 |  Short floating-point number
---------------------------
|  minor axis component 1 |  Short floating-point number
---------------------------
|  minor axis component 2 |  Short floating-point number
---------------------------
```

Ellipse 3 (GPEL3)

This structure element defines a three-dimensional ellipse primitive.

```
---------------------------
|      x position         |  Short floating-point number
---------------------------
|      y position         |  Short floating-point number
---------------------------
|      z position         |  Short floating-point number
---------------------------
|  major axis component 1 |  Short floating-point number
---------------------------
|  major axis component 2 |  Short floating-point number
---------------------------
|  major axis component 3 |  Short floating-point number
---------------------------
|  minor axis component 1 |  Short floating-point number
---------------------------
|  minor axis component 2 |  Short floating-point number
---------------------------
|  minor axis component 3 |  Short floating-point number
---------------------------
```

Elliptical Arc 2 (GPELA2)

This structure element defines a two-dimensional elliptical arc primitive.

```
---------------------------
|      x position         |  Short floating-point number
---------------------------
|      y position         |  Short floating-point number
---------------------------
|  major axis component 1 |  Short floating-point number
---------------------------
|  major axis component 2 |  Short floating-point number
---------------------------
|  minor axis component 1 |  Short floating-point number
---------------------------
|  minor axis component 2 |  Short floating-point number
---------------------------
```

start angle	Short floating-point number
end angle	Short floating-point number

Elliptical Arc 3 (GPELA3)

This structure element defines a three-dimensional elliptical arc primitive.

x position	Short floating-point number
y position	Short floating-point number
z position	Short floating-point number
major axis component 1	Short floating-point number
major axis component 2	Short floating-point number
major axis component 3	Short floating-point number
minor axis component 1	Short floating-point number
minor axis component 2	Short floating-point number
minor axis component 3	Short floating-point number
start angle	Short floating-point number
end angle	Short floating-point number

Pixel 2 (GPPXL2)

This structure element defines a pixel 2 primitive in modeling space.

x position	Short floating-point number
y position	Short floating-point number
packing factor	Fullword integer
number of rows in array	Fullword integer
number of cols in array	Fullword integer
array of color indexes	Array of fullword integers

Pixel 3 (GPPXL3)

This structure element defines a pixel 3 primitive in modeling space.

x position	Short floating-point number
y position	Short floating-point number
z position	Short floating-point number
packing factor	Fullword integer
number of rows in array	Fullword integer
number of cols in array	Fullword integer

```
--------------------------
/        array of        /  Array of fullword integers
/      color indexes     /
--------------------------
```

Polygon 2 (GPPG2)

This structure element defines a polygon in two-dimensional modeling space. All points specified are placed in the *x*, *y* plane. The WIDTH parameter is discarded when creating the structure element.

```
--------------------------
| number of subareas (n) |  Fullword integer
--------------------------
|# of points in subarea 1|  Fullword integer
--------------------------
/                         /
/                         /
--------------------------
|# of points in subarea n|  Fullword integer
--------------------------
|                         |
/ array of point values   /  Array of short
/ stored X,Y,X,Y,X....    /  floating-point numbers
|                         |
--------------------------
```

Polygon 3 (GPPG3)

This structure element defines a polygon in three-dimensional modeling space. All points specified must lie in the same plane but no check is made to verify this. The WIDTH parameter is discarded when creating the structure element.

```
--------------------------
| number of subareas (n) |  Fullword integer
--------------------------
|# of points in subarea 1|  Fullword integer
--------------------------
/                         /
/                         /
--------------------------
|# of points in subarea n|  Fullword integer
--------------------------
|                         |
/ array of point values   /  Array of short
/ stored X,Y,Z,X,Y,Z,X... /  floating-point numbers
|                         |
--------------------------
```

Polyline 2 (GPPL2)

This structure element defines a series of two-dimensional points that are to be connected by straight lines. The WIDTH parameter is discarded when creating the structure element.

```
--------------------------
|   number of points      |  Fullword integer
--------------------------
|                         |
/ array of point values   /  Array of short
/ stored X,Y,X,Y,X....    /  floating-point numbers
|                         |
--------------------------
```

Polyline 3 (GPPL3)

This structure element defines a series of three-dimensional points that are to be connected by straight lines. The WIDTH parameter is discarded when creating the structure element.

```
--------------------------
|   number of points     |  Fullword integer
--------------------------
|                        |
/ array of point values  /  Array of short
/ stored X,Y,Z,X,Y,Z,X... /  floating-point numbers
|                        |
--------------------------
```

Polymarker 2 (GPPM2)

This structure element defines a series of two-dimensional points which are to be identified with markers. The `WIDTH` parameter is discarded when creating the structure element.

```
--------------------------
|   number of points     |  Fullword integer
--------------------------
|                        |
/ array of point values  /  Array of short
/ stored X,Y,X,Y,X,Y,X... /  floating-point numbers
|                        |
--------------------------
```

Polymarker 3 (GPPM3)

This structure element defines a series of three-dimensional points, which are to be identified with markers. The `WIDTH` parameter is discarded when creating the structure element.

```
--------------------------
|   number of points     |  Fullword integer
--------------------------
|                        |
/ array of point values  /  Array of short
/ stored X,Y,Z,X,Y,Z,    /  floating-point numbers
|        X,Y,Z...        |
--------------------------
```

Geometric Text 2 (GPTX2)

This structure element defines a text string in modeling space. It is drawn at the location specified in the XY plane.

```
--------------------------
|      x position        |  Short floating-point number
--------------------------
|      y position        |  Short floating-point number
--------------------------
|  length of text string |  Fullword integer
--------------------------
| character set identifier|  Fullword integer
--------------------------
/    characters of text  /  Variable-length character string
/        string          /
--------------------------
```

Geometric Text 3 (GPTX3)

This structure element defines a text string and its orientation in modeling space. It is drawn at the location specified and in the plane defined by the position and two reference points.

```
--------------------------
|      x position        |  Short floating-point number
--------------------------
|      y position        |  Short floating-point number
--------------------------
|      z position        |  Short floating-point number
--------------------------
|  x reference point 1   |  Short floating-point number
```

```
--------------------------
|  y reference point 1    |  Short floating-point number
--------------------------
|  z reference point 1    |  Short floating-point number
--------------------------
|  x reference point 2    |  Short floating-point number
--------------------------
|  y reference point 2    |  Short floating-point number
--------------------------
|  z reference point 2    |  Short floating-point number
--------------------------
|  length of text string  |  Fullword integer
--------------------------
| character set identifier|  Fullword integer
--------------------------
|                         |  Variable-length character string
/    characters of text   /
/        string           /
|                         |
--------------------------
```

Attributes

Set Polyline Index (GPPLI)

This structure element sets the current polyline bundle index to the specified value. All subsequent polyline primitives use the contents of the specified bundle table entry for all polyline attributes whose attribute source flag is set to BUNDLED.

```
--------------------------
|  polyline bundle index  |  Fullword integer
--------------------------
```

Set Polymarker Index (GPPMI)

This structure element sets the current polymarker bundle index to the specified value. All subsequent polymarker primitives use the contents of the specified bundle table entry for all polymarker attributes whose attribute source flags are set to BUNDLED.

```
--------------------------
| polymarker bundle index |  Fullword integer
--------------------------
```

Set Text Index (GPTXI)

This structure element sets the current text bundle index to the specified value. All subsequent text primitives use the contents of the specified bundle table entry for all text attributes whose attribute source flags are set to BUNDLED.

```
--------------------------
|     text bundle index   |  Fullword integer
--------------------------
```

Set Interior Index (GPII)

This structure element sets the interior bundle index to the specified value. All subsequent polygon primitives use the contents of the specified bundle table entry for all interior attributes whose attribute source flags are set to BUNDLED.

```
--------------------------
|   interior bundle index |  Fullword integer
--------------------------
```

Set Edge Index (GPEI)

This structure element sets the edge bundle index to the specified value. All subsequent polygon primitives use the contents of the specified bundle table entry for all edge attributes whose attribute source flags are set to BUNDLED.

```
---------------------------
|   edge bundle index     |  Fullword integer
---------------------------
```

Set Linetype (GPLT)

This structure element sets the current line type to the specified value. All subsequent polyline primitives use this line type for drawing the primitive if the line type attribute source flag is set to INDIVIDUAL.

```
---------------------------
|        linetype         |  Fullword integer
---------------------------
```

Set Linewidth Scale Factor (GPLWSC)

This structure element sets the current line width scale factor. All subsequent polyline primitives use this value to determine the line width of lines to be drawn if the attribute source flag is set to INDIVIDUAL.

```
---------------------------
|  linewidth scale factor |  Short floating-point number
---------------------------
```

Set Polyline Color Index (GPPLCI)

This structure element sets the current polyline color index to the specified value. All subsequent polyline primitives use this color index for drawing the primitive if the polyline color index attribute source flag is set to INDIVIDUAL.

```
---------------------------
|   polyline color index  |  Fullword integer
---------------------------
```

Set Polyline Endtype (GPPLET)

This structure element sets the polyline end type to the specified value.

```
---------------------------
|        endtype          |  Fullword integer
---------------------------
```

Set Marker Type (GPMT)

This structure element sets the current marker type. All subsequent polymarker primitives use this marker type for identifying each point if the marker type attribute source flag is set to INDIVIDUAL.

```
---------------------------
|      marker type        |  Fullword integer
---------------------------
```

Set Marker Size Scale Factor (GPMSSC)

This structure element sets the current marker size scale factor. All subsequent polymarker primitives use this value to determine the size to draw the markers if the attribute source flag is set to INDIVIDUAL.

```
---------------------------
| marker size scale factor|  Short floating-point number
---------------------------
```

Set Polymarker Color Index (GPPMCI)

This structure element sets the current polymarker color index to the specified value. All subsequent polymarker primitives use this color index for drawing the primitive if the polymarker color index attribute source flag is set to INDIVIDUAL.

```
---------------------------
|  polymarker color index |  Fullword integer
---------------------------
```

Set Text Font (GPTXFO)

This structure element sets the current text font to the one specified. All subsequent text primitives are drawn in this font if the text font attribute source flag is set to INDIVIDUAL.

```
---------------------------
|     text font #         |  Fullword integer
---------------------------
```

Set Text Precision (GPTXPR)

This structure element sets the current text precision to that specified. All subsequent text primitives are drawn at this precision if the text precision attribute source flag is set to INDIVIDUAL.

```
---------------------------
|      text precision     |  Fullword integer
---------------------------
```

Set Character Expansion Factor (GPCHXP)

This structure element sets the current character expansion factor. All subsequent text primitives are drawn with this value for the character expansion factor if its attribute source flag is set to INDIVIDUAL. The value is a fraction of the width/height ratio that the font designer specified. A value of 1.0 reproduces the font designer's aspect ratio.

```
----------------------------
|character expansion factor|  Short floating-point number
----------------------------
```

Set Character Spacing (GPCHSP)

This structure element sets the current character spacing. All subsequent text primitives are drawn with this value for the character spacing if its attribute source flag is set to INDIVIDUAL.

```
---------------------------
|    character spacing    |  Short floating-point number
---------------------------
```

Set Annotation Height Scale Factor (GPAHSC)

This structure element sets the current annotation height scale factor. All subsequent annotation text primitives are drawn with this value for the height scale factor.

```
---------------------------
|     annotation h.s.f.   |  Short floating-point number
---------------------------
```

Set Text Color Index (GPTXCI)

This structure element sets the text color index to the specified value. All subsequent text primitives use this color index for drawing the primitive if the text color index attribute source flag is set to INDIVIDUAL.

```
---------------------------
|    text color index     |  Fullword integer
---------------------------
```

Set Character Height (GPCHH)

This structure element sets the current character height. All subsequent non-annotation text primitives will be drawn with this value for the character height.

```
-------------------------
|    character height   |  Short floating-point number
-------------------------
```

Set Character Up Vector (GPCHUP)

This structure element sets the current character up vector. All subsequent text primitives are drawn with this value for the character up vector.

```
-------------------------
|    X  direction       |  Short floating-point number
-------------------------
|    Y  direction       |  Short floating-point number
-------------------------
```

Set Geometric Text Path (GPTXPT)

This structure element sets the current geometric text path. All subsequent text primitives are drawn with this value for the text path.

```
-------------------------
|       text path       |  Fullword integer
-------------------------
```

Set Geometric Text Alignment (GPTXAL)

This structure element sets the current geometric text alignment. All subsequent geometric text primitives are drawn with this value for text alignment.

```
-------------------------
|       horizontal      |  Fullword integer
-------------------------
|        vertical       |  Fullword integer
-------------------------
```

Set Interior Style (GPIS)

This structure element sets the current interior style. All subsequent primitives use this attribute when drawing the interior if the corresponding attribute source flag is set to `INDIVIDUAL`.

```
-------------------------
|    interior style     |  Fullword integer
-------------------------
```

Set Interior Style Index (GPISI)

This structure element sets the current interior style index. All subsequent primitives use this attribute when drawing the interior if the corresponding attribute source flag is set to `INDIVIDUAL` and the current interior style is `HATCH` or `PATTERN`.

```
-------------------------
| interior style index  |  Fullword integer
-------------------------
```

Set Interior Color Index (GPICI)

The current interior color index is set by this structure element. All subsequent primitives use this attribute when drawing the interior if the corresponding attribute source flag is set to `INDIVIDUAL` and the current interior style is `SOLID` or `HATCH`.

```
-------------------------
|  interior color index |  Fullword integer
-------------------------
```

Set Edge Flag (GPEF)

The visibility of edges is set by this structure element. All subsequent primitives use this attribute to determine whether the edges should be drawn if the attribute source flag is set to INDIVIDUAL.

```
--------------------------
|       edge flag        |  Fullword integer
--------------------------
```

Set Edge Linetype (GPELT)

This structure element sets the current edge line type to the specified value. All subsequent primitives use this edge line type for drawing the primitive if the edge line type attribute source flag is set to INDIVIDUAL and the edge flag is set to ON.

```
--------------------------
|     edge linetype      |  Fullword integer
--------------------------
```

Set Edge Color Index (GPECI)

This structure element sets the current edge color index to the specified value. All subsequent primitives use this color index for drawing the edges if the edge color index attribute source flag is set to INDIVIDUAL and the edge flag is set to ON.

```
--------------------------
|    edge color index    |  Fullword integer
--------------------------
```

Set Edge Scale Factor (GPESC)

This structure element sets the current edge scale factor to the specified value. All subsequent primitives use this value to determine the width of the edges to be drawn if the attribute source flag is set to INDIVIDUAL and the edge flag is set to ON.

```
--------------------------
|    edge scale factor   |  Short floating-point number
--------------------------
```

Set Attribute Source Flag (GPASF)

This structure element defines whether the attribute used for rendering each bundled attribute should be the bundled or current individual attribute setting.

```
--------------------------
|          count         |  Fullword integer
--------------------------
|                        |
/        array of        /  Array of fullword integers
/  attribute identifiers /
|                        |
--------------------------
|                        |
/  array of corresponding /  Array of fullword integers
/  attribute ASF values  /
|                        |
--------------------------
```

Modeling and Viewing

Set Modeling Transformation 3 (GPMLX3)

This structure element specifies a modification for local modeling transformation.

