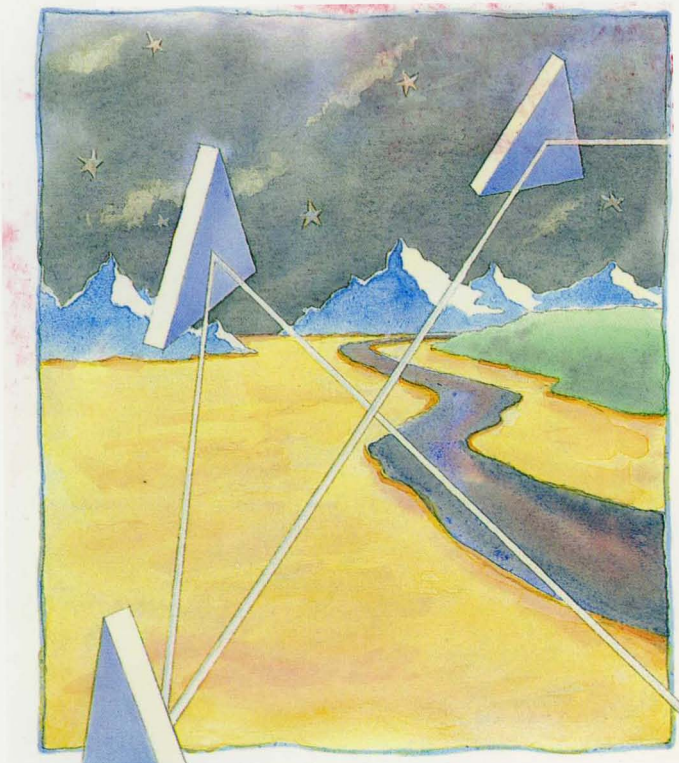

Domain Standard Graphics
Quick Reference: GPR and CTM

apollo

The Domain Standard Graphics Quick Reference: GPR and CTM

Order No. 010430-A00

First Printing: August, 1987
Last Printing: December, 1989

10 9 8 7 6 5 4 3 2 1

Preface

Domain Standard Graphics Quick Reference: GPR and CTM summarizes, in a highly compact form, all GPR and CTM information.

We've organized this manual as follows:

Chapter 1	Categorizes each GPR routine according to its purpose.
Chapter 2	Contains the syntax of every GPR routine.
Chapter 3	Describes each GPR data type.
Chapter 4	Provides the raster operations truth tables.
Chapter 5	Categorizes each CTM routine according to its purpose.
Chapter 6	Contains the syntax of every CTM routine.
Chapter 7	Describes each CTM data type.

Related Manuals

The file **/install/doc/apollo/os.v.***latest software release number***__manuals** lists current titles and revisions for all available manuals.

For example, at SR10.2 refer to **/install/doc/apollo/os.v.10.2__manuals** to check that you are using the correct version of manuals. You may also want to use this file to check that you have ordered all of the manuals that you need.

(If you are using the Aegis environment, you can access the same information through the Help system by typing **help manuals**.)

Refer to the *Domain Documentation Quick Reference* (002685) and the *Domain Documentation Master Index* (011242) for a complete list of related documents. For more information on GPR and CTM, refer to the following documents:

- *Domain Standard Graphics Call Reference: GPR and CTM* (007194) which details the syntax for every GPR call.

- *Programming with Domain Graphics Primitives* (005808) which demonstrates how to write GPR programs.

C and FORTRAN programmers should also see

- *Programming with General System Calls* (005506) which explains how to emulate certain Pascal data types in C or FORTRAN programs.

You can order Apollo documentation by calling **1-800-225-5290**. If you are calling from outside the U.S., you can dial **(508) 256-6600** and ask for **Apollo Direct Channel**.

Documentation Conventions

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

< >	Angle brackets enclose the name of a key on the keyboard.
status_t	Output parameters in Chapters 2 and 6 are in blue.
gpr_$mask_t	Boldfaced words indicate the names of routines and predefined data types in Pascal and C.
gpr_$frame	Italicized words in the description of enumerated types in Chapter 3 indicate legal enumerated values for the data type.
CTRL/Z	The notation CTRL/ followed by the name of a key indicates a control character sequence. You should hold down <CTRL> while typing the character.
	Change bars in the margin indicate technical changes from the last revision of this manual.

Contents

Chapter 1 GPR Calls Categorized by Purpose

Chapter 7 CTM Data Types

Tables

Chapter 1

GPR Calls Categorized by Purpose

The following is a list of GPR routines by grouped by function. Some routines are included in more than one category.

1.1 Initializing/Terminating the Graphics Package

gpr_$init — Initializes the graphics primitives package and allocates an initial bitmap.

gpr_$initialize — Initializes the graphics primitives package, allocates an initial bitmap, and sets the pixel format, projection format, and video format.

gpr_$terminate — Terminates the graphics primitives package.

gpr_$inq_disp_characteristics — Allows the application program to obtain a variety of information about the nature of the actual display device. You can call this routine before or after you call **gpr_$init**.

gpr_$inq_display_characteristics — Allows the application program to obtain a variety of information about the nature of the actual display device. You can call this routine before or after you call gpr_$initialize.

gpr_$inq_pixel_formats — Returns the pixel formats available on a device.

1.2 Inquiring

gpr_$inq_background — Returns the background color of the window.

gpr_$inq_bitmap — Returns the descriptor of the current bitmap.

gpr_$inq_bitmap_dimensions — Returns the size and number of planes of a bitmap.

gpr_$inq_bitmap_file_color_map — Returns the specified entries from the external bitmap color map.

gpr_$inq_bitmap_pixel_format — Returns the pixel format for the specified bitmap.

gpr_$inq_bitmap_pointer — Returns a pointer to bitmap storage in virtual address space. Also returns offset in memory from beginning of one scan line to the next.

gpr_$inq_bitmap_position — Returns the position of the upper left corner of the specified bitmap.

gpr_$inq_bitmap_proj_format — Returns the projection format for the specified bitmap.

gpr_$inq_bitmap_video_format — Returns the video format for the bitmap.

gpr_$inq_blank_timeout — Returns the time period before the screen is turned off.

gpr_$inq_bm_bit_offset — Returns the bit offset that corresponds to the left edge of a bitmap in virtual address space.

gpr_$inq_character16_width — Returns the width of the specified 16-bit character in the specified font.

gpr_$inq_character_width — Returns the width of the specified character in the specified font.

gpr_$inq_color_map — Returns the current color map values.

gpr_$inq_color_map_char — Returns the compatibility of color maps with the specified display mode.

gpr_$inq_config — Returns the current display configuration.

gpr_$inq_constraints — Returns the clipping window and plane mask used for the current bitmap.

gpr_$inq_coordinate_origin — Returns the x- and y-offsets added to all x- and y-coordinates used as input to move, draw, and BLT operations on the current bitmap.

gpr_$inq_cp — Returns the current position in the current bitmap.

gpr_$inq_curr_color_map — Returns the current color map ID.

gpr_$inq_cursor — Returns information about the cursor.

gpr_$inq_cursor_mode — Returns the current cursor mode.

gpr_$inq_disp_characteristics — Allows the application program to obtain a variety of information about the nature of the actual display device. You can call this routine before or after you call **gpr_$init**.

gpr_$inq_display_characteristics — Allows the application program to obtain a variety of information about the nature of the actual display device. You can call this routine before or after you call **gpr_$initialize**.

gpr_$inq_draw_pattern — Returns the pattern used in drawing all line and curve primitives.

gpr_$inq_draw_value — Returns the color used for drawing lines.

gpr_$inq_draw_width — Returns the line width in pixels for all line and curve primitives.

gpr_$inq_event_data — Returns the time that an event occurred, and, if it is a dial event, the dial number and dial value.

gpr_$inq_fill_background_value — Returns the color of the background used for tile fills.

gpr_$inq_fill_pattern — Returns the fill pattern for the current bitmap.

gpr_$inq_fill_value — Returns the color used to fill circles, rectangles, triangles, and trapezoids.

gpr_$inq_foreground — Returns the foreground color of the window.

gpr_$inq_horizontal_spacing — Returns the parameter for the width of spacing between displayed characters for the specified font.

gpr_$inq_imaging_format — Returns the current imaging format.

gpr_$inq_line_pattern — Returns the pattern used in drawing lines.

gpr_$inq_linestyle — Returns information about the current line style.

gpr_$inq_mult_constraints — Returns the dimensions and the number of the clipping windows for the current bitmap.

gpr_$inq_overlay_color_map — Returns the current overlay color map values.

gpr_$inq_pgon_decomp_technique — Returns the mode that controls the algorithm used to decompose and rasterize polygons.

gpr_$inq_pixel_formats — Returns the pixel formats available on a device.

gpr_$inq_plane_mask32 — Returns a 32-bit plane mask for the current bitmap.

gpr_$inq_raster_op_prim_set — Returns the primitive(s) that will be affected by the next **gpr_$set_raster_op** call, or the primitive(s) for which **gpr_$inq_raster_op** will return the current raster-op.

gpr_$inq_raster_ops — Returns the raster operation for the primitives (lines, fills, and bit-block transfers) specified with the **gpr_$raster_op_prim_set** routine.

gpr_$inq_refresh_entry — Returns a pointer to the procedure that refreshes the window, and a pointer to the procedure that refreshes hidden display memory.

gpr_$inq_space_size — Returns the width of the space to be displayed when a character requested is not in the specified font.

gpr_$inq_text — Returns the text font and text path used for the current bitmap.

gpr_$inq_text16_extent — Returns the extent of an array of 16-bit characters.

gpr_$inq_text16_offset — Returns the x- and y-offsets of an array of 16-bit characters.

gpr_$inq_text_extent — Returns the x- and y-offsets that a string spans when written by **gpr_$text**.

gpr_$inq_text_offset — Returns the x- and y-offsets from the top left pixel of a string to the origin of the string's first character. This routine also returns the x- or y-offset to the pixel that is the new current position after the text is written with **gpr_$text**.

gpr_$inq_text_path — Returns the direction for writing a line of text.

gpr_$inq_text_values — Returns the text color and the text background color used in the current bitmap.

gpr_$inq_triangle_fill_criteria — Returns the filling criteria used with polygons decomposed into triangles and rendered with **gpr_$render_exact**.

gpr_$inq_vis_list — Returns a list of the visible sections of an obscured window.

gpr_$inq_visible_buffer — Tells you whether it is the primary bitmap or the buffer bitmap that is currently being displayed.

gpr_$inq_window_id — Returns the character that identifies the current bitmap's window.

1.3 Controlling the Cursor

gpr_$inq_cursor — Returns information about the cursor.

gpr_$inq_cursor_mode — Returns the current cursor mode.

gpr_$set_cursor_active — Specifies whether the cursor is displayed.

gpr_$set_cursor_mode — Sets the cursor mode.

gpr_$set_cursor_origin — Defines one of the cursor's pixels as the cursor origin.

gpr_$set_cursor_pattern — Loads a cursor pattern.

gpr_$set_cursor_position — Establishes a position on the screen for display of the cursor.

