DOMAIN 3D Graphics Metafile Resource Call Reference

Order No. 005812
Revision 00
Software Release 9.0

Apollo Computer Inc.
330 Billerica Road
Chelmsford, MA 01824

First Printing: December 1985
Latest Printing:
Updated:

This document was produced using the Interleaf Workstation Publishing Software (WPS). Interleaf and WPS are trademarks of Interleaf, Inc.

Preface

The *DOMAIN 3D Graphics Metafile Resource Call Reference* describes the constants, data types, and user-callable routines used by the DOMAIN 3D Graphics Metafile Resource (GMR™) system for developing three-dimensional graphics applications.

Audience

This manual is for programmers who use the DOMAIN 3D Graphics Metafile Resource to develop application programs. Users of this manual have some knowledge of computer graphics and have experience in using the DOMAIN system.

We suggest that you read the task-oriented handbook *Programming with DOMAIN 3D Graphics Metafile Resource* before using this reference.

Organization of this Manual

This manual contains two chapters:

Chapter 1	Presents the constants and data types used by the 3D Graphics Metafile Resource package.
Chapter 2	Presents a description of each routine including format and parameters. The organization of routines is alphabetical.
Quick Reference	Presents two listings of 3D GMR routines. The first is a listing of routines and descriptions by function. The second is an alphabetical listing of call formats.

Additional Reading

Use this reference as a companion to the *Programming With 3D Graphics Metafile Reference* manual (005807).

The *Programming With DOMAIN 2D Graphics Metafile Resource* manual (005696) describes how to write programs that use the DOMAIN 2D Graphics Metafile Resource.

The *Programmer's Guide to DOMAIN Graphics Primitives* manual (005808) describes how to write graphics programs using DOMAIN Graphics Primitives.

The *Programming With General System Calls* manual (005506) describes how to write programs that use standard DOMAIN systems calls.

The *DOMAIN Language Level Debugger Reference* (001525) describes the high-level language debugger.

For language-specific information, see the *DOMAIN FORTRAN Language Reference* (000530), the *DOMAIN Pascal User's Guide* (000792), and the *DOMAIN C Language Reference* (002093).

3D GMR creates POSTSCRIPT files for hardcopy output to laser printers that support POSTCRIPT. If you want to modify the POSTSCRIPT files, see the *POSTSCRIPT Language Reference* (007765).

Documentation Conventions

Unless otherwise noted in the text, this manual uses the following symbolic conventions.

UPPERCASE	Uppercase words or characters in formats and command descriptions represent commands or keywords that you must use literally.
lowercase	Lowercase words or characters in formats and command descriptions represent values that you must supply.
[ ]	Square brackets enclose optional items in formats and command descriptions. In sample Pascal statements, square brackets assume their Pascal meanings.
{ }	Braces enclose a list from which you must choose an item in formats and command descriptions. In sample Pascal statements, braces assume their Pascal meanings.
CTRL/Z	The notation CTRL/ followed by the name of a key indicates a control character sequence. You should hold down the <CTRL> key while typing the character.
	Vertical ellipses represent additional information in a program fragment that is either too lengthy to include or not relevant to the example.

Problems, Questions, and Suggestions

We appreciate comments from the people who use our system. In order to make it easy for you to communicate with us, we provide the User Change Request (UCR) system for software-related comments, and the Reader's Response form for documentation comments. By using these formal channels, you make it easy for us to respond to your comments.

You can get more information about how to submit a UCR by consulting the *DOMAIN System Command Reference* manual. Refer to the CRUCR (Create User Change Request) Shell command. You can also view the same description on-line by typing:

```
$ HELP CRUCR <RETURN>
```

For your comments on documentation, a Reader's Response form is located at the back of this manual.

Contents

Chapter 1 Constants and Data Types 1-1

Chapter 2 3D GMR Routines 2-1

Quick Reference 1

Chapter 1
Constants and Data Types

This chapter describes the constants and data types used by the 3D Graphics Metafile Resource package (hereafter referred to as 3D GMR). Each data type description includes an atomic data type translation (i.e., GMR_$ACC_CREATE_T = 2-byte integer) as well as a brief description of the type's purpose. The description includes any predefined values associated with the type. The following is an example of a data type description for the GMR_$CONC_MODE_T type:

GMR_$CONC_MODE_T

A 2-byte integer. Defines the number of concurrent users a file may have. One of the following predefined values:

GMR_$1W
Allows n readers or one writer.

GMR_$COWRITERS
Allows more than one writer, but all must be on the same node.

This chapter also illustrates the record data types in detail. These illustrations will help FORTRAN programmers construct record-like structures, as well as provide useful information for all programmers. Each record type illustration:

- Shows FORTRAN programmers the structure of the record that they must construct using standard FORTRAN data-type statements. The illustrations show the size and type of each field.
- Describes the fields that make up the record.
- Lists the byte offsets for each field. Use these offsets to access individual fields. Bytes are numbered from left to right and bits are numbered from right to left.
- Indicates whether any fields of the record are, in turn, predefined records.

Minimum and maximum limits

```
gmr_$max_file                    =  16
gmr_$max_structure_name_length   =  12
gmr_$max_tag_length              =  16#7FFF
gmr_$min_color_id                =  0
gmr_$max_color_id                =  255
gmr_$max_string_length           =  120
gmr_$max_structure_id            =  16#7FFF
gmr_$max_array_len               =  1024
gmr_$max_pick_depth              =  32
gmr_$max_instance_depth          =  32
gmr_$max_element_count           =  1024
gmr_$max_line_type               =  4
gmr_$max_name_element            =  255
```

For convenience in establishing text path.

```
gmr_$text_path_right  =   0.0         ( radians )
gmr_$text_path_up     =   1.5707963   ( radians )
gmr_$text_path_left   =   3.1415927   ( radians )
gmr_$text_path_down   =   4.7123890   ( radians )
```

Attribute default values

```
gmr_$line_color_def              =   1
gmr_$line_inten_def              =   1.0
gmr_$mark_color_def              =   1
gmr_$mark_inten_def              =   1.0
gmr_$fill_color_def              =   1
gmr_$fill_inten_def              =   1.0
gmr_$text_color_def              =   1
gmr_$text_inten_def              =   1.0
gmr_$text_height_def             =   0.01
gmr_$text_slant_def              =   0.0
gmr_$text_up_x_def               =   1.0
gmr_$text_up_y_def               =   0.0
gmr_$text_expansion_def          =   1.0
gmr_$text_spacing_def            =   0.0
gmr_$text_path_def               =   0.0
gmr_$mark_scale_def              =   1.0
gmr_$mark_type_def               =   1
gmr_$line_type_def               =   1
```

Highlight default values

```
gmr_$line_color_hl_def           =   3
gmr_$line_inten_hl_def           =   1.0
gmr_$mark_color_hl_def           =   3
gmr_$mark_inten_hl_def           =   1.0
gmr_$fill_color_hl_def           =   3
gmr_$fill_inten_hl_def           =   1.0
gmr_$text_color_hl_def           =   3
```

```
gmr_$text_inten_hl_def              =    1.0
gmr_$text_height_hl_def             =    0.01
gmr_$text_slant_hl_def              =    0.0
gmr_$text_up_x_hl_def               =    1.0
gmr_$text_up_y_hl_def               =    0.0
gmr_$text_expansion_hl_def          =    1.0
gmr_$text_spacing_hl_def            =    0.0
gmr_$text_path_hl_def               =    0.0
gmr_$mark_scale_hl_def              =    1.0
gmr_$mark_type_hl_def               =    1
gmr_$line_type_hl_def               =    1
```

Defaults for Visibility and Picking

```
gmr_$structure_value_def            =    255
gmr_$structure_mask_def             =    16#FFFFFFFF

gmr_$viewport_vis_low_range_def     =    0
gmr_$viewport_vis_hgh_range_def     =    16#7FFFFFFF
gmr_$viewport_vis_mask_def          =    16#FFFFFFFF

gmr_$viewport_pick_low_range_def    =    0
gmr_$viewport_pick_hgh_range_def    =    16#7FFFFFFF
gmr_$viewport_pick_mask_def         =    16#FFFFFFFF

gmr_$pick_aperture_xsize_def        =    0.10
gmr_$pick_aperture_ysize_def        =    0.10
gmr_$pick_aperture_zsize_def        =    2.00
gmr_$pick_aperture_xcenter_def      =    0.00
gmr_$pick_aperture_ycenter_def      =    0.00
gmr_$pick_aperture_zcenter_def      =    0.00
```

Default ablocks

```
gmr_$default_ablock                 =    1
gmr_$nochange_ablock                =    0
```

Predefined line patterns

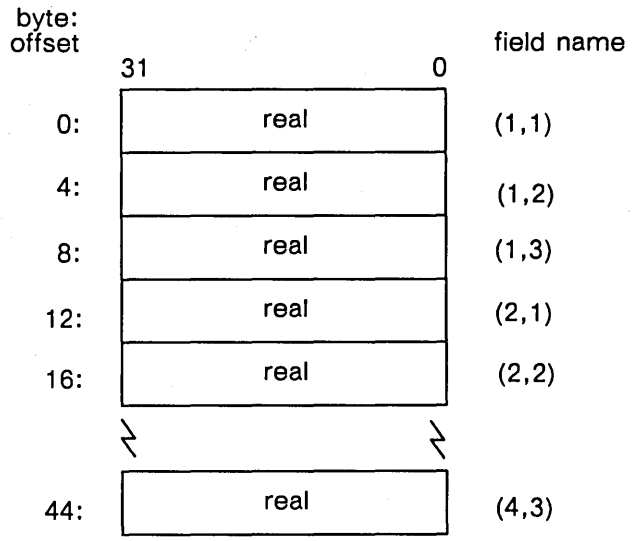
```
gmr_$line_solid                     =    1
gmr_$line_dashed                    =    2
gmr_$line_dotted                    =    3
gmr_$line_dashed_dotted             =    4
```

Miscellaneous constants

```
gmr_$default_viewport               =    1
gmr_$bg_color_id_def                =    0
gmr_$bg_inten_def                   =    0
gmr_$nil_element_index              =   -1
gmr_$viewport_state_block_size      =    128   ( long words )
```

GMR_$4X3_MATRIX_T

A two-dimensional array of 4-byte real values. Four rows and three columns stored in row-major form. The following diagram illustrates this data type:

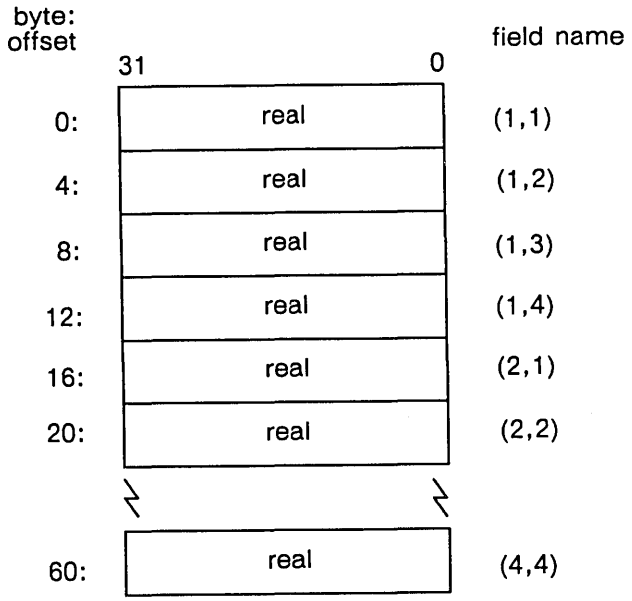


For FORTRAN users:

GMR_$4X3_MATRIX_T expects data to be stored in row-major form. Both C and Pascal store two-dimensional arrays this way. FORTRAN stores data in column-major form, so an array (n,m) in FORTRAN has m as the major dimension and n as the minor. Both C and Pascal use n as the major and m as the minor. Refer to the examples in the Usage section under GMR_$4X3_MATRIX_CONCATENATE.

GMR_$4X4_MATRIX_T

A two-dimensional array of 4-byte real values. Four rows and four columns stored in row-major form (FORTRAN programmers, see note for GMR_$4X3_MATRIX_T above). The following diagram illustrates this data type:



GMR_$ABLOCK_ID_T

A 2-byte integer. Specifies the ablock identification number.

GMR_$ACC_CREATE_T

A 2-byte integer. Specifies the access mode. One of the following predefined values:

GMR_$WRITE
Returns an error if the file already exists.

GMR_$OVERWRITE
Overwrites the previous version if the file already exists.

GMR_$UPDATE
Opens the previous version if the file already exists.

GMR_$ACC_OPEN_T

A 2-byte integer. Specifies the read/write accessibility. One of the following predefined values:

GMR_$WR
Specifies read or write access.

GMR_$R
Specifies read only access.

GMR_$CWR
Specifies read or write access; if the file does not exist, creates it.

GMR_$ACLASS_ID_T

A 2-byte integer. Specifies the aclass identification number.

GMR_$ATTRIBUTE_T

A 2-byte integer. Specifies the attribute type with an attribute source flag set by GMR_$ATTRIBUTE_SOURCE. One of the following predefined values:

GMR_$ATTR_LINE_COLOR
Line color for line primitives.

GMR_$ATTR_LINE_INTEN
Line intensity for line primitives.

GMR_$ATTR_LINE_TYPE
Line type for line primitives.

GMR_$ATTR_FILL_COLOR
Fill color for filled primitives.

GMR_$ATTR_FILL_INTEN
Fill intensity for filled primitives.

GMR_$ATTR_MARK_COLOR
Color for polymarker elements.

GMR_$ATTR_MARK_INTEN
Intensity for polymarker elements.

GMR_$ATTR_MARK_SCALE
Scale for polymarker elements.

GMR_$ATTR_MARK_TYPE
Type for polymarker elements.

GMR_$ATTR_TEXT_COLOR
Text color.

GMR_$ATTR_TEXT_INTEN
Text intensity.

GMR_$ATTR_TEXT_HEIGHT
Text height.

GMR_$ATTR_TEXT_EXPANSION
Text expansion factor.

GMR_$ATTR_TEXT_SLANT
Text slant factor.

GMR_$ATTR_TEXT_SPACING
Text spacing.

GMR_$ATTR_TEXT_UP
Text up vector.

GMR_$ATTR_TEXT_PATH
Text path angle.

GMR_$ATTRIBUTE_SOURCE_T

A 2-byte integer. Specifies the attribute source flag as either direct or aclass. One of the following predefined values:

GMR_$ATTRIBUTE_DIRECT
Use the last direct attribute element routine.

GMR_$ATTRIBUTE_ACLASS
Use the last specified attribute block.

GMR_$AXIS_T

A 2-byte integer. Identifies x-, y-, or z-axis. One of the following predefined values:

GMR_$X_AXIS
Identifies the x-axis.

GMR_$Y_AXIS
Identifies the y-axis.

GMR_$Z_AXIS
Identifies the z-axis.

GMR_$BG_COLOR_T

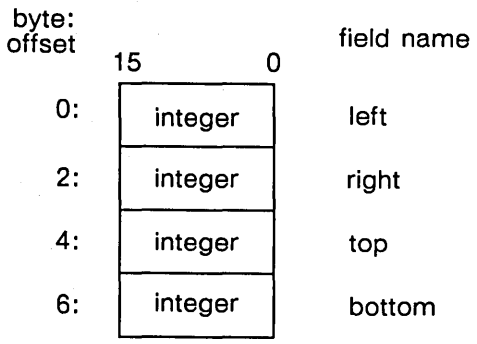
A 2-byte integer. Specifies the background color. One of the following predefined values:

GMR_$COLOR_AND_INTEN
Specifes the color and intensity values supplied by the call parameters color_id and intensity.

GMR_$DM_WINDOW_BACKGROUND
Specifes the current Display Manager window background color.

GMR_$BORDER_WIDTH_T

A four-element array of 2-byte integers. Specifies the viewport border width in pixels. The following diagram illustrates this data type:



Field Description:

left
The width of the left border of the viewport, in pixels.

right
The width of the right border of the viewport, in pixels.

top
The width of the top border of the viewport, in pixels.

bottom
The width of the bottom border of the viewport, in pixels.

GMR_$BUFFER_MODE_T

A 2-byte integer. Identifies single- or double-buffer mode. One of the following predefined values:

GMR_$SINGLE_BUFFER
Single-buffer mode.

GMR_$DOUBLE_BUFFER
Double-buffer mode.

GMR_$BUFFER_T

A 2-byte integer. In double-buffer mode, identifies first or second buffer. One of the following predefined values:

GMR_$1ST_BUFFER
Identifies first buffer.

GMR_$2ND_BUFFER
Identifies second buffer.

GMR_$CHANGE_STATE_T

A 2-byte integer. Specifies the change state in an attribute block. One of the following predefined values:

GMR_$SET_VALUE_AND_ENABLE
Stores the attribute value and sets the value as enabled.

GMR_$SET_VALUE_AND_DISABLE
Stores the attribute value but sets the no-change attribute, thus disabling the use of this attribute.

GMR_$NO_VALUE_AND_ENABLE
Ignores the attribute value but enables what was previously set as the attribute value.

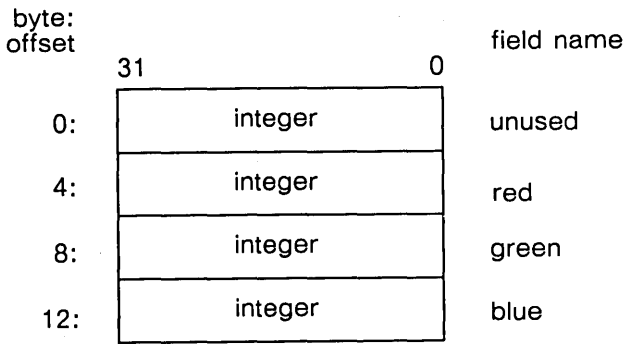
GMR_$NO_VALUE_AND_DISABLE
Ignores the attribute value and disables the attribute's use. With this last state, the previous attribute value is preserved.

GMR_$COLOR_ID_T

Specifies the color identification number as a 2-byte integer in the range [0, GMR_$MAX_COLOR_ID], inclusive.

GMR_$COLOR_VECTOR_T

An array of 4-byte integers, of up to 256 elements. Specifies a list of color values. The following diagram illustrates this data type:



Field Description:

unused
The first two bytes are unused.

red
The amount of red color in the range [0, 255]

inclusive, where 255 is the full scale intensity of red.

green
The amount of green color in the range [0, 255] inclusive, where 255 is the full scale intensity of green.

blue
The amount of blue color in the range [0, 255] inclusive, where 255 is the full scale intensity of blue.

The color vector can be constructed as follows:

```
color_vector[i] := blue + 256(green + (256*red));
```

GMR_$CONC_MODE_T

A 2-byte integer. Defines the number of concurrent users a file may have. One of the following predefined values:

GMR_$1W
Allows n readers or one writer.

GMR_$COWRITERS
Allows more than one writer, but all must be on the same node.

GMR_$COORD_SYSTEM_T

A 2-byte integer. Specifies whether a right- or left-handed coordinate system is used for a particular viewport. One of the following predefined values:

GMR_$COORD_RIGHT
Use a right-handed coordinate system. This is the default.

GMR_$COORD_LEFT
Use a left-handed coordinate system.

GMR_$CURSOR_PATTERN_T

A 16-element array of 2-byte integers. Specifies the values that set the cursor pattern.

GMR_$CURSOR_STYLE_T

A 2-byte integer. Specifies the type of cursor. Uses the predefined value GMR_$BITMAP.

GMR_$DISPLAY_MODE_T

A 2-byte integer. Specifies the mode of operation. One of the following predefined values:

GMR_$BORROW
Uses the entire screen.

GMR_$MAIN_BITMAP
Displays within a bitmap allocated in main memory.

GMR_$DIRECT
Displays within a Display Manager window.

GMR_$NO_BITMAP
Allows editing of files without display.

GMR_$DYNAMIC_DRAW_METHOD_T

A 2-byte integer. Specifies the type of redraw used when dynamic mode is enabled. One of the following predefined values:

GMR_$DYN_METHOD_REDRAW
Each subsequent redraw operation erases the temporary element to the background color and then draws the element in the new position.

GMR_$DYN_METHOD_XOR
Specifies an XOR raster operation. This method preserves the background, but the redrawn elements may have pixels turned off when overlapped with other geometry

GMR_$ELEMENT_ATTRIB_TYPE_T

A 2-byte integer. Identifies the classification of an attribute element. One of the following predefined values:

GMR_$LINE_ATTRIBUTE_ELEMENT
Classifies the element as a line attribute element. The element has an associated attribute type that is described in GMR_$ELEMENT_LINE_ATTRIB_TYPE_T.

GMR_$FILL_ATTRIBUTE_ELEMENT
Classifies the element as a fill attribute element. The element has an associated attribute type that is described in GMR_$ELEMENT_FILL_ATTRIB_TYPE_T.

GMR_$TEXT_ATTRIBUTE_ELEMENT
Classifies the element as a text attribute element. The element has an associated attribute type that is described in GMR_$ELEMENT_TEXT_ATTRIB_TYPE_T.

GMR_$MARK_ATTRIBUTE_ELEMENT
Classifies the element as a mark attribute element. The element has an associated attribute type that is described in GMR_$ELEMENT_MARK_ATTRIB_TYPE_T.

GMR_$NAME_SET_ATTRIBUTE_ELEMENT
Classifies the element as a name set attribute element. The element has an associated attribute type that is described in GMR_$ELEMENT_NAME_ATTRIB_TYPE_T.

GMR_$ELEMENT_FILL_ATTRIB_TYPE_T

A 2-byte integer. Identifies the fill attribute type. One of the following predefined values:

GMR_$FILL_RESERVED_ELEMENT
Identifies a reserved element.

GMR_$FILL_INTEN_ELEMENT
Identifies a fill intensity element.

GMR_$FILL_COLOR_ELEMENT
Identifies a fill color element.

GMR_$ELEMENT_LINE_ATTRIB_TYPE_T

A 2-byte integer. Identifies the line element type. One of the following predefined values:

GMR_$LINE_RESERVED_ELEMENT
Identifies a reserved element.

GMR_$LINE_INTEN_ELEMENT
Identifies a line intensity element.

GMR_$LINE_COLOR_ELEMENT
Identifies the color ID for polylines and multilines.

GMR_$LINE_TYPE_ELEMENT
Identifies the line type ID for polylines and multilines.

GMR_$ELEMENT_INDEX_T

A 4-byte integer. The position of an element within a structure.

GMR_$ELEMENT_MARK_ATTRIB_TYPE_T

A 2-byte integer. Identifies the polymarker attribute type. One of the following predefined values:

GMR_$MARK_RESERVED_ELEMENT
Identifies a reserved element.

GMR_$MARK_INTEN_ELEMENT
Identifies a polymarker intensity element.

GMR_$MARK_COLOR_ELEMENT
Identifies a polymarker color element.

GMR_$MARK_SCALE_ELEMENT
Identifies a polymarker scale element.

GMR_$MARK_TYPE_ELEMENT
Identifies a polymarker type element.

GMR_$ELEMENT_NAME_ATTRIB_TYPE_T

A 2-byte integer. Identifies the name set attribute type. One of the following predefined values:

GMR_$NAME_SET_RESERVED_ELEMENT
Identifies a reserved element.

GMR_$ADD_NAME_SET_ELEMENT
Identifies an add name set element.

GMR_$REMOVE_NAME_SET_ELEMENT
Identifies a remove name set element.

GMR_$ELEMENT_TEXT_ATTRIB_TYPE_T — A 2-byte integer. Identifies the text attribute type. One of the following predefined values:

GMR_$TEXT_RESERVED_ELEMENT
Identifies a reserved element.

GMR_$TEXT_INTEN_ELEMENT
Identifies the text intensity value.

GMR_$TEXT_COLOR_ELEMENT
Identifies the color for text elements.

GMR_$TEXT_UP_ELEMENT
Identifies the "up" vector for text.

GMR_$TEXT_PATH_ELEMENT
Identifies the path angle for text.

GMR_$TEXT_SLANT_ELEMENT
Identifies the slant angle for text.

GMR_$TEXT_HEIGHT_ELEMENT
Identifies the text height.

GMR_$TEXT_SPACING_ELEMENT
Identifies the text spacing.

GMR_$TEXT_EXPANSION_ELEMENT
Identifies the text expansion factor.

GMR_$ELEMENT_TYPE_T — A 2-byte integer. Specifies the element type. One of the following predefined values:

GMR_$RESERVED_ELEMENT
Specifies a reserved element.

GMR_$F3_OPEN_POLY_ELEMENT
Specifies an open polygon element.

GMR_$F3_CLOSED_POLY_ELEMENT
Specifies a closed polygon element.

GMR_$F3_POLYGON_ELEMENT
Specifies a polygon element.

GMR_$F3_MESH_ELEMENT
Specifies a mesh element.

GMR_$INSTANCE_ELEMENT
Specifies an instance element.

GMR_$ACLASS_ELEMENT
Specifies an aclass element.

GMR_$ATTRIBUTE_SOURCE_ELEMENT
Specifies an attribute source flag element.

GMR_$TEXT_ELEMENT
Specifies a text element.

GMR_$POLYMARKER_ELEMENT
Specifies a polymarker element.

GMR_$TAG_ELEMENT
Specifies a tag element.

GMR_$LINE_ATTRIBUTE_ELEMENT
Classifies the element as a line attribute element. The element has an associated attribute type that is described in GMR_$ELEMENT_LINE_ATTRIB_TYPE_T.

GMR_$FILL_ATTRIBUTE_ELEMENT
Classifies the element as a fill attribute element. The element has an associated attribute type that is described in GMR_$ELEMENT_FILL_ATTRIB_TYPE_T.

GMR_$TEXT_ATTRIBUTE_ELEMENT
Classifies the element as a text attribute element. The element has an associated attribute type that is described in GMR_$ELEMENT_TEXT_ATTRIB_TYPE_T.

GMR_$MARK_ATTRIBUTE_ELEMENT
Classifies the element as a polymarker attribute element. The element has an associated attribute type that is described in GMR_$ELEMENT_MARK_ATTRIB_TYPE_T.

GMR_$F3_MULTILINE_ELEMENT
Classifies the element as a multiline element. The element has an associated attribute type that is described in GMR_$LINE_ATTRIB_TYPE_T.

GMR_$NAME_SET_ATTRIBUTE_ELEMENT
Classifies the element as a name set attribute element. The element has an associated attribute type that is described in GMR_$NAME_ATTRIB_TYPE_T.

GMR_$EVENT_T

A 2-byte integer. Specifies the type of input event. One of the following predefined values:

GMR_$KEYSTROKE
Identifies a keyboard character.

GMR_$BUTTONS
Is returned when you press a button on the mouse or bitpad puck.

GMR_$LOCATOR
Is returned when you move the mouse or bitpad puck or use the touchpad.

GMR_$ENTERED_WINDOW
Is returned when the cursor enters a Display Manager window in which the 3D GMR package is running. Direct mode only.

GMR_$LEFT_WINDOW
Is returned when the cursor leaves a Display Manager window in which the 3D GMR package is running. Direct mode only.

GMR_$LOCATOR_STOP
Is returned when you stop moving the mouse or bitpad puck, or stop using the touchpad.

GMR_$NO_EVENT
Indicates that no event has occurred.

GMR_$F_T

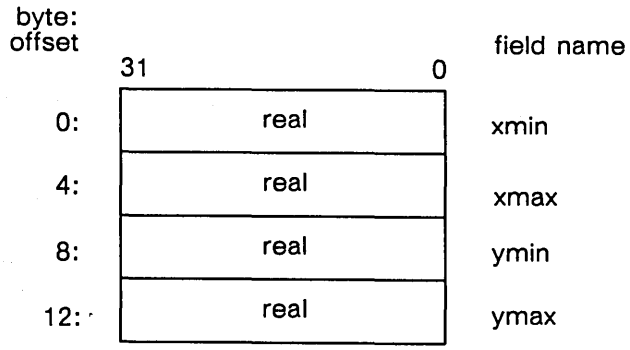
A 4-byte real number.

GMR_$F_ARRAY_T

An array of 4-byte real values with GMR_$MAX_ARRAY_LEN elements.

GMR_$F2_LIMITS_T

Defines the bounds of a rectangular (2D) area. The following diagram illustrates this data type:



Field Description:

xmin
The x-coordinate of the bottom-left corner of the rectangle.

xmax
The x-coordinate of the top-right corner of the rectangle.

ymin
The y-coordinate of the bottom-left corner of the rectangle.

ymax
The y-coordinate of the top-right corner of the rectangle.

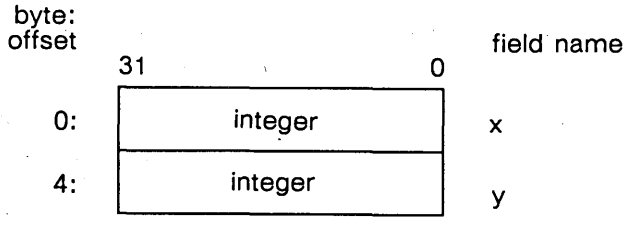
When used to establish the window on the viewing plane (GMR_$VIEW_SET_WINDOW), xmin/max are equivalent to umin/max and ymin/max are equivalent to vmin/max.

GMR_$F_PTR_T

A 4-byte pointer to a GMR_$F_T data type.

GMR_$F2_POINT_T

Specifies the x- and y-coordinates of a 2D point as a pair of 4-byte real values. The following diagram illustrates this data type:



Field Description:

x
The x-coordinate of the point.

y
The y-coordinate of the point.

GMR_$F2_POINT_ARRAY_T

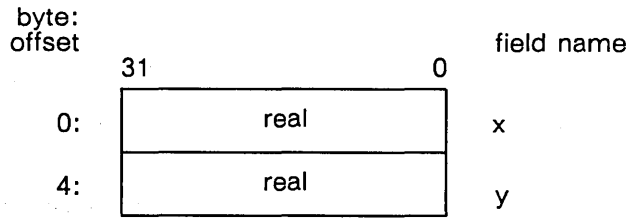
An array of GMR_$F2_POINT_T with GMR_$MAX_ARRAY_LEN elements (one x,y pair per element). The diagram for GMR_$F2_POINT_T illustrates a single element.

GMR_$F2_POINT_PTR_T

A 4-byte pointer to a GMR_$F2_POINT_T data type.

GMR_$F2_VECTOR_T

Specifies the x- and y-coordinates of a 2D vector as a pair of 4-byte real values. Has the same structure as GMR_$F2_POINT_T. The following diagram illustrates this data type:



Field Description:

x
The x-coordinate of the vector.

y
The y-coordinate of the vector.

GMR_$F2_VECTOR_ARRAY_T

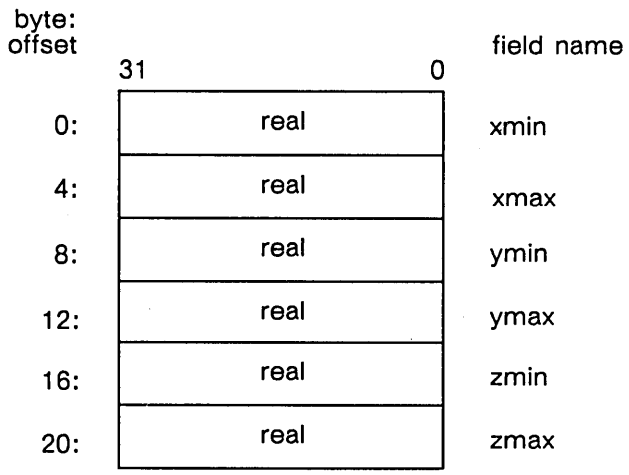
An array of GMR_$F2_VECTOR_T with GMR_$MAX_ARRAY_LEN elements (one x,y pair per element). The diagram for GMR_$F2_VECTOR_T illustrates a single element.

GMR_$F2_VECTOR_PTR_T

A 4-byte pointer to a GMR_$F2_VECTOR_T data type.

GMR_$F3_LIMITS_T

Defines the bounds of a 3D box (a rectangular parallelepiped, i.e., a rectangular prism whose bases are parallelograms). The following diagram illustrates this data type:



Field Description:

xmin
The smallest x-coordinate of the box corners.

xmax
The largest x-coordinate of the box corners.

ymin
The smallest y-coordinate of the box corners.

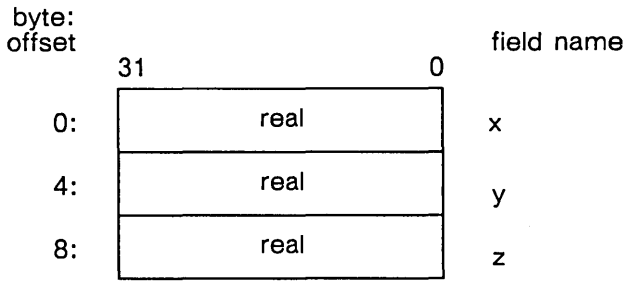
ymax
The largest y-coordinate of the box corners.

zmin
The smallest z-coordinate of the box corners.

zmax
The largest z-coordinate of the box corners.

GMR_$F3_POINT_T

Specifies the x-, y-, and z-coordinates of a 3D point as three 4-byte real values. The following diagram illustrates this data type:



Field Description:

x
The x-coordinate of the point.

y
The y-coordinate of the point.

z
The z-coordinate of the point.

GMR_$F3_POINT_ARRAY_T

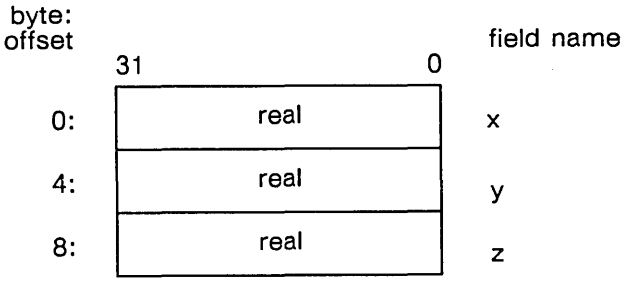
An array of GMR_$F3_POINT_T with GMR_$MAX_ARRAY_LEN elements (one x,y,z triplet per element). The diagram for GMR_$F3_POINT_T illustrates a single element.

GMR_$F3_POINT_PTR_T

A 4-byte pointer to a GMR_$F3_POINT_T data type.

GMR_$F3_VECTOR_T

Specifies the x-, y-, and z-coordinates of a 3D vector as three 4-byte real values. Has the same structure as GMR_$F3_POINT_T. The following diagram illustrates this data type:



Field Description:

x
The x-coordinate of the vector.

y
The y-coordinate of the vector.

z
The z-coordinate of the vector.

GMR_$F3_VECTOR_ARRAY_T

An array of GMR_$F3_VECTOR_T with GMR_$MAX_ARRAY_LEN elements (one x,y,z triplet per element). The diagram for GMR_$F3_VECTOR_T illustrates a single element.

GMR_$F3_VECTOR_PTR_T

A 4-byte pointer to a GMR_$F3_VECTOR_T data type.

GMR_$FILE_ID_T

Specifies the file identification as a 2-byte integer.

GMR_$HILIGHT_METHOD_T

A 2-byte integer. The highlight method of the specified viewport. One of the following predefined values:

GMR_$ELEMENT_HL_ABLOCK
Uses the highlight method defined by the highlighting attribute block assigned to the viewport.

GMR_$ELEMENT_HL_BBOX
Draws a bounding box around the structure. If an element is specified, draws a bounding box around the structure containing the element.

GMR_$HSV_COLOR_T

A three-element array of real values. Specifies color values in this order: hue, saturation, value.

GMR_$I_T

A 2-byte integer.

GMR_$I_ARRAY_T

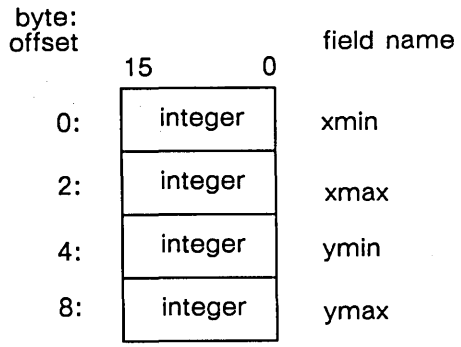
An array of 2-byte integers with GMR_$MAX_ARRAY_LEN elements.

GMR_$I_PTR_T

A 4-byte pointer to a GMR_$I_T data type.

GMR_$I2_LIMITS_T

Defines the bounds of a rectangular (2D) area as a list of 2-byte integers. The following diagram illustrates this data type:



Field Description:

xmin
The x-coordinate of the bottom-left corner of the rectangle.

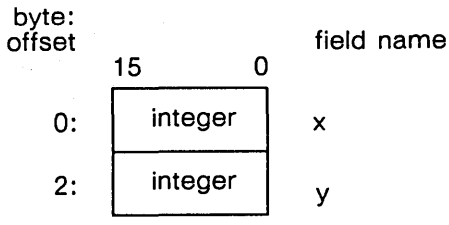
xmax
The x-coordinate of the top-right corner of the rectangle.

ymin
The y-coordinate of the bottom-left corner of the rectangle.

ymax
The y-coordinate of the top-right corner of the rectangle.

GMR_$I2_POINT_T

Specifies the x- and y-coordinates of a 2D point as a pair of 2-byte integers. The following diagram illustrates this data type:



Field Description:

x
The x-coordinate of the point.

y
The y-coordinate of the point.

GMR_$I2_POINT_ARRAY_T

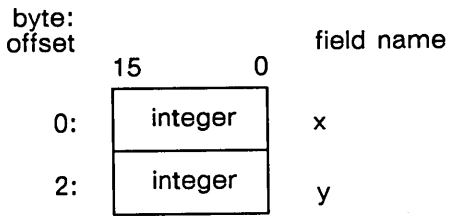
An array of GMR_$I2_POINT_T with GMR_$MAX_ARRAY_LEN elements (one x,y pair per element). The diagram for GMR_$I2_POINT_T illustrates a single element.

GMR_$I2_POINT_PTR_T

A 4-byte pointer to a GMR_$I2_POINT_T data type.

GMR_$I2_VECTOR_T

Specifies the x- and y-coordinates of a 2D vector as a pair of 2-byte integers. Has the same structure as GMR_$I2_POINT_T. The following diagram illustrates this data type:



Field Description:

x
The x-coordinate of the vector.

y
The y-coordinate of the vector.

GMR_$I2_VECTOR_ARRAY_T

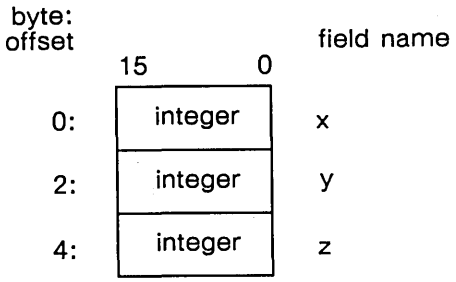
An array of GMR_$I2_VECTOR_T with GMR_$MAX_ARRAY_LEN elements (one x,y pair per element). The diagram for GMR_$I2_VECTOR_T illustrates a single element.

GMR_$I2_VECTOR_PTR_T

A 4-byte pointer to a GMR_$I2_VECTOR_T data type.

GMR_$I3_POINT_T

Specifies the x-, y-, and z-coordinates of a 3D point as three 2-byte integers. The following diagram illustrates this data type:



Field Description:

x
The x-coordinate of the point.

y
The y-coordinate of the point.

z
The z-coordinate of the point.

GMR_$I3_POINT_ARRAY_T

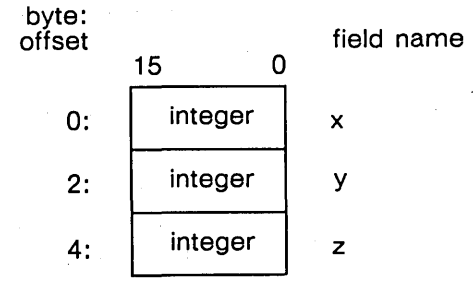
An array of GMR_$I3_POINT_T with GMR_$MAX_ARRAY_LEN elements (one x,y,z triplet per element). The diagram for GMR_$I3_POINT_T illustrates a single element.

GMR_$I3_POINT_PTR_T

A 4-byte pointer to a GMR_$I3_POINT_T data type.

GMR_$I3_VECTOR_T

Specifies the x-, y-, and z-coordinates of a 3D vector as three 2-byte integers. Has the same structure as GMR_$I3_POINT_T. The following diagram illustrates this data type:



Field Description:

x
The x-coordinate of the vector.

y
The y-coordinate of the vector.

z
The z-coordinate of the vector.

GMR_$I3_VECTOR_ARRAY_T

An array of GMR_$I3_VECTOR_T with GMR_$MAX_ARRAY_LEN elements (one x,y,z triplet per element). The diagram for GMR_$I2_VECTOR_T illustrates a single element.

GMR_$I3_VECTOR_PTR_T

A 4-byte pointer to a GMR_$I3_VECTOR_T data type.

GMR_$INQ_TYPE_T

A 2-byte integer. Indicates whether values that are set (specified) or realized are to be returned. One of the following predefined values:

GMR_$SET
Returns set values.

GMR_$REALIZED
Returns realized values.

GMR_$INSTANCE_ECHO_METHOD_T

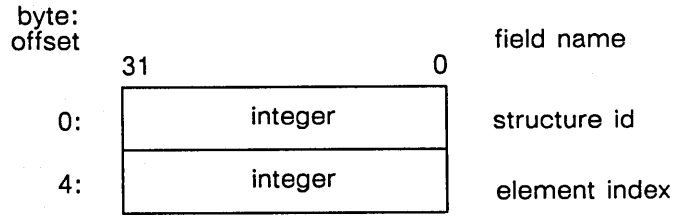
A 2-byte integer. Specifies the instance echo method for a viewport as either ablock or bounding box. One of the following predefined values:

GMR_$ELEMENT_HL_ABLOCK
Use the echo method described in the highlight attribute block associated with the viewport.

GMR_$ELEMENT_HL_BBOX
Draw a bounding box around the structure containing the selected element or subtree. This is the default method.

GMR_$INSTANCE_LEVEL_T

Two 4-byte integers that together identify an element within a structure hierarchy. The following diagram illustrates this data type:



Field Description:

structure id
The structure ID of the element in this particular level of the path.

element index
The position of the element within the structure. The index of the first element is 1.

GMR_$INSTANCE_PATH_ORDER_T

A 2-byte integer. Indicates how an instance path is interpreted or returned. One of the following predefined values:

GMR_$TOP_FIRST
Interpret the path top-first (chosen element last).

GMR_$BOTTOM_FIRST
Interpret the path bottom-first (chosen element first).

GMR_$INSTANCE_PATH_T

An array of GMR_$INSTANCE_LEVEL_T in the range [1, GMR_$MAX_INSTANCE_DEPTH], inclusive. See GMR_$INSTANCE_LEVEL_T for an illustration of one element.

GMR_$INSTANCE_PATHLENGTH_T

A 2-byte integer. Specifies the length of an instance path.

GMR_$INTEN_T

A 4-byte real value between 0 and 1. Specifies the intensity, which (along with color_id) is used to determine line, fill, and background color.

GMR_$KEYSET_T

A 16-element array of 2-byte integers. Specifies the set of characters that make up a keyset associated with the graphics input event types GMR_$KEYSTROKE and GMR_$BUTTONS. For a FORTRAN subroutine to use in building a set of characters, see the routine GMR_$INPUT_ENABLE in this volume.

GMR_$L_T

A 4-byte integer.

GMR_$L_ARRAY_T

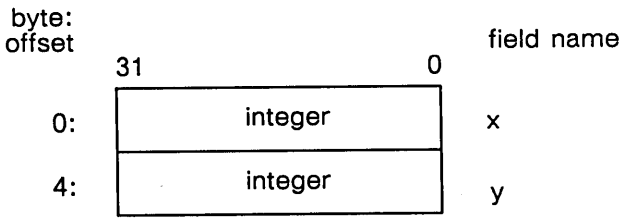
An array of 4-byte integers with GMR_$MAX_ARRAY_LEN elements.

GMR_$L_PTR_T

A 4-byte pointer to a GMR_$L_T data type.

GMR_$L2_POINT_T

Specifies the x- and y-coordinates of a 2D point as a pair of 4-byte integers. The following diagram illustrates this data type:



Field Description:

x
The x-coordinate of the point.

y
The y-coordinate of the point.

GMR_$L2_POINT_ARRAY_T

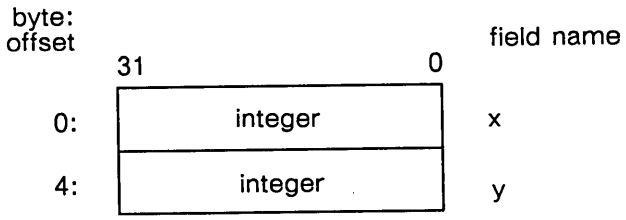
An array of GMR_$L2_POINT_T with GMR_$MAX_ARRAY_LEN elements (one x,y pair per element). The diagram for GMR_$L2_POINT_T illustrates a single element.

GMR_$L2_POINT_PTR_T

A 4-byte pointer to a GMR_$L2_POINT_T data type.

GMR_$L2_VECTOR_T

Specifies the x- and y-coordinates of a 2D vector as a pair of 4-byte integers. Has the same structure as GMR_$L2_POINT_T. The following diagram illustrates this data type:



Field Description:

x
The x-coordinate of the vector.

y
The y-coordinate of the vector.

GMR_$L2_VECTOR_ARRAY_T

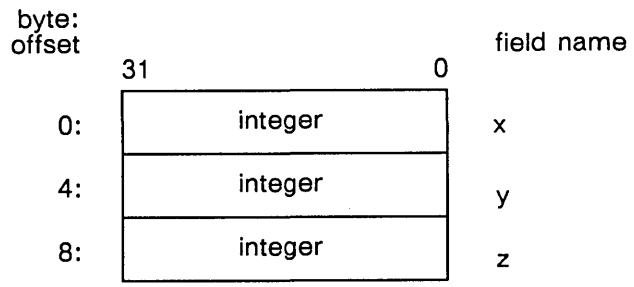
An array of GMR_$L2_VECTOR_T with GMR_$MAX_ARRAY_LEN elements (one x,y pair per element). The diagram for GMR_$L2_VECTOR_T illustrates a single element.

GMR_$L2_VECTOR_PTR_T

A 4-byte pointer to a GMR_$L2_VECTOR_T data type.

GMR_$L3_POINT_T

Specifies the x-, y-, and z-coordinates of a 3D point as three 4-byte integers. The following diagram illustrates this data type:



Field Description:

x
The x-coordinate of the point.

y
The y-coordinate of the point.

z
The z-coordinate of the point.

GMR_$L3_POINT_ARRAY_T

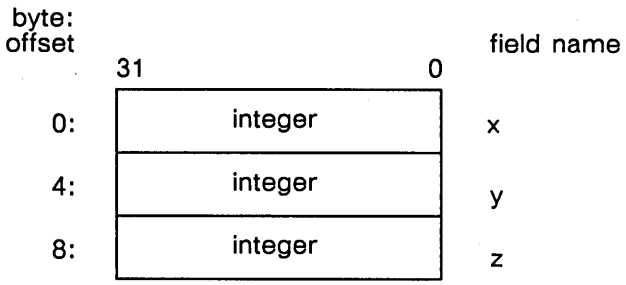
An array of GMR_$L3_POINT_T with GMR_$MAX_ARRAY_LEN elements (one x,y,z triplet per element). The diagram for GMR_$L3_POINT_T illustrates a single element.

GMR_$L3_POINT_PTR_T

A 4-byte pointer to a GMR_$L3_POINT_T data type.

GMR_$L3_VECTOR_T

Specifies the x-, y-, and z-coordinates of a 3D vector as three 4-byte integers. The following diagram illustrates this data type:



Field Description:

x
The x-coordinate of the vector.

y
The y-coordinate of the vector.

z
The z-coordinate of the vector.

GMR_$L3_VECTOR_ARRAY_T

An array of GMR_$L3_VECTOR_T with GMR_$MAX_ARRAY_LEN elements (one x,y,z triplet per element). The diagram for GMR_$L3_VECTOR_T illustrates a single element.

GMR_$L3_VECTOR_PTR_T

A 4-byte pointer to a GMR_$L3_VECTOR_T data type.

GMR_$LINE_TYPE_T

A 2-byte integer in the range [1, GMR_$MAX_LINE_TYPE], inclusive. Specifies the line type used for rendering line primitives.

GMR_$MARK_SCALE_T

A 4-byte real number.

GMR_$MARK_TYPE_T

A 2-byte integer.

GMR_$MULT_ORDER_T

A 2-byte integer. Indicates the order of concatenation of matrices. One of the following predefined values:

GMR_$MAT_PRE_MULT
Concatenates on the left.

GMR_$MAT_POST_MULT
Concatenates on the right.

GMR_$MAT_REPLACE
Replaces the matrix.

GMR_$NAME_SET_T

An array of 2-byte integers with GMR_$MAX_NAME_ELEMENT elements.

GMR_$PICK_DATA_T

A variable length record that returns the path of an element that is selected by GMR_$PICK. The constant GMR_$PICK_DATA_SIZE determines the maximum amount of space needed. This is a Pascal record that contains different data types. Parameters are listed below along with the format, description, and byte offset. Refer to the appropriate data type in this section for more information about the structure of each type.