```
--------------------------
|     composition type   |  Fullword integer
--------------------------
```

```
|row 1 col 1 matrix element|  Short floating-point number
----------------------------
|row 1 col 2 matrix element|  Short floating-point number
----------------------------
|row 1 col 3 matrix element|  Short floating-point number
----------------------------
|row 1 col 4 matrix element|  Short floating-point number
----------------------------
|row 2 col 1 matrix element|  Short floating-point number
----------------------------
|row 2 col 2 matrix element|  Short floating-point number
----------------------------
|row 2 col 3 matrix element|  Short floating-point number
----------------------------
|row 2 col 4 matrix element|  Short floating-point number
----------------------------
|row 3 col 1 matrix element|  Short floating-point number
----------------------------
|row 3 col 2 matrix element|  Short floating-point number
----------------------------
|row 3 col 3 matrix element|  Short floating-point number
----------------------------
|row 3 col 4 matrix element|  Short floating-point number
----------------------------
|row 4 col 1 matrix element|  Short floating-point number
----------------------------
|row 4 col 2 matrix element|  Short floating-point number
----------------------------
|row 4 col 3 matrix element|  Short floating-point number
----------------------------
|row 4 col 4 matrix element|  Short floating-point number
----------------------------
```

Set Modeling Transformation 2 (GPMLX2)

This structure element specifies a modification for local modeling transformation. The matrix returned by **GPQE** is the expanded, 4[default]4 matrix.

```
----------------------------
|     composition type     |  Fullword integer
----------------------------
|row 1 col 1 matrix element|  Short floating-point number
----------------------------
|row 1 col 2 matrix element|  Short floating-point number
----------------------------
|row 1 col 3 matrix element|  Short floating-point number
----------------------------
|row 1 col 4 matrix element|  Short floating-point number
----------------------------
|row 2 col 1 matrix element|  Short floating-point number
----------------------------
|row 2 col 2 matrix element|  Short floating-point number
----------------------------
|row 2 col 3 matrix element|  Short floating-point number
----------------------------
|row 2 col 4 matrix element|  Short floating-point number
----------------------------
|row 3 col 1 matrix element|  Short floating-point number
----------------------------
|row 3 col 2 matrix element|  Short floating-point number
----------------------------
|row 3 col 3 matrix element|  Short floating-point number
----------------------------
|row 3 col 4 matrix element|  Short floating-point number
----------------------------
|row 4 col 1 matrix element|  Short floating-point number
----------------------------
```

```
|row 4 col 2 matrix element|  Short floating-point number
----------------------------
|row 4 col 3 matrix element|  Short floating-point number
----------------------------
|row 4 col 4 matrix element|  Short floating-point number
----------------------------
```

Set Global Transformation 3 (GPGLX3)

This structure element specifies a global modeling transformation.

```
----------------------------
|row 1 col 1 matrix element|  Short floating-point number
----------------------------
|row 1 col 2 matrix element|  Short floating-point number
----------------------------
|row 1 col 3 matrix element|  Short floating-point number
----------------------------
|row 1 col 4 matrix element|  Short floating-point number
----------------------------
|row 2 col 1 matrix element|  Short floating-point number
----------------------------
|row 2 col 2 matrix element|  Short floating-point number
----------------------------
|row 2 col 3 matrix element|  Short floating-point number
----------------------------
|row 2 col 4 matrix element|  Short floating-point number
----------------------------
|row 3 col 1 matrix element|  Short floating-point number
----------------------------
|row 3 col 2 matrix element|  Short floating-point number
----------------------------
|row 3 col 3 matrix element|  Short floating-point number
----------------------------
|row 3 col 4 matrix element|  Short floating-point number
----------------------------
|row 4 col 1 matrix element|  Short floating-point number
----------------------------
|row 4 col 2 matrix element|  Short floating-point number
----------------------------
|row 4 col 3 matrix element|  Short floating-point number
----------------------------
|row 4 col 4 matrix element|  Short floating-point number
----------------------------
```

Set Global Transformation 2 (GPGLX2)

This structure element specifies a global modeling transformation. The matrix returned by **GPQE** is the expanded, 4[default]4 matrix.

```
----------------------------
|row 1 col 1 matrix element|  Short floating-point number
----------------------------
|row 1 col 2 matrix element|  Short floating-point number
----------------------------
|row 1 col 3 matrix element|  Short floating-point number
----------------------------
|row 1 col 4 matrix element|  Short floating-point number
----------------------------
|row 2 col 1 matrix element|  Short floating-point number
----------------------------
|row 2 col 2 matrix element|  Short floating-point number
----------------------------
|row 2 col 3 matrix element|  Short floating-point number
----------------------------
|row 2 col 4 matrix element|  Short floating-point number
----------------------------
|row 3 col 1 matrix element|  Short floating-point number
```

```
---------------------------
|row 3 col 2 matrix element|  Short floating-point number
---------------------------
|row 3 col 3 matrix element|  Short floating-point number
---------------------------
|row 3 col 4 matrix element|  Short floating-point number
---------------------------
|row 4 col 1 matrix element|  Short floating-point number
---------------------------
|row 4 col 2 matrix element|  Short floating-point number
---------------------------
|row 4 col 3 matrix element|  Short floating-point number
---------------------------
|row 4 col 4 matrix element|  Short floating-point number
---------------------------
```

Miscellaneous Structure Elements

Add Class Name to Set (GPADCN)

This structure element adds class names to the current class set.

```
---------------------------
|   # of class names (n)  |  Fullword integer
---------------------------
|      class name 1       |  Fullword integer
---------------------------
/                         /
/                         /
---------------------------
|      class name n       |  Fullword integer
---------------------------
```

Execute Structure (GPEXST)

This structure element defines a call or invocation of another structure.

```
---------------------------
|     structure id        |  Fullword integer
---------------------------
```

Set Highlighting Color Index (GPHLCI)

This structure element sets the index into the workstation-dependent color table, which is used for highlighted primitives.

```
---------------------------
| highlighting color index|  Fullword integer
---------------------------
```

Insert Application Data (GPINAD)

This structure element contains application specified data.

```
---------------------------
|   application defined   |
/                         /
|          data           |
---------------------------
```

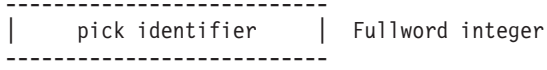
Insert Label (GPINLB)

This structure element defines a label that the application uses to reference and modify structure elements.