1.4 Writing Text

gpr_$inq_character16_width — Returns the width of the specified 16-bit character in the specified font.

gpr_$inq_character_width — Returns the width of the specified character in the specified font.

gpr_$inq_horizontal_spacing — Returns the parameter for the width of spacing between displayed characters for the specified font.

gpr_$inq_space_size — Returns the width of the space to be displayed when a character requested is not in the specified font.

gpr_$inq_text — Returns the text font and text path used for the current bitmap.

gpr_$inq_text16_extent — Returns the x- and y-offsets of an array of 16-bit characters.

gpr_$inq_text16_offset — Returns the x- and y-offsets of an array of 16-bit characters.

gpr_$inq_text_extent — Returns the x- and y-offsets that a string spans when written by gpr_$text.

gpr_$inq_text_offset — Returns the x- and y-offsets from the top left pixel of a string to the origin of the string's first character. This routine also returns the x- or y-offset to the pixel that is the new current position after the text is written with **gpr_$text**.

gpr_$inq_text_path — Returns the direction for writing a line of text.

gpr_$inq_text_values — Returns the text color and the text background color used in the current bitmap.

gpr_$load_font_file — Loads a font from a file into the display's font storage area.

gpr_$replicate_font — Creates and loads a modifiable copy of a font.

gpr_$set_character16_width — Specifies the width of the specified 16-bit character in the specified modifiable font.

gpr_$set_character_width — Specifies the width of the specified character in the specified modifiable font.

gpr_$set_horizontal_spacing — Specifies the parameter for horizontal spacing of the specified font.

gpr_$set_space_size — Specifies the amount of horizontal space that GPR should leave blank when printing a character not defined in the current font.

gpr_$set_text_background_value — Specifies the color to use for text background.

gpr_$set_text_font — Establishes a new font for subsequent text operations.

gpr_$set_text_path — Specifies the direction for writing a line of text.

gpr_$set_text_value — Specifies the color to use for writing text.

gpr_$text — Writes text to the current bitmap, beginning at the current position.

gpr_$text16 — Writes text consisting of 16-bit characters to the current bitmap, beginning at the current position.

gpr_$unload_font_file — Unloads a font that has been loaded by **gpr_$load_font_file**.

1.5 Coordinate Positions

gpr_$inq_coordinate_origin — Returns the x- and y-offsets added to all x- and y-coordinates used as input to move, draw, and BLT operations on the current bitmap.

gpr_$inq_cp — Returns the current position in the current bitmap.

gpr_$move — Sets the current position to the given position.

gpr_$set_coordinate_origin — Establishes x- and y-offsets to add to all x- and y-coordinates used for move, draw, text, fill, and BLT operations on the current bitmap.

1.6 Controlling Draws

gpr_$inq_draw_pattern — Returns the pattern used in drawing all line and curve primitives.

gpr_$inq_draw_value — Returns the color used for drawing lines.

gpr_$inq_draw_width — Returns the line width in pixels for all line and curve primitives.

gpr_$inq_fill_pattern — Returns the fill pattern for the current bitmap.

gpr_$inq_line_pattern — Returns the pattern used in drawing lines.

gpr_$inq_linestyle — Returns information about the current line style.

gpr_$set_draw_pattern — Specifies the line pattern to use in drawing all line and curve primitives.

gpr_$set_draw_value — Specifies the color to use when drawing lines.

gpr_$set_draw_width — Sets the line width in pixels for line and curve primitives.

gpr_$set_line_pattern — Specifies the pattern to use in drawing lines.

gpr_$set_linestyle — Sets the line-style attribute of the current bitmap.

1.7 Drawing Lines, Arcs, Splines, and Circles

gpr_$arc_3p — Draws an arc from the current position through two other specified points.

gpr_$arc_c2p — Draws an arc from the current position to the point where the arc intersects a user-defined ray.

gpr_$circle — Draws an unfilled circle.

gpr_$draw_box — Draws an unfilled box based on the coordinates of two opposing corners.

gpr_$line — Draws a line from the current position to the end point supplied. The current position is updated to the end point.

gpr_$multiline — Draws a series of disconnected lines.

gpr_$polyline — Draws a series of connected lines.

gpr_$spline_cubic_p — Draws a parametric cubic spline through the control points.

gpr_$spline_cubic_x — Draws a cubic spline as a function of x through the control points.

gpr_$spline_cubic_y — Draws a cubic spline as a function of y through the control points.

1.8 Controlling Fills

gpr_$pgon_decomp_technique — Sets a mode that controls the algorithm used to decompose and render polygons.

gpr_$set_fill_background_value — Specifies the color to be used for drawing the background of tile fills.

gpr_$set_fill_pattern — Specifies the fill pattern used for the current bitmap.

gpr_$set_fill_value — Specifies the color to use to fill circles, rectangles, triangles, and trapezoids.

gpr_$set_triangle_fill_criteria — Sets the filling criteria used with polygons that are decomposed into triangles before being rendered or polygons that are rendered directly (decomposition technique set to render exact).

1.9 Filling Regions

gpr_$circle_filled — Draws and fills a circle.

gpr_$close_fill_pgon — Closes and fills the currently open polygon.

gpr_$close_return_pgon — Closes the currently open polygon and returns the list of trapezoids within its interior.

gpr_$close_return_pgon_tri — Closes the currently open polygon and returns a list of triangles within its interior.

gpr_$multitrapezoid — Draws and fills a list of trapezoids in the current bitmap.

gpr_$multitriangle — Draws and fills a list of triangles in the current bitmap.

gpr_$pgon_polyline — Defines a series of line segments forming part of a polygon boundary.

gpr_$rectangle — Draws and fills a rectangle.

gpr_$start_pgon — Defines the starting position of a polygon.

gpr_$trapezoid — Draws and fills a trapezoid.

gpr_$triangle — Draws and fills a triangle.

1.10 Creating/Accessing Bitmaps

gpr_$allocate_attribute_block — Allocates a data structure that contains a set of default bitmap attribute settings, and returns the descriptor for the data structure.

gpr_$allocate_bitmap — Allocates a bitmap in main memory and returns a bitmap descriptor.

gpr_$allocate_bitmap_nc — Allocates a bitmap in main memory without setting all the pixels in the bitmap to 0, and returns a bitmap descriptor.

gpr_$allocate_hdm_bitmap — Allocates a bitmap in hidden display memory.

gpr_$allocate_projection — Allocates a new projection for an existing bitmap.

gpr_$attribute_block — Returns the descriptor of the attribute block associated with the given bitmap.

gpr_$deallocate_attribute_block — Deallocates an attribute block allocated by gpr_$allocate_attribute_block.

gpr_$deallocate_bitmap — Deallocates an allocated bitmap.

gpr_$enable_direct_access — Ensures completion of display hardware operations before the program uses the pointer to access display memory.

gpr_$init — Initializes the graphics primitives package and allocates an initial bitmap.

gpr_$initialize — Initializes the graphics primitives package, allocates an initial bitmap, and sets the pixel format, projection format and video format.

gpr_$inq_bitmap — Returns the descriptor of the current bitmap.

gpr_$inq_bitmap_dimensions — Returns the size and number of planes of a bitmap.

gpr_$inq_bitmap_pointer — Returns a pointer to bitmap storage in virtual address space. Also returns offset in memory from beginning of one scan line to the next.

gpr_$inq_bitmap_position — Returns the position of the upper left corner of the specified bitmap. This is normally the screen position; although, it does have some significance for main memory bitmaps.

gpr_$inq_bm_bit_offset — Returns the bit offset that corresponds to the left edge of a bitmap in virtual address space.

gpr_$make_bitmap_from_array — Creates a bitmap descriptor pointing to a given memory address (containing the image data).

gpr_$open_bitmap_file — Opens a bitmap stored on disk for creating or accessing.

gpr_$remap_color_memory — Defines the plane in color display memory for which a pointer will be returned when gpr_$inq_bitmap_pointer is used. This allows a single plane of color display memory to be accessed directly.

gpr_$remap_color_memory_1 — Defines the plane in hidden color display memory for which a pointer is returned when gpr_$inq_bitmap_pointer is used.

gpr_$remap_pixels — Remaps the display bitmap to pixel mode.

gpr_$select_color_frame — Selects whether frame 0 or frame 1 of color display memory is visible.

gpr_$set_attribute_block — Associates an attribute block with the current bitmap.

gpr_$set_bitmap — Establishes a bitmap as the current bitmap for subsequent operations.

gpr_$set_bitmap_dimensions — Modifies the size and the number of planes of a bitmap.

1.11 Block Transfers (BLTs)

gpr_$additive_blt — Transfers a single plane of any bitmap to all active planes of the current bitmap.

gpr_$bit_blt — Performs a bit-block transfer from a single plane of any bitmap to a single plane of the current bitmap.

gpr_$pixel_blt — Performs a pixel-block transfer from any bitmap to the current bitmap.

1.12 Double Buffering

gpr_$allocate_buffer — Allocates a buffer bitmap in display memory, having the same size and attributes as a specified display bitmap.

gpr_$deallocate_buffer — Deallocates a buffer bitmap.

gpr_$inq_visible_buffer — Tells you whether it is the primary bitmap or the buffer bitmap that is currently being displayed.

gpr_$select_display_buffer — Switches the buffers in a double buffering program so that the displayed buffer becomes invisible and the invisible buffer becomes displayed.

gpr_$set_bitmap — Establishes a bitmap as the current bitmap for subsequent operations.

1.13 Clearing Pixel Values

gpr_$clear — Sets all pixels in the current bitmap to the given color.

1.14 Converting Arrays to Bitmaps and Vice Versa

gpr_$make_bitmap_from_array — Creates a bitmap descriptor pointing to a given memory address (containing the image data).

gpr_$read_pixels — Reads the pixel values from a window of the current bitmap and stores the values in a pixel array.

gpr_$write_pixels — Writes the pixel values from a pixel array into a window of the current bitmap.

1.15 Windows

gpr_$acquire_display — Establishes exclusive access to the display hardware.

gpr_$force_release — Releases the display regardless of how many times it has previously been acquired.

gpr_$inq_blank_timeout — Returns the time period before the screen is turned off.

gpr_$inq_constraints — Returns the clipping window and plane mask used for the current bitmap.

gpr_$inq_mult_constraints — Returns the dimensions and the number of the clipping windows for the current bitmap.

gpr_$inq_refresh_entry — Returns a pointer to the procedure that refreshes the window, and a pointer to the procedure that refreshes hidden display memory.

gpr_$inq_vis_list — Returns a list of the visible sections of an obscured window.

gpr_$release_display — Decrements a counter associated with the number of times a display has been acquired.

gpr_$set_acq_time_out — Establishes the length of time the display will be acquired.

gpr_$set_auto_refresh — Directs the Display Manager to refresh the window automatically.

gpr_$set _blank_timeout — Establishes the time period that the system waits before it shuts off the screen.

gpr_$set_clip_window — Changes the clipping window for the current bitmap.

gpr_$set_clipping_active — Enables/disables a clipping window for the current bitmap.

gpr_$set_mult_clip_window — Sets multiple clipping windows for the current attribute block and bitmap.

gpr_$set_refresh_entry — Specifies the entry points of application-supplied procedures that refresh the displayed image in a direct window and hidden display memory.

gpr_$set_obscured_opt — Establishes the action to be taken when a window to be acquired is obscured.

