PRO/GIDIS Manual

Order No. AA-Y660A-TK

December 1983

This document describes PRO/GIDIS, DIGITAL's General Image Display Instruction Set, as implemented for the Professional Developer's Tool Kit. It is a user guide and reference manual for programmers developing graphics applications for the Professional personal computers.

DEVELOPMENT SYSTEM:	Professional Host Tool Kit V1.7 or later PRO/Tool Kit V1.0 or later
SOFTWARE SYSTEM:	PRO/GIDIS V1.7

DIGITAL EQUIPMENT CORPORATION
Maynard, Massachusetts 01754

First Printing, December 1983

CONTENTS

PREFACE vii

CHAPTER 1 INTRODUCTION TO PRO/GIDIS

1.1 OVERVIEW 1-1
1.2 RELATIONSHIP TO OTHER GRAPHICS TOOLS 1-2
1.2.1 WHEN TO USE PRO/GIDIS 1-2
1.2.2 WHEN NOT TO USE PRO/GIDIS 1-3
1.3 THE PRO/GIDIS INSTRUCTION SET 1-4
1.4 CONTROL INSTRUCTIONS 1-5
1.5 THE VIEWING TRANSFORMATION 1-6
1.5.1 Aspect Ratios 1-7
1.5.2 GIDIS Output Space 1-8
1.5.3 Imposed Device Space 1-10
1.5.4 The Viewport 1-11
1.5.5 Hardware Address Space 1-11
1.5.6 The Viewing Transformation 1-12
1.5.7 Addressing Pixels 1-13
1.6 DRAWING INSTRUCTIONS 1-13
1.7 FILLED FIGURE INSTRUCTIONS 1-14
1.8 TEXT INSTRUCTIONS 1-15
1.9 COLOR ATTRIBUTES 1-15
1.10 THE WRITING MODE 1-16
1.11 LINE AND CURVE ATTRIBUTES 1-19
1.11.1 Line Texture 1-19
1.11.2 Pixel Size 1-20
1.12 FILLED FIGURE ATTRIBUTES 1-21
1.12.1 Area Texture 1-21
1.12.2 Area Texture Size 1-22
1.12.3 Area Texture Cell Size 1-22
1.12.4 Shading to a Line or Point 1-22
1.12.5 Filled Figure Examples 1-23
1.13 TEXT ATTRIBUTES 1-25
1.13.1 Alphabets 1-25
1.13.2 Cell Rendition 1-27
1.13.3 Cell Rotation 1-28
1.13.4 Cell Oblique 1-28
1.13.5 Cell Unit Size 1-29
1.13.6 Cell Display Size 1-29
1.13.7 Cell Movement 1-29
1.14 AREA OPERATIONS INSTRUCTIONS 1-31
1.15 REPORT HANDLING INSTRUCTIONS 1-31

CHAPTER 2 INTERACTING WITH THE PRO/GIDIS INTERPRETER

2.1 THE PRO/GIDIS INTERFACE 2-1
2.1.1 Write Special Data (IO.WSD) 2-3
2.1.2 Read Special Data (IO.RSD) 2-4

2.2 PRO/GIDIS INSTRUCTION SYNTAX 2-6
2.2.1 Operation Codes 2-6
2.2.2 Parameter Blocks 2-7
2.3 SAMPLE MACRO-11 PROGRAM 2-8
2.4 SAMPLE FORTRAN PROGRAM 2-9

CHAPTER 3 CONTROL INSTRUCTIONS

3.1 INITIALIZE 3-1
3.2 NEW_PICTURE 3-6
3.3 END_PICTURE 3-7
3.4 FLUSH_BUFFERS 3-8
3.5 SET_OUTPUT_CURSOR 3-8
3.6 NOP 3-10
3.7 END_LIST 3-11

CHAPTER 4 VIEWING TRANSFORMATION INSTRUCTIONS

4.1 SET_OUTPUT_IDS 4-1
4.2 SET_OUTPUT_VIEWPORT 4-3
4.3 SET_GIDIS_OUTPUT_SPACE 4-4
4.4 SET_OUTPUT_CLIPPING_REGION 4-6

CHAPTER 5 GLOBAL ATTRIBUTES INSTRUCTIONS

5.1 SET_PRIMARY_COLOR 5-1
5.2 SET_SECONDARY_COLOR 5-2
5.3 SET_COLOR_MAP_ENTRY 5-3
5.4 SET_PLANE_MASK 5-4
5.5 SET_WRITING_MODE 5-7
5.6 SET_PIXEL_SIZE 5-7
5.7 SET_LINE_TEXTURE 5-9
5.8 SET_AREA_TEXTURE 5-10
5.9 SET_AREA_TEXTURE_SIZE 5-11
5.10 SET_AREA_CELL_SIZE 5-12

CHAPTER 6 DRAWING INSTRUCTIONS

6.1 SET_POSITION 6-1
6.2 SET_REL_POSITION 6-2
6.3 DRAW_LINES 6-3
6.4 DRAW_REL_LINES 6-5
6.5 DRAW_ARC 6-7
6.6 DRAW_REL_ARC 6-9