```
---------------------------
|          label          |  Fullword integer
---------------------------
```

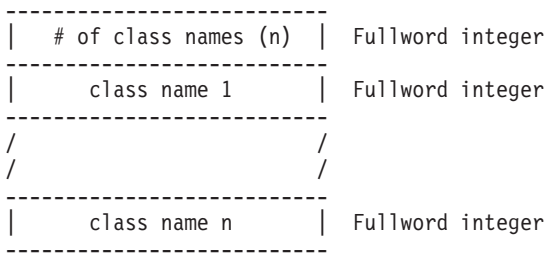
Set Pick Identifier (GPPKID)

This structure element sets the current pick identifier to the specified value.



Remove Class Name from Set (GPRCN)

This structure element removes class names from the current class set.



Appendix A. State Lists

You may wish to know the specific data types and representations supported by the graPHIGS API.

This appendix shows you the description of the data fields and their data types for State Lists and description tables which are used by the API and may be queried by the application program. They are:

- Operating States List (OSL)
- graPHIGS API Descriptor Table (PDT)
- graPHIGS API State List (PSL)
- Structure Store State List (SSL)
- Workstation State List (WSL)
- graPHIGS API Error State List (ESL)
- Utility Function State List (USL)

Operating States List (OSL)

The graPHIGS API defines four state variables. These variables determine the current operating status of each component in the system. Before the API is invoked, all states are CLOSED.

The right-hand column lists the inquiry function that you can use when you want your application to know the operating states of the system.

Data Type Field

In the tables and lists presented in this appendix, the following correspondence applies to the abbreviated data types:

Table 124. Operating States List (OSL) Data Type Field Definition

Data Type	Definition
I Integer	A whole number
R Real	A floating-point number
S String	A character string
E Enumeration	A data type comprised of a set of values. The set is defined by enumerating the identifiers denoting the values.
n **Quantity**	This specifies an undesignated quantity of data. **Note:** The notation of *n* (number) x *t* (data type) indicates a collection of data of that type. This can be indicated in one of two ways: • By using notation such as 3xR (three real numbers), which could specify something like the *x*, *y*, and *z* coordinates of a three-dimensional point or RGB values • By using a variable number such as *n*xI, which specifies a collection of *n* integers.

Table 125. Operating States List (OSL) Data Type Field Description

Description of Field	Data Type	Inquiry
Archive state value (AROP, ARCL)	E	**GPQASV***[state]*

Table 125. Operating States List (OSL) Data Type Field Description (continued)

Description of Field	Data Type	Inquiry
System state value (CLOSED, OPEN)	E	**GPQSYV***[state]*
Workstation state value (CLOSED, OPEN, SELECTED)	E	**GPQWSV***[state]*
Structure state value (STRUCTURE STORE SELECTED BUT NO STRUCTURE IS OPEN [STCL], STRUCTURE STORE SELECTED AND A STRUCTURE IS OPEN [STOP], NO STRUCTURE STORE ATTACHED/CREATED [SSCL], STRUCTURE STORE ATTACHED/CREATED BUT NOT SELECTED [SSOP], [NROP])	E	**GPQSTV***[state]*

The graPHIGS API Descriptor Table (PDT)

This list indicates the values which describe the capabilities of the graPHIGS API. The right-hand column lists the inquiry function that you can use when you want your application to know the capabilities of the graPHIGS API.

Data Type Field

In the tables and lists presented in this appendix, the following correspondence applies to the abbreviated data types:

Table 126. Descriptor Table (PDT) Data Type Field Definition

Data Type	Definition
I Integer	A whole number
R Real	A floating-point number
S String	A character string
E Enumeration	A data type comprised of a set of values. The set is defined by enumerating the identifiers denoting the values.
n **Quantity**	This specifies an undesignated quantity of data. **Note:** The notation of *n* (number) x *t* (data type) indicates a collection of data of that type. This can be indicated in one of two ways: • By using notation such as 3xR (three real numbers), which could specify something like the *x*, *y*, and *z* coordinates of a three-dimensional point or RGB values • By using a variable number such as *n*xI, which specifies a collection of *n* integers.

Table 127. Descriptor Table (PDT) Data Type Field Description

Description of Field	Data Type	Inquiry
Number of available connection methods	I	**GPQCMM***[totnum]*
List of available connection methods	*n*xI	**GPQCMM***[conn]*
Number of available application image formats	I	**GPQAI***[totnum]*
List of available application image formats	*n*xI	**GPQAI***[format]*

The graPHIGS API State List (PSL)

This list indicates the values maintained by the graPHIGS API which describe its current state. As your application runs, the values can change. The right-hand column lists the inquiry function that you can use when you want your application to acquire the values.

Data Type Field

The following correspondence applies to the abbreviated data types:

Table 128. State List (PSL) Data Type Definitions

Data Type	Definition
I Integer	A whole number
R Real	A floating-point number
S String	A character string
E Enumeration	A data type comprised of a set of values. The set is defined by enumerating the identifiers denoting the values.
n **Quantity**	This specifies an undesignated quantity of data. **Note:** The notation of *n* (number) x *t* (data type) indicates a collection of data of that type. This can be indicated in one of two ways: • By using notation such as 3xR (three real numbers), which could specify something like the *x*, *y*, and *z* coordinates of a three-dimensional point or RGB values • By using a variable number such as *n*xI, which specifies a collection of *n* integers.

Table 129. State List (PSL) Data Type Description

Description of Field	Data Type	Inquiry
Shell product level	I	**GPQSPL***[level]*
Shell identifier on a nucleus	I	**GPQSH***[shid]*
Application environment descriptor	4xS	**GPQSH***[env]*
Shell deferral mode (FLUSH, DEFERRED, DEFERRED_PLUS_MSGS)	E	**GPQSHD***[deferral]*
Convexity checking mode	E	
Update notification mode (NO, YES)	E	**GPQSHD***[update]*
Current selected structure store	I	**GPQSSS***[ssid]*
Nucleus resource identifier for a resource	I	**GPQNCR***[rid]*
Number of resources attached to the shell	I	**GPQATR***[totnum]*
List of resources attached to the shell	*n*xI	**GPQATR***[id]*
Number of nuclei connected to the shell	I	**GPQCNC***[totnum]*
List of nucleus identifiers connected to the shell	*n*xI	**GPQCNC***[ncid]*
Current event report	*n*	**GPQCEV***[major, class, minor]*
Current edit mode (INSERT_MODE, REPLACE_MODE)	E	**GPQEDM***[mode]*
More simultaneous events input flag (NOMORE, MORE)	E	**GPQSEV***[simevnt]*

Table 129. State List (PSL) Data Type Description (continued)

Description of Field	Data Type	Inquiry
Current character set identifier	I	**GPQCS***[csid]*
Current direct color model (RGB, HSV, CMY, CIELUV)	E	**GPQDCM***[model]*
Number of open workstations on a nucleus	I	**GPQOPW***[totnum]*
List of open workstations on a nucleus	*n*xI	**GPQOPW***[lwsid]*
Conflict Resolution State (ABANDON, MAINTAIN, UPDATE)	E	**GPQCNR***[state]*

Structure Store State List (SSL)

This list indicates the values maintained by the graPHIGS API which describe the current operating state of a structure store resource. Content of the structure store state list may change during application processing.

The right-hand column lists the inquiry function that you can use when you want your application to acquire the current values.

Data Type Field

The following correspondence applies to the abbreviated data types:

Table 130. Structure Store State List (SSL) Data Type Field Definition

Data Type	Definition
I Integer	A whole number
R Real	A floating-point number
S String	A character string
E Enumeration	A data type comprised of a set of values. The set is defined by enumerating the identifiers denoting the values.
n **Quantity**	This specifies an undesignated quantity of data. **Note:** The notation of *n* (number) x *t* (data type) indicates a collection of data of that type. This can be indicated in one of two ways: • By using notation such as 3xR (three real numbers), which could specify something like the *x*, *y*, and *z* coordinates of a three-dimensional point or RGB values • By using a variable number such as *n*xI, which specifies a collection of *n* integers.

Table 131. Structure Store State List (SSL) Data Type Field Description

Description of Field	Data Type	Inquiry
Current open structure identifier	I	**GPQOPS***[strid]*
Current element pointer	I	**GPQEP***[value]*
Structure existence (NON_EXISTENT, EXISTENT)	E	**GPQSTE***[flag]*

Table 131. Structure Store State List (SSL) Data Type Field Description (continued)

Description of Field	Data Type	Inquiry
Number of existing structures	I	**GPQSTI***[totnum]*
List of existing structures	*n*xI	**GPQSTI***[lstrid]*
Number of existing execute structures	I	**GPQEXS***[totnum]*
List of existing execute structures	*n*xI	**GPQEXS***[lstrid]*
List of element headers	*n*x S	**GPQEHD***[header]*
List of element data	*n*xS	**GPQHD***[data]*
Number of workstations to which the structure is associated	I	**GPQWSA***[totnum]*
List of workstations to which the structure is associated	*n*xI	**GPQWSA***[lwsid]*

Workstation State List (WSL)

This list describes the current operating state of a given workstation. One WSL exists for each open workstation. Content of the Workstation State List may change during application processing.

The right-hand column lists the inquiry subroutine call that you can use when you want your application to acquire the current values.

Data Type Field

The following correspondence applies to the abbreviated data types:

Table 132. Workstation State List (WSL) Data Type Field Definition

Data Type	Definition
I Integer	A whole number
R Real	A floating-point number
S String	A character string
E Enumeration	A data type comprised of a set of values. The set is defined by enumerating the identifiers denoting the values.
n **Quantity**	This specifies an undesignated quantity of data. **Note:** The notation of *n* (number) x *t* (data type) indicates a collection of data of that type. This can be indicated in one of two ways: • By using notation such as 3xR (three real numbers), which could specify something like the *x*, *y*, and *z* coordinates of a three-dimensional point or RGB values • By using a variable number such as *n*xI, which specifies a collection of *n* integers.

Table 133. Workstation State List (WSL) Data Type Field Description

Description of Field	Data Type	Inquiry
Connection identifier	S	**GPQRCT***[olen,connid]*
Workstation type (actual)	S	**GPQRCT***[wstype]*
Requested workstation windows	6xR	**GPQWSX***[rwindow]*
Current workstation windows	6xR	**GPQWSX***[cwindow]*

Table 133. Workstation State List (WSL) Data Type Field Description (continued)

Description of Field	Data Type	Inquiry
Requested workstation viewports	6xR	**GPQWSX***[rviewpt]*
Current workstation viewports	6xR	**GPQWSX***[cviewpt]*
Table of Requested Viewing Operation Information:		
Total number of requested viewing table entries in output priority order	I	**GPQRVO***[nview]*
Total number of requested viewing table entries in input priority order	I	**GPQRVE***[nview]*
For Each View Entry:		
Viewing transformation matrix	4x4xR	**GPQRVR***[data—group 18, 19]*
View mapping matrix	4x4xR	**GPQRVR***[data—group 22, 23]*
Window viewing coordinates	4xR	**GPQRVR***[data—group 16, 17]*
Viewport (normalized projection coordinates)	6xR	**GPQRVR***[data—group 14, 15]*
Projection (reference point viewing coordinates)	3xR	**GPQRVR***[data—group 17]*
View plane distance	R	**GPQRVR***[data—group 17]*
Near distance	R	**GPQRVR***[data—group 17]*
Far distance	R	**GPQRVR***[data—group 17]*
Projection type (`PARALLEL, PERSPECTIVE`)	E	**GPQRVR***[data—group 17]*
Window clipping indicator (`CLIP, NOCLIP`)	E	**GPQRVR***[data—group 1]*
Near clipping indicator (`CLIP, NOCLIP`)	E	**GPQRVR***[data—group 2]*
Far clipping indicator (`CLIP, NOCLIP`)	E	**GPQRVR***[data—group 3]*
Shielding indicator (`OFF, ON`)	E	**GPQRVR***[data—group 4]*
Shielding color type (`INDEXED, DIRECT`)	I	**GPQRVR***[data—group 5]*
Shielding color	I or 3xR	**GPQRVR***[data—group 5]*
View border indicator (`OFF, ON`)	E	**GPQRVR***[data—group 6]*
View border color type (`INDEXED, DIRECT`)	I	**GPQRVR***[data—group 7]*
View border color	I or 3xR	**GPQRVR***[data—group 7]*

Table 133. Workstation State List (WSL) Data Type Field Description (continued)

Description of Field	Data Type	Inquiry
View active/inactive indicator for input (OFF, ON)	E	**GPQRVR***[data—group 20]*
View active/inactive indicator for output (OFF, ON)	E	**GPQRVR***[data—group 21]*
Temporary view indicator (OFF, ON)	E	**GPQRVR***[data—group 9]*
HLHSR (hidden line hidden surface removal mode) (OFF, ON_THE_FLY)	E	**GPQRVR***[data—group 10]*
Transparency mode (OFF, PARTIAL_TRANSPARENT, BLEND, BLEND_ALL)	E	**GPQRVR***[data—group 11]*
Antialiasing mode (OFF, SUBPIXEL_ON_THE_FLY, NON_SUBPIXEL_ON_THE_FLY)	E	**GPQRVR***[data—group 24]*
Color processing index	I	**GPQRVR***[data—group 12]*
Frame buffer write protect mask	I	**GPQRVR***[data—group 13]*
Table of Current Viewing Operation Information:		
Total number of current viewing table entries in output priority order	I	**GPQCVO***[nview]*
Total number of current viewing table entries in input priority order	I	**GPQCVE***[nview]*
For Each View Entry:		
Viewing transformation matrix	4x4xR	**GPQCVR***[data—group 18, 19]*
View mapping matrix	4x4xR	**GPQCVR***[data—group 22, 23]*
Window viewing coordinates	4xR	**GPQCVR***[data—group 16, 17]*
Viewport normalized projection coordinates	6xR	**GPQCVR***[data—group 14, 15]*
Projection reference point viewing coordinates	3xR	**GPQCVR***[data—group 17]*
View plane distance	R	**GPQCVR***[data—group 17]*
Near distance	R	**GPQCVR***[data—group 17]*
Far distance	R	**GPQCVR***[data—group 17]*
Projection type (PARALLEL, PERSPECTIVE)	E	**GPQCVR***[data—group 17]*
Window clipping indicator (CLIP, NOCLIP)	E	**GPQCVR***[data—group 1]*
Near clipping indicator (CLIP, NOCLIP)	E	**GPQCVR***[data—group 2]*

Table 133. Workstation State List (WSL) Data Type Field Description (continued)

Description of Field	Data Type	Inquiry
Far clipping indicator (CLIP, NOCLIP)	E	**GPQCVR***[data—group 3]*
Shielding indicator (OFF, ON)	E	**GPQCVR***[data—group 4]*
Shielding color type (INDEXED, DIRECT)	I	**GPQCVR***[data—group 5]*
Shielding color	I or 3xR	**GPQCVR***[data—group 5]*
View border indicator (OFF, ON)	E	**GPQCVR***[data—group 6]*
View border color	I or 3xR	**GPQCVR***[data—group 7]*
View border color type (INDEXED, DIRECT)	I	**GPQCVR***[data—group 7]*
View active/inactive indicator for input (OFF, ON)	E	**GPQCVR***[data—group 20]*
View active/inactive indicator for output (OFF, ON)	E	**GPQCVR***[data—group 21]*
Temporary view indicator (OFF, ON)	E	**GPQCVR***[data—group 9]*
HLHSR (hidden line hidden surface removal mode) (OFF, ON_THE_FLY)	E	**GPQCVR***[data—group 10]*
Transparency mode (OFF, PARTIAL_TRANSPARENT, BLEND, BLEND_ALL)	E	**GPQCVR***[data—group 11]*
Antialiasing mode (OFF, SUBPIXEL_ON_THE_FLY, NON_SUBPIXEL_ON_THE_FLY)	E	**GPQCVR***[data—group 24]*
Color processing index	I	**GPQCVR***[data—group 12]*
Frame buffer write protect mask	I	**GPQCVR***[data—group 13]*
Current deferral mode (AS SOON AS POSSIBLE [ASAP], BEFORE NEXT INTERACTION GLOBALLY [BNIG], BEFORE NEXT INTERACTION LOCALLY [BNIL], AT SOME TIME [ASTI], WHEN APPLICATION REQUESTS IT [WAIT])	E	**GPQDV***[defer]*
Current modification mode (NO_IMMEDIATE_VISUAL_EFFECT, UPDATE_WITHOUT_REGEN, QUICK_UPDATE)	E	**GPQDV***[modify]*
Display surface empty flag (NOT_EMPTY, IS_EMPTY)	E	**GPQDV***[dissurf]*
Display status (CORRECT, DEFERRED, SIMULATED)	E	**GPQDV***[dstat]*
Update flag (NOT_PENDING, PENDING)	E	

Table 133. Workstation State List (WSL) Data Type Field Description (continued)

Description of Field	Data Type	Inquiry
Polyline Bundle Table:		
Number of polyline bundle table entries	I	
For Each Polyline Bundle Table Entry:		
Line type (SOLID_LINE, DASHED, DOTTED, DASH_DOT, LONG_DASH, DOUBLE_DOT, DASH_DOUBLE_DOT) (1..*n*)	E	**GPQXLR***[data—group 1]*
Line width scale factor	R	**GPQXLR***[data—group 2]*
Polyline color Type Color	I or 3xR	**GPQXLR***[data—group 3]*
Line Pattern Table:		
Number of line pattern table entries	I	
For Each Line Pattern Table Entry:		
Number of sections in line pattern	I	**GPQLTR***[number]*
List of each section in the line pattern	*n*xI	**GPQLTR***[pattern]*
Polymarker Bundle Table:		
Number of polymarker bundle table entries	I	
For Each Polymarker Bundle Table Entry:		
Marker type (DOT, PLUS_SIGN, ASTERISK, CIRCLE, DIAGONAL_CROSS) (1..*n*)	E	**GPQXMR***[data—group 1]*
Marker size scale factor	R	**GPQXMR***[data—group 2]*
Polymarker color Type Color	I or 3xR	**GPQXMR***[data—group 3]*
Marker Pattern Table:		
Number of marker pattern table entries	I	
For Each Marker Pattern Table Entry:		
Marker pattern format (VECTOR)	I	**GPQMTR***[format]*
Length of marker pattern definition	I	**GPQMTR***[length]*
Marker pattern definition	I	**GPQMTR***[data]*
Text Bundle Table:		
Number of text bundle table entries	I	
For Each Text Bundle Table Entry:		
Text font (1..*n*)	I	**GPQXTR***[data—group 1]*
Text precision (STRING_PREC, CHAR_PREC, STROKE_PREC)	E	**GPQXTR***[data—group 2]*
Character expansion factor	R	**GPQXTR***[data—group 3]*

Table 133. Workstation State List (WSL) Data Type Field Description (continued)

Description of Field	Data Type	Inquiry
Character spacing	R	**GPQXTR***[data—group 4]*
Text color Type Color	I or 3xR	**GPQXTR***[data—group 5]*
Interior Bundle Table:		
Number of interior bundle table entries	I	
For Each Interior Bundle Table Entry:		
Interior style (HOLLOW, SOLID, PATTERN, HATCH, EMPTY) (1..*n*)	E	**GPQXIR***[data—group 1]*
Interior style index (1..*n*)	I	**GPQXIR***[data—group 2]*
Interior color Type Color	I or 3xR	**GPQXIR***[data—group 3]*
Edge Bundle Table:		
Number of edge bundle table entries	I	
For Each Edge Bundle Table Entry:		
Edge flag (OFF, ON, GEOMETRY_ONLY)	E	**GPQXER***[data—group 1]*
Edge line type (SOLID, DASHED_DOTTED_DASH_DOT, LONG_DASH, DOUBLE_DOT, DASH_DOUBLE_DOT) (1..*n*)	E	**GPQXER***[data—group 2]*
Edge line width scale factor	R	**GPQXER***[data—group 3]*
Edge color Type Color	I or 3xR	**GPQXER***[data—group 4]*
Pattern Table:		
Number of pattern table entries	I	
For Each Pattern Table Entry:		
Pattern array dimension (1..*n*)	2xI	**GPQPAR***[drow,dcol]*
Pattern array (0..n)	*n*x*n*xI	**GPQPAR***[array]*
Hatch Table:		
Number of hatch table entries	I	
For Each Hatch Table Entry:		
Hatch format (bit array)	I	**GPQHR***[format]*
Hatch array dimension (1...*n*)	2xI	**GPQHR***[length]*
Hatch pattern array	*n*x*n*xI	**GPQHR***[data]*
Character Set Table:		
Number of character set ID/ fonts in active pool	I	**GPQFO***[nfont]*
For Each Active Character Set Entry:		
Character set identifier (1..255)	I	**GPQFO***[lcsid]*