1.16 Input Events

gpr_$cond_event_wait — Returns information about the occurrence of any event without entering a wait state.

gpr_$disable_input — Disables a previously enabled event type.

gpr_$enable_input — Enables an event type and a selected set of keys.

gpr_$event_wait — Waits for an event.

gpr_$get_ec — Returns the event count associated with a graphic event.

gpr_$inq_event_data — Returns the time that an event occurred, and, if it is a dial event, the dial number and dial value.

gpr_$inq_window_id — Returns the character that identifies the current bitmap's window.

gpr_$light_pfk_buttons — Turns the specified LPFK buttons on or off.

gpr_$set_icon_opt — Allows applications to continue program execution when the window is iconized.

gpr_$set_input_sid — Specifies the input pad from which graphics input is to be taken.

gpr_$set_quit_event — Defines the quit character event.

gpr_$set_window_id — Establishes the character that identifies the current bitmap's window.

1.17 Raster Operations and Plane Masks

gpr_$inq_constraints — Returns the clipping window and plane mask used for the current bitmap.

gpr_$inq_plane_mask32 — Returns a 32-bit plane mask for the current bitmap.

gpr_$inq_raster_op_prim_set — Returns the primitive(s) that will be affected by the next **gpr_$set_raster_op** call, or the primitive(s) for which **gpr_$inq_raster_op** will return the current raster operation.

gpr_$inq_raster_ops — Returns the raster operation for the primitives (lines, fills, and bit-block transfers) specified with the gpr_$raster_op_prim_set routine.

gpr_$raster_op_prim_set — Specifies the primitive(s) that will be affected by the next **gpr_$set_raster_op** call, or the primitive(s) for which gpr_$inq_raster_op will return the current raster operation.

gpr_$set_plane_mask — Establishes a 16-bit plane mask for subsequent write operations.

gpr_$set_plane_mask_32 — Establishes a 32-bit plane mask for subsequent write operations.

gpr_$set_raster_op — Specifies a raster operation for the primitives established with the **gpr_$raster_op_prim_set** routine.

gpr_$set_raster_op_mask — Sets raster operations on several planes.

1.18 Controlling Color

gpr_$inq_background — Returns the background color of the window.

gpr_$inq_bitmap_file_color_map — Returns the specified entries from the external-bitmap color map.

gpr_$inq_color_map — Returns the current color map values.

gpr_$inq_color_map_char — Returns the compatibility of color maps with the specified display mode.

gpr_$inq_curr_color_map — Returns the current color map ID.

gpr_$inq_overlay_color_map — Returns the current overlay color map values.

gpr_$inq_draw_value — Returns the color used for drawing lines.

gpr_$inq_fill_background_value — Returns the tile fill background color.

gpr_$inq_fill_value — Returns the color used to fill circles, rectangles, triangles, and trapezoids.

gpr_$inq_foreground — Returns the foreground color of the window.

gpr_$inq_text_values — Returns the text font and text path used for the current bitmap.

gpr_$set_bitmap_file_color_map — Establishes new values for the external-bitmap color map.

gpr_$set_color_map — Establishes new values for the color map.

gpr_$set_curr_color_map — Sets the current color map.

gpr_$set_overlay_color_map — Establishes new values for the overlay color map.

gpr_$set_draw_value — Specifies the color to use for drawing lines.

gpr_$set_fill_background_value — Specifies the color to use for drawing the background of tile fills.

gpr_$set_fill_value — Specifies the color to use for filling circles, rectangles, triangles, and trapezoids.

gpr_$set_text_value — Specifies the color to use for writing text.

gpr_$wait_frame — Waits for the current frame refresh cycle to end before executing operations that modify the display.

1.19 Pixel, Projection, and Video Formats

gpr_$allocate_projection — Allocates a new projection for an existing bitmap.

gpr_$initialize — Initializes the graphics primitives package, allocates an initial bitmap, and sets the pixel format, projection format and video format.

gpr_$inq_bitmap_pixel_format — Returns the pixel format for the specified bitmap.

gpr_$inq_bitmap_proj_format — Returns the projection format for the specified bitmap.

gpr_$inq_bitmap_video_format — Returns the video format for the bitmap.

gpr_$inq_display_characteristics — Allows the application program to obtain a variety of information about the nature of the actual display device. You can call this routine before or after you call **gpr_$initialize**.

gpr_$inq_pixel_formats — Returns the pixel formats available on a device.

gpr_$set_pixel_format — Changes the pixel format.

1.20 Imaging Mode

gpr_$inq_imaging_format — Returns the current imaging format.

gpr_$set_imaging_format — Sets the imaging format of the color display.

1.21 Zooming

gpr_$color_zoom — Sets the magnification scale factor for a color display.

Chapter 2

GPR Calls and Their Parameters

This chapter describes the required arguments for each GPR call. For instance, consider the description of the **gpr_$allocate_attribute_block** call. The listing shows that this call takes two arguments (attrib and status). The first argument takes the gpr_$attribute_desc_t data type, and the second argument takes the status_$t data type. (See Chapter 3 for an explanation of each data type.)

Most GPR calls are Pascal procedures (which corresponds to a function returning void in C and a subroutine in FORTRAN). Six calls (**gpr_$acquire_display**, **gpr_$attribute_block**, **gpr_$event_wait**, **gpr_$cond_event_wait**, **gpr_$inq_background**, and **gpr_$inq_foreground**) are Pascal functions (which corresponds to a function returning a nonvoid value in C and a function in FORTRAN).

The calls are listed in alphabetical order. Parameters shown in black are input parameters, and parameters shown in blue are output parameters returned by GPR.

unobscured = gpr_$acquire_display(status)

unobscured	boolean
status	status_$t

gpr_$additive_blt(source_bitmap,source_window, source_plane, destination_origin, status)

source_bitmap	gpr_$bitmap_desc_t
source_window	gpr_$window_t
source_plane	gpr_$rgb_plane_t
destination_origin	gpr_$position_t
status	status_$t

gpr_$allocate_attribute_block(attrib, status)

attrib	gpr_$attribute_desc_t
status	status_$t

gpr_$allocate_bitmap(size, hi_plane, attr, bitmap, status)

size	gpr_$offset_t
hi_plane	gpr_$rgb_plane_t
attr	gpr_$attribute_desc_t
bitmap	gpr_$bitmap_desc_t
status	status_$t

gpr_$allocate_bitmap_nc(size, hi_plane, attr, bitmap, status)

size	gpr_$offset_t
hi_plane	gpr_$rgb_plane_t
attr	gpr_$attribute_desc_t
bitmap	gpr_$bitmap_desc_t
status	status_$t

gpr_$allocate_buffer(primary_bitmap, buffer_bitmap, status)

primary_bitmap	gpr_$bitmap_desc_t
buffer_bitmap	gpr_$bitmap_desc_t
status	status_$t

gpr_$allocate_hdm_bitmap(size, hi_plane, attr, bitmap, status)

size	gpr_$offset_t
hi_plane	gpr_$rgb_plane_t
attr	gpr_$attribute_desc_t
bitmap	gpr_$bitmap_desc_t
status	status_$t

gpr_$allocate_projection(main_bitmap,options,proj_form,proj_bitmap, status)

main_bitmap	gpr_$bitmap_desc_t
options	gpr_$init_options_set_t
proj_form	UNIV gpr_$proj_format_t
proj_bitmap	gpr_$bitmap_desc_t
status	status_$t

gpr_$arc_c2p(center, p2, direction, option, status)

center	gpr_$position_t
p2	gpr_$position_t
direction	gpr_$arc_direction_t
option	gpr_$arc_option_t
status	status_$t

gpr_$arc_3p(p2, p3, status)

p2	gpr_$position_t
p3	gpr_$position_t
status	status_$t

attr_desc = gpr_$attribute_block(bitmap, status)

attr_desc	gpr_$attribute_desc_t
bitmap	gpr_$bitmap_desc_t
status	status_$t

gpr_$bit_blt(source_bitmap, source_window, source_plane, destination_origin, destination_plane, status)

source_bitmap	gpr_$bitmap_desc_t
source_window	gpr_$window_t
source_plane	gpr_$rgb_plane_t
destination_origin	gpr_$position_t
destination_plane	gpr_$rgb_plane_t
status	status_$t

gpr_$circle(center, radius, status)

center	gpr_$position_t
radius	2-byte integer
status	status_$t

gpr_$circle_filled(center, radius, status)

center	gpr_$position_t
radius	2-byte integer
status	status_$t

gpr_$clear(color_value, status)

color_value	gpr_$pixel_value_t
status	status_$t

gpr_$close_fill_pgon(status)

status	status_$t

gpr_$close_return_pgon(list_size, trap_list, n_traps, status)

list_size	2-byte integer
trap_list	gpr_$trap_list_t
n_traps	2-byte integer
status	status_$t

gpr_$close_return_pgon_tri(list_size, triangles_list, n_triangles, status)

list_size	2-byte integer
triangles_list	gpr_$triangle_list_t
n_triangles	2-byte integer
status	status_$t

gpr_$color_zoom(x, y, status)

x	2-byte integer
y	2-byte integer
status	status_$t

unobscured = gpr_$cond_event_wait(event_type, event_data, position, status)

unobscured	boolean
event_type	gpr_$event_t
event_data	char
position	gpr_$position_t
status	status_$t

gpr_$deallocate_attribute_block(attrib, status)

attrib	gpr_$attribute_desc_t
status	status_$t

gpr_$deallocate_bitmap(bitmap, status)

bitmap	gpr_$bitmap_desc_t
status	status_$t

gpr_$deallocate_buffer(primary_bitmap, buffer_bitmap, status)

primary_bitmap	gpr_$bitmap_desc_t
buffer_bitmap	gpr_$bitmap_desc_t
status	status_$t

gpr_$disable_input(event_type, status)

event_type	gpr_$event_t
status	status_$t

gpr_$draw_box(left_edge, top_edge, right_edge, bottom_edge, status)

left_edge	gpr_$coordinate_t
top_edge	gpr_$coordinate_t
right_edge	gpr_$coordinate_t
bottom_edge	gpr_$coordinate_t
status	status_$t

gpr_$enable_direct_access(status)

status	status_$t

gpr_$enable_input(event_type, key_set, status)

event_type	gpr_$event_t
key_set	gpr_$keyset_t
status	status_$t

unobscured = gpr_$event_wait(event_type, event_data, position, status)

unobscured	boolean
event_type	gpr_$event_t
event_data	char
position	gpr_$position_t
status	status_$t

gpr_$force_release(acq_release_count, status)

acq_release_count	2-byte integer
status	status_$t

gpr_$get_ec(gpr_key, ec_ptr, status)

gpr_key	gpr_$ec_key_t
ec_ptr	ec2_$ptr_t
status	status_$t

gpr_$init(op, unit_or_pad, size, hi_plane, init_bitmap, status)