CHAPTER 7 FILLED FIGURE INSTRUCTIONS

7.1 BEGIN_FILLED_FIGURE 7-1
7.2 END_FILLED_FIGURE 7-2

CHAPTER 8 TEXT INSTRUCTIONS

8.1 SET_ALPHABET 8-2
8.2 CREATE_ALPHABET 8-2
8.3 LOAD_CHARACTER_CELL 8-4
8.4 SET_CELL_RENDITION 8-5
8.5 SET_CELL_ROTATION 8-6
8.6 SET_CELL_OBLIQUE 8-6
8.7 SET_CELL_UNIT_SIZE 8-7
8.8 SET_CELL_DISPLAY_SIZE 8-8
8.9 SET_CELL_MOVEMENT_MODE 8-9
8.10 SET_CELL_EXPLICIT_MOVEMENT 8-10
8.11 DRAW_CHARACTERS 8-11

CHAPTER 9 AREA OPERATION INSTRUCTIONS

9.1 ERASE_CLIPPING_REGION 9-1
9.2 PRINT_SCREEN 9-2

CHAPTER 10 REPORT HANDLING

10.1 REQUEST_CURRENT_POSITION 10-1
10.2 REQUEST_STATUS 10-2
10.3 REQUEST_CELL_STANDARD 10-3

APPENDIX A PRO/GIDIS INSTRUCTION SUMMARIES

A.1 INSTRUCTIONS GROUPED BY FUNCTION A-1
A.2 INSTRUCTIONS IN OPCODE ORDER A-2
A.3 INSTRUCTIONS IN ALPHABETIC ORDER A-3
A.4 REPORT TAGS A-5

APPENDIX B DEC MULTINATIONAL CHARACTER SET

APPENDIX C GLOSSARY

EXAMPLES

1-1 Context for Filled Figure Examples 1-23
1-2 A Filled Figure Using DRAW_LINES 1-24
1-3 A Filled Figure Using DRAW_ARC 1-24
1-4 A Filled Figure Using DRAW_ARC 1-24
1-5 A Filled Figure Using DRAW_REL_ARC 1-24
1-6 A Filled Figure Using DRAW_REL_ARC 1-25
1-7 A Filled Figure Using DRAW_LINES 1-25
1-8 A Filled Figure Using DRAW_LINES 1-25
2-1 Instruction with Fixed-Length Parameter Block . . 2-7
2-2 Instruction with Variable-Length Parameter Block . 2-7

FIGURES

1-1 PRO/GIDIS Sample Output 1-1
1-2 PRO/GIDIS Interface, Programs/Video Hardware . . . 1-3
1-3 Isotropic Mapping: Window to Viewport 1-9
1-4 Isotropic Mapping: IDS to HAS 1-10
1-5 Video Hardware Address Space 1-11
1-6 The Viewing Transformation 1-12
1-7 The Extended Bitmap Option 1-15
1-8 The Writing Modes Shown with Line Texture . . . 1-17
1-9 The Logical Drawing Pixel 1-21
1-10 Filled Figure Images 1-23
1-11 Alphabet 0 1-26
1-12 Character Cell Rotation 1-27
1-13 Character Unit Cell and Display Cell 1-28
1-14 Character Cell Movement 1-30
2-1 PRO/GIDIS Data Path 2-2
3-1 INITIALIZE Subsystem Initialization Bit Mask . . . 3-2
8-1 SET_CELL_RENDITION Bit Mask 8-5

TABLES

3-1 Control Instructions Summary Chart 3-1
3-2 Initialization Subsystems 3-2
3-3 Initialization Variable States 3-3
4-1 Viewing Transformation Instructions Summary Chart 4-1
4-2 State Variables Affected by SET_OUTPUT_IDS . . . 4-2
4-3 State Variables Affected by SET_GIDIS_OUTPUT_SPACE 4-5
5-1 Global Attribute Instructions Summary Chart . . . 5-1
5-2 Color Map Values for the Professional 5-4
5-3 Plane Mask Values 5-5
6-1 Drawing Instructions Summary Chart 6-1
7-1 Filled Figure Summary Chart 7-1
8-1 Text Instructions Summary Chart 8-1
8-2 Cell Rotation Angles 8-6
9-1 Area Operations Summary Chart 9-1
10-1 Report Handling Summary Chart 10-1

WHO SHOULD READ THIS MANUAL

You should read this manual if you are developing a graphics application for the Professional 300 Series computer and need information about PRO/GIDIS, the General Image Display Instruction Set that runs on the Professional computer. PRO/GIDIS is one of the tools that can be used in developing graphics applications for the Professional computer.

This document is intended for programmers who have had experience with systems programming and graphics applications software. The reader also is expected to be familiar with the Professional 300 Series Tool Kit, and either MACRO-11 or FORTRAN.

It is recommended that you also read the CORE Graphics Library (CGL) Manual for a more tutorial approach to graphics software development on the Professional computer.

SCOPE OF MANUAL

This manual describes PRO/GIDIS and is intended to be used as both a reference manual and user guide. It covers applications running on the Professional and also provides programming information about device-independent text and graphics programming with PRO/GIDIS.

ORGANIZATION OF MANUAL

The manual has ten chapters and three appendixes. The contents are summarized in the following subsections.

Chapter 1 -- Introduction to PRO/GIDIS

This chapter is the "user guide" for PRO/GIDIS. It provides an overview of PRO/GIDIS and its relationship to other graphics products, suggests when PRO/GIDIS should and should not be used, summarizes the PRO/GIDIS instruction groups, and discusses each group in detail.

Chapter 2 -- Interacting with the PRO/GIDIS Interpreter

Describes the software interface to PRO/GIDIS. It describes the Queue I/O (QIO) directives that send instructions to PRO/GIDIS and return GIDIS reports, and provides PRO/GIDIS programming syntax rules and programming examples.