Table 133. Workstation State List (WSL) Data Type Field Description (continued)

Description of Field	Data Type	Inquiry
Font identifier (1..*n*) (**Note:** The first character set entry is always the primary character set)	I	**GPQFO***[lfont]*
Color Tables:		
Number of color tables	I	**GPQCID***[totnum]*
For Each Color Table:		
Color table identifier	I	**GPQCID***[ctid]*
Color model (RGB, HSV, CMY, CIELUV)	I	**GPQCCH***[model]*
Size of the color table	I	**GPQCCH***[length]*
For Each Color Table Entry:		
Color components	3xR	**GPQXCR***[color]*
Light Source Table:		
Number of light source table entries	I	
For Each Light Souce Table Entry:		
Light source type (AMBIENT, DIRECTIONAL, POSITIONAL, SPOT)	I	**GPQLSR***[lstype]*
Light source color Type Color	I or 3xR	**GPQLSR***[color]*
Light source data	*n*xR	**GPQLSR***[data]*
Depth Cue Table:		
Number of depth cue table entries	I	
For Each Depth Cue Table Entry:		
Depth cue mode	I	**GPQDCR***[data—group 1]*
Depth cue reference planes	2xR	**GPQDCR***[data—group 2]*
Depth cue scale factors	2xR	**GPQDCR***[data—group 3]*
Depth cue color Type Color	I or 3xR	**GPQDCR***[data—group 4]*
Color Processing Mode Table:		
Number of color processing mode table entries	I	
For Each Color Processing Table Entry:		
Rendering color model (RGB, RGB_B_ONLY)	E	**GPQCPR***[model]*
Color quantization method (WORKSTATION_DEPENDENT, BITWISE,)	E	**GPQCPR***[quant]*
Color quantization data		**GPQCPR***[data]*
Cull Size Table:		

Table 133. Workstation State List (WSL) Data Type Field Description (continued)

Description of Field	Data Type	Inquiry
Number of cull size table entries	I	
For Each Cull Size Table Entry:		
Cull size	R	**GPQCSR***[size]*
Highlighting Information:		
Number of classes in inclusion filter	I	**GPQHLF***[inclen]*
List of classes in inclusion filter	*n*xI	**GPQHLF***[incl]*
Number of classes in exclusion filter	I	**GPQHLF***[exclen]*
List of classes in exclusion filter	*n*xI	**GPQHLF***[excl]*
Invisibility Information:		
Number of classes in inclusion filter	I	**GPQIVF***[inclen]*
List of classes in inclusion filter	*n*xI	**GPQIVF***[incl]*
Number of classes in exclusion filter	I	**GPQIVF***[exclen]*
List of classes in exclusion filter	*n*xI	**GPQIVF***[excl]*
Image Definition Table:		
Number of defined images on the workstation	I	**GPQIW***[totnum]*
List of defined images on the workstation	*n*xI	**GPQIW***[limage]*
For Each Defined Image Table Entry:		
Connection type (`FRAME_BUFFER_COMPATIBLE, COMPONENT, INDEXED`)	E	**GPQICH***[conn]*
Color table identifier	I	**GPQICH***[ctid]*
Number of image boards that form image	I	**GPQICH***[totnum]*
List of image boards that form image	*n*xI	**GPQICH***[libid]*
For Each Image Board:		
Bit depth of the image board	I	**GPQIBC***[depth]*
Size of the image board	2xI	**GPQIBC***[h,v]*
Image Display:		
Number of image mappings of the image	I	**GPQIMI***[totnum]*
List of image mappings of the image	*n*xI	**GPQIMI***[limid]*
Number of image mappings for the view	I	**GPQIMV***[totnum]*
List of image mappings for the view	*n*xI	**GPQIMV***[limid]*
Number of image mappings on the workstation	I	**GPQIMW***[totnum]*
List of image mappings on the workstation	*n*xI	**GPQIMW***[limid]*
For Each Image Mapping:		
Image mapping method (`PIXEL_BY_PIXEL`)	E	**GPQIMC***[method]*
View index	I	**GPQIMC***[vindex]*
Priority	R	**GPQIMC***[priority]*
Image definition index	I	**GPQIMC***[iindex]*
Image rectangle origin	I	**GPQIMC***[origin]*
Image rectangle size	2xI	**GPQIMC***[size]*

Table 133. Workstation State List (WSL) Data Type Field Description (continued)

Description of Field	Data Type	Inquiry
Lower left corner of the image mapping	3xR	**GPQIMC***[p]*
Lower right corner of the image mapping	3xR	**GPQIMC***[q]*
Upper left corner of the image mapping	3xR	**GPQIMC***[r]*
Break Action:		
Trigger type	I	**GPQBKS***[trigger]*
Trigger qualifier	I	**GPQBKS***[trigger]*
Number of locator devices	I	N/A
Number of stroke devices	I	N/A
Number of valuator devices	I	N/A
Number of choice devices	I	N/A
Number of pick devices	I	N/A
Number of string devices	I	N/A
Table of Input Devices: Entries in this group do not exist for workstations of category output.		
For Every Logical Input Device of Class Locator:		
Operating mode (`REQUEST, PRELE, EVENT`)	E	**GPQLC***[mode]*
Echo switch (`NOECHO, ECHO`)	E	**GPQLC***[echosw]*
Prompt and echo type	I	**GPQLC***[echo]*
Current echo area (DC)	6xR	**GPQLC***[area]*
Length of locator data record	I	**GPQLC***[datalen]*
Current locator data record		**GPQLC***[data]*
Current input character set identifier	E	**GPQICS***[csid]*
Initial view index	I	**GPQLC***[view]*
Initial locator position (WC)	3xR	**GPQLC***[pos]*
Current trigger list(s) (one per trigger list identifier)	*n* (3xI)	**GPQITS***[ltrigs]*
For Every Logical Input Device of Class Stroke:		
Operating mode (`REQUEST, PRELE, EVENT`)	E	**GPQSK***[mode]*
Echo switch (`NOECHO, ECHO`)	E	**GPQSK***[echosw]*
Prompt and echo type	I	**GPQSK***[echo]*
Current echo area (DC)	6xR	**GPQSK***[area]*
Length of stroke data record	I	**GPQSK***[datalen]*
Current stroke data record		**GPQSK***[data]*
Current input character set identifier	I	**GPQICS***[csid]*
Initial view index	I	**GPQSK***[view]*
Number of points in initial stroke	I	**GPQSK***[npoint]*
List of initial points (WC)	*n*xR	**GPQSK***[point array]*
Input buffer size	I	**GPQSK***[buffer]*

Table 133. Workstation State List (WSL) Data Type Field Description (continued)

Description of Field	Data Type	Inquiry
Editing position	I	**GPQSK***[editpos]*
Current trigger list(s) (one per trigger list identifier)	*n* (3xI)	**GPQITS***[ltrigs]*
For Every Logical Input Device of Class Valuator:		
Operating mode (REQUEST, PRELE, EVENT)	E	**GPQVL***[mode]*
Echo switch (NOECHO, ECHO)	E	**GPQVL***[echosw]*
Prompt and echo type	I	**GPQVL***[echo]*
Current echo area (DC)	6xR	**GPQVL***[area]*
Length of valuator data record	I	**GPQVL***[datalen]*
Current valuator data record	I	**GPQVL***[data]*
Current input character set identifier	I	**GPQICS***[cisd]*
Current initial value	R	**GPQVL***[ivalue]*
Current range low value	R	**GPQVL***[lovalue]*
Current range high value	R	**GPQVL***[hivalue]*
Current trigger list(s) (one per trigger list identifier)	*n* (3xI)	**GPQITS***[ltrigs]*
For Every Logical Input Device of Class Choice:		
Operating mode (REQUEST, PRELE, EVENT)	E	**GPQCH***[mode]*
Echo switch (NOECHO, ECHO)	E	**GPQCH***[echosw]*
Prompt/echo type	E	**GPQCH***[echo]*
Current echo area (DC)	6xR	**GPQCH***[area]*
Length of choice data record	I	**GPQCH***[datalen]*
Current choice data record		**GPQCH***[data]*
Current input character set identifier	I	**GPQICS***[csid]*
Initial choice number	I	**GPQICS***[choice]*
Current trigger list(s) (one per trigger list identifier)	*n* (3xI)	**GPQITS***[ltrigs]*
For Every Logical Input Device of Class Pick:		
Operating mode (REQUEST, PRELE, EVENT)	E	**GPQPK***[mode]*
Echo switch (NOECHO, ECHO)	E	**GPQPK***[echosw]*
Prompt and echo type	I	**GPQPK***[echo]*
Current echo area (DC)	6xR	**GPQPK***[area]*
Length of pick data record	I	**GPQPK***[datalen]*
Current pick data record		**GPQPK***[data]*
Current input character set identifier	I	**GPQICS***[csid]*
Pick path order (TOP_FIRST, BOTTOM_FIRST)	E	**GPQPK***[order]*
Current pick path depth	I	**GPQPK***[depth]*

Table 133. Workstation State List (WSL) Data Type Field Description (continued)

Description of Field	Data Type	Inquiry
Current pick path	*n*xI	**GPQPK***[pickpath]*
Pick aperture	R	**GPQPKA***[size]*
Pick selection criteria (FIRST, LAST, ALL, FIRST_VISIBLE, LAST_VISIBLE, ALL_VISIBLE)	E	N/A
Pick correlation state (OFF, ON)	E	N/A
Pick Filters:		
Number of classes in inclusion filter	I	**GPQPK***[inclen]*
List of classes in inclusion filter	*n*xI	**GPQPK***[incl]*
Number of classes in exclusion filter	I	**GPQPK***[exclen]*
List of classes in exclusion filter	*n*xI	**GPQPK***[excl]*
Current trigger list(s) (one per trigger list identifier)	*n* (3xI)	**GPQITS***[ltrigs]*
For Every Logical Input Device of Class String:		
Operating mode (REQUEST, PRELE, EVENT)	E	**GPQST***[mode]*
Echo switch (NOECHO, ECHO)	E	**GPQST***[echosw]*
Prompt and echo type	I	**GPQST***[echo]*
Current echo area (DC)	6xR	**GPQST***[area]*
Length of string data record	I	**GPQST***[datalen]*
Current string data record		**GPQST***[data]*
Current input character set identifier	I	**GPQICS***[csid]*
Length of initial string	I	**GPQST***[strlen]*
Initial string	S	**GPQST***[string]*
Input buffer size	I	**GPQST***[buffer]*
Initial editing position	I	**GPQST***[editpos]*
Current trigger list(s) (one per trigger list identifier)	*n* (3xI)	**GPQITS***[ltrigs]*

The graPHIGS API Error State List (ESL)

The Error State List provides information on the current graPHIGS API error state.

The right-hand column lists the inquiry function that you can use when you want your application to know the error states of the system.

Data Type Field

The following correspondence applies to the abbreviated data types:

Table 134. Error State List (ESL) Data Type Field Definition

Data Type	Definition
I Integer	A whole number
R Real	A floating-point number

Table 134. Error State List (ESL) Data Type Field Definition (continued)

Data Type	Definition
S String	A character string
E Enumeration	A data type comprised of a set of values. The set is defined by enumerating the identifiers denoting the values.
n **Quantity**	This specifies an undesignated quantity of data. **Note:** The notation of *n* (number) x *t* (data type) indicates a collection of data of that type. This can be indicated in one of two ways: • By using notation such as 3xR (three real numbers), which could specify something like the *x*, *y*, and *z* coordinates of a three-dimensional point or RGB values • By using a variable number such as *n*xI, which specifies a collection of *n* integers.

Table 135. Error State List (ESL) Data Type Field Description

Description of Field	Data Type	Inquiry
Error state (OFF, ON)	E	N/A
Error reporting mode (OFF, ON)	E	**GPQEMO***[mode]*
Error File:		
Name	S	(passed on **GPOPPH**)
Information of Last Error:		
Workstation identifier	I	(passed to User Error Handler)
Message number	I	**GPQEMS***[number]*
Message text	S	**GPQEMS***[message]*

Utility Function State List (USL)

Utility functions aid in the definition of transformation viewing matrices. The USL is a temporary storage facility.

Data Type Field

The following correspondence applies to the abbreviated data types:

Table 136. Utility Function State List (USL) Data Type Field Definition

Data Type	Definition
I Integer	A whole number
R Real	A floating-point number
S String	A character string
E Enumeration	A data type comprised of a set of values. The set is defined by enumerating the identifiers denoting the values.

Table 136. Utility Function State List (USL) Data Type Field Definition (continued)

Data Type	Definition
n **Quantity**	This specifies an undesignated quantity of data. **Note:** The notation of *n* (number) x *t* (data type) indicates a collection of data of that type. This can be indicated in one of two ways: • By using notation such as 3xR (three real numbers), which could specify something like the *x*, *y*, and *z* coordinates of a three-dimensional point or RGB values • By using a variable number such as *n*xI, which specifies a collection of *n* integers.

Table 137. Utility Function State List (USL) Data Type Field Description

Description of Field	Data Type	Inquiry
View reference point (WC)	3xR	N/A
View plane normal (WC)	3xR	N/A
View up (WC)	3xR	N/A

Appendix B. Event Data Formats

Event Summary

The following table summarizes all events supported by the graPHIGS API. In this table, columns titled as ″major″ and ″minor″ show the contents of major and minor code parameters for various event related functions. When the column has ″None″, the corresponding parameter is not set by the graPHIGS API. The last column shows a GET function to be used for retrieving the detail event data of each event. When it has ″None″, the event has no event data to be retrieved. In this case, a GET function will result in an error.

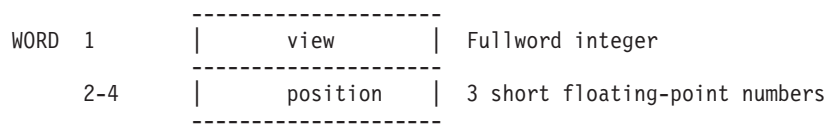
Table 138. Supported Events

Class	Meaning	Major	Minor	Data
1	Locator	Workstation ID	Device number	Locator
2	Stroke	Workstation ID	Device number	Stroke
3	Valuator	Workstation ID	Device number	Valuator
4	Choice	Workstation ID	Device number	Choice
5	Pick	Workstation ID	Device number	(Extended) Pick
6	String	Workstation ID	Device number	String
11	Locator_Break Event	Workstation ID	Device Number	None
12	Stroke_Break Event	Workstation ID	Device Number	None
13	Valuator_Break Event	Workstation ID	Device Number	None
14	Choice_Break Event	Workstation ID	Device Number	None
15	Pick_Break Event	Workstation ID	Device Number	None
16	String_Break Event	Workstation ID	Device Number	None
101	Link switch out	Workstation ID	0	None
102	Link switch in	Workstation ID	0	None
103	Update completion	Workstation ID	Display status	None
104	Input overflow events	Workstation ID	0	None
105	Window Resize Notification Event	Workstation ID	0	None
106	Window Exposure Notification Event	Workstation ID	0	Exposure data
107	Window Deletion Notification Event	Workstation ID	0	None
201	Broadcast message	Sender supplied	Sender supplied	Message
202	Private message	Sender supplied	Sender supplied	Message
301	Threshold Exceeded	Structure Store ID	Threshold value	None
401	Error Event	Error Number	0	None

Event Data Format

When the application specifies an event exit routine, the following event data and its length are also passed to the routine.

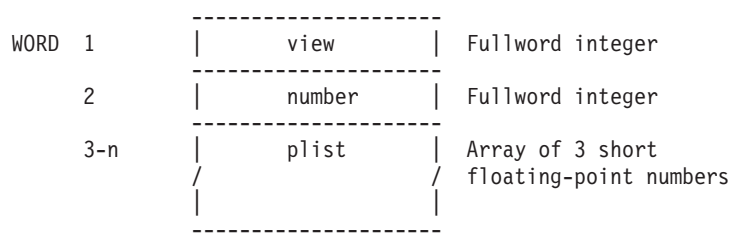
Locator Event (Event Class 1)



view View index in which the locator position resides.
position Locator 3-D position in world coordinates.

The length of this event data is always 16.

Stroke Event (Event Class 2)



view View index in which the stroke points reside.
number Number of points in the stroke point list.
plist A list of 3-D points in world coordinates.

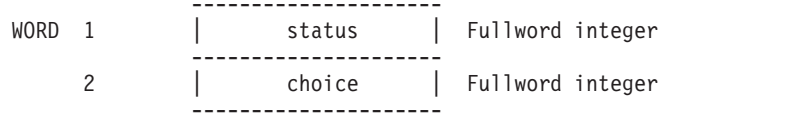
The length of this event data is *n* x 12 + 8.

Valuator Event (Event Class 3)

value Valuator value.

The length of this event data is always 4.

Choice Event (Event Class 4)



status Choice device status. This parameter takes one of the following values:

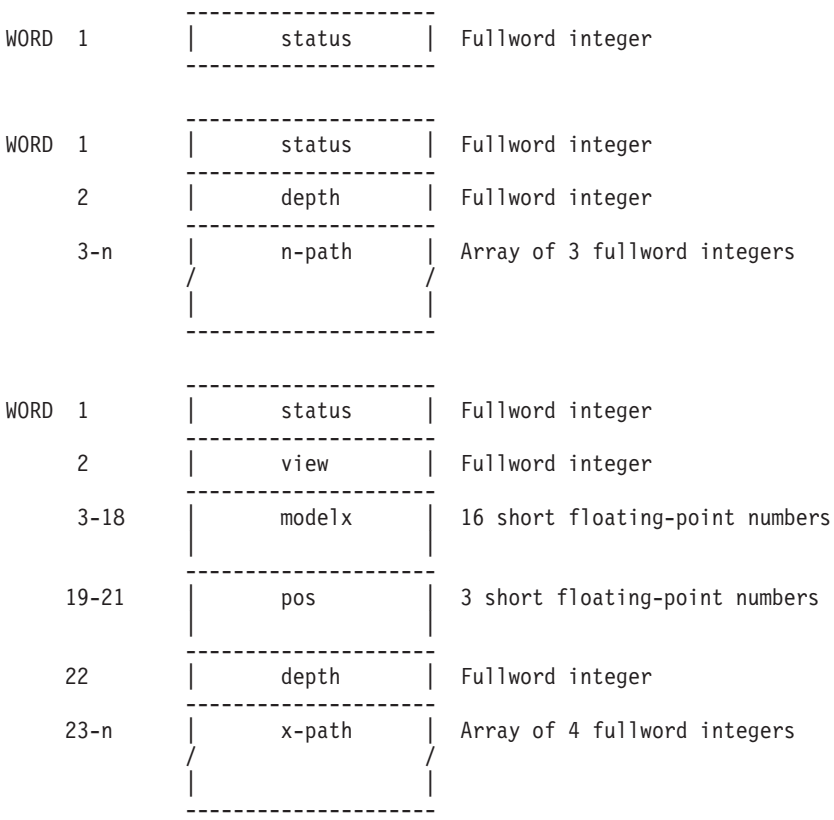
1 = NO_CHOICE

2 = OK

choice Choice alternative. When the choice status is 2 = OK, this parameter includes one of the choice alternatives available on the choice device. Otherwise, this parameter may contain any number.

The length of this event data is always 8.

Pick Event (Event Class 5)



status Pick status. This parameter takes one of the following values and specifies which format is actually used.

0 = NO_PICK

1 = OK_NORMAL PICK

2 = OK_EXTENDED PICK

When the pick status is 0 = NO PICK, the first format is used and so the length of the event data is 4. Otherwise, the event data is represented by the second or third format according to the pick device's type. If the device is the normal pick device, the second format is used. If the device is the extended pick device, the third format is used.

depth Pick path depth.

***n*-path** Normal pick path. This parameter is a list of pick path triplets, — a structure identifier, a pick identifier and an element number.

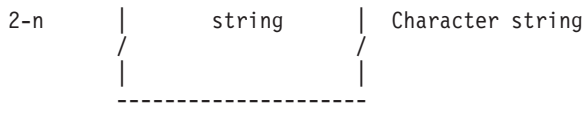
view View index of a view in which the picked primitive resides.

modelx Composite modeling transformation applied to the picked primitive. Elements of the transformation matrix are returned in the order M_{11},M_{12},M_{13},M_{14}...

pos 3-D position in NPC where the center of pick aperture existed when the pick occurred.

x-path Extended pick path. This parameter is a list of pick path quadruples, — a structure identifier, a pick identifier, a label identifier and a structure element number.

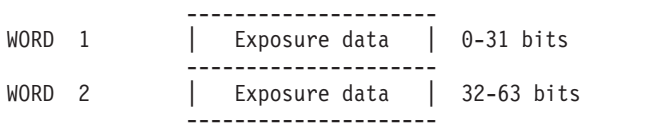
String Event (Event Class 6)



length Length of the character string in bytes (does not include the length field itself).

string Character string. When the length of this character string is not a multiple of 4, up to 3 padding bytes are supplied. Therefore, the length of this event data is ((*length*+7) /4)x4.

Window Exposure Event (Event Class 106)

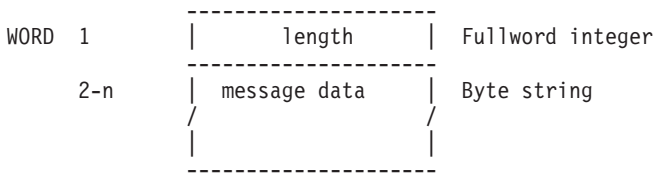


exposure data 64 bits of data indicating which of the views in numerical order are affected by the exposure event. Bit 0 is the flag for view 0 and is the most significant bit. Each bit is set as follows:

0 = NOT_AFFECTED_BY_EXPOSURE

1 = IS_AFFECTED_BY_EXPOSURE

Application Message Event (Event Class 201 and 202)



length Length of the character string in bytes (does not include the length field itself).

message data A byte string supplied by the sender. When the length of this datais not a multiple of 4, up to 3 padding bytes are supplied. Therefore, the length of this event data is ((*length*+7) /4)x4.

Appendix C. Plotting with graPHIGS

The graPHIGS API provides the capability for plotting CGM and GDF files produced by the graPHIGS API. You can plot GDF files on an IBM Color Plotter (6180, 6182, 6184, 6186, 7371, 7372, 7374, 7375). You can also plot GDF files on non-IBM plotters, as described in this section. You can plot CGM files on any plotter with HP GL2 support. For a list of supported plotters, as well as initial set-up instructions, see the appropriate graPHIGS API installation or customization manual:

- *The graPHIGS Programming Interface: Customization and Problem Diagnosis*
- *The GDDM/graPHIGS Programming Interface: Installation and Problem Diagnosis*.

Plotting on the RS/6000

When you select IBM plotter support at installation time, the `cpsI1` plot module for IBM plotters is installed in the **/usr/lpd** subdirectory. Likewise, if you select CalComp and/or Versatec support, the `cpsC1` plot module for CalComp plotters and/or the `cpsVI` plot module for Versatec plotters is installed in the **/usr/lpd** subdirectory.

Before proceeding, follow the instructions for setting up your environment:

For IBM Plotters	*The graPHIGS Programming Interface: Customization and Problem Diagnosis*
For CalComp Plotters	**/usr/lpd/README.ccp** **Note:** The graPHIGS API cpsC1 backend produces CalComp 906/907 format data.
For Versatec Plotters	**/usr/lpd/README.VERSA**
For HP GL2 Plotters	*The graPHIGS Programming Interface: Customization and Problem Diagnosis* and **/usr/lpd/cgm2hp2.readme**

You can use the graPHIGS API plot modules as printer backends, or execute them directly, bypassing the print queueing facilities.

Note: The graPHIGS API plot modules use the current working directory for certain input and output files. See the discussion of the *-ldir* option for more information.

Plotting GDF Files

Executing Plot Modules Directly

To plot a file, the following syntax can be used:

```
/usr/lpd/plotmodule -option  filename.gdf
                    ^      |
                    |------|
```

where:

- *plotmodule* is one of: `cpsI1` (for IBM), `cpsC1` (for CalComp), or `cpsV1` (for Versatec).
- *filename*.gdf is the name of the file to be plotted. *filename* is derived from the connection identifier. The GDF *filename* must be the last parameter passed to the plot module
- - *option* can be any of the options described below for controlling certain plotting functions.

Note:

- The symbol & may be added after the GDF filename to plot in the background:

```
/usr/lpd/cpsI1 -option  filename.gdf &
               ^       |
               |-------|
```

- If you are using the IBM plot module, `cpsI1`, and plotting to a plotter connected to a serial port, the option `-ttyx` must be specified, and **must** appear as the first option after the plot module name. `-ttyx` is the name of the `tty` port returned when the port was defined (for example, `tty0`, `tty1`, etc.). The `-ttyx` option is not needed if *-nopl* is specified to plot IBM data to a file.

  Examples:

  1. To plot the file **tmp0001.gdf** to an IBM plotter attached to the tty0 port:

     ```
     /usr/lpd/cpsI1 -tty0 tmp0001.gdf
     ```

  2. To plot the same file in the background, and to rotate the plot 90 degrees:

     ```
     /usr/lpd/cpsI1 -tty0 -rot tmp0001.gdf &
     ```

  3. To convert the file **tmp0001.gdf** to CalComp data:

     ```
     /usr/lpd/cpsC1 -ispi tmp0001.gdf
     ```

  4. To convert the file **tmp0001.gdf** to Versatec data:

     ```
     /usr/lpd/cpsV1 -ispi tmp0001.gdf
     ```

Using Plot Modules as Printer Backends

A plot can be scheduled through **smit** or by using the **enq** command.

```
enq -P queuename -o option  filename.gdf
                 ^        |
                 |--------|
```

where:

- *queuename* is the name of the print queue.
- *-option* is any option to be passed to the backend program.
- *filename*.gdf is the name of the GDF file to be plotted.

Plotting Options

The following is a list of optional parameters which can be passed to the plot modules. You can specify plot options on the command line or you can include, in your **plot** command, an option that identifies a file containing the plot options you wish to use. This is particularly useful if you use many options or if you use the same options repeatedly.

Note: When they are used in the option file, some options use syntax different from that described below. See the description of using options in an option file.

-af

Area Fill (*-af*) allows for the generation of plotter hardware polygon commands, which greatly reduces the amount of data the application sends to the plotter for polygons. These commands are not supported on the 7371, 7372, or the 6180 plotters.

When you specify the *-af* option, the graPHIGS API must determine the maximum number of polygon vertices that can be sent to your plotter. This value is called *max_poly_points* and has a default of 2000. You can override the default by specifying a new value in the option file (not on the command line) using the *max_poly_points* keyword. If this value is exceeded by your polygon data, the area fill will be processed by software.

> **Note:** *-af* is supported on the 6180 if a Graphics Enhancement Cartridge (P/N 5452389) is installed on the plotter.
>
> Early models of the 7374 and 7375 plotters did not support polygon mode commands. If you are using this option parameter while plotting to one of these plotters, you may experience unexpected area fill output. To plot normally, remove this option from the list of parameters.

Supported on: cpsI1

-angXX

Angles *XX* (*-angXX*) is used to chop angles below *XX* degrees where *XX* is an angle between 01 and 40. The default is 40 if *-angxx* is not specified. See the description of *-nochop*.

Supported on: cpsI1, cpsC1, cpsV1

-chopall

Chop all (*-chopall*) forces chopping of both inside and outside facing angles. See the descriptions of *-chopout* and *-nochop*. *-chopall* is the default when no other chopping options are used.

Supported on: cpsI1, cpsC1, cpsV1

-chopout

Chop Outside (*-chopout*) forces chopping of outside facing angles only. See the description of *-nochop*.

Supported on: cpsI1, cpsC1, cpsV1

-c16

CalComp 16 (*-c16*) specifies that 16 colors are to be processed when using the information.

Supported on: cpsC1

-es

Exact Scaling (*-es*) causes the plot to be scaled to the appropriate size (for example, 1mm design = 1mm plot length) if you specify the *-es* parameter option and use the graPHIGS API escape function 1003 (GDF/CGM plot size).

Supported on: cpsI1

-esx

Exact Scaling, Expanded (*-esx*) plotting area functions as *-es* above, but the plotter hard-clip limits are used, allowing for a larger plotting area. However, plotter pinch wheels may run over plotted lines causing ink smears. Refer to your plotter operations manual for more information on hard-clip limits. The page size is determined from the escape function 1003 (GDF/CGM plot size).

Supported on: cpsI1

-hinXX=nn.nnn

Hatchspace (inches) allows you to specify the spacing for a fill pattern where *nn.nnn*=.001-99.999, defining the space in inches, and *XX*=01-16, identifying the fill pattern. For example, `-hin12=1.275`. The pattern number specified refers to the graPHIGS pattern only when the default hatch table is used. This option is used in conjunction with either the *-es* or *-esx* option.

-hcmXX=nn.nnn

Hatchspace (centimeters) allows you to specify the spacing for a fill pattern where *nn.nnn*=0.001-99.999, defining the space in centimeters, and *XX*=01-16, identifying the fill pattern. For example, `-hcm08=0.075`. The pattern number specified refers to the graPHIGS pattern only when the default hatch table is used. This option is used in conjunction with either the *-es* or *-esx* option.

Supported on: cpsI1, cpsC1, cpsV1

-hsfix

Hatchspace (fixed) forces fixed spacing of fill patterns regardless of plot size and scale. This option is used in conjunction with either the *-es* or *-esx* option.

Supported on: cpsI1, cpsC1, cpsV1

-hsprop

Hatchspace (proportional) forces proportional spacing in fill patterns by maintaining a constant number of lines with spacing proportional to the scale of the data file. This option does not affect any fill patterns with fixed values set by the *-hinXX* or *hcmXX* options. This option is used in conjunction with either the *-es* or *-esx* option.

Supported on: cpsI1, cpsC1, cpsV1

-ispi

Industry Standard Plotting Interface (*-ispi*) enables the use of the Industry Standard Interface calls.

Note: The *-es* option is automatically in effect when *-ispi* is used.

Supported on: cpsC1, cpsV1

-ldir

Local directory specifies which directory is to be used for plotting inputs and outputs. This directory will be passed to the graPHIGS API plotting backends so that plot files need not be restricted to the current directory or the **/usr/lpd/qdir** directory when queueing facilities are used. For example, the command,

```
enq -Pgdf -o -ldir/u/design/plot  fileb.gdf
```

tells the graPHIGS API plotting backend to use the **/u/design/plot** directory for input and output, however, the current directory would still be searched for **fileb.gdf** because the *-ldir* option does not affect the search location for GDF files when queueing is used. The qdaemon automatically passes a fully qualified pathname, based on your command line input, to the backend. If direct backend plotting is used, the *-ldir* option affects all files.

The **log.fil**, generated whenever a plotting backend is executed, includes a message indicating whether *-ldir* was successfully used.

The *-ldir* option cannot be specified in an option file.

Supported on: cpsI1, cpsC1, cpsV1

-nobord

No border suppresses the drawing of a smoothing edge around the perimeter of filled polygons drawn with the edge flag off. The smoothing edge eliminates the jagged edge often seen in filled circles drawn with a wide pen. It consists of a solid polyline of the same color specified for the interior fill, drawn in a width of 1. Because the smoothing edge centers on the border of the polygon, the radius will be drawn half a pen-width larger than specified. If this is a problem specify *-nobord* and use a smaller pen-width to reduce the jagged edges.

Supported on: cpsI1, cpsC1, cpsV1

-nochop

No chopping (*-nochop*) turns all chopping off. When plotting wide lines using the default multiple pen stroking, adjacent line segments are connected using a mitered join style. This can cause undesired results when adjacent line segments form a sharp angle. The join style applied can cause the join at one of these angles to 'spike' to a point farther than expected. This can be seen quite often when plotting arrowheads or geometric text.

By default, this effect is avoided by *chopping* the spike back to the desired position when stroking wide lines, but specifying *-nochop* turns all chopping off. The default chopping occurs for any angle less than 40 degrees. Both inside and outside facing angles are chopped by default.

This parameter is ignored if *-nolw* is being used.

Supported on: cpsI1, cpsC1, cpsV1

-noin

No Initialization (*-noin*) allows the user to set up the P1 and P2 positions on the plotter. This allows the user to position the plot anywhere on the paper. For more information on P1 and P2, refer to your plotter operations manual.

Supported on: cpsI1

-nolw

No Line Width (*-nolw*) disables line width processing and causes lines to be single stroked. This can be used to improve pen plotting performance or to work in conjunction with the *-pens* parameter described below.

Supported on: cpsI1, cpsC1, cpsV1

-nopl

No plot (*-nopl*) redirects the IBM-GL output to the file *fname*.gl.

Supported on: cpsI1

-optf

Option file is used as an alternative to entering plotting options on the command line. You can specify some or all of your options in an option file that is specified with a filename right after the *-optf* option parameter, for example `-optfmyopt1.fil` identifies **myopt1.fil** as the option file to be used in the plot.

Note: There is no blank between the *-optf* option and the filename specified.

Although only one option file may be specified from the command line, an option file may call additional option files up to a total of ten. Options specified on the command line, however, always override those specified in an option file.

Supported on: cpsI1, cpsC1, cpsV1

-pens

The *-pens* parameter option causes the file **pentbl.fil** to be used for mapping GDF color and line width values to the actual pens being used on the plotter. Use this option to provide the plotting routines with information about the plotter pens. This information is then used when deciding which pen number to select based on the current line color and thickness. The *pentable* option can be used in the option file to identify a different filename for pen information.

The **pentbl.fil** file should be placed in the current working directory. This file is only accessed when *-pens* is specified.

Note: If you do not specify *-pens*, then the default pen selection is based on a default set of pen values, which is the same as the example **pentbl.fil** shown.

Supported on: cpsI1, cpsC1, cpsV1

-ps=x

Paper Size (*-ps=x*) determines the page size, where *x* is defined as a, b, c, d, or e. This option parameter is needed to allow for proper area fill and line thickness when plotting data to a file, and is not needed when plotting directly to an IBM plotter. It is not necessary to use *-ps=x* when *-es* or *-ispi* is used, because the page size is determined from the exact scale escape function 1003 (GDF/CGM plot size). If you do not specify the *-ps=x* option, the default value is *-ps=a*.

Supported on: cpsI1

-rot >Rotates (*-rot*) the plot 90 degrees. This is not supported on the 7371 and cannot be used in conjunction with *-noin*.

Supported on: cpsI1

-slow Slow (*-slow*) pen speed, slows down the the plotter pen velocity. Use this parameter option for transparencies.

Supported on: cpsI1

-ttyX The serial port option allows you to specify the name of the port, `tty0` for example, where the plotter is connected. This option must be the first one specified after the plot module name. The default is none. This option is not used if *-nopl* is specified.

Supported on: cpsI1

VEROUT Versatec output is for use with the **cpsV1** command. This option allows you to specify the path and filename for the Versatec plot output file. For example,