op	gpr_$display_mode_t
unit_or_pad	stream_$id_t
size	gpr_$offset_t
hi_plane	gpr_$rgb_plane_t
init_bitmap	gpr_$bitmap_desc_t
status	status_$t

gpr_$initialize (resource_type,resource_id,options,size, pix_form, proj_form,vid_form,init_bitmap,status)

resource_type	gpr_$resource_type_t
resource_id	4-byte integer
options	gpr_$init_options_set_t
size	gpr_$offset_t
pix_form	UNIV gpr_$pixel_format_t
proj_form	UNIV gpr_$proj_format_t
vid_form	UNIV gpr_$video_format_t
init_bitmap	gpr_$bitmap_desc_t
status	status_$t

pix_value = gpr_$inq_background (status)

pix_value	gpr_$pixel_value_t
status	status_$t

gpr_$inq_bitmap(bitmap, status)

bitmap	gpr_$bitmap_desc_t
status	status_$t

gpr_$inq_bitmap_dimensions(bitmap, size, hi_plane, status)

bitmap	gpr_$bitmap_desc_t
size	gpr_$offset_t
hi_plane	gpr_$rgb_plane_t
status	status_$t

gpr_$inq_bitmap_file_color_map(bitmap, start, entries, color, status)

bitmap	gpr_$bitmap_desc_t
start	2-byte integer
entries	2-byte integer
color	gpr_$color_vector_t
status	status_$t

gpr_$inq_bitmap_pixel_format (bitmap,pixform,status)

bitmap	gpr_$bitmap_desc_t
pixform	UNIV gpr_$pixel_format_t
status	status_$t

gpr_$inq_bitmap_pointer(bitmap, storage_ptr, line_width, status)

bitmap	gpr_$bitmap_desc_t
storage_ptr	pointer to a 4-byte integer
line_width	2-byte integer
status	status_$t

gpr_$inq_bitmap_position(bitmap, origin, status)

bitmap	gpr_$bitmap_desc_t
origin	gpr_$position_t
status	status_$t

gpr_$inq_bitmap_proj_format (bitmap,projform,status)

bitmap	gpr_$bitmap_desc_t
projform	UNIV gpr_$proj_format_t
status	status_$t

gpr_$inq_bitmap_video_format (bitmap,videoform,status)

bitmap	gpr_$bitmap_desc_t
videoform	UNIV gpr_$video_format_t
status	status_$t

gpr_$inq_blank_timeout (timeout,status)

timeout	time_$clock_t
status	status_$t

gpr_$inq_bm_bit_offset(bitmap, bit_offset, status)

bitmap	gpr_$bitmap_desc_t
bit_offset	2-byte integer
status	status_$t

gpr_$inq_character16_width(font_id, character, width, status)

font_id	2-byte integer
character	unsigned 2-byte integer
width	2-byte integer
status	status_$t

gpr_$inq_character_width(font_id, character, width, status)

font_id	2-byte integer
character	char
width	2-byte integer
status	status_$t

gpr_$inq_color_map (start_index, n_entries, color_map, status)

start_index	gpr_$pixel_value_t
n_entries	2-byte integer
color_map	gpr_$color_vector_t
status	status_$t

gpr_$inq_color_map_char (disp_mode,options,display_unit, color_map_char_len,color_map_char, color_map_char_len_returned,status)

disp_mode	gpr_$display_mode_t
options	4-byte integer
display_unit	2-byte integer
color_map_char_len	2-byte integer
color_map_char	UNIV gpr_$color_map_char_t
color_map_char_len_returned	integer
status	status_$t

gpr_$inq_config (config, status)

config	gpr_$display_config_t
status	status_$t

gpr_$inq_constraints(window, active, mask, status)

window	gpr_$window_t
active	boolean
mask	gpr_$mask_t
status	status_$t

gpr_$inq_coordinate_origin(origin, status)

origin	gpr_$position_t
status	status_$t

gpr_$inq_cp(x, y, status)

x, y	gpr_$coordinate_t
status	status_$t

gpr_$inq_curr_color_map(color_map_id,options, display_unit,status)

color_map_id	gpr_$color_map_id_t
options	4-byte integer
display_unit	2-byte integer
status	status_$t

gpr_$inq_cursor(cursor_pattern, cursor_rops, cursor_active, cursor_position, cursor_origin, status)

cursor_pattern	gpr_$bitmap_desc_t
cursor_rops	gpr_$raster_op_array_t
cursor_active	boolean
cursor_position	gpr_$position_t
cursor_origin	gpr_$position_t
status	status_$t

gpr_$inq_cursor_mode (mode,status)

mode	gpr_$cursor_mode_t
status	status_$t

gpr_$inq_disp_characteristics(op_mode,unit_or_pad,disp_len,disp, disp_len_returned, status)

op_mode	gpr_$display_mode_t
unit_or_pad	stream_$id_t
disp_len	2-byte integer
disp	gpr_$disp_char_t
disp_len_returned	2-byte integer
status	status_$t

gpr_$inq_display_characteristics(resource_type,resource_id,disp_len, disp, disp_len_returned, status)

resource_type	gpr_$resource_type_t
resource_id	4-byte integer
disp_len	2-byte integer
disp	gpr_$disp_char_t
disp_len_returned	2-byte integer
status	status_$t

gpr_$inq_draw_pattern(repeat_count, pattern, length, status)

repeat_count	2-byte integer
pattern	gpr_$line_pattern_t
length	2-byte integer
status	status_$t

gpr_$inq_draw_value(color_value, status)

color_value	gpr_$pixel_value_t
status	status_$t

gpr_$inq_draw_width(width, status)

width	2-byte integer
status	status_$t

gpr_$inq_event_data(event_type, length, event_data, length_ret, time_stamp, status)

event_type	gpr_$event_t
length	2-byte integer
event_data	UNIV gpr_$event_data_t
length_ret	2-byte integer
time_stamp	time_$clock_t
status	status_$t

gpr_$inq_fill_background_value(color_value, status)

color_value	gpr_$pixel_value_t
status	status_$t

gpr_$inq_fill_pattern(pattern, scale, status)

pattern	gpr_$bitmap_desc_t
scale	2-byte integer
status	status_$t

gpr_$inq_fill_value(color_value, status)

color_value	gpr_$pixel_value_t
status	status_$t

pix_value = gpr_$inq_foreground (status)

pix_value	gpr_$pixel_value_t
status	status_$t

gpr_$inq_horizontal_spacing(font_id, horizontal_spacing, status)

font_id	2-byte integer
horizontal_spacing	2-byte integer
status	status_$t

gpr_inq_imaging_format(format, status)

format	gpr_$imaging_format_t
status	status_$t

gpr_$inq_line_pattern(repeat_count, pattern, length, status)

repeat_count	2-byte integer
pattern	gpr_$line_pattern_t
length	2-byte integer
status	status_$t

gpr_$inq_linestyle(style, scale, status)

style	gpr_$linestyle_t
scale	2-byte integer
status	status_$t

gpr_$inq_mult_constraints(windows,n_windows, active, status)

windows	UNIV gpr_$window_list_t
n_windows	2-byte integer
active	boolean
status	status_$t

gpr_$inq_overlay_color_map (start_index,n_entries, color_map, status)

start_index	gpr_$pixel_value_t
n_entries	2-byte integer
color_map	gpr_$color_vector_t
status	status_$t

gpr_$inq_pgon_decomp_technique(technique, status)

technique	gpr_$decomp_technique_t
status	status_$t

gpr_$inq_pixel_formats(resource_type,resource_id, max_formats,format_size,numformats,formats,status)

resource_type	gpr_$resource_type_t
resource_id	4-byte integer
max_formats	4-byte integer
format_size	4-byte integer
numformats	4-byte integer
formats	gpr_$pixel_format_array_t
status	status_$t

gpr_$inq_plane_mask_32 (mask,status)

mask	gpr_$mask_32_t
status	status_$t

gpr_$inq_raster_op_prim_set(prim_set, status)

prim_set	gpr_$rop_prim_set_t
status	status_$t

gpr_$inq_raster_ops(ops, status)

ops	gpr_$raster_op_array_t
status	status_$t

gpr_$inq_refresh_entry(ptr_to_window_refresh_proc, ptr_to_hidden_memory_refresh_proc, status)

ptr_to_window_refresh_proc	gpr_$rwin_pr_t
ptr_to_hidden_memory_refresh_proc	gpr_$rhdm_pr_t
status	status_$t

gpr_$inq_space_size(font_id, space_size, status)

font_id	2-byte integer
space_size	2-byte integer
status	status_$t

gpr_$inq_text(font_id, direction, status)

font_id	2-byte integer
direction	gpr_$direction_t
status	status_$t

gpr_$inq_text16_extent(t_array,t_arrayl, size, status)

t_array	UNIV gpr_$16bit_character_array_t
t_arrayl	2-byte integer
size	gpr_$offset_t
status	status_$t

gpr_$inq_text_extent(string, string_length, size, status)

string	gpr_$string_t
string_length	2-byte integer
size	gpr_$offset_t
status	status_$t

gpr_$inq_text16_offset(t_array,t_arrayl, start, xy_end, status)

t_array	UNIV gpr_$16bit_character_array_t
t_arrayl	2-byte integer
start	gpr_$offset_t
xy_end	2-byte integer
status	status_$t

gpr_$inq_text_offset(string, string_length, start, xy_end, status)

string	gpr_$string_t
string_length	2-byte integer
start	gpr_$offset_t
xy_end	2-byte integer
status	status_$t

gpr_$inq_text_path(direction, status)

direction	gpr_$direction_t
status	status_$t

gpr_$inq_text_values(text_color_value, text_background, status)

text_color_value	gpr_$pixel_value_t
text_background	gpr_$pixel_value_t
status	status_$t

gpr_$inq_triangle_fill_criteria(fill_criteria, status)

fill_criteria	gpr_$triangle_fill_criteria_t
status	status_$t

gpr_$inq_vis_list(slots_available, slots_total, list_of_vis_windows, status)

slots_available	2-byte integer
slots_total	2-byte integer
list_of_vis_windows	gpr_$window_list_t
status	status_$t

gpr_$inq_visible_buffer(bitmap, status)

bitmap	gpr_$bitmap_desc_t
status	status_$t

gpr_$inq_window_id(id_char, status)

id_char	char
status	status_$t

gpr_$light_pfk_buttons (light_set ,on_off,status)

light_set	gpr_$keyset_t
on_off	Boolean
status	status_$t

gpr_$line(x_coord, y_coord, status)

x_coord	gpr_$coordinate_t
y_coord	gpr_$coordinate_t
status	status_$t

gpr_$load_font_file(pathname_of_font, pathname_length,font_id,status)

pathname_of_font	name_$pname_t
pathname_length	2-byte integer
font_id	2-byte integer
status	status_$t

gpr_$make_bitmap_from_array(size, attribs, groups, g_headers, bitmap, status)

size	gpr_$offset_t
attribs	gpr_$attribute_desc_t
groups	integer
g_headers	gpr_$bmf_group_header_array_t (input and output)
bitmap	gpr_$bitmap_desc_t
status	status_$t

gpr_$move(x_coord, y_coord, status)