Parameter	Format	Description	Byte offset
element_type	gmr_$element_type_t	1 2-byte integer	0
pick_path_depth	gmr_$instance_pathlength_t	1 2-byte integer	2
pick_path	gmr_$instance_path_t;	An array of records	4

In the above list the parameter pick_path is an array of records in the range [1, GMR_$MAX_INSTANCE_DEPTH]. Each record in the pick_path array contains two 4-byte integers. The length of pick_path is (GMR_MAX_INSTANCE_DEPTH*8).

The length (in bytes) of the entire record is

```
(( GMR_$MAX_INSTANCE_DEPTH *8) + 4)

    where:
         4  is the combined length of element_type and
            pick_path_depth.
```

FORTRAN users, refer to the Usage section of GMR_$PICK for an example.

GMR_$PICK_ECHO_METHOD_T

A 2-byte integer. Specifies the pick echo method for a viewport. One of the following predefined values:

GMR_$PICK_ECHO_NONE
Do not echo the picked element. This is the default method.

GMR_$PICK_ECHO_ABLOCK
Use the echo method described in the highlight attribute block associated with the viewport.

GMR_$PICK_ECHO_BBOX
Draw a bounding box around the structure containing the selected element or subtree.

GMR_$PICK_METHOD_T

Specifies the pick method for a viewport. Currently, there is only one predefined value:

GMR_$PICK_ELEMENT
Pick the nth element that crosses the pick aperture and also satisfies the pick criteria. The value of n is defined by the pick index argument of GMR_$PICK.

GMR_$PLANE_T

A 2-byte integer. Used for setting the color map and determining how many definable colors are available.

GMR_$PRINT_STYLE_T

A 2-byte integer. Specifies the print style used by GMR_$PRINT_DISPLAY and GMR_$PRINT_VIEWPORT. Currently, there is only one predefined value: GMR_$POSTSCRIPT.

GMR_$PROJECTION_T

A 2-byte integer. Specifies the view projection. One of the following predefined values:

GMR_$PERSPECTIVE
Gives a perspective effect centered at any point within the world coordinate system. The view volume is in the shape of a frustum.

GMR_$ORTHOGRAPHIC
Standard parallel projection. The view volume is in the shape of a rectangular parallelepiped. This is the default.

GMR_$PLAN_OBLIQUE
Parallel projection using a foreshortening ratio and a receding angle. The foreshortening ratio specifies how much any lines perpendicular to the view plane are foreshortened in projection. The receding angle is the angle between the U axis and the horizontal. Measure the receding angle counterclockwise from the horizontal ("right") direction at which the U axis is displayed. Receding lines are displayed vertically on the screen.

GMR_$ELEV_OBLIQUE
Parallel projection using both a foreshortening ratio and a receding angle. The receding angle specifies the angle of receding lines relative to the positive U-axis. This is the direction on the view plane onto which the positive gaze direction is projected.

GMR_$REFRESH_PTR_T

Pointer to procedure for refreshing windows, with the following argument protocol (note that for C and FORTRAN, the Pascal Boolean type is 1 byte):

```
IN unobscured        : boolean
IN pos_change        : boolean
IN old_device_limits : GMR_$F3_LIMITS_T
IN old_max_device    : GMR_$F3_LIMITS_T
```

GMR_$REFRESH_STATE_T	A 2-byte integer. Specifies the refresh state of the viewport. One of the following predefined values: GMR_$REFRESH_WAIT When you modify elements in the file, the viewport is rewritten when you call GMR_$VIEWPORT_REFRESH or GMR_$DISPLAY_REFRESH. GMR_$REFRESH_INHIBIT When you modify elements in the file, the viewport is rewritten only when you call GMR_$VIEWPORT_REFRESH. GMR_$DISPLAY_REFRESH does not affect a viewport in this refresh state. GMR_$REFRESH_PARTIAL Individual elements are updated as they are changed in the metafile. When deleting or replacing an element (or subtree if the element is an instance), the element (or subtree) is erased by drawing in the background color. When inserting or replacing, the new element (or subtree) is drawn without regard to other elements on the display. The viewport is completely redrawn when you call GMR_$VIEWPORT_REFRESH or GMR_$DISPLAY_REFRESH. GMR_$REFRESH_UPDATE The viewport is completely redrawn every time that you change a displayed structure.
GMR_$RGB_COLOR_T	A three-element array of real values. Specifies color values in this order: red, green, blue.
GMR_$STRING_T	An array of characters. The maximum number of characters is GMR_$MAX_STRING_LENGTH.
GMR_$STRUCTURE_ID_T	Specifies the structure identification as a 4-byte integer.
GMR_$STRUCTURE_MASK_T	A 4-byte integer. Specifies the mask value for a viewport visibility or pick test.
GMR_$STRUCTURE_NAME_T	Specifies the structure name. An array of characters in the range [0, GMR_$MAX_STRUCTURE_NAME_LENGTH -1].
GMR_$STRUCTURE_VALUE_T	A 4-byte integer. Specifies the structure value for a viewport visibility or pick range test.

GMR_$TEXT_EXPANSION_T

A 4-byte real value. The expansion factor for text size. This is the ratio of width to height as different from the font in use. A value of 1.0 makes the text the same width as the width in the font.

GMR_$TEXT_HEIGHT_T

A 4-byte real value. Specifies the text height in viewing coordinates (same as world).

GMR_$TEXT_PATH_T

A 4-byte real value. The text path angle. This is the angle that determines where the second and subsequent characters in a string are placed. A 4-byte real value. An angle of 0 degrees is to the right of the up vector. Angles greater than 0 degrees are measured counterclockwise from the 0 degree position.

GMR_$TEXT_SLANT_T

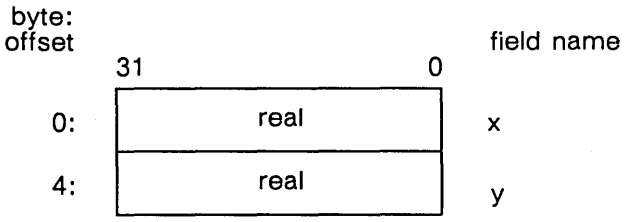
A 4-byte real value. The amount that the top of the text is shifted. The amount is determined by mutiplying the text attribute slant, height, and expansion factor. Zero is the default. >0 but <1 yields an italics-like character.

GMR_$TEXT_SPACING_T

A 4-byte real value. Specifies the spacing between text characters as a fraction of the text height.

GMR_$TEXT_UP_T

A pair of 4-byte real values with the same structure as GMR_$F2_VECTOR_T. Specifies the up direction of text as a 2D vector in world coordinates. Text is oriented on the projection plane. An up vector of (0.0, 1.0) is commonly used. The following diagram illustrates this data type:



Field Description:

x
The x-coordinate of the vector.

y
The y-coordinate of the vector.

GMR_$VIEWPORT_ID_T

Specifies the viewport identification as a 2-byte integer.

GMR_$VIEW_PARAM_BLOCK_T

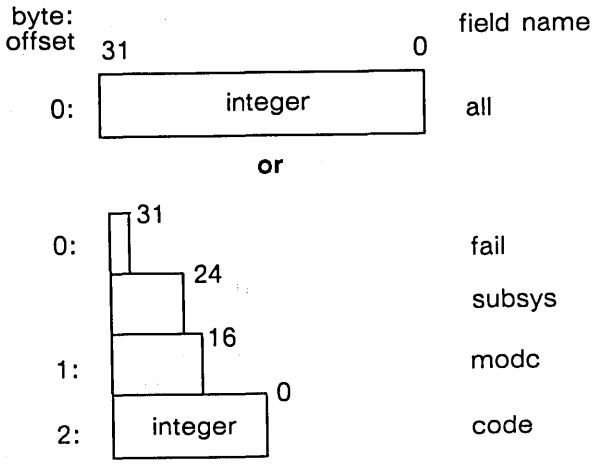
A 76-byte record that returns complete information about the viewing operation. This is a Pascal record that contains different data types. The listing below shows parameters along with the format, description, and byte offset. Refer to the appropriate data type in this section for more information about the structure of each type.

Parameter	Format	Description	Byte offset
reference	GMR_$F3_POINT_T	3 4-byte reals	0
normal	GMR_$F3_VECTOR_T	3 4-byte reals	12
up	GMR_$F3_VECTOR_T	3 4-byte reals	24
window	GMR_$F2_LIMITS_T	4 4-byte reals	36
h_dist	GMR_$F_T	4-byte real	52
y_dist	GMR_$F_T	4-byte real	56
v_dist	GMR_$F_T	4-byte real	60
fshorten	GMR_$F_T	4-byte real	64
recede	GMR_$F_T	4-byte real	68
proj_type	GMR_$PROJECTION_T	2-byte integer	72
coord_sys	GMR_$COORD_SYSTEM_T	2-byte integer	74

The normalizing matrix is derived from GMR_$VIEW_PARAM_BLOCK_T if it has not been specified directly by GMR_$VIEW_SET_TRANSFORM.

STATUS_$T

A status code. The following diagram illustrates this data type:



Field Description:

all
All 32 bits in the status code.

code
A signed number that identifies the type of error that occurred.

modc
The module that encountered the error.

subsys
The subsystem that encountered the error.

fail
The fail bit. If this bit is set, the error was not within the scope of the module invoked, but occurred within a lower-level module.

Chapter 2
3D GMR Routines

This chapter lists user-callable routine descriptions alphabetically for quick reference. Each routine description contains:

- An abstract of the routine's function
- The order of the routine parameters
- A brief description of each parameter
- A description of the routine's function and use

If the parameter can be declared using a predefined data type, the description contains the phrase "in XXX format", where XXX is the predefined data type. Pascal and C programmers, look for this phrase to determine how to declare a parameter.

FORTRAN programmers, look for the phrase that describes the data type in atomic terms, such as "This parameter is a 2-byte integer." For a complete description of each data type see Chapter 1.

The rest of the parameter description describes the use of the parameter and the values it may hold.

The following is an example of a parameter description:

access The access mode, in GMR_$ACC_CREATE_T format. This parameter is a 2-byte integer. Specify only one of the following predefined values:

GMR_$WRITE If the file already exists, an error code is returned in the status parameter.

GMR_$OVERWRITE
If the file already exists, the previous version is deleted.

GMR_$UPDATE
If the file already exists, the previous version is opened.

GMR_$4X3_MATRIX_CONCATENATE

Concatenates the two given 4x3 matrices and returns the resulting matrix.

FORMAT

```
GMR_$4X3_MATRIX_CONCATENATE (matrix1, matrix2, matrix, status)
```

INPUT PARAMETERS

matrix1
A 4x3 matrix, in GMR_$4X3_MATRIX_T format. This parameter is a two-dimensional array of 4-byte real values. See the Data Types section for more information.

matrix2
A 4x3 matrix, in GMR_$4X3_MATRIX_T format. This parameter is a two-dimensional array of 4-byte real values. See the Data Types section for more information.

OUTPUT PARAMETERS

matrix
The 4x3 matrix, in GMR_$4X3_MATRIX_T format, resulting from concatenating matrix1 and matrix2. This parameter is a two-dimensional array of 4-byte real values (see the Data Types section for more information). This operation is mathematically equivalent to augmenting matrix1 and matrix2 with the fourth column of the identity matrix, forming the matrix product, and returning the first three columns of the result. Matrix1 is post-multiplied by matrix2.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The order of concatenation is Matrix := matrix1 X matrix2

For FORTRAN Users:

The matrix calls expect data to be stored in row-major form. Both C and Pascal store two-dimensional arrays this way. FORTRAN stores data in column-major form, so an array (n,m) in FORTRAN has m as the major dimension and n as the minor. Both C and Pascal use n as the major and m as the minor.

The following example shows a point array of M rows and N columns in each language:

```
C:          GMR_$F3_POINT_T  my_array [M] [N]

FORTRAN:    REAL MY_ARRAY (N)(M)(3)

Pascal:
     my_array : ARRAY [1 .. M] OF ARRAY [1 .. N] OF GMR_$F3_POINT_T
```

GMR_$4X3_MATRIX_IDENTITY

Returns the 4x3 identity modeling matrix.

FORMAT

```
GMR_$4X3_MATRIX_IDENTITY (matrix, status)
```

OUTPUT PARAMETERS

matrix
The 4x3 identity modeling matrix, in GMR_$4X3_MATRIX_T format. This is the 4x4 identity matrix minus the last column.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

FORTRAN users: See Usage under GMR_$4X3_MATRIX_CONCATENATE.

GMR_$4X3_MATRIX_INVERT

Returns the inverse of a 4x3 matrix.

FORMAT

```
GMR_$4X3_MATRIX_INVERT (matrix, inverse, status)
```

INPUT PARAMETERS

matrix

The original matrix, in GMR_$4X3_MATRIX_T format. This parameter is a two-dimensional array of 4-byte real values. See the Data Types section for more information.

OUTPUT PARAMETERS

inverse

The inverse of the input matrix, in GMR_$4X3_MATRIX_T format. This is mathematically equivalent to appending the fourth column of the identity to the original matrix and returning the first three columns of the inverse of the resulting matrix.

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

FORTRAN users: See Usage under GMR_$4X3_MATRIX_CONCATENATE.

GMR_$4X3_MATRIX_REFLECT

Specifies a reflection (or mirroring) through an arbitrary plane.

FORMAT

```
GMR_$4X3_MATRIX_REFLECT (order, point, vector, matrix, status)
```

INPUT PARAMETERS

order

The order of concatenation, in GMR_$MULT_ORDER_T format. The specified matrix is optionally concatenated on the left or on the right by the reflection matrix, or simply replaced by it. The three possible values are GMR_$MAT_PRE_MULT, GMR_$MAT_POST_MULT, and GMR_$MAT_REPLACE.

point

A point on the reflection plane, in GMR_$F3_POINT_T format. These are three 4-byte real values that specify the x-, y-, and z-coordinates of the point. See the Data Types section for more information.

vector

A vector normal to the reflection plane, in GMR_$F3_VECTOR_T format. These are three 4-byte real values that specify the x-, y-, and z-coordinates of the vector. See the Data Types section for more information.

matrix

A 4x3 modeling matrix, in GMR_$4X3_MATRIX_T format. This matrix is concatenated with the reflection matrix.

OUTPUT PARAMETERS

matrix

The original 4x3 matrix concatenated with, or replaced by, the new matrix.

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

FORTRAN users: See Usage under GMR_$4X3_MATRIX_CONCATENATE.

GMR_$4X3_MATRIX_ROTATE

Concatenates the specified 4x3 modeling matrix with a rotation matrix.

FORMAT

```
GMR_$4X3_MATRIX_ROTATE (order, axis, angle, matrix, status)
```

INPUT PARAMETERS

order
The order of concatenation, in GMR_$MULT_ORDER_T format. The specified matrix is optionally concatenated on the left or on the right by the rotation matrix, or simply replaced by it. The three possible values are GMR_$MAT_PRE_MULT, GMR_$MAT_POST_MULT, and GMR_$MAT_REPLACE.

axis
The coordinate axis about which the rotation is to occur, in GMR_$AXIS_T format.

angle
The angle (in radians) of right-handed rotation about the specified axis, in GMR_$F_T format. This parameter is a 4-byte real value.

matrix
A 4x3 modeling matrix, in GMR_$4X3_MATRIX_T format. This matrix is concatenated with the rotation matrix.

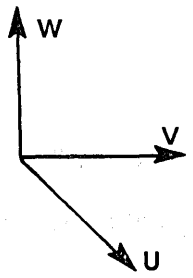
OUTPUT PARAMETERS

matrix
The original 4x3 matrix concatenated with, or replaced by, the rotation matrix.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

In the general case, if U X V = W (cross product), then U is rotated into V (see illustration below). For example, rotation about U is V towards W.



GMR_$4X3_MATRIX_ROTATE_AXIS

Specifies a rotation about an arbitrary axis.

FORMAT

```
GMR_$4X3_MATRIX_ROTATE_AXIS (order, point, vector, angle, matrix, status)
```

INPUT PARAMETERS

order
The order of concatenation, in GMR_$MULT_ORDER_T format. The specified matrix is optionally concatenated on the left or on the right by the rotation matrix, or simply replaced by it. The three possible values are GMR_$MAT_PRE_MULT, GMR_$MAT_POST_MULT, and GMR_$MAT_REPLACE.

point
A point on the rotation axis, in GMR_$F3_POINT_T format. These are three 4-byte real values that specify the x-, y-, and z-coordinates of the point. See the Data Types section for more information.

vector
A non-zero vector that specifies the orientation of the axis, in GMR_$F3_VECTOR_T format. These are three 4-byte real values that specify the x-, y-, and z-coordinates of the vector. See the Data Types section for more information.

angle
The angle (in radians) of right-handed rotation about the specified axis, in GMR_$F_T format. This parameter is a 4-byte real value.

matrix
A 4x3 modeling matrix, in GMR_$4X3_MATRIX_T format. This matrix is concatenated with the rotation matrix.

OUTPUT PARAMETERS

matrix
The original 4x3 matrix concatenated with, or replaced by, the rotation matrix.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

See Usage under GMR_$4X3_MATRIX_ROTATE.

FORTRAN users: See Usage under GMR_$4X3_MATRIX_CONCATENATE.

GMR_$4X3_MATRIX_SCALE

Concatenates the specified 4x3 modeling matrix with a scaling matrix.

FORMAT

```
GMR_$4X3_MATRIX_SCALE (order, scale, matrix, status)
```

INPUT PARAMETERS

order
The order of concatenation, in GMR_$MULT_ORDER_T format. The specified matrix is optionally concatenated on the left or on the right by the scaling matrix, or simply replaced by it. The three possible values are GMR_$MAT_PRE_MULT, GMR_$MAT_POST_MULT, and GMR_$MAT_REPLACE.

scale
The scaling vector, in GMR_$F3_VECTOR_T format. These are three 4-byte real values that specify the x, y, and z scale factors.

matrix
A 4x3 modeling matrix, in GMR_$4X3_MATRIX_T format. This matrix is concatenated with the scaling matrix.

OUTPUT PARAMETERS

matrix
The original 4x3 matrix concatenated with, or replaced by, the scaling matrix.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

FORTRAN users: See Usage under GMR_$4X3_MATRIX_CONCATENATE.

GMR_$4X3_MATRIX_TRANSLATE

Concatenates the specified 4x3 modeling matrix with a 4x3 translation matrix.

FORMAT

```
GMR_$4X3_MATRIX_TRANSLATE (order, translation, matrix, status)
```

INPUT PARAMETERS

order
The order of concatenation, in GMR_$MULT_ORDER_T format. The specified matrix is optionally concatenated on the left or on the right by the translation matrix, or simply replaced by it. The three possible values are GMR_$MAT_PRE_MULT, GMR_$MAT_POST_MULT, and GMR_$MAT_REPLACE. This parameter is a 2-byte integer.

translation
The translation vector, in GMR_$F3_VECTOR_T format. These are three 4-byte real values that specify the amount of translation in modeling coordinates.

matrix
A 4x3 modeling matrix, in GMR_$4X3_MATRIX_T format. This matrix is concatenated with the translation matrix.

OUTPUT PARAMETERS

matrix
The original 4x3 matrix concatenated with, or replaced by, the translation matrix.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

FORTRAN users: See Usage under GMR_$4X3_MATRIX_CONCATENATE.

GMR_$ABLOCK_ASSIGN_DISPLAY

Assigns an attribute block (by number) to an attribute class, for all viewports of the display that have not already explicitly assigned that attribute class.

FORMAT

```
GMR_$ABLOCK_ASSIGN_DISPLAY (aclass_id, ablock_id, status)
```

INPUT PARAMETERS

aclass_id
The identification number of the attribute class to which the attribute block will be assigned, in GMR_$ACLASS_ID_T format. This parameter is a 2-byte integer.

ablock_id
The identification number of the attribute block to be assigned to the attribute class, in GMR_$ABLOCK_ID_T FORMAT. This parameter is a 2-byte integer.

To assign the default attributes to an attribute class for the display, use ablock_id = GMR_$DEFAULT_ABLOCK.

To ensure that no attribute values are changed from their previous value, use ablock_id = GMR_$NOCHANGE_ABLOCK.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$ABLOCK_ASSIGN_DISPLAY to assign an existing attribute block to an attribute class for all viewports in the display that have not explicitly assigned that attribute class.

Use GMR_$ABLOCK_INQ_ASSIGN_DISPLAY to inquire about the current attribute block number assigned to a particular class for the display.

Assignments of attribute blocks to attribute classes for individual viewports using GMR_$ABLOCK_ASSIGN_VIEWPORT override assignments made by GMR_$ABLOCK_ASSIGN_DISPLAY.

If you do not assign an ablock to the display, the default ablock is used (GMR_$DEFAULT_ABLOCK).

GMR_$ABLOCK_ASSIGN_VIEWPORT

Assigns an attribute block (by number) to an attribute class for one viewport.

FORMAT

```
GMR_$ABLOCK_ASSIGN_VIEWPORT (aclass_id, viewport_id, ablock_id, status)
```

INPUT PARAMETERS

aclass_id
The identification number of the attribute class to which the attribute block will be assigned, in GMR_$ACLASS_ID_T format. This parameter is a 2-byte integer.

viewport_id
The identification number of the viewport in which to assign the attribute block to the attribute class, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer.

ablock_id
The identification number of the attribute block to be assigned to the attribute class, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

To assign the default attributes to an attribute class for one viewport, use ablock_id = GMR_$DEFAULT_ABLOCK.

To ensure that no attribute values are changed from their previous value, use ablock_id = GMR_$NOCHANGE_ABLOCK.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$ABLOCK_ASSIGN_VIEWPORT to assign an existing attribute block to an attribute class for one viewport in the display.

Use GMR_$ABLOCK_INQ_ASSIGN_VIEWPORT to inquire about the current attribute block number assigned to a particular attribute class for a particular viewport.

Assignments of attribute blocks to attribute classes for individual viewports using GMR_$ABLOCK_ASSIGN_VIEWPORT override assignments made by GMR_$ABLOCK_ASSIGN_DISPLAY.

If you do not assign an ablock to the viewport, the display ablock for the attribute class is used.

GMR_$ABLOCK_COPY

Copies all attributes from one existing attribute block to another.

FORMAT

```
GMR_$ABLOCK_COPY (source_ablock_id, destination_ablock_id, status)
```

INPUT PARAMETERS

source_ablock_id
The identification number of the existing attribute block from which attributes will be copied, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

destination_ablock_id
The identification number of the existing attribute block to which the attributes of the attribute block source_block_id will be copied, in GMR_$ABLOCK_ID_T format.

You may copy attributes from the default or no-change attribute blocks, but you cannot copy attributes into them. GMR_$NOCHANGE_ABLOCK is a list of no-change attribute values; GMR_$DEFAULT_ABLOCK is a list of default attribute values.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

GMR_$ABLOCK_CREATE

Creates an attribute block and initializes it equivalent to an existing block.

FORMAT

```
GMR_$ABLOCK_CREATE (source_ablock_id, ablock_id, status)
```

INPUT PARAMETERS

source_ablock_id
The identification number of the existing attribute block used as the source for the block generated with GMR_$ABLOCK_CREATE, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

ablock_id
The identification number assigned to the attribute block generated by GMR_$ABLOCK_CREATE, in GMR_$ABLOCK_ID_T format.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$ABLOCK_CREATE to establish a new attribute block identical to an existing one. Use GMR_$ABLOCK_COPY to copy attributes from an existing attribute block to another existing ablock.

To delete an attribute block, use GMR_$ABLOCK_DELETE. This releases the attribute block identification number; this released number may then be reassigned in response to another call to GMR_$ABLOCK_CREATE.

When 3D GMR is initialized, two attribute blocks are created. The first is the no-change attribute block GMR_$NOCHANGE_ABLOCK). The second is the default attribute block GMR_$DEFAULT_ABLOCK.

GMR_$ABLOCK_DELETE

Deletes an attribute block and releases the attribute block identification number.

FORMAT

```
GMR_$ABLOCK_DELETE (ablock_id, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the existing attribute block to be deleted, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$ABLOCK_DELETE to release the attribute block identification number; this released number may then be reassigned in response to another call to GMR_$ABLOCK_CREATE.

The no-change attribute block (GMR_$NOCHANGE_ABLOCK) and the default attribute block (GMR_$DEFAULT_ABLOCK) cannot be deleted.

GMR_$ABLOCK_INQ_ASSIGN_DISPLAY

Returns the current attribute block number assigned to a particular attribute class for the display.

FORMAT

```
GMR_$ABLOCK_INQ_ASSIGN_DISPLAY (aclass_id, ablock_id, status)
```

INPUT PARAMETERS

aclass_id
The identification number of the attribute class for which to return the current attribute block assignment, in GMR_$ACLASS_ID_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

ablock_id
The identification number of the attribute block currently assigned to the specified attribute class for the display, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

If you have not assigned an attribute block to the specified attribute class for the display, the returned value is the default attribute block (GMR_$DEFAULT_ABLOCK).

Use GMR_$ABLOCK_ASSIGN_DISPLAY to assign an attribute block to all viewports in the display.

Use GMR_$ABLOCK_ASSIGN_VIEWPORT to assign an attribute block to the aclass of a specific viewport. This overrides the attributes assigned by GMR_$ABLOCK_ASSIGN_DISPLAY.

See also GMR_$ABLOCK_ASSIGN_VIEWPORT.

GMR_$ABLOCK_INQ_ASSIGN_VIEWPORT

Returns the current attribute block number assigned to a particular attribute class for a specified viewport.

FORMAT

```
GMR_$ABLOCK_INQ_ASSIGN_VIEWPORT (aclass_id, viewport_id, ablock_id, status)
```

INPUT PARAMETERS

aclass_id
The identification number of the attribute class for which to return the current attribute block assignment, in GMR_$ACLASS_ID_T format. This parameter is a 2-byte integer.

viewport_id
The identification number of the viewport for which to return the current attribute block identification number, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

ablock_id
The identification number of the attribute block assigned to the attribute class for the display, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

If you have not assigned an attribute block to the specified attribute class for the viewport, the returned value is the attribute block assigned to the attribute class for the display.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$ABLOCK_ASSIGN_VIEWPORT to assign an attribute block to a viewport.

Use GMR_$ABLOCK_ASSIGN_DISPLAY to assign the aclass of all viewports to the attribute block. This assignment is overridden by GMR_$ABLOCK_ASSIGN_VIEWPORT.

See also GMR_$ABLOCK_ASSIGN_DISPLAY.

GMR_$ABLOCK_INQ_FILL_COLOR

Returns the color used for the interior of polygons and meshes and the enabled state for the specified attribute block.

FORMAT

```
GMR_$ABLOCK_INQ_FILL_COLOR (ablock_id, color, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

color
The fill color for this attribute block, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer.

If the attribute is enabled for use, the value returned is GMR_$SET_VALUE_AND_ENABLE. If the attribute is in no-change state (disabled), the value returned is GMR_$SET_VALUE_AND_DISABLE.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

If the fill color for the attribute block has not been modified, the default fill color of the source block specified when the block was created (or last copied) is returned. The color of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$FILL_COLOR_DEF. This is equivalent to 1.

Use GMR_$ABLOCK_SET_FILL_COLOR to change the color of polygons and meshes and the enabled attribute state in an attribute block.

GMR_$ABLOCK_INQ_FILL_INTEN

Returns the fill intensity used for polygons and meshes and the enabled state for the specified attribute block.

FORMAT

```
GMR_$ABLOCK_INQ_FILL_INTEN (ablock_id, intensity, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

intensity
The fill intensity for this attribute block, in GMR_$INTEN_T format. This is 4-byte real value in the range [0.0, 1.0], inclusive.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer. If the attribute is enabled for use, the value returned is GMR_$SET_VALUE_AND_ENABLE. If the attribute is in no-change state (disabled), the value returned is GMR_$SET_VALUE_AND_DISABLE.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

If the fill intensity for the attribute block has not been modified, the default fill intensity of the source block specified when the block was created (or last copied) is returned. The intensity of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$FILL_INTEN_DEF. This is equivalent to 1.0.

Use GMR_$ABLOCK_SET_FILL_INTEN to change the fill intensity and the enabled attribute state in an attribute block.

GMR_$ABLOCK_INQ_LINE_COLOR

Returns the polyline/multiline color and the enabled state for the specified attribute block.

FORMAT

```
GMR_$ABLOCK_INQ_LINE_COLOR (ablock_id, color, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

color
The line color for this attribute block, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2 byte integer.

If the attribute is enabled for use, the value returned is GMR_$SET_VALUE_AND_ENABLE. If the attribute is in no-change state (disabled), the value returned is GMR_$SET_VALUE_AND_DISABLE.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

If the line color for the attribute block has not been modified, the default color of the source block specified when the block was created (or last copied) is returned. The color of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$LINE_COLOR_DEF. This is equivalent to 1.

Use GMR_$ABLOCK_SET_LINE_COLOR to change the polyline/multiline color and enabled attribute state in an attribute block.

GMR_$ABLOCK_INQ_LINE_INTEN

Returns the polyline/multiline intensity and the enabled state for the specified attribute block.

FORMAT

```
GMR_$ABLOCK_INQ_LINE_INTEN (ablock_id, intensity, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

intensity
The line intensity for this attribute block, in GMR_$INTEN_T format. This is 4-byte real value in the range [0.0, 1.0], inclusive.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer. If the attribute is enabled for use, the value returned is GMR_$SET_VALUE_AND_ENABLE. If the attribute is in no-change state (disabled), the value returned is GMR_$SET_VALUE_AND_DISABLE.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

If the line intensity for the attribute block has not been modified, the default intensity of the source block specified when the block was created (or last copied) is returned. The intensity of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$LINE_INTEN_DEF. This is equivalent to 1.0.

Use GMR_$ABLOCK_SET_LINE_INTEN to change the polyline/multiline intensity and enabled attribute state in an attribute block.

GMR_$ABLOCK_INQ_LINE_TYPE

Returns the polyline/multiline type ID and the enabled state the specified attribute block.

FORMAT

```
GMR_$ABLOCK_INQ_LINE_TYPE (ablock_id, type_id, change, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

type_id
The line type ID color for this attribute block, in GMR_$LINE_TYPE_T format. This parameter is a 2-byte integer. Values are currently restricted to 1, 2, 3, and 4 as follows:

```
1 = Solid
2 = Dashed
3 = Dotted
4 = Dashed-dotted
```

change
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer.

If the attribute is enabled for use, the value returned is GMR_$SET_VALUE_AND_ENABLE. If the attribute is in no-change state (disabled), the value returned is GMR_$SET_VALUE_AND_DISABLE.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

If the line type ID of the attribute block has not been modified, the line type ID of the source block specified when the block was created (or last copied) is returned. The line type ID of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$LINE_TYPE_DEF. This is equivalent to 1 (solid).

Use GMR_$ABLOCK_SET_LINE_TYPE to change the polyline/multiline type ID and the enabled attribute state of an attribute block.

GMR_$ABLOCK_INQ_MARK_COLOR

Returns the polymarker color and the enabled state for the specified attribute block.

FORMAT

```
GMR_$ABLOCK_INQ_MARK_COLOR (ablock_id, color, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

color
The polymarker color for this attribute block, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer.

If the attribute is enabled for use, the value returned is GMR_$SET_VALUE_AND_ENABLE. If the attribute is in no-change state (disabled), the value returned is GMR_$SET_VALUE_AND_DISABLE.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

If the polymarker color of the attribute block has not been modified, the polymarker color of the source block specified when the block was created (or last copied) is returned. The color of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$MARK_COLOR_DEF. This is equivalent to 1.

Use GMR_$ABLOCK_SET_MARK_COLOR to change the polymarker color and the enabled attribute state of an attribute block.

GMR_$ABLOCK_INQ_MARK_INTEN

Returns the polymarker intensity and the enabled state for the specified attribute block.

FORMAT

```
GMR_$ABLOCK_INQ_MARK_INTEN (ablock_id, intensity, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

intensity
The polymarker intensity for this attribute block, in GMR_$INTEN_T format. This parameter is a 4-byte real value in the range 0.0 to 1.0 inclusive.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer. If the attribute is enabled for use, the value returned is GMR_$SET_VALUE_AND_ENABLE. If the attribute is in no-change state (disabled), the value returned is GMR_$SET_VALUE_AND_DISABLE.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

If the polymarker intensity of the attribute block has not been modified, the polymarker intensity of the source block specified when the block was created (or last copied) is returned. The intensity of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$MARK_INTEN_DEF. This is equivalent to 1.0.

Use GMR_$ABLOCK_SET_MARK_INTEN to change the polymarker intensity and the enabled attribute state of an attribute block.

GMR_$ABLOCK_INQ_MARK_SCALE

Returns the polymarker scale factor and the enabled state for the specified attribute block.

FORMAT

```
GMR_$ABLOCK_INQ_MARK_SCALE (ablock_id, scale_factor, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

scale_factor
The polymarker scale factor for this attribute block, in GMR_$MARK_SCALE_T format. This parameter is a 4-byte real value. The default scale factor is 1.0.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer.

If the attribute is enabled for use, the value returned is GMR_$SET_VALUE_AND_ENABLE. If the attribute is in no-change state (disabled), the value returned is GMR_$SET_VALUE_AND_DISABLE.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

If the scale factor of the attribute block has not been modified, the polymarker scale factor of the source block specified when the block was created (or last copied) is returned. The scale factor of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$MARK_SCALE_DEF. This is equivalent to 1.

Use GMR_$ABLOCK_SET_MARK_SCALE to change the scale factor and the enabled attribute state of an attribute block.

The scale factor does not affect the size of marker type 1 (one pixel).

GMR_$ABLOCK_INQ_MARK_TYPE

Returns the polymarker type and the enabled state for the specified attribute block.

FORMAT

```
GMR_$ABLOCK_INQ_MARK_INTEN (ablock_id, type, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

type
The polymarker type for this attribute block, in GMR_$MARK_TYPE_T format. This parameter is a 2-byte integer in the range [1, 5], inclusive.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer.

If the attribute is enabled for use, the value returned is GMR_$SET_VALUE_AND_ENABLE. If the attribute is in no-change state (disabled), the value returned is GMR_$SET_VALUE_AND_DISABLE.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

If the type of the attribute block has not been modified, the polymarker type of the source block specified when the block was created (or last copied) is returned. The type of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$MARK_TYPE_DEF. This is equivalent to 1.

Use GMR_$ABLOCK_SET_MARK_INTEN to change the polymarker type and the enabled attribute state of an attribute block.

Note that the scale factor does not affect the size of marker type 1 (one pixel).

See GMR_$ABLOCK_SET_MARK_TYPE for a graphic example of marker types.

GMR_$ABLOCK_INQ_TEXT_COLOR

Returns the text color and the enabled state for the specified attribute block.

FORMAT

```
GMR_$ABLOCK_INQ_TEXT_COLOR (ablock_id, color, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

color
The text color for this attribute block, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer.

If the attribute is enabled for use, the value returned is GMR_$SET_VALUE_AND_ENABLE. If the attribute is in no-change state (disabled), the value returned is GMR_$SET_VALUE_AND_DISABLE.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

If the text color for the attribute block has not been modified, the default color of the source block specified when the block was created (or last copied) is returned. The color of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$TEXT_COLOR_DEF. This is equivalent to 1.

Use GMR_$ABLOCK_SET_TEXT_COLOR to change the text color and enabled attribute state in an attribute block.

GMR_$ABLOCK_INQ_TEXT_EXPANSION

Returns the text expansion and the enabled state for the specified attribute block. Text expansion controls the ratio of height to width of text characters.

FORMAT

```
GMR_$ABLOCK_INQ_TEXT_EXPANSION (ablock_id, expansion, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

expansion
The text character expansion for this attribute block, in GMR_$TEXT_EXPANSION_T format. This is a 4-byte real value. This attribute controls the aspect ratio for the font. The default value is 1.0 which preserves the aspect ratio defined in the font.

Values greater than 1.0 create wider characters. Values less than 1.0 create slimmer characters.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer.

If the attribute is enabled for use, the value returned is GMR_$SET_VALUE_AND_ENABLE. If the attribute is in no-change state (disabled), the value returned is GMR_$SET_VALUE_AND_DISABLE.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

If the text expansion for the attribute block has not been modified, the default expansion of the source block specified when the block was created (or last copied) is returned. The expansion of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$TEXT_EXPANSION_DEF. This is equivalent to 1.0.

Use GMR_$ABLOCK_SET_TEXT_EXPANSION to change the text character expansion and the enabled attribute state in an attribute block.

GMR_$ABLOCK_INQ_TEXT_HEIGHT

Returns the text height and the enabled state for the specified attribute block. Text height controls the actual size of text characters.

FORMAT

```
GMR_$ABLOCK_INQ_TEXT_HEIGHT (ablock_id, height, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

height
The text character height for this attribute block, in GMR_$TEXT_HEIGHT_T format. This is a 4-byte real value in viewing coordinates (same as world coordinates).

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This is a 2-byte integer.

If the attribute is enabled for use, the value returned is GMR_$SET_VALUE_AND_ENABLE. If the attribute is in no-change state (disabled), the value returned is GMR_$SET_VALUE_AND_DISABLE.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

If the text height for the attribute block has not been modified, the default height of the source block specified when the block was created (or last copied) is returned. The height of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$TEXT_HEIGHT_DEF. This is equivalent to 0.01.

Use GMR_$ABLOCK_SET_TEXT_HEIGHT to change the text character height and enabled attribute state in an attribute block.

GMR_$ABLOCK_INQ_TEXT_INTEN

Returns the text intensity and the enabled state for the specified attribute block.

FORMAT

```
GMR_$ABLOCK_INQ_TEXT_INTEN (ablock_id, intensity, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

intensity
The text intensity for this attribute block, in GMR_$INTEN_T format. This is a 4-byte real value in the range [0.0, 1.0], inclusive.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This is a 2-byte integer.

If the attribute is enabled for use, the value returned is GMR_$SET_VALUE_AND_ENABLE. If the attribute is in no-change state (disabled), the value returned is GMR_$SET_VALUE_AND_DISABLE.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

If the text intensity for the attribute block has not been modified, the default intensity of the source block specified when the block was created (or last copied) is returned. The intensity of GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$TEXT_INTEN_DEF. This is equivalent to 1.0.

Use GMR_$ABLOCK_SET_TEXT_INTEN to change the text intensity and enabled attribute state in an attribute block.

GMR_$ABLOCK_INQ_TEXT_PATH

Returns the text path angle and the enabled state for the specified attribute block. Text path determines where the second and subsequent characters in a text string are placed.

FORMAT

```
GMR_$ABLOCK_INQ_TEXT_PATH (ablock_id, angle, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

angle
The angle that determines where the second and subsequent characters in a text string are placed, in GMR_$TEXT_PATH_T format. This parameter is a 4-byte real value.

An angle of 0.0 radians is to the right of the up vector. Angles greater than 0.0 radians are measured counterclockwise from the 0.0 radian position.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This is a 2-byte integer.

If the attribute is enabled for use, the value returned is GMR_$SET_VALUE_AND_ENABLE. If the attribute is in no-change state (disabled), the value returned is GMR_$SET_VALUE_AND_DISABLE.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

If the text path for the attribute block has not been modified, the default path of the source block specified when the block was created (or last copied) is returned. The path of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$TEXT_PATH_DEF. This is equivalent to 0.0 and places characters to the right of preceding characters.

Use GMR_$ABLOCK_SET_TEXT_PATH to change the text character path and enabled attribute state in an attribute block.

GMR_$ABLOCK_INQ_TEXT_SLANT

Returns the text slant factor and the enabled state for the specified attribute block. A negative value produces a left slant. A positive value produces a right slant.

FORMAT

```
GMR_$ABLOCK_INQ_TEXT_SLANT (ablock_id, slant, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

slant
The amount that the top of the charatcer is shifted, in GMR_$TEXT_SLANT_T format. This parameter is a 4-byte real value.

The amount is determined by multiplying the text attributes for slant, height, and expansion factor. A value in the range [0.0, 1.0] exclusive yields an italics-like character. The default value is GMR_$TEXT_SLANT_DEF which is equivalent to 0.0 (no slant).

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer.

If the attribute is enabled for use, the value returned is GMR_$SET_VALUE_AND_ENABLE. If the attribute is in no change state (disabled), the value returned is GMR_$SET_VALUE_AND_DISABLE.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

If the text slant for the attribute block has not been modified, the default slant of the source block specified when the block was created (or last copied) is returned. The slant of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$TEXT_SLANT_DEF (no slant). This is equivalent to 0.0.

Use GMR_$ABLOCK_SET_TEXT_SLANT to change the text slant of the font and enabled attribute state in an attribute block.

GMR_$ABLOCK_INQ_TEXT_SPACING

Returns the intercharacter spacing and the enabled state for the specified attribute block.

FORMAT

```
GMR_$ABLOCK_INQ_TEXT_SPACING (ablock_id, spacing, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

spacing
The intercharacter spacing for this attribute block, in GMR_$TEXT_SPACING_T format. This is a 4-byte real value that defines spacing as a fraction of text height.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer.

If the attribute is enabled for use, the value returned is GMR_$SET_VALUE_AND_ENABLE. If the attribute is in no-change state (disabled), the value returned is GMR_$SET_VALUE_AND_DISABLE.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

If the text spacing for the attribute block has not been modified, the default intercharacter spacing of the source block specified when the block was created (or last copied) is returned.

The spacing of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$TEXT_SPACING_DEF. This is equivalent to 0.0. This places each character next to the preceding character in the character path direction. For more spacing between characters, make the spacing value positive. To have characters appear to overlay, make the spacing value negative.

Use GMR_$ABLOCK_SET_TEXT_SPACING to change the intercharacter spacing and enabled attribute state in an attribute block.

GMR_$ABLOCK_INQ_TEXT_UP

Returns the up direction of text on the projection plane and the enabled state for the specified attribute block.

FORMAT

```
GMR_$ABLOCK_INQ_TEXT_UP (ablock_id, up_vector, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

up_vector
The text character up vector for this attribute block, in GMR_$TEXT_UP_T format. This parameter is an array of two 4-byte real values in viewing coordinates (same as world coordinates). These values specify an up direction of text on the projection plane.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer If the attribute is enabled for use, the value returned is GMR_$SET_VALUE_AND_ENABLE. If the attribute is in no-change state (disabled), the value returned is GMR_$SET_VALUE_AND_DISABLE.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

If the text up vector for the attribute block has not been modified, the default up direction of the source block specified when the block was created (or last copied) is returned. The up direction of GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$TEXT_UP_X_DEF and GMR_$TEXT_UP_Y_DEF. This is equivalent to (0.0, 1.0) with the height running vertically up the display. A value of (1.0, 0.0) places characters on their side with the top to the right.

Use GMR_$ABLOCK_SET_TEXT_UP to change the text character up direction and enabled attribute state in an attribute block.

GMR_$ABLOCK_SET_FILL_COLOR

Sets the color used to fill polygons and meshes and the enabled state for the specified attribute block.

FORMAT

```
GMR_$ABLOCK_SET_FILL_COLOR (ablock_id, color, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

color
The fill color for this attribute block, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer. Possible values are the following:

GMR_$SET_VALUE_AND_ENABLE
Stores the attribute value and sets the value as enabled.

GMR_$SET_VALUE_AND_DISABLE
Stores the attribute value but sets the no-change attribute, thus disabling the use of this attribute.

GMR_$NO_VALUE_AND_ENABLE
Ignores the attribute value but enables what was previously set as the attribute value.

GMR_$NO_VALUE_AND_DISABLE
Ignores the attribute value and disables the attribute's use. With this last state, the previous attribute value is preserved.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The default is the color of the source block specified when the block was created (or last copied). The color of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$FILL_COLOR_DEF. This is equivalent to 1.

Use GMR_$ABLOCK_INQ_FILL_COLOR to return the fill color and the enabled attribute state in an attribute block.

GMR_$ABLOCK_SET_FILL_INTEN

Sets the fill intensity for polygons and meshes and the enabled state for the specified attribute block.

FORMAT

```
GMR_$ABLOCK_SET_FILL_INTEN (ablock_id, intensity, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

intensity
The fill intensity for this attribute block, in GMR_$INTEN_T format. This parameter is a 4-byte real value in the range [0.0, 1.0], inclusive.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer. Possible values are the following:

GMR_$SET_VALUE_AND_ENABLE
Stores the attribute value and sets the value as enabled.

GMR_$SET_VALUE_AND_DISABLE
Stores the attribute value but sets the no-change attribute, thus disabling the use of this attribute.

GMR_$NO_VALUE_AND_ENABLE
Ignores the attribute value but enables what was previously set as the attribute value.

GMR_$NO_VALUE_AND_DISABLE
Ignores the attribute value and disables the attribute's use. With this last state, the previous attribute value is preserved.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The default intensity for the attribute block is the intensity of the source block specified when the block was created (or last copied). The intensity of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$FILL_INTEN_DEF. This is equivalent to 1.0.

Use GMR_$ABLOCK_INQ_FILL_INTEN to return the fill intensity and the enabled attribute state in an attribute block.

GMR_$ABLOCK_SET_LINE_COLOR

Sets the polyline/multiline color and the enabled state for the specified attribute block.

FORMAT

```
GMR_$ABLOCK_SET_LINE_COLOR (ablock_id, color, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

color
The line color for this attribute block, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer. Possible values are the following:

GMR_$SET_VALUE_AND_ENABLE
: Stores the attribute value and sets the value as enabled.

GMR_$SET_VALUE_AND_DISABLE
: Stores the attribute value but sets the no-change attribute, thus disabling the use of this attribute.

GMR_$NO_VALUE_AND_ENABLE
: Ignores the attribute value but enables what was previously set as the attribute value.

GMR_$NO_VALUE_AND_DISABLE
: Ignores the attribute value and disables the attribute's use. With this last state, the previous attribute value is preserved.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The default is the color of the source block specified when the block was created (or last copied). The color of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$LINE_COLOR_DEF. This is equivalent to 1.

Use GMR_$ABLOCK_INQ_LINE_COLOR to return the polyline/multiline color and the enabled attribute state in an attribute block.

GMR_$ABLOCK_SET_LINE_INTEN

Sets the polyline/multiline intensity and the enabled state for the specified attribute block.

FORMAT

```
GMR_$ABLOCK_SET_LINE_INTEN (ablock_id, intensity, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

intensity
The line intensity for this attribute block, in GMR_$INTEN_T format. This is 4-byte real value in the range [0.0, 1.0], inclusive.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer. Possible values are the following:

GMR_$SET_VALUE_AND_ENABLE
: Stores the attribute value and sets the value as enabled.

GMR_$SET_VALUE_AND_DISABLE
: Stores the attribute value but sets the no-change attribute, thus disabling the use of this attribute.

GMR_$NO_VALUE_AND_ENABLE
: Ignores the attribute value but enables what was previously set as the attribute value.

GMR_$NO_VALUE_AND_DISABLE
: Ignores the attribute value and disables the attribute's use. With this last state, the previous attribute value is preserved.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The default is the intensity of the source block specified when the block was created (or last copied). The intensity of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$LINE_INTEN_DEF. This is equivalent to 1.0.

Use GMR_$ABLOCK_INQ_LINE_INTEN to return the polyline/multiline intensity and the enabled attribute state in an attribute block.

GMR_$ABLOCK_SET_LINE_TYPE

Sets the polyline/multiline type ID and the enabled state for the specified attribute block.

FORMAT

```
GMR_$ABLOCK_SET_LINE_TYPE (ablock_id, type_id, change, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

type_id
The line type ID for this attribute block, in GMR_$LINE_TYPE_T format. This parameter is a 2-byte integer. Values are currently restricted to 1, 2, 3, and 4 as follows:

```
1 = Solid
2 = Dashed
3 = Dotted
4 = Dashed-dotted
```

change
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer. Possible values are the following:

GMR_$SET_VALUE_AND_ENABLE
Stores the attribute value and sets the value as enabled.

GMR_$SET_VALUE_AND_DISABLE
Stores the attribute value but sets the no-change attribute, thus disabling the use of this attribute.

GMR_$NO_VALUE_AND_ENABLE
Ignores the attribute value but enables what was previously set as the attribute value.

GMR_$NO_VALUE_AND_DISABLE
Ignores the attribute value and disables the attribute's use. With this last state, the previous attribute value is preserved.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The default is the line type ID of the source block specified when the block was created (or last copied). The line type ID of the GMR_$DEFAULT_ABLOCK and the

GMR_$NOCHANGE_ABLOCK is GMR_$LINE_TYPE_DEF. This is equivalent to 1 (solid).

Use GMR_$ABLOCK_INQ_LINE_TYPE to return the polyline/multiline type ID and the enabled attribute state of an attribute block.

GMR_$ABLOCK_SET_MARK_COLOR

Sets the polymarker color and the enabled state for the specified attribute block.

FORMAT

```
GMR_$ABLOCK_SET_MARK_COLOR (ablock_id, color, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

color
The polymarker color for this attribute block, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer. Possible values are the following:

GMR_$SET_VALUE_AND_ENABLE
: Stores the attribute value and sets the value as enabled.

GMR_$SET_VALUE_AND_DISABLE
: Stores the attribute value but sets the no-change attribute, thus disabling the use of this attribute.

GMR_$NO_VALUE_AND_ENABLE
: Ignores the attribute value but enables what was previously set as the attribute value.

GMR_$NO_VALUE_AND_DISABLE
: Ignores the attribute value and disables the attribute's use. With this last state, the previous attribute value is preserved.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The default color is the color of the source block specified when the block was created (or last copied). The color of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$MARK_COLOR_DEF. This is equivalent to 1.