Chapter 3 -- Control Instructions

Details the PRO/GIDIS instructions for program start-up, initialization, along with syntax-required instructions.

Chapter 4 -- Viewing Transformation Instructions

Describes the instructions used for mapping graphics addressable image areas to hardware output devices and the concepts involved in area transformation.

Chapter 5 -- Global Attributes Instructions

Details the global parameters that govern the appearance of drawing primitives (writing modes, line characteristics, pixel size, area texture, and primary/secondary color).

Chapter 6 -- Drawing Instructions

Describes the instructions used for drawing lines and curves.

Chapter 7 -- Filled Figures Instructions

Details those PRO/GIDIS instructions that provide shading for closed figures.

Chapter 8 -- Text Instructions

Describes the instructions that control text subsystem characteristics (current alphabet, character selection for display, unit and display sizes, cell rotation and rendition, and so forth).

Chapter 9 -- Area Operation Instructions

Details those instructions that affect the display area and the instruction for dumping screen contents to the Professional's printing device.

Chapter 10 -- Report Handling

Describes report handling instructions that return state information from the output device.

Appendix A -- Instruction Summary

Lists the PRO/GIDIS instructions, their opcodes, argument list lengths, and associated parameters. Also features an instruction list sorted into opcode order.

Appendix B -- DEC Multinational Character Set

Shows the code table for the Professional's alphabet 0, the DEC Multinational Character Set.

Appendix C -- Glossary

RELATED DOCUMENTATION

Please refer to the other manuals in the Tool Kit Documentation Set for more information on developing applications for the Professional.

CHAPTER 1

INTRODUCTION TO PRO/GIDIS

PRO/GIDIS, the General Image Display Instruction Set (GIDIS), is one of several tools used to develop graphics applications for the Professional 300 Series computer. It consists of a set of instructions that provide the lowest-level, virtual device interface to the Professional's graphics hardware.

1.1 OVERVIEW

PRO/GIDIS is aimed at applications creating "synthetic graphics," those in which images can be described using geometrical entities such as lines, arcs, and shaded areas. PRO/GIDIS can also be used to display mixed text and graphics. Figure 1-1 shows typical PRO/GIDIS output, a graphical representation of some sample statistical data.

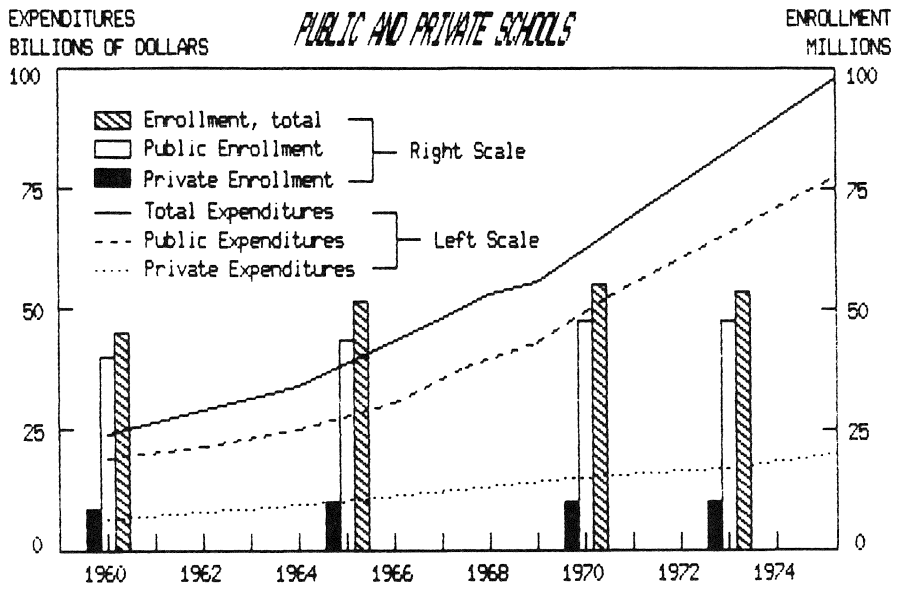


Figure 1-1: PRO/GIDIS Sample Output

PRO/GIDIS is implemented as a layer of P/OS software that receives and interprets binary instructions and executes them on the Professional's graphics hardware. The interpretation of instructions is dependent on the current state of PRO/GIDIS as well as on any explicit parameters that might be supplied with the instruction. Many PRO/GIDIS instructions, like those that control global attributes or the viewing transformation, do not actually cause any drawing to take place, but rather, change PRO/GIDIS's state.

1.2 RELATIONSHIP TO OTHER GRAPHICS TOOLS

The other graphics tools include:

- The Professional Developer's Tool Kit CORE Graphics Library (CGL), a library of high-level graphics subroutines based on the ACM SIGGRAPH CORE Standard.

  NOTE

  Do not use PRO/GIDIS and CGL in the same program. CGL was implemented with PRO/GIDIS and must have exclusive control of it for proper operation.

- ReGIS (Remote Graphics Instruction Set), a Digital-developed, ASCII-based protocol, is used to transmit graphics instructions from a host computer to a remote Professional, or VT125 or GIGI graphics terminal. ReGIS currently cannot be used by applications that reside on the Professional itself; it can only be used when communicating with the Professional over a communications line.