x_coord	gpr_$coordinate_t
y_coord	gpr_$coordinate_t
status	status_$t

gpr_$multiline(array_of_x_coords,array_of_y_coords, n_points, status)

array_of_x_coords	gpr_$coordinate_array_t
array_of_y_coords	gpr_$coordinate_array_t
n_points	2-byte integer
status	status_$t

gpr_$multitrapezoid(trapezoid_list, n_trapezoids, status)

trapezoid_list	gpr_$trap_list_t
n_trapezoids	2-byte integer
status	status_$t

gpr_$multitriangle(triangle_list, n_triangles, status)

triangle_list	gpr_$triangle_list_t
n_triangles	2-byte integer
status	status_$t

gpr_$open_bitmap_file(access, filename, filename_size, version, size, groups, g_headers, attribs, bitmap, created, status)

access	gpr_$access_mode_t
filename	name_$pname_t
filename_size	2-byte integer
version	gpr_$version_t (input and output)
size	gpr_$offset_t (input and output)
groups	2-byte integer (input and output)
g_headers	gpr_$bmf_group_header_array_t (input and output)
attribs	gpr_$attribute_desc_t
bitmap	gpr_$bitmap_desc_t
created	boolean
status	status_$t

gpr_$pgon_decomp_technique(technique, status)

technique	gpr_$decomp_technique_t
status	status_$t

gpr_$pgon_polyline(array_of_x_coords, array_of_y_coords, n_points, status)

array_of_x_coords	gpr_$coordinate_array_t
array_of_y_coords	gpr_$coordinate_array_t
n_points	2-byte integer
status	status_$t

gpr_$pixel_blt(source_bitmap,source_window, destination_origin, status)

source_bitmap	gpr_$bitmap_desc_t
source_window	gpr_$window_t
destination_origin	gpr_$position_t
status	status_$t

gpr_$polyline(array_of_x_coords, array_of_y_coords, n_points, status)

array_of_x_coords	gpr_$coordinate_array_t
array_of_y_coords	gpr_$coordinate_array_t
n_points	2-byte integer
status	status_$t

gpr_$raster_op_prim_set(prim_set, status)

prim_set	gpr_$rop_prim_set_t
status	status_$t

gpr_$read_pixels(source_window, pixel_array, status)

source_window	gpr_$window_t
pixel_array	gpr_$pixel_array_t
status	status_$t

gpr_$rectangle(rectangle, status)

rectangle	gpr_$window_t
status	status_$t

gpr_$release_display(status)

status	status_$t

gpr_$remap_color_memory(plane, status)

plane	gpr_$rgb_plane_t
status	status_$t

gpr_$remap_color_memory_1(plane, status)

plane	gpr_$rgb_plane_t
status	status_$t

gpr_$remap_pixels (section_number,group_header,status)

section_number	2-byte integer
group_header	gpr_$bmf_group_header_t (input and output)
status	status_$t

gpr_$replicate_font(font_id, repl_font_id, status)

font_id	2-byte integer
repl_font_id	2-byte integer
status	status_$t

gpr_$select_color_frame(frame, status)

frame	2-byte integer
status	status_$t

gpr_$select_display_buffer(display_desc,option_desc,option_value, options, status)

display_desc	gpr_$bitmap_desc_t
option_desc	gpr_$bitmap_desc_t
option_value	4-byte integer
options	gpr_$double_buffer_option_t
status	status_$t

gpr_$set_acq_time_out(time_out, status)

time_out	time_$clock_t
status	status_$t

gpr_$set_attribute_block(attrib, status)

attrib	gpr_$attribute_desc_t
status	status_$t

gpr_$set_auto_refresh(auto_refresh, status)

auto_refresh	boolean
status	status_$t

gpr_$set_bitmap(bitmap, status)

bitmap	gpr_$bitmap_desc_t
status	status_$t

gpr_$set_bitmap_dimensions(bitmap, size, hi_plane, status)

bitmap	gpr_$bitmap_desc_t
size	gpr_$offset_t
hi_plane	gpr_$rgb_plane_t
status	status_$t

gpr_$set_bitmap_file_color_map(bitmap, start, entries, color, status)

bitmap	gpr_$bitmap_desc_t
start	2-byte integer
entries	2-byte integer
color	gpr_$color_vector_t
status	status_$t

gpr_$set_blank_timeout(timeout, status)

timeout	time_$clock_t
status	status_$t

gpr_$set_character16_width(font_id, character, width, status)

font_id	2-byte integer
character	unsigned 2-byte integer
width	2-byte integer
status	status_$t

gpr_$set_character_width(font_id, character, width, status)

font_id	2-byte integer
character	char
width	2-byte integer
status	status_$t

gpr_$set_clip_window(window, status)

window	gpr_$window_t
status	status_$t

gpr_$set_clipping_active(active, status)

active	boolean
status	status_$t

gpr_$set_color_map(start_index, n_entries, color_array, status)

start_index	gpr_$pixel_value_t
n_entries	2-byte integer
color_array	gpr_$color_vector_t
status	status_$t

gpr_$set_coordinate_origin(origin, status)

origin	gpr_$position_t
status	status_$t

gpr_$set_curr_color_map(color_map_id,options, display_unit,status)

color_map_id	gpr_$color_map_id_t
options	4-byte integer
display_unit	2-byte integer
status	status_$t

gpr_$set_cursor_active(active, status)

active	boolean
status	status_$t

gpr_$set_cursor_mode (mode,status)

mode	gpr_$cursor_mode_t
status	status_$t

gpr_$set_cursor_origin(origin, status)

origin	gpr_$position_t
status	status_$t

gpr_$set_cursor_pattern(cursor, status)

cursor	gpr_$bitmap_desc_t
status	status_$t

gpr_$set_cursor_position(pos, status)

pos	gpr_$position_t
status	status_$t

gpr_$set_draw_pattern(repeat_count, pattern, length, status)

repeat_count	2-byte integer
pattern	gpr_$line_pattern_t
length	2-byte integer
status	status_$t

gpr_$set_draw_value(color_value, status)

color_value	gpr_$pixel_value_t
status	status_$t

gpr_$set_draw_width(width, status)

width	2-byte integer
status	status_$t

gpr_$set_fill_background_value(color_value, status)

color_value	gpr_$pixel_value_t
status	status_$t

gpr_$set_fill_pattern(pattern, scale, status)

pattern	gpr_$bitmap_desc_t
scale	2-byte integer
status	status_$t

gpr_$set_fill_value(color_value, status)

color_value	gpr_$pixel_value_t
status	status_$t

gpr_$set_horizontal_spacing(font_id, horizontal_spacing, status)

font_id	2-byte integer
horizontal_spacing	2-byte integer
status	status_$t

gpr_$set_icon_opt (icon_opt,status)

icon_opt	gpr_$icon_opt_t
status	status_$t

gpr_$set_imaging_format(format, status)

format	gpr_$imaging_format_t
status	status_$t

gpr_$set_input_sid(stream_id, status)

stream_id	stream_$id_t
status	status_$t

gpr_$set_line_pattern(repeat_count, pattern, length, status)

repeat_count	2-byte integer
pattern	gpr_$line_pattern_t
length	2-byte integer
status	status_$t

gpr_$set_linestyle(style, scale, status)

style	gpr_$linestyle_t
scale	2-byte integer
status	status_$t

gpr_$set_mult_clip_window (clip_list,n_clip_list, n_multiple_clips_used,status)

clip_list	gpr_$window_list_t
n_clip_list	2-byte integer
n_multiple_clips_used	2-byte integer
status	status_$t

gpr_$set_obscured_opt(if_obscured, status)

if_obscured	gpr_$obscured_opt_t
status	status_$t

gpr_$set_overlay_color_map(start_index,n_entries, color_array, status)

start_index	gpr_$pixel_value_t
n_entries	2-byte integer
color_array	gpr_$color_vector_t
status	status_$t

gpr_$set_pixel_format (bitmap,pix_form,status)

bitmap	gpr_$bitmap_desc_t
pix_form	gpr_$pixel_format_t
status	status_$t

gpr_$set_plane_mask(16_bit_mask, status)

16_bit_mask	gpr_$mask_t
status	status_$t

gpr_$set_plane_mask_32(32_bit_mask, status)

32_bit_mask	gpr_$mask_32_t
status	status_$t

gpr_$set_quit_event(event_type,code,status)

event_type	gpr_$event_t
code	char
status	status_$t

gpr_$set_raster_op(plane, op, status)

plane	gpr_$rgb_plane_t
op	gpr_$raster_op_t
status	status_$t

gpr_$set_raster_op_mask(pl_mask, op, status)

pl_mask	gpr_$mask_32_t
op	gpr_$raster_op_t
status	status_$t

gpr_$set_refresh_entry(ptr_to_window_refresh_proc, ptr_to_hidden_memory_refresh_proc, status)

ptr_to_window_refresh_proc	gpr_$rwin_pr_t
ptr_to_hidden_memory_refresh_proc	gpr_$rhdm_pr_t
status	status_$t

gpr_$set_space_size(font_id, space_size, status)

font_id	2-byte integer
space_size	2-byte integer
status	status_$t

gpr_$set_text_background_value(color_value, status)

color_value	gpr_$pixel_value_t
status	status_$t

gpr_$set_text_font(font_id, status)

font_id	2-byte integer
status	status_$t

gpr_$set_text_path(path, status)

path	gpr_$direction_t
status	status_$t

gpr_$set_text_value(color_value, status)

color_value	gpr_$pixel_value_t
status	status_$t

gpr_$set_triangle_fill_criteria(fill_crit, status)

fill_crit	gpr_$triangle_fill_criteria_t
status	status_$t

gpr_$set_window_id(id_char, status)

id_char	char
status	status_$t

gpr_$spline_cubic_p(array_of_x_coords, array_of_y_coords, n_points, status)

array_of_x_coords	gpr_$coordinate_array_t
array_of_y_coords	gpr_$coordinate_array_t
n_points	2-byte integer
status	status_$t

gpr_$spline_cubic_x(array_of_x_coords, array_of_y_coords, n_points, status)

array_of_x_coords	gpr_$coordinate_array_t
array_of_y_coords	gpr_$coordinate_array_t
n_points	2-byte integer
status	status_$t

gpr_$spline_cubic_y(array_of_x_coords, array_of_y_coords, n_points, status)

array_of_x_coords	gpr_$coordinate_array_t
array_of_y_coords	gpr_$coordinate_array_t
n_points	2-byte integer
status	status_$t

gpr_$start_pgon(x_coord, y_coord, status)

x_coord	gpr_$coordinate_t
y_coord	gpr_$coordinate_t
status	status_$t

gpr_$terminate(delete_disp, status)

delete_disp	boolean
status	status_$t

gpr_$text(string, string_length, status)

string	gpr_$string_t
string_length	2-byte integer
status	status_$t

gpr_$text16(t_array,t_arrayl,status)

t_array	gpr_$16bit_character_array_t
t_arrayl	2-byte integer
status	status_$t

gpr_$trapezoid(trap, status)

trap	gpr_$trap_t
status	status_$t

gpr_$triangle(point1, point2, point3, status)

point1	gpr_$position_t
point2	gpr_$position_t
point3	gpr_$position_t
status	status_$t

gpr_$unload_font_file(font_id, status)

font_id	2-byte integer
status	status_$t

gpr_$wait_frame(status)

status	status_$t

gpr_$write_pixels(pix, destination_window, status)

pix	gpr_$pixel_array_t
destination_window	gpr_$window_t
status	status_$t

Chapter 3

GPR Data Types

This chapter details all of the GPR data types used as parameters in Chapter 2. Because Pascal and C support a greater variety of data structures and data types than FORTRAN, we've defined the parameters of Chapter 3 by using the Pascal and C data types. Nevertheless, all GPR data types can be represented or simulated in FORTRAN programs.