Use GMR_$ABLOCK_INQ_MARK_COLOR to return the polymarker color and the enabled attribute state of an attribute block.

GMR_$ABLOCK_SET_MARK_INTEN

Sets the polymarker intensity and the enabled state for the specified attribute block.

FORMAT

```
GMR_$ABLOCK_SET_MARK_INTEN (ablock_id, intensity, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

intensity
The polymarker intensity for this attribute block, in GMR_$INTEN_T format. This parameter is a 4-byte real value in the range [0.0, 1.0] inclusive.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer. Possible values are the following:

GMR_$SET_VALUE_AND_ENABLE
Stores the attribute value and sets the value as enabled.

GMR_$SET_VALUE_AND_DISABLE
Stores the attribute value but sets the no-change attribute, thus disabling the use of this attribute.

GMR_$NO_VALUE_AND_ENABLE
Ignores the attribute value but enables what was previously set as the attribute value.

GMR_$NO_VALUE_AND_DISABLE
Ignores the attribute value and disables the attribute's use. With this last state, the previous attribute value is preserved.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The default intensity for the attribute block is the intensity of the source block specified when the block was created (or last copied). The intensity of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$MARK_INTEN_DEF. This is equivalent to 1.0.

Use GMR_$ABLOCK_INQ_MARK_INTEN to return the polymarker intensity and the enabled attribute state of an attribute block.

GMR_$ABLOCK_SET_MARK_SCALE

Sets the polymarker scale factor and the enabled state for the specified attribute block.

FORMAT

```
GMR_$ABLOCK_SET_MARK_SCALE (ablock_id, scale_factor, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

scale_factor
The polymarker scale factor for this attribute block, in GMR_$MARK_SCALE_T format. This is 4-byte real value. The default scale factor is 1.0.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer. Possible values are the following:

GMR_$SET_VALUE_AND_ENABLE
Stores the attribute value and sets the value as enabled.

GMR_$SET_VALUE_AND_DISABLE
Stores the attribute value but sets the no-change attribute, thus disabling the use of this attribute.

GMR_$NO_VALUE_AND_ENABLE
Ignores the attribute value but enables what was previously set as the attribute value.

GMR_$NO_VALUE_AND_DISABLE
Ignores the attribute value and disables the attribute's use. With this last state, the previous attribute value is preserved.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Conceptually, consider a polymarker in its own coordinate system with its center at the marker's origin. Scale multiplies each coordinate by the scale factor and then truncates to an integer.

Scale factors less than 1.0 are not supported. If you specify a value less than 1.0, the 3D GMR package uses 1.0.

The default scale factor is the polymarker type of the source block specified when the block was created (or last copied). The type of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$MARK_SCALE_DEF. This is equivalent to 1.0.

Use GMR_$ABLOCK_SET_MARK_SCALE to change the polymarker type and the enabled attribute state of an attribute block.

Scaling does not affect the size of marker type 1 (one pixel).

GMR_$ABLOCK_SET_MARK_TYPE

Sets the polymarker type and the enabled state for the specified attribute block.

FORMAT

```
GMR_$ABLOCK_SET_MARK_TYPE (ablock_id, type, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

type
The polymarker type for this attribute block, in GMR_$MARK_TYPE_T format. This is a 2-byte integer in the range [1, 5], inclusive. The default is 1 (one pixel).

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer. Possible values are the following:

GMR_$SET_VALUE_AND_ENABLE
Stores the attribute value and sets the value as enabled.

GMR_$SET_VALUE_AND_DISABLE
Stores the attribute value but sets the no-change attribute, thus disabling the use of this attribute.

GMR_$NO_VALUE_AND_ENABLE
Ignores the attribute value but enables what was previously set as the attribute value.

GMR_$NO_VALUE_AND_DISABLE
Ignores the attribute value and disables the attribute's use. With this last state, the previous attribute value is preserved.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The default is the polymarker type of the source block specified when the block was created (or last copied). The type of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$MARK_TYPE_DEF. This is equivalent to 1.

Use GMR_$ABLOCK_INQ_MARK_TYPE to inquire the polymarker type and the enabled attribute state of an attribute block.

Note that the scale factor does not affect the size of type 1 (one pixel).
The five types of markers are shown below:

Type ID	Marker
1	• (single pixel)
2	+
3	✳
4	O
5	×

The default is type 1 (one pixel).

GMR_$ABLOCK_SET_TEXT_COLOR

Sets the text color and the enabled state for the specified attribute block.

FORMAT

```
GMR_$ABLOCK_SET_TEXT_COLOR (ablock_id, color, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

color
The text color for this attribute block, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer. Possible values are the following:

GMR_$SET_VALUE_AND_ENABLE
Stores the attribute value and sets the value as enabled.

GMR_$SET_VALUE_AND_DISABLE
Stores the attribute value but sets the no-change attribute, thus disabling the use of this attribute.

GMR_$NO_VALUE_AND_ENABLE
Ignores the attribute value but enables what was previously set as the attribute value.

GMR_$NO_VALUE_AND_DISABLE
Ignores the attribute value and disables the attribute's use. With this last state, the previous attribute value is preserved.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The default is the color of the source block specified when the block was created (or last copied). The color of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$TEXT_COLOR_DEF. This is equivalent to 1.

Use GMR_$ABLOCK_INQ_TEXT_COLOR to return the text color and enabled attribute state in an attribute block.

GMR_$ABLOCK_SET_TEXT_EXPANSION

Sets the text expansion and the enabled state for the specified attribute block. Text expansion controls the ratio of height to width of text characters.

FORMAT

```
GMR_$ABLOCK_SET_TEXT_EXPANSION (ablock_id, expansion, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

expansion
The text character expansion for this attribute block, in GMR_$TEXT_EXPANSION_T format. This is a 4-byte real value. This attribute controls the aspect ratio for the font. The default value is 1.0 which preserves the aspect ratio defined in the font.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer. Possible values are the following:

GMR_$SET_VALUE_AND_ENABLE
Stores the attribute value and sets the value as enabled.

GMR_$SET_VALUE_AND_DISABLE
Stores the attribute value but sets the no-change attribute, thus disabling the use of this attribute.

GMR_$NO_VALUE_AND_ENABLE
Ignores the attribute value but enables what was previously set as the attribute value.

GMR_$NO_VALUE_AND_DISABLE
Ignores the attribute value and disables the attribute's use. With this last state, the previous attribute value is preserved.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The default is the expansion of the source block specified when the block was created (or last copied). The expansion of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$TEXT_EXPANSION_DEF. This is equivalent to 1.0.

Use GMR_$ABLOCK_INQ_TEXT_EXPANSION to return the text character expansion and enabled attribute state in an attribute block.

GMR_$ABLOCK_SET_TEXT_HEIGHT

Sets the text height and the enabled state for the specified attribute block. Text height controls the actual size of text characters.

FORMAT

```
GMR_$ABLOCK_SET_TEXT_HEIGHT (ablock_id, height, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

height
The text character height for this attribute block, in GMR_$TEXT_HEIGHT_T format. This is a 4-byte real value in viewing coordinates (same as world coordinates).

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer. Possible values are the following:

GMR_$SET_VALUE_AND_ENABLE
Stores the attribute value and sets the value as enabled.

GMR_$SET_VALUE_AND_DISABLE
Stores the attribute value but sets the no-change attribute, thus disabling the use of this attribute.

GMR_$NO_VALUE_AND_ENABLE
Ignores the attribute value but enables what was previously set as the attribute value.

GMR_$NO_VALUE_AND_DISABLE
Ignores the attribute value and disables the attribute's use. With this last state, the previous attribute value is preserved.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The default is the height of the source block specified when the block was created (or last copied) is returned. The height of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$TEXT_HEIGHT_DEF. This is equivalent to 0.01.

Use GMR_$ABLOCK_INQ_TEXT_HEIGHT to return the text character height and enabled attribute state in an attribute block.

GMR_$ABLOCK_SET_TEXT_INTEN

Sets the text intensity and the enabled state for the specified attribute block.

FORMAT

```
GMR_$ABLOCK_SET_TEXT_INTEN (ablock_id, intensity, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

intensity
The text intensity for this attribute block, in GMR_$INTEN_T format. This is a 4-byte real value in the range [0.0, 1.0], inclusive.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer. Possible values are the following:

GMR_$SET_VALUE_AND_ENABLE
Stores the attribute value and sets the value as enabled.

GMR_$SET_VALUE_AND_DISABLE
Stores the attribute value but sets the no-change attribute, thus disabling the use of this attribute.

GMR_$NO_VALUE_AND_ENABLE
Ignores the attribute value but enables what was previously set as the attribute value.

GMR_$NO_VALUE_AND_DISABLE
Ignores the attribute value and disables the attribute's use. With this last state, the previous attribute value is preserved.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The default is the intensity of the source block specified when the block was created (or last copied). The intensity of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$TEXT_INTEN_DEF. This is equivalent to 1.0.

Use GMR_$ABLOCK_INQ_TEXT_INTEN to return the text intensity and enabled attribute state in an attribute block.

GMR_$ABLOCK_SET_TEXT_PATH

Sets the text path angle and the enabled state for the specified attribute block. Text path determines where the second and subsequent characters in a text string are placed.

FORMAT

```
GMR_$ABLOCK_SET_TEXT_PATH (ablock_id, angle, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

angle
The angle that determines where the second and subsequent characters in a string are placed, in GMR_$TEXT_PATH_T format. This parameter is a 4-byte real value.

An angle of 0.0 radians is to the right of the up vector. Angles greater than 0.0 radians are measured counterclockwise from the 0.0 radian position.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer. Possible values are the following:

GMR_$SET_VALUE_AND_ENABLE
: Stores the attribute value and sets the value as enabled.

GMR_$SET_VALUE_AND_DISABLE
: Stores the attribute value but sets the no-change attribute, thus disabling the use of this attribute.

GMR_$NO_VALUE_AND_ENABLE
: Ignores the attribute value but enables what was previously set as the attribute value.

GMR_$NO_VALUE_AND_DISABLE
: Ignores the attribute value and disables the attribute's use. With this last state, the previous attribute value is preserved.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The default is the path direction of the source block specified when the block was created (or last copied). The path of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$TEXT_PATH_DEF. This is equivalent to 0.0 and places characters to the right of preceding ones.

For convenience, use the following default values:

```
GMR_$TEXT_PATH_RIGHT
GMR_$TEXT_PATH_UP
GMR_$TEXT_PATH_LEFT
GMR_$TEXT_PATH_DOWN
```

Use GMR_$ABLOCK_INQ_TEXT_PATH to return the text character path and enabled attribute state in an attribute block.

GMR_$ABLOCK_SET_TEXT_SLANT

Sets the text slant factor and the enabled state for the specified attribute block. A negative value produces a left slant. A positive value produces a right slant.

FORMAT

```
GMR_$ABLOCK_SET_TEXT_SLANT (ablock_id, slant, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

slant
The amount that the top of the character is shifted, in GMR_$TEXT_SLANT_T format. This parameter is a 4-byte real value.

The amount is determined by multiplying the text attributes for slant, height, and expansion factor. A value in the range [0.0, 1.0] exclusive yields an italics-like character. The default value is GMR_$TEXT_SLANT_DEF which is equivalent to 0.0 (no slant).

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer. Possible values are the following:

GMR_$SET_VALUE_AND_ENABLE
: Stores the attribute value and sets the value as enabled.

GMR_$SET_VALUE_AND_DISABLE
: Stores the attribute value but sets the no-change attribute, thus disabling the use of this attribute.

GMR_$NO_VALUE_AND_ENABLE
: Ignores the attribute value but enables what was previously set as the attribute value.

GMR_$NO_VALUE_AND_DISABLE
: Ignores the attribute value and disables the attribute's use. With this last state, the previous attribute value is preserved.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The default is the slant factor of the source block specified when the block was created (or last copied). The slant of the GMR_$DEFAULT_ABLOCK and the

GMR_$NOCHANGE_ABLOCK is GMR_$TEXT_SLANT_DEF. This is equivalent to 0.0 (no slant).

Use GMR_$ABLOCK_INQ_TEXT_SLANT to return the text slant factor and enabled attribute state in an attribute block.

GMR_$ABLOCK_SET_TEXT_SPACING

Sets the intercharacter spacing and the enabled state for the specified attribute block.

FORMAT

```
GMR_$ABLOCK_SET_TEXT_SPACING (ablock_id, spacing, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

spacing
The intercharacter spacing for this attribute block, in GMR_$TEXT_SPACING_T format. This is 4-byte real value that defines spacing as a fraction of text height.

The default is GMR_$TEXT_SPACING_DEF which is equivalent to 0.0. This places each character next to the preceding character in the character path direction.

For more spacing between characters, make the spacing value positive. To have characters appear to overlay, make the spacing value negative.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer. Possible values are the following:

GMR_$SET_VALUE_AND_ENABLE
Stores the attribute value and sets the value as enabled.

GMR_$SET_VALUE_AND_DISABLE
Stores the attribute value but sets the no-change attribute, thus disabling the use of this attribute.

GMR_$NO_VALUE_AND_ENABLE
Ignores the attribute value but enables what was previously set as the attribute value.

GMR_$NO_VALUE_AND_DISABLE
Ignores the attribute value and disables the attribute's use. With this last state, the previous attribute value is preserved.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The default is the intercharacter spacing of the source block specified when the block was

created (or last copied). The spacing of GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is GMR_$TEXT_SPACING_DEF. This is equivalent to 0.0.

Use GMR_$ABLOCK_INQ_TEXT_SPACING to return the intercharacter spacing and enabled attribute state in an attribute block.

GMR_$ABLOCK_SET_TEXT_UP

Sets the up direction of text on the projection plane and the enabled state for the specified attribute block.

FORMAT

```
GMR_$ABLOCK_SET_TEXT_UP (ablock_id, up_vector, enable_state, status)
```

INPUT PARAMETERS

ablock_id
The identification number of the attribute block, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

up_vector
The up direction of text on the projection plane, in GMR_$TEXT_UP_T format. This parameter is a pair of real values in viewing coordinates (same as world coordinates). Both values cannot be zero.

enable_state
The enabled state of the attribute, in GMR_$CHANGE_STATE_T format. This parameter is a 2-byte integer. Possible values are the following:

GMR_$SET_VALUE_AND_ENABLE
: Stores the attribute value and sets the value as enabled.

GMR_$SET_VALUE_AND_DISABLE
: Stores the attribute value but sets the no-change attribute, thus disabling the use of this attribute.

GMR_$NO_VALUE_AND_ENABLE
: Ignores the attribute value but enables what was previously set as the attribute value.

GMR_$NO_VALUE_AND_DISABLE
: Ignores the attribute value and disables the attribute's use. With this last state, the previous attribute value is preserved.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The default direction is the up vector of the source block specified when the block was created (or last copied). The up vector of the GMR_$DEFAULT_ABLOCK and the GMR_$NOCHANGE_ABLOCK is (GMR_$TEXT_UP_X_DEF, GMR_$TEXT_UP_Y_DEF). This is equivalent to (0.0, 1.0) which is the typical way to display text.

A value of (1.0, 0.0) places characters on their side with the top to the right.

Use GMR_$ABLOCK_INQ_TEXT_UP to return the up direction of text characters and enabled attribute state in an attribute block.

GMR_$ACLASS

Inserts an element into the current open structure. The element selects an attribute class.

FORMAT

```
GMR_$ACLASS (aclass_id, status)
```

INPUT PARAMETERS

aclass_id
The identification number of the attribute class to use, in GMR_$ACLASS_ID_T format. This is 2-byte integer.

The maximum number of attribute classes is 16.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The routine sets the attribute class used for all subsequent elements and structure instances in the current structure. The attribute class is assigned to an attribute block using either GMR_$ABLOCK_ASSIGN_DISPLAY or GMR_$ABLOCK_ASSIGN_VIEWPORT. The viewport binding takes precedence over the display binding.

If no GMR_$ACLASS calls are issued, the default aclass is used. If no GMR_$ABLOCK_ASSIGN_DISPLAY or GMR_$ABLOCK_ASSIGN_VIEWPORT calls are issued, the default ablock (GMR_$DEFAULT_ABLOCK) is assigned to all aclasses.

Use GMR_$INQ_ACLASS to retrieve the aclass ID established by the current (GMR_$ACLASS) element.

GMR_$ADD_NAME_SET

Inserts an element into the current open structure. The element adds names to the current name set.

FORMAT

```
GMR_$ADD_NAME_SET (n_names, name_set, status)
```

INPUT PARAMETERS

n_names
The number of names in the name set. This parameter is a 2 byte integer.

name_set
The list of names to be added, in GMR_$NAME_SET_T format. Each name is in the range [1, GMR_$MAX_NAME_ELEMENT], inclusive.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

A name set is an attribute used to set invisibility and pick eligibility for primitive. Using this method, you can classify elements by name.

At display time, name sets are compared to an invisibility filter and a pick filter. Each filter contains an inclusion set and an exclusion set that determines whether the primitive is eligible for the operation. The filter lists are specified by the following two calls:

```
GMR_$VIEWPORT_SET_INVIS_FILTER
GMR_$VIEWPORT_SET_PICK_FILTER
```

GMR_$REMOVE_NAME_SET removes names from the current name set.

Visibility and Pick Eligibility Criteria
The relationship between the viewport inclusion and exclusion sets and the current name set follows:

```
Ii = Viewport invisibility inclusion set
Ei = Viewport invisibility exclusion set
Ip = Viewport pick inclusion set
Ep = Viewport pick exclusion set
int = set intersection
N = current name set
.EQ. = equals
.NE. = not equal to
```

For an element within a visible structure to be *invisible*, at least one name in the current name set must be in the viewport invisibility inclusion set AND all names in the name set

must be absent from the viewport invisibility exclusion set. This is stated mathematically as follows:

```
Invisible <=> (Ii int N .NE. 0) AND (Ei int N .EQ. 0)
```

For an element within a visible structure to be *visible*, EITHER all names in the current name set must be absent from the viewport invisibility inclusion set OR at least one name in the current name set must be in the viewport exclusion set. This is stated mathematically as follows:

```
Visible   <=> (Ii int N .EQ. 0) OR  (Ei int N .NE. 0)
```

For a display element within a visible, pickable structure to be *pickable*, The above visibility criteria must be met AND at least one name in the current name set must be in the viewport pick inclusion set AND all names in the current name set must be absent from the viewport pick exclusion set. This is stated mathematically as follows:

```
Pickable  <=> [(Ii int N .EQ. 0) OR  (Ei int N .NE. 0)] AND
              (Ip int N .NE. 0) AND (Ep int N .EQ. 0)
```

Notice above that in order to be pickable, an object must meet the visibility criteria as well as the pick eligibility criteria.

Example:

```
floor_1     :=1;
floor_2     :=2;
electrical  :=3;
plumbing    :=4;

name_set[1] := floor_1;
name_set[2] := electrical;
n_names     := 2;
GMR_$ADD_NAME_SET (n_names, name_set, status);
 .
 .          { Add primitives or instanced structures for
 .            first floor electrical.                    }

name_set[1] := electrical;
n_names     := 1;
GMR_$REMOVE_NAME_SET (n_names, name_set, status);

name_set[1] := plumbing;
n_names     := 1;
GMR_$ADD_NAME_SET (n_names, name_set, status);
 .
 .          { Add primitives or instanced structures for
 .            first floor plumbing.                      }
 .

name_set[1] := floor_1;
n_names     := 1;
GMR_$REMOVE_NAME_SET (n_names, name set, status);

name_set[1] := floor_2;
n_names     := 1;
GMR_$ADD_NAME_SET (n_names, name_set, status);
```

```
.
.           { Add primitives or instanced structures for
.             second floor plumbing.                    }
.
```

With the above example, one way to allow only picking of plumbing on the second floor is to set the viewport pick filter with an inclusion list of (plumbing, floor_2) and an exclusion list of (electrical, floor_1).

GMR_$ATTRIBUTE_SOURCE

Sets the attribute source flag for an attribute type to direct (use explicit attribute element) or aclass (use current aclass definition).

FORMAT

```
GMR_$ATTRIBUTE_SOURCE (attribute, source, status)
```

INPUT PARAMETERS

attribute

The attribute to be set, in GMR_$ATTRIBUTE_T format. This parameter is a 2 byte integer. Possible values are:

GMR_$ATTR_LINE_COLOR
: Line color for polylines and multilines.

GMR_$ATTR_LINE_INTEN
: Line intensity for polylines and multilines.

GMR_$ATTR_LINE_TYPE
: Line type for polylines and multilines.

GMR_$ATTR_FILL_COLOR
: Fill color for polygons and meshes.

GMR_$ATTR_FILL_INTEN
: Fill intensity for polygons and meshes.

GMR_$ATTR_MARK_COLOR
: Color for polymarker elements.

GMR_$ATTR_MARK_INTEN
: Intensity for polymarker elements.

GMR_$ATTR_MARK_SCALE
: Scale for polymarker elements.

GMR_$ATTR_MARK_TYPE
: Type for polymarker elements.

GMR_$ATTR_TEXT_COLOR
: Text color.

GMR_$ATTR_TEXT_INTEN
: Text intensity.

GMR_$ATTR_TEXT_HEIGHT
: Text height.

GMR_$ATTR_TEXT_EXPANSION
: Text expansion factor.

GMR_$ATTR_TEXT_SLANT
Text slant factor.

GMR_$ATTR_TEXT_SPACING
Text spacing.

GMR_$ATTR_TEXT_UP
Text up vector.

GMR_$ATTR_TEXT_PATH
Text path angle.

source
The source flag, in GMR_$ATTRIBUTE_SOURCE_T format. This parameter is a 2-byte integer. Possible values are GMR_$ATTRIBUTE_DIRECT and GMR_$ATTRIBUTE_ACLASS.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

At display time, an attribute type is not used unless its source flag has been set. This means that if you insert an explicit attribute element (e.g., GMR_$LINE_COLOR) before a primitive element (e.g., GMR_$F3_POLYLINE), the attribute is used only if the direct source flag is in effect.

At display time, this allows you the flexibility of turning the attribute elements or aclass elements in the metafile on and off.

The default is direct. This means that you can only use an aclass element if you set the source flag to aclass for each type of attribute in the ablock. Likewise, after you set an attribute type to aclass, you can only use an explicit attribute element of that type if you set the flag for that type to direct.

Use GMR_$INQ_ATTRIBUTE_SOURCE to retrieve the source information of the current (GMR_$ATTRIBUTE_SOURCE) element.

GMR_$COLOR_DEFINE_HSV

Updates the section of the color map that corresponds to the input color ID using the hue, saturation, and value color model.

FORMAT

```
GMR_$COLOR_DEFINE_HSV (color_id, low_color, high_color, status)
```

INPUT PARAMETERS

color_id
The color identifier, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer.

low_color
The low color, in GMR_$HSV_COLOR_T format. This parameter is an array of three real values. The first represents hue, the second is saturation, and the third is value.

high_color
The high color, in GMR_$HSV_COLOR_T format. This parameter is an array of three real values as described for low_color.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

In updating the color map, saturation and value must be between 0 and 1. The fractional part of the hue specification determines the hue. The color is linearly interpolated between the two input colors, and the interpolated values are assigned to the range of color map indices.

Hue: Red has a hue of 0, green has a hue of 1/3, and blue has a hue of 2/3. A hue of 1 is also red. For example, if value = 1 and saturation = 1, varying the hue from 2/3 to 1 changes the color from blue to magenta to red.

Saturation: A saturation of 0 gives a gray-scale color, and a saturation of 1 yields the "pure" hue. For example, if hue = 0 and value = 1, then varying the saturation from 0 to 1 changes the color from white to pink to red.

Value: A value of 0 gives black, and a value of 1 gives a bright color. For example, if saturation = 0, then regardless of the hue, varying the value from 0 to 1 changes the color from black to gray to white.

GMR_$COLOR_DEFINE_RGB

Updates the section of the color map that corresponds to the input color ID by specifying the amounts of red, green, and blue.

FORMAT

```
GMR_$COLOR_DEFINE_RGB (color_id, low_color, high_color, status)
```

INPUT PARAMETERS

color_id
The color identifier, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer.

low_color
The low color, in GMR_$RGB_COLOR_T format. This parameter is an array of three real values. The first represents the red contribution, the second is blue, and the third is green.

high_color
The high color, in GMR_$RGB_COLOR_T format. This parameter is an array of three real values as described for low_color.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

In updating the color map, the red, green, and blue contributions are all values between 0.0 and 1.0. The colors are linearly interpolated between the two input colors, and the interpolated values are assigned to the range of color map indices.

GMR_$COLOR_HSV_TO_RGB

Translates an HSV (hue, saturation, value) color specification to a RGB (red, green, blue) color specification.

FORMAT

```
GMR_$COLOR_HSV_TO_RGB (hsv_color, rgb_color, status)
```

INPUT PARAMETERS

hsv_color
The color specification in the HSV model, in GMR_$HSV_COLOR_T format. This parameter is an array of three real values.

OUTPUT PARAMETERS

rgb_color
The color specification in the RGB model, in GMR_$RGB_COLOR_T format. This parameter is an array of three real values.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

See the routines GMR_$COLOR_DEFINE_RGB and GMR_$COLOR_DEFINE_HSV for definitions of these color models.

GMR_$COLOR_INQ_HSV

Returns the color values at the low and high extremes of the range for a color ID.

FORMAT

```
GMR_$COLOR_INQ_HSV (color_id, inq_type, low_color, high_color, status)
```

INPUT PARAMETERS

color_id
Color identifier, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer.

inq_type
Inquiry type, in GMR_$INQ_TYPE_T format. This parameter is a 2-byte integer. Possible values are GMR_$SET and GMR_$REALIZED.

OUTPUT PARAMETERS

low_color
The low color, in GMR_$HSV_COLOR_T format. This parameter is an array of three real values. The first value represents hue, the second is saturation, and the third is value.

high_color
The high color, in GMR_$HSV_COLOR_T format. This parameter is an array of three real values as described for low_color.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

This routine returns the color values at the low and high extremes of the range for a color ID. If the set value is requested, the values specified by using GMR_$COLOR_DEFINE_HSV are returned unless the color ID was not defined using GMR_$DEFINE_HSV, in which case an error status is returned. If the realized value is requested, the actual color value used is translated to HSV.

Gray colors can cause ambiguities as described in the Usage section of GMR_$COLOR_RGB_TO_HSV.

GMR_$COLOR_INQ_MAP

Returns the values stored in the current color map.

FORMAT

```
GMR_$COLOR_INQ_MAP (start_index, count, color_array, status)
```

INPUT PARAMETERS

start_index
The index of the first color in the color map to be returned, in GMR_$L_T format. This parameter is a 4-byte integer.

count
The number of contiguous entries in the color map to be returned, in GMR_$I_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

color_array
The color array, in GMR_$COLOR_VECTOR_T format. This is an array of 4-byte integers. See the Data Types section in Chapter 1 for information on how to build this array.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

You can set Ranges of color map values corresponding to a color ID using GMR_$COLOR_DEFINE_HSV or GMR_$COLOR_DEFINE_RGB. Alternately, you can change the color map directly using GMR_$COLOR_SET_MAP.

GMR_$COLOR_INQ_RANGE

Accepts a color ID and returns the starting color map index and the range of color map indices for the color ID.

FORMAT

```
GMR_$COLOR_INQ_RANGE (color_id, start, range, status)
```

INPUT PARAMETERS

color_id
Color identifier, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

start
The starting value in the color map, GMR_$I_T format. This parameter is a 2-byte integer.

range
The range of contiguous color map indices to associate with the color ID, in GMR_$I_T format. This parameter is a 2-byte integer.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$COLOR_SET_RANGE to associate a color ID with a range of color map indices.

GMR_$COLOR_INQ_RGB

Returns the color values at the low and high extremes of the range for a color ID.

FORMAT

```
GMR_$COLOR_INQ_RGB (color_id, inq_type, low_color, high_color, status)
```

INPUT PARAMETERS

color_id
The color identifier, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer.

inq_type
The inquiry type, in GMR_$INQ_TYPE_T format. This is a 2-byte integer. Possible values are GMR_$SET and GMR_$REALIZED.

OUTPUT PARAMETERS

low_color
The low color, in GMR_$RGB_COLOR_T format. This parameter is an array of three real values.

high_color
The high color, in GMR_$RGB_COLOR_T format. This parameter is an array of three real values.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

If the set value is requested, the values specified by using GMR_$COLOR_DEFINE_RGB are returned. An error status is returned if the color ID was not defined with the RGB color model. If the realized value is requested, the actual color value used is translated to RGB.

GMR_$COLOR_RGB_TO_HSV

Translates an RGB (red, green, blue) color specification to an HSV (hue, saturation, value) color specification.

FORMAT

```
GMR_$COLOR_RGB_TO_HSV (rgb_color, hsv_color, status)
```

INPUT PARAMETERS

rgb_color
The color specification in the RGB model, in GMR_$RGB_COLOR_T format. This parameter is an array of three real values.

OUTPUT PARAMETERS

hsv_color
The color specification in the HSV model, in GMR_$HSV_COLOR_T format. This parameter is an array of three real values.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

See the routines GMR_$COLOR_DEFINE_RGB and GMR_$COLOR_DEFINE_HSV for definitions of these color models.

This utility translates an RGB color specification to an HSV color specification. This translation is ambiguous for colors that are shades of gray from black to white (i.e., colors with a saturation of 0) as these colors are independent of hue. In these cases, the returned specification always has hue of 0 (red).

GMR_$COLOR_SET_MAP

Updates the current color map.

FORMAT

```
GMR_$COLOR_SET_MAP (start_index, range, color_array, status)
```

INPUT PARAMETERS

start_index
The index of the first color in the color map to be set, in GMR_$L_T format. This parameter is a 2-byte integer.

range
The number of contiguous color map entries to set using the color_array, in GMR_$I_T format. This parameter is a 2-byte integer.

color_array
Color array, in GMR_$COLOR_VECTOR_T format. This is an array of 4-byte integers. See the data types section in Chapter 1 for information on how to build the color array.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The actual transfer of the color map to the display device happens immediately.

This call provides access to the color map directly. In contrast, GMR_$COLOR_DEFINE_HSV or GMR_$COLOR_DEFINE_RGB perform interpolations of a given color in a given range.

GMR_$COLOR_SET_RANGE

Accepts a color ID number, a start index in the color map, and a range that is the number of contiguous color map indices to associate with the color ID.

FORMAT

```
GMR_$COLOR_SET_RANGE (color_id, start, range, status)
```

INPUT PARAMETERS

color_id
Color identifier, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer.

start
The starting value in the color map, in GMR_$I_T format. This parameter is a 2-byte integer.

range
The range of contiguous color map indices to associate with the color ID, in GMR_$I_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Because this is a display-time routine, reallocation of the colors does not require editing of the metafile. This allows application programs to trade off having many colors with coarse intensity interpolation against having few colors with very smooth intensity interpolation.

To set the colors for a given range, use GMR_$COLOR_DEFINE_HSV or GMR_$COLOR_DEFINE_RGB.

GMR_$COORD_DEVICE_TO_LDC

Converts device coordinates to logical device coordinates.

FORMAT

```
GMR_$COORD_DEVICE_TO_LDC (device_coord, ldc_coord, status)
```

INPUT PARAMETERS

device_coord
The device coordinates, in GMR_$F3_POINT_T. This parameter is an array of three real values that represents x-, y-, and z-coordinates.

The x- and y- device coordinates determine pixel coordinates on the screen or a main bit map rounded to the nearest integer. Currently, z is unused.

OUTPUT PARAMETERS

ldc_coord
The logical device coordinates, in GMR_$F3_POINT_T. This parameter is an array of three real values that represents x-, y-, and z-coordinates.

The transformation from device coordinates to logical device coordinates is determined by the limits that can be retrieved with the GMR_$COORD_INQ_DEVICE_LIMITS and GMR_$COORD_INQ_LDC_LIMITS.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Logical device and device coordinates have their origin in the lower left-hand corner with y increasing up and x increasing to the right. This is different from the DOMAIN Graphics Primitives package that has y in the top left-hand corner, increasing downward.

GMR_$COORD_INQ_DEVICE_LIMITS

Returns the device coordinates to which the logical device limits are mapped.

FORMAT

```
GMR_$COORD_INQ_DEVICE_LIMITS (device_limits, status)
```

OUTPUT PARAMETERS

device_limits
The current limits of device space, in GMR_$F3_LIMITS_T format. This parameter is an array of six non-negative, real values. Currently, the z limits are unused.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$COORD_SET_DEVICE_LIMITS to alter the limits of available device space.

GMR_$COORD_INQ_LDC_LIMITS

Returns the current logical device coordinate limits.

FORMAT

```
GMR_$COORD_INQ_LDC_LIMITS (ldc_limits, status)
```

OUTPUT PARAMETERS

ldc_limits
Current bounds of logical device coordinate space, in GMR_$F3_LIMITS_T format. This parameter is an array of six real values that specify xmin, xmax, ymin, ymax, zmin, and zmax.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$COORD_SET_LDC_LIMITS to alter the default ldc limits.

GMR_$COORD_INQ_MAX_DEVICE

Returns the maximum range of the device coordinates.

FORMAT

```
GMR_$COORD_INQ_MAX_DEVICE (max_device, status)
```

OUTPUT PARAMETERS

max_device
Current maximal limits of the device, in GMR_$F3_LIMITS_T format. This parameter is an array of six real values. Currently, the z limits are unused.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The device limits cannot be set larger than the values returned by max_device.

GMR_$COORD_INQ_WORK_PLANE

Returns a point in world coordinates and a normal vector that together define the work plane associated with a viewport.

FORMAT

```
GMR_$COORD_INQ_WORK_PLANE (viewport_id, point, normal, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

point
A point on the work plane, in GMR_$F3_POINT_T format. The point is in world coordinates.

normal
A vector that is orthogonal to the current work plane in the given viewport, in GMR_$F3_VECTOR_T format. This parameter is an array of three real values.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Each viewport has a work plane associated with it. The work plane provides a means of mapping logical device coordinates into world coordinates. This is useful when receiving cursor input.

Only one work plane per viewport can be active at a time but you can change work planes frequently.

Use GMR_$COORD_SET_WORK_PLANE to establish a work plane for a viewport.

GMR_$COORD_LDC_TO_DEVICE

Converts logical device coordinates to device coordinates.

FORMAT

```
$GMR_$COORD_LDC_TO_DEVICE (ldc_coord, device_coord, status)
```

INPUT PARAMETERS

ldc_coord
The logical device coordinates, in GMR_$F3_POINT_T. This parameter is an array of three real values that represent x-, y-, and z-coordinates.

The transformation from device coordinates to logical device coordinates is determined by the limits that can be retrieved with the GMR_$COORD_INQ_DEVICE_LIMITS and GMR_$COORD_INQ_LDC_LIMITS.

OUTPUT PARAMETERS

device_coord
The device coordinates, in GMR_$F3_POINT_T. This parameter is an array of three real values that represents x-, y-, and z-coordinates.

The x- and y-device coordinates determine pixel coordinates on the screen rounded to the nearest integer. Currently, z is unused.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Logical device and device coordinates have their origin in the lower left-hand corner with y increasing up and x increasing to the right. This is different from the DOMAIN Graphics Primitives package that has y in the top left-hand corner, increasing downward.

GMR_$COORD_LDC_TO_WORK_PLANE

Maps a coordinate in logical device space onto the work plane of the specified viewport. The result is a point in world coordinates.

FORMAT

```
GMR_$COORD_LDC_TO_WORK_PLANE (viewport_id, ldc_coord, plane_coord, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

ldc_coord
A point in logical device coordinates, in GMR_$F3_POINT_T format. This parameter is an array of three real values that represents x-, y-, and z-coordinates.

OUTPUT PARAMETERS

plane_coord
The world coordinates of the point given in logical device coordinates, in GMR_$F3_POINT_T format.

The returned world coordinates are on the work plane specified by GMR_$COORD_SET_WORK_PLANE. Each viewport defines its own mapping from logical device coordinates to world coordinates by way of its current viewing parameters and work plane.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

If the logical device coordinates are outside the viewport, the mapping is performed and GMR_$POINT_OUTSIDE_VIEWPORT is returned in the status parameter.

Mapping is still performed if the work plane is behind the reference point. An error occurs only if the work plane is parallel to the line from the reference point to the input position (line of sight parallel to the work plane).

See the Usage section of GMR_$COORD_LDC_TO_WORLD for a comparison of GMR_$COORD_LDC_TO_WORK_PLANE and GMR_$COORD_LDC_TO_WORLD.

GMR_$COORD_LDC_TO_WORLD

Maps a point in 3D logical device coordinates into world coordinates via the viewing parameters associated with the specified viewport.

FORMAT

```
GMR_$COORD_LDC_TO_WORLD (viewport_id, ldc_coord, world_coord, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

ldc_coord
A point in logical device coordinates, in GMR_$F3_POINT_T format. This parameter is an array of three real values that represents x-, y-, and z-coordinates.

OUTPUT PARAMETERS

world_coord
The world coordinates of the point given in logical device coordinates, in GMR_$F3_POINT_T format.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

This call carries out the one-to-one mapping between the 3D viewport limits in LDC space and the 3D viewing volume in world coordinates defined by the viewing parameters (or the 4x3 normalizing matrix if it was specified directly via GMR_$VIEW_SET_TRANSFORM).

Both GMR_$COORD_LDC_TO_WORK_PLANE and GMR_$COORD_LDC_TO_WORLD map LDC coordinates to world coordinates. The differences between the two routines are as follows:

- GMR_$COORD_LDC_TO_WORK_PLANE
  The result is independent of the z-coordinate of the LDC point. The point is projected onto the work plane.

- GMR_$COORD_LDC_TO_WORLD
  The z-coordinate is affected by the z-range of the viewport. The z-range of the viewport is mapped to the distance between the hither and yon planes in world coordinates. This defines a one-to-one mapping between the 3D viewport volume in LDC coordinates and the 3D view volume in world coordinates.

GMR_$COORD_SET_DEVICE_LIMITS

Specifies the limits of device space.

FORMAT

```
GMR_$COORD_SET_DEVICE_LIMITS (device_limits, status)
```

INPUT PARAMETERS

device_limits

The new limits of device space, in GMR_$F3_LIMITS_T format. This parameter is an array of six real values. The first two values give the limits in x; the second two values give the limits in y; the last two values give the limits in z.

OUTPUT PARAMETERS

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The limits set by this call are constrained to be within the maximum limits for the device, which can be inquired with GMR_$COORD_INQ_MAX_DEVICE. Currently, the z limits are unused. In direct mode, values are based on the Display Manager window size.

The device coordinate system has its origin in the lower left-hand corner with y increasing up and x increasing to the right.

GMR_$COORD_SET_LDC_LIMITS

Specifies the limits of logical device coordinate space.

FORMAT

```
GMR_$COORD_SET_LDC_LIMITS (ldc_limits, status)
```

INPUT PARAMETERS

ldc_limits
The new limits of logical device coordinate space, in GMR_$F3_LIMITS_T format. This parameter is an array of six real values that specify xmin, xmax, ymin, ymax, zmin, and zmax.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The default limits are the following:

```
[0,1] X [0,1] X [0,1]
```

This routine allows the application to define a convenient coordinate system for the device that is independent of actual bitmap dimensions.

GMR_$COORD_SET_WORK_PLANE

Establishes a plane for mapping between logical device coordinates and world coordinates in a specified viewport.

FORMAT

```
GMR_$COORD_SET_WORK_PLANE (viewport_id, point, normal, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport for which the work plane is to be established, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

point
A point on the work plane, in GMR_$F3_POINT_T format. The point is in world coordinates. The default is (0.0, 0.0, 0.0).

normal
A vector that is orthogonal to the desired work plane, in GMR_$F3_VECTOR_T format. Any non-zero vector is valid. The default is (0.0, 0.0, 1.0).

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Each viewport may have a different work plane associated with it. Only one work plane per viewport can be active at a time, but you can change work planes frequently.

GMR_$COORD_WORLD_TO_LDC

Returns the logical device coordinates of a point specified in world coordinates.

FORMAT

```
GMR_$COORD_WORLD TO_LDC (viewport_id, world_coord, ldc_coord, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

world_coord
A point in world coordinates, in GMR_$F3_POINT_T format. This is an array of three real values that represents x-, y-, and z-coordinates.

OUTPUT PARAMETERS

ldc_coord
The logical device coordinates of the point given in world coordinates, in GMR_$F3_POINT_T.

Each viewport defines its own mapping from world coordinates to logical device coordinates by way of its current viewing parameters.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The work plane is not used for this mapping.

GMR_$CURSOR_INQ_ACTIVE

Returns the status of the cursor: displayed or not displayed.

FORMAT

```
GMR_$CURSOR_INQ_ACTIVE (active, status)
```

OUTPUT PARAMETERS

active
A Boolean (logical) value that indicates whether or not the cursor is displayed. The parameter is set to true if the cursor is displayed; it is set to false if the cursor is not displayed.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$CURSOR_SET_ACTIVE to change the display status of the cursor.

Use GMR_$CURSOR_SET_PATTERN to change the pattern of the cursor.

Use GMR_$CURSOR_SET_POSITION to change the position of the cursor.

GMR_$CURSOR_INQ_PATTERN

Returns the type, pattern, and offset of the cursor.

FORMAT

```
GMR_$CURSOR_INQ_PATTERN (style, pattern_size, pattern, offset, status)
```

OUTPUT PARAMETERS

style

The cursor style, in GMR_$CURSOR_STYLE_T format. his parameter is a 2-byte integer. Currently, the only valid value is GMR_$BITMAP.

pattern_size

The size of the cursor pattern, in GMR_$I2_POINT_T format. This parameter is a two-element array of 2-byte integers. Currently, neither coordinate size may exceed 16. See the Data Types section for more information.

pattern

The cursor pattern, in UNIV GMR_$CURSOR_PATTERN_T format. This parameter is an array of (pattern_size.y) 2-byte integers. The length of the array is determined by the y value of pattern_size.

offset

The offset from the pixel at the upper left of the cursor to the pixel at the origin of the cursor, in GMR_$I2_POINT_T format. This parameter is a two-element array of 2-byte integers in the range [(0,0), (15,15)], inclusive.

When the cursor is moved using GMR_$CURSOR_SET_POSITION, the pixel that is the cursor's origin is placed at the specified location.

The first element (x) indicates the number of cursor pixels that will be displayed to the left of the specified cursor location. The second element (y) indicates the number of cursor lines that will be displayed above the specified cursor location.

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$CURSOR_SET_PATTERN to change the pattern of the cursor.

Use GMR_$CURSOR_SET_ACTIVE to change the display status of the cursor.

Use GMR_$CURSOR_SET_POSITION to change the position of the cursor.

GMR_$CURSOR_INQ_POSITION

Returns the position of the cursor.

FORMAT

```
GMR_$CURSOR_INQ_POSITION (position, status)
```

OUTPUT PARAMETERS

position
The cursor position in logical device coordinates, in GMR_$F3_POINT_T format. This parameter is a three-element array of real values. See the Data Types section for more information.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$CURSOR_SET_POSITION to change the position of the cursor.

Use GMR_$CURSOR_SET_PATTERN to change the pattern of the cursor.

Use GMR_$CURSOR_SET_ACTIVE to change the display status of the cursor.

GMR_$CURSOR_SET_ACTIVE

Specifies whether or not the cursor will be displayed.

FORMAT

```
GMR_$CURSOR_SET_ACTIVE (active, status)
```

INPUT PARAMETERS

active

A Boolean (logical) value that indicates whether or not the cursor will be displayed. The parameter is set to true if the cursor will be displayed; it is set to false if the cursor will not be displayed.

The default value for active is false.

OUTPUT PARAMETERS

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$CURSOR_INQ_ACTIVE to retrieve the display status of the cursor.

Refer to Example 2 under GMR_$CURSOR_SET_PATTERN.

GMR_$CURSOR_SET_PATTERN

Specifies a cursor pattern, type, and offset (origin).

FORMAT

```
GMR_$CURSOR_SET_PATTERN (style, pattern_size, pattern, offset, status)
```

INPUT PARAMETERS

style
The cursor style, in GMR_$CURSOR_STYLE_T format. Currently, the only valid value is GMR_$BITMAP.

pattern_size
The size of the cursor pattern, in GMR_$I2_POINT_T format. This parameter is a two-element array of 2-byte integers. Currently, neither coordinate size may exceed 16. See the Data Types section for more information.

pattern
The cursor pattern, in UNIV GMR_$CURSOR_PATTERN_T format. This parameter is an array of (pattern_size.y) 2-byte integers. The length of the array is determined by the y value of pattern_size.

offset
The offset from the pixel at the upper left of the cursor to the pixel at the origin of the cursor, in GMR_$I2_POINT_T format. This parameter is a two-element array of 2-byte integers in the range [(0,0), (15,15)], inclusive.

When the cursor is moved using GMR_$CURSOR_SET_POSITION, the pixel that is the cursor's origin is placed at the specified location.

The first element (x) indicates the number of cursor pixel columns that will be displayed to the left of the specified cursor location. The second element (y) indicates the number of cursor pixel rows that will be displayed above the specified cursor location.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The default value is the standard Display Manager cursor pattern.

Use GMR_$CURSOR_INQ_PATTERN to retrieve the current pattern of the cursor.

You must place a cursor pattern smaller than 16x16 in the high-order bits of the first words of the pattern.

Example 1

```
VAR
{ A cursor pattern smaller than 16x16
  starts in the high order bits, and starts
  in word 1 of the array. }

cursor_pattern1 : gmr_$cursor_pattern_t
                := [16#8080,16#4100,16#2200,16#1400,
                    16#800,16#1400,16#2200,16#4100,16#8080];
cursor_size : gmr_$i2_point_t := [9,9];
cursor_origin : gmr_$i2_point_t := [4,4];
 .
 .
 .
gmr_$cursor_set_pattern(gmr_$bitmap,cursor_size,
                        cursor_pattern1,cursor_origin, status);
```

Example 2

```
VAR
 { Cursor pattern info: }

cursor_pos  : gmr_$f3_point_t       := [0.80, 0.40, 0.00];
cur_style   : gmr_$cursor_style_t   := gmr_$bitmap;
cur_size    : gmr_$i2_point_t       := [16, 16];
cur_offset  : gmr_$i2_point_t       := [8, 8];
cur_pattern : gmr_$cursor_pattern_t := [2#0000000110000000,
                                        2#0000000110000000,
                                        2#0000000110000000,
                                        2#0000000110000000,
                                        2#0000000110000000,
                                        2#0000000110000000,
                                        2#0000000110000000,
                                        2#1111111111111111,
                                        2#1111111111111111,
                                        2#0000000110000000,
                                        2#0000000110000000,
                                        2#0000000110000000,
                                        2#0000000110000000,
                                        2#0000000110000000,
                                        2#0000000110000000,
                                        2#0000000110000000];

PROCEDURE set_cursor_and_init_input;

 BEGIN
 gmr_$input_enable(gmr_$keystroke, [CHR(0)..CHR(127)], status);
 gmr_$input_enable(gmr_$locator, [], status);
 gmr_$input_enable(gmr_$buttons, [CHR(0)..CHR(127)], status);
 gmr_$cursor_set_position(cursor_pos, status);
 gmr_$cursor_set_pattern(cur_style, cur_size, cur_pattern, cur_offset,
                         status);
 gmr_$cursor_set_active(TRUE, status);
 END;
```

GMR_$CURSOR_SET_POSITION

Moves the cursor on the screen.

FORMAT

```
GMR_$CURSOR_SET_POSITION (position, status)
```

INPUT PARAMETERS

position
The new cursor position in logical device coordinates, in GMR_$F3_POINT_T format. This parameter is a three-element array of real values. See the Data Types section for more information.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$CURSOR_INQ_POSITION to retrieve the current position of the cursor.

See Example 2 under GMR_$CURSOR_SET_PATTERN.

GMR_$DBUFF_INQ_MODE

Returns the current mode, which is either single- or double-buffer mode.

FORMAT

```
GMR_$DBUFF_INQ_MODE (buffer_mode, status)
```

OUTPUT PARAMETERS

buffer_mode
Buffer mode, in GMR_$BUFFER_MODE_T format. Specify one of the following predefined values: GMR_$SINGLE_BUFFER or GMR_$DOUBLE_BUFFER. This is a 2-byte integer.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$DBUFF_SET_MODE to toggle between single- and double-buffering.

GMR_$DBUFF_INQ_SELECT_BUFFER

Returns the number of the buffer that was last selected by GMR_$DBUFF_SET_SELECT_BUFFER for the specified viewport.

FORMAT

```
GMR_$DBUFF_INQ_SELECT_BUFFER (viewport_id, buffer_number, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

buffer_number
The buffer_number, in GMR_$BUFFER_T format. Specify one of the following predefined values: GMR_$1ST_BUFFER or GMR_$2ND_BUFFER. This parameter is a 2-byte integer.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

See GMR_$DBUFF_SET_SELECT_BUFFER for additional information.

GMR_$DBUFF_SET_DISPLAY_BUFFER

Displays the specified buffer, which is either buffer 1 or buffer 2, in a specified viewport.

FORMAT

```
GMR_$DBUFF_SET_DISPLAY_BUFFER (buffer_number, viewport_id, status)
```

INPUT PARAMETERS

buffer_number
Buffer number, in GMR_$BUFFER_T format. Specify one of the following predefined values: GMR_$1ST_BUFFER or GMR_$2ND_BUFFER. This parameter is a 2-byte integer.

viewport_id
The ID of the viewport in which the buffer is to be displayed, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

This call is ignored if the current mode is single buffering.