PRO/GIDIS supports CGL and ReGIS and is designed to support Digital's future graphics products. As shown in Figure 1-2, both CGL and ReGIS are implemented as layers above PRO/GIDIS.

1.2.1 WHEN TO USE PRO/GIDIS

- Use PRO/GIDIS if you are already well-versed in graphics programming.

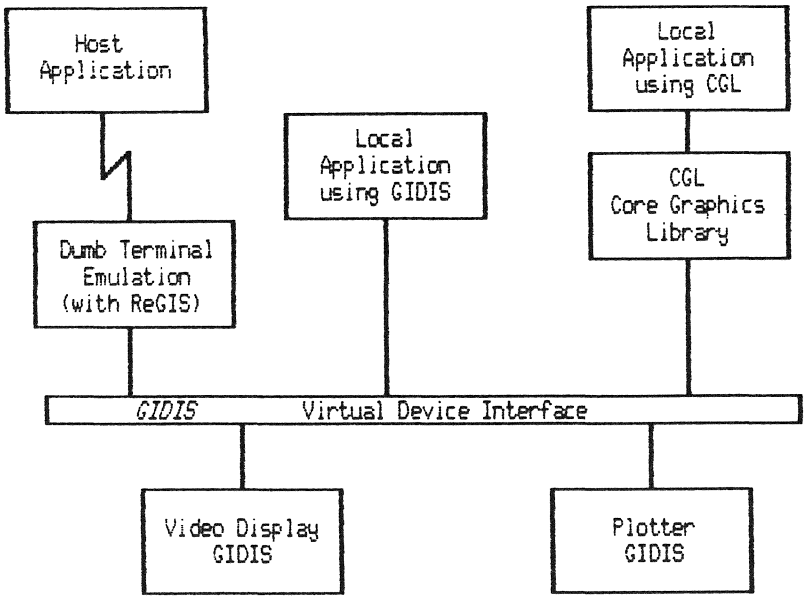


Figure 1-2: PRO/GIDIS Interface, Programs/Video Hardware

- Use PRO/GIDIS for applications that require fast execution speed and that can make use of its level of functionality. Some examples are interactive drawing packages, graphics terminal emulators, and scientific/engineering data display packages.
- Use PRO/GIDIS for applications that require efficient storage of images. Your program can store sequences of binary instructions on disk or in memory and send them to PRO/GIDIS for rapid display.
- Use PRO/GIDIS to implement graphics utility layers, such as CORE and GKS.
- Use PRO/GIDIS to implement application-specific graphics subroutine libraries. This provides flexibility, permits optimization of routines for a particular application, and avoids use of excessive virtual address space.

1.2.2 WHEN NOT TO USE PRO/GIDIS

- If you are not well-versed in graphics programming, consider using the CORE Graphics Library instead of PRO/GIDIS.

- If your program requires support for real (floating point) coordinates, curves, markers, and so forth, use the CORE Graphics Library.
- If you are concerned with portability of programs and programmers, and industry-standard program interfaces to graphics routines, use the CORE Graphics Library.
- If you require VT125 compatibility, use ReGIS with the Professional Terminal Emulator.
- Do not mix graphics protocols. Do not use PRO/GIDIS if you intend to use the CORE Graphics Library and/or direct access to the video hardware.

1.3 THE PRO/GIDIS INSTRUCTION SET

PRO/GIDIS instructions are defined in the form of an mnemonic instruction name and accompanying arguments. Some instructions require no arguments; some require a fixed number of arguments; and some accept a variable number of arguments. Chapter 2 describes how to construct calls to PRO/GIDIS instructions.

The PRO/GIDIS instruction set can be divided into the following functionally related groups. Each group is described in detail in subsequent sections of this chapter.

- **Control Instructions**

  These instructions initialize the PRO/GIDIS interpreter, begin and end pictures, set the cursor, and so forth.

- **Viewing Transformation Instructions**

  The instructions control the PRO/GIDIS address spaces and extents, and the mapping between them.

- **Global Attribute Instructions**

  These instructions set the PRO/GIDIS general state variables that control the appearance of images.

- **Drawing Instructions**

  These instructions draw the actual lines and curves that make up images.

- **Filled Figure Instructions**

  These instructions draw solid figures by shading specific areas.

- **Text Instructions**

  These instructions control alphabets and draw graphics character text.

- **Area Operation Instructions**

  These instruction perform operations such as scrolling and printing on specific areas.

- **Report Instructions**

  These instructions cause PRO/GIDIS to return specific information about the current state.

1.4 CONTROL INSTRUCTIONS

These are the instructions that control the operation of the PRO/GIDIS interpreter.

- INITIALIZE

  The INITIALIZE instruction restores power-on status to one or more graphics subsystems (addressing, global attributes, text, or all subsystems).

- NEW_PICTURE

  The NEW_PICTURE instruction clears the view surface to indicate the start of a new picture.

- END_PICTURE

  The END_PICTURE instruction indicates the end of a group of picture-drawing instructions.

- FLUSH_BUFFERS

  The FLUSH_BUFFERS instruction forces the execution of pending PRO/GIDIS instructions.

- **SET_OUTPUT_CURSOR**

  The SET_OUTPUT_CURSOR instruction specifies the particular character or object to be used as the output cursor (a visible object used to mark the current screen output location).

- **NOP**

  The NOP instruction performs no operation and changes nothing.

- **END_LIST**

  The END_LIST instruction indicates the end of a variable-length argument list.