```
cpsV1 -ispi -o -VEROUT$HOME/output.plot  fname.gdf
```

would produce a plot output file in the user's home directory with the name, **output.plot**

Supported on: cpsV1

-VMSGS Versatec message is for use with the **cpsV1** command. This option allows you to specify the path and filename for the Versatec plot summary and error messages file. See *-VEROUT* for a command example.

Supported on: cpsV1

-VPARM Versatec parameter is for use with the **cpsV1** command. This option allows you to specify the path and filename for the Versatec plotting parameters file. See *-VEROUT* for a command example.

Supported on: cpsV1

-v16 Versatec 16 (*-v16*) specifies that 16 colors are to be processed when using the more information.

Supported on: cpsV1

Using the Option File

You can specify options in an option file by using *-optf*. You can create the option file by using an editor and following these syntax rules:

- All options must begin in column 1.
- Each option must be specified as exceptions cited below.
- Each option must be followed by a colon (:) delimiter and a repetition of the option without the leading hyphen (-), or by the word `none`. Specifying `none` turns the option off.
- Comment lines can be used by specifying a number sign (#) in column 1.
- Continuation lines are identified by specifying a backslash (\) at the end of the preceding line.

For example:

```
# option file for cpsI1
#
-es:es
-pens:pens
-nopl:none
```

Exceptions:

-chopall, *-chopout*, *-nochop* These options are mutually exclusive. Specifying the desired option turns the others off. Specifying `none` is not supported for these options.

-c16 and *-v16*
These options are specified in the option file using the same syntax as that to force either of these option off, specify *-cv16:none* in an option file.

hatchspace
The *-hinXX* and *-hcmXX* option parameters are not used in an option file. Instead, use *hatchspaceX:in=nn.nnn* or *hatchspaceX:cm=nn.nnn* where *X* is any hatch pattern from 1 to 16, and *nn.nnn* is any spacing value, expressed in inches or centimeters, from 0.001 to 99.999. If hatchspace is specified for any hatch pattern in *both* an option file and on the command line, then the value from the command line has priority.

Supported on: cpsI1, cpsC1, cpsV1

log_name
In order to use the *logfile* option, you must specify it as the first option in the first option file. Otherwise, the default log file **log.fil** will be used.

There are three ways in which you can use the *logfile* option:

Specify *log_name:date* when you want to create a file that will include the current date in the filename. For example, **apr05.log** would be created for the first plot run on April 5, and would be used for each successive plot that day with each log appended to the end of the previous log.

Use logfile to have the same filename as the gdf file to be plotted. For example, a plot of file **gear.gdf** would generate a logfile named gear.log.

Use *log_name:XXXXXX* to specify a logfile name of your choice in an operating system-recognizable format. The logfile is affected by the current directory.

Supported on: cpsI1, cpsC1, cpsV1

max_poly_points
For use with the *-af* option, *max_poly_points* specifies the maximum number of polygon vertices that can be sent to your plotter. The default value for *max_poly_points* is 2000 and the maximum is 8185 points. If you do not specify *-af*, then *max_poly_points* is ignored.

The graPHIGS API generates plotter data using approximately 11 bytes per coordinate pair. Depending on size of your plotter's polygon buffer (the plotter's *Guide to Operations* contains this information), you may be able to optimize performance by setting the maximum vertices higher or lower than the default value of 2000. For example, a 1k polygon buffer might operate most efficiently with *max_poly_points*=90 because 11 bytes[default]90=990. Any vertices exceeding the 1k capacity of the plotter hardware are handled by the backend, reducing performance. The 11 bytes/vertex formula is only an approximation, however, and trial and error may reveal a more efficient setting.

Supported on: cpsI1

optfname
Additional option files are specified differently within an option file than on the command line. The *-optf* parameter is not recognized within an option file. Instead, specify it as *optfname:*, for example, `optfname:myopt2.fil` could be used to call **myopt2.fil** from within another option file.

Use fully-qualified pathnames with the *optfname* option.

Supported on: cpsI1, cpsC1, cpsV1

pentable
Specify *pentable:XXXXX.fil* to identify a file other than the default **pentbl.fil** for use when the *pens* option is specified. For example, `pentable:/tmp/carousl2.fil` would identify `/tmp/carousl2.fil` for use as a pentable file.

Use fully-qualified pathnames with the *pentable* option.

Supported on: cpsI1, cpsV1

serial_port

To select a serial port from within an option file, specify *serial_port:XXX*, where *XXX* identifies the port. For example, `serial_port:tty3` selects plotter port tty3. On the command line, the serial port is specified as the first option following the **plot** command, and includes only the port name preceded by a hyphen, for example, `cpsI1 -tty1`.

Supported on: cpsI1

Option Priority: Options are assigned values in the order in which they are processed. For example, if an option file sets the paper size (`-ps=a:ps=a`) but then calls a second option file that also sets the paper size (`-ps=b:ps=b`), size `b` remains in effect. However, if the first option file specifies `-ps=a:ps=a` at a point *after* it has called the second option file, then paper size `a` remains in effect because it was the last paper size setting processed. Command line options supersede those specified in any option file. Thus, if you specify a plotting option differently on the command from your option file specifications, the command line options are used.

In the following example, an option file (**myopt1.fil**) is called, which in turn, calls **myopt2.fil**:

```
cpsI1 -ps=b -optfmyopt1.fil wheelc3.gdf
```

If **myopt1.fil** contains:

```
# option file 1
-logfilename:mylog.fil
-chopout:chopout
optfname:myopt2.fil
-ang35:ang35
-pens:pens
```

And **myopt2.fil** contains:

```
# option file 2
-ps=c:ps=c
-pens:none
```

The options that remain in effect for the plot are *chopout, ang35*, and *pens* because the *ang40* and *-pens:none* specifications in **myopt2.fil** are superseded by the specifications later encountered when processing returns to **myopt1.fil**. The paper size is determined by *-ps=b* on the command line.

Plotting Options Summary

The following table summarizes the plotting options:

Table 139. Plotting Options

Option	cpsI1	cpsC1	cpsV1
-af	X		
-angXX	X	X	X
-chopall	X	X	X
-chopout	X	X	X
-c16		X	
-es	X		
-esx	X		
-hinXX=nn.nnn	X	X	X
-hcmXX=nn.nnn	X	X	X
-hsfix	X	X	X
-hsprop	X	X	X

Table 139. Plotting Options (continued)

Option	cpsl1	cpsC1	cpsV1
-ispi		X	X
-ldir	X	X	X
-nobord	X	X	X
-nochop	X	X	X
-noin	X		
-nolw	X	X	X
-nopl	X		
-optf	X	X	X
-pens	X	X	X
-ps=x	X		
-rot	X		
-slow	X		
-ttyX	X		
-VEROUT			X
-VMSGS			X
-VPARM			X
-v16			X
hatchspace	X	X	X
log_name	X	X	X
max_poly_points	X		
optfname	X	X	X
pentable	X		X
serial_port	X		

Controlling Directories

Depending on the combination of options you specify, plot log and output files can be written to the current directory, the **/usr/lpd/qdir** directory, or another directory of your choice. The following sequence of events explains how the various directories are selected:

1. If the **enq** command is used to schedule the plot:
   a. The qdaemon adds the full path to the filename of the gdf data file.
   b. The qdaemon changes the current directory to **/usr/lpd/qdir**.

   If **enq.** is not used, then execution
2. The **cps** *X* plot module determines the directory as follows:
   a. Is the *-ldir* option used? If so, change to the directory specified by the *-ldir* option and continue
   b. Is **GPPLOTDIR** defined as a system environment variable? If so, change to the directory specified in **GPPLOTDIR** and continue as shown in If not,
   c. Is PWD defined as a system environment variable? If so, change to the directory specified in PWD and continue as If not, the current directory is not changed.
3. The plot is executed in whatever directory is current.
4. The gdf, logfile, pentable file, and option files are opened using the exact names specified. If **enq** was used to schedule the plot, then the gdf file has a fully-qualified pathname.

5. Is the *-nopl* option specified? If so, then the output gl file is opened using the name of the gdf file, and the .gdf extension is replaced by the .gl extension. If the gdf filename was a fully qualified pathname, then the gl file will be opened as a fully qualified filename.

pentbl.fil description

Table 140. pentbl.fil File

pentbl.fil File

```
 2            <------ Row 1: Options
 1    1    0.3      --
 2    2    0.3     |
 3    3    0.3     |
 4    4    0.3 <---|  Rows 2 - 9: Pen data
 5    5    0.3     |
 6    6    0.3     |
 7    7    0.3     |
 8    8    0.3      --
---  ---  -----
 |    |     |
 |    |     |-- Diameter (columns 9 - 15): Range = 0.0 - 99.9999 (mm)
 |    |--Color (columns 3 - 8): Range = 0 - 65535
 |--Slot (columns 1 - 2): Range = 1 - 8
```

Table 141. pentbl.fil File Parameters

Row	Columns	Type	Range	Description
1 (Options)	1-2	Integer	1,2	Search Priority 1. Line thickness (LWSC) - the pen will be selected whose pen tip diameter is most appropriate for the current line thickness. If more than one exists, then the correct color pen will be selected from these choices. 2. Color index (CI) - the pen will be selected whose color matches the current color index. If more than one exists, these will be searched for the one whose pen tip diameter is most appropriate for the current line thickness.
2-9 (Pen data)	1-2	Integer	1-8 (or 1-16, see below)	Slot in pen carousel on plotter. This field has been included for clarity and its values are ignored. Note: There must be a minimum of eight pen-data entries unless the -c16 or -v16 options are used which require a minimum of 16 pen-data entries.

Table 141. pentbl.fil File Parameters (continued)

Row	Columns	Type	Range	Description
2-9 (Pen data)	3-8	Integer	0,1,2, ... 65535	Color of pen. This field specifies the color of the pen in the corresponding slot in the carousel. The following GDDM/GDF defined values are supported by the graPHIGS API. See below for the R,G,B components associated with these colors. 1. Blue 2. Red 3. Magenta 4. Green 5. Cyan 6. Yellow 7. Black 8. Background 9. Dark Blue 10. Orange 11. Purple 12. Dark Green 13. Dark Cyan 14. Mustard 15. Grey 16. Brown
2-9 (Pen data)	9-15	Floating-point	0.0 ... 99.9999	Pen tip diameter (in millimeters). This value will be used to map line thickness to an appropriate pen number based on the current line width scale factor and the graPHIGS API nominal line width of 0.269mm.

Red, Green, Blue Components of GDF colors

The graPHIGS colors described above can be derived from Red, Green, and Blue components in the following way:

Find the values of Rval, Gval, and Bval using the following chart and the appropriate Red, Green, and Blue components, then refer to the table below to find the color based on these values:

Table 142. Color Values

Amount of Red	Value of Rval
<0.33	0
<0.66	1
>=0.66	2
Amount of Green	**Value of Gval**
<0.25	0
<0.50	1
<0.75	2
>=0.75	3
Amount of Blue	**Value of Bval**
<0.33	0
<0.66	1

Table 142. Color Values (continued)

Amount of Red	Value of Rval
>=0.66	2

Table 143. R,G,B Mapping Table

Rval	Gval	Bval	Color Number	Color
0	0	0	8	Background (not drawn)
0	0	1	9	Dark Blue
0	0	2	1	Blue
0	1	0	12	Dark Green
0	1	1	9	Dark Blue
0	1	2	1	Blue
0	2	0	12	Dark Green
0	2	1	13	Turquoise
0	2	2	5	Cyan
0	3	0	4	Green
0	3	1	13	Turquoise
0	3	2	5	Cyan
1	0	0	2	Red
1	0	1	11	Purple
1	0	2	3	Magenta/Pink
1	1	0	16	Brown
1	1	1	15	Grey
1	1	2	3	Magenta/Pink
1	2	0	16	Brown
1	2	1	15	Grey
1	2	2	15	Grey
1	3	0	6	Yellow
1	3	1	15	Grey
1	3	2	7	Black
2	0	0	2	Red
2	0	1	11	Purple
2	0	2	3	Magenta/Pink
2	1	0	10	Orange
2	1	1	11	Purple
2	1	2	3	Magenta/Pink
2	2	0	14	Mustard
2	2	1	15	Grey
2	2	2	7	Black
2	3	0	6	Yellow
2	3	1	7	Black
2	3	2	7	Black

Mapping GDF Colors to Pen Numbers

By default, all three plotting backends will map the 16 GDF to see how this can be changed for the CalComp and Versatec backends.

If the *-pens* option was NOT specified, then the mapping from 16 colors to 8 pens is done using a 'MOD 8' function. That is, color 9 is mapped to pen 1, color 10 is mapped to pen 2, etc. The exception to this is color 16, which is mapped to pen 7.

If the *-pens* option is used, the mapping from 16 colors to 8 pens is done using the **pentbl.fil** file. See the descriptions of the `pentbl.fil` and the *-pens* option.

EXAMPLE:

An application that uses graPHIGS sets up a color table in order to be able to apply color to a given object. The colors are created by specifying an amount of red, green, and blue in the range of 0.0 to 1.0.

Application specifies Blue as RED=0.1, GREEN=0.3, BLUE=0.9 Application specifies Purple as RED=0.3, GREEN=0.27, BLUE=0.7.

Note: The application color Blue is just a label that represents the three values of red, green, and blue. The same values of red, green, blue could also have been the application color Cyan.

graPHIGS then converts the amount of red, green, and blue into red, green, blue value numbers according to

EXAMPLE:

From the previous example, the application colors Blue and Purple would be converted into:

	Rval	Gval	Bval
Blue	0	1	2
Purple	0	1	2

Note: Two distinct application colors can be mapped to the same set of red (Rval), green (Gval), blue (Bval) values.

The red, green, blue values are then transformed into GDF file format

EXAMPLE:

From the previous example, the application colors Blue and Purple would both be converted to color 1 (Blue) in the GDF file.

Note: A graPHIGS generated GDF file will only contain color numbers 1 - 16.

Once a GDF file has been created it can then be plotted. The graPHIGS plotting code is also capable of remapping colors with the *-pens* option. The setup of the colors on the plotter also plays an important part in the color mapping process.

Example 1

The graPHIGS plotting code is invoked WITHOUT the *-pens* option. This is the same as invoking the plotter code with the -pens option and the default **pentbl.fil** file. Assume the plotter's pen colors are set up to be the same as the GDF color numbers.

pentbl.fil (default)		Plotter
2 (color priority)		
Slot GDF Color Diameter		Slot GDF Color Diameter
1 1 Blue .3		1 1 Blue .3
2 2 Red .3		2 2 Red .3
3 3 Magenta .3		3 3 Magenta .3
4 4 Green .3		4 4 Green .3
5 5 Cyan .3		5 5 Cyan .3
6 6 Yellow .3		6 6 Yellow .3
7 7 Black .3		7 7 Black .3
8 8 Background .3		8 8 Background .3

GDF file contains	Explanation
1) a color 2 (Red) line .3mm thick	1) pen 2 is selected because it is the only red pen
2) a color 1 (Blue) line .9mm thick	2) pen 1 is selected because it is the only blue pen. line will be blue and stroked multiple times to achieve line thickness if -nolw option is not used.
3) a color 5 (Cyan) line .1mm thick	3) pen 5 is selected because it is the only cyan pen. Line will be drawn .2mm too wide because only .3mm pen available
4) a color 9-16 line any thickness	4) colors 9-15 will be mapped to GDF colors 1-7 (pens 1-7) respectively and color 16 will be mapped to GDF color 7 (pen 7) NOTE: colors 9-16 will not be remapped if -c16 or -v16 option is used

Example 2

The graPHIGS plotting code is invoked WITHOUT the *-pens* option. This is the same as invoking the plotter code with the *-pens* option and the default **pentbl.fil** file. The plotter IS NOT set up according to GDF color defaults.

pentbl.fil (default)		Plotter
2 (color priority)		
Slot GDF Color Diameter		Slot GDF Color Diameter
1 1 Blue .3		1 7 Black .1
2 2 Red .3		2 4 Green .3
3 3 Magenta .3		3 3 Magenta .3
4 4 Green .3		4 2 Red .3
5 5 Cyan .3		5 5 Cyan .3
6 6 Yellow .3		6 6 Yellow .3
7 7 Black .3		7 1 Blue .3
8 8 Background .3		8 8 Background .3

GDF file contains	Explanation
1) a color 2 (Red) line .3mm thick	1) pen 2 is selected because pen 2 is still red according

	to the default color table. line will be drawn green.
2) a color 1 (Blue) line .9mm thick	2) pen 1 is selected because pen 1 is still blue according to the default color table. line will be drawn black. the line will have .2mm gaps because the plotting code thinks it has a .3mm pen and moves the pens accordingly to stroke the line to .9mm
3) a color 12 (Dark Green) line .3mm thick	3) color 12 is mapped to color 4 (green) pen 4 is selected because pen 4 is still green according to the default color table. line will be drawn red.

The *-pens* option causes the file **pentbl.fil** to be used for mapping GDF colors and line width values. If the number at the top of the file is a 1 then a pen will be selected whose pen tip diameter is most appropriate for the current line thickness. If more than one exist then the pens will be checked to see if one of them is the desired color. If none of them are the desired color, then the first pen of the correct size is picked. If the number at the top of the file is a 2 then the correct color will be picked. If more than 1 pen of the correct color exists, the pen best suited for the current line thickness will be chosen.

Example 3

The graPHIGS plotting code is invoked WITH the -pens option. The plotter IS setup according to the `pentbl.fil`. The `pentbl.fil` option specifies LINE THICKNESS search priority.

pentbl.fil	
1 (line thickness priority)	
Slot GDF Color Diameter	
1 7 Black .2	
2 7 Black .25	
3 7 Black .3	
4 4 Green .3	
5 5 Cyan .7	
6 6 Yellow .7	
7 7 Black .4	
8 8 Background .3	

Plotter	
Slot GDF Color Diameter	
1 7 Black .2	
2 7 Black .25	
3 7 Black .3	
4 4 Green .3	
5 5 Cyan .7	
6 6 Yellow .7	
7 7 Black .4	
8 8 Background .3	

GDF file contains	Explanation
1) a color 7 (Black) line .3mm thick	1) first all pens = .3mm are selected (pens 3,4,8) then these pens are searched for the desired color. pen 3 is selected.
2) a color 7 (Black) line .7mm thick	2) first all pens = .7mm are selected (pens 5,6) then these pens are searched for the desired color. since neither pen is black the first pen of the appropriate size is selected (pen 5) line will be drawn cyan.
3) a color 5 (Cyan) line .35mm thick	3) since there are no .35mm pens in the table the closest pen size(s) without going over are selected (pens 3,4,8)

	these pens are searched for desired color. Since none of the pens are cyan, pen 3 is selected. line is drawn black.
4) a color 4 (Green) line .1mm thick	4) since there are no .1mm pens in the table and all the pens are > .1mm the closest pen(s) is selected (pen 1). line is drawn black.

Example 4

The graPHIGS plotting code is invoked WITH the *-pens* option. The graPHIGS plotting code is also invoked WITH the *-nolw* option. The plotter IS setup according to the `pentbl.fil`. The `pentbl.fil` option specifies LINE THICKNESS search priority.

pentbl.fil

1 (line thickness priority)

Slot	GDF Color	Diameter
1	7 Black	.2
2	7 Black	.25
3	7 Black	.3
4	4 Green	.3
5	5 Cyan	.7
6	6 Yellow	.7
7	7 Black	.4
8	8 Background	.3

Plotter

Slot	GDF Color	Diameter
1	7 Black	.2
2	7 Black	.25
3	7 Black	.3
4	4 Green	.3
5	5 Cyan	.7
6	6 Yellow	.7
7	7 Black	.4
8	8 Background	.3

GDF file contains	Explanation
1) a color 7 (Black) line .23mm thick	1) since there are no .23mm pens in the pen table the closest pen is selected (pen 2). NOTE: the pen was selected even though it was larger than the line to be drawn. the -nolw option tells the plotting code to pick the pen closest in size to the desired pen size whether it is larger or smaller.

Example 5

The graPHIGS plotting code is invoked WITH the *-pens* option. The plotter IS setup according to the `pentbl.fil`. The `pentbl.fil` option specifies COLOR search priority.

pentbl.fil

2 (Color search priority)

Slot	GDF Color	Diameter
1	11 Purple	.2
2	2 Red	.25
3	10 Orange	.3
4	4 Green	.3
5	5 Cyan	.7
6	7 Black	.8

Plotter

Slot	GDF Color	Diameter
1	11 Purple	.2
2	2 Red	.25
3	10 Orange	.3
4	4 Green	.3
5	5 Cyan	.7
6	7 Black	.8