This call is typically used with GMR_$DBUFF_SET_SELECT_BUFFER.

GMR_$DBUFF_SET_MODE

Sets the current mode to single- or double-buffer mode.

FORMAT

```
GMR_$DBUFF_SET_MODE (buffer_mode, status)
```

INPUT PARAMETERS

buffer_mode
Buffer mode, in GMR_$BUFFER_MODE_T format. Specify one of the following predefined values: GMR_$SINGLE_BUFFER or GMR_$DOUBLE_BUFFER. This is a 2-byte integer.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Subsequent calls to modify the color range table or color map apply only to the current buffering mode.

GMR_$DBUFF_SET_SELECT_BUFFER

Indicates which buffer is to be updated.

FORMAT

```
GMR_$DBUFF_SET_SELECT_BUFFER (buffer_number, viewport_id, status)
```

INPUT PARAMETERS

buffer_number
The buffer_number, in GMR_$BUFFER_T format. Specify one of the following predefined values: GMR_$1ST_BUFFER or GMR_$2ND_BUFFER. This parameter is a 2-byte integer.

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

In typical double buffering applications, the buffer to be updated is not the buffer currently displayed. This call is ignored if the current mode is single buffering.

The viewport ID is used to automatically refresh the viewport in the same buffer after a window grow or move operation.

This call is typically used with GMR_$DBUFF_SET_DISPLAY_BUFFER.

GMR_$DISPLAY_CLEAR_BG

Clears the background of the display to its current color setting.

FORMAT

```
GMR_$DISPLAY_CLEAR_BG (status)
```

OUTPUT PARAMETERS

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The background is cleared up to the device limits. Use GMR_$COORD_SET_DEVICE_LIMITS to change the device limits.

In direct mode, the default color is the Display Manager window color. In borrow mode, it is color ID 0. Use GMR_$DISPLAY_SET_BG_COLOR to override these defaults.

GMR_$DISPLAY_INQ_BG_COLOR

Returns the current background color and intensity of the display.

FORMAT

```
GMR_$DISPLAY_INQ_BG_COLOR (bg_type, color_id, intensity, status)
```

OUTPUT PARAMETERS

bg_type
The background color type, in GMR_$BG_COLOR_T format. Specify one of the following predefined values: GMR_$DM_WINDOW_BACKGROUND or GMR_$COLOR_AND_INTEN. This is a 2-byte integer.

color_id
The color identifier, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer between 0 and GMR_$MAX_COLOR_ID.

intensity
The intensity used within the range specified by the color ID, in GMR_$INTEN_T format. This is a 4-byte real value in the range [0.0, 1.0], inclusive.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$VIEWPORT_INQ_BG_COLOR to retrieve the background color and intensity of individual viewports.

USE GMR_$DISPLAY_SET_BG_COLOR to set the background color and intensity for the display.

GMR_$DISPLAY_REFRESH

Redisplays all viewports that have a refresh state of GMR_$REFRESH_WAIT, GMR_$REFRESH_UPDATE, or GMR_$REFRESH_PARTIAL.

FORMAT

```
GMR_$DISPLAY_REFRESH (status)
```

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Viewports that are in the GMR_$REFRESH_INHIBIT refresh state are not redisplayed.

Use GMR_$VIEWPORT_SET_REFRESH_STATE to set the refresh state for a viewport.

The default refresh state for all defined viewports is GMR_$REFRESH_WAIT.

GMR_$DISPLAY_SET_BG_COLOR

Set the background color and intensity for the display.

FORMAT

```
GMR_$DISPLAY_SET_BG_COLOR (bg_type, color_id, intensity, status)
```

INPUT PARAMETERS

bg_type
The background color type, in GMR_$BG_COLOR_T format. Specify one of the following predefined values: GMR_$DM_WINDOW_BACKGROUND or GMR_$COLOR_AND_INTEN. This is a 2-byte integer.

If you use GMR_$DM_WINDOW_BACKGROUND, then color_id and intensity are not needed and are ignored.

If you use GMR_$COLOR_AND_INTEN, then you must also enter values for color_id, and intensity (using the following two arguments).

color_id
The color identifier, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer in the range [0, GMR_$MAX_COLOR_ID], inclusive.

intensity
The intensity used within the range specified by the color ID, in GMR_$INTEN_T format. This is a 4-byte real value in the range [0.0, 1.0], inclusive.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use:

```
bg_type = GMR_$DM_WINDOW_BACKGROUND without color_id and intensity

or

bg_type = GMR_$COLOR_AND_INTEN with color_id and intensity
```

Use GMR_$VIEWPORT_SET_BG_COLOR to set the background color of individual viewports.

GMR_$DM_REFRESH_ENTRY

Specifies a user-defined routine to be called when the display is refreshed as a result of a Display Manager refresh window or <POP> command.

FORMAT

```
GMR_$DM_$REFRESH_ENTRY (refresh_procedure_ptr, status)
```

INPUT PARAMETERS

refresh_procedure_ptr
Entry point for the application-supplied procedure to refresh the display, in GMR_$REFRESH_PTR_T format. This parameter is a pointer-to-procedure (see Usage).

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

This call allows you to control the way that the Display Manager refreshes the screen. For example, you can change the action that happens when a user performs a window-grow or move operation.

Some uses of this call are the following:

- After a window-grow operation, keep the object the same size but let the viewport grow, thus showing more of a large object.
- Refresh overlapping viewports in a particular order.
- Change the logical device coordinate range.
- Change the device coordinate range.
- Clear the background.
- Invoke GMR_$DISPLAY_REFRESH to redisplay all viewports.

The pointer-to-procedure data type is an extension of the Pascal language. See the *DOMAIN Pascal Language Reference* for an explanation of this extension.

This routine requires you to pass procedure pointers. To do so in FORTRAN programs, use the following technique. First declare the subroutines to be passed as EXTERNAL. Then pass their names using the IADDR function to simulate the Pascal pointer mechanism.

For example:

```
EXTERNAL REFRESH_WINDOW
.
.
.
CALL GMR_$DM_REFRESH_ENTRY (IADDR(REFRESH_WINDOW),STATUS)
```

In FORTRAN, use 0 (not NIL) to indicate a zero value.

Successive calls to GMR_$DM_REFRESH_ENTRY override previously defined entry points.

GMR_$REFRESH_PTR_T must have a specific call sequence as described below:

```
GMR_$REFRESH_PTR_T(unobscured, pos_change, old_device_limts, old_max_device)
```

INPUT PARAMETERS

unobscured
When false, this Boolean value indicates that the window is obscured.

pos_change
When true, this Boolean value indicates that the window has moved or grown since the display was released.

old_device_limits
The device limits set when the GMR package was initialized, in GMR_$F3_LIMITS_T format. You can change the default limits. To find the values, use GMR_$COORD_INQ_DEVICE_LIMITS.

old_max_device
These are the maximum device limits that you are allowed to use. These limits depend on how the 3D GMR package was initialized. For example, in direct mode, old_max_device corresponds to the Display Manager window bounds. You must stay within these bounds when setting device limits. You can change the default bounds also. To find the values use GMR_$COORD_INQ_MAX_DEVICE.

The old_max_device argument must be in GMR_$F3_LIMITS_T format.

GMR_$DYN_MODE_INQ_DRAW_METHOD

Returns the type of dynamic drawing method that is enabled for dynamic mode drawing.

FORMAT

```
GMR_$DYN_MODE_INQ_DRAW_METHOD( draw_method, status )
```

OUTPUT PARAMETERS

draw_method
The redraw method, in GMR_$DYNAMIC_DRAW_METHOD_T format. This parameter is a 2-byte integer. Specify be one of the following predefined values:

GMR_$DYN_METHOD_REDRAW
Each subsequent redraw operation erases the enabled element to the background color and then redraws the element in the new position.

GMR_$DYN_METHOD_XOR
Specifies an XOR raster operation.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$DYN_MODE_SET_DRAW_METHOD to set the draw method.

The default is GMR_$DYN_METHOD_REDRAW.

GMR_$DYN_MODE_INQ_ENABLE

Returns whether a dynamic mode is enabled and, if so, identifies the path, path depth, and path order.

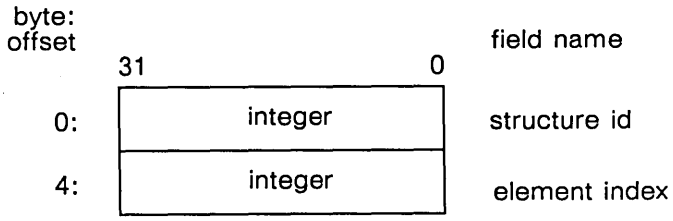
FORMAT

```
GMR_$DYN_MODE_INQ_ENABLE (enabled, dyn_instance_path, dyn_path_depth,
                          dyn_path_order, status)
```

OUTPUT PARAMETERS

enabled
A Boolean value.

dyn_instance_path
The path to the element that will be dynamically changed. This is a path consisting of structure IDs and element indices that uniquely defines one instance of an element in a structure hierarchy. It is an array of 4-byte integers of size GMR_$MAX_INSTANCE_DEPTH, in GMR_$INSTANCE_PATH_T format. The following diagram illustrates one element:




```
Field Description:

structure ID  - The structure ID of the element in this particular
                level of the path.

element index - The position of the element within the structure.
                The index of the first element is 1.
```

dyn_path_depth
A 4-byte integer that identifies the number of levels in the path. This indicates the depth of nesting to the element or subtree to be affected. It is less than or equal to GMR_$MAX_INSTANCE_DEPTH.

dyn_path_order
A 2-byte integer that indicates how the path is interpreted or returned. In GMR_$INSTANCE_PATH_ORDER_T format. One of the following values:

GMR_$TOP_FIRST
: Interpret the path top down (chosen element last).

GMR_$BOTTOM_FIRST
Interpret the path bottom-up (chosen element first).

status
Completion status, in STATUS_$T format. This data type is 4 bytes long. See the Data Types section for more information.

USAGE

The dyn_path_depth points to the specific element to be redrawn. If the element is an instance, the entire instance tree is redrawn.

If enabled is false, the rest of the parameters are not set.

GMR_$DYN_MODE_SET_DRAW_METHOD

Identifies the type of redraw method that is used when dynamic mode is enabled.

FORMAT

```
GMR_$DYN_MODE_SET_DRAW_METHOD (draw_method, status)
```

INPUT PARAMETERS

draw_method

The redraw method, in GMR_$DYNAMIC_DRAW_METHOD_T format. his parameter is a 2-byte integer. Specify one of the following predefined values:

GMR_$DYN_METHOD_REDRAW
: Each subsequent redraw operation erases the enabled element to the background color and then draws the element in the new position.

GMR_$DYN_METHOD_XOR
: Specifies an XOR raster operation.

OUTPUT PARAMETERS

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

With the redraw method, the dynamically changed elements remain correctly drawn as they move, but might partially erase background elements. In the XOR method, the background is preserved but the redrawn elements may have pixels turned off when overlapped with other geometry.

The first bit plane is used for the XOR method. A bit plane is a one-bit-deep layer of the available bitmap.

The default is GMR_$DYN_METHOD_REDRAW.

GMR_$DYN_MODE_SET_ENABLE

Turns the dynamic mode on and off for viewports that are set for partial refresh.

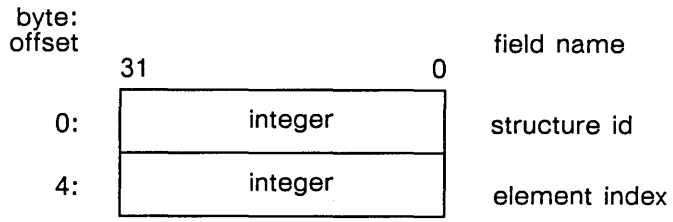
FORMAT

```
GMR_$DYN_MODE_SET_ENABLE (enabled, dyn_instance_path, dyn_path_depth,
                          dyn_path_order, status)
```

INPUT PARAMETERS

enabled
A Boolean value. TRUE enables the dynamic mode.

dyn_instance_path
The path to the element that will be dynamically changed. This is a path that uniquely defines one instance of an element in a structure hierarchy. The path consists of structure IDs and element indices. It is an array of 4-byte integers of size GMR_$MAX_INSTANCE_DEPTH, in GMR_$INSTANCE_PATH_T format. The following diagram illustrates one element:




```
Field Description:

structure ID  - The structure ID of the element in this particular
                level of the path.

element index - The position of the element within the structure.
                The index of the first element is 1.
```

dyn_path_depth
A 4-byte integer that identifies the number of levels in the path. This indicates the depth of nesting to the element to be affected and is less than or equal to GMR_$MAX_INSTANCE_DEPTH. If the element is an instance, then a subtree is affected.

dyn_path_order
A 2-byte integer that indicates how the path is interpreted or returned. In GMR_$INSTANCE_PATH_ORDER_T format. One of the following values:

GMR_$TOP_FIRST
Interpret the path top-down (chosen element last).

GMR_$BOTTOM_FIRST
Interpret the path bottom-up (chosen element first).

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This data type is 4 bytes long. See the Data Types section for more information.

USAGE

If "enabled" is FALSE, then all other arguments are ignored.

If "enabled" is TRUE, any viewport with a refresh state of GMR_$REFRESH_UPDATE is demoted to GMR_$REFRESH_WAIT. Any viewport with partial refresh state (GMR_$REFRESH_PARTIAL) now reflects the dynamic mode if the path is contained in the structure asigned to the viewport by GMR_$VIEWPORT_SET_STRUCTURE.

The dynamic mode allows the user to update an item on the screen dynamically without changing the metafile. It is for fast redraw operations and will only work on one element. The element is identified by dyn_instance_path. Dyn_instance_path is a list of structure IDs and element indices that leads to a particular instance of an element of a metafile.

When dynamic mode is enabled, normal refresh is disabled until another call to GMR_$DYN_MODE_SET_ENABLE sets "enabled" to FALSE.

Use the dyn_instance_path and dyn_path_depth to specify which element is to be dynamically redrawn. The dyn_path_depth points to the specific element to be redrawn. If the element is an instance, the entire instance tree is redrawn.

Editing is locked while dynamic mode is enabled. This means that opening structures, setting element indices, and inserting or replacing elements other than the dynamic type are not allowed.

GMR_$ELEMENT_DELETE

Deletes the current element.

FORMAT

```
GMR_$ELEMENT_DELETE (status)
```

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

When an element is deleted, there are two possible situations:

1. There are more elements after the deleted element. In this case, the next element becomes the current element, and the element index remains unchanged.

2. The deleted element was the last element in the structure. In this case, the previous element (if any) becomes the current element, and the element index is decremented.

If there is only one element in a structure and you delete it, then the structure is empty and there is no current element. The element index is set to 0.

Use GMR_$ELEMENT_SET_INDEX to position the current element index to the element to be deleted.

You must use GMR_$STRUCTURE_OPEN to open the structure containing the element to be deleted.

GMR_$ELEMENT_INQ_INDEX

Returns the value stored for the current element index.

FORMAT

```
GMR_$ELEMENT_INQ_INDEX (element_index, status)
```

OUTPUT PARAMETERS

element_index
The value of the current element index. This parameter is a 4-byte integer.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

A value of 1 is the first element and n is the nth element. A value of 0 is valid and is used to insert an element before element 1.

GMR_$ELEMENT_SET_INDEX

Sets the current element to the value indicated.

FORMAT

```
GMR_$ELEMENT_SET_INDEX (element_index, status)
```

INPUT PARAMETERS

element_index
The value of the current element index. This parameter is a 4-byte integer.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

A value of 1 specifies the first element and n is the nth element. A value of 0 is valid and is used to insert an element before element 1.

GMR_$F3_MESH

Inserts a primitive element into the current open structure. The element draws a mesh.

FORMAT

```
GMR_$F3_MESH (major_dim_of_mesh, minor_dim_of_mesh, points, status)
```

INPUT PARAMETERS

major_dim_of_mesh
The number of points in the major dimension of the mesh. Major dimension corresponds to the number of rows stored in a two-dimensional array (row-major form). This parameter is a 2-byte integer.

minor_dim_of_mesh
The number of points in the minor dimension of the mesh. Minor dimension corresponds to the number of columns stored in a two-dimensional array. This parameter is a 2-byte integer.

points
The points of the current mesh element. This parameter is an array of points in GMR_$F3_POINT_ARRAY_T format. Each point is an array of three real values.

The maximum number of points is defined by the constant GMR_$MAX_ARRAY_LEN.

OUTPUT PARAMETERS

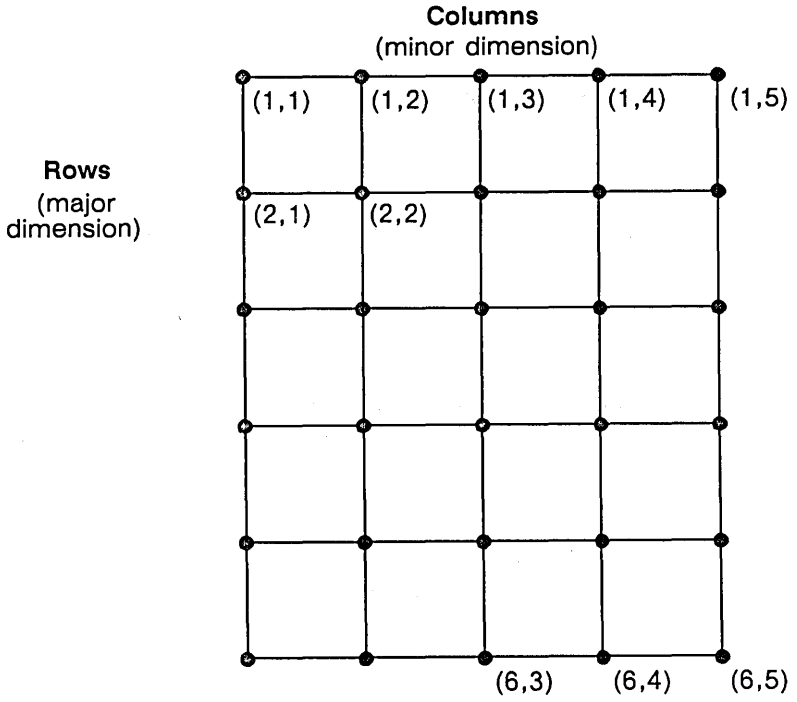
status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The rendering of a mesh is improved if the individual quadrilaterals are approximately planar.

To set and inquire the fill color and intensity for a mesh, use GMR_$FILL_COLOR/INTEN and GMR_$INQ_FILL_COLOR/INTEN.

Use GMR_$INQ_F3_MESH to inquire about a mesh element stored in a metafile.



A Mesh With 5x4 Quadrilaterals Requires 30 Points

For FORTRAN Users:

The mesh call expects the data to be stored in row-major form. Both C and Pascal store two-dimensional arrays this way. FORTRAN stores data in column-major form, so an array (n,m) in FORTRAN has m as the major dimension and n as the minor.

The following example shows a point array in each language for a mesh with M rows and N columns of points:

```
C:          GMR_$F3_POINT_T  my_array [M] [N]

FORTRAN:    REAL MY_ARRAY (N)(M)(3)

Pascal:
 my_array : ARRAY [1 .. M] OF ARRAY [1 .. N] OF GMR_$F3_POINT_T
```

GMR_$F3_MULTILINE

Inserts a primitive element into the current open structure. The element draws a sequence of disconnected line segments.

FORMAT

```
GMR_$F3_MULTILINE (n_points, points, status)
```

INPUT PARAMETERS

n_points
The number of points in current multiline element. This is a 2-byte integer. The number must be even.

The maximum number of points is defined by the constant GMR_$MAX_ARRAY_LEN.

points
The points of the current multiline element. This is an array of points in GMR_$F3_POINT_ARRAY_T format. Each point is an array of three real values in modeling coordinates.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This data type is 4 bytes long. See the Data Types section for more information.

USAGE

A line is drawn between points one and two, points three and four, etc. The number of points must be even.

This call is used to draw disconnected line segments. Use GMR_$F3_POLYLINE to draw connected line segments.

Use GMR_$INQ_F3_MULTILINE to return the points associated with a multiline element.

The multiline is drawn using the current line type and the color is determined by the current polyline color ID and intensity.

GMR_$F3_POLYGON

Inserts a primitive element into the current open structure. The element draws a polygon.

FORMAT

```
GMR_$F3_POLYGON (n_points, points, status)
```

INPUT PARAMETERS

n_points
Number of points in current polygon element. This is a 2-byte integer.

The maximum number of points is defined by the constant GMR_$MAX_ARRAY_LEN.

points
The points of the current polygon element. This parameter is an array of points in UNIV GMR_$F3_POINT_ARRAY_T format. Each point is an array of three real values.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Set and inquire the polygon fill color and intensity using GMR_$FILL_COLOR/INTEN and GMR_$INQ_FILL_COLOR/INTEN.

Use GMR_$INQ_F3_POLYGON to inquire about a polygon element stored in a metafile.

GMR_$F3_POLYLINE

Inserts a primitive element into the current open structure. The element draws a sequence of connected lines.

FORMAT

```
GMR_$F3_POLYLINE (n_points, points, closed, status)
```

INPUT PARAMETERS

n_points
Number of points in current polyline element. This is a 2-byte integer.

The maximum number of points is defined by the constant GMR_$MAX_ARRAY_LEN.

points
The points of the current polyline element. This parameter is an array of points in UNIV GMR_$F3_POINT_ARRAY_T format. Each point is an array of three real values.

closed
A Boolean (logical) flag indicating whether or not to draw a line connecting the initial and final points.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$LINE_COLOR/INTEN and GMR_$INQ_LINE_COLOR/INTEN to set and retrieve the polyline color and intensity.

Use GMR_$INQ_F3_POLYLINE to retrieve the geometric values of a polyline element.

GMR_$F3_POLYMARKER

Inserts a primitive element into the current open structure. The element draws a set of markers.

FORMAT

```
GMR_$F3_POLYMARKER (n_points, points, status)
```

INPUT PARAMETERS

n_points
The number of markers to be inserted. This is a 2-byte integer.

The maximum value of n_points is defined by the constant GMR_$MAX_ARRAY_LEN.

points
The locations of the markers, in UNIV GMR_$F3_POINT_ARRAY_T format. Each location is an array of three real values representing x, y, and z locations in modeling coordinates.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

A marker is used to graphically identify a specific location in modeling coordinate space. A typical use of markers is to represent data points on a graph.

The type of marker is set by GMR_$MARK_SET_TYPE. The default value is 1 (one pixel).

You can insert multiple markers with one polymarker element. If you then create an instance of that element, you create a copy of all the markers in the element. Likewise, if you delete the element, all of the markers created by that polymarker element are deleted.

Each marker has a nominal size. To set the scale factor for subsequently created markers, use GMR_$MARK_SCALE. Scaling does not have an effect on marker type 1.

Picking must be done at the anchor point (center) of the marker.

Markers are clipped by their anchor point. A marker can be partially visible if its anchor point is inside the viewport.

Use GMR_$INQ_F3_POLYMARKER to inquire the number of markers in a polymarker element and the location of each marker.

GMR_$FILE_CLOSE

Closes the current file, saving revisions or not as specified.

FORMAT

```
GMR_$FILE_CLOSE (save, status)
```

INPUT PARAMETERS

save

A Boolean (logical) value that indicates whether or not to save revisions. Set to true to save revisions to the currently open structure; set to false not to save revisions.

If a structure is open in this file, the structure is closed and then the file is closed. If no structure was open, save is ignored.

OUTPUT PARAMETERS

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

GMR_$FILE_CREATE

Creates a new graphics metafile and makes it the current file.

FORMAT

```
GMR_$FILE_CREATE (name, name_length, access, concurrency, file_id, status)
```

INPUT PARAMETERS

name
The pathname of the file, in UNIV NAME_$PNAME_T format. This parameter is an array of up to 256 characters.

name_length
The number of characters in the pathname. This parameter is a 2-byte integer.

access
The access mode, in GMR_$ACC_CREATE_T format. This parameter is a 2-byte integer. Specify only one of the following predefined values:

GMR_$WRITE If the file already exists, an error code is returned in the status parameter.

GMR_$OVERWRITE
If the file already exists, the previous version is deleted.

GMR_$UPDATE
If the file already exists, the previous version is opened.

concurrency
The concurrency mode, defining the number of concurrent users the file may have, in GMR_$CONC_MODE_T format. This parameter is a 2-byte integer. Specify only one of the following predefined values:

GMR_$1W N readers or one writer is permitted.

GMR_$COWRITERS
More than one writer is permitted, but all users must be on the same node.

In GMR_$COWRITERS concurrency mode, only one structure in the file may be open at a time, and only one writer may be writing to a structure at a time.

OUTPUT PARAMETERS

file_id
The identification number assigned to the file, in GMR_$FILE_ID_T format. This parameter is a 2-byte integer.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The GMR_$UPDATE access mode of GMR_$FILE_CREATE and the GMR_$CWR access mode of GMR_$FILE_OPEN produce identical results.

Use file_id with GMR_$FILE_SELECT to change the current file.

GMR_$FILE_INQ_PRIMARY_STRUCTURE

Returns the identification number of the structure assumed to be the start of the current file.

FORMAT

```
GMR_$FILE_INQ_PRIMARY_STRUCTURE (structure_id, status)
```

OUTPUT PARAMETERS

structure_id
The number of the primary structure of the current file, in GMR_$STRUCTURE_ID_T format. This parameter is a 4-byte integer.

The primary structure is assumed to be the start of the picture.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$FILE_SET_PRIMARY_STRUCTURE to set the primary structure.

GMR_$FILE_OPEN

Reopens an existing file and makes it the current file.

FORMAT

```
GMR_$FILE_OPEN (name, name_length, access, concurrency, file_id, status)
```

INPUT PARAMETERS

name
The pathname of the file, in UNIV NAME_$PNAME_T format. This parameter is an array of up to 256 characters.

name_length
The number of characters in the pathname. This parameter is a 2-byte integer.

access
The read/write accessibility, in GMR_$ACC_OPEN_T format. This parameter is a 2-byte integer. Specify only one of the following predefined values:

GMR_$WR	Read or write. In this access mode, it is an error to attempt to open a nonexistent file.
GMR_$R	Read only. In this access mode, it is an error to attempt to open a nonexistent file.
GMR_$CWR	Read or write; if file does not exist, create it.

concurrency
The concurrency mode, defining the number of concurrent users the file may have, in GMR_$CONC_MODE_T format. This parameter is a 2-byte integer. Specify only one of the following predefined values:

GMR_$1W	N readers or one writer is permitted.
GMR_$COWRITERS	More than one writer is permitted, but all users must be on the same node. In GMR_$COWRITERS concurrency mode, only one structure in the file may be open at a time, and only one writer may be writing to a structure at a time.

OUTPUT PARAMETERS

file_id
The identification number assigned to the file, in GMR_$FILE_ID_T format. This parameter is a 2-byte integer.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long.

USAGE

The GMR_$UPDATE access mode of GMR_$FILE_CREATE and the GMR_$CWR access mode of GMR_$FILE_OPEN produce identical results.

Use file_id with GMR_$FILE_SELECT to change the current file.

GMR_$FILE_SELECT

Makes the specified file the current file.

FORMAT

```
GMR_$FILE_SELECT (file_id, status)
```

INPUT PARAMETERS

file_id
The identification number of the file which is to become the current file, in GMR_$FILE_ID_T format. This parameter is a 2-byte integer.

The 3D GMR package assigns a file identification number when GMR_$FILE_CREATE or GMR_$FILE_OPEN is called.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

When a file is created or opened, it becomes the current file. After closing the current file, you must select any other open file before you can use it.

GMR_$FILE_SET_PRIMARY_STRUCTURE

Sets the structure number assumed to be the start of the current file.

FORMAT

```
GMR_$FILE_SET_PRIMARY_STRUCTURE (structure_id, status)
```

INPUT PARAMETERS

structure_id
The number of the primary structure of the current file, in GMR_$STRUCTURE_ID_T format. This parameter is a 4-byte integer.

The primary structure is assumed to be the start of the picture.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Structures are not assigned to viewports by default. You must explicitly assign a structure to a viewport using GMR_$VIEWPORT_SET_STRUCTURE.

The concept of a primary structure lets an application earmark one structure as special, so that a subsequent display program can find out which structure to assign to a viewport.

GMR_$FILL_COLOR

Inserts an attribute element into the current open structure. The element establishes fill color for polygons and meshes.

FORMAT

```
GMR_$FILL_COLOR (color, status)
```

INPUT PARAMETERS

color

The current fill color, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer.

The default value is GMR_$FILL_COLOR_DEF. This is equivalent to 1.

OUTPUT PARAMETERS

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$INQ_FILL_COLOR to return the fill color of the current (GMR_$FILL_COLOR) element.

GMR_$FILL_INTEN

Inserts an attribute element into the current open structure. The element establishes fill intensity for polygons and meshes.

FORMAT

```
GMR_$FILL_INTEN (intensity, status)
```

INPUT PARAMETERS

intensity

The fill intensity, in GMR_$INTEN_T format. This parameter is a 4-byte real value in the range [0.0, 1.0], inclusive.

The default value is GMR_$FILL_INTEN_DEF. This is equivalent to 1.0.

OUTPUT PARAMETERS

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$INQ_FILL_INTEN to return the fill intensity of the current (GMR_$FILL_INTEN) element.

GMR_$INIT

Initializes the 3D GMR package and opens the display.

FORMAT

```
GMR_$INIT (display_mode, unit, size, n_planes, status)
```

INPUT PARAMETERS

display_mode

One of four modes of operation, in GMR_$DISPLAY_MODE_T format. Specify only one of the following predefined values:

GMR_$BORROW
: Displays on the full screen, which is temporarily borrowed from the Display Manager.

GMR_$DIRECT
: Displays within a Display Manager window, which is aquired from the Display Manager.

GMR_$MAIN_BITMAP
: Displays within a bitmap allocated in main memory without a display bitmap.

GMR_$NO_BITMAP
: Allows editing of files without a main memory or a display bitmap.

unit

This parameter has three possible meanings:

The display unit, if the display mode is GMR_$BORROW. This parameter is a 2-byte integer. Currently, the only valid display unit number for borrow-display mode is the integer value of 1.

The stream identifier for the pad, if the display mode is GMR_$DIRECT, in STREAM_$ID_T format. This parameter is a 2-byte integer.

Any value, such as zero in GMR_$MAIN_BITMAP or GMR_$NO_BITMAP.

size

The size of the bitmap, in GMR_$I2_POINT_T format. This parameter is a two-element array of 2-byte integers. The first element is the bitmap width in pixels; the second element is the bitmap height in pixels. Each value may be any number between 1 and 4096 (limits are reduced to the display or window size if necessary). See the Data Types section for more information.

n_planes
The number of bitmap planes. This parameter is a 2-byte integer. The following are acceptable values:

```
For display memory bitmaps:

1      For monochromatic displays
1 - 4  For color displays in two-board configuration
1 - 8  For color displays in three-board configuration

For main memory bitmaps: 1 - 8 for all displays
```

If n_planes is larger than the number of planes in the configuration, the actual number is used.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

You can use the "unit" parameter to display metafiles in a window other than the window from which you executed your 3D GMR program:

```
VAR
   wndw : pad_$window_desc_t;
   instid,stid : stream_$id_t;
   bitmap_size : gmr_$point16_t := [1024,1024];

BEGIN {program}

  wndw.top := 0;
  wndw.left := 0;
  wndw.width := 300;
  wndw.height := 300;

  pad_$create_window ('',0, pad_$transcript, 1,
                      wndw, stid, st);
  pad_$create ('',0,pad_$input,stid, pad_$bottom,
               [pad_$init_raw],5, instid, st);

              { The "unit" parameter is the stream id of the pad
                in which you want to display metafiles. }

  gmr_$init (gmr_$direct, stid, bitmap_size, 8, st);
```

Note that C programs must also build a Pascal set structure.

Use the size argument to create smaller windows and use n_planes to partition planes.

To display a file created in main-bitmap mode or no-bitmap mode, terminate the mode and reinitialze in borrow or direct mode.

GMR_$INPUT_DISABLE

Disables an input event type.

FORMAT

```
GMR_$INPUT_DISABLE (event_type, status)
```

INPUT PARAMETERS

event_type

The input event type to be disabled, in GMR_$EVENT_T format. This parameter is a 2-byte integer. Specify only one of the following predefined values:

GMR_$KEYSTROKE
: Returned when you type a keyboard character.

GMR_$BUTTONS
: Returned when you press a button on the mouse or bitpad puck.

GMR_$LOCATOR
: Returned when you move the mouse or bitpad puck, or the touchpad.

GMR_$ENTERED_WINDOW
: Returned when the cursor enters a Display Manager window in which the 3D GMR application is running. Direct mode is required.

GMR_$LEFT_WINDOW
: Returned when the cursor leaves a Display manager window in which the 3D GMR application is running. Direct mode is required.

GMR_$LOCATOR_STOP
: Returned when you stop moving the mouse or bitpad puck, or stop using the touchpad.

OUTPUT PARAMETERS

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$INPUT_ENABLE to enable an input event type. Use GMR_$INPUT_EVENT_WAIT to get an event from an enabled device.

This call cannot be used under the DOMAIN/Dialogue system. Use DP_$GMR_INPUT_DISABLE and/or initialize in task definitions instead.

GMR_$INPUT_ENABLE

Enables an input event type.

FORMAT

```
GMR_$INPUT_ENABLE (event_type, key_set, status)
```

INPUT PARAMETERS

event_type
The event type to be enabled, in GMR_$EVENT_T format. This parameter is a 2-byte integer. Specify only one of the following predefined values:

GMR_$KEYSTROKE
Returned when you type a keyboard character.

GMR_$BUTTONS
Returned when you press a button on the mouse or bitpad puck.

GMR_$LOCATOR
Returned when you move the mouse or bitpad puck, or the touchpad.

GMR_$ENTERED_WINDOW
Returned when the cursor enters a Display Manager window in which the 3D GMR application is running. Direct mode is required.

GMR_$LEFT_WINDOW
Returned when the cursor leaves a Display Manager window in which the 3D GMR application is running. Direct mode is required.

GMR_$LOCATOR_STOP
Returned when you stop moving the mouse or bitpad puck, or stop using the touchpad.

key_set
The set of specifically enabled characters when the event type is GMR_$KEYSTROKE or GMR_$BUTTONS, in GMR_$KEYSET_T format. This parameter is an array of up to 256 characters.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The default is that all events are disabled.

Use GMR_$INPUT_EVENT_WAIT to get an event from an enabled device.

Use GMR_$INPUT_DISABLE to disable an input event type.

This call cannot not be used under the DOMAIN/Dialogue system. Use DP_GMR_INPUT_ENABLE and/or initialize in task definitions instead.

Currently, only the characters 'a' through 'd' and 'A' through 'D' are valid button events.

GMR_$INPUT_ENABLE expects a Pascal set of characters as one input argument. It is only used if the event_type equals GMR_$KEYSTROKE or GMR_$BUTTONS. The following two examples show how to build a set of characters for FORTRAN and C using this call.

Programming Examples

The two fragments included here have the following information in common:

```
build_set -- Builds a Pascal set of characters.

Input arguments

    list           -- A 16-bit array, up to 256 entries long.
                      This array contains the ordinal values of the
                      characters to be included in the set.  For
                      example, if you wish to include the capital
                      letters A through Z, make the array
                      26 entries long, including the values 65
                      through 90.

    no_of_entries  -- The number of entries used in list.
                      A 16-bit scalar.

Output arguments

    returned_set   -- The equivalent of the Pascal set of
                      characters. This can be of any type,
                      as long as it is 32 bytes long.
                      Use integer*4 returned_set(8) for FORTRAN.
                      Use LONG returned_set[8] for C.

The following fragments do not check for errors.  Therefore,
values can be outside the range 0 to 255, although this can give
unpredictable results.  The program does not check to see if the
value has already appeared in the list.

Each fragment (one for FORTRAN and one for C) builds the set anew
each time; they do not allow you to add new elements to an existing set.
```

Example 1 - FORTRAN subroutine

The following fragment builds a Pascal set of characters for FORTRAN users.

```
      subroutine build_set(list,no_of_entries,returned_set)

      integer*2 list(1),no_of_entries,returned_set(0:15)
      integer*2 i,mask(0:15),word,bit
      data mask/1,2,4,8,16#10,16#20,16#40,16#80,16#100,16#200,
     1   16#400,16#800,16#1000,16#2000,16#4000,16#8000/

c     A Pascal set of characters is a 256-bit "array." The bit
c     corresponding to the ordinal position of the character is
c     1 if the bit is in the set and 0 if the character is absent
c     from the set. In this example, the set is initialized
c     to 0, that is, no characters are present.

      do 100 i=0,15
         returned_set(i) = 0
100      continue
c
c     Go through the list, setting the bits for each character listed.
c     Note that Pascal numbers the bits right to left.
c     Therefore, a set containing only char(0), that is NULL, has
c     only the least-significant bit set in the last word of the set.

      do 110 i=1,no_of_entries
c
c     Set the appropriate bit.

         word = 15 - (list(i)/16)
         bit = mod(list(i),16)
         returned_set(word) = or(returned_set(word),mask(bit))
110   continue
c
      return
      end
```

Example 2 - C example

This fragment builds a Pascal set of characters for C users.

```
      build_set(list,no_of_entries,return_set)
      char list[];
      int no_of_entries;
      short return_set[16];
      {
         int i;
         short word;

 /* A Pascal set of characters is a 256-bit "array." The bit
    corresponding to the ordinal position of the character is
    1 if the bit is in the set and 0 if the character is absent
    from the set. In this example, the set is initialized
    to 0, that is, no characters are present.
 */
         for (i = 0; i < 16; i++)
```

```
            return_set[i] = 0;

/* Go through the list, setting the bits for each character listed.
   Note that Pascal numbers the bits right to left.  Therefore, a
   set containing only char(0), that is NULL, has only the
   least-significant bit set in the last word of the set.
*/

        for (i = 0; i < no_of_entries; i++)
        {
            /* Determine which word to set. */

            word = 15 - (list[i]/16);

            /* OR in a value of 1 shifted the correct number of bits */

            return_set[word] |= 1 << (list[i] % 16);
        }
    }
```

GMR_$INPUT_EVENT_WAIT

Checks for or waits until an occurrence of an enabled input event.

FORMAT

```
GMR_$INPUT_EVENT_WAIT (wait, event_type, event_data, position, status)
```

INPUT PARAMETERS

wait

A Boolean (logical) value that specifies when control returns to the calling program. Set to true to wait for an enabled event to occur; set to false to return control to the calling program immediately, whether or not an event has occurred.

OUTPUT PARAMETERS

event_type

The event type which occurred, in GMR_$EVENT_T format. This parameter is a 2-byte integer. Specify only one of the following predefined values:

GMR_$KEYSTROKE
: Returned when you type a keyboard character.

GMR_$BUTTONS
: Returned when you press a button on the mouse or bitpad puck.

GMR_$LOCATOR
: Returned when you move the mouse or bitpad puck, or the touchpad.

GMR_$ENTERED_WINDOW
: Returned when the cursor enters a Display Manager window in which the 3D GMR package is running. Direct mode is required.

GMR_$LEFT_WINDOW
: Returned when the cursor leaves a Display Manager window in which the 3D GMR package is running. Direct mode is required.

GMR_$LOCATOR_STOP
: Returned when you stop moving the mouse or bitpad puck, or stop using the touchpad.

event_data

The keystroke or button character associated with the event. This is a character. This parameter is not modified for other events.

position

The position in logical device coordinates at which graphics input occurred, in GMR_$F3_POINT_T format. This parameter is a three-element array of real values. See the Data Types section for more information.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$COORD_LDC_TO_WORLD to retrieve the world coordinates of the corresponding point on the view plane. Use GMR_$COORD_LDC_TO_WORK_PLANE to retrieve the world coordinates of the corresponding point on the work plane.

This call cannot be used under the DOMAIN/Dialogue system. Use DP_$EVENT_WAIT instead.

GMR_$INQ_ACLASS

Returns the attribute class for the current (GMR_$ACLASS) element.

FORMAT

```
GMR_$INQ_ACLASS (aclass_id, status)
```

OUTPUT PARAMETERS

aclass_id
The identification number of the of the attribute class, in GMR_$ACLASS_ID_T format. This parameter is a 2-byte integer.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

This call returns the attribute class used for all subsequent elements and structure instances in the current structure. The attribute class is bound to an attribute block using either the GMR_$ABLOCK_ASSIGN_DISPLAY or GMR_$ABLOCK_ASSIGN_VIEWPORT commands. The viewport binding takes precedence over the display binding.

Use GMR_$ACLASS to set the aclass ID of an aclass element in a structure.

GMR_$INQ_ADD_NAME_SET

Returns the list of names in the current (GMR_$ADD_NAME_SET) element.

FORMAT

```
GMR_$INQ_ADD_NAME_SET (n_names, name set, status)
```

OUTPUT PARAMETERS

n_names
The number of names in the add name set. This parameter is a 2 byte integer.

name set
The list of names, in GMR_$NAME_SET_T format. Each name is in the range [1, GMR_$MAX_NAME_ELEMENT], inclusive.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The current name set defined through the use of GMR_$ADD_NAME_SET and GMR_$REMOVE_NAME_SET is used to determine invisibility and pick eligibility for primitives within a structure.

See GMR_$ADD_NAME_SET for the visibility and pick eligibility criteria and an example.

GMR_$INQ_ATTRIBUTE_SOURCE

Returns the attribute type and source flag for the current (GMR_$ATTRIBUTE_SOURCE) element.

FORMAT

```
GMR_$INQ_ATTRIBUTE_SOURCE (attribute, source, status)
```

INPUT PARAMETERS

attribute

The attribute that is set, in GMR_$ATTRIBUTE_T format. This parameter is a 2 byte integer. Possible values are:

GMR_$ATTR_LINE_COLOR
: Line color for polylines and multilines.

GMR_$ATTR_LINE_INTEN
: Line intensity for polylines and multilines.

GMR_$ATTR_LINE_TYPE
: Line type for multilines and polylines.

GMR_$ATTR_FILL_COLOR
: Fill color for polygons and meshes.

GMR_$ATTR_FILL_INTEN
: Fill intensity for polygons and meshes.

GMR_$ATTR_MARK_COLOR
: Color for polymarker elements.

GMR_$ATTR_MARK_INTEN
: Intensity for polymarker elements.

GMR_$ATTR_MARK_SCALE
: Scale for polymarker elements.

GMR_$ATTR_MARK_TYPE
: Type for polymarker elements.

GMR_$ATTR_TEXT_COLOR
: Text color.

GMR_$ATTR_TEXT_INTEN
: Text intensity.

GMR_$ATTR_TEXT_HEIGHT
: Text height.

GMR_$ATTR_TEXT_EXPANSION
: Text expansion factor.

GMR_$ATTR_TEXT_SLANT
Text slant factor.

GMR_$ATTR_TEXT_SPACING
Text spacing.

GMR_$ATTR_TEXT_UP
Text up vector.

GMR_$ATTR_TEXT_PATH
Text path angle.

source
The source flag, in GMR_$ATTRIBUTE_SOURCE_T format. This parameter is a 2-byte integer. Possible values are GMR_$ATTRIBUTE_DIRECT and GMR_$ATTRIBUTE_ACLASS.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

At display time, an attribute type is not used unless its source flag has been set. This means that if you insert an explicit attribute element (e.g., GMR_$LINE_COLOR) before a primitive element (e.g., GMR_$F3_POLYLINE), the attribute is used only if the direct source flag is in effect.

This allows great flexibility at display time because the attribute elements or aclass elements in the metafile can be turned on and off.

The default is direct. This means that you can only use an aclass element if you set the source flag to aclass for each type of attribute in the ablock. Likewise, after you set an attribute type to aclass, you can only use an explicit attribute element of that type if you set the flag for that type to direct.

GMR_$INQ_CONFIG

Returns the number of planes and the size of the current display device.

FORMAT

```
GMR_$INQ_CONFIG (op, unit_or_pad, numplanes, size, status)
```

INPUT PARAMETERS

op

One of four modes of operation, in GMR_$DISPLAY_MODE_T format. This parameter is ignored for this release.

unit_or_pad

This parameter is ignored for this release. It has three possible meanings:

1. The display unit, if the display mode is GMR_$BORROW. This parameter is a 2-byte integer.

2. The stream identifier for the pad, if the display mode is GMR_$DIRECT, in STREAM_$ID_T format. This parameter is a 2-byte integer.

3. Any value, such as zero in GMR_$MAIN_BITMAP or GMR_$NO_BITMAP.

OUTPUT PARAMETERS

numplanes

The number of available bit planes in GMR_$I_T format. This parameter is a 2-btye integer.

size

The size of the display in GMR_$I2_POINT_T format. This is a two-element array of 2-byte integers. For example, in a 1024x800 display, the first integer contains 1024 and the second contains 800.

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

This routine is useful when setting the color map or deciding whether to use double buffering.

The two input parameters are ignored for this release. They must be in the correct format, but any value can be used.

GMR_$INQ_CONFIG is the only 3D GMR routine (besides GMR_$INIT) that is usable when the graphics metafile package is not initialized.

GMR_$INQ_ELEMENT_TYPE

Returns the type of the current element in the current open structure.

FORMAT

```
GMR_$INQ_ELEMENT_TYPE (element_type, attribute_type, status)
```

OUTPUT PARAMETERS

element_type
Element type, in GMR_$ELEMENT_TYPE_T format. This parameter is a 2-byte integer. See the Data Types section for more information.

attribute_type
Attribute type, in GMR_$ELEMENT_ATTRIB_TYPE_T format. This parameter is a 2-byte integer. See the Data Types section for more information.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

GMR_$INQ_F3_MESH

Returns the major and minor mesh dimensions and the list of mesh points associated with the current (GMR_$F3_MESH) element.

FORMAT

```
GMR_$INQ_F3_MESH (array_size, major_dim_of_mesh, minor_dim_of_mesh,
                  points, status)
```

INPUT PARAMETERS

array_size
The size of the output array. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

major_dim_of_mesh
Number of points in the major dimension of the mesh (stored in row major form). This corresponds to the number of rows in a two-dimensional array. This is a 2-byte integer.

minor_dim_of_mesh
Number of points in the minor dimension of the mesh. This corresponds to the number of columns in a two-dimensional array. This is a 2-byte integer.

points
The points of the current mesh element. This parameter is an array of points in GMR_$F3_POINT_ARRAY_T format. Each point is an array of three real values.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

If array_size is less than the number of points in the mesh, only array_size points are returned and the status is set accordingly to error code GMR_$INQ_ARRAY_SIZE_SMALL. Note that the correct column and row sizes are returned. You can use that information to determine the size of the array needed.

Set and inquire mesh fill color and intensity using GMR_$FILL_COLOR/INTEN and GMR_$INQ_FILL_COLOR/INTEN.

For FORTRAN Users:

The mesh call expects the data to be stored in row-major form. See the Usage section of GMR_$F3_MESH for more information.

GMR_$INQ_F3_MULTILINE

Returns the list of multiline points associated with the current (GMR_$F3_MULTILINE) element.

FORMAT

```
GMR_$INQ_F3_MULTILINE (array_size, n_points, points, status)
```

INPUT PARAMETERS

array_size
The size of the output array. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

n_points
Number of points in current multiline element. This is a 2-byte integer. The number is always even.

points
The points of the current multiline element. This parameter is an array of points in GMR_$F3_POINT_ARRAY_T format. Each point is an array of three real values.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

If array_size is less than the number of points in the multiline, only array_size is returned and the status is set to error code GMR_$INQ_ARRAY_SIZE_SMALL. Note that the correct number of points is returned. You can use that information to determine the size of the array needed.

GMR_$INQ_F3_POLYGON

Returns the list of polygon points associated with the current (GMR_$F3_POLYGON) element.

FORMAT

```
GMR_$INQ_F3_POLYGON (array_size, n_points, points, status)
```

INPUT PARAMETERS

array_size
The size of the output array. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

n_points
Number of points in current polygon element. This is a 2-byte integer.

The constant GMR_$MAX_ARRAY_LEN defines the maximum number of points.

points
The points of the current polygon element, in GMR_$F3_POINT_ARRAY_T format. This parameter is an array of points. Each point is an array of three real values specifying x, y, and z.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

If array_size is less than the number of points in the polygon, only array_size is returned and the status is set to error code GMR_$INQ_ARRAY_SIZE_SMALL. Note that the correct number of points is returned. You can use that information to determine the size of the array needed.

GMR_$INQ_F3_POLYLINE

Returns the list of polyline points associated with the current (GMR_$F3_POLYLINE) element.

FORMAT

```
GMR_$INQ_F3_POLYLINE (array_size, n_points, points, closed, status)
```

INPUT PARAMETERS

array_size
The size of the output array. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

n_points
Number of points in current polyline element. This is a 2-byte integer.

The maximum number of points is defined by the constant GMR_$MAX_ARRAY_LEN.

points
The points of the current polyline element. This parameter is an array of points in GMR_$F3_POINT_ARRAY_T format. Each point is an array of three real values.

closed
A Boolean (logical) value indicating whether the current polyline element is closed or not.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

If array_size is less than the number of points in the polyline, only array_size is returned and the status is set to error code GMR_$INQ_ARRAY_SIZE_SMALL. Note that the correct number of points is returned. You can use that information to determine the size of the array needed.