1.5 THE VIEWING TRANSFORMATION

The graphical world is two-dimensional; we visualize it as a plane. The Cartesian coordinate system provides a convenient way of describing a plane. A coordinate pair specifies a discrete point in the form:

```
X,Y
```

where X is the horizontal axis and Y is the vertical axis.

The finite area of a plane that can be specified by coordinate pairs is called an address space. The origin of an address space is (0,0). Coordinate values increase in magnitude to the right and downward.

PRO/GIDIS deals with three address spaces:

- **GIDIS Output Space (GOS)**

  GIDIS Output Space is the address space referenced by the GIDIS drawing instructions.

- **Imposed Device Space (IDS)**

  Imposed Device Space is the user-defined address space that provides a device-independent means of describing the view surface.

- **Hardware Address Space (HAS)**

  Hardware address space (HAS) is a fixed address space that reflects the characteristics of the particular graphical device being used. PRO/GIDIS does not allow the explict use of HAS coordinates.

PRO/GIDIS maps coordinates between the three address spaces based on built-in and user-supplied information. This double mapping is called the viewing transformation (or "graphics pipeline") and is discussed in Section 1.5.6.

A specific rectangular portion of an address space is called an extent. PRO/GIDIS deals with three user-defined extents:

- **The Window**

  The window is an extent within GIDIS Output Space that maps to the viewport.

- **The Clipping Region**

  The clipping region is an extent within GIDIS Output Space that determines which portion of the view surface is available for drawing.

- **The Viewport**

  The viewport is an extent within Imposed Device Space that maps to a portion of the view surface (HAS).

Normally, you need only be concerned with GIDIS Output Space coordinates. However, you must be aware that the viewing transformation can result in some loss of resolution, and that some mapping relationships will be more optimal than others.

1.5.1 Aspect Ratios

Aspect ratio is a way of representing the shape (the relationship between the horizontal and vertical dimensions) of an area. There are two types of aspect ratio.

- **Unit Aspect Ratio**

  Unit aspect ratio is the ratio of the physical size of one unit in the horizontal direction to one unit in the vertical direction. For example, in an address space whose unit aspect ratio is 1:1, a rectangle whose four sides are exactly one unit in length would appear perfectly square.

The logical address spaces (GIDIS Output Space and Imposed Device Space) have a unit aspect ratio of 1:1. A physical address space (such as Hardware Address Space) has a unit aspect ratio that is dependent on specific hardware.

- **Picture Aspect Ratio**

  Picture aspect ratio is the ratio of an area's width to its height. In order to map extents from one address space to another the picture aspect ratios of the extents must match. Otherwise, only a portion of the lower-level extent is used. This is called isotropic mapping.

  For example, a window defined to be four GOS units by eight GOS units has a picture aspect ratio of 1:2 and would map perfectly to a viewport defined to be nine IDS units by 18 IDS units.

1.5.2 GIDIS Output Space

GIDIS Output Space (GOS) is the device-independent, address space referenced by PRO/GIDIS instructions for the purposes of drawing an image. GOS has a unit aspect ratio of 1:1. GOS is bounded by the set of 16-bit, signed, two's-complement integers. PRO/GIDIS does not check for integer overflow; thus, GOS coordinates, in effect, "wrap around".

NOTE

It is recommended that you restrict the use of GOS coordinates to the range (-16384 to 16384) so that PRO/GIDIS can compute, without risk of overflow, the locations of all points required to draw an image.

1.5.2.1 The Window - The window is the extent within GIDIS Output Space that maps to the viewport in Imposed Device Space. The SET_GIDIS_OUTPUT_SPACE instruction specifies the upper-left corner, width, and height of the window, and thus its resolution and picture aspect ratio.

You can adjust the window to match the data available to your application. It can be any portion of GIDIS Output Space and does not have to include the origin (0,0).

In the viewing transformation, GIDIS isotropically maps the window to the viewport. If the picture aspect ratios of the two extents do not match, GIDIS maps the window onto the uppermost or leftmost portion of the viewport (see Figure 1-3).

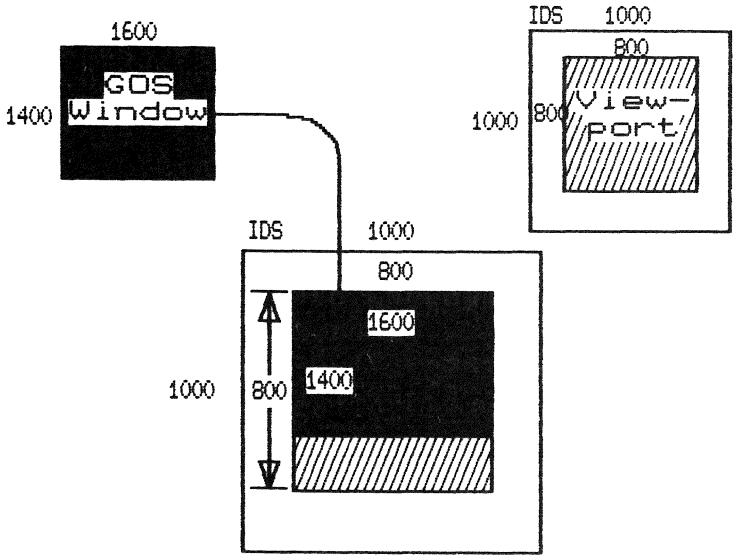


Figure 1-3: Isotropic Mapping: Window to Viewport

1.5.2.2 The Clipping Region - The clipping region is a rectangular area in GIDIS Output Space that determines which portion of the view surface is accessible for drawing. In order for a point to be drawn on the view surface, it must lie within the clipping region. The SET_OUTPUT_CLIPPING_REGION instruction specifies the clipping region, which is bounded only by the limits of the view surface.