```
| 7    7 Black       .4    |          | 7    7 Black       .4    |
| 8    8 Background  .3    |          | 8    8 Background  .3    |
-----------------------------         -----------------------------

-------------------------------------------------------------------
|     GDF file contains          |        Explanation             |
-------------------------------------------------------------------
| 1) a color 11 (Purple) line    | 1) pen 1 is selected because it |
|    .8mm thick                  |    is the only purple pen even  |
|                                |    though pen 6 would draw the  |
|                                |    line faster                  |
| 2) a color 7 (Black) line .9mm | 2) first all color 7 (black)    |
|    thick                       |    pens are selected (pens 6,7) |
|                                |    these pens are then check    |
|                                |    to see which would fill the  |
|                                |    line faster. pen 6 is        |
|                                |    selected.                    |
| 3) a color 14 (Mustard) line   | 3) first color 14 is searched   |
|    .3mm thick                  |    for in the pen table. since  |
|                                |    it is not found it is mapped |
|                                |    to color 6. Since color 6 is |
|                                |    also not in the pen table    |
|                                |    pen 1 is used by default.    |
|                                |    line is drawn purple.        |
-------------------------------------------------------------------
```

Notes:

1. The above examples work the same way with *-c16* or *-v16* option except the pen table can have slots 1 - 16 instead of just 1 - 8 and colors 9 - 15 are not mapped to colors 1 - 7 respectively nor is color 16 mapped to color 7.
2. The *-nolw* option disables line width processing and causes lines to be single stroked. This option may causes pens sizes to be selected that are greater than the width of the line to be drawn. It is intended as a means to increase performance.
3. CalComp and Versatec each have color definition files which can be used to further map colors. For further information, please refer to **/usr/lpd/README.ccp** for CalComp and **/usr/lpd/README.VERSA** for Versatec.

Example Summary

There are many levels in which colors can be mapped. Mapping can take place in just one level or in multiple levels.

- APPLICATION COLOR - All colors mapped to GDF colors 1 - 16
- GDF FILE COLOR - GDF colors 1 - 16 mapped to graPHIGS pen table file
- graPHIGS PEN TABLE - Maps colors in the pen table to the colors on the plotter
- CalComp color definition file - If CalComp plotter, maps colors from graPHIGS pen table file to plotter
- Versatec color definition file - If Versatec plotter, maps colors from graPHIGS pen table file to plotter
- PLOTTER - Color output

Overview of Color Processing Algorithms used

OPTION 1 (*-pens* used, priority set to 1 (line thickness))

```
IF the -nolw option was specified THEN
    search the pen table for the pen(s) that are closest to
    the line width to be drawn. there may be multiple pens equally
    close. the pen(s) selected may be either equal, greater than,
    or less than the line width to be drawn.
  ELSE
    search the pen table for the pen(s) that are closest to
    the line width to be drawn. there may be multiple pens equally
    close. the pen(s) selected may be either less than or equal to
    the line width to be drawn if possible.
    IF all pens in the pen table are greater than desired width THEN
      select the pen closest to the line width to be drawn.
```

```
IF multiple pens were selected THEN
  from the pens selected as having the desired width search for a pen
  that also has the desired color. there may be multiple pens of the
  desired color. the first one will be selected.
  IF the desired color can not be found THEN
    IF (the -c16 option was not specified) and
       (the -v16 option was not specified) and
       (the desired color is a GDF color in the range of 9 to 16) THEN
      map colors 9-15 to colors 1-7 respectively and map color 16
      to color 7.
      from the pens selected as having the desired width search
      for a pen that also has the remapped color. there may be
      multiple pens of the remapped color. the first one will be
      selected.
      IF the remapped color can not be found THEN
        from the pens selected as having the desired width select
        the first pen.
    ELSE
      from the pens selected as having the desired width select
      the first pen.
```

OPTION 2 (*-pens* not used, OR *-pens* used with priority set to 2 (color))

Note: if *-pens* not used, the pen table refers to the default table.

```
search the pen table for the pen(s) that are the desired color.
multiple pens can be selected.
IF the desired color can not be found THEN
  IF (the -c16 option was not specified) and
     (the -v16 option was not specified) and
     (the desired color is a GDF color in the range of 9 to 16) THEN
    map colors 9-15 to colors 1-7 respectively and map color 16
    to color 7.
    search the pen table for the pen(s) that are the remapped color.
    multiple pens can be selected.
    IF the remapped color can not be found THEN
      select the first pen in the pen table.
  ELSE
    select the first pen in the pen table.
IF multiple pens were selected THEN
  IF the -nolw option was specified THEN
    search the pens that have the desired color for the pen closest
    to the line width to be drawn. there may be multiple pens equally
    close. the first one will be selected. the pen selected may be
    either equal, greater than, or less than the line width to be
    drawn.
  ELSE
    search the pens that have the desired color for a pen closest
    to the line width to be drawn. there may be multiple pens equally
    close. the first one will be selected. the pen selected will be
    less than or equal to the line width to be drawn if possible.
    IF all pens in the pen table are greater than desired width THE
      select the pen closest to the line width to be drawn.
```

Extended Color Support

When plotting to a CalComp or Versatec plotter using the cpsC1 or cpsV1 programs, the number of colors supported can be increased to 16 from the default 8 colors by specifying either *-c16* as a parameter to the cpsC1 program or *-v16* for the cpsV1 program. These parameters will work in conjunction with *-pens* and the `pentbl.fil`, but *-pens* is not required for their use. If the `pentbl.fil` is used with these parameters, then the number of records in the `pentbl.fil` must be increased to 17. The first 9 records will remain as described above, and records 10 through 17 must be added for pen numbers 9 through 16, following the same format as records 2 through 9. For example:

```
1
1    1    0.3
2    2    0.3
```

```
3    3    0.3
4    4    0.3
5    5    0.3
6    6    0.3
7    7    0.3
8    8    0.3
9    9    0.3
10   10   0.3
11   11   0.3
12   12   0.3
13   13   0.3
14   14   0.3
15   15   0.3
16   16   0.3
```

Refer to the CalComp and Versatec README files for more information on extended color support.

Note: When using the cpsC1 program for CalComp plotters, the `pentbl.fil` must be used even if *-pens* was not passed to the backend. If the `pentbl.fil` does not exist, plotting will take place using a Calcomp default color table with all pens set to black.

Plotting Limitations

There are some differences between what you see on your display and what is plotted:

Reverse Clipping

The graPHIGS API uses underpainting and overpainting to achieve certain results. It does this by first drawing an object and then drawing over it. On a display, the second color simply overlays the first. In addition, the background color is often used to block out portions of the underpainted segment. This graphics feature of blocking out a section of a previously drawn object with another object or graphic element is called *reverse clipping* or *shielding*. When you use a plotter, both objects are drawn.

Patterns

Some of the patterns plotted may not match those used by the graPHIGS API. In addition, user-defined patterns are mapped to the sixteen supplied patterns using MOD 16 results.

Picture Size

Unless *-ispi* or *-es* is specified, plots fill the entire page, keeping the correct aspect ratio. This means that a circle is always a circle, and a square is always a square.

Annotation Text

Plotters only support those characters with ASCII values less then 128.

Plotting Using the ISPI Interface

The following parameters are ignored when *-ispi* is used: *-af, -noin, -rot, -nopl, -es, -esx* and *-slow* when using *-ispi*.

ISPI Limitations

Line types

Only the `SOLID` line type is supported through ISPI.

Annotation Text

Character shear and direction are not supported. Character spacing and size also differ. Geometric text should be used for ISPI.

Refer to the CalComp and Versatec README files in the **/usr/lpd** directory for more information on plotting to CalComp and Versatec plotters.

Problem Determination

If problems are encountered using any of the plotting backends, refer to the log file created for the plot for any messages which may help determine the cause of the problem.

If it becomes necessary to contact IBM support for resolution, be prepared to forward the following data:

- hardcopy plot
- gdf file
- gl file
- log file
- pen table
- operating system level
- graPHIGS level
- plotter model
- plotter setup
  - baud rate, parity
  - pen assignments
- name of plotting backend used
- detailed problem description

Forward all items which are available. If it is possible to produce a correct plot as well, forward all applicable items from this list for the correct plot, along with an explanation of what was done differently.

Plotting on AIX PS/2

Plotting a file on the operating system using AIX PS/2 Personal graPHIGS is essentially the same as plotting on an RS/6000 using the graPHIGS API with the following exceptions:

- **queuing** - the command used to schedule a plot on AIX PS/2 is **print**, not **enq**. For example,

  ```
  print -gdffname.gdf -nolw -es
  ```

- **ISPI Interface** - the ISPI interface is provided on the PS/2, but no CalComp or Versatec ISPI libraries are shipped with AIX PS/2 Personal graPHIGS. Refer to the file **/usr/bin/makecps** for instructions on link editing your own ISPI library.
- **Options supported** - the following limitations apply to parameters which may be passed to AIX PS/2 Personal graPHIGS:

These parameters are not supported:	*-af, -angXX, -chopout, -c16, -esx, -hinXX, -hcmXX, -hsfix, -hsprop, -ldir, -nobord, -nochop, -optf, -VEROUT, -VMSGS, -VPARM, -v16*
-ps=x	This parameter must be specified when using `-ispi`, or when plotting solid area fill on paper larger than a-size.

Plotting on VM/MVS

GDDM/graPHIGS ships a GDF conversion utility which functions in much the same way as the plotting modules for the RS/6000 and AIX PS/2.

To enable the supported options parameters, you may edit the **CONVERT EXEC** or **CONVERT CLIST** provided.

Plotting CGM Files

Executing Plot Modules Directly

To plot a file, the following syntax can be used:

```
/usr/lpd/cgm2hp2 -option  filename.cgm
                 ^       |
                 |------|
```

where:

- *filename*.cgm is the name of the file to be plotted.
- *filename* is derived from the connection identifier. The CGM *filename* must be the last parameter passed to the plot module
- - *option* can be any of the options described below for controlling certain plotting functions.

Note:

- The symbol & may be added after the CGM filename to plot in the background:

```
/usr/lpd/cgm2hp2 -option  filename.cgm &
                 ^       |
                 |------|
```

Examples:

1. To plot the file **file.cgm** to a plotter attached to the `tty0` port:

```
/usr/lpd/cgm2hp2 -f /dev/tty0 file.cgm
```

2. To plot the same file in the background, and to rotate the plot 90 degrees:

```
/usr/lpd/cgm2hp2 -tty0 -rot tmp0001.cgm &
```

3. To convert the file **file.cgm** to GL2 data:

```
/usr/lpd/cgm2hp2 -f plot.gl2 file.cgm
```

Using Plot Modules as Printer Backends

A plot can be scheduled through **smit** or by using the **enq** command.