GMR_$INQ_F3_POLYMARKER

Returns the list of marker points associated with the current (GMR_$F3_POLYMARKER) element.

FORMAT

```
GMR_$INQ_F3_POLYMARKER (array_size, n_points, points, status)
```

INPUT PARAMETERS

array_size
The size of the output array. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

n_points
The number of markers in the current polymarker element. This is a 2-byte integer.

The constant GMR_$MAX_ARRAY_LEN defines the maximum value of n_points.

points
The locations of the markers in the current polymarker element, in GMR_$F3_POINT_ARRAY_T format. Each location is an array of three real values representing x, y, and z locations in modeling coordinates.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

If array_size is less than the number of points in the polymarker, only array_size is returned and the status is set to error code GMR_$INQ_ARRAY_SIZE_SMALL. Note that the correct number of points is returned. You can use that information to determine the size of the array needed.

GMR_$INQ_FILL_COLOR

Returns the color ID specified by the current (GMR_$FILL_COLOR) element.

FORMAT

```
GMR_$INQ_FILL_COLOR (color_id, status)
```

OUTPUT PARAMETERS

color_id
The color_id set by the current fill_color element, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The GMR_$FILL_COLOR routine establishes the color used to fill polygons and meshes.

GMR_$INQ_FILL_INTEN

Returns the intensity of the current (GMR_$FILL_INTEN) element.

FORMAT

```
GMR_$INQ_FILL_INTEN (intensity, status)
```

OUTPUT PARAMETERS

intensity
The intensity value set by the current fill intensity element, in GMR_$INTEN_T format. This parameter is a 4-byte real value in the range [0.0, 1.0], inclusive.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

GMR_$FILL_INTEN established the fill intensity for polygons and meshes.

GMR_$INQ_INSTANCE_TRANSFORM

Returns the structure ID and the transformation applied at rendering time of the current (GMR_$INSTANCE_TRANSFORM) element.

FORMAT

```
GMR_$INQ_INSTANCE_TRANSFORM (structure_id, transform_matrix, status)
```

OUTPUT PARAMETERS

structure_id
The ID of the instanced structure, in GMR_STRUCTURE_ID_T format. This parameter is a 4-byte integer.

transform_matrix
The transformation being applied by the current GMR_$INSTANCE_TRANSFORM element, in GMR_$4X3_MATRIX_T format. This parameter is a two-dimensional array of 4-byte real values. Refer to the Data Types section for more information.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use this routine to retrieve the values of instance elements created by both GMR_$INSTANCE_TRANSFORM and GMR_$INSTANCE_TRANSFORM_FWD_REF.

GMR_$INQ_LINE_COLOR

Returns the color_id specified by the current (GMR_$LINE_COLOR) element.

FORMAT

```
GMR_$INQ_LINE_COLOR (color, status)
```

OUTPUT PARAMETERS

color

The color_id set by the current line color element, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer.

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The color established by a GMR_$LINE_COLOR element is used for drawing polylines and multilines.

GMR_$INQ_LINE_INTEN

Returns the intensity of the current (GMR_$LINE_INTEN) element.

FORMAT

```
GMR_$INQ_LINE_INTEN (inten, status)
```

OUTPUT PARAMETERS

inten
The intensity value set by the current line intensity element, in GMR_$INTEN_T format. This parameter is a 4-byte real value in the range [0.0, 1.0], inclusive.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The intensity established by a GMR_$LINE_INTEN element is used for rendering both polylines and multilines.

GMR_$INQ_LINE_TYPE

Returns the line type ID of the current (GMR_$LINE_TYPE) element.

FORMAT

```
GMR_$INQ_LINE_TYPE (type_id, status)
```

OUTPUT PARAMETERS

type_id

The line type ID, in GMR_$LINE_TYPE_T format. This parameter is a 2-byte integer. Values are currently restricted to 1, 2, 3, and 4 as follows:

```
1 = Solid
2 = Dashed
3 = Dotted
4 = Dashed-dotted
```

The default is 1 (solid).

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$LINE_TYPE to insert a line type attribute into the metafile.

GMR_$INQ_MARK_COLOR

Returns the color_id specified by the current (GMR_$MARK_COLOR) element.

FORMAT

```
GMR_$INQ_MARK_COLOR (color_id, status)
```

OUTPUT PARAMETERS

color_id
The color_id set by the current polymarker_color element, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$MARK_COLOR to establish the color for polymarker elements.

GMR_$INQ_MARK_INTEN

Returns the intensity of the current (GMR_$MARK_INTEN) element.

FORMAT

```
GMR_$INQ_MARK_INTEN (intensity, status)
```

OUTPUT PARAMETERS

intensity
The intensity value set by the current polymarker intensity element, in GMR_$INTEN_T format. This parameter is a 4-byte real value in the range [0.0, 1.0], inclusive.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$MARK_INTEN to establish the intensity for polymarker elements.

GMR_$INQ_MARK_SCALE

Returns the scale factor for the current (GMR_$MARK_SCALE) element.

FORMAT

```
GMR_$INQ_MARK_SCALE (scale_factor, status)
```

OUTPUT PARAMETERS

scale_factor
The polymarker scale factor, in GMR_$MARK_SCALE_T format. This is 4-byte real value. The default scale factor is 1.0.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$MARK_SCALE to establish the scale factor for polymarker elements.

GMR_$INQ_MARK_TYPE

Returns the mark type for the current (GMR_$MARK_TYPE) element.

FORMAT

```
GMR_$INQ_MARK_INTEN (type, status)
```

OUTPUT PARAMETERS

type
The mark type in GMR_$MARK_TYPE_T format. This is 2-byte integer in the range [1, 5], inclusive. The default is 1 (one pixel).

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$MARK_TYPE to establish the polymarker type.

GMR_$INQ_REMOVE_NAME_SET

Returns the list of names in the current (GMR_$REMOVE_NAME_SET) element.

FORMAT

```
GMR_$INQ_REMOVE_NAME_SET (n_names, name_set, status)
```

OUTPUT PARAMETERS

n_names
The number of names in the name set. This parameter is a 2-byte integer.

name_set
The list of names in the remove name set, in GMR_$NAME_SET_T format. Eache name is in the range [1, GMR_$MAX_NAME_ELEMENT], inclusive.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

GMR_$ADD_NAME_SET and GMR_$REMOVE_NAME_SET define the current name set.

GMR_$INQ_TAG

Returns the length of the text stored in the current (GMR_$TAG) element and a specified substring of that text.

FORMAT

```
GMR_$INQ_TAG (tag_start, tag_copy, tag, tag_length, status)
```

INPUT PARAMETERS

tag_start
The index of the first character of the substring to be returned. This parameter is a 4-byte integer greater than 0 (the integer value of 1 corresponds to the first character in the tag).

tag_copy
The number of characters in the substring to be returned. This parameter is a 4-byte integer.

OUTPUT PARAMETERS

tag
The text substring, in GMR_$STRING_T format. This is an array of characters that must be large enough to hold tag_copy characters.

tag_length
The length of the entire stored tag text. This parameter is a 4-byte integer.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$TAG to change the text string stored in this element.

To inquire the entire text for a tag when you do not know how long it is, use two calls to GMR_$INQ_TAG. First, supply a small value in tag_copy to determine how much space the text requires. Then, make the second call to retrieve the other parameters.

GMR_$INQ_TEXT

Returns the text string, the number of characters, and the anchor point stored in the current (GMR_$TEXT) element.

FORMAT

```
GMR_$INQ_TEXT (string, string_length, position, status)
```

OUTPUT PARAMETERS

string
The string stored in the text element, in GMR_$STRING_T format. This parameter is an array of up to GMR_$MAX_STRING_LENGTH characters.

string_length
The length of the string. This parameter is a 2-byte integer.

position
The anchor point of the text string, in GMR_$F3_POINT_T format. This is a point in model coordinates.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

GMR_$INQ_TEXT_COLOR

Returns the color_id stored in the currrent (GMR_$TEXT_COLOR) element.

FORMAT

```
GMR_$INQ_TEXT_COLOR (color_id, status)
```

OUTPUT PARAMETERS

color_id
The color identifier, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer in the range [0, GMR_$MAX_COLOR_ID], inclusive.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The color established by a GMR_$TEXT_COLOR element is used for rendering text.

GMR_$INQ_TEXT_EXPANSION

Returns the expansion factor stored for the currrent (GMR_$TEXT_EXPANSION) element. Text expansion controls the ratio of height to width of text characters.

FORMAT

```
GMR_$INQ_TEXT_EXPANSION (expansion, status)
```

OUTPUT PARAMETERS

expansion

The text character expansion for this attribute block, in GMR_$TEXT_EXPANSION_T format. This is a 4-byte real value. This attribute controls the aspect ratio for the font. The default value is 1.0 which preserves the aspect ratio defined in the font.

Values greater than 1.0 create wider characters. Values less than 1.0 create thinner characters.

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

GMR_$INQ_TEXT_HEIGHT

Returns the height stored for the currrent (GMR_$TEXT_HEIGHT) element. Text height controls the actual size of text characters.

FORMAT

```
GMR_$INQ_TEXT_HEIGHT (height, status)
```

OUTPUT PARAMETERS

height
The text character height, in GMR_$TEXT_HEIGHT_T format. This parameter is a 4-byte real value in viewing coordinates (same as world coordinates).

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

GMR_$INQ_TEXT_INTEN

Returns the value stored for the currrent (GMR_$TEXT_INTEN) element.

FORMAT

```
GMR_$INQ_TEXT_INTEN (intensity, status)
```

OUTPUT PARAMETERS

intensity
The value for text intensity, in GMR_$INTEN_T format. This parameter is a 4-byte real value in the range [0.0, 1.0], inclusive.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The intensity established by a GMR_$TEXT_INTEN element applies only to the rendering of text elements.

GMR_$INQ_TEXT_PATH

Returns the text path angle stored for the currrent (GMR_$TEXT_PATH) element. Text path determines where the second and subsequent characters in a text string are placed.

FORMAT

```
GMR_$INQ_TEXT_PATH (angle, status)
```

OUTPUT PARAMETERS

angle

The angle that determines where the second and subsequent characters in a string are placed, in GMR_$TEXT_PATH_T format. This parameter is a 4-byte real value.

An angle of 0.0 radians is to the right of the up vector. Angles greater than 0.0 radians are measured counterclockwise from the 0.0 radian position.

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

GMR_$INQ_TEXT_SLANT

Returns the slant factor stored for the currrent (GMR_$TEXT_SLANT) element. A negative value produces a left slant. A positive value produces a right slant.

FORMAT

```
GMR_$INQ_TEXT_SLANT (slant, status)
```

OUTPUT PARAMETERS

slant

The amount that the top of the character is shifted, in GMR_$TEXT_SLANT_T format. This parameter is a 4-byte real value.

The amount is determined by multiplying the text attributes slant, height, and expansion_factor. Zero is the default; a value between 0.0 and 1.0 yields an italics-like character (slanting to the right).

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

GMR_$INQ_TEXT_SPACING

Returns the intercharacter spacing stored for the currrent (GMR_$TEXT_SPACING) element.

FORMAT

```
GMR_$INQ_TEXT_SPACING (spacing, status)
```

OUTPUT PARAMETERS

spacing
The intercharacter spacing, in GMR_$TEXT_SPACING_T format. This parameter is a 4-byte real value that defines spacing as a fraction of text height.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

GMR_$INQ_TEXT_UP

Returns the up direction of text on the projection plane stored for the currrent (GMR_$TEXT_UP) element.

FORMAT

```
GMR_$INQ_TEXT_UP (up_vector, status)
```

OUTPUT PARAMETERS

up_vector
The up direction of text on the projection plane in viewing coordinates (same as world coordinates), in GMR_$TEXT_UP_T format. This parameter is a pair of 4-byte real values. An up vector of (0.0, 1.0) is most commonly used.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

GMR_$INSTANCE_ECHO

Echos an element or a subtree of an application-supplied instance path in a specified viewport.

FORMAT

```
GMR_$INSTANCE_ECHO (viewport_id, depth, path, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

depth
The depth within the path at which to begin echoing, in GMR_$INSTANCE_PATHLENGTH_T format. This parameter is a 4-byte integer.

path
The instance path, in GMR_$INSTANCE_PATH_T format. This is a list of (structure ID, element index) pairs that uniquely defines an element.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The 3D GMR package defines two types of echoing -- pick echo and instance echo. Pick echo is the system response to a pick operation. Instance echo uses an application-supplied instance path that does not have to come from a pick operation. This feature allows the application to customize the echo function.

For example, an application can choose to echo all objects with a certain property (e.g., size or price). The application can generate the path and then pass it to GMR_$INSTANCE_ECHO.

You can use depth to specify how far down the instance path to begin echoing. In doing so, you can echo a subtree or the lowest level individual element. For example, a depth of 1 causes the entire top level structure to be echoed when the path order is top-first.

Use GMR_$INSTANCE_ECHO_SET_METHOD to set the instance echo method for a viewport.

See GMR_$PICK_SET_ECHO_METHOD for pick echo.

GMR_$INSTANCE_ECHO_INQ_METHOD

Returns the instance echo method for a specified viewport.

FORMAT

```
GMR_$INSTANCE_ECHO_INQ_METHOD (viewport_id, method, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

method
The instance echo method for the viewport, in GMR_$INSTANCE_ECHO_METHOD_T format. This parameter is a 2-byte integer. The possible values are:

GMR_$ELEMENT_ECHO_ABLOCK
Uses the echo method described by the highlight attribute block associated with the viewport.

GMR_$ELEMENT_ECHO_BBOX
Draws a bounding box around the structure containing the selected element or subtree. The 3D GMR package automatically maintains a bounding box for each element (and structure).

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$INSTANCE_ECHO_SET_METHOD to set the instance echo method for a viewport.

The default is GMR_$ELEMENT_ECHO_BBOX.

GMR_$INSTANCE_ECHO_SET_METHOD

Sets the instance echo method for a viewport to either ablock or bounding box.

FORMAT

```
GMR_$INSTANCE_ECHO_SET_METHOD (viewport_id, method, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

method
The instance echo method for the viewport, in GMR_$INSTANCE_ECHO_METHOD_T format. This parameter is a 2-byte integer. The possible values are:

GMR_$ELEMENT_ECHO_ABLOCK
Uses the echo method described by the highlight attribute block associated with the viewport.

GMR_$ELEMENT_ECHO_BBOX
Draws a bounding box around the structure containing the selected element or subtree. This is the default. The 3D GMR package automatically maintains a bounding box for each element (and structure).

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

GMR_$INSTANCE_ECHO echos an element or subtree using the method specified by GMR_$INSTANCE_ECHO_SET_METHOD.

Use GMR_$VIEWPORT_SET_HILIGHT_ABLOCK to set the highlight attribute block for a viewport.

When you specify the ablock method and do not assign a highlighting attribute block to the viewport, the default highlighting attribute block is used.

The default is GMR_$ELEMENT_ECHO_BBOX.

GMR_$INSTANCE_TRANSFORM

Inserts an instance element into the current open structure. The elementinstances an identified structure with a specified transformation matrix.

FORMAT

```
GMR_$INSTANCE_TRANSFORM (structure_id, trans_matrix, status)
```

INPUT PARAMETERS

structure_id
The ID of the instanced structure, in GMR_STRUCTURE_ID_T format. This parameter is a 4-byte integer.

trans_matrix
The transformation being applied to the instanced structure, in GMR_$4X3_MATRIX_T format. This parameter is a two-dimensional array of 4-byte real values. Refer to the Data Types section for more information.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use this call to instance a structure as described by the transformation matrix with any combination of scaling, translation, and rotation applied to it. For example, a structure with an ID of bolt_id can be instanced from any other structure in the metafile using the syntax:

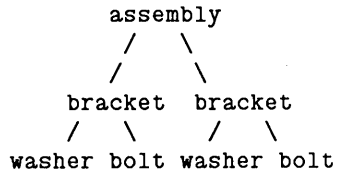
```
GMR_$INSTANCE_TRANSFORM(bolt_id, trans_matrix, status)
```

where trans_matrix is an appropriate transformation matrix.

You can use the matrix utilities to create the transformation matrix (see the GMR_$4X3_MATRIX group of calls).

Example

The following fragment creates a hierarchical metafile that can be represented graphically as follows:



```
gmr_$structure_create('bolt', 4, bolt_id, status);
 .
 . {Add bolt geometry here.}
 .
gmr_$structure_close(TRUE, status);


gmr_$structure_create('washer', 6, washer_id, status);
 .
 . {Add washer geometry here.}
 .
gmr_$structure_close(TRUE, status);


gmr_$structure_create('bracket', 7, bracket_id, status);
 .
 . {Create bracket geometry here. }
 .
 . {Next add two bolts and two washer to the bracket.}
 .
 gmr_$instance_transform(bolt_id,mat1,status);
 gmr_$instance_transform(washer_id,mat1,status);
 gmr_$instance_transform(bolt_id,mat2,status);
 gmr_$instance_transform(washer_id,mat2,status);
gmr_$structure_close(TRUE, status);

gmr_$structure_create('assembly', 8, assembly_id, status);
 .
 . {Create final assembly housing geometry here.}
 .
 . {Next add two complete brackets to the final assembly.}
 .
 gmr_$instance_transform(bracket_id,mat3,status);
 gmr_$instance_transform(bracket_id,mat4,status);
gmr_$structure_close(TRUE, status);

{Assign the assembly structure to the default viewport and draw it.}

vpid := 1;
gmr_$viewport_set_structure(vpid, assembly_id, status);
gmr_$viewport_clear(vpid, status);
gmr_$viewport_refresh(vpid, status);
```

Use GMR_$INQ_INSTANCE_TRANSFORM to retrieve the instance properties of the current (GMR_$INSTANCE_TRANSFORM) element.

See also GMR_$INSTANCE_TRANSFORM_FWD_REF.

GMR_$INSTANCE_TRANSFORM_FWD_REF

A forward-referencing instance routine. This routine creates a new structure, returns the structure ID, and inserts an instance element into the current open structure.

FORMAT

```
GMR_$INSTANCE_TRANSFORM_FWD_REF (name, namelength, trans_matrix,
                                 structure_id, status)
```

INPUT PARAMETERS

name
The name of the new structure, in NAME_$PNAME_T format. This parameter is a character string.

namelength
The number of characters in the name. This parameter is a 2-byte integer. If a namelength is 0, the structure is not given a unique name.

trans_matrix
The transformation being applied to the new structure, in GMR_$4X3_MATRIX_T format. This parameter is a two-dimensional array of 4-byte real values. See the Data Types section for more information.

OUTPUT PARAMETERS

structure_id
The ID of the instanced structure, in GMR_$STRUCTURE_ID_T format. This parameter is a 4-byte integer.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

This call combines GMR_$STRUCTURE_CREATE and GMR_$INSTANCE_TRANSFORM. As with GMR_$STRUCTURE_CREATE, you do not have to name the structure. Instead, you can use a name length of 0 as follows:

```
GMR_$INSTANCE_TRANSFORM_FWD_REF('', 0, trans_matrix, structure_id, status)
```

Use GMR_$INQ_INSTANCE_TRANSFORM to retrieve the instance properties of the current (GMR_$INSTANCE_TRANSFORM) element created by this forward referencing routine.

See also GMR_$STRUCTURE_CREATE and GMR_$INSTANCE_TRANSFORM.

GMR_$LINE_COLOR

Inserts an attribute element into the current open structure. The element establishes line color for polylines and multilines.

FORMAT

```
GMR_$LINE_COLOR (color, status)
```

INPUT PARAMETERS

color

The current line color, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer.

The default value is GMR_$LINE_COLOR_DEF. This is equivalent to an integer value of 1.

OUTPUT PARAMETERS

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$INQ_LINE_COLOR to return the color of the current (GMR_$LINE_COLOR) element.

GMR_$LINE_INTEN

Inserts an attribute element into the current open structure. The element establishes line intensity for polylines and multilines.

FORMAT

```
GMR_$LINE_INTEN (intensity, status)
```

INPUT PARAMETERS

intensity

The line intensity, in GMR_$INTEN_T format. This parameter is a 4-byte real value in the range [0.0, 1.0], inclusive.

The default value is GMR_$LINE_INTEN_DEF. This is equivalent to 1.0.

OUTPUT PARAMETERS

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$INQ_LINE_INTEN to return the intensity of the current (GMR_$LINE_INTEN) element.

GMR_$LINE_TYPE

Inserts an attribute element into the current open structure. The element establishes the line type for polylines and multilines.

FORMAT

```
GMR_$LINE_TYPE (type_id, status)
```

INPUT PARAMETERS

type_id
The line type ID, in GMR_$LINE_TYPE_ID_T format. This parameter is a 2-byte integer. Values are currently restricted to 1, 2, 3, and 4 as follows:

```
1 = Solid
2 = Dashed
3 = Dotted
4 = Dashed-dotted
```

The default type is 1 (solid).

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$INQ_LINE_TYPE to retrieve the line type ID of the current (GMR_$LINE_TYPE) element.

Use GMR_$ABLOCK_SET_LINE_TYPE to set the line type ID for an attribute block.

GMR_$MARK_COLOR

Inserts an attribute element into the current open structure. The element establishes the color for polymarker elements.

FORMAT

```
GMR_$MARK_COLOR (color, status)
```

INPUT PARAMETERS

color

The current marker color, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer.

The default value is GMR_$MARK_COLOR_DEF. This is equivalent to 1.

OUTPUT PARAMETERS

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$INQ_MARK_COLOR to return the color ID of the current (GMR_$MARK_COLOR) element.

GMR_$MARK_INTEN

Inserts an attribute element into the current open structure. The element establishes the intensity for polymarker elements.

FORMAT

```
GMR_$MARK_INTEN (intensity, status)
```

INPUT PARAMETERS

intensity

The polymarker intensity, in GMR_$INTEN_T format. This parameter is a 4-byte real value in the range [0.0, 1.0], inclusive.

The default value is GMR_$MARK_INTEN_DEF. This is equivalent to 1.0.

OUTPUT PARAMETERS

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$INQ_MARK_INTEN to return the intensity of the current (GMR_$MARK_INTEN) element.

GMR_$MARK_SCALE

Inserts an attribute element into the current open structure. The element establishes the scale factor for polymarker elements.

FORMAT

```
GMR_$MARK_SCALE (scale_factor, status)
```

INPUT PARAMETERS

scale_factor
The polymarker scale factor, in GMR_$MARK_SCALE_T format. This is 4-byte real value. The default scale factor is GMR_$MARK_SCALE_DEF. This is equivalent to 1.0.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Consider a marker as being in its own coordinate system with its center at the origin. Scale multiplies each coordinate by the scale factor and then truncates to an integer.

Scale factors less than 1.0 are not supported. If you specify a value less than 1.0, the 3D GMR package uses 1.0.

Scaling does not have any effect on marker type 1 (a one-pixel marker).

Use GMR_$INQ_MARK_SCALE to return the scale factor for the current (GMR_$MARK_SCALE) element.

GMR_$MARK_TYPE

Inserts an attribute element into the current open structure. The element establishes the polymarker type.

FORMAT

```
GMR_$MARK_TYPE (type, status)
```

INPUT PARAMETERS

type

The polymarker type, in GMR_$MARK_TYPE_T format. This is 2-byte integer in the range [1, 5], inclusive. The default is 1 (one pixel).

OUTPUT PARAMETERS

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$INQ_MARK_TYPE to return the marker type for the current (GMR_$MARK_TYPE) element.

The five marker types are shown below.

Type ID	Marker
1	• (single pixel)
2	+
3	✳
4	○
5	×

The default is type 1 (one pixel).

GMR_$PICK

Traverses the metafile using the current pick method and returns the path of an element that crosses the pick aperture.

FORMAT

```
GMR_$PICK (viewport_id, center, pick_index, pick_data_size, pick_data,
          status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport where the pick is to be found, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

center
The (x,y,z) coordinates of the center of the pick aperture, in GMR_$F3_POINT_T format. This parameter is a three-element array of real values specified in logical device coordinates.

pick_index
The index of the element to be picked, in GMR_$INSTANCE_PATH_INDEX_T format. That is, the pick_index defines n. Using the current pick method, the nth element that crosses the pick aperture and meets the pick criteria is selected.

pick_data_size
The size (in bytes) of the record containing pick data. This parameter is a 4-byte integer. To return all the data in pick_data, set pick_data _size to GMR_$PICK_DATA_SIZE.

If pick_data_size indicates the pick_data record is not large enough, then 3D GMR returns only pick_data_size bytes of data.

OUTPUT PARAMETERS

pick_data
A variable length record, in GMR_$PICK_DATA_T format containing the following information:

element_type	The type of element picked, in GMR_$ELEMENT_TYPE_T format. This data type is 2-bytes long.
path_depth	The length of the path, in GMR_$INSTANCE_PATHLENGTH_T format. This data type is 2-bytes long.
pick_path	The path of the picked element, in GMR_$INSTANCE_PATH_T format. This is a list of (structure ID, element index) pairs that uniquely defines the picked element. The order of the returned path is either picked element first or picked element last. There are up to (GMR_$MAX_INSTANCE_PATH * 8) bytes in the array.

FORTRAN users, refer to the Usage section for an example of how to set up this data type.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

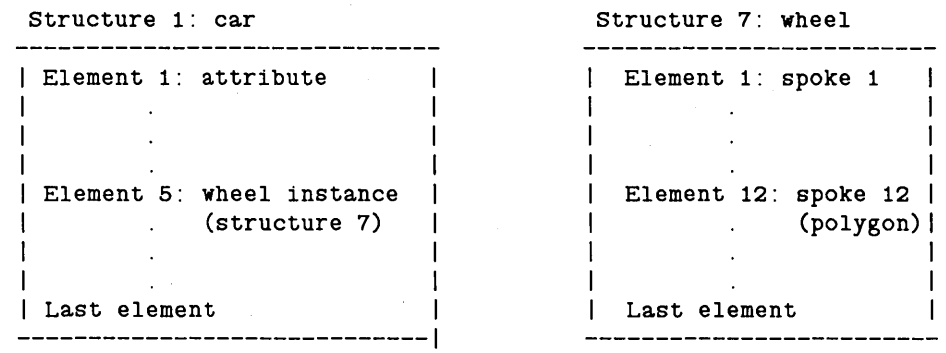
USAGE

The pick path consists of all instances and the primitive within the lowest level. For example, if the spoke of a car's wheel is instanced from a higher level structure of the entire car, its pick path is the following:

```
- stucture ID of car, element index of wheel instance
- structure ID of wheel, element index of spoke primitive
```

The path length is two. Each level contains one structure ID and one element index.

Example:



Assume that the operator identified spoke 12 using a mouse button:

```
pick_index     := 1;
pick_data_size := gmr_$pick_data_size;

GMR_$INPUT_EVENT_WAIT(TRUE, event, ch, center, status);
GMR_$PICK(viewport_id, pick_index, pick_data_size, pick_data, status);
```

The fragment above returns this information in pick_data:

```
element type   = polygon
path depth     = 2
instance path  = 1,5     (level 1)
                 7,12    (level 2)
```

FORTRAN users:

The following fragment defines a variable (pick_data) that is of type GMR_$PICK_DATA_T.

The value 130 assumes a GMR_$MAX_INSTANCE_DEPTH value of 32. Using the formula defined in the Data Types section of Chapter 1, you must allocate a total of 260 bytes. The INTEGER*2 declaration requires an array size of 130.

```
INTEGER *2 pick_data(130)
INTEGER *2 element_type
INTEGER *2 pick_path_depth
INTEGER *4 pick_path(2, 32)

EQUIVALENCE (element_type, pick_data(1))
EQUIVALENCE (pick_path_depth, pick_data(2))
EQUIVALENCE (pick_path(1,1), pick_data(3))
```

Set the pick aperture size with GMR_$PICK_SET_APERTURE_SIZE.

Set the pick method using GMR_$PICK_SET_METHOD.

Use GMR_$VIEWPORT_SET_PATH_ORDER to set the path order.

GMR_$PICK_INQ_APERTURE_SIZE

Returns the width, height, and depth of the pick aperture in a specified viewport.

FORMAT

```
GMR_$PICK_INQ_APERTURE_SIZE (viewport_id, width, height, depth, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

width
The width of the pick aperture, in GMR_$F_T format. This parameter is a real value in logical device coordinates.

height
The height of the pick aperture, in GMR_$F_T format. This parameter is a real value in logical device coordinates.

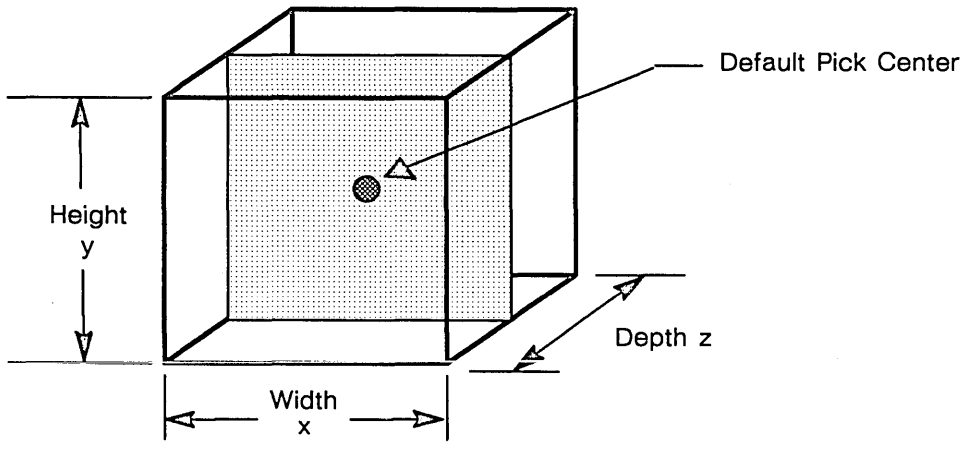
depth
The depth of the pick aperture, in GMR_$F_T format. This parameter is a real value in logical device coordinates.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

GMR_$PICK searches for structures/elements that fall into the following region:

```
(center.x - 0.5*width  to  center.x + 0.5*width,
 center.y - 0.5*height to  center.y + 0.5*height,
 center.z - 0.5*depth  to  center.z + 0.5*depth)
```



The default depth is 1. This is the depth of the default viewport (0 to 1). The default height and width is 0.1.

Use GMR_$PICK_SET_APERTURE to establish the pick aperture for a viewport.

GMR_$PICK_INQ_CENTER

Returns the center of the pick aperture in a specified viewport.

FORMAT

```
GMR_$PICK_INQ_CENTER (viewport_id, center, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

center
The (x,y,z) coordinates of the center of the pick aperture, in GMR_$F3_POINT_T format. This parameter is a three-element array of real values specified in logical device coordinates.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

GMR_$PICK sets the pick center. An input device such as a mouse or touch pad usually supplies the location.

The point that is returned is the center from the last pick operation in this viewport. The routine returns the default (0.0, 0.0, 0.0) if no calls to GMR_$PICK have been issued.

GMR_$PICK_INQ_ECHO_METHOD

Returns the current pick echo method for a viewport.

FORMAT

```
GMR_$PICK_INQ_ECHO_METHOD (viewport_id, method, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

method
The pick echo method for the viewport, in GMR_$PICK_ECHO_METHOD_T format. This parameter is a 2-byte integer. Select one of the following predefined values:

GMR_$PICK_ECHO_NONE
Does not echo the picked element.

GMR_$PICK_ECHO_ABLOCK
Echo the picked element using the echo method described by the highlight attribute block associated with the viewport.

GMR_$PICK_ECHO_BBOX
Draws a bounding box around the structure containing the picked element. The 3D GMR package automatically maintains a bounding box for each element (and structure).

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$PICK_SET_ECHO_METHOD to set the pick echo method for a viewport.

Use GMR_$VIEWPORT_SET_HILIGHT_ABLOCK to set the ablock for GMR_$PICK_ECHO_METHOD.

GMR_$PICK_INQ_METHOD

Returns the pick method in use for a particular viewport.

FORMAT

```
GMR_$PICK_INQ_METHOD (viewport_id, method, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

method
The method used for picking, in GMR_$PICK_METHOD_T format. This parameter is a 2-byte integer. Currently, there is only one predefined value:

GMR_$PICK_ELEMENT
Picks the nth element that crosses the pick aperture and also satisfies the pick criteria. The value of n is defined by the pick index argument of GMR_$PICK.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

GMR_$PICK_SET_METHOD sets the pick method for a specific viewport.

The pick criteria follows:

At display-time, the structure mask is tested against the viewport visibility mask and the viewport pick mask. The structure value is tested against the viewport pick range and the viewport visibility range.

The structure is visible under these conditions:

1. The structure value must be within the viewport's visibility range, inclusive.
2. The logical AND of the structure mask and the viewport visibility mask must be nonzero.

The structure is pickable under these conditions:

1. The structure must meet the above visibility criteria.
2. The structure value must be within the viewport's pick range, inclusive.
3. The logical AND of the structure mask and the viewport pick mask must be nonzero.

GMR_$PICK_SET_APERTURE_SIZE

Specifies the width, height, and depth of the pick aperture for a particular viewport.

FORMAT

```
GMR_$PICK_SET_APERTURE_SIZE (viewport_id, width, height, depth, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

width
The width of the pick aperture, in GMR_$F_T format. This parameter is a real value in logical device coordinates.

height
The height of the pick aperture, in GMR_$F_T format. This parameter is a real value in logical device coordinates.

depth
The depth of the pick aperture, in GMR_$F_T format. This parameter is a real value in logical device coordinates.

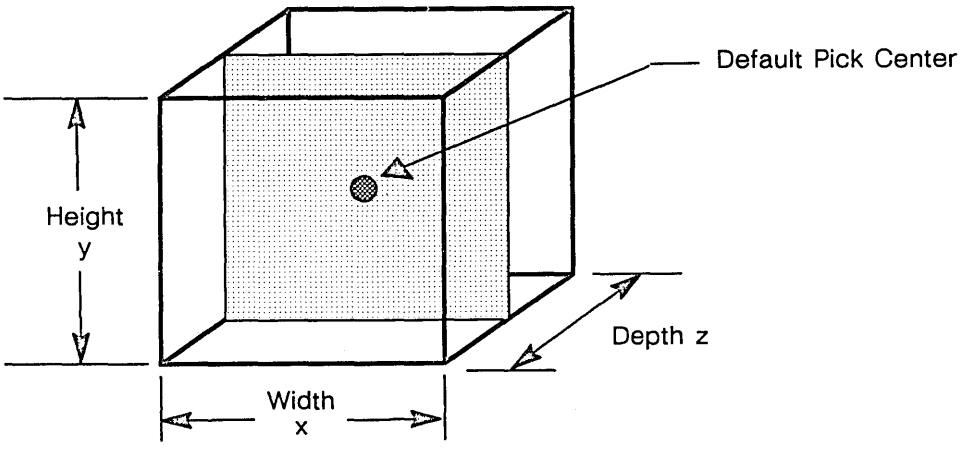
OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

GMR_$PICK searches for structures/elements that fall into the following region:

```
(center.x - 0.5*width  to  center.x + 0.5*width,
 center.y - 0.5*height to  center.y + 0.5*height,
 center.z - 0.5*depth  to  center.z + 0.5*depth)
```



The pick aperture is initialized to center (0.0, 0.0, 0.0) in logical device coordinates. The default depth is 1. The default height and width is 0.1.

Use GMR_$PICK_INQ_APERTURE to retrieve the pick aperture for a viewport.

GMR_$PICK_SET_ECHO_METHOD

Sets the pick echo method for a viewport.

FORMAT

```
GMR_$PICK_SET_ECHO_METHOD (viewport_id, method, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

method
The pick echo method for the viewport, in GMR_$PICK_ECHO_METHOD_T format. This parameter is a 2-byte integer. Possible values are the following:

GMR_$PICK_ECHO_NONE
Does not echo the picked element. This is the default echo type.

GMR_$PICK_ECHO_ABLOCK
Echo the element using the echo method described by the highlight attribute block associated with the viewport.

GMR_$PICK_ECHO_BBOX
Draws the bounding box around the structure containing the picked element. The 3D GMR package automatically maintains a bounding box for each element (and structure).

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The 3D GMR package defines two types of echoing -- pick echo and instance echo. Pick echo is the system response to a pick operation. Instance echo uses an application-supplied instance path that does not have to come from a pick operation (see GMR_$INSTANCE_ECHO).

Use GMR_$VIEWPORT_SET_HILIGHT_ABLOCK to set the highlight attribute block for a viewport.

If the ablock method is specified and no highlighting attribute block is assigned to the viewport, the default highlighting attribute block is used.

Use GMR_$PICK_INQ_ECHO_METHOD to return the current pick echo method for a viewport.

Note that a pick echo echos only the element at the end of the instance path identified by GMR_$PICK. To echo an entire subtree, use GMR_$INSTANCE_ECHO.

GMR_$PICK_SET_METHOD

Specifies the pick method for a particular viewport.

FORMAT

```
GMR_$PICK_SET_METHOD (viewport_id, method, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

method
The method used for picking, in GMR_$PICK_METHOD_T format. This parameter is a 2-byte integer. Currently, there is only one predefined value:

GMR_$PICK_ELEMENT
Picks the nth element that crosses the pick aperture and also satisfies the pick criteria. The pick index argument of GMR_$PICK defines the value of n.

If n is greater than 1, performance may be affected because the method may require multiple passes through the metafile.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The pick criteria follows:

At display-time, the structure mask is tested against the viewport visibility mask and the viewport pick mask. The structure value is tested against the viewport pick range and the viewport visibility range.

The structure is visible under these conditions:

1. The structure value must be within the viewport's visibility range, inclusive.
2. The logical AND of the structure mask and the viewport visibility mask must be nonzero.

The structure is pickable under these conditions:

1. The structure must meet the above visibility criteria.
2. The structure value must be within the viewport's pick range, inclusive.

3. The logical AND of the structure mask and the viewport pick mask must be nonzero.

Use GMR_$STRUCTURE_SET_VALUE_MASK, GMR_$VIEWPORT_SET_PICK, and GMR_$VIEWPORT_SET_VISIBILITY to set the pick criteria.

Also refer to GMR_$ADD_NAME_SET for an additional method of specifying visibility and pick eligibility.

GMR_$PICK_INQ_METHOD returns the pick method for a specific viewport.

GMR_$PRINT_DISPLAY

Creates a POSTSCRIPT file from the entire 3D GMR display.

FORMAT

```
GMR_$PRINT_DISPLAY (name, namelength, print_style, paper_width,
                    paper_height, status)
```

INPUT PARAMETERS

name
The name of the POSTSCRIPT file to be created, in NAME_$PNAME_T format. This parameter is an array of up to 256 characters.

namelength
The length of the name in GMR_$I_T format. This parameter is a 2-byte integer.

print_style
The style, in GMR_$PRINT_STYLE_T format. For this release, the only value is GMR_$POSTSCRIPT. This parameter is a 2-byte integer.

paper_width
The width of the paper, in inches, that the file will be printed on, in GMR_$F_T format. This parameter is a 4-byte real number.

paper_height
The height of the paper, in inches, that the file will be printed on, in GMR_$F_T format. This parameter is a 4-byte real number.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

You can print a POSTSCRIPT file on a printer that supports POSTSCRIPT (for example the Genicom Model 3404 dot matrix printer with a POSTSCRIPT license or an Apple LaserWriter).

Use the following procedure to create and print a POSTSCRIPT file:

1. At display-time, use GMR_$PRINT_DISPLAY or GMR_$PRINT_VIEWPORT to create a POSTSCRIPT file. For example:

```
GMR_$PRINT_DISPLAY('test plot', 9, gmr_$postscript, 8.5, 11.0,
                   status);
```

2. Use the PRF command to print the file. For example:

```
$ prf test plot -pr <printername> -trans
```

A quarter inch margin all around the paper is used. The picture is centered on the paper and fills as much of either the x or the y direction (within the margins) as possible while maintaining the aspect ratio of the screen display. The aspect ratio is the ratio of height to width.

Currently, the picture is positioned so that the longest edge of the paper is vertical. For example, on an 8.5x11 sheet of paper, the 11-inch side is vertical.

To print a smaller picture on the same size paper, specify a smaller paper width and height. In this case the picture will not be centered on the page since 3D GMR uses the lower left-hand corner of the specified paper size as the origin.

The POSTSCRIPT file is an ASCII file. Do not edit the file unless you are familiar with POSTSCRIPT. See *Programing With DOMAIN 3D Graphics Metafiles Resource* for more information.

Use GMR_$PRINT_VIEWPORT to create a POSTSCRIPT file of a single viewport.

GMR_$PRINT_VIEWPORT

Creates a POSTSCRIPT file from a single, specified viewport.

FORMAT

```
GMR_$PRINT_VIEWPORT (viewport_id, name, namelength, print_style, paper_width,
                     paper_height, status)
```

INPUT PARAMETERS

viewport_id
The identification number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

name
The name of the POSTSCRIPT file to be created, in NAME_$PNAME_T format. This parameter is an array of up to 256 characters.

namelength
The length of the name in GMR_$I_T format. This parameter is a 2-byte integer.

print_style
The style, in GMR_$PRINT_STYLE_T format. For this release, the only possible value is GMR_$POSTSCRIPT. This parameter is a 2-byte integer.

paper_width
The width of the paper, in inches, that the file will be printed on, in GMR_$F_T format. This parameter is a 4-byte real number.

paper_height
The height of the paper, in inches, that the file will be printed on, in GMR_$F_T format. This parameter is a 4-byte real number.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

See the Usage section under GMR_$PRINT_DISPLAY.

GMR_$REMOVE_NAME_SET

Inserts an attribute element into a structure. The element removes names from the current name set.

FORMAT

```
GMR_$REMOVE_NAME_SET (n_names, name_set, status)
```

INPUT PARAMETERS

n_names
The number of names to be removed from the current name set. This parameter is a 2 byte integer.

name_set
The list of names to be removed, in GMR_$NAME_SET_T format. Each name is in the range [1, GMR_$MAX_NAME_ELEMENT], inclusive.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

See GMR_$ADD_NAME_SET for an example.

GMR_$REPLACE_INQ_FLAG

Returns the current value of the replace flag.

FORMAT

```
GMR_$REPLACE_INQ_FLAG (yes_no, status)
```

OUTPUT PARAMETERS

yes_no
A Boolean value indicating whether or not the replace flag is set. True indicates that the flag is set (i.e., new elements replace the current element); false indicates that the flag is cleared (i.e., new elements are inserted after the current element). The default value is false.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$REPLACE_SET_FLAG to set the replace flag value.

GMR_$REPLACE_SET_FLAG

Sets or clears a flag that causes subsequent elements to replace the current element rather than being inserted after the current element.

FORMAT

```
GMR_$REPLACE_SET_FLAG (yes_no, status)
```

INPUT PARAMETERS

yes_no
A Boolean value indicating whether or not the replace flag is set. Use true to set the flag (i.e., new elements replace the current element); use false to clear the flag (i.e., new elements are inserted after the current element).

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The default value is false (i.e., new elements are inserted after the current element).

When the replace flag is set and the program calls a routine that creates an element (e.g., GMR_$F3_TEXT), the new element replaces the current element.

GMR_$STRUCTURE_CLOSE

Closes the current structure, saving revisions or not.

FORMAT

```
GMR_$STRUCTURE_CLOSE (save, status)
```

INPUT PARAMETERS

save
A Boolean (logical) value that indicates whether or not to save revisions. Set to true to save revisions; set to false not to save revisions.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

You must explicitly set save to true in order to have revisions saved. Do not assume that it is true by default.

GMR_$STRUCTURE_COPY

Copies the entire contents of another structure into the current structure.

FORMAT

```
GMR_$STRUCTURE_COPY (file_id, structure_id, status)
```

INPUT PARAMETERS

file_id
The identification number of the file that contains the structure you want to copy, in GMR_$FILE_ID_T format.

structure_id
The identification number of the structure to be copied, in GMR_$STRUCTURE_ID_T format. This parameter is a 4-byte integer.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

GMR_$STRUCTURE_COPY copies the entire contents of the specified structure into the current structure after the current element. The current element is set equal to the last copied element.

Use file_id to copy structures from one open file to another. To copy within the same file, set file_id to the current file_id.

If you are copying from one file to another, you can only copy structures that contain no references to other structures in the file being copied from (that is, there may be no instance elements in that structure).

Use the following procedure to make a new structure named "newcopy", which is an exact copy of an existing structure. The identification of the existing structure is source_seg_id:

```
GMR_$STRUCTURE_CREATE('newcopy', 7, structure_id, status);
GMR_$STRUCTURE_COPY(file_id, source_seg_id, status)
GMR_$STRUCTURE_CLOSE(true, status)
```

You can edit and instance the two copies independently.

GMR_$STRUCTURE_CREATE

Creates a new structure and assigns it a structure identification number and (optionally) an application-supplied name.

FORMAT

```
GMR_$STRUCTURE_CREATE (name, name_length, structure_id, status)
```

INPUT PARAMETERS

name
The pathname of the structure, in NAME_$PNAME_T format. This parameter is a character string.

name_length
The number of characters in the pathname. This parameter is a 2-byte integer. The constant GMR_$MAX_STRUCTURE_NAME_LENGTH sets the maximum name length.

If the name_length is 0, the structure does not receive a unique name.

OUTPUT PARAMETERS

structure_id
The identification number assigned to the structure, in GMR_$STRUCTURE_ID_T format. This parameter is a 4-byte integer.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

You must close the current structure before creating a new structure.

When a structure is created, its structure value is set to 255. Its structure mask is set to all ones. Structure value and structure mask are used to determine visibility and pick eligibility.

For a structure name, you can use any collection of byte values of length 1 through 12. Trailing blanks in structure names are not discarded.

To use an integer for a structure name, use equivalence features of your programming language and equivalence integers to characters.

The 3D GMR package truncates names longer than GMR_$MAX_STRUCTURE_NAME_LENGTH to that length.

Structures in the same file must have different structure names. Note that "STRUC" and "struc" are different structure names; the comparison is case-sensitive.

Verification that each name is unique carries a performance penalty. Therefore, you have the option of not naming structures (you use the structure ID to refer to structures). To create an unnamed structure, set the value for name to 0:
GMR_$STRUCTURE_CREATE ('', 0, struc_id, status).

The routine GMR_$INSTANCE_TRANSFORM_FWD_REF combines the features of GMR_$STRUCTURE_CREATE and GMR_$INSTANCE_TRANSFORM.

GMR_$STRUCTURE_DELETE

Deletes the current open structure.

FORMAT

```
GMR_$STRUCTURE_DELETE (status)
```

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

A structure must be open to delete it.

The 3D GMR package reassigns the identification number of deleted structures, assigning the smallest unused number first.

A structure cannot contain references to a deleted structure. Therefore, you must delete all instances from containing structures before you delete the open structure. However, a structure which contains instances can be deleted.

For example, consider the following four structures:

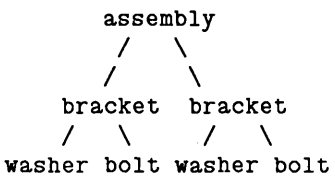
```
gmr_$structure_create('bolt', 4, bolt_id, status);
 .
 .
 .
gmr_$structure_close(TRUE, status);


gmr_$structure_create('washer', 6, washer_id, status);
 .
 .
 .
gmr_$structure_close(TRUE, status);


gmr_$structure_create('bracket', 7, bracket_id, status);
 .
 .
 .
 gmr_$instance_transform(bolt_id,mat1,status);
 gmr_$instance_transform(washer_id,mat1,status);
 gmr_$instance_transform(bolt_id,mat2,status);
 gmr_$instance_transform(washer_id,mat2,status);
 gmr_$structure_close(TRUE, status);
```

```
gmr_$structure_create('assembly', 8, assembly_id, status);
gmr_$instance_transform(bracket_id,mat3,status);
gmr_$instance_transform(bracket_id,mat4,status);
gmr_$structure_close(TRUE, status);
```

The above fragment creates a hierarchical metafile that can be represented as follows:



You can delete the structures as follows:

1. You can delete assembly without making any changes to the other structures.

2. You can delete bracket only after you take out both references to it in assembly. You don't have to change washer or bolt to be able to delete bracket.

3. You can delete washer or bolt only after you have taken out the references to them in bracket.

GMR_$STRUCTURE_ERASE

Deletes all elements in the current structure.

FORMAT

```
GMR_$STRUCTURE_ERASE (status)
```

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

A structure must be open to erase it.

After execution of this routine, the current structure contains no elements and the element index is set to 0.

You can erase an instanced structure.

GMR_$STRUCTURE_INQ_BOUNDS

Returns the limits of the bounding box that encloses a structure and any subtrees that the structure calls.

FORMAT

```
GMR_$STRUCTURE_INQ_BOUNDS(structure_id, bounds, status)
```

INPUT PARAMETERS

structure_id
The structure ID, in GMR_$STRUCTURE_ID_T format.

OUTPUT PARAMETERS

bounds
The limits of the rectangular parallelepiped that encloses the structure and any subtrees that the structure calls, in GMR_$F3_LIMITS_T format. This parameter is an array of six real values in the following order: xmin, xmax, ymin, ymax, zmin, and zmax.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The 3D GMR package automatically creates a bounding box around each structure. This is the box that is used when you specify bounding box echoing (see GMR_$PICK_SET_ECHO_METHOD). The bounding box is also used by the 3D GMR package for clip testing at display-time.

GMR_$STRUCTURE_INQ_COUNT

Returns the number of structures in the current metafile and a structure number guaranteed to be greater than or equal to the largest structure number.

FORMAT