By default, the clipping region is the same as the window. The SET_GIDIS_OUTPUT_SPACE instruction sets the clipping region to the newly specified window. That allows the window/clipping region to function as the extent within GOS that is "visible through the viewport." In other words, for some applications, you may find it convenient to think of the window and the clipping region as the same extent.

A clipping region that is different from the window allows you to display portions of an image without changing the viewing transformation (and state variables). Changing the clipping region is more efficient than changing the window, viewport or IDS.

1.5.3 Imposed Device Space

Imposed device space (IDS) is a user-defined logical address space that provides a device-independent way of describing the view surface of any hardware device. The SET_OUTPUT_IDS instruction specifies the size of Imposed Device Space, which has a fixed origin of (0,0). The size is restricted to positive integers in a signed, 16-bit word.

NOTE

It is recommended that you restrict the use of IDS coordinates to the range (-16384 to 16384) so that PRO/GIDIS can compute, without risk of overflow, the locations of all points required to draw an image.

IDS has a unit aspect ratio of 1:1, and its origin always maps to the upper-left corner of the Hardware Address Space. By default, the picture aspect ratio and resolution of IDS match that of the HAS.

In the viewing transformation, GIDIS isotropically maps IDS to the hardware address space such that the specified IDS rectangle uses as much as possible of the HAS while still maintaining the desired picture aspect ratio (see Figure 1-4). Thus, if the picture aspect ratios of the two rectangular areas do not match, only a portion of the view surface is used.

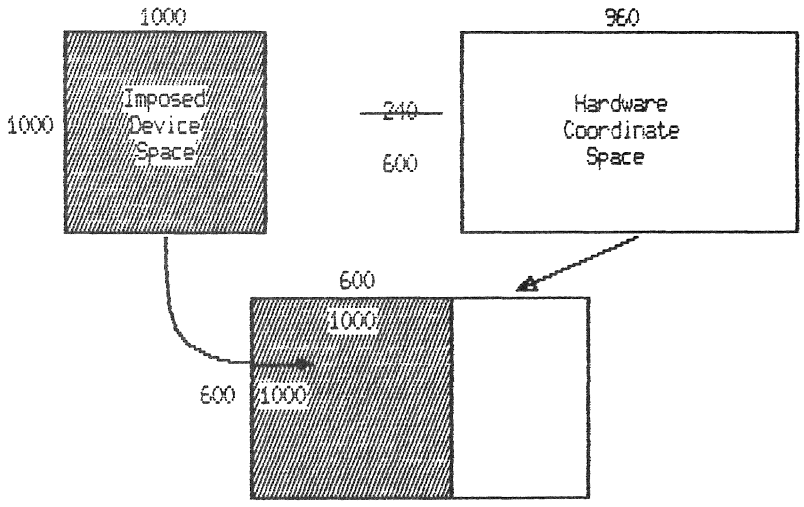


Figure 1-4: Isotropic Mapping: IDS to HAS

1.5.4 The Viewport

The SET_OUTPUT_VIEWPORT instruction specifies a rectangular extent within IDS that corresponds to the window in GIDIS Output Space. The viewing transformation maps the window to the viewport and the viewport to Hardware Address Space. Thus, if the window and clipping region are the same, the viewport specifies the area of the view surface to which drawing is confined.

1.5.5 Hardware Address Space

Hardware address space (HAS) reflects the characteristics of the particular device being used. Although PRO/GIDIS does not allow the explicit use of hardware coordinates, an understanding of them can help you optimize the viewing transformation for your application.

The Professional video monitor's Hardware Address Space is 960 units in the horizontal dimension by 240 units in the vertical dimension (as shown in Figure 1-5). It's unit aspect ratio (width to height) is 2:5 or 1:2.5.

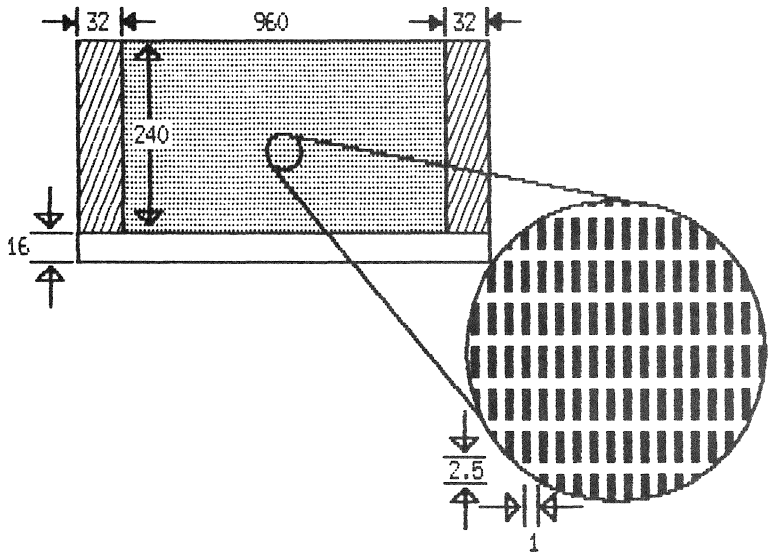


Figure 1-5: Video Hardware Address Space

The two 32-unit bands along the sides are not part of HAS. You can draw in them but to do so will make your application device-dependent because they do not necessarily exist on other devices. The 16-unit band along the bottom is not accessible for drawing.