3.1 Understanding GPR Data Types

We wrote the following notes to help FORTRAN programmers understand GPR data types. (Note 2 should also be helpful to C programmers.)

Note 1 — Enumerated Variables

Pascal and C both support enumerated variables, but FORTRAN does not. However, the Domain system stores enumerated Pascal variables and short enum C variables the same way it stores FORTRAN integer*2 variables. Therefore, we've simulated enumerated variables in the gpr.ins.ftn insert file by defining integer*2 parameters.

If a GPR call requires an enumerated variable, declare the variable in your FORTRAN program as an integer*2. To set the variable's

value, you merely specify one of the listed choices and the insert file will convert it to the necessary internal representation.

Consider our description of the gpr_$display_mode_t type, which reads

gpr_$display_mode_t Pascal/C
integer*2 FORTRAN (see Note 1)
Enumerated type; possible values are

gpr_$frame
gpr_$direct
gpr_$direct_rgb
gpr_$borrow_rgb_nc
gpr_$borrow
gpr_$no_display
gpr_$borrow_nc
gpr_$borrow_rgb

The listing tells a FORTRAN programmer to declare gpr_$display_mode_t parameters as integer*2 variables; for example:

```
integer*2  my_display_mode_variable
```

You can set this variable to any one of the eight choices listed in italics; for example:

```
my_display_mode_variable = gpr_$direct
```

Note 2 — Set Variables

Pascal supports set variables; but C and FORTRAN do not. However, C and FORTRAN programmers can emulate Pascal set variables. If the base type of the Pascal set contains 32 or fewer members, then you can emulate the set by declaring an integer type. If the base type contains more than 32 members, then you should use special set emulation functions. The descriptions in this chapter tell you which emulation method is appropriate for a specific data type. For full details on set emulation, see the *Programming With General System Calls* manual.

Note 3 — Record and Structure Variables

Pascal supports record types that are identical to C's structure types. However, FORTRAN does not support such a structure. Nevertheless, you can usually use a FORTRAN array variable to simulate a Pascal record/C structure variable. For example, consider the following description of the gpr_$offset_t type:

gpr_$offset_t Pascal/C
integer*2 var(2) FORTRAN (see Note 3)

Record/structure containing two fields defined as follows:

Name of Field	Data Type of Field in Pascal or C	Element # in FTN Array
x_size	gpr_$coordinate_t	1
y_size	gpr_$coordinate_t	2

The preceding listing shows that FORTRAN programmers can emulate the gpr_$offset_t data type by declaring a 2-element array of integer*2's. A FORTRAN programmer can access the x_size field by accessing element number 1 in the array, and access the y_size field by accessing element number 2 in the array.

Note 4 — Pointer Types

Both Pascal and C support pointer data types, but standard FORTRAN does not. However, you can emulate a pointer variable in Domain FORTRAN by declaring an integer*4 variable and then writing addresses into it with the IADDR function. For example, you can emulate a gpr_$rhdm_pr_t or a gpr_$rwin_pr_t variable by declaring an integer*4 variable and then using the IADDR function to store the starting address of a refresh subroutine.

The gpr_$bmf_group_header_t and gpr_$group_header_t data types are somewhat more difficult to emulate. Both are record types containing five 2-byte integer fields and one pointer field. You can do the following to emulate a field of this type:

```
character  bitmap_array(size_of_bitmap)
integer*2  first_five_fields(5)
integer*4 p1, p2
pointer /p1/ first_five_fields, p2
pointer /p2/ bitmap_array
```

where p1 is the variable that emulates either the gpr_$bmf_group_header_t or the gpr_$group_header_t data type.

3.2 Alphabetical Listing of GPR Data Types

Following is an alphabetical list of all GPR data types. For more information, please see the "Data Types" section of the *Domain Standard Graphics Call Reference: GPR and CTM.*

gpr_$16bit_character_array_t Pascal/C
integer*2 var(256) FORTRAN
A 256-element array of unsigned 16-bit integers.

gpr_$accelerator_type_t Pascal/C
integer*2 FORTRAN (see Note 1)
Enumerated type; possible values are
gpr_$accel_none *gpr_$accel_1*

gpr_$access_allocation_t Pascal/C
integer*2 FORTRAN (see Note 1)
Enumerated type; possible values are
gpr_$alloc_1 *gpr_$alloc_2*
gpr_$alloc_4 *gpr_$alloc_8*
gpr_$alloc_16 *gpr_$alloc_32*

gpr_$access_mode_t Pascal/C
integer*2 FORTRAN (see Note 1)
Enumerated type; possible options are
gpr_$create *gpr_$update*
gpr_$write *gpr_$readonly*

gpr_$access_set_t Pascal
short int C (see Note 2)
integer*2 FORTRAN (see Note 2)
Set of gpr_$access_allocation_t type. This is a 6-element set.

gpr_$arc_direction_t Pascal/C
integer*2 FORTRAN (see Note 1)
Enumerated type; possible values are
gpr_$arc_ccw *gpr_$arc_cw*

gpr_$arc_option_t Pascal/C
integer*2 FORTRAN (see Note 1)
Enumerated type; possible values are
gpr_$arc_drawn_none *gpr_$arc_draw_full*

gpr_$attribute_desc_t Pascal/C
integer*4 FORTRAN
An unsigned 4-byte integer type.

gpr_$bitmap_desc_t Pascal/C
integer*4 FORTRAN
An unsigned 4-byte integer type.

gpr_$bmf_group_header_array_t Pascal/C
(see Note 4) FORTRAN
A 1-element array of gpr_$bmf_group_header_t.
The Pascal insert file defines this type as
gpr_$bmf_group_header_array_t =
array[0..gpr_$max_bmf_group] of gpr_$bmf_group_header_t
The C insert file defines this type as
gpr_$bmf_group_header_t
gpr_$bmf_group_header_array_t[gpr_$max_bmf_group+1]

gpr_$bmf_group_header_t Pascal/C
(see Note 4) FORTRAN

Record/structure containing six fields. Note that the final field (storage_offset) is defined as a UNIV_PTR in Pascal and as a pointer to a char in C. The record/structure is defined as follows:

Name of Field	Data Type of Field in Pascal or C	Element # in FTN Array
n_sects	2-byte integer	1
pixel_size	2-byte integer	2
allocated_size	2-byte integer	3
bytes_per_line	2-byte integer	4
bytes_per_sect	4-byte integer	5,6
storage_offset	pointer	7,8

gpr_$color_t Pascal/C
integer*4 FORTRAN
An unsigned 4-byte integer type.

gpr_$color_map_char_t Pascal/C
integer*4 num_color_maps FORTRAN
integer*2 color_map(gpr_$max_physical_color_maps)
Record/structure containing two fields defined as follows:

Name of Field	Data Type of Field in Pascal or C
num_color_maps	integer32/linteger
color_map[gpr_$max_physical_color_maps]	gpr_$curr_color_map_status_t

Each element of color_map must be equal to one of the following predefined values:
gpr_$compatible_color_map
gpr_$incompatible_color_map

gpr_$color_map_id_t Pascal/C
integer*2 FORTRAN

gpr_$color_t Pascal/C
integer*4 FORTRAN

gpr_$color_vector_t Pascal/C
integer*4 var(256) FORTRAN
A 256-element array of gpr_$color_t type.

gpr_$controller_type_t Pascal/C
integer*2 FORTRAN (see Note 1)
Enumerated type; possible values are

gpr_$ctl_none	*gpr_$ctl_mono_1*
gpr_$ctl_mono_2	*gpr_$ctl_color_1*
gpr_$ctl_color_2	*gpr_$ctl_color_3*
gpr_$ctl_color_4	*gpr_$ctl_mono_4*
gpr_$ctl_color_5	*gpr_$ctl_mono_5*
gpr_$ctl_color_6	*gpr_$ctl_color_7*
gpr_$ctl_color_10	

gpr_$coordinate_array_t Pascal/C
integer*2 num_of_coords(16384) FORTRAN (see Note 3)
A predefined Pascal and C array data type that contains 16,384 elements. You can use this predefined type or you can define your own variables. To define your own variable in Pascal, use the following format:

var: array[1..number_of_coords] of gpr_$coordinate_t

To define your own variable in C, use the following format:

gpr_$coordinate_t var[number_of_coords]

gpr_$coordinate_t Pascal/C
integer*2 FORTRAN
A 2-byte integer type.

gpr_$curr_color_map_status_t Pascal/C
integer*2 FORTRAN (see Note 1)
Enumerated type; possible options are
gpr_$compatible_color_map *gpr_$compatible_color_map*

gpr_$cursor_mode_set_t Pascal
short int C
integer*2 FORTRAN
Set of gpr_cursor_mode_t.

gpr_$cursor_mode_t Pascal/C
integer*2 FORTRAN (see Note 1)
Enumerated type; possible options are
gpr_$software_cursor *gpr_$hardware_cursor*

gpr_$decomp_technique_t Pascal/C
integer*2 FORTRAN (see Note 1)
Enumerated type; possible values are
gpr_$fast_traps *gpr_$precise_traps*
gpr_$non_overlapping_tris *gpr_$render_exact*

gpr_$direction_t Pascal/C
integer*2 FORTRAN (see Note 1)
Enumerated type; possible options are
gpr_$up *gpr_$down*
gpr_$left *gpr_$right*

gpr_$disp_char_t Pascal/C
integer*2 var(34) FORTRAN(see Note 3)

Record/structure containing 34 fields defined as follows:

Name of Field	Data Type of Field in Pascal or C	Element in FTN Array
controller_type	gpr_$controller_type_t	1
accelerator_type	gpr_$accelerator_type_t	2
x_window_origin	2-byte integer	3
y_window_origin	2-byte integer	4
x_window_size	2-byte integer	5
y_window_size	2-byte integer	6
x_visible_size	2-byte integer	7
y_visible_size	2-byte integer	8
x_extension_size	2-byte integer	9
y_extension_size	2-byte integer	10
x_total_size	2-byte integer	11
y_total_size	2-byte integer	12
x_pixels_per_cm	2-byte integer	13
y_pixels_per_cm	2-byte integer	14
n_planes	2-byte integer	15
n_buffers	2-byte integer	16
delta_x_per_buffer	2-byte integer	17
delta_y_per_buffer	2-byte integer	18
delta_planes_per_buffer	2-byte integer	19
mem_overlaps	gpr_$overlap_set_t	20
x_zoom_max	2-byte integer	21
y_zoom_max	2-byte integer	22
video_refresh_rate	2-byte integer	23
n_primaries	2-byte integer	24
lut_width_per_primary	2-byte integer	25
avail_formats	gpr_$format_set_t	26
avail_access	gpr_$access_set_t	27
access_address_space	2-byte integer	28
invert	gpr_$disp_invert_t	29
num_lookup_tables	2-byte integer	30
rgb_color	gpr_$rgb_modes_set_t	31
default_cursor_mode	gpr_$cursor_mode_t	32
avail_cursor_modes	gpr_$cursor_mode_set_t	33
n_mult_clips	integer16/short int	34

gpr_$disp_invert_t Pascal/C
integer*2 FORTRAN (see Note 1)
Enumerated type; possible values are
gpr_$invert_simulate *gpr_$no_invert*
gpr_$invert_hardware

gpr_$display_config_t Pascal/C
integer*2 FORTRAN (see Note 1)
Enumerated type; possible values are
gpr_$bw_800x1024 *gpr_$bw_1024x800*
gpr_$color_1024x1024x4 *gpr_$color_1024x1024x8*
gpr_$color_1024x800x4 *gpr_$color_1024x800x8*
gpr_$color_1280x1024x8 *gpr_$color1_1024x800x8*
gpr_$color2_1024x800x4 *gpr_$bw_1280x1024*
gpr_$color2_1024x800x8 *gpr_$color2_1280x1024x8*
gpr_$color10_1280x1024 *gpr_$color7_1280x1024*
gpr_$mono9_2kx1k

gpr_$display_mode_t Pascal/C
integer*2 FORTRAN (see Note 1)
Enumerated type; possible values are
gpr_$borrow *gpr_$frame*
gpr_$no_display *gpr_$direct*
gpr_$borrow_nc *gpr_$direct_rgb*
gpr_$borrow_rgb *gpr_$borrow_rgb_nc*

gpr_$double_buffer_option_t Pascal/C
integer*2 FORTRAN (see Note 1)
Enumerated type; possible options are
gpr_$undisturbed_buffer *gpr_$clear_buffer*
gpr_$copy_buffer

gpr_$ec_key_t Pascal/C
integer*2 FORTRAN (see Note 1)
Enumerated type; only possible value is
gpr_$input_ec

gpr_$event_data_t Pascal/C
integer*4 var (2) FORTRAN
In C, this is a union with one member, gpr_$dial, which is a structure. In Pascal, this is a variant record with a case of gpr_$event_t, which has one possible constant, gpr_$dial. For the gpr_$dial constant, the variant record contains a record with two fields:

Name of Field	Data Type of Field in Pascal or C
dial_number	integer32/long int
dial_value	integer32/long int

gpr_$event_t Pascal/C
integer*2 FORTRAN (See Note 1)
Enumerated type; possible values are

gpr_$keystroke	*gpr_$buttons*
gpr_$locator	*gpr_$entered_window*
gpr_$left_window	*gpr_$locator_stop*
gpr_$no_event	*gpr_$locator_update*
gpr_$dial	*gpr_$coded_keys*
gpr_$function_keys	*gpr_$pfk*
gpr_$physical_keys	*gpr_$kbd_entered_window*
gpr_$kbd_left_window	

gpr_$format_set_t Pascal
short int C (see Note 2)
integer*2 FORTRAN (see Note 2)
Set of gpr_$imaging_format_t. This is a 6-element set.

gpr_$horiz_seg_t Pascal/C
integer*2 var(3) FORTRAN (see Note 3)
Record/structure containing three fields defined as follows:

Name of Field	Data Type of Field in Pascal or C	Element # in FTN Array
x_coord_l	gpr_$coordinate_t	1
x_coord_r	gpr_$coordinate_t	2
y_coord	gpr_$coordinate_t	3

gpr_$imaging_format_t Pascal/C
integer*2 FORTRAN (see Note 1)
Enumerated type; possible values are
gpr_$interactive *gpr_$imaging_1024x1024x8*
gpr_$imaging_512x512x24

gpr_$init_options_set_t Pascal
long int C (see Note 2)
integer*4 FORTRAN (see Note 2)
Set of gpr_$init_options_t type. This is a 32-element set.

gpr_$init_options_t Pascal/C
integer*4 FORTRAN (see Note 1)
Enumerated type; possible values are
gpr_$no_clear *gpr_$plane_mode*
gpr_$pixel_mode

gpr_$keyset_t Pascal
(Use set emulation functions) C (see Note 2)
(Use set emulation functions) FORTRAN (see Note 2)
Set of char type. This is a 256-element set.

gpr_$line_pattern_t Pascal/C
integer*2 var(4) FORTRAN
A 4-element array of 2-byte integers.

gpr_$linestyle_t Pascal/C
integer*2 FORTRAN (see Note 1)
Enumerated type; possible options are
gpr_$solid *gpr_$dotted*

gpr_$mask_t Pascal
short int var C (see Note 2)
integer*2 var FORTRAN (see Note 2)
Set of gpr_$plane_t type. This is a 16-element set.

gpr_$mask_32_t Pascal
long int var C (see Note 2)
integer*4 var FORTRAN (see Note 2)
Set of gpr_$rgb_plane_t type. This is a 32-element set.

gpr_$memory_overlap_t Pascal/C
integer*2 FORTRAN (see Note 1)
Enumerated type; possible values are
gpr_$hdm_with_bitm_ext *gpr_$hdm_with_buffers*
gpr_$bitm_ext_with_buffers *gpr_$bitm_ext_with_zbuffer*

gpr_$obscured_opt_t Pascal/C
integer*2 FORTRAN (see Note 1)
Enumerated type; possible values are
gpr_$ok_if_obs *gpr_$error_if_obs*
gpr_$pop_if_obs *gpr_$block_if_obs*
gpr_$input_ok_if_obs

gpr_$offset_t Pascal/C
integer*2 var(2) FORTRAN (see Note 3)
Record/structure containing two fields defined as follows:

Name of Field	Data Type of Field in Pascal or C	Element # in FTN Array
x_size	gpr_$coordinate_t	1
y_size	gpr_$coordinate_t	2

gpr_$overlap_set_t Pascal
short int var C (see Note 2)
integer*2 var FORTRAN (see Note 2)
Set of gpr_$memory_overlap_t. This is a 3-element set.

gpr_$pixel_array_t Pascal/C
integer*4 var(131072) FORTRAN

A predefined Pascal and C array data type that contains 131,072 elements. You can use this predefined type or you can define your own variable. To define your own variable in Pascal, use the following format:

var : array[1..n_of_elements] of gpr_$pixel_value_t

To define your own variable in C, use the following format:

gpr_$pixel_value_t var[n_of_elements] in C

gpr_$pixel_format_array_t Pascal/C
integer*4 var(16,gpr_$max_formats) FORTRAN

An array of type gpr_$pixel_format_t with gpr_$max_formats elements.

gpr_$pixel_format_t Pascal/C
integer*4 var(16) FORTRAN (see Note 3)

Record/structure containing the following fields:

Name of Field	Data Type of Field in Pascal or C	Element in FTN Array
length	integer32/long int	1
pixel_mode	integer32/long int	2
image_depth	integer32/long int	3
buffer_count	integer32/long int	4
red_depth	integer32/long int	5
green_depth	integer32/long int	6
blue_depth	integer32/long int	7
ovlay_mode	integer32/long int	8
ovlay_depth	integer32/long int	9
ovlay_buffer_count	integer32/long int	10
z_mode	integer32/long int	11
z_depth	integer32/long int	12
z_buffer_count	integer32/long int	13
alpha_mode	integer32/long int	14
alpha_depth	integer32/long int	15
alpha_buffer_count	integer32/long int	16

gpr_$pixel_value_t Pascal/C
integer*4 FORTRAN

An unsigned 4-byte integer type.

gpr_$plane_ptr_t Pascal/C
integer*4 FORTRAN (see Note 4)

A pointer type.
In Pascal, this type is defined as
gpr_$plane_ptr_t = univ_ptr
In C, this type is defined as
*typedef char *gpr_$plane_ptr_t*

gpr_$plane_t Pascal/C
integer*2 FORTRAN
A 2-byte integer type.
In Pascal, this type is a subrange of integers from 0 to 7.
In C, this type is defined as an unsigned short int.

gpr_$position_t Pascal/C
integer*2 var(2) FORTRAN (see Note 3)
Record/structure containing two fields defined as follows:

Name of Field	Data Type of Field in Pascal or C	Element # in FTN Array
x_coord	gpr_$coordinate_t	1
y_coord	gpr_$coordinate_t	2

gpr_$proj_format_t Pascal/C
integer*4 var(4) FORTRAN (see Note 3)
Record/structure containing two fields defined as follows:

Name of Field	Data Type of Field in Pascal or C	Element # in FTN Array
length	integer32/long int	1
proj_mode	integer32/long int	2
proj_buffer	integer32/long int	3
reserved	integer32/long int	4

gpr_$raster_op_array_t Pascal/C
integer*2 var(32) FORTRAN
A 32-element array of raster operation values.
In Pascal, this type is defined as
gpr_$raster_op_array_t = array[gpr_$rgb_plane_t] of gpr_$raster_op_t;
In C, this type is defined as
gpr_$raster_op_t gpr_$raster_op_array_t[32];

gpr_$raster_op_t Pascal/C
integer*2 FORTRAN
Pascal integer subrange variable (subrange = 0..15),
C unsigned short variable.

gpr_$resource_type_t Pascal/C
integer*2 FORTRAN (see Note 1)
Enumerated type; possible values are
gpr_$memory_bitmap *gpr_$pad_id*
gpr_$pad_frame_id reserved
gpr_$screen *gpr_$x_window_id*

gpr_$rgb_modes_set_t Pascal
short int var C (see Note 2)
integer*2 var FORTRAN (see Note 2)
Set of gpr_$rgb_modes_t. This is a 2-element set.

gpr_$rgb_modes_t Pascal/C
integer*2 FORTRAN (see Note 1)
Enumerated type; possible values are
gpr_$rgb_none,
gpr_$rgb_24

gpr_$rgb_plane_t Pascal/C
integer*2 FORTRAN
A 2-byte integer type.
In Pascal, this type is a subrange of integers from 0 to 31.
In C, this type is defined as an unsigned short int.

gpr_$rhdm_pr_t Pascal/C
integer*4 FORTRAN (see Note 4)
A pointer type.
In Pascal, this type is defined as
gpr_$rhdm_pr_t = ^PROCEDURE;
In C, this type is defined as
*typedef void (*gpr_$rhdm_pr_t)();*

gpr_$rop_prim_set_elems_t Pascal/C
integer*2 FORTRAN (see Note 1)
Enumerated type; possible options are
gpr_$rop_blt *gpr_$rop_line*
gpr_$rop_fill

gpr_$rop_prim_set_t Pascal
short int var C (see Note 2)
integer*2 var FORTRAN (see Note 2)
Set of gpr_$rop_prim_set_elems_t type. This is a 3-element set.