```
GMR_$STRUCTURE_INQ_COUNT (count, max_structure_id, status)
```

OUTPUT PARAMETERS

count
The number of structures in the metafile. This parameter is a 4-byte integer.

max_structure_id
A number greater than or equal to the largest structure ID in the file, in GMR_$STRUCTURE_ID_T format. This parameter is a 4-byte integer.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

When you retrieve the count and maximum structure ID, you can then examine every structure by checking structure numbers from 0 to this maximum value (0 is used).

To determine the number of instance elements that invoke a particular structure, see GMR_$STRUCTURE_INQ_INSTANCES.

GMR_$STRUCTURE_INQ_ID

Returns the structure identification number of the named structure.

FORMAT

```
GMR_$STRUCTURE_INQ_ID (name, name_length, structure_id, status)
```

INPUT PARAMETERS

name
The pathname of the structure, in NAME_$PNAME_T format. This parameter is an array of up to 256 characters.

name_length
The number of characters in the pathname. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

structure_id
The identification number assigned to the structure of specified name, in GMR_$STRUCTURE_ID_T format. This parameter is a 4-byte integer.

In creating instances of (i.e.,references to) this structure within other structures, you must use the returned structure identification number.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

This routine only searches the current file to identify the structure number of the named structure.

Use GMR_$STRUCTURE_INQ_NAME to return the name of a structure.

GMR_$STRUCTURE_INQ_INSTANCES

Returns both the number of instance elements that invoke a particular structure and the maximum number of levels of instancing that occur beneath the structure.

FORMAT

```
GMR_$STRUCTURE_INQ_INSTANCES (structure_id, n_instances, max_depth, status)
```

INPUT PARAMETERS

structure_id
The identification number of the structure, in GMR_$STRUCTURE_ID_T format. This parameter is a 4-byte integer.

OUTPUT PARAMETERS

n_instances
The number of instance elements elsewhere in the file that invoke the given structure. This parameter is a 4-byte integer.

max_depth
The maximum number of levels of instancing that occur beneath the given structure. For example, a structuure containing *no* instance elements has a max_depth of 0. A structure containing instance elements that refer *only* to structures with *no* instances has a max_depth of 1.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Since more than one instance element of the named structure may lie in another given structure, the value of n_instances is not the same as the number of other structures that instance this particular structure.

The value of max_depth tells you something about what lies *below* the specified structure in the metafile. Max_depth can be expressed mathematically as follows:

```
max_depth of structure A := MAXIMUM OF ( 0,
          1 + max_depth of first structure instanced by A,
          1 + max_depth of second structure instanced by A,
          etc. );
```

GMR_$STRUCTURE_INQ_NAME

Returns the name of the structure with the specified structure identification number.

FORMAT

```
GMR_$STRUCTURE_INQ_NAME (structure_id, name, name_length, status)
```

INPUT PARAMETERS

structure_id
The identification number assigned to the structure, in GMR_$STRUCTURE_ID_T format. This parameter is a 4-byte integer.

OUTPUT PARAMETERS

name
The pathname of the structure, in NAME_$PNAME_T format. This parameter is an array of up to 256 characters.

name_length
The number of characters in the pathname. This parameter is a 2-byte integer.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

This routine only searches the current open file to identify the name with the given structure ID.

Use GMR_$STRUCTURE_INQ_ID to return the ID of a structure given its name.

GMR_$STRUCTURE_INQ_OPEN

Returns the identification number of the open structure.

FORMAT

```
GMR_$STRUCTURE_INQ_OPEN (structure_id, status)
```

OUTPUT PARAMETERS

structure_id
The identification number assigned to the structure, in GMR_$STRUCTURE_ID_T format. This parameter is a 4-byte integer.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

GMR_$STRUCTURE_INQ_TEMPORARY

Returns whether the specified structure is temporary or not.

FORMAT

```
GMR_$STRUCTURE_INQ_TEMPORARY (structure_id, temporary, status)
```

INPUT PARAMETERS

structure_id
The identification number of the structure for which the temporary/permanent status is to be retrieved, in GMR_$STRUCTURE_ID_T format. This parameter is a 4-byte integer.

OUTPUT PARAMETERS

temporary
A Boolean (logical) value that indicates whether or not the structure is temporary. A value of true indicates that the structure is temporary; false indicates that the structure is permanent.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The following rules apply to temporary structures when you close the file:

1. Temporary structures that are not instanced by other structures are deleted.

2. Temporary structures that are instanced by permanent structures are not deleted.

3. A temporary structure that is instanced by other temporary structures is deleted *if* all the temporary structures that instance it are deleted.

GMR_$STRUCTURE_INQ_VALUE_MASK

Returns the value and mask of a structure. These are used to determine visibility and pick eligibility.

FORMAT

```
GMR_$STRUCTURE_INQ_VALUE_MASK (structure_id, value, mask, status)
```

INPUT PARAMETERS

structure_id
The identification number of the structure, in GMR_$STRUCTURE_ID_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

value
The structure value, in GMR_$STRUCTURE_VALUE_T format. This parameter is a 4-byte integer.

mask
The structure mask, in GMR_$STRUCTURE_MASK_T format. This parameter is a 4-byte integer.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

See GMR_$STRUCTURE_SET_VALUE_MASK for more information.

GMR_$STRUCTURE_OPEN

Reopens an existing structure and optionally creates a backup version.

FORMAT

```
GMR_$STRUCTURE_OPEN (structure_id, back_up, status)
```

INPUT PARAMETERS

structure_id
The identification number of the structure to open, in GMR_$STRUCTURE_ID_T format. This parameter is a 4-byte integer.

back_up
A boolean value that specifies whether or not to create a backup version of the structure before opening it.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Within each open metafile, you must close the current structure before opening another structure.

Use GMR_$STRUCTURE_INQ_OPEN to retrieve the identification number of the current open structure.

You can open a structure in a file that is open for read access. However, in this case you can only set the element index and call the various inquiry routines. You cannot change the contents of the structure. When you open a structure the element index is set to 0 by default.

If back_up is FALSE, you cannot call GMR_$STRUCTURE_CLOSE with the save parameter = FALSE (GMR cannot restore the original contents of the structure unless it saves a backup version when the structure is originally opened).
GMR_$STRUCTURE_CLOSE will *not* restore the original contents of the structure, and the value of the save parameter will be ignored.

If the file containing the structure is opened in read-only mode, back_up versions are *not* created and the back_up parameter is ignored (GMR cannot create a backup version if it cannot write to the file).

If a structure is frequently opened and appended to, free space is fragmented and the file grows. The back_up option is *not* recommended except when a structure is opened for a lengthy period of interactive editing which the user may want to retract.

GMR_$STRUCTURE_SET_NAME

Renames an existing structure.

FORMAT

```
GMR_$STRUCTURE_SET_NAME (structure_id, name, name_length, status)
```

INPUT PARAMETERS

structure_id
The identification number of the structure to rename, in GRM_$STRUCTURE_ID_T format. This parameter is a 4-byte integer.

The structure number remains the same when you rename the structure.

name
The new name of the structure, in NAME_$PNAME_T format. This parameter is an array of up to 256 characters.

name_length
The number of characters in the new name of the structure. This is a 2-byte integer. Currently, the 3D GMR package truncates structure names to 12 characters.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

To find the structure_id of an existing structure for which you know the name, use GMR_$STRUCTURE_INQ_ID.

If another structure already has the new name, an error code is returned in the status parameter, and the old name is not changed.

Verification that each name is unique carries a performance penalty. Therefore, you have the option of not naming structures (you use the structure ID to refer to structures). To remove the name of a structure, set the length of the name to 0: GMR_$STRUCTURE_SET_NAME(", 0, struc_id, status).

Use GMR_$STRUCTURE_SET_NAME to name an unnamed structure or to remove the name of a structure.

GMR_$STRUCTURE_SET_TEMPORARY

Makes the specified structure temporary or not. The 3D GMR package deletes temporary structures when you close the file.

FORMAT

```
GMR_$STRUCTURE_SET_TEMPORARY (structure_id, temporary, status)
```

INPUT PARAMETERS

structure_id
The identification number of the structure to make temporary, in GMR_$STRUCTURE_ID_T format. This parameter is a 4-byte integer.

temporary
A Boolean value that indicates whether the structure is temporary. Set to true to make temporary; set to false to make permanent.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

By default, a newly created structure is permanent (temporary = false).

A temporary structure is useful for picture data that you want to display now but not store for future use (e.g., a superimposed grid).

The following rules apply to temporary structures when you close the file:

1. Temporary structures that are not instanced by other structures are deleted.

2. Temporary structures that are instanced by permanent structures are not deleted.

3. A temporary structure that is instanced by other temporary structures is deleted *if* all the temporary structures that instance it are deleted.

GMR_$STRUCTURE_SET_VALUE_MASK

Sets the structure value and structure mask. These are used to determine visibility and pick eligibility.

FORMAT

```
GMR_$STRUCTURE_SET_VALUE_MASK (structure_id, value, mask, status)
```

INPUT PARAMETERS

structure_id
The identification number assigned to the structure, in GMR_$STRUCTURE_ID_T format. This parameter is a 4-byte integer.

value
The structure value, in GMR_$STRUCTURE_VALUE_T format. This parameter is a 4-byte integer.

mask
The structure mask, in GMR_$STRUCTURE_MASK_T format. This parameter is a 4-byte integer.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

At display-time, 3D GMR package tests the structure mask against the viewport visibility mask and the viewport pick mask. The package also tests the structure value against the viewport visibility range and the viewport pick range.

The structure is visible under these conditions:

1. The structure value must be within the viewport's visibility range, inclusive.
2. The logical AND of the structure mask and the viewport visibility mask must be nonzero.

The structure is pickable under these conditions:

1. The structure must meet the above visibility criteria.
2. The structure value must be within the viewport's pick range, inclusive.
3. The logical AND of the structure mask and the viewport pick mask must be nonzero.

Use GMR_$VIEWPORT_SET_VISIBILITY to set the visibility range and mask of a viewport.

Use GMR_$VIEWPORT_SET_PICK to set the pick range and mask of a viewport.

GMR_$TAG

Inserts a comment into the current open structure.

FORMAT

```
GMR_$TAG (tag, tag_length, status)
```

INPUT PARAMETERS

tag
The text string to store in the file, in GMR_$STRING_T format. This is an array of characters.

tag_length
The length of the string. This parameter is a 4-byte integer. The only limits on the length of the string are those imposed by the size of the metafile and the applications program.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use tags to include application specific data in a metafile.

Use GMR_$INQ_TAG to return the text stored for the current GMR_$TAG element.

GMR_$TAG_LOCATE

Searches for the specified tag in the specified range of structures and returns the structure ID of the lowest numbered structure in which the tag is found.

FORMAT

```
GMR_$TAG_LOCATE (text_locate, text_length, structure_start, structure_stop,
                 index_start, index_stop, character_start, character_stop,
                 tag_structure, tag_index, tag_character, status)
```

INPUT PARAMETERS

text_locate
The string to be searched for, in GMR_$STRING_T format. This parameter is an array of up to 120 characters.

The string to be matched is passed through the pathname wildcard parser, as described in the *DOMAIN System Command Reference*. To guarantee noninterference with the wildcard parser, you may place an escape character (@) between every byte of the string you wish to search for.

text_length
The length of the text_locate string. This parameter is a 4-byte integer.

Because this string is passed through the wild card parser, the text length may be different from the length of the matching string.

structure_start
The smallest structure identification number to search, in GMR_$STRUCTURE_ID_T format. This parameter is a 4-byte integer. An integer value of 0 defaults to the first structure in the metafile.

structure_stop
The largest structure identification number to search, in GMR_$STRUCTURE_ID_T format. This parameter is a 4-byte integer. An integer value of 0 defaults to the last structure in the metafile.

index_start
The index of the first element in the first structure to search. This parameter is a 4-byte integer. An integer value of 0 defaults to the first element in the structure.

index_stop
The index of the last element in the last structure to search. This is a 4-byte integer. An integer value of 0 defaults to the last element in the structure.

character_start
An offset into the tag text, if any, associated with the first element in the first structure to be searched. This parameter is a 4-byte integer. An integer value of 0 defaults to the beginning of the tag text (see Usage).

character_stop
Same as character_start, except that 0 defaults to the end of the tag text. This parameter is a 4-byte integer.

tag_structure
The identification number of the structure in which the tag was found, in GMR_$STRUCTURE_ID_T format. This is a 4-byte integer.

tag_index
The element index of the tag. This parameter is a 4-byte integer.

tag_character
An offset into the tag text showing where the pattern was found. This parameter is a 4-byte integer.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

To find all occurrences of a tag, you must make successive calls to GMR_$TAG_LOCATE.

The wild card parser and regular expression handler cannot search more than 32K characters of text at one time. If you need to use tags larger than 32K, place any possible search keys in the first 32K of the tag.

The character_start and character_stop parameters let you specify precisely where the read takes place. For example, you can use these parameters to continue searching a given tag for more than one occurence of a given search string. On each successive call, specify character_start as 1+ the tag_character value returned in the previous call.

GMR_$TERMINATE

Terminates the 3D GMR package and closes the display.

FORMAT

```
GMR_$TERMINATE (status)
```

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Any open structures are closed, and revisions are saved.

Any open files are closed. Revisions to these files are saved.

GMR_$TEXT

Inserts a primitive element into the current open structure. The element defines and positions a text string.

FORMAT

```
GMR_$TEXT (string, string_length, position, status)
```

INPUT PARAMETERS

string
The string to insert in the text element, in GMR_$STRING_T format. This parameter is an array of up to GMR_$MAX_STRING_LENGTH characters.

string_length
The length of the string. This parameter is a 2-byte integer.

position
The anchor point of the text string, in GMR_$F3_POINT_T format. This is a point in modeling coordinates used to position the text.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The anchor point indicates the placement of text. The anchor may also specify where to clip text (see GMR_$TEXT_SET_ANCHOR_CLIP).

Use modeling coordinates to specify text location; but use viewing coordinates (same as world coordinates) to specify text height.

GMR_$TEXT_COLOR

Inserts an attribute element into the current open structure. The element sets the color ID used when rendering text.

FORMAT

```
GMR_$TEXT_COLOR (color_id, status)
```

INPUT PARAMETERS

color_id
The color identifier, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer in the range [0, GMR_$MAX_COLOR_ID], inclusive.

The default value is GMR_$TEXT_COLOR_DEF. This is equivalen to 1.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_INQ_TEXT_COLOR to retrieve the color of the current (GMR_$TEXT_COLOR) element.

GMR_$TEXT_EXPANSION

Inserts an attribute element into the current open structure. The element sets the expansion factor used when rendering text. Text expansion controls the ratio of height to width of text characters.

FORMAT

```
GMR_$TEXT_EXPANSION (expansion, status)
```

INPUT PARAMETERS

expansion

The text character expansion for this attribute block, in GMR_$TEXT_EXPANSION_T format. This is a 4-byte real value. This attribute controls the aspect ratio for the font. The default value is 1.0 which preserves the aspect ratio defined in the font.

Values greater than 1.0 create wider characters. Values less than 1.0 create thinner characters.

OUTPUT PARAMETERS

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$INQ_TEXT_EXPANSION to retrieve the expansion factor of the current (GMR_$TEXT_EXPANSION) element.

GMR_$TEXT_HEIGHT

Inserts an attribute element into the current open structure. The element sets the text height. Text height controls the actual size of text characters.

FORMAT

```
GMR_$TEXT_HEIGHT (height, status)
```

INPUT PARAMETERS

height

The text character height, in GMR_$TEXT_HEIGHT_T format. This is 4-byte real value in viewing coordinates (same as world coordinates). The default height is GMR_$TEXT_HEIGHT_DEF. This is equivalent to 0.01.

OUTPUT PARAMETERS

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$INQ_TEXT_HEIGHT to retrieve the text height setting of the current (GMR_$TEXT_HEIGHT) element.

GMR_$TEXT_INQ_ANCHOR_CLIP

Returns the mode for clipping text.

FORMAT

```
GMR_$TEXT_INQ_ANCHOR_CLIP (anchor_clip, status)
```

OUTPUT PARAMETERS

anchor_clip
A Boolean value that specifies whether or not clipping by anchor point (default) is on.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$TEXT_SET_ANCHOR_CLIP to set anchor point clipping.

GMR_$TEXT_INTEN

Inserts an attribute element into the current open structure. The element sets the text intensity.

FORMAT

```
GMR_$TEXT_INTEN (intensity, status)
```

INPUT PARAMETERS

intensity

The value of the text intensity, in GMR_$INTEN_T format. This parameter is a 4-byte real value in the range [0.0, 1.0], inclusive.

The default value is GMR_$TEXT_INTEN_DEF. This is equivalent to 1.0.

OUTPUT PARAMETERS

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$INQ_TEXT_INTEN to retrieve the intensity of the current (GMR_$TEXT_INTEN) element.

GMR_$TEXT_PATH

Inserts an attribute element into the current open structure. The element sets the angle of the text path. Text path determines where the second and subsequent characters in a text string are placed.

FORMAT

```
GMR_$TEXT_PATH (angle, status)
```

INPUT PARAMETERS

angle

The angle that determines where the second and subsequent characters in a string are placed, in GMR_$TEXT_PATH_T format. This parameter is a 4-byte real value.

An angle of 0.0 radians is to the right of the up vector. Angles greater than 0.0 radians are measured counterclockwise from the 0.0 radian position.

The default angle is GMR_$TEXT_PATH_DEF. This is equivalent to 0.0.

OUTPUT PARAMETERS

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

For convenience, use the following default values:

```
GMR_$TEXT_PATH_RIGHT
GMR_$TEXT_PATH_UP
GMR_$TEXT_PATH_LEFT
GMR_$TEXT_PATH_DOWN
```

Use GMR_$INQ_TEXT_PATH to retrieve the text path of the current (GMR_$TEXT_PATH) element.

GMR_$TEXT_SET_ANCHOR_CLIP

Specifies whether text is clipped by anchor point.

FORMAT

```
GMR_$TEXT_SET_ANCHOR_CLIP (anchor_clip, status)
```

INPUT PARAMETERS

anchor_clip
A Boolean value that specifies whether clipping by anchor point (default) is on.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

When you clip by anchor point, the entire text string is clipped whenever the anchor point is outside the viewport. This is the default mode.

Each inserted text string has an anchor point on the first character (see GMR_$TEXT). The text path determines the anchor point's position as follows:

Text path	Anchor point position on first character
right	lower left
left	lower right
down	top middle
up	bottom middle

Clipping the entire text string whenever the anchor point is outside the viewport results in faster execution time. However, pieces of text inside the viewport that have anchor points outside the viewport are not displayed.

GMR_$TEXT_SLANT

Inserts an attribute element into the current open structure. The element sets the slant of text. A negative value produces a left slant. A positive value produces a right slant.

FORMAT

```
GMR_$TEXT_SLANT (slant, status)
```

INPUT PARAMETERS

slant

The amount that the top of the character is shifted, in GMR_$TEXT_SLANT_T format. This parameter is a 4-byte real value.

The amount is determined by multiplying the text attributes for slant, height, and expansion_factor. A value between 0.0 and 1.0 yields an italics-like character (slanting to the right). The default value is GMR_$TEXT_SLANT_DEF which is equivalent to 0.0 (no slant).

OUTPUT PARAMETERS

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$INQ_TEXT_SLANT to retrieve the text slant setting of the current (GMR_$TEXT_SLANT) element.

GMR_$TEXT_SPACING

Inserts an attribute element into the current open structure. The element sets the spacing between text characters.

FORMAT

```
GMR_$TEXT_SPACING (spacing, status)
```

INPUT PARAMETERS

spacing

The inter-character spacing, in GMR_$TEXT_SPACING_T format. This parameter is a 4-byte real value that defines spacing as a fraction of text height.

The default is GMR_$TEXT_SPACING_DEF which is equivalent to 0.0. This places each character next to the preceding character in the character path direction.

OUTPUT PARAMETERS

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

For more spacing between characters, make the spacing value positive. To have characters appear to overlay, make the spacing value negative.

Use GMR_$INQ_TEXT_SPACING to retrieve the spacing value set by the current (GMR_$TEXT_SPACING) element.

GMR_$TEXT_UP

Inserts an attribute element into the current open structure. The element specifies the up direction for text on the projection plane.

FORMAT

```
GMR_$TEXT_UP (up_vector, status)
```

INPUT PARAMETERS

up_vector
The up direction of text on the projection plane, in GMR_$TEXT_UP_T format. This parameter is a pair of real values in viewing coordinates (same as world coordinates). Both values cannot be zero.

The default direction is (GMR_$TEXT_UP_X_DEF, GMR_$TEXT_UP_Y_DEF). This is equivalent to (0.0, 1.0), which is the typical way to display text.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$INQ_TEXT_UP to retrieve the text up vector for the current (GMR_$TEXT_UP) element.

GMR_$VIEW_INQ_COORD_SYSTEM

Returns the coordinate system type (right- or left-handed) of the given viewport.

FORMAT

```
GMR_$VIEW_INQ_COORD_SYSTEM (viewport_id, coord_system, status)
```

INPUT PARAMETERS

viewport_id
The identification number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

coord_system
The handedness of the viewing coordinate system, in GMR_$COORD_SYSTEM_T format. The argument can have the value GMR_$COORD_LEFT, or GMR_$COORD_RIGHT. This parameter is a 2-byte integer.

status
Completion status, in STATUS_$T format. This data type is 4 bytes long. See the Data Types section for more information.

USAGE

The default is GMR_$COORD_RIGHT. See GMR_$VIEW_SET_COORD_SYSTEM.

GMR_$VIEW_INQ_HITHER_DISTANCE

Returns the N-coordinate of the near clipping plane.

FORMAT

```
GMR_$VIEW_INQ_HITHER_DISTANCE (viewport_id, hither_dist, status)
```

INPUT PARAMETERS

viewport_id
The identification number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

hither_dist
The N-coordinate of the hither (near) clipping plane, in GMR_$F_T format. This parameter is a real value. The reference point is N = 0 in UVN coordinate space.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The absolute value of the hither distance is the geometric distance between the view reference point and the hither clipping plane.

The default hither and yon distances are -1.0E10 and 1.0E10, respectively. This puts the reference point in the middle of the default view volume.

For a perspective projection, both hither and yon values must be positive in a left-handed UVN system and negative in a right-handed system.

GMR_$VIEW_INQ_OBLIQUE

Returns the foreshortening ratio and angle of receding lines of a specific viewport.

FORMAT

```
GMR_$VIEW_INQ_OBLIQUE (viewport_id, foreshorten, recede, status)
```

INPUT PARAMETERS

viewport_id
The identification number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

foreshorten
The foreshortening ratio, in GMR_$F_T format. When you have an oblique projection, receding lines are scaled by this amount.

recede
The angle (in radians) of receding lines, in GMR_$F_T format.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

For elevation oblique projections, the receding angle is measured counterclockwise from the positive U-axis in the viewing coordinate system.

For plan oblique projections, the receding angle is measured counterclockwise from the horizontal ("right") direction at which the U axis is displayed.

GMR_$VIEW_INQ_PROJECTION_TYPE

Returns the type of projection used in a specific viewport.

FORMAT

```
GMR_$VIEW_INQ_PROJECTION_TYPE (viewport_id, proj_type, status)
```

INPUT PARAMETERS

viewport_id
The identification number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

proj_type
The type of projection, in GMR_$PROJECTION_T format. One of the following predefined values:

GMR_$PERSPECTIVE
Gives a perspective effect centered at any point within the world coordinate system. The view volume is in the shape of a frustum.

GMR_$ORTHOGRAPHIC
Standard parallel projection. The view volume is in the shape of a rectangular parallelepiped.

GMR_$PLAN_OBLIQUE
Parallel projection using a foreshortening ratio and a receding angle. The foreshortening ratio specifies how much any lines perpendicular to the view plane are foreshortened in projection. The receding angle is the angle between the U axis and the horizontal. Measure the receding angle counterclockwise from the horizontal ("right") direction at which the U axis is displayed. Receding lines are displayed vertically on the screen.

GMR_$ELEV_OBLIQUE
Parallel projection using a foreshortening ratio and a receding angle. The receding angle specifies the angle of receding lines relative to the positive U-axis. This is the direction on the view plane onto which the positive gaze direction is projected.

The default is orthographic.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$VIEW_SET_PROJECTION_TYPE to establish the projection type for a viewport.

GMR_$VIEW_INQ_REFERENCE_POINT

Returns the value of the viewing reference point for a given viewport.

FORMAT

```
GMR_$VIEW_INQ_REFERENCE_POINT (viewport_id, reference, status)
```

INPUT PARAMETERS

viewport_id
The identification number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

reference
The viewing reference point, in GMR_$F3_POINT_T format. This is the point in world coordinates at which the "eye" is positioned for perspective projections (that is, the center of projection). This is also the point from which the package measures other viewing parameters, such as hither and yon clipping distances, for both perspective and parallel projections.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

GMR_$VIEW_INQ_STATE

Returns the viewing parameter values for a viewport.

FORMAT

```
GMR_$VIEW_INQ_STATE (viewport_id, view_state, status)
```

INPUT PARAMETERS

viewport_id
The identification number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

view_state
The view state, in GMR_$VIEW_PARAM_BLOCK_T format. This block contains all the specified viewing parameters. The list below includes parameters along with the format, description, and byte offset. See the appropriate data type in this manual for more information about the structure of each type.

Parameter	Format	Description	Byte offset
reference	GMR_$F3_POINT_T	3 4-byte reals	0
normal	GMR_$F3_VECTOR_T	3 4-byte reals	12
up	GMR_$F3_VECTOR_T	3 4-byte reals	24
window	GMR_$F2_LIMITS_T	4 4-byte reals	36
h_dist	GMR_$F_T	4-byte real	52
y_dist	GMR_$F_T	4-byte real	56
v_dist	GMR_$F_T	4-byte real	60
fshorten	GMR_$F_T	4-byte real	64
recede	GMR_$F_T	4-byte real	68
proj_type	GMR_$PROJECTION_T	2-byte integer	72
coord_sys	GMR_$COORD_SYSTEM_T	2-byte integer	74

If you have not specified the normalizing matrix directly, the 3D GMR package derives it from GMR_$VIEW_PARAM_BLOCK_T.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

You can use this call as shorthand to inquire parameters for a view, modify some parameters, then set up a new view using GMR_$VIEW_SET_STATE.

The 3D GMR package returns an error if the view state was set by GMR_$VIEW_SET_TRANSFORM.

GMR_$VIEW_INQ_TRANSFORM

Returns a viewport's normalizing matrix.

FORMAT

```
GMR_$VIEW_INQ_TRANSFORM (viewport_id, matrix, projection_type, clip_zmin,
                         status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

matrix
The 4x3 normalizing matrix, in GMR_4X3_MATRIX_T. This matrix maps the world space view volume into the canonical view volume for the given projection type.

projection_type
The type of the projection for which the above matrix is intended, in GMR_$PROJECTION_T format.

clip_zmin
The distance from the coordinate origin to the front of the canonical perspective view volume, in GMR_$F_T format. This parameter is a real value. This value is meaningful only for perspective projections and is in the range [0.0, 1.0], exclusive.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

You can use this call to obtain the transformation matrix of an existing view and then map it to another view via GMR_$VIEW_SET_TRANSFORM.

GMR_$VIEW_INQ_UP_VECTOR

Returns the value of a viewport's viewing orientation.

FORMAT

```
GMR_$VIEW_INQ_UP_VECTOR (viewport_id, up, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

up
The vertical direction, in GMR_$F3_VECTOR_T format. The default is the positive y-axis of the world coordinate system.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

This vector determines the V-axis of the viewing coordinate system which cooresponds to the vertical axis on the screen.. That is, given the view plane normal direction (which establishes the N-axis of the viewing coordinate system), this vector determines the half-plane in world coordinates that contains the V axis. You need not specify the vector as a unit vector.

Use GMR_$VIEW_SET_UP_VECTOR to establish a viewport's up direction.

GMR_$VIEW_INQ_VIEW_DISTANCE

Returns the distance from the reference point to the viewing plane in a given viewport.

FORMAT

```
GMR_$VIEW_INQ_VIEW_DISTANCE (viewport_id, view_dist, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

view_dist
The distance from the reference point to the projection plane, in GMR_$F_T format. This parameter is a 4-byte real value.

The distance is measured along the view plane normal for both right-handed and left-handed coordinate systems. The default is -1.0

For perspective projections, the distance must be negative in a right-handed system and positive in a left-handed system.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

For perspective projections, the view distance alters the divergence of the projection rays between the center of projection (the reference point) and the window bounds of the view plane.

For elevation oblique and plan oblique projections, changing the view distance slides the projection across the view plane.

For orthographic projections, you can get the same results with any meaningful view distance. Since the projection is parallel to the N axis, the position of the view plane along the N axis is not important.

Use GMR_$VIEW_SET_VIEW_DISTANCE to establish the view distance of a viewport.

GMR_$VIEW_INQ_VIEW_PLANE_NORMAL

Returns the view plane normal for a specified viewport, in world coordinates.

FORMAT

```
GMR_$VIEW_INQ_VIEW_PLANE_NORMAL (viewport_id, normal, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

normal
A vector in world coordinates representing a normal to the viewing plane, in GMR_$F3_VECTOR_T format.

The view plane normal is the gaze direction in a left-handed viewing coordinate system, and points opposite the direction of gaze in a right-handed system. This direction is the normal to the projection plane, but does not need to be of unit length.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$VIEW_SET_VIEW_PLANE_NORMAL to set the view plane normal for a viewport.

GMR_$VIEW_INQ_WINDOW

Returns the minimum and maximum limits of the window on the viewing plane of a given viewport.

FORMAT

```
GMR_$VIEW_INQ_WINDOW (viewport_id, window, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

window
The min-max limits of the window on the viewing plane, in GMR_$F2_LIMITS_T format. This parameter is an array of 4-byte real values that specifies umin, umax, vmin, and vmax.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The part of the world that is visible through this window is displayed in the viewport. This window is defined at view distance from the reference point.

If the window's aspect ratio does not match that of the viewport, the image is stretched in either the x or y direction to fit the viewport exactly. The window's aspect ratio is the ratio of (umax-umin) to (vmax-vmin).

Use GMR_$VIEW_SET_WINDOW to establish the window for a viewport.

GMR_$VIEW_INQ_YON_DISTANCE

Returns the N-coordinate of the far clipping plane in a specified viewport.

FORMAT

```
GMR_$VIEW_INQ_YON_DISTANCE (viewport_id, yon_distance, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

yon_distance
The N-coordinate of the far clipping plane, in GMR_$F_T format. This parameter is a real value. The reference point is N = 0 in UVN coordinates.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The absolute value of the yon distance is the geometric distance between the view reference point and the far clipping plane.

The default hither and yon distances are -1.0E10 and 1.0E10 respectively. This puts the reference point in the middle of the default view volume.

For a perspective projection, both hither and yon values must be positive in a left-handed UVN system and negative in a right-handed system.

Use GMR_$VIEW_SET_YON_DISTANCE to establish the far clipping plane for a viewport.

GMR_$VIEW_SET_COORD_SYSTEM

Sets the coordinate system type of the given viewport.

FORMAT

```
GMR_$VIEW_SET_COORD_SYSTEM (viewport_id, coord_system, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

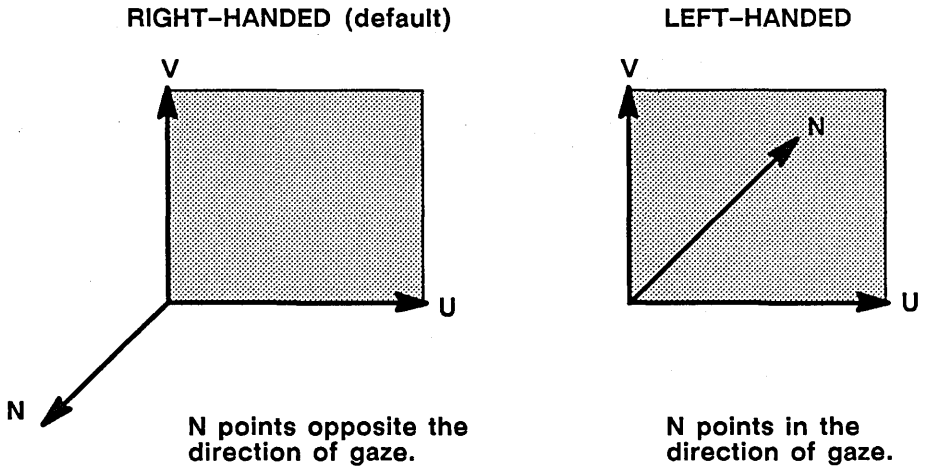
coord_system
The handedness of the viewing coordinate system, in GMR_$COORD_SYSTEM_T format. The argument can have the value GMR_$COORD_LEFT or GMR_$COORD_RIGHT. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This data type is 4 bytes long. See the Data Types section for more information.

USAGE

In a left-handed coordinate system, the view plane normal points in the direction of gaze (see illustration below). This direction is reversed in a right-handed system. The default is right-handed.



GMR_$VIEW_SET_HITHER_DISTANCE

Sets the N-coordinate of the near clipping plane in a specified viewport.

FORMAT

```
GMR_$VIEW_SET_HITHER_DISTANCE (viewport_id, hither_dist, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

hither_dist
The N-coordinate of the hither (near) clipping plane, in GMR_$F_T format. This parameter is a real value. The reference point is N = 0 in UVN coordinate space.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Only geometry that is between the hither and yon clipping plane (within the the view volume) is displayed.

The absolute value of the hither distance is the geometric distance between the view reference point and the hither clipping plane.

The default hither and yon distances are -1.0E10 and 1.0E10, respectively. This puts the reference point in the middle of the default view volume.

For a perspective projection, both hither and yon values must be positive in a left-handed UVN system and negative in a right-handed system.

GMR_$VIEW_SET_OBLIQUE

Sets the foreshortening ratio and the angle of receding lines for a specific viewport.

FORMAT

```
GMR_$VIEW_SET_OBLIQUE (viewport_id, foreshorten, recede, status)
```

INPUT PARAMETERS

viewport_id
The identification number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

foreshorten
The foreshortening ratio, in GMR_$F_T format. When you use oblique projection, receding lines are scaled by this amount.

recede
The angle (in radians) of receding lines when an oblique projection is being used, in GMR_$F_T format.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

These parameters do not apply to perspective projections.

For elevation oblique projections, the receding angle is measured counterclockwise from the positive U-axis in the viewing coordinate system.

For plan oblique projections, the receding angle is measured counterclockwise from the horizontal ("right") direction at which the U axis is displayed.

GMR_$VIEW_SET_PROJECTION_TYPE

Selects the type of viewing projection in a specified viewport.

FORMAT

```
GMR_$VIEW_SET_PROJECTION_TYPE (viewport_id, proj_type, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

proj_type
The type of projection, in GMR_$PROJECTION_T format. This parameter is a 2-byte integer. Use one of the following predefined values:

GMR_$PERSPECTIVE
Gives a perspective effect centered at any point within the world coordinate system. The view volume is in the shape of a frustum.

GMR_$ORTHOGRAPHIC
Standard parallel projection. The view volume is in the shape of a rectangular parallelepiped.

GMR_$PLAN_OBLIQUE
Parallel projection using a foreshortening ratio and a receding angle. The foreshortening ratio specifies how much any lines perpendicular to the view plane are foreshortened in projection. The receding angle is the angle between the U axis and the horizontal. Measure the receding angle counterclockwise from the horizontal ("right") direction at which the U axis is displayed. Receding lines are displayed vertically on the screen.

GMR_$ELEV_OBLIQUE
Parallel projection using a foreshortening ratio and a receding angle. The receding angle specifies the angle of receding lines relative to the positive U-axis. This is the direction on the view plane onto which the positive gaze direction is projected.

The default is orthographic.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

GMR_$VIEW_SET_REFERENCE_POINT

Sets a viewport's viewing reference point.

FORMAT

```
GMR_$VIEW_SET_REFERENCE_POINT (viewport_id, reference, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

reference
The viewing reference point, in GMR_$F3_POINT_T format. The default reference point is (0.0, 0.0, 0.0) of the world coordinate system.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

This is the position of the "eye" for perspective projections (that is, the center of projection in world coordinates). This is also the point from which to measure other viewing parameters, such as hither and yon clipping distances, for both perspective and parallel projections.

GMR_$VIEW_SET_STATE

Establishes all viewing parameters at once for a given viewport.

FORMAT

```
GMR_$VIEW_SET_STATE (viewport_id, view_state, status)
```

INPUT PARAMETERS

viewport_id

The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

view_state

View state, in GMR_$VIEW_PARAM_BLOCK_T format. This block sets all the viewing parameters. Parameters are listed below along with the format, description, and byte offset. Refer to the appropriate data type in this manual for more information about the structure of each type.

Parameter	Format	Description	Byte offset
reference	GMR_$F3_POINT_T	3 4-byte reals	0
normal	GMR_$F3_VECTOR_T	3 4-byte reals	12
up	GMR_$F3_VECTOR_T	3 4-byte reals	24
window	GMR_$F2_LIMITS_T	4 4-byte reals	36
h_dist	GMR_$F_T	4-byte real	52
y_dist	GMR_$F_T	4-byte real	56
v_dist	GMR_$F_T	4-byte real	60
fshorten	GMR_$F_T	4-byte real	64
recede	GMR_$F_T	4-byte real	68
proj_type	GMR_$PROJECTION_T	2-byte integer	72
coord_sys	GMR_$COORD_SYSTEM_T	2-byte integer	74

If you have not specified the normalizing matrix directly, the 3D GMR package derives it from GMR_$VIEW_PARAM_BLOCK_T.

OUTPUT PARAMETERS

status

Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$VIEW_INQ_STATE to retrieve the parameters from one viewport, and then use GMR_$VIEW_SET_STATE to transfer them to another viewport.

GMR_$VIEW_SET_TRANSFORM

Sets a viewport's normalizing matrix directly.

FORMAT

```
GMR_$VIEW_SET_TRANSFORM (viewport_id, matrix, projection_type, clip_zmin,
                         status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

matrix
The 4x3 normalizing matrix, in GMR_$4X3_MATRIX_T. This matrix is used to map the world space view volume into the canonical view volume for the given projection type.

projection_type
The type of the projection for which the above matrix is intended, in GMR_$PROJECTION_T format. This parameter is a 2-byte integer. Possible values are the following: GMR_$PERSPECTIVE, GMR_$ORTHOGRAPHIC, GMR_$PLAN_OBLIQUE, and GMR_$ELEV_OBLIQUE.

clip_zmin
The distance from the coordinate origin to the front of the canonical perspective view volume, in GMR_$F_T format. This parameter is a real value. This value is meaningful only for perspective projections and must be in the range [0.0, 1.0], exclusive.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. An error condition results when the projection type is GMR_$PERSPECTIVE and zmin is outside of the range [0.0, 1.0]. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The transformation matrix may either be application-generated or obtained from a previous view transform inquiry (GMR_$VIEW_INQ_TRANSFORM). If it is application generated, it must map world coordinates to the canonical viewing volume for correct results. There is no default as the defaults for the individual parameters determine the viewing transformation.

If you use GMR_$VIEW_SET_TRANSFORM, you cannot obtain the viewing parameters directly using GMR_$VIEW_INQ_STATE. That is, you cannot derive the parameters directly from the matrix.

GMR_$VIEW_SET_UP_VECTOR

Establishes viewing orientation in a given viewport.

FORMAT

```
GMR_$VIEW_SET_UP_VECTOR (viewport_id, up, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

up
The vertical direction, in GMR_$F3_VECTOR_T format. This parameter is a 4-byte, real value. The default direction is the positive y-axis of the world coordinate system.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. A zero up vector results in an error condition. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

This vector determines the V-axis of the viewing coordinate system which corresponds to the vertical axis on the screen. That is, given the view plane normal direction (which establishes the N-axis of the viewing coordinate system), this vector determines the half-plane in world coordinates that contains the V axis. You need not specify the vector as a unit vector.

GMR_$VIEW_SET_VIEW_DISTANCE

Sets the distance from the reference point to the viewing plane for a given viewport.

FORMAT

```
GMR_$VIEW_SET_VIEW_DISTANCE (viewport_id, view_dist, status )
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

view_dist
Specifies the distance from the reference point to the projection plane, in GMR_$F_T format. This parameter is a 4-byte real value.

The distance is measured along the view plane normal for both right-handed and left-handed coordinate systems. The default is -1.0.

For perspective projections, the distance must be negative in a right-handed system and positive in a left-handed system.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

For perspective projections, the view distance alters the divergence of the projection rays between the center of projection (the reference point) and the window bounds of the view plane.

For elevation oblique and plan oblique projections, changing the view distance slides the projection across the view plane.

For orthographic projections, you can get the same results with any meaningful view distance. Since the projection is parallel to the N axis, the position of the view plane along the N axis is not important.

Use GMR_$VIEW_SET_VIEW_DISTANCE to retrieve the view distance of a viewport.

GMR_$VIEW_SET_VIEW_PLANE_NORMAL

Sets the view plane normal for a given viewport, in world coordinates.

FORMAT

```
GMR_$VIEW_SET_VIEW_PLANE_NORMAL(viewport_id, normal, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

normal
A vector in world coordinates representing a normal to the viewing plane, in GMR_$F3_VECTOR_T format.

The view plane normal is the gaze direction in a left-handed viewing coordinate system, and points opposite the direction of gaze in a right-handed system. This direction is the normal to the projection plane, but does not need to be of unit length.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. A zero vector results in an error condition. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$VIEW_SET_VIEW_PLANE_NORMAL to set the view plane normal for a viewport.

GMR_$VIEW_SET_WINDOW

Establishes the window on the viewing plane of a viewport.

FORMAT

```
GMR_$VIEW_SET_WINDOW (viewport_id, window, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

window
The min-max limits of the window on the viewing plane, in GMR_$F2_LIMITS_T format. This parameter is an array of four 4-byte real values that specify umin, umax, vmin, and vmax.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The default window is a 1x1 square in world coordinates centered at the origin of the view plane.

The part of the world that is visible through this window is displayed in the viewport. This window is defined at view distance from the reference point.

If the window's aspect ratio does not match that of the viewport, the image is stretched in either the x direction or the y direction to fit the viewport exactly. The window's aspect ratio is the ratio of (umax-umin) to (vmax-vmin).

An error condition occurs if either of the dimensions of the window is zero.

GMR_$VIEW_SET_YON_DISTANCE

Sets the N-coordinate of the far clipping plane for a given viewport.

FORMAT

```
GMR_$VIEW_SET_YON_DISTANCE (viewport_id, yon_distance, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

yon_distance
The N-coordinate of the yon (far) clipping plane, in GMR_$F_T format. This is a real value. The reference point is N = 0 in UVN coordinate space.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The 3D GMR packge only displays geometry that is between the hither and yon clipping planes.

The absolute value of the yon distance is the geometric distance between the view reference point and the far clipping plane.

The default hither and yon distances are -1.0E10 and 1.0E10, respectively. This puts the reference point in the middle of the default view volume.

For a perspective projection, both hither and yon values must be positive in a left-handed UVN system and negative in a right-handed system.

GMR_$VIEWPORT_CLEAR

Clears a specified viewport to the background color.

FORMAT

```
GMR_$VIEWPORT_CLEAR (viewport_id, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport to be cleared, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

This feature is not automatic on a GMR_$VIEWPORT_REFRESH call.

GMR_$VIEWPORT_CREATE

Creates an additional viewport and assigns it an ID number.

FORMAT

```
GMR_$VIEWPORT_CREATE (vbounds, viewport_id, status)
```

INPUT PARAMETERS

vbounds
The bounds of the new viewport, in GMR_$F3_LIMITS_T format. This parameter is a six-element array of real values corresponding to xmin, xmax, ymin, ymax, zmin, and zmax. The bounds are expressed in logical device coordinates and must be within the specified logical device coordinate range.

OUTPUT PARAMETERS

viewport_id
The ID assigned by the 3D GMR package to the newly created viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

GMR_$INIT initializes the GMR package and viewports, creates one viewport called viewport 1 (which fills the display bitmap), and makes that viewport the selected viewport. Currently, overlapping viewports are not supported, so it is best to change the bounds of viewport 1 before creating additional viewports. You must supply bounds for the new viewport, in logical device coordinates. The GMR package assigns a number to the viewport.

Use the following procedure to change the original viewport to fill only the left half of the screen and create a second viewport in the center right of the screen. This assumes a logical device coordinate range [0.0, 1.0] in x, y, and z:

```
bounds.xmin := 0.0; bounds.ymin := 0.0;bounds.zmin := 0.0;
bounds.xmax := 0.5; bounds.ymax := 1.0;bounds.zmax := 1.0;
GMR_$VIEWPORT_SET_BOUNDS (viewport_id, bounds, status);
bounds.xmin := 0.6; bounds.ymin := 0.25; bounds.zmin := 0.0;
bounds.xmax := 1.0; bounds.ymax := 0.75; bounds.zmax := 1.0;
GMR_$VIEWPORT_CREATE (bounds, viewport_id, status);
```

GMR_$VIEWPORT_DELETE

Deletes a specified viewport.

FORMAT

```
GMR_$VIEWPORT_DELETE (viewport_id, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Because overlapping viewports are not supported, you should delete all but one viewport if a single viewport is to be expanded to fill the entire 3D GMR display area.

GMR_$VIEWPORT_INQ_BG_COLOR

Returns the background color ID and intensity of a specified viewport.

FORMAT

```
GMR_$VIEWPORT_INQ_BG_COLOR (viewport_id, color_id, intensity, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

color_id
The color ID number of specified viewport, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer in the range [0, GMR_$MAX_COLOR_ID], inclusive.

intensity
The intensity used within the range specified by the color ID, in GMR_$INTEN_T format. This is 4-byte real value in the range [0.0, 1.0], inclusive.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$VIEWPORT_SET_BG_COLOR to set the background properties of a viewport.

GMR_$VIEWPORT_INQ_BORDER

Returns the width of the four edges of the specified viewport, the border-on flag, and color attributes.

FORMAT

```
GMR_$VIEWPORT_INQ_BORDER (viewport_id, border_width, border_on,
                          border_color_id, border_inten, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

border_width
The width of each side of the border in pixels, in GMR_$BORDER_WIDTH_T format. This parameter is a four-element array of 2-byte integers. The width of the border is specified in terms of left, right, top, and bottom.

border_on
A Boolean (logical) flag that indicates whether border is displayed.

border_color_id
Indicates the color range for the border, GMR_$COLOR_ID_T format. This parameter is a 2-byte integer in the range [0, GMR_$MAX_COLOR_ID], inclusive

border_inten
The intensity used within the range specified by the color ID, in GMR_$INTEN_T format. This is 4-byte real value in the range [0.0, 1.0], inclusive.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$VIEWPORT_SET_BORDER to change the border properties of a viewport.

GMR_$VIEWPORT_INQ_BOUNDS

Returns the bounds of the specified viewport.

FORMAT

```
GMR_$VIEWPORT_INQ_BOUNDS (viewport_id, vbounds, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

vbounds
The bounds of the specified viewport, in GMR_$F3_LIMITS_T format. This parameter is a six-element array of real values that specify xmin, xmax, ymin, ymax, zmin and zmax in logical device coordinates. See the Data Types section for more information.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$VIEWPORT_SET_BOUNDS to change a viewport's size.

GMR_$VIEWPORT_INQ_CULLING

Returns whether culling is enabled in a specified viewport and the current minimum screen size for displayed structures.

FORMAT

```
GMR_$VIEWPORT_INQ_CULLING (viewport_id, cull, min_area, status)
```

INPUT PARAMETERS

viewport_id
The identification number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

cull
A Boolean value that specifies whether culling is turned on (TRUE) or off (FALSE).

min_area
The minimum size for rendered structures, in GMR_$F_T format. This parameter is a 4-byte real value that specifies area in square device coordinates (i.e., pixels).

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Culling allows you to display only structures greater than a given size. This is a way of removing features from the display that have become too small to be useful.

When culling is enabled, all structures whose approximate projected area in device coordinates (i.e., pixels) is less than the value specified are not rendered.

The projected area of a structure is approximated by the projection of its bounding box. This typically produces approximate areas that are slightly larger than the actual screen size of the structure.

Culling is disabled in each viewport by default.

Use GMR_$VIEWPORT_SET_CULLING to turn culling on and off.

GMR_$VIEWPORT_INQ_GLOBAL_MATRIX

Returns the global modeling matrix for a specific viewport.

FORMAT

```
GMR_$VIEWPORT_INQ_GLOBAL_MATRIX (viewport_id, matrix, status)
```

INPUT PARAMETERS

viewport_id
The identification number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

matrix
A 4x3 matrix, in GMR_$4X3_MATRIX_T format.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$VIEWPORT_SET_GLOBAL_MATRIX to set the global modeling matrix.

GMR_$VIEWPORT_INQ_HILIGHT_ABLOCK

Returns the identification number of the highlighting attribute block of a specified viewport.