On a properly adjusted monochrome monitor, HAS occupies a space 7.87 inches (horizontal) by 4.92 inches (vertical). Thus, a rectangle one unit by one unit in HAS coordinates would create an image that is 0.008 inches horizontal (7.87 / 960) by 0.02 inches vertical (4.92 / 240).

On a properly adjusted color monitor, HAS occupies a space 9.45 inches (horizontal) by 5.91 inches (vertical). Thus, a rectangle one unit by one unit in HAS coordinates would create an image that is 0.01 inches horizontal (9.45 / 960) by 0.025 inches vertical (5.91 / 240).

1.5.6 The Viewing Transformation

The process of creating an image on a view surface can be thought of as a two-step process, as shown in Figure 1-6.

1. GIDIS maps GOS coordinates to IDS coordinates using the window and viewport.

2. GIDIS maps IDS coordinates to HAS coordinates using the given size of IDS and the known size of HAS.

Thus, by establishing the window and viewport, you also establish the relationship of all GOS and IDS coordinates, including those outside the window/viewport.

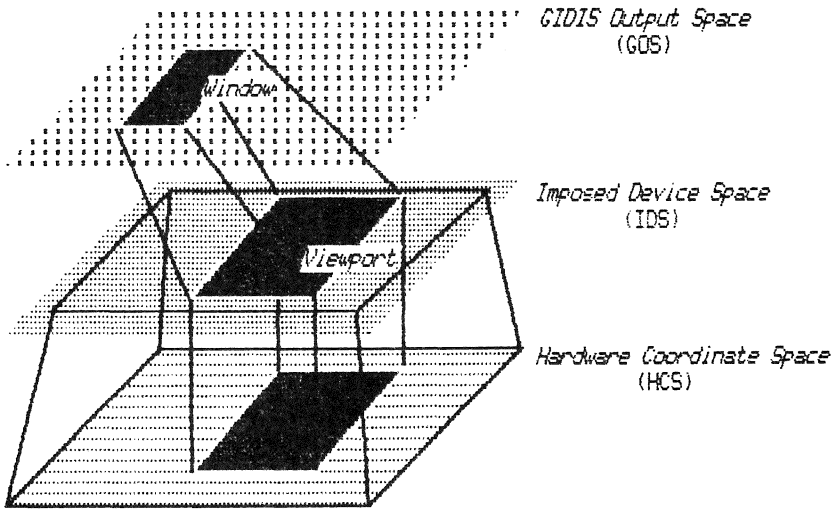


Figure 1-6: The Viewing Transformation

In effect, you change window parameters in order to "view" a different portion and/or scaling of Gidis Output Space. You change viewport parameters in order to "view" the same portion of GOS in a different portion of the view surface.

Viewing transformation parameters only affect subsequently received GIDIS instructions. Changing the parameters does not affect images already visible on the view surface. In order to do that, you must change the mapping parameters and redraw the image.

1.5.7 Addressing Pixels

Some applications, such as a natural image display, must address individual pixels directly. To accomplish this, set IDS space to width 1920 and height 1200. The picture aspect ratio is 8:5, a ratio that maps to the entire screen. Pixel X (from the left) on row Y is IDS coordinate [X*2, Y*5].

There is a pixel at every X coordinate divisible by two and Y coordinate divisible by five. For example, [100,20], [22,15], and [78,505] all map directly to a pixel. PRO/GIDIS automatically truncates coordinates that map partway between pixels. For example, [101,23], [23,16], and [78,509] map to the same pixels as the first list of coordinates.

NOTE

> An IDS of 960 by 240 does not accomplish the same thing. It would have a picture aspect ratio of 4:1 (the unit aspect ratio is 1:1) which would not match the picture aspect ratio of HAS (8:5).

1.6 DRAWING INSTRUCTIONS

Part of the PRO/GIDIS state is a coordinate pair called the current position that corresponds to the current drawing location in GIDIS Output Space. The visual representation of the current position is the cursor. Thus, some of these instructions do not draw anything; they simply change the current position.

- **SET_POSITION**

  The SET_POSITION instruction specifies the new current position as an absolute location in GIDIS Output Space.

- **SET_REL_POSITION**

  The SET_REL_POSITION instruction specifies the new current position as a point relative to (an offset from) the old current position.

- **DRAW_LINES**

  The DRAW_LINES instruction draws one or more straight lines starting at the current position.

- **DRAW_REL_LINES**

  Draws one or more straight lines starting at the current position. Coordinates specified are relative to the current position or the previous point.

- **DRAW_ARC**

  Draws a section of a circle using the current position as a reference.

- **DRAW_REL_ARC**

  Draws a section of a circle using an offset from the current position as a reference.

1.7 FILLED FIGURE INSTRUCTIONS

A filled-figure is a closed, shaded figure that can be bordered by either straight lines, circular arcs, or any combination of these. The instructions that begin and end filled figures are:

- **BEGIN_FILLED_FIGURE**

  The BEGIN_FILLED_FIGURE instruction starts a filled figure definition.

- **END_FILLED_FIGURE**

  The END_FILLED_FIGURE instruction ends a filled figure definition and causes the entire figure to be filled in with the current area texture.