```
enq -P queuename -o option  filename.cgm
                 ^         |
                 |--------|
```

where:

- *queuename* is the name of the print queue.
- *-option* is any option to be passed to the backend program.
- *filename*.cgm is the name of the CGM file to be plotted.

CGM2HP2 Plotting Options

The following is a list of optional parameters which can be passed to the cgm2hp2 plot module:

-a#	Arc Granularity where # is the maximum error in millimeters allowed between straight line segments used to approximate arcs and corresponding points on the arc. Decreasing the value will yield smoother curves.
-c#	Hatch space in centimeters where # is a float in the range.
-d dirname	Directory for reading and writing files. Equivalent to *-ldir* option of `cpsI1`.
-f filename	Override the default output filename.
-g filename	Map colors to gdf colors (as defined in before applying pen mapping. CGM color table will not be downloaded when this option is used.
-i#	Hatch space in inches where # is a float in the range.

-j#,# Line end or join style. Implements the hpgl2 LA command to control line ends and join styles. Format of argument is:

```
K,V
K,V,K,V
K,V,K,V,K,V
```

-l No line width. This option prohibits the multi-stroking of lines to emulate wide lines on pen plotters. On raster devices, all lines will be drawn with the default pen width.

-m Use penmapping. This option causes `cgm2hp2` to use a pen table file `pentbl.fil`) to indicate the color indices and thickness of pens installed in a pen plotter (or virtual pens in a raster device). For raster devices, if the same color scheme that was imbedded in the original file is desired, this option should not be used.

-n# Indicate the maximum number of virtual pens to use.

-o# Overlap merge control. Controls the combining of colors for overlapping objects when drawn with a raster plotter.

-r Rotate drawings 90 degrees.

-q# Quality level. # is an integer in the range 0-100. 0 Represents draft quality, and 100 equals premium quality. Effects and granularity are device dependent. Generally, this controls pen speed on pen plotters.

-s# Replace the scale value used in the file. # is the length in meters along the X axis. Replacing the scale value results in ALL drawings in a CGM file being rescaled. Line width and annotation text size are not scaled.

-t filename Specify a new name for the `pentbl.fil` or a relative path.

-x filename Output the CGM color table to cross reference file, *filename*.

-M# Media format. Places a media type (MT) instruction into the gl2 file to allow for media dependent plotter setup. Effects on drawing are device dependent. Valid values for # are:

- 0 - paper
- 1 - transparency film
- 2 - vellum
- 3 - polyester film
- 4 - translucent paper
- 5 - special paper
- 6 - glossy paper

-S# Control the type of pen sorting done at the plotter.

- 0 - no sorting
- 1 - Pen sorting
- 2 - Endpoint Swap
- 4 - Geographic sorting
- 8 - optimum sorting (device dependent)

Boolean combination of 1-4 are allowed (i.e. 3=Pen and endpoint sort)

Defaults

Output filename If the input filename ends in .cgm, the .cgm extension is replaced with .gl2. Otherwise, the extension .gl2 is appended to the filename. If the filename is longer than 16 characters, the the first 12 characters of the filename are taken and . filenames are the same, the first character of the output filename is incremented (If it the first character was C, it will be changed to a D).

Number of Pens 255

Penwidth Used

Penmap file Not used

Penmap file name pentbl.fil

Drawing Scale .

Dirname	Set to the directory indicated in the GPPLOTDIR environment variable. If GPPLOTDIR is not set, uses the directory set in the PWD environment variable. If neither of them is set, the current directory is used.
Hatchspace	Variable with drawing scale
Quality Level	Device default
Sort Type	Device default
Overly Merge	Off
join style	1,1,2,1,3,5 - Butted ends, mitered joins, with a limit of 5.
arc granularity	.

Limitations

HPGL2	This code is designed to convert CGM produced by the graPHIGS API to hpgl2. It is assumed that the plotter buffers are large enough to render polygons and annotation text generated by the graPHIGS application. The default font of the output device is used to render annotation text.
CGM	No attempt is made to interpret CGM parameters that are not varied by graPHIGS CGM. As a result, it is unlikely that CGM files from non-graPHIGS applications would be correctly converted by this application. Unsupported CGM orders are skipped and a warning is written to standard error. Cell Array, produced by the use of the graPHIGS pixel primitives (GPPXL2, GPPXL3) is not supported.

Usage Notes

For raster plotters the following options are recommended:

-n#	Where # is the largest CGM color table accessed

For pen plotters the following options are recommended:

-n#	To skip downloading color table beyond the number of pens in the pentable file.
-m	To use penmap file
-g	To map pen colors to 16 gdf pen colors

Printing to devices attached to the parallel ports

In order to print to a device attached to the parallel port, you need to first generate a .gl2 file using the methodology above. (The queueing system protocols do not support directly writing to the parallel port.) The .gl2 file can then be printed using **lpr** or **enq** printer is **not** already supported as a virtual printer, you need to define it as type opp (other parallel printer).

Differences between raster and pen plotter devices.

HPGL2 color raster devices generally support user defined color tables with up to 255 entries. The graPHIGS CGM includes a 255 entry color table. By default, this entire color table is downloaded to the output device, so that the printed colors will match the colors displayed by the graPHIGS application.

Pen plotters are limited to the colors of pens inserted in their carousels. By default **cgm2hp2** will attempt to set the plotter pen to match the entry number of the CGM color table. The default CGM color table follows:

```
0.         0,   0,   0   Black
1.       255, 255, 255   White
2.       255,   0,   0   Red
3.         0, 255,   0   Green
4.         0,   0, 255   Blue
5.       255, 255,   0   Yellow
6.       255,   0, 255   Magenta
7.         0, 255, 255   Cyan
8.       255, 255, 255   White
               screen
               screen
               screen
255.     255, 255, 255   White
```

Note: Black, (background) is color table entry 0. On pen plotters, pen 0 corresponds to draw with no pen, or draw in background. If color table entry 0 is used and it represents a color other than background, you need to map it to another pen using a pen table file. See the description below.

Pen table file

The format is the same as the `pentbl.file` for the cpsX1 utilities described in However, the number of entries may range between 1 and 255. If the number of pens exceeds the number of slots available on the plotter, the plotter will perform modular division to pick the pen number. Therefore, if the CGM color table is larger than the number pens available multiple pens can be mapped to a single slot by repeating as follows:

```
1  1     1.3
2  2     0.3
3  4     0.3
4  3     1.3
5  6     1.3
6  6     0.3
7  7     1.3
8  8     1.3
1  9     1.3
2  2     0.3
3  4     0.3
4  3     1.3
5  6     1.3
6  6     0.3
7  7     1.3
8  10    1.3
```

This pentbl will map colors 1,9 to pen 1 and colors 8 and 10 to pen 8. If pen mapping by color is selected and there is no entry for a color table entry, the pen number defaults to pen 0 (background color).

To map the default CGM colors to the default gdf colors, use the following color table:

```
                CGM Table   GDF Color
                Entry       Number        R    G    B     Name
                ---------  -----------  ---- ---- ----   --------
1  4    0.3        4           1       0,     0  255    Blue
2  2    0.3        2           2       255    0    0    Red
3  6    0.3        6           3       255    0  255    Magenta
4  3    0.3        3           4       0    255    0    Green
5  7    0.3        7           5       0    255  255    Cyan
6  5    0.3        5           6       255  255    0    Yellow
7  7    0.3        0           7       0      0    0    Black
8  8    0.3        1           8       255  255  255    White
```

Additionally, if pens of multiple widths are used, you may want to to select pens based on penwidth. The pentable can also accomplish this. Selecting pens based on penwidth may may reduce or eliminate multi-stroking. If the penwidth exactly matches the width of the line being drawn multi-stroking can be

completely eliminated using the *-l* option. Otherwise, the multi-stroking can be reduced by using the pentable file to pick the closest pen. Be sure to set the plotters front panel to accurately reflect the actual pen width in use.

Appendix D. Printing with graPHIGS

The graPHIGS API provides the capability of converting CGM files produced by the graPHIGS API to PostScript for printing on any PostScript printer.

For more information about converting CGM to Postscript, see the file `/usr/lpd/cgm2ps.readme`.

Appendix E. How the Mnemonics are Generated

Table 144. Tuning Information

Diagnosis, Modification or Tuning Information
Diagnosis, Modification or Tuning Information is provided as additional guidance on the creation of mnemonics used in the product. This information should never be used as programming interface information.

You may want to know how mnemonics are generated for subroutine calls in the graPHIGS API. Where possible with the two-letter sentinel used by all graPHIGS API calls, “GP,” the abbreviations match those used in the GKS language binding. The GKS abbreviation is noted in parentheses when the graPHIGS API abbreviation differs. Words from which no letters are used for forming the mnemonic are categorized as deletions.

(X) indicates deletion (in some cases). **( )** indicates the GKS abbreviation, if different.

Deletions

```
Active
And
At
Available
Between
Capabilities
Class(CL)
Contain
Content
Control
Device
Entries
Factor
From
Function
Identifier
In
Internal
List
Local
Maximum
More
Of
Output
Point
Set
Supported
Surface
Table
To
Vector
Which
With
```

Abbreviations

Table 145. Abbreviations

Action	AC (X)
Activate	AC
Actual	A (X)

Table 145. Abbreviations (continued)

Add	AD
Alarm	AL
Alignment	AL
All	A (X)
Annotation	AN,A
Aperture	A
Application	A
Approximation	A
Arc	A
Aspect	A
Associate	A
Asynchronous	A
Attach	AT
Attribute	A
Await	AW
B-Spline	B
Back	B
Base	B
Break	BK
Broadcast	B
Buffer	B
Calculation	C
Category	C
Character	CH
Character Set	CS,S
Characteristics	CH,C
Choice	CH
Circle	CR
Circular	CR
Class Name	CN
Close	CL
Code	CD
Color	C
Comparison	C
Compose	C
Compute	C
Conditionally	C
Configuration	C
Connect	C
Connection	C
Convert	CV

Table 145. Abbreviations (continued)

Coordinate	CO
Copy	CP
Create	CR
Criteria	C
Cue	C
Cull	C
Culling	C
Current	C
Curve	C
Data	D (X)
Deactivate	DA
Default	D
Defer	DF
Deferral	DF,D
Define	DF (X)
Definition	D
Delete	DL,D (D)
Depth	D
Detach	DT
Direct	D
Disassociate	D
Disconnect	D
Disjoint	D
Display	D
Distinguish	D
Draw	D
Edge	E
Edit	ED
Element	E
Ellipse	EL
Elliptical	EL
Empty	E
End	E
Entries	E (X)
Error	E (X)
Escape	ES (ESC)
Event	EV,E
Execute	EX
Existence	E
Exit	EXIT
Expansion	XP

Table 145. Abbreviations (continued)

Extended	X
Extent	EX,XT
Face	F
Facilities	F (X)
Fill	F
Filter	F
Flag	F
Flush	FL,F
Font	FO,F
Font Directory	FD
Frame	F
Generalized	G
Geometric	G
Get	GT
Global	GL
Grid	G
Handling	HND (X)
Hatch	HA,H
Height	H
Hierarchical	H
Highlight	HL (HLIT)
HLHSR	H
Identifier	ID,I (X)
Image	I
Image Board	IB
Index	I
Initialize	IN,I
Initiate	IN
Input	I
Inquire	Q
Insert	IN
Interior	I
Invisibility	IV
Label	LB
Length	L
Level	L
Light	L
Line	L
Linetype	LT (LN)
Linewidth	LW
Locator	LC

Table 145. Abbreviations (continued)

Logging	LOG
Logical	L
Mapping	MP
Marker	M (MK)
Markertype	MT (MK)
Mask	M
Matrix	MT (M)
Message	MSG,MS
Methods	M
Mode	MO (M)
Model	M,ML
Modeling	ML
Network	NT,N
Non-Uniform	N
Normal	N
Nucleus	NC,N
Number	N (X)
Offset	O
Open	OP
Operations	O
Overflow	O
Pack	P
Parameter	P
Password	PW
Path	PT (P)
Pattern	PA,P
Physical	P
Pick	PK
Pixel	PXL (PX)
Plane	PL (X)
Pointer	P
Polygon	PG,G
Polyhedron	PH
Polyline	PL,L
Polymarker	PM,M
Pool	P
Positioning	P
Precision	PR
Predefined	P
Primary	P
Primitive	P (X)

Table 145. Abbreviations (continued)

Prior	P
Priority	P
Private	P
Processing	P
Properties	P
Quantization	Q
Queue	Q
Range	R
Ratio	R
Record	REC
Rectangle	RCT
Redraw	R
Reference	R (RF)
Remove	R
Rendering	R
Representation	R
Request	RQ,R
Resource	R
Return	R
Root	R
Rotate	ROT
Sample	SM
Scale	SC
Secondary	S
Select	S
Send	S
Shell	SH
Simultaneous	S
Size	S (X)
Sound	SD
Source	S
Spacing	SP
Specular	S
Sphere	SPH
State	S (X)
Storage	S
String	ST
Strip	S
Stroke	SK
Structure	ST,S (X)
Structure Store	SS

Table 145. Abbreviations (continued)

Style	S
Surface	S
Synchronize	SYNC
System	SY (X)
Terminate	TM
Test	T
Text	TX,T
Trace	TRCE
Transformation	XF,X (T)
Translate	TRL
Transparency	T
Triangle	T
Trigger	T
Trimmed	T
Type	T
Up	UP
Update	UP (X)
Utilization	U
Valuator	VL
Value	V
Variability	V
View	V
Workstation	WS,W (WK)
Write	W

Appendix F. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Dept. LRAS/Bldg. 003
11400 Burnet Road
Austin, TX 78758-3498
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:
(c) (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. (c) Copyright IBM Corp. _enter the year or years_. All rights reserved.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

AIX
AIXwindows
GDDM
IBM
RS/6000

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be the trademarks or service marks of others.

Index

A

C

D

U

V

W

X

Readers' Comments — We'd Like to Hear from You

The graPHIGS Programming Interface: Technical Reference

Publication No. SC33-8193-11

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:

- Send your comments to the address on the reverse side of this form.
- Send your comments via e-mail to: aix6koub@austin.ibm.com

If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	E-mail address

Readers' Comments — We'd Like to Hear from You
SC33-8193-11

IBM

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Information Development
Department 04XA-905-6C006
11501 Burnet Road
Austin, TX 78758-3493

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Printed in U.S.A.

SC33-8193-11