FORMAT

```
GMR_$VIEWPORT_INQ_HILIGHT_ABLOCK (viewport_id, ablock_id, status)
```

INPUT PARAMETERS

viewport_id
The identification number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

ablock_id
The ID number of the ablock in GMR_$ABLOCK_ID_T format.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$VIEWPORT_SET_HILIGHT_ABLOCK to assign a highlighting attribute ablock to a viewport.

The highlighting attribute block is used by GMR_$PICK and GMR_$INSTANCE_ECHO if the echo method is ablock.

GMR_$VIEWPORT_INQ_INVIS_FILTER

Returns the inclusion list and exclusion list for name sets that will be invisible in a specific viewport.

FORMAT

```
GMR_$VIEWPORT_INQ_INVIS_FILTER (viewport_id, n_incl_names, inclusion_set,
                                n_excl_names, exclusion_set, status)
```

INPUT PARAMETERS

viewport_id
The identification number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

n_incl_names
The number of names in the inclusion name set. This parameter is a 2 byte integer.

inclusion_set
The list of names in the inclusion set, in GMR_$NAME_SET_T format. Each name is in the range [1, GMR_$MAX_NAME_ELEMENT], inclusive.

n_excl_names
The number of elements in the exclusion name set. This parameter is a 2 byte integer.

exclusion_set
The list of names in the exclusion set, in GMR_$NAME_SET_T format. Each name is in the range [1, GMR_$MAX_NAME_ELEMENT], inclusive.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Primitives are invisible if no names in the current name set are in the viewport invisibility exclusion set AND at least one name in the current name set is in the viewport invisibility inclusion set.

GMR_$VIEWPORT_SET_INVIS_FILTER establishes the viewport invisibility inclusion and exclusion sets.

GMR_$ADD_NAME_SET and GMR_$REMOVE_NAME_SET add and remove names from the current name set.

See GMR_$ADD_NAME_SET for an example.

GMR_$VIEWPORT_INQ_PATH_ORDER

Returns the path order used for picking and instance echoing in a specified viewport.

FORMAT

```
GMR_$VIEWPORT_INQ_PATH_ORDER (viewport_id, order, status)
```

INPUT PARAMETERS

viewport_id
The identification number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

order
The path order, in GMR_$INSTANCE_PATH_ORDER_T format. Specify one of the following predefined values: GMR_$TOP_FIRST (specified element last) and GMR_$BOTTOM_FIRST (specified element first).

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The path order is used by both GMR_$PICK and GMR_$INSTANCE_ECHO.

Use GMR_$VIEWPORT_SET_PATH_ORDER to set the path order.

GMR_$VIEWPORT_INQ_PICK

Returns the pickability range and mask for the specified viewport.

FORMAT

```
GMR_$VIEWPORT_INQ_PICK (viewport_id, pick_low_range, pick_high_range,
                        pick_mask, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

pick_low_range
The low boundary for the pickability test, in GMR_$STRUCTURE_VALUE_T format. This parameter is a 4-byte integer.

pick_high_range
The high boundary for the pickability test, in GMR_$STRUCTURE_VALUE_T format. This parameter is a 4-byte integer.

pick_mask
The mask value for the pickability test, in GMR_$STRUCTURE_MASK_T format. This parameter is a 4-byte integer.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$VIEWPORT_SET_PICK to change the current values for viewport pickability.

GMR_$VIEWPORT_INQ_PICK_FILTER

Returns the inclusion list and exclusion list for name sets that are pickable for a specific viewport.

FORMAT

```
GMR_$VIEWPORT_INQ_PICK_FILTER (viewport_id, n_incl_names, inclusion_set,
                               n_excl_names, exclusion_set, status)
```

INPUT PARAMETERS

viewport_id
The identification number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

n_incl_names
The number of names in the inclusion name set. This parameter is a 2-byte integer.

inclusion_set
The list of names in the inclusion set, in GMR_$NAME_SET_T format. Each name is in the range [1, GMR_$MAX_NAME_ELEMENT], inclusive.

n_excl_names
The number of names in the exclusion name set. This parameter is a 2-byte integer.

exclusion_set
The list of names in the exclusion set, in GMR_$NAME_SET_T format. Each name is in the range [1, GMR_$MAX_NAME_ELEMENT], inclusive.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Primitives are pickable if visibility criteria are met AND at least one name in the current name set is in the viewport pick inclusion set AND no name in the current name set are in the viewport pick exclusion set.

GMR_$VIEWPORT_SET_INVIS_FILTER and GMR_$VIEWPORT_SET_PICK_FILTER establish the viewport inclusion and exclusion sets.

GMR_$ADD_NAME_SET and GMR_$REMOVE_NAME_SET add and remove names from the current name set.

See GMR_$ADD_NAME_SET for an example.

GMR_$VIEWPORT_INQ_REFRESH_STATE

Returns the current refresh state of the specified viewport.

FORMAT

```
GMR_$VIEWPORT_INQ_REFRESH_STATE (viewport_id, refresh_state, status)
```

INPUT PARAMETERS

viewport_id
The identification number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

refresh_state
The refresh state of the viewport, in GMR_$REFRESH_STATE_T format. This is a 2-byte integer. Specify one of the following predefined values:

GMR_$REFRESH_WAIT
When you modify elements in the file, the viewport is rewritten when you call GMR_$VIEWPORT_REFRESH or GMR_$DISPLAY_REFRESH.

GMR_$REFRESH_INHIBIT
When you modify elements in the file, the viewport is rewritten only when you call GMR_$VIEWPORT_REFRESH. GMR_$DISPLAY_REFRESH does not affect a viewport in this refresh state.

GMR_$REFRESH_PARTIAL
Individual elements are updated as they are changed in the metafile. When deleting or replacing an element (or subtree if the element is an instance), the element (or subtree) is erased by drawing in the background color. When inserting or replacing, the new element (or subtree) is drawn without regard to other elements on the display. The viewport is completely redrawn when you call GMR_$VIEWPORT_REFRESH or GMR_$DISPLAY_REFRESH.

GMR_$REFRESH_UPDATE
The viewport is completely redrawn every time that you change a displayed structure.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$VIEWPORT_SET_REFRESH_STATE to establish the refresh state of a viewport.

GMR_$VIEWPORT_INQ_STATE

Returns all the user-defineable parameters of a specified viewport.

FORMAT

```
GMR_$VIEWPORT_INQ_STATE (viewport_id, array_size, size, viewport_state,
                         status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

array_size
The size of the array passed. This parameter is a 2-byte integer.

OUTPUT PARAMETERS

size
The size that was actually used. This parameter is a 2-byte integer.

viewport_state
The parameters of the viewport, in GMR_$L_ARRAY_T format. This parameter is an array of 4-byte integers. The array should be at least as long as the constant GMR_$VIEWPORT_STATE_BLOCK_SIZE.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use this call when you want to create a new viewport similar to an existing viewport. Use GMR_$VIEWPORT_INQ_STATE to retrieve the parameters of the old viewport. Then use GMR_$VIEWPORT_SET_STATE to set the parameters of the new viewport.

The array viewport_state is useful only for setting the parameters of a viewport via GMR_$VIEWPORT_SET_STATE as described above. The data in the array should not be modified.

An error condition occurs if the array passed to this routine is not long enough to contain the state. No information is copied in this case.

GMR_$VIEWPORT_INQ_STRUCTURE

Returns the structure ID and the file ID of the structure that is assigned to a specific viewport.

FORMAT

```
GMR_$VIEWPORT_INQ_STRUCTURE (viewport_id, structure_id, file_id, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

structure_id
The identification number of the structure assigned to the viewport, in GMR_$STRUCTURE_ID_T format. This parameter is a 4-byte integer.

file_id
The identification number of the file that the structure belongs to, in GMR_$FILE_ID_T format. This parameter is a 2-byte integer.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$VIEWPORT_SET_STRUCTURE to assign a structure to a viewport.

GMR_$VIEWPORT_INQ_VISIBILITY

Returns the visibility range and mask values for the specified viewport.

FORMAT

```
GMR_$VIEWPORT_INQ_VISIBILITY (viewport_id, vis_low_range, vis_high_range,
                              vis_mask, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

vis_low_range
The low boundary for the visibility test, in GMR_$STRUCTURE_VALUE_T format. This parameter is a 4-byte integer.

vis_high_range
The high boundary for the visibility test, in GMR_$STRUCTURE_VALUE_T format. This parameter is a 4-byte integer.

vis_mask
The mask value for the visibility test, in GMR_$STRUCTURE_MASK_T format. This parameter is a 4-byte integer.

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$VIEWPORT_SET_VISIBILITY to change the current value for viewport visibility.

At display-time, the structure mask is tested against the viewport visibility mask (and the viewport pick mask). The structure value is tested against the viewport visibility range (and the viewport pick range).

The structure is visible under these conditions:

1. The structure value must be within the viewport's visibility range, inclusive.

2. The logical AND of the structure mask and the viewport visibility mask must be nonzero.

If a structure does not meet these criteria, it is not traversed.

GMR_$VIEWPORT_MOVE

Translates the specified viewport, carrying the view with it.

FORMAT

```
GMR_$VIEWPORT_MOVE (viewport_id, translate, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

translate
The amount of translation, in GMR_$F2_POINT_T format. This parameter is a two-element array (i.e., an x,y pair) of real values. The translation is strictly a screen space operation and does not affect the z-coordinates of the viewport.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

GMR_$VIEWPORT_REFRESH

Redraws the contents of the specified viewport.

FORMAT

```
GMR_$VIEWPORT_REFRESH (viewport_id, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

This call refreshes a particular viewport as opposed to GMR_$DISPLAY_REFRESH which refreshes all viewports that are not in GMR_$REFRESH_INHIBIT state.

This call refreshes a viewport in any refresh mode.

GMR_$VIEWPORT_SET_BG_COLOR

Sets the background color and intensity of a specified viewport.

FORMAT

```
GMR_$VIEWPORT_SET_BG_COLOR (viewport_id, color_id, intensity, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

color_id
The color identifier, in GMR_$COLOR_ID_T format. This parameter is a 2-byte integer in the range [0, GMR_$MAX_COLOR_ID], inclusive.

intensity
The intensity used within the range specified by the color ID, in GMR_$INTEN_T format. This is 4-byte real value in the range [0.0, 1.0], inclusive.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

The default for color_id is 0. The default for intensity is 1.0.

GMR_$VIEWPORT_SET_BORDER

Specifies the border size of the specified viewport in pixels, the border color, and whether the border is displayed.

FORMAT

```
GMR_$VIEWPORT_SET_BORDER (viewport_id, border_width, border_on,
                          border_color_id, border_inten, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

border_width
The width of each side of the border in pixels, in GMR_$BORDER_WIDTH_T format. This parameter is a four-element array of 2-byte integers. The width of the border is specified in terms of left, right, top, and bottom.

border_on
A Boolean (logical) flag that indicates whether border is displayed.

border_color_id
Indicates the color range for the border, GMR_$COLOR_ID_T format. This parameter is a 2-byte integer in the range [0, GMR_$MAX_COLOR_ID], inclusive.

border_inten
The intensity used within the range specified by the color ID, in GMR_$INTEN_T format. This is 4-byte real value in the range [0.0, 1.0], inclusive.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use GMR_$VIEWPORT_INQ_BORDER to retrieve the border properties of a given viewport.

The default for border_on is false, the default for color is 1, and the default for intensity is 1.0.

GMR_$VIEWPORT_SET_BOUNDS

Changes the display bounds for the specified viewport.

FORMAT

```
GMR_$VIEWPORT_SET_BOUNDS (viewport_id, vbounds, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

vbounds
The bounds of the new viewport, in GMR_$F3_LIMITS_T format. This parameter is a six-element array of real values specifying xmin, xmax, ymin, ymax, zmin, and zmax in logical device coordinates. See the Data Types section for more information.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

GMR_$VIEWPORT_SET_BOUNDS sets the bounds of the specified viewport. You must provide the minimum and maximum values for x, y, and z. By default, coordinates are expressed as logical device coordinates as follows: bottom left = (0.0, 0.0, 0.0); top right = (1.0, 1.0, 1.0).

Currently, 3D GMR does not support overlapping viewports.

Use the following procedure to change the bounds of the specified viewport to fill only the left half of the screen (assuming a logical device coordinate range of [0.0, 1.0] for x, y, and z).

```
bounds.xmin := 0.0; bounds.ymin := 0.0; bounds.zmin := 0.0;
bounds.xmax := 0.5; bounds.ymax := 1.0; bounds.zmax := 1.0;
GMR_$VIEWPORT_SET_BOUNDS (viewport_id, bounds, status);
```

GMR_$VIEWPORT_SET_CULLING

Enables culling in a specified viewport: only structures larger than a given screen-space area are displayed.

FORMAT

```
GMR_$VIEWPORT_SET_CULLING (viewport_id, cull, min_area, status)
```

INPUT PARAMETERS

viewport_id
The identification number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

cull
A Boolean value that specifies whether culling is to be turned on (TRUE) or off (FALSE).

min_area
The minimum size for rendered structures, in GMR_$F_T format. This parameter is a 4-byte real value that specifies area in square device coordinates (i.e., pixels).

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

This feature allows you to display only structures greater than a given size. This is a way of removing features from the display that have become too small to be useful.

When culling is enabled, these structures are not rendered: structures with an approximate projected area in square device coordinates (i.e., the number of pixels covered) less than the value specified.

This call operates on the structure level, not on the element level.

The projected area of a structure is approximated by the projection of its bounding box. This typically produces approximate areas that are slightly larger than the actual screen-space area of the structure.

Culling is disabled in each viewport by default.

Use GMR_$STRUCTURE_INQ_BOUNDS to return the limits of the bounding box of a structure.

NOTE: If you turn text anchor clipping off and turn culling on, then text within a structure that would have been culled is still drawn.

GMR_$VIEWPORT_SET_GLOBAL_MATRIX

Associates a global modeling matrix with a viewport.

FORMAT

```
GMR_$VIEWPORT_SET_GLOBAL_MATRIX (viewport_id, matrix, status)
```

INPUT PARAMETERS

viewport_id
The identification number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

matrix
A 4x3 matrix, in GMR_$4X3_MATRIX_T format.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

This modeling transformation is applied to all coordinate data in the metafile as the last step before the normalizing transformation and projection.

The default is the identity matrix.

GMR_$VIEWPORT_SET_HILIGHT_ABLOCK

Assigns a highlighting attribute block to a specified viewport.

FORMAT

```
GMR_$VIEWPORT_SET_HILIGHT_ABLOCK (viewport_id, ablock_id, status)
```

INPUT PARAMETERS

viewport_id
The identification number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

ablock_id
The ID number of the ablock, in GMR_$ABLOCK_ID_T format. This parameter is a 2-byte integer.

If the ablock_id is invalid, then the default highlight_ablock is set and a warning is returned in the status parameter.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

You can specify highlighting in selected viewports or highlight different viewports with different types of highlighting.

Both pick echo and instance echo use the highlighting attribute block feature (see GMR_$PICK_SET_ECHO_METHOD and GMR_$INSTANCE_ECHO_SET_METHOD).

GMR_$VIEWPORT_SET_INVIS_FILTER

Specifies an inclusion list and an exclusion list for name sets that will be invisible in a specific viewport.

FORMAT

```
GMR_$VIEWPORT_SET_INVIS_FILTER (viewport_id, n_incl_names, inclusion_set,
                                n_excl_names, exclusion_set, status)
```

INPUT PARAMETERS

viewport_id
The identification number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

n_incl_names
The number of names in the inclusion name set. This parameter is a 2-byte integer.

inclusion_set
The list of names in the inclusion set, in GMR_$NAME_SET_T format. Each name is in the range [1, GMR_$MAX_NAME_ELEMENT], inclusive.

n_excl_names
The number of names in the exclusion name set. This parameter is a 2-byte integer.

exclusion_set
The list of names in the exclusion set, in GMR_$NAME_SET_T format. Each name is in the range [1, GMR_$MAX_NAME_ELEMENT], inclusive.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Primitives are invisible if no names in the current name set are in the viewport invisibility exclusion set AND at least one name in the current name set is in the viewport invisibility inclusion set.

GMR_$ADD_NAME_SET and GMR_$REMOVE_NAME_SET add and remove names from the current name set.

The relationship between the inclusion and exclusion sets and the current name set can be stated mathematically as follows:

```
Ii = invisibility inclusion set
Ei = invisibility exclusion set
N = current name set
int = set intersection
.EQ. = equals
```

```
.NE. = not equal to

Invisible <=> (Ii int N .NE. 0) AND (Ei int N .EQ. 0)
Visible   <=> (Ii int N .EQ. 0) OR  (Ei int N .NE. 0)
```

See GMR_$ADD_NAME_SET for an expanded explanation and an example.

GMR_$VIEWPORT_SET_PATH_ORDER

Sets the path order used for picking and instance echoing in a specified viewport.

FORMAT

```
GMR_$VIEWPORT_SET_PATH_ORDER (viewport_id, order, status)
```

INPUT PARAMETERS

viewport_id
The identification number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

order
The path order, in GMR_$INSTANCE_PATH_ORDER_T format. This is a 2-byte integer. Specify one of the following predefined values: GMR_$TOP_FIRST (specified element last) and GMR_$BOTTOM_FIRST (specified element first).

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

This call determines the order of the path returned by a GMR_$PICK operation. It also defines the order of an input path to GMR_$INSTANCE_ECHO.

GMR_$VIEWPORT_INQ_PATH_ORDER returns the current path order for a viewport.

GMR_$VIEWPORT_SET_PICK

Sets the pickability range and mask for the specified viewport

FORMAT

```
GMR_$VIEWPORT_SET_PICK (viewport_id, pick_low_range, pick_high_range,
                        pick_mask, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

pick_low_range
The low boundary for the pickability test, in GMR_$STRUCTURE_VALUE_T format. This parameter is a 4-byte integer.

pick_high_range
The high boundary for the pickability test, in GMR_$STRUCTURE_VALUE_T format. This parameter is a 4-byte integer.

pick_mask
The mask value for the pickability test, in GMR_$STRUCTURE_MASK_T format. This parameter is a 4-byte integer.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

At display-time, the 3D GMR package tests the structure mask against the viewport visibility mask and the viewport pick mask. The package also tests the structure value against the viewport pick range and the viewport visibility range.

The structure is visible under these conditions:

1. The structure value must be within the viewport's visibility range, inclusive.
2. The logical AND of the structure mask and the viewport visibility mask must be nonzero.

The structure is pickable under these conditions:

1. The structure must meet the above visibility criteria.
2. The structure value must be within the viewport's pick range, inclusive.
3. The logical AND of the structure mask and the viewport pick mask must be nonzero.

If the visibility and pickability criteria are met, then a structure is pickable even though it may not be visible on the screen. This means that you can pick an invisible object as follows: change the visibility range and mask to match the viewport pickability range and mask and then pick the structure.

GMR_$VIEWPORT_SET_PICK_FILTER

Specifies an inclusion list and an exclusion list for name sets that are pickable for a specific viewport.

FORMAT

```
GMR_$VIEWPORT_SET_PICK_FILTER (viewport_id, n_incl_names, inclusion_set,
                               n_excl_names, exclusion_set, status)
```

INPUT PARAMETERS

viewport_id
The identification number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

n_incl_names
The number of names in the inclusion set. This parameter is a 2-byte integer.

inclusion_set
The list of names in the inclusion set, in GMR_$NAME_SET_T format. Each name is in the range [1, GMR_$MAX_NAME_ELEMENT], inclusive.

n_excl_names
The number of names in the exclusion set. This parameter is a 2-byte integer.

exclusion_set
The list of names in the exclusion set, in GMR_$NAME_SET_T format. Each name is in the range [1, GMR_$MAX_NAME_ELEMENT], inclusive.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Primitives are pickable if visibility criteria are met AND at least one name in the current name set is in the viewport pick inclusion set AND no names in the current name set are in the viewport pick exclusion set.

The relationship between the inclusion and exclusion sets and the current name set can be stated mathematically as follows:

```
Ii = invisibility inclusion set
Ei = invisibility exclusion set
Ip = pick inclusion set
Ep = pick exclusion set
N  = current name set
int = set intersection
.EQ. = equals
.NE. = not equal to
```

```
Invisible <=> (Ii int N .NE. 0) AND (Ei int N .EQ. 0)
Visible   <=> (Ii int N .EQ. 0) OR  (Ei int N .NE. 0)

Pickable  <=> [(Ii int N .EQ. 0) OR  (Ei int N .NE. 0)] AND
              (Ip int N .NE. 0) AND (Ep int N .EQ. 0)
```

If the above criteria is met, the primitive is pickable even though it may not be visible on the screen. To pick an invisible object, do the following: change the name set and then pick it without calling a viewport clear/refresh combination.

See GMR_$ADD_NAME_SET for an expanded explanation and an example.

GMR_$VIEWPORT_SET_INVIS_FILTER establishes the name set visibility criteria.

GMR_$VIEWPORT_SET_REFRESH_STATE

Sets the refresh state of a specified viewport.

FORMAT

```
GMR_$VIEWPORT_SET_REFRESH_STATE (viewport_id, refresh_state, status )
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

refresh_state
The refresh state of the viewport, in GMR_$REFRESH_STATE_T format. This is a 2-byte integer. Specify only one of the following predefined values:

GMR_$REFRESH_WAIT
When you modify elements in the file, the viewport is rewritten when you call GMR_$VIEWPORT_REFRESH or GMR_$DISPLAY_REFRESH.

GMR_$REFRESH_INHIBIT
When you modify elements in the file, the viewport is rewritten only when you call GMR_$VIEWPORT_REFRESH. GMR_$DISPLAY_REFRESH does not affect a viewport in this refresh state.

GMR_$REFRESH_PARTIAL
Individual elements are updated as they are changed in the metafile. When deleting or replacing an element (or subtree if the element is an instance), the element (or subtree) is erased by drawing in the background color. When inserting or replacing, the new element (or subtree) is drawn without regard to other elements on the display. The viewport is completely redrawn when you call GMR_$VIEWPORT_REFRESH or GMR_$DISPLAY_REFRESH.

GMR_$REFRESH_UPDATE
The viewport is completely redrawn every time that you change a displayed structure.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This data type is 4 bytes long. See the Data Types section for more information.

USAGE

GMR_$VIEWPORT_SET_REFRESH_STATE allows you to control the frequency at which the display in a viewport is refreshed. This routine allows you to change the metafile

and have the package automatically update one or more viewports to incorporate these changes, without calling a refresh routine. One use of this feature is in a rubber-banding procedure when you are trying to find the right place to put a line (see GMR_$DYN_MODE_SET_ENABLE).

GMR_$VIEWPORT_SET_STATE

Sets the user-definable parameters for a specified viewport.

FORMAT

```
GMR_$VIEWPORT_SET_STATE (viewport_id, viewport_state, status)
```

INPUT PARAMETERS

viewport_id
The identification number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

viewport_state
The parameters of the viewport, in GMR_$L_ARRAY_T format. This is an array of 4-byte integers. The array should be at least as long as the constant GMR_$VIEWPORT_STATE_BLOCK_SIZE.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

This call sets all user-definable viewport parameters: viewport size, border and color information, viewing parameters, display structure, picking information, etc.

This call is useful when you want to create a new viewport that is similar to an existing viewport. Obtain the parameters of the exisiting viewport through GMR_$VIEWPORT_INQ_STATE. Then use those parameters for the new viewport.

The call does not automatically redisplay the viewport. You will see the results the next time you refresh the viewport. For example:

```
GMR_$VIEWPORT_INQ_STATE(vpid, array_size, size, view_state, status);
GMR_$VIEWPORT_SET_STATE(vpid2, view_state, status);
GMR_$VIEWPORT_CLEAR(vpid2, status);
GMR_$VIEWPORT_REFRESH(vpid2, status);
```

GMR_$VIEWPORT_SET_STRUCTURE

Binds a structure (and possibly a subtree) to a viewport for future refreshing.

FORMAT

```
GMR_$VIEWPORT_SET_STRUCTURE (viewport_id, structure_id, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

structure_id
The identification number of the structure, in GMR_$STRUCTURE_ID_T format. This parameter is a 4-byte integer.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

Use this call to bind a structure as the root structure for the viewport. Use GMR_$VIEWPORT_CLEAR followed by GMR_$VIEWPORT_REFRESH or GMR_$DISPLAY_REFRESH to display the structure in the viewport.

There is no default structure assigned to a viewport.

You can assign any structure in the current file to a viewport or you can assign the primary structure of the file if one exists.

Before you can assign the primary structure to a viewport you must first set the primary structure of the file using GMR_$FILE_SET_PRIMARY_STRUCTURE. To retrieve the structure ID of the primary structure, use GMR_$FILE_INQ_PRIMARY_STRUCTURE.

GMR_$VIEWPORT_INQ_STRUCTURE returns the structure ID and the file ID of the structure that is assigned to a specific viewport.

GMR_$VIEWPORT_SET_VISIBILITY

Sets the visibility range and mask value for the viewport.

FORMAT

```
GMR_$VIEWPORT_SET_VISIBILITY (viewport_id, vis_low_range, vis_high_range,
                              vis_mask, status)
```

INPUT PARAMETERS

viewport_id
The number of the viewport, in GMR_$VIEWPORT_ID_T format. This parameter is a 2-byte integer. The 3D GMR package assigns this value.

vis_low_range
The low boundary for the visibility test, in GMR_$STRUCTURE_VALUE_T format. This parameter is a 4-byte integer.

vis_high_range
The high boundary for the visibility test, in GMR_$STRUCTURE_VALUE_T format. This parameter is a 4-byte integer.

vis_mask
The mask value for the visibility test, in GMR_$STRUCTURE_MASK_T format. This parameter is a 4-byte integer.

OUTPUT PARAMETERS

status
Completion status, in STATUS_$T format. This parameter is 4 bytes long. See the Data Types section for more information.

USAGE

At display-time, the 3D GMR package tests the structure mask against the viewport visibility mask (and the viewport pick mask). The package also tests the structure value against the viewport visibility range (and the viewport pick range).

The structure is visible under these conditions:

1. The structure value must be within the viewport's visibility range, inclusive.

2. The logical AND of the structure mask and the viewport visibility mask must be nonzero.

If a structure does not meet these criteria, it is not traversed.

Quick Reference

This section provides a quick reference to 3D GMR routines. Information is presented in two parts: a list organized by function followed by an alphabetical list of calls with their formats.

3D GMR Routines

The following is a list of routines organized by functional category. Some routines are included in more than one category. The method of organization is similar to that in *Programming With DOMAIN 3D Graphics Metafile Resource.*

Controlling 3D GMR

Controlling the 3D GMR Package

GMR_$INIT
: Initializes the 3D graphics metafile package and opens the display.

GMR_$TERMINATE
: Terminates the 3D GMR package and closes the display.

Controlling Files

GMR_$FILE_CLOSE
: Closes the current file, saving revisions or not as specified.

GMR_$FILE_CREATE
: Creates a new graphics metafile and makes it the current file.

GMR_$FILE_INQ_PRIMARY_STRUCTURE
: Returns the identification number of the structure assumed to be the start of the current file.

GMR_$FILE_OPEN
: Reopens an existing file and makes it the current file.

GMR_$FILE_SELECT
: Makes the specified file the current file.

GMR_$FILE_SET_PRIMARY_STRUCTURE
: Sets the structure number assumed to be the start of the current file.

Controlling Structures

GMR_$FILE_INQ_PRIMARY_STRUCTURE
: Returns the identification number of the structure assumed to be the start of the current file.

GMR_$FILE_SET_PRIMARY_STRUCTURE
: Sets the structure number assumed to be the start of the current file.

GMR_$STRUCTURE_CLOSE
Closes the current structure, saving revisions or not.

GMR_$STRUCTURE_COPY
Copies the entire contents of another structure into the current open structure.

GMR_$STRUCTURE_CREATE
Creates a new structure and assigns it a structure identification number and (optionally) an application-supplied name.

GMR_$STRUCTURE_DELETE
Deletes the current open structure.

GMR_$STRUCTURE_ERASE
Deletes all elements in the current structure.

GMR_$STRUCTURE_INQ_BOUNDS
Returns the limits of the bounding box that encloses a structure and any subtrees that the structure calls.

GMR_$STRUCTURE_INQ_COUNT
Returns the number of structures in the current metafile and a structure number guaranteed to be greater than or equal to the largest structure number.

GMR_$STRUCTURE_INQ_ID
Returns the structure identification number of the named structure.

GMR_$STRUCTURE_INQ_INSTANCES
Returns both the number of instance elements that invoke a particular structure and the maximum number of levels of instancing that occur beneath the structure.

GMR_$STRUCTURE_INQ_NAME
Returns the name of the structure with the specified structure identification number.

GMR_$STRUCTURE_INQ_OPEN
Returns the identification number of the open structure.

GMR_$STRUCTURE_INQ_TEMPORARY
Returns whether the specified structure is temporary or not.

GMR_$STRUCTURE_INQ_VALUE_MASK
Returns the value and mask of a structure that are used to determine visibility and pick eligibility.

GMR_$STRUCTURE_OPEN
Reopens an existing structure.

GMR_$STRUCTURE_SET_NAME
Renames an existing structure.

GMR_$STRUCTURE_SET_TEMPORARY
Makes the specified structure temporary or not. Temporary structures are deleted when the file is closed.

GMR_$STRUCTURE_SET_VALUE_MASK
Sets the structure value and structure mask that are used to determine visibility and pick eligibility

GMR_$VIEWPORT_INQ_STRUCTURE
Returns the structure ID and the file ID of the structure that is assigned to a specific viewport.

GMR_$VIEWPORT_SET_STRUCTURE
Binds a structure (and possibly a subtree) to a viewport for future refreshing.

Editing Structures and Elements

GMR_$ELEMENT_DELETE
Deletes the current element.

GMR_$ELEMENT_INQ_INDEX
Returns the value stored for the current element index.

GMR_$ELEMENT_SET_INDEX
Sets the current element to the value indicated.

GMR_$INQ_ELEMENT_TYPE
Returns the type of the current element in the current open structure.

GMR_$REPLACE_INQ_FLAG
Returns the current value of the replace flag.

GMR_$REPLACE_SET_FLAG
Sets or clears a flag that causes subsequent elements to replace the current element rather than being inserted after the current element.

GMR_$STRUCTURE_COPY
Copies the entire contents of another structure into the current structure.

GMR_$STRUCTURE_DELETE
Deletes the current structure.

GMR_$STRUCTURE_ERASE
Deletes all elements in the current structure.

GMR_$STRUCTURE_INQ_BOUNDS
Returns the limits of the bounding box that encloses a structure and any subtrees that the structure calls.

GMR_$STRUCTURE_INQ_COUNT
Returns the number of structures in the current metafile and a structure number guaranteed to be greater than or equal to the largest structure number.

GMR_$STRUCTURE_INQ_INSTANCES
Returns both the number of instance elements that invoke a particular structure and the maximum number of levels of instancing that occur beneath the structure.

Using Tags

GMR_$INQ_TAG
: Returns the length of the text stored in the current (GMR_$TAG) element and a specified substring of that text.

GMR_$TAG
: Inserts a comment into the current open structure.

GMR_$TAG_LOCATE
: Searches for the specified tag in the specified range of structures and returns the structure ID of the lowest numbered structure in which the tag is found.

Drawing Primitives Routines

GMR_$F3_MESH
: Inserts a primitive element into the current open structure. The element draws a mesh.

GMR_$F3_MULTILINE
: Inserts a primitive element into the current open structure. The element draws a sequence of disconnected line segments.

GMR_$F3_POLYGON
: Inserts a primitive element into the current open structure. The element draws a polygon.

GMR_$F3_POLYLINE
: Inserts a primitive element into the current open structure. The element draws a sequence of connected lines.

GMR_$F3_POLYMARKER
: Inserts a primitive element into the current open structure. The element draws a set of markers.

GMR_$INQ_F3_MESH
: Returns the major and minor dimensions and the list of mesh points associated with the current (GMR_$F3_MESH) element.

GMR_$INQ_F3_MULTILINE
: Returns the list of multiline points associated with the current (GMR_$F3_MULTILINE) element.

GMR_$INQ_F3_POLYGON
: Returns the list of polygon points associated with the current (GMR_$F3_POLYGON) element.

GMR_$INQ_F3_POLYLINE
: Returns the list of polyline points associated with the current (GMR_$F3_POLYLINE) element.

GMR_$INQ_F3_POLYMARKER
: Returns the list of marker points associated with the current (GMR_$F3_POLYMARKER) element.

GMR_$INQ_TEXT
Returns the text string, the number of characters, and the anchor point stored in the current (GMR_$TEXT) element.

GMR_$TEXT
Inserts a primitive element into the current open structure. The element defines and positions a text string.

Attribute Routines

Direct Attributes

GMR_$ADD_NAME_SET
Inserts an attribute element into the current open structure. The element adds names to the current name set.

GMR_$FILL_COLOR
Inserts an attribute element into the current open structure. The element establishes fill color for polygons and meshes.

GMR_$FILL_INTEN
Inserts an attribute element into the current open structure. The element establishes fill intensity for polygons and meshes.

GMR_$INQ_ADD_NAME_SET
Returns the list of names in the current (GMR_$ADD_NAME_SET) element.

GMR_$INQ_FILL_COLOR
Returns the color ID specified by the current (GMR_$FILL_COLOR) element.

GMR_$INQ_FILL_INTEN
Returns the intensity of the current (GMR_$FILL_INTEN) element.

GMR_$INQ_LINE_COLOR
Returns the color ID specified by the current (GMR_$LINE_COLOR) element.

GMR_$INQ_LINE_INTEN
Returns the intensity of the current (GMR_$LINE_INTEN) element.

GMR_$INQ_LINE_TYPE
Returns the line type ID of the current (GMR_$LINE_TYPE) element.

GMR_$INQ_MARK_COLOR
Returns the color ID specified by the current (GMR_$MARK_COLOR) element.

GMR_$INQ_MARK_INTEN
Returns the intensity of the current (GMR_$MARK_INTEN) element.

GMR_$INQ_MARK_SCALE
Returns the scale factor for the current (GMR_$MARK_SCALE) element.

GMR_$INQ_MARK_TYPE
Returns the mark type for the current (GMR_$MARK_TYPE) element.

GMR_$INQ_REMOVE_NAME_SET
Returns the list of names in the current (GMR_$REMOVE_NAME_SET) element.

GMR_$INQ_TEXT_COLOR
Returns the color ID stored in the currrent (GMR_$TEXT_COLOR) element.

GMR_$INQ_TEXT_EXPANSION
Returns the expansion factor stored for the currrent (GMR_$TEXT_EXPANSION) element.

GMR_$INQ_TEXT_HEIGHT
Returns the height stored for the currrent (GMR_$TEXT_HEIGHT) element.

GMR_$INQ_TEXT_INTEN
Returns the value stored for the currrent (GMR_$TEXT_INTEN) element.

GMR_$INQ_TEXT_PATH
Returns the text angle stored for the currrent (GMR_$TEXT_PATH) element.

GMR_$INQ_TEXT_SLANT
Returns the slant factor stored for the currrent (GMR_$TEXT_SLANT) element.

GMR_$INQ_TEXT_SPACING
Returns the spacing stored for the currrent (GMR_$TEXT_SPACING) element.

GMR_$INQ_TEXT_UP
Returns the up vector stored for the currrent (GMR_$TEXT_UP) element.

GMR_$LINE_COLOR
Inserts an attribute element into the current open structure. The element establishes line color for polylines and multilines.

GMR_$LINE_INTEN
Inserts an attribute element into the current open structure. The element establishes line intensity for polylines and multilines.

GMR_$LINE_TYPE
Inserts an attribute element into the current open structure. The element establishes line type for polylines and multilines.

GMR_$MARK_COLOR
Inserts an attribute element into the current open structure. The element establishes color for polymarker elements.

GMR_$MARK_INTEN
Inserts an attribute element into the current open structure. The element establishes intensity for polymarker elements.

GMR_$MARK_SCALE
Inserts an attribute element into the current open structure. The element establishes the scale factor for polymarker elements.

GMR_$MARK_TYPE
Inserts an attribute element into the current open structure. The element establishes the polymarker type.

GMR_$REMOVE_NAME_SET
Inserts an attribute element into the current open structure. The element removes names from the current name set.

GMR_$TEXT_COLOR
Inserts an attribute element into the current open structure. The element establishes the color ID used for rendering text.

GMR_$TEXT_EXPANSION
Inserts an attribute element into the current open structure. The element establishes the expansion factor for text.

GMR_$TEXT_HEIGHT
Inserts an attribute element into the current open structure. The element establishes the height for text.

GMR_$TEXT_INQ_ANCHOR_CLIP
Returns the mode for clipping text.

GMR_$TEXT_INTEN
Inserts an attribute element into the current open structure. The element establishes intensity for text.

GMR_$TEXT_PATH
Inserts an attribute element into the current open structure. The element establishes the angle of the text path.

GMR_$TEXT_SET_ANCHOR_CLIP
Inserts an element into the metafile that specifies whether text is clipped by anchor point.

GMR_$TEXT_SLANT
Inserts an attribute element into the current open structure. The element establishes the slant of text.

GMR_$TEXT_SPACING
Inserts an attribute element into the current open structure. The element sets the spacing between text characters.

GMR_$TEXT_UP
Inserts an attribute element into the current open structure. The element specifies the up direction for text on the projection plane.

Attribute Source Flags

GMR_$ATTRIBUTE_SOURCE
Sets the attribute source flag for an attribute type to direct (use explicit attribute element) or aclass (use current aclass definition).

GMR_$INQ_ATTRIBUTE_SOURCE
Returns the attribute type and source flag for the current (GMR_$ATTRIBUTE_SOURCE) element.

Attribute Classes

GMR_$ACLASS
Inserts an element into the current open structure. The element selects an attribute class.

GMR_$INQ_ACLASS
Returns the attribute class for the current (GMR_$ACLASS) element.

Controlling Attribute Blocks

GMR_$ABLOCK_ASSIGN_DISPLAY
Assigns an attribute block (by number) to an attribute class, for all viewports of the display that have not already explicitly assigned that attribute class.

GMR_$ABLOCK_ASSIGN_VIEWPORT
Assigns an attribute block (by number) to an attribute class for one viewport.

GMR_$ABLOCK_COPY
Copies all attributes from one existing attribute block to another.

GMR_$ABLOCK_CREATE
Creates an attribute block and initializes it equivalent to an existing block.

GMR_$ABLOCK_DELETE
Deletes an attribute block and releases the attribute block identification number.

GMR_$ABLOCK_INQ_ASSIGN_DISPLAY
Returns the current attribute block number assigned to a particular attribute class for the display.

GMR_$ABLOCK_INQ_ASSIGN_VIEWPORT
Returns the current attribute block number assigned to a particular attribute class for a specified viewport.

Assigning and Retrieving Attribute Block Values

GMR_$ABLOCK_INQ_FILL_COLOR
Returns the color used for the interior of polygons and meshes and the enabled state for the specified attribute block.

GMR_$ABLOCK_INQ_FILL_INTEN
Returns the fill intensity used for polygons and meshes and the enabled state for the specified attribute block.

GMR_$ABLOCK_INQ_LINE_COLOR
Returns the polyline/multiline color and the enabled state for the specified attribute block.

GMR_$ABLOCK_INQ_LINE_INTEN
Returns the polyline/multiline intensity and the enabled state for the specified attribute block.

GMR_$ABLOCK_INQ_LINE_TYPE
Returns the polyline/multiline type ID and the enabled state for the specified attribute block.

GMR_$ABLOCK_INQ_MARK_COLOR
Returns the polymarker color and the enabled state for the specified attribute block.

GMR_$ABLOCK_INQ_MARK_INTEN
Returns the polymarker intensity and the enabled state for the specified attribute block.

GMR_$ABLOCK_INQ_MARK_SCALE
Returns the polymarker scale factor and the enabled state for the specified attribute block.

GMR_$ABLOCK_INQ_MARK_TYPE
Returns the polymarker type and the enabled state for the specified attribute block.

GMR_$ABLOCK_INQ_TEXT_COLOR
Returns the text color and the enabled state for the specified attribute block.

GMR_$ABLOCK_INQ_TEXT_EXPANSION
Returns the text expansion and the enabled state for the specified attribute block.

GMR_$ABLOCK_INQ_TEXT_HEIGHT
Returns the text height and the enabled state for the specified attribute block.

GMR_$ABLOCK_INQ_TEXT_INTEN
Returns the text intensity and the enabled state for the specified attribute block.

GMR_$ABLOCK_INQ_TEXT_PATH
Returns the text path and the enabled state for the specified attribute block.

GMR_$ABLOCK_INQ_TEXT_SLANT
Returns the text slant factor and the enabled state for the specified attribute block.

GMR_$ABLOCK_INQ_TEXT_SPACING
Returns the intercharacter spacing and the enabled state for the specified attribute block.

GMR_$ABLOCK_INQ_TEXT_UP
Returns the text up direction and the enabled state for the specified attribute block.

GMR_$ABLOCK_SET_FILL_COLOR
Sets the color used to fill polygons and meshes and the enabled state for the specified attribute block.

GMR_$ABLOCK_SET_FILL_INTEN
Sets the fill intensity for polygons and meshes and the enabled state for the specified attribute block.

GMR_$ABLOCK_SET_LINE_COLOR
Sets the poly/multiline color and the enabled state for the specified attribute block.

GMR_$ABLOCK_SET_LINE_INTEN
Sets the poly/multiline intensity and the enabled state for the specified attribute block.

GMR_$ABLOCK_SET_LINE_TYPE
Sets the poly/multiline type id and the enabled state for the specified attribute block.

GMR_$ABLOCK_SET_MARK_COLOR
Sets the polymarker color and the enabled state for the specified attribute block.

GMR_$ABLOCK_SET_MARK_INTEN
Sets the polymarker intensity and the enabled state for the specified attribute block.

GMR_$ABLOCK_SET_MARK_SCALE
Sets the polymarker scale factor and the enabled state for the specified attribute block.

GMR_$ABLOCK_SET_MARK_TYPE
Sets the polymarker type and the enabled state for the specified attribute block.

GMR_$ABLOCK_SET_TEXT_COLOR
Sets the text color and the enabled state for the specified attribute block.

GMR_$ABLOCK_SET_TEXT_EXPANSION
Sets the text expansion and the enabled state for the specified attribute block.

GMR_$ABLOCK_SET_TEXT_HEIGHT
Sets the text height and the enabled state for the specified attribute block.

GMR_$ABLOCK_SET_TEXT_INTEN
Sets the text intensity and the enabled state for the specified attribute block.

GMR_$ABLOCK_SET_TEXT_PATH
Sets the text path angle and the enabled state for the specified attribute block.

GMR_$ABLOCK_SET_TEXT_SLANT
Sets the text slant factor and the enabled state for the specified attribute block.

GMR_$ABLOCK_SET_TEXT_SPACING
Sets the intercharacter spacing and the enabled state for the specified attribute block.

GMR_$ABLOCK_SET_TEXT_UP
Sets the text up direction and the enabled state for the specified attribute block.

GMR_$VIEWPORT_INQ_HILIGHT_ABLOCK
Returns the identification number of the highlighting attribute block of a specified viewport.

GMR_$VIEWPORT_SET_HILIGHT_ABLOCK
Assigns a highlighting attribute block to a specified viewport.

Modeling Routines

Instancing

GMR_$INQ_INSTANCE_TRANSFORM
Returns the structure ID and the transformation applied at rendering time of the current (GMR_$INSTANCE_TRANSFORM) element.

GMR_$INSTANCE_TRANSFORM
Inserts an instance element into the current open structure. The element instances an identified structure with a specified transformation matrix.

GMR_$INSTANCE_TRANSFORM_FWD_REF
A forward-referencing instance routine. Creates a new structure, returns the structure ID, and inserts an instance element into the current open structure.

Matrix Routines

GMR_$4X3_MATRIX_CONCATENATE
Concatenates the two given 4x3 matrices and returns the resulting matrix.

GMR_$4X3_MATRIX_IDENTITY
Returns the 4x3 identity modeling matrix.

GMR_$4X3_MATRIX_INVERT
Returns the inverse of a 4x3 matrix.

GMR_$4X3_MATRIX_REFLECT
Specifies a reflection (or mirroring) through an arbitrary plane.

GMR_$4X3_MATRIX_ROTATE
Concatenates the specified 4x3 modeling matrix with a rotation matrix.

GMR_$4X3_MATRIX_ROTATE_AXIS
Specifies a rotation about an arbitrary axis.

GMR_$4X3_MATRIX_SCALE
Concatenates the specified 4x3 modeling matrix with a scaling matrix.

GMR_$4X3_MATRIX_TRANSLATE
Concatenates the specified 4x3 modeling matrix with a 4x3 translation matrix.

GMR_$VIEWPORT_INQ_GLOBAL_MATRIX
Returns the global modeling matrix for a specific viewport.

GMR_$VIEW_INQ_TRANSFORM
Returns a viewport's normalizing matrix.

GMR_$VIEWPORT_SET_GLOBAL_MATRIX
Associates a global modeling matrix with a viewport.

GMR_$VIEW_SET_TRANSFORM
Sets a viewport's normalizing matrix directly.

Viewing Parameter Routines

GMR_$VIEW_INQ_COORD_SYSTEM
Returns the coordinate system type (right- or left-handed) of the given viewport.

GMR_$VIEW_INQ_HITHER_DISTANCE
Returns the N-coordinate of the near clipping plane.

GMR_$VIEW_INQ_OBLIQUE
Returns the values of the foreshortening ratio and angle of receding lines of a specific viewport.

GMR_$VIEW_INQ_PROJECTION_TYPE
Returns the type of projection used in a specific viewport.

GMR_$VIEW_INQ_REFERENCE_POINT
Returns the value of the viewing reference point for a given viewport.

GMR_$VIEW_INQ_STATE
Returns the viewing parameter values for a specified viewport.

GMR_$VIEW_INQ_TRANSFORM
Returns a viewport's normalizing matrix.

GMR_$VIEW_INQ_VIEW_DISTANCE
Returns the distance from the reference point to the viewing plane in a given viewport.

GMR_$VIEW_INQ_VIEW_PLANE_NORMAL
Returns a vector that is normal to the view plane for a specified viewport.

GMR_$VIEW_INQ_WINDOW
Returns the minimum and maximum limits of the window on the viewing plane.

GMR_$VIEW_INQ_YON_DISTANCE
Returns the N-coordinate of the far clipping plane in a specified viewport.

GMR_$VIEW_SET_COORD_SYSTEM
Sets the coordinate system type of the given viewport.

GMR_$VIEW_SET_HITHER_DISTANCE
Sets the N-coordinate of the near clipping plane in a specified viewport.

GMR_$VIEW_SET_OBLIQUE
Sets the foreshortening ratio and angle of receding lines for a specific viewport.

GMR_$VIEW_SET_PROJECTION_TYPE
Selects the type of viewing projection for a given viewport.

GMR_$VIEW_SET_REFERENCE_POINT
Sets a viewport's viewing reference point.

GMR_$VIEW_SET_STATE
Establishes all viewing parameters at once for a given viewport.

GMR_$VIEW_SET_TRANSFORM
Sets a viewport's normalizing matrix directly.

GMR_$VIEW_SET_UP_VECTOR
Establishes viewing orientation in a given viewport.

GMR_$VIEW_SET_VIEW_DISTANCE
Sets the distance from the reference point to the viewing plane for a given viewport.

GMR_$VIEW_SET_VIEW_PLANE_NORMAL
Establishes a vector that is normal to the view plane of a specified viewport, in world coordinates.

GMR_$VIEW_SET_WINDOW
Establishes the window on the viewing plane of a viewport.

GMR_$VIEW_SET_YON_DISTANCE
Sets the N-coordinate of the far clipping plane for a given viewport.

Display and Viewport Routines

Setting the Display

GMR_$COORD_INQ_DEVICE_LIMITS
Returns the device coordinates to which the logical device limits are mapped.

GMR_$COORD_INQ_LDC_LIMITS
Returns the current logical device coordinate limits.

GMR_$COORD_INQ_MAX_DEVICE
Returns the maximum range of the device coordinates.

GMR_$COORD_SET_DEVICE_LIMITS
Specifies the limits of device space.

GMR_$COORD_SET_LDC_LIMITS
Specifies the limits of logical device coordinate space.

GMR_$DISPLAY_CLEAR_BG
Clears the background of the display to its current color setting.

GMR_$DISPLAY_INQ_BG_COLOR
Returns the current background color and intensity of the display.

GMR_$DISPLAY_REFRESH
Redisplays all viewports that have a refresh state of GMR_$REFRESH_WAIT, GMR_$REFRESH_UPDATE, or GMR_$REFRESH_PARTIAL.

GMR_$DISPLAY_SET_BG_COLOR
Sets the background color and intensity for the display.

GMR_$DM_REFRESH_ENTRY
Specifies a user-defined routine to be called when the display is refreshed as a result of a Display Manager refresh window or <POP> command.

GMR_$INQ_CONFIG
Returns the number of planes and the size of the current display device.

Setting Viewport Parameters

GMR_$VIEWPORT_CLEAR
Clears a specified viewport to the background color.

GMR_$VIEWPORT_CREATE
Creates an additional viewport and assigns it an ID number.

GMR_$VIEWPORT_DELETE
Deletes a specified viewport.

GMR_$VIEWPORT_INQ_BG_COLOR
Returns the background color ID and intensity of a specified viewport.

GMR_$VIEWPORT_INQ_BORDER
Returns the width of the four edges of the specified viewport, the border-on flag, and color attributes.