1.8 TEXT INSTRUCTIONS

PRO/GIDIS text is independent from the Professional terminal subsystem's text mode. The instruction that draws text is:

- DRAW_CHARACTERS

  The DRAW_CHARACTERS instruction displays each of the characters specified by each character index in the parameter list. The characters are taken from the currently selected alphabet.

1.9 COLOR ATTRIBUTES

The Professional's video subsystem includes an internal bitmap (also known as a raster image or frame buffer). The bitmap consists of one or more two-dimensional matrices of bits called planes. Each bitmap position controls a pixel, which represents a physical hardware coordinate position.

The basic Professional bitmap has a single plane, providing one bit for each pixel. Because a single bit can be in one of two possible binary states: 0 or 1, only dark and light images are possible. A value of 0 specifies dark and 1 specifies light. The actual colors depend on the phosphors used in the monitor.

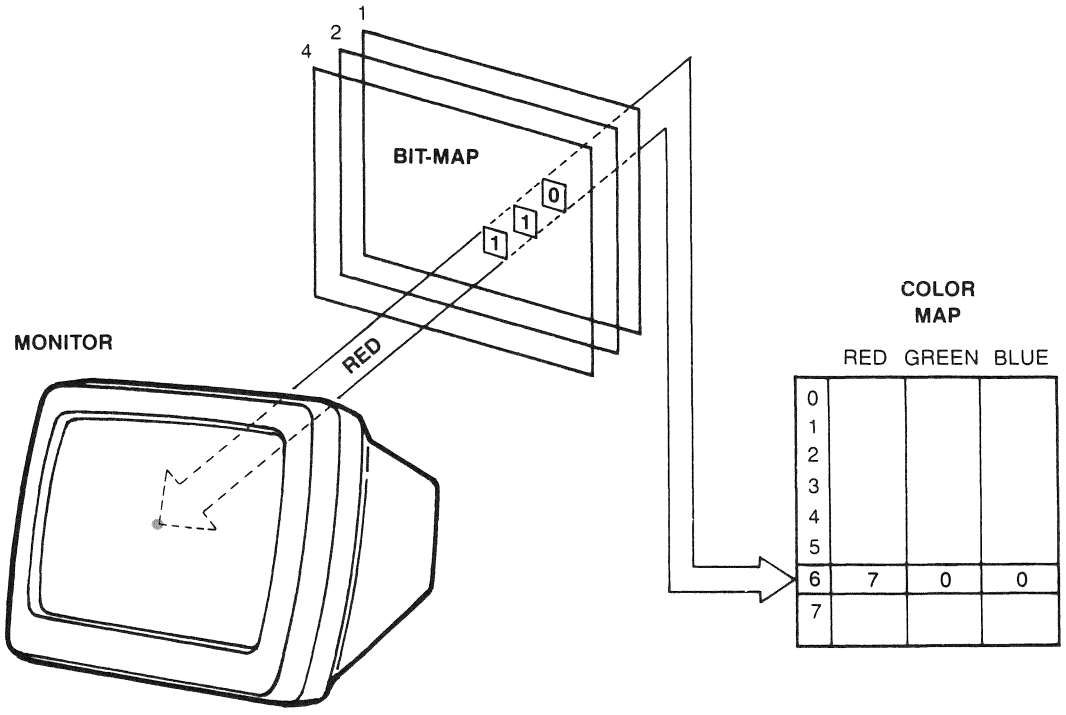


Figure 1-7: The Extended Bitmap Option

When equipped with the Extended Bitmap Option (EBO) board, the Professional's bitmap has three planes. The SET_PLANE_MASK instruction specifies which planes are accessible to PRO/GIDIS and which are "write-locked."

Three planes provide three bits per position. Thus each position can contain a binary number in the range 000 to 111 (0 to 7 decimal). This number is an index into an internal table called the color map, which contains descriptions of colors.

The color map consists of eight entries, numbered 0 to 7, as shown in Figure 1-7. Each color map entry consists of three fields, red, green, and blue, that contain intensity values for each primary color in light. The SET_COLOR_MAP_ENTRY instruction allows you to specify these red, green, and blue values by proportion.

By changing the contents of the color map, you can form 256 different colors. If no color monitor is present, you can use the color map to form eight shades of display intensity.

NOTE

When you change the contents of a color map entry, you instantly change the color displayed for all pixels that were drawn with that entry.

The red and green fields each consist of three bits and thus have eight possible values. The blue field has only two bits and thus has four possible values. The human eye is less sensitive to changes in blue than either of the other primaries.

When drawing an image, PRO/GIDIS places a color map index into each pixel required by the image. The actual color map index placed in a particular bitmap position depends on the previous contents of that pixel, the current pattern, and the writing mode (more information on this in the following sections.)

The PRO/GIDIS state includes two color map index values: the primary color and the secondary color, specified by SET_PRIMARY_COLOR and SET_SECONDARY_COLOR respectively. In general, the primary color indicates the presence of an image (the foreground) and the secondary color indicates the absence of an image (the background).

1.10 THE WRITING MODE

PRO/GIDIS characters, line textures and area textures each form a binary pattern which, for the purpose of describing the writing

modes, we will call the "current pattern." The exact way in which PRO/GIDIS uses the current pattern to create images depends on the writing mode and the primary and secondary colors.

Figure 1-8 shows the same line texture (which includes ON and OFF pixels) drawn over light and dark areas in all eight visible writing modes.