gpr_$rwin_pr_t Pascal/C
integer*4 FORTRAN (see Note 4)
A pointer type.
In Pascal, this type is defined as

```
gpr_$rwin_pr_t = ^PROCEDURE( IN unobscured: boolean;
                             IN pos_change: boolean );
```

In C, this type is defined as

```
typedef void (*gpr_$rwin_pr_t)();
```

gpr_$string_t Pascal/C
char var[256] FORTRAN
Array of 256 characters.

gpr_$trap_list_t Pascal/C
integer*2 var(number_of_trapezoids,6) FORTRAN (see Note 3)
A predefined Pascal and C array data type that contains gpr_$default_list_size elements. You can use this predefined type or you can define your own variable. To define your own variable in Pascal, use the following format:

```
var : array[1..number_of_trapezoids] of gpr_$trap_t
```

To define your own variable in C, use the following format:

```
gpr_$trap_t  var[number_of_trapezoids]  in C
```

gpr_$trap_t Pascal/C
integer*2 var(6) FORTRAN (see Note 3)
Record/structure containing two fields defined as follows:

Name of Field	Data Type of Field in Pascal or C	Element # in FTN Array
top	gpr_$horiz_seg_t	1,2,3
bot	gpr_$horiz_seg_t	4,5,6

gpr_$triangle_fill_criteria_t Pascal/C
integer*2 var(2) FORTRAN (see Note 3)

Record/structure containing two fields defined as follows:

Name of Field	Data Type of Field in Pascal or C	Element # in FTN Array
wind_type	gpr_$winding_set_t	1
winding_no	2-byte integer	2

gpr_$triangle_list_t Pascal/C
integer*2 var(number_of_triangles,7) FORTRAN

A predefined Pascal and C array data type that contains gpr_$default_list_size elements. You can use this predefined type or you can define your own variables. To define your own variable in Pascal, use the following format:

var: array[1..number_of_triangles] of gpr_$triangle_t

To define your own variable in C, use the following format:

gpr_$triangle_t var[number_of_triangles]

gpr_$triangle_t Pascal/C
integer*2 var(7) FORTRAN (see Note 3)

Record/structure containing four fields defined as follows:

Name of Field	Data Type of Field in Pascal or C	Element # in FTN Array
p1	gpr_$position_t	1,2
p2	gpr_$position_t	3,4
p3	gpr_$position_t	5,6
winding	2-byte integer	7

gpr_$version_t Pascal/C
integer*2 var(2) FORTRAN (see Note 3)

Record/structure containing two fields defined as follows:

Name of Field	Data Type of Field in Pascal or C	Element # in FTN Array
major	2-byte integer	1
minor	2-byte integer	2

gpr_$video_format_t Pascal/C
integer*4 var(2) FORTRAN (see Note 3)

Record/structure containing two fields defined as follows:

Name of Field	Data Type of Field in Pascal or C	Element # in FTN Array
length	integer32/long int	1
video_buffer	integer32/long int	2

gpr_$winding_set_t Pascal/C
integer*2 FORTRAN (see Note 1)

Enumerated type; possible values are
gpr_$parity *gpr_$nonzero*
gpr_$specific

gpr_$window_list_t Pascal/C
integer*2 var(4,n_windows) FORTRAN (see Note 3)

This is a predefined Pascal and C array data type that contains gpr_$default_list_size elements. You can use this predefined type or you can define your own variables. To define your own variable in Pascal, use the following format:

var : array[1..number_of_windows] of gpr_$window_t

To define your own variable in C, use the following format:

gpr_$window_t var[number_of_windows] in C

gpr_$window_t Pascal/C
integer*2 var(4) FORTRAN (see Note 3)

Record/structure containing two fields defined as follows:

Name of Field	Data Type of Field in Pascal or C	Element # in FTN Array
window_base	gpr_$position_t	1,2
window_size	gpr_$offset_t	3,4

3.3 Other Data Types Used in GPR Calls

In Chapter 2, we also use the following additional data types:

2-byte integer	
integer16	Pascal
short int	C
integer*2	FORTRAN
4-byte integer	
integer32	Pascal
long int	C
integer*4	FORTRAN
boolean	Pascal
short int	C
logical	FORTRAN
char	Pascal/C
character	FORTRAN

Chapter 4

Raster Operations

Table 4–1. Raster Operations Truth Table

SOURCE BIT VALUE		0	0	1	1
DESTINATION BIT VALUE		0	1	0	1
RESULTANT BIT VALUES FOR THE FOLLOWING RASTER OP CODES:	0	0	0	0	0
	1	0	0	0	1
	2	0	0	1	0
	3	0	0	1	1
	4	0	1	0	0
	5	0	1	0	1
	6	0	1	1	0
	7	0	1	1	1
	8	1	0	0	0
	9	1	0	0	1
	10	1	0	1	0
	11	1	0	1	1
	12	1	1	0	0
	13	1	1	0	1
	14	1	1	1	0
	15	1	1	1	1

Table 4–2. Raster Operations and Their Functions

Raster Op Code	Logical Function
0	Assign zero to all new destination values.
1	Assign source AND destination to new destination.
2	Assign source AND complement of destination to new destination.
3	Assign all source values to new destination. (Default)
4	Assign complement of source AND destation to new destination.
5	Assign all destination values to new destination.
6	Assign source EXCLUSIVE OR destination to new destination.
7	Assign source OR destination to new destination.
8	Assign complement of source AND complement of destination to new destination.
9	Assign source EQUIVALENCE destination to new destination.
10	Assign complement of destination to new destination.
11	Assign source OR complement of destination to new destination.
12	Assign complement of source to new destination.
13	Assign complement of source OR destination to new destination.
14	Assign complement of source OR complement of destination to new destination.
15	Assign one to all new destination values.

See the *Programming with Domain Graphics Primitives* manual for details on raster operations.

Chapter 5

CTM Calls Categorized by Purpose

The following is a list of CTM routines grouped by function.

5.1 Accessing Colors

ctm_$allocate_pv — Allocates pixel values and sets their use counts to one.

ctm_$find_color — Finds the specified color value.

ctm_$inc_use_count — Increments pixel value use counts

ctm_$mark_read_only — Shares pixel values with other processes.

ctm_$release_pv — Decrements pixel value use counts.

5.2 Using Multiple Color Maps

ctm_$inq_curr_color_map — Returns the color map ID for CTM.

ctm_$set_curr_color_map — Sets the current CTM color map.

Chapter 6

CTM Calls and Their Parameters

This chapter describes the required arguments for each CTM call. For instance, consider the description of the **ctm_$allocate_pv** call. The listing shows that this call takes five arguments (count, option, plane, pixel_values, and status). It also shows that the first argument is a 2-byte integer; the second argument takes the ctm_$alloc_options_t data type; the third argument is a 2-byte integer; the fourth argument takes the ctm_$pixel_value_vector_t data type; and the fifth argument takes the status_$t data type. (See Chapter 7 for an explanation of the CTM data types.)

Parameters shown in black are input parameters, and parameters shown in blue are output parameters returned by GPR.

All CTM calls are Pascal procedures (which corresponds to a function returning void in C and a subroutine in FORTRAN).

Here are the calls listed in alphabetical order:

ctm_$alloc_pv (count,option,plane,pixel_values,status)

count	2-byte integer
option	ctm_$alloc_options_t
plane	2-byte integer
pixel_values	ctm_$pixel_value_vector_t (input and output)
status	status_$t

ctm_$find_color (color,color_radius,pixel_value,status)

color	gpr_$color_t
color_radius	2-byte integer
pixel_value	gpr_$pixel_value_t
status	status_$t

ctm_$inc_use_count (count,option,plane,pixel_values,status)

count	2-byte integer
option	ctm_$alloc_options_t
plane	2-byte integer
pixel_values	ctm_$pixel_value_vector_t
status	status_$t

ctm_$inq_curr_color_map (color_map_id,display_unit,options, status)

color_map_id	gpr_$color_map_id
display_unit	2-byte integer
status	status_$t

ctm_$mark_read_only (count,option,plane,pixel_values,status)

count	2-byte integer
option	ctm_$alloc_options_t
plane	2-byte integer
pixel_values	ctm_$pixel_value_vector_t
status	status_$t

ctm_$release_pv (count,option,plane,pixel_values,status)

count	2-byte integer
option	ctm_$alloc_options_t
plane	2-byte integer
pixel_values	ctm_$pixel_value_vector_t
status	status_$t

ctm_$set_curr_color_map (color_map_id,display_unit,options, status)

color_map_id	gpr_$color_map_id
display_unit	2-byte integer
options	ctm_$color_map_options_t
status	status_$t

Chapter 7

CTM Data Types

This chapter details all of the CTM data types used as parameters in Chapter 6. Because Pascal and C support a greater variety of data structures and data types than FORTRAN, we've defined the parameters of Chapter 6 by using the Pascal and C data types. Nevertheless, all CTM data types can be represented or simulated in FORTRAN programs.

7.1 Understanding CTM Data Types

Pascal supports set variables; but C and FORTRAN do not. However, C and FORTRAN programmers can emulate Pascal set variables. If the base type of the Pascal set contains 32 or fewer members, then you can emulate the set by declaring an integer type. If the base type contains more than 32 members, then you should use special set emulation functions. The descriptions in this chapter tell you which emulation method is appropriate for a specific data type. For full details on set emulation, see the *Programming With General System Calls* manual.

7.2 Alphabetical Listing of CTM Data Types

Following is an alphabetical list of all CTM data types. For more information, please see the "Data Types" section of the *Domain Standard Graphics Call Reference: GPR and CTM.*

ctm_$alloc_options_t Pascal/C
integer*2 FORTRAN (see Note)
Set of 2-byte integer type. This is a single-element set.

ctm_$color_map_options_t Pascal/C
integer*2 FORTRAN (see Note)
Set of 2-byte integer type. This is a single-element set.

ctm_$pixel_value_vector_t Pascal/C
integer*4 var (1) FORTRAN
A single-element array of 4-byte integers.

Reader's Response

Please take a few minutes to send us the information we need to revise and improve our manuals from your point of view.

Document Title: *Domain Standard Graphics Quick Reference: GPR and CTM*
Order No.010430-A00

Your Name _____ Date _____

Organization _____

Street Address _____

City _____ State _____ Zip _____

Telephone number (_____) _____

When you use the Apollo system, what **job(s)** do you perform?

- ☐ Programming
- ☐ Hardware Engineering
- ☐ Other (describe)_____
- ☐ Application End User
- ☐ System Administration

How many years of **experience** do you have in using the Apollo system:

_____

What programming **languages** do you use with the Apollo system?

_____

How would you **evaluate** this book?

	Excellent		Average		Poor
Completeness	1	2	3	4	5
Accuracy	1	2	3	4	5
Usability	1	2	3	4	5

Additional Comments: _____

_____

_____

No postage necessary if mailed in the U.S.

cut or fold along dotted line

fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 78 CHELMSFORD, MA 01824

POSTAGE WILL BE PAID BY ADDRESSEE

APOLLO COMPUTER INC.
Technical Publications
P.O. Box 451
Chelmsford, MA 01824

fold

D10430-A00