GMR_$VIEWPORT_INQ_BOUNDS
Returns the bounds of the specified viewport.

GMR_$VIEWPORT_INQ_CULLING
Returns whether culling is enabled in a specified viewport and the current minimum screen size for displayed structures.

GMR_$VIEWPORT_INQ_GLOBAL_MATRIX
Returns the global modeling matrix for a specific viewport.

GMR_$VIEWPORT_INQ_HILIGHT_ABLOCK
Returns the identification number of the highlighting attribute block of a specified viewport.

GMR_$VIEWPORT_INQ_INVIS_FILTER
Returns the inclusion list and exclusion list for name sets that will be invisible in a specific viewport.

GMR_$VIEWPORT_INQ_PATH_ORDER
Returns the path order used for picking and echoing in a specified viewport.

GMR_$VIEWPORT_INQ_PICK
Returns the pickability range and mask for the specified viewport.

GMR_$VIEWPORT_INQ_PICK_FILTER
Returns the inclusion list and exclusion list for name sets that are pickable for a specific viewport.

GMR_$VIEWPORT_INQ_REFRESH_STATE
Returns the current refresh state of the specified viewport.

GMR_$VIEWPORT_INQ_STATE
Returns all the user-definable parameters of a specified viewport.

GMR_$VIEWPORT_INQ_STRUCTURE
Returns the structure ID and the file ID of the structure that is assigned to a specific viewport.

GMR_$VIEWPORT_INQ_VISIBILITY
Returns the visibility range and mask value for the specified viewport.

GMR_$VIEWPORT_MOVE
Translates the specified viewport, carrying the view with it.

GMR_$VIEWPORT_REFRESH
Redraws the contents of the specified viewport.

GMR_$VIEWPORT_SET_BG_COLOR
Sets the background color and intensity of a specified viewport.

GMR_$VIEWPORT_SET_BORDER
Specifies the border size of the specified viewport in pixels, the border color, and whether the border is displayed.

GMR_$VIEWPORT_SET_BOUNDS
Changes the display bounds for the specified viewport.

GMR_$VIEWPORT_SET_CULLING
Enables culling in a specified viewport: only structures larger than a given screen-space area are displayed.

GMR_$VIEWPORT_SET_GLOBAL_MATRIX
Associates a global modeling matrix with a viewport.

GMR_$VIEWPORT_SET_HILIGHT_ABLOCK
Assigns a highlighting attribute block to a specified viewport.

GMR_$VIEWPORT_SET_INVIS_FILTER
Specifies an inclusion list and an exclusion list for name sets that will be invisible in a specific viewport.

GMR_$VIEWPORT_SET_PATH_ORDER
Sets the order of the path returned by GMR_$PICK for a specified viewport. This value is used by GMR_$INSTANCE_ECHO.

GMR_$VIEWPORT_SET_PICK
Sets the pickability range and mask for the specified viewport

GMR_$VIEWPORT_SET_PICK_FILTER
Specifies an inclusion list and an exclusion list for name sets that are pickable for a specific viewport.

GMR_$VIEWPORT_SET_REFRESH_STATE
Sets the refresh state of a specified viewport.

GMR_$VIEWPORT_SET_STATE
Sets the user-definable parameters for a specified viewport.

GMR_$VIEWPORT_SET_STRUCTURE
Binds a structure (and possibly a subtree) to a viewport for future refreshing.

GMR_$VIEWPORT_SET_VISIBILITY
Sets the visibility range and mask value for the viewport.

Coordinate Transformation Routines

GMR_$COORD_DEVICE_TO_LDC
Converts device coordinates to logical device coordinates.

GMR_$COORD_LDC_TO_DEVICE
Converts logical device coordinates to device coordinates.

GMR_$COORD_LDC_TO_WORK_PLANE
Maps a coordinate in logical device space onto the work plane of the specified viewport. The result is a point in world coordinates.

GMR_$COORD_LDC_TO_WORLD
Maps a point in 3D logical device coordinates into world coordinates via the viewing parameters associated with the specified viewport.

GMR_$COORD_WORLD_TO_LDC
Returns the logical device coordinates of a point specified in world coordinates.

Display-Time Routines

Controlling Visibility

GMR_$STRUCTURE_INQ_VALUE_MASK
Returns the value and mask of a structure that are used to determine visibility and pick eligibility.

GMR_$STRUCTURE_SET_VALUE_MASK
Sets the structure value and structure mask. These two values are tested against the viewport pick and visibility range and masks to determine whether a structure is pickable.

GMR_$VIEWPORT_INQ_INVIS_FILTER
Returns the inclusion list and exclusion list for name sets that will be invisible in a specific viewport.

GMR_$VIEWPORT_INQ_VISIBILITY
Returns the visibility range and mask value for the specified viewport.

GMR_$VIEWPORT_SET_INVIS_FILTER
Specifies an inclusion list and an exclusion list for name sets that will be invisible in a specific viewport.

GMR_$VIEWPORT_SET_VISIBILITY
Sets the visibility range and mask value for the viewport.

Viewport Refresh Routines

GMR_$DISPLAY_REFRESH
Redisplays all viewports that have a refresh state of GMR_$REFRESH_WAIT, GMR_$REFRESH_UPDATE, or GMR_$REFRESH_PARTIAL.

GMR_$DM_REFRESH_ENTRY
Specifies a user-defined routine to be called when the display is refreshed as a result of a Display Manager refresh window or <POP> command.

GMR_$VIEWPORT_INQ_REFRESH_STATE
Returns the current refresh state of the specified viewport.

GMR_$VIEWPORT_REFRESH
Redraws the contents of the specified viewport.

GMR_$VIEWPORT_SET_REFRESH_STATE
Sets the refresh state of a specified viewport.

Dynamic Mode Routines

GMR_$DYN_MODE_INQ_DRAW_METHOD
Returns the type of dynamic drawing method that is enabled for dynamic mode drawing.

GMR_$DYN_MODE_INQ_ENABLE
Returns whether a dynamic mode is enabled and, if so, identifies the path, path depth, and path order.

GMR_$DYN_MODE_SET_DRAW_METHOD
Identifies the type of redraw method that is used when either a viewport is in partial refresh or when dynamic mode is enabled.

GMR_$DYN_MODE_SET_ENABLE
Turns the dynamic mode on and off for viewports that are set for partial refresh.

Double Buffering Routines

GMR_$DBUFF_INQ_MODE
Returns the current mode, which is either single- or double-buffer mode.

GMR_$DBUFF_INQ_SELECT_BUFFER
Returns the number of the buffer that was last selected by GMR_$DBUFF_SET_SELECT_BUFFER for the specified viewport.

GMR_$DBUFF_SET_DISPLAY_BUFFER
Displays the specified buffer, which is either buffer 1 or buffer 2 for the specified viewport.

GMR_$DBUFF_SET_MODE
Sets the current mode to single- or double-buffer mode.

GMR_$DBUFF_SET_SELECT_BUFFER
Indicates which buffer is to be updated.

Input Routines

Setting the Work Plane

GMR_$COORD_INQ_WORK_PLANE
Returns a point in world coordinates and a normal vector that define the work plane associated with a viewport.

GMR_$COORD_LDC_TO_WORK_PLANE
Maps a coordinate in logical device space onto the work plane of the specified viewport. The result is a point in world coordinates.

GMR_$COORD_SET_WORK_PLANE
Establishes a plane for mapping between logical device coordinates and world coordinates.

Controlling the Cursor

GMR_$CURSOR_INQ_ACTIVE
Returns the status of the cursor: displayed or not displayed.

GMR_$CURSOR_INQ_PATTERN
Returns the type, pattern, and offset of the cursor.

GMR_$CURSOR_INQ_POSITION
Returns the position of the cursor.

GMR_$CURSOR_SET_ACTIVE
Specifies whether or not the cursor will be displayed.

GMR_$CURSOR_SET_PATTERN
Specifies a cursor pattern, type, and offset (origin).

GMR_$CURSOR_SET_POSITION
Moves the cursor on the screen.

Controlling Input Operations

GMR_$INPUT_DISABLE
Disables an input event type.

GMR_$INPUT_ENABLE
Enables an input event type.

GMR_$INPUT_EVENT_WAIT
Checks for or waits until an occurrence of an enabled input event.

Picking

GMR_$PICK
Traverses the metafile using the current pick method and returns the path of an element that crosses the pick aperture.

GMR_$PICK_INQ_APERTURE_SIZE
Returns the width, height, and depth of the pick aperture in a specified viewport.

GMR_$PICK_INQ_CENTER
Returns the center of the pick aperture in a specified viewport.

GMR_$PICK_INQ_ECHO_METHOD
Returns the current pick echo method for a viewport.

GMR_$PICK_INQ_METHOD
Returns the pick method in use for a particular viewport.

GMR_$PICK_SET_APERTURE_SIZE
Specifies the width, height, and depth of the pick aperture for a particular viewport.

GMR_$PICK_SET_ECHO_METHOD
Sets the pick echo method for a viewport.

GMR_$PICK_SET_METHOD
Specifies the pick method for a particular viewport.

GMR_$VIEWPORT_INQ_PICK
Returns the pickability range and mask for the specified viewport.

GMR_$VIEWPORT_INQ_PICK_FILTER
Returns the inclusion list and exclusion list for name sets that are pickable for a specific viewport.

GMR_$VIEWPORT_SET_PICK
Sets the pickability range and mask for the specified viewport

GMR_$VIEWPORT_SET_PICK_FILTER
Specifies an inclusion list and an exclusion list for name sets that are pickable for a specific viewport.

Echoing

GMR_$INSTANCE_ECHO
Echos an element or a subtree of an application-supplied instance path in a specific viewport.

GMR_$INSTANCE_ECHO_INQ_METHOD
Returns the instance echo method for a specified viewport.

GMR_$INSTANCE_ECHO_SET_METHOD
Sets the instance echo method for a viewport to either ablock or bounding box.

GMR_$PICK_INQ_ECHO_METHOD
Returns the current pick echo method for a viewport.

GMR_$PICK_SET_ECHO_METHOD
Sets the pick echo method for a viewport.

GMR_$VIEWPORT_INQ_HILIGHT_ABLOCK
Returns the identification number of the highlighting attribute block of a specified viewport.

GMR_$VIEWPORT_SET_HILIGHT_ABLOCK
Assigns a highlighting attribute block to a specified viewport.

Using Color

GMR_$COLOR_DEFINE_HSV
Updates the section of the color map that corresponds to the input color ID using the hue, saturation, and value color model.

GMR_$COLOR_DEFINE_RGB
Updates the section of the color map that corresponds to the input color ID by specifying the amounts of red, green, and blue.

GMR_$COLOR_HSV_TO_RGB
Translates an HSV (hue, saturation, value) color specification to a RGB (red, green, blue) color specification.

GMR_$COLOR_INQ_HSV
Returns the color values at the low and high extremes of the range for a color ID.

GMR_$COLOR_INQ_MAP
Returns the values stored in the current color map.

GMR_$COLOR_INQ_RANGE
Accepts a color ID and returns the starting color map index and the range of color map indices for the color ID.

GMR_$COLOR_INQ_RGB
Returns the color values at the low and high extremes of the range for a color ID.

GMR_$COLOR_RGB_TO_HSV
Translates an RGB (red, green, blue) color specification to an HSV (hue, saturation, value) color specification.

GMR_$COLOR_SET_MAP
Updates the current color map.

GMR_$COLOR_SET_RANGE

Accepts a color ID number, a start index in the color map, and a range that is the number of contiguous color map indices to associate with the color ID.

Output Routines

GMR_$PRINT_DISPLAY

Creates a POSTSCRIPT file from the entire 3D GMR display.

GMR_$PRINT_VIEWPORT

Creates a POSTSCRIPT file from a single, specified viewport.

Format for User-Callable Routines: Alphabetical Listing

```
GMR_$4X3_MATRIX_CONCATENATE (matrix1, matrix2, matrix, status)

GMR_$4X3_MATRIX_IDENTITY (matrix, status)

GMR_$4X3_MATRIX_INVERT (matrix, inverse, status)

GMR_$4X3_MATRIX_REFLECT (order, point, vector, matrix, status)

GMR_$4X3_MATRIX_ROTATE (order, axis, angle, matrix, status)

GMR_$4X3_MATRIX_ROTATE_AXIS (order, point, vector, angle, matrix, status)

GMR_$4X3_MATRIX_SCALE (order, scale, matrix, status)

GMR_$4X3_MATRIX_TRANSLATE (order, translation, matrix, status)

GMR_$ABLOCK_ASSIGN_DISPLAY (aclass_id, ablock_id, status)

GMR_$ABLOCK_ASSIGN_VIEWPORT (aclass_id, viewport_id, ablock_id, status)

GMR_$ABLOCK_COPY (source_ablock_id, destination_ablock_id, status)

GMR_$ABLOCK_CREATE (source_ablock_id, ablock_id, status)

GMR_$ABLOCK_DELETE (ablock_id, status)

GMR_$ABLOCK_INQ_ASSIGN_DISPLAY (aclass_id, ablock_id, status)

GMR_$ABLOCK_INQ_ASSIGN_VIEWPORT (aclass_id, viewport_id, ablock_id, status)

GMR_$ABLOCK_INQ_FILL_COLOR (ablock_id, color, enable_state, status)

GMR_$ABLOCK_INQ_FILL_INTEN (ablock_id, intensity, enable_state, status)

GMR_$ABLOCK_INQ_LINE_COLOR (ablock_id, color, enable_state, status)

GMR_$ABLOCK_INQ_LINE_INTEN (ablock_id, intensity, enable_state, status)

GMR_$ABLOCK_INQ_LINE_TYPE (ablock_id, type_id, change, status)

GMR_$ABLOCK_INQ_MARK_COLOR (ablock_id, color, enable_state, status)

GMR_$ABLOCK_INQ_MARK_INTEN (ablock_id, intensity, enable_state, status)

GMR_$ABLOCK_INQ_MARK_SCALE (ablock_id, scale_factor, enable_state, status)
```

```
GMR_$ABLOCK_INQ_MARK_INTEN (ablock_id, type, enable_state, status)

GMR_$ABLOCK_INQ_TEXT_COLOR (ablock_id, color, enable_state, status)

GMR_$ABLOCK_INQ_TEXT_EXPANSION (ablock_id, expansion, enable_state, status)

GMR_$ABLOCK_INQ_TEXT_HEIGHT (ablock_id, height, enable_state, status)

GMR_$ABLOCK_INQ_TEXT_INTEN (ablock_id, intensity, enable_state, status)

GMR_$ABLOCK_INQ_TEXT_PATH (ablock_id, path, enable_state, status)

GMR_$ABLOCK_INQ_TEXT_SLANT (ablock_id, slant, enable_state, status)

GMR_$ABLOCK_INQ_TEXT_SPACING (ablock_id, spacing, enable_state, status)

GMR_$ABLOCK_INQ_TEXT_UP (ablock_id, up_vector, enable_state, status)

GMR_$ABLOCK_SET_FILL_COLOR (ablock_id, color, enable_state, status)

GMR_$ABLOCK_SET_FILL_INTEN (ablock_id, intensity, enable_state, status)

GMR_$ABLOCK_SET_LINE_COLOR (ablock_id, color, enable_state, status)

GMR_$ABLOCK_SET_LINE_INTEN (ablock_id, intensity, enable_state, status)

GMR_$ABLOCK_SET_LINE_TYPE (ablock_id, type_id, change, status)

GMR_$ABLOCK_SET_MARK_COLOR (ablock_id, color, enable_state, status)

GMR_$ABLOCK_SET_MARK_INTEN (ablock_id, intensity, enable_state, status)

GMR_$ABLOCK_SET_MARK_SCALE (ablock_id, scale_factor, enable_state, status)

GMR_$ABLOCK_SET_MARK_TYPE (ablock_id, type, enable_state, status)

GMR_$ABLOCK_SET_TEXT_COLOR (ablock_id, color, enable_state, status)

GMR_$ABLOCK_SET_TEXT_EXPANSION (ablock_id, expansion, enable_state, status)

GMR_$ABLOCK_SET_TEXT_HEIGHT (ablock_id, height, enable_state, status)

GMR_$ABLOCK_SET_TEXT_INTEN (ablock_id, intensity, enable_state, status)

GMR_$ABLOCK_SET_TEXT_PATH (ablock_id, angle, enable_state, status)

GMR_$ABLOCK_SET_TEXT_SLANT (ablock_id, slant, enable_state, status)

GMR_$ABLOCK_SET_TEXT_SPACING (ablock_id, spacing, enable_state, status)
```

```
GMR_$ABLOCK_SET_TEXT_UP (ablock_id, up_vector, enable_state, status)

GMR_$ACLASS (aclass_id, status)

GMR_$ADD_NAME_SET (n_names, name_set, status)

GMR_$ATTRIBUTE_SOURCE (attribute, source, status)

GMR_$COLOR_DEFINE_HSV (color_id, low_color, high_color, status)

GMR_$COLOR_DEFINE_RGB (color_id, low_color, high_color, status)

GMR_$COLOR_HSV_TO_RGB (hsv_color, rgb_color, status)

GMR_$COLOR_INQ_HSV (color_id, inq_type, low_color, high_color, status)

GMR_$COLOR_INQ_MAP (start_index, count, color_array, status)

GMR_$COLOR_INQ_RANGE (color_id, start, range, status)

GMR_$COLOR_INQ_RGB (color_id, inq_type, low_color, high_color, status)

GMR_$COLOR_RGB_TO_HSV (rgb_color, hsv_color, status)

GMR_$COLOR_SET_MAP (start_index, range, color_array, status)

GMR_$COLOR_SET_RANGE (color_id, start, range, status)

GMR_$COORD_DEVICE_TO_LDC (device_coord, ldc_coord, status)

GMR_$COORD_INQ_DEVICE_LIMITS (device_limits, status)

GMR_$COORD_INQ_LDC_LIMITS (ldc_limits, status)

GMR_$COORD_INQ_MAX_DEVICE (max_device, status)

GMR_$COORD_INQ_WORK_PLANE (viewport_id, point, normal, status)

$GMR_$COORD_LDC_TO_DEVICE (ldc_coord, device_coord, status)

GMR_$COORD_LDC_TO_WORK_PLANE (viewport_id, ldc_coord, plane_coord, status)

GMR_$COORD_LDC_TO_WORLD (viewport_id, ldc_coord, world_coord, status)

GMR_$COORD_SET_DEVICE_LIMITS (device_limits, status)

GMR_$COORD_SET_LDC_LIMITS (ldc_limits, status)

GMR_$COORD_SET_WORK_PLANE (viewport_id, point, normal, status)
```

```
GMR_$COORD_WORLD TO_LDC (viewport_id, world_coord, ldc_coord, status)

GMR_$CURSOR_INQ_ACTIVE (active, status)

GMR_$CURSOR_INQ_PATTERN (style, pattern_size, pattern, offset, status)

GMR_$CURSOR_INQ_POSITION (position, status)

GMR_$CURSOR_SET_ACTIVE (active, status)

GMR_$CURSOR_SET_PATTERN (style, pattern_size, pattern, offset, status)

GMR_$CURSOR_SET_POSITION (position, status)

GMR_$DBUFF_INQ_MODE (buffer_mode, status)

GMR_$DBUFF_INQ_SELECT_BUFFER (viewport_id, buffer_number, status)

GMR_$DBUFF_SET_DISPLAY_BUFFER (buffer_number, viewport_id, status)

GMR_$DBUFF_SET_MODE (buffer_mode, status)

GMR_$DBUFF_SET_SELECT_BUFFER (buffer_number, viewport_id, status)

GMR_$DISPLAY_CLEAR_BG (status)

GMR_$DISPLAY_INQ_BG_COLOR (bg_type, color_id, intensity, status)

GMR_$DISPLAY_REFRESH (status)

GMR_$DISPLAY_SET_BG_COLOR (bg_type, color_id, intensity, status)

GMR_$DM_$REFRESH_ENTRY (refresh_procedure_ptr, status)

GMR_$DYN_MODE_INQ_DRAW_METHOD( draw_method, status )

GMR_$DYN_MODE_INQ_ENABLE (enabled, dyn_instance_path, dyn_path_depth,
                          dyn_path_order, status)

GMR_$DYN_MODE_SET_DRAW_METHOD (draw_method, status)

GMR_$DYN_MODE_SET_ENABLE (enabled, dyn_instance_path, dyn_path_depth,
                          dyn_path_order, status)

GMR_$ELEMENT_DELETE (status)

GMR_$ELEMENT_INQ_INDEX (element_index, status)

GMR_$ELEMENT_SET_INDEX (element_index, status)
```

```
GMR_$F3_MESH (major_dim_of_mesh, minor_dim_of_mesh, points, status)

GMR_$F3_MULTILINE (n_points, points, status)

GMR_$F3_POLYGON (n_points, points, status)

GMR_$F3_POLYLINE (n_points, points, closed, status)

GMR_$F3_POLYMARKER (n_points, points, status)

GMR_$FILE_CLOSE (save, status)

GMR_$FILE_CREATE (name, name_length, access, concurrency, file_id, status)

GMR_$FILE_INQ_PRIMARY_STRUCTURE (structure_id, status)

GMR_$FILE_OPEN (name, name_length, access, concurrency, file_id, status)

GMR_$FILE_SELECT (file_id, status)

GMR_$FILE_SET_PRIMARY_STRUCTURE (structure_id, status)

GMR_$FILL_COLOR (color, status)

GMR_$FILL_INTEN (intensity, status)

GMR_$INIT (display_mode, unit, size, n_planes, status)

GMR_$INPUT_DISABLE (event_type, status)

GMR_$INPUT_ENABLE (event_type, key_set, status)

GMR_$INPUT_EVENT_WAIT (wait, event_type, event_data, position, status)

GMR_$INQ_ACLASS (aclass_id, status)

GMR_$INQ_ADD_NAME_SET (n_names, name set, status)

GMR_$INQ_ATTRIBUTE_SOURCE (attribute, source, status)

GMR_$INQ_CONFIG (op, unit_or_pad, numplanes, size, status)

GMR_$INQ_ELEMENT_TYPE (element_type, attribute_type, status)

GMR_$INQ_F3_MESH (array_size, major_dim_of_mesh, minor_dim_of_mesh,
                  points, status)

GMR_$INQ_F3_MULTILINE (array_size, n_points, points, status)
```

```
GMR_$INQ_F3_POLYGON (array_size, n_points, points, status)

GMR_$INQ_F3_POLYLINE (array_size, n_points, points, closed, status)

GMR_$INQ_F3_POLYMARKER (array_size, n_points, points, status)

GMR_$INQ_FILL_COLOR (color_id, status)

GMR_$INQ_FILL_INTEN (intensity, status)

GMR_$INQ_INSTANCE_TRANSFORM (structure_id, transform_matrix, status)

GMR_$INQ_LINE_COLOR (color, status)

GMR_$INQ_LINE_INTEN (inten, status)

GMR_$INQ_LINE_TYPE (type_id, status)

GMR_$INQ_MARK_COLOR (color_id, status)

GMR_$INQ_MARK_INTEN (intensity, status)

GMR_$INQ_MARK_SCALE (scale_factor, status)

GMR_$INQ_MARK_INTEN (type, status)

GMR_$INQ_REMOVE_NAME_SET (n_names, name set, status)

GMR_$INQ_TAG (tag_start, tag_copy, tag, tag_length, status)

GMR_$INQ_TEXT (string, string_length, position, status)

GMR_$INQ_TEXT_COLOR (color_id, status)

GMR_$INQ_TEXT_EXPANSION (width_to_height_ratio, status)

GMR_$INQ_TEXT_HEIGHT (height, status)

GMR_$INQ_TEXT_INTEN (intensity, status)

GMR_$INQ_TEXT_PATH (angle, status)

GMR_$INQ_TEXT_SLANT (slant, status)

GMR_$INQ_TEXT_SPACING (spacing, status)

GMR_$INQ_TEXT_UP (up_vector, status)

GMR_$INSTANCE_ECHO (viewport_id, depth, path, status)
```

```
GMR_$INSTANCE_ECHO_INQ_METHOD (viewport_id, method, status)

GMR_$INSTANCE_ECHO_SET_METHOD (viewport_id, method, status)

GMR_$INSTANCE_TRANSFORM (structure_id, trans_matrix, status)

GMR_$INSTANCE_TRANSFORM_FWD_REF (name, namelength, trans_matrix,
                                 structure_id, status)

GMR_$LINE_COLOR (color, status)

GMR_$LINE_INTEN (intensity, status)

GMR_$LINE_TYPE (type_id, status)

GMR_$MARK_COLOR (color, status)

GMR_$MARK_INTEN (intensity, status)

GMR_$MARK_SCALE (scale_factor, status)

GMR_$MARK_TYPE (type, status)

GMR_$PICK (viewport_id, center, pick_index, pick_data_size, pick_data,
           status)

GMR_$PICK_INQ_APERTURE_SIZE (viewport_id, width, height, depth, status)

GMR_$PICK_INQ_CENTER (viewport_id, center, status)

GMR_$PICK_INQ_ECHO_METHOD (viewport_id, method, status)

GMR_$PICK_INQ_METHOD (viewport_id, method, status)

GMR_$PICK_SET_APERTURE_SIZE (viewport_id, width, height, depth, status)

GMR_$PICK_SET_ECHO_METHOD (viewport_id, method, status)

GMR_$PICK_SET_METHOD (viewport_id, method, status)

GMR_$PRINT_DISPLAY (name, namelength, print_style, paper_width,
                    paper_height, status )

GMR_$PRINT_VIEWPORT (viewport_id, name, namelength, print_style,
                     paper_width, paper_height, status)

GMR_$REMOVE_NAME_SET (n_names, name set, status)

GMR_$REPLACE_INQ_FLAG (yes_no, status)
```

```
GMR_$REPLACE_SET_FLAG (yes_no, status)

GMR_$STRUCTURE_CLOSE (save, status)

GMR_$STRUCTURE_COPY (file_id, structure_id, status)

GMR_$STRUCTURE_CREATE (name, name_length, structure_id, status)

GMR_$STRUCTURE_DELETE (status)

GMR_$STRUCTURE_ERASE (status)

GMR_$STRUCTURE_INQ_BOUNDS(structure_id, bounds, status)

GMR_$STRUCTURE_INQ_COUNT (count, max_structure_id, status)

GMR_$STRUCTURE_INQ_ID (name, name_length, structure_id, status)

GMR_$STRUCTURE_INQ_INSTANCES (structure_id, n_instances, max_depth, status)

GMR_$STRUCTURE_INQ_NAME (structure_id, name, name_length, status)

GMR_$STRUCTURE_INQ_OPEN (structure_id, status)

GMR_$STRUCTURE_INQ_TEMPORARY (structure_id, temporary, status)

GMR_$STRUCTURE_INQ_VALUE_MASK (structure_id, value, mask, status)

GMR_$STRUCTURE_OPEN (structure_id, status)

GMR_$STRUCTURE_SET_NAME (structure_id, name, name_length, status)

GMR_$STRUCTURE_SET_TEMPORARY (structure_id, temporary, status)

GMR_$STRUCTURE_SET_VALUE_MASK (structure_id, value, mask, status)

GMR_$TAG (tag, tag_length, status)

GMR_$TAG_LOCATE (text_locate, text_length, structure_start, structure_stop,
                 index_start, index_stop, character_start, character_stop,
                 tag_structure, tag_index, tag_character, status)

GMR_$TERMINATE (status)

GMR_$TEXT (string, string_length, position, status)

GMR_$TEXT_COLOR (color_id, status)

GMR_$TEXT_EXPANSION (width_to_height_ratio, status)
```

```
GMR_$TEXT_HEIGHT (height, status)

GMR_$TEXT_INQ_ANCHOR_CLIP (anchor_clip, status)

GMR_$TEXT_INTEN (intensity, status)

GMR_$TEXT_PATH (angle, status)

GMR_$TEXT_SET_ANCHOR_CLIP (anchor_clip, status)

GMR_$TEXT_SLANT (slant, status)

GMR_$TEXT_SPACING (spacing, status)

GMR_$TEXT_UP (up_vector, status)

GMR_$VIEW_INQ_COORD_SYSTEM (viewport_id, coord_system, status)

GMR_$VIEW_INQ_HITHER_DISTANCE (viewport_id, hither_dist, status)

GMR_$VIEW_INQ_OBLIQUE (viewport_id, foreshorten, recede, status)

GMR_$VIEW_INQ_PROJECTION_TYPE (viewport_id, proj_type, status)

GMR_$VIEW_INQ_REFERENCE_POINT (viewport_id, reference, status)

GMR_$VIEW_INQ_STATE (viewport_id, view_state, status)

GMR_$VIEW_INQ_TRANSFORM (viewport_id, matrix, projection_type, clip_zmin,
                         status)

GMR_$VIEW_INQ_UP_VECTOR (viewport_id, up, status)

GMR_$VIEW_INQ_VIEW_DISTANCE (viewport_id, view_dist, status)

GMR_$VIEW_INQ_VIEW_PLANE_NORMAL (viewport_id, normal, status)

GMR_$VIEW_INQ_WINDOW (viewport_id, window, status)

GMR_$VIEW_INQ_YON_DISTANCE (viewport_id, yon_distance, status)

GMR_$VIEW_SET_COORD_SYSTEM (viewport_id, coord_system, status)

GMR_$VIEW_SET_HITHER_DISTANCE (viewport_id, hither_dist, status)

GMR_$VIEW_SET_OBLIQUE (viewport_id, foreshorten, recede, status)

GMR_$VIEW_SET_PROJECTION_TYPE (viewport_id, proj_type, status)
```

```
GMR_$VIEW_SET_REFERENCE_POINT (viewport_id, reference, status)

GMR_$VIEW_SET_STATE (viewport_id, view_state, status)

GMR_$VIEW_SET_TRANSFORM (viewport_id, matrix, projection_type, clip_zmin,
                         status)

GMR_$VIEW_SET_UP_VECTOR (viewport_id, up, status)

GMR_$VIEW_SET_VIEW_DISTANCE (viewport_id, view_dist, status )

GMR_$VIEW_SET_VIEW_PLANE_NORMAL(viewport_id, normal, status)

GMR_$VIEW_SET_WINDOW (viewport_id, window, status)

GMR_$VIEW_SET_YON_DISTANCE (viewport_id, yon_distance, status)

GMR_$VIEWPORT_CLEAR (viewport_id, status)

GMR_$VIEWPORT_CREATE (vbounds, viewport_id, status)

GMR_$VIEWPORT_DELETE (viewport_id, status)

GMR_$VIEWPORT_INQ_BG_COLOR (viewport_id, color_id, intensity, status)

GMR_$VIEWPORT_INQ_BORDER (viewport_id, border_width, border_on,
                          border_color_id, border_inten, status)

GMR_$VIEWPORT_INQ_BOUNDS (viewport_id, vbounds, status)

GMR_$VIEWPORT_INQ_CULLING (viewport_id, cull, min_area, status)

GMR_$VIEWPORT_INQ_GLOBAL_MATRIX (viewport_id, matrix, status)

GMR_$VIEWPORT_INQ_HILIGHT_ABLOCK (viewport_id, ablock_id, status)

GMR_$VIEWPORT_INQ_INVIS_FILTER (viewport_id, n_incl_names, inclusion_set,
                                n_excl_names, exclusion_set, status)

GMR_$VIEWPORT_INQ_PATH_ORDER (viewport_id, order, status)

GMR_$VIEWPORT_INQ_PICK (viewport_id, pick_low_range, pick_high_range,
                        pick_mask, status)

GMR_$VIEWPORT_INQ_PICK_FILTER (viewport_id, n_incl_names, inclusion_set,
                               n_excl_names, exclusion_set, status)

GMR_$VIEWPORT_INQ_REFRESH_STATE (viewport_id, refresh_state, status)
```

```
GMR_$VIEWPORT_INQ_STATE (viewport_id, array_size, size, viewport_state,
                         status)

GMR_$VIEWPORT_INQ_STRUCTURE (viewport_id, structure_id, file_id, status)

GMR_$VIEWPORT_INQ_VISIBILITY (viewport_id, vis_low_range, vis_high_range,
                              vis_mask, status)

GMR_$VIEWPORT_MOVE (viewport_id, translate, status)

GMR_$VIEWPORT_REFRESH (viewport_id, status)

GMR_$VIEWPORT_SET_BG_COLOR (viewport_id, color_id, intensity, status)

GMR_$VIEWPORT_SET_BORDER (viewport_id, border_size, border_on,
                          border_color_id, border_inten, status)

GMR_$VIEWPORT_SET_BOUNDS (viewport_id, vbounds, status)

GMR_$VIEWPORT_SET_CULLING (viewport_id, cull, min_area, status)

GMR_$VIEWPORT_SET_GLOBAL_MATRIX (viewport_id, matrix, status)

GMR_$VIEWPORT_SET_HILIGHT_ABLOCK (viewport_id, ablock_id, status)

GMR_$VIEWPORT_SET_INVIS_FILTER (viewport_id, n_incl_names, inclusion_set,
                                n_excl_names, exclusion_set, status)

GMR_$VIEWPORT_SET_PATH_ORDER (viewport_id, order, status)

GMR_$VIEWPORT_SET_PICK (viewport_id, pick_low_range, pick_high_range,
                        pick_mask, status)

GMR_$VIEWPORT_SET_PICK_FILTER (viewport_id, n_incl_names, inclusion_set,
                               n_excl_names, exclusion_set, status)

GMR_$VIEWPORT_SET_REFRESH_STATE (viewport_id, refresh_state, status )

GMR_$VIEWPORT_SET_STATE (viewport_id, viewport_state, status)

GMR_$VIEWPORT_SET_STRUCTURE (viewport_id, structure_id, status)

GMR_$VIEWPORT_SET_VISIBILITY (viewport_id, vis_low_range, vis_high_range,
                              vis_mask, status)
```

Index

GMR_$4X3_MATRIX_CONCATENATE 2-2
GMR_$4X3_MATRIX_IDENTITY 2-3
GMR_$4X3_MATRIX_INVERT 2-4
GMR_$4X3_MATRIX_REFLECT 2-5
GMR_$4X3_MATRIX_ROTATE 2-6
GMR_$4X3_MATRIX_ROTATE_AXIS 2-7
GMR_$4X3_MATRIX_SCALE 2-8
GMR_$4X3_MATRIX_TRANSLATE 2-9
GMR_$ABLOCK_ASSIGN_DISPLAY 2-10
GMR_$ABLOCK_ASSIGN_VIEWPORT 2-11
GMR_$ABLOCK_COPY 2-12
GMR_$ABLOCK_CREATE 2-13
GMR_$ABLOCK_DELETE 2-14
GMR_$ABLOCK_INQ_ASSIGN_DISPLAY 2-15
GMR_$ABLOCK_INQ_ASSIGN_VIEWPORT 2-16
GMR_$ABLOCK_INQ_FILL_COLOR 2-17
GMR_$ABLOCK_INQ_FILL_INTEN 2-18
GMR_$ABLOCK_INQ_LINE_COLOR 2-19
GMR_$ABLOCK_INQ_LINE_INTEN 2-20
GMR_$ABLOCK_INQ_LINE_TYPE 2-21
GMR_$ABLOCK_INQ_MARK_COLOR 2-22
GMR_$ABLOCK_INQ_MARK_INTEN 2-23
GMR_$ABLOCK_INQ_MARK_SCALE 2-24
GMR_$ABLOCK_INQ_MARK_TYPE 2-25
GMR_$ABLOCK_INQ_TEXT_COLOR 2-26
GMR_$ABLOCK_INQ_TEXT_EXPANSION 2-27
GMR_$ABLOCK_INQ_TEXT_HEIGHT 2-28
GMR_$ABLOCK_INQ_TEXT_INTEN 2-29
GMR_$ABLOCK_INQ_TEXT_PATH 2-30
GMR_$ABLOCK_INQ_TEXT_SLANT 2-31
GMR_$ABLOCK_INQ_TEXT_SPACING 2-32
GMR_$ABLOCK_INQ_TEXT_UP 2-33
GMR_$ABLOCK_SET_FILL_COLOR 2-34
GMR_$ABLOCK_SET_FILL_INTEN 2-35
GMR_$ABLOCK_SET_LINE_COLOR 2-36
GMR_$ABLOCK_SET_LINE_INTEN 2-37
GMR_$ABLOCK_SET_LINE_TYPE 2-38
GMR_$ABLOCK_SET_MARK_COLOR 2-40
GMR_$ABLOCK_SET_MARK_INTEN 2-41
GMR_$ABLOCK_SET_MARK_SCALE 2-42
GMR_$ABLOCK_SET_MARK_TYPE 2-44
GMR_$ABLOCK_SET_TEXT_COLOR 2-46

GMR_$ABLOCK_SET_TEXT_EXPANSION 2-47
GMR_$ABLOCK_SET_TEXT_HEIGHT 2-49
GMR_$ABLOCK_SET_TEXT_INTEN 2-50
GMR_$ABLOCK_SET_TEXT_PATH 2-51
GMR_$ABLOCK_SET_TEXT_SLANT 2-53
GMR_$ABLOCK_SET_TEXT_SPACING 2-55
GMR_$ABLOCK_SET_TEXT_UP 2-57
GMR_$ACLASS 2-59
GMR_$ADD_NAME_SET 2-60
GMR_$ATTRIBUTE_SOURCE 2-63
GMR_$COLOR_DEFINE_HSV 2-65
GMR_$COLOR_DEFINE_RGB 2-66
GMR_$COLOR_HSV_TO_RGB 2-67
GMR_$COLOR_INQ_HSV 2-68
GMR_$COLOR_INQ_MAP 2-69
GMR_$COLOR_INQ_RANGE 2-70
GMR_$COLOR_INQ_RGB 2-71
GMR_$COLOR_RGB_TO_HSV 2-72
GMR_$COLOR_SET_MAP 2-73
GMR_$COLOR_SET_RANGE 2-74
GMR_$COORD_DEVICE_TO_LDC 2-75
GMR_$COORD_INQ_DEVICE_LIMITS 2-76
GMR_$COORD_INQ_LDC_LIMITS 2-77
GMR_$COORD_INQ_MAX_DEVICE 2-78
GMR_$COORD_INQ_WORK_PLANE 2-79
GMR_$COORD_LDC_TO_DEVICE 2-80
GMR_$COORD_LDC_TO_WORK_PLANE 2-81
GMR_$COORD_LDC_TO_WORLD 2-82
GMR_$COORD_SET_DEVICE_LIMITS 2-83
GMR_$COORD_SET_LDC_LIMITS 2-84
GMR_$COORD_SET_WORK_PLANE 2-85
GMR_$COORD_WORLD_TO_LDC 2-86
GMR_$CURSOR_INQ_ACTIVE 2-87
GMR_$CURSOR_INQ_PATTERN 2-88
GMR_$CURSOR_INQ_POSITION 2-89
GMR_$CURSOR_SET_ACTIVE 2-90
GMR_$CURSOR_SET_PATTERN 2-91
GMR_$CURSOR_SET_POSITION 2-93
GMR_$DBUFF_INQ_MODE 2-94
GMR_$DBUFF_INQ_SELECT_BUFFER 2-95
GMR_$DBUFF_SET_DISPLAY_BUFFER 2-96
GMR_$DBUFF_SET_MODE 2-97
GMR_$DBUFF_SET_SELECT_BUFFER 2-98
GMR_$DISPLAY_CLEAR_BG 2-99
GMR_$DISPLAY_INQ_BG_COLOR 2-100

GMR_$DISPLAY_REFRESH 2-101
GMR_$DISPLAY_SET_BG_COLOR 2-102
GMR_$DM_REFRESH_ENTRY 2-103
GMR_$DYN_MODE_INQ_DRAW_METHOD 2-105
GMR_$DYN_MODE_INQ_ENABLE 2-106
GMR_$DYN_MODE_SET_DRAW_METHOD 2-108
GMR_$DYN_MODE_SET_ENABLE 2-109
GMR_$ELEMENT_DELETE 2-111
GMR_$ELEMENT_INQ_INDEX 2-112
GMR_$ELEMENT_SET_INDEX 2-113
GMR_$F3_MESH 2-114
GMR_$F3_MULTILINE 2-116
GMR_$F3_POLYGON 2-117
GMR_$F3_POLYLINE 2-118
GMR_$F3_POLYMARKER 2-119
GMR_$FILE_CLOSE 2-121
GMR_$FILE_CREATE 2-122
GMR_$FILE_INQ_PRIMARY_STRUCTURE 2-124
GMR_$FILE_OPEN 2-125
GMR_$FILE_SELECT 2-127
GMR_$FILE_SET_PRIMARY_STRUCTURE 2-128
GMR_$FILL_COLOR 2-129
GMR_$FILL_INTEN 2-130
GMR_$INIT 2-131
GMR_$INPUT_DISABLE 2-133
GMR_$INPUT_ENABLE 2-134
GMR_$INPUT_EVENT_WAIT 2-138
GMR_$INQ_ACLASS 2-140
GMR_$INQ_ADD_NAME_SET 2-141
GMR_$INQ_ATTRIBUTE_SOURCE 2-142
GMR_$INQ_CONFIG 2-144
GMR_$INQ_ELEMENT_TYPE 2-145
GMR_$INQ_F3_MESH 2-146
GMR_$INQ_F3_MULTILINE 2-147
GMR_$INQ_F3_POLYGON 2-148
GMR_$INQ_F3_POLYLINE 2-149
GMR_$INQ_F3_POLYMARKER 2-150
GMR_$INQ_FILL_COLOR 2-151
GMR_$INQ_FILL_INTEN 2-152
GMR_$INQ_INSTANCE_TRANSFORM 2-153
GMR_$INQ_LINE_COLOR 2-154
GMR_$INQ_LINE_INTEN 2-155
GMR_$INQ_LINE_TYPE 2-156
GMR_$INQ_MARK_COLOR 2-157
GMR_$INQ_MARK_INTEN 2-158

GMR_$INQ_MARK_SCALE 2-159
GMR_$INQ_MARK_TYPE 2-160
GMR_$INQ_REMOVE_NAME_SET 2-161
GMR_$INQ_TAG 2-162
GMR_$INQ_TEXT 2-163
GMR_$INQ_TEXT_COLOR 2-164
GMR_$INQ_TEXT_EXPANSION 2-165
GMR_$INQ_TEXT_HEIGHT 2-166
GMR_$INQ_TEXT_INTEN 2-167
GMR_$INQ_TEXT_PATH 2-168
GMR_$INQ_TEXT_SLANT 2-169
GMR_$INQ_TEXT_SPACING 2-170
GMR_$INQ_TEXT_UP 2-171
GMR_$INSTANCE_ECHO 2-172
GMR_$INSTANCE_ECHO_INQ_METHOD 2-173
GMR_$INSTANCE_ECHO_SET_METHOD 2-174
GMR_$INSTANCE_TRANSFORM 2-175
GMR_$INSTANCE_TRANSFORM_FWD_REF 2-177
GMR_$LINE_COLOR 2-178
GMR_$LINE_INTEN 2-179
GMR_$LINE_TYPE 2-180
GMR_$MARK_COLOR 2-181
GMR_$MARK_INTEN 2-182
GMR_$MARK_SCALE 2-183
GMR_$MARK_TYPE 2-184
GMR_$PICK 2-185
GMR_$PICK_INQ_APERTURE_SIZE 2-188
GMR_$PICK_INQ_CENTER 2-190
GMR_$PICK_INQ_ECHO_METHOD 2-191
GMR_$PICK_INQ_METHOD 2-192
GMR_$PICK_SET_APERTURE_SIZE 2-193
GMR_$PICK_SET_ECHO_METHOD 2-195
GMR_$PICK_SET_METHOD 2-197
GMR_$PRINT_DISPLAY 2-199
GMR_$PRINT_VIEWPORT 2-201
GMR_$REMOVE_NAME_SET 2-202
GMR_$REPLACE_INQ_FLAG 2-203
GMR_$REPLACE_SET_FLAG 2-204
GMR_$STRUCTURE_CLOSE 2-205
GMR_$STRUCTURE_COPY 2-206
GMR_$STRUCTURE_CREATE 2-207
GMR_$STRUCTURE_DELETE 2-209
GMR_$STRUCTURE_ERASE 2-211
GMR_$STRUCTURE_INQ_BOUNDS 2-212
GMR_$STRUCTURE_INQ_COUNT 2-213

GMR_$STRUCTURE_INQ_ID 2-214
GMR_$STRUCTURE_INQ_INSTANCES 2-215
GMR_$STRUCTURE_INQ_NAME 2-216
GMR_$STRUCTURE_INQ_OPEN 2-217
GMR_$STRUCTURE_INQ_TEMPORARY 2-218
GMR_$STRUCTURE_INQ_VALUE_MASK 2-219
GMR_$STRUCTURE_OPEN 2-220
GMR_$STRUCTURE_SET_NAME 2-221
GMR_$STRUCTURE_SET_TEMPORARY 2-222
GMR_$STRUCTURE_SET_VALUE_MASK 2-223
GMR_$TAG 2-225
GMR_$TAG_LOCATE 2-226
GMR_$TERMINATE 2-228
GMR_$TEXT 2-229
GMR_$TEXT_COLOR 2-230
GMR_$TEXT_EXPANSION 2-231
GMR_$TEXT_HEIGHT 2-232
GMR_$TEXT_INQ_ANCHOR_CLIP 2-233
GMR_$TEXT_INTEN 2-234
GMR_$TEXT_PATH 2-235
GMR_$TEXT_SET_ANCHOR_CLIP 2-236
GMR_$TEXT_SLANT 2-237
GMR_$TEXT_SPACING 2-238
GMR_$TEXT_UP 2-239
GMR_$VIEW_INQ_COORD_SYSTEM 2-240
GMR_$VIEW_INQ_HITHER_DISTANCE 2-241
GMR_$VIEW_INQ_OBLIQUE 2-242
GMR_$VIEW_INQ_PROJECTION_TYPE 2-243
GMR_$VIEW_INQ_REFERENCE_POINT 2-244
GMR_$VIEW_INQ_STATE 2-245
GMR_$VIEW_INQ_TRANSFORM 2-246
GMR_$VIEW_INQ_UP_VECTOR 2-247
GMR_$VIEW_INQ_VIEW_DISTANCE 2-248
GMR_$VIEW_INQ_VIEW_PLANE_NORMAL 2-249
GMR_$VIEW_INQ_WINDOW 2-250
GMR_$VIEW_INQ_YON_DISTANCE 2-251
GMR_$VIEW_SET_COORD_SYSTEM 2-252
GMR_$VIEW_SET_HITHER_DISTANCE 2-253
GMR_$VIEW_SET_OBLIQUE 2-254
GMR_$VIEW_SET_PROJECTION_TYPE 2-255
GMR_$VIEW_SET_REFERENCE_POINT 2-256
GMR_$VIEW_SET_STATE 2-257
GMR_$VIEW_SET_TRANSFORM 2-258
GMR_$VIEW_SET_UP_VECTOR 2-259
GMR_$VIEW_SET_VIEW_DISTANCE 2-260

GMR_$VIEW_SET_VIEW_PLANE_NORMAL 2-261
GMR_$VIEW_SET_WINDOW 2-262
GMR_$VIEW_SET_YON_DISTANCE 2-263
GMR_$VIEWPORT_CLEAR 2-264
GMR_$VIEWPORT_CREATE 2-265
GMR_$VIEWPORT_DELETE 2-266
GMR_$VIEWPORT_INQ_BG_COLOR 2-267
GMR_$VIEWPORT_INQ_BORDER 2-268
GMR_$VIEWPORT_INQ_BOUNDS 2-269
GMR_$VIEWPORT_INQ_CULLING 2-270
GMR_$VIEWPORT_INQ_GLOBAL_MATRIX 2-271
GMR_$VIEWPORT_INQ_HILIGHT_ABLOCK 2-272
GMR_$VIEWPORT_INQ_INVIS_FILTER 2-273
GMR_$VIEWPORT_INQ_PATH_ORDER 2-274
GMR_$VIEWPORT_INQ_PICK 2-275
GMR_$VIEWPORT_INQ_PICK_FILTER 2-276
GMR_$VIEWPORT_INQ_REFRESH_STATE 2-277
GMR_$VIEWPORT_INQ_STATE 2-278
GMR_$VIEWPORT_INQ_STRUCTURE 2-279
GMR_$VIEWPORT_INQ_VISIBILITY 2-280
GMR_$VIEWPORT_MOVE 2-281
GMR_$VIEWPORT_REFRESH 2-282
GMR_$VIEWPORT_SET_BG_COLOR 2-283
GMR_$VIEWPORT_SET_BORDER 2-284
GMR_$VIEWPORT_SET_BOUNDS 2-285
GMR_$VIEWPORT_SET_CULLING 2-286
GMR_$VIEWPORT_SET_GLOBAL_MATRIX 2-287
GMR_$VIEWPORT_SET_HILIGHT_ABLOCK 2-288
GMR_$VIEWPORT_SET_INVIS_FILTER 2-289
GMR_$VIEWPORT_SET_PATH_ORDER 2-291
GMR_$VIEWPORT_SET_PICK 2-292
GMR_$VIEWPORT_SET_PICK_FILTER 2-294
GMR_$VIEWPORT_SET_REFRESH_STATE 2-296
GMR_$VIEWPORT_SET_STATE 2-298
GMR_$VIEWPORT_SET_STRUCTURE 2-299
GMR_$VIEWPORT_SET_VISIBILITY 2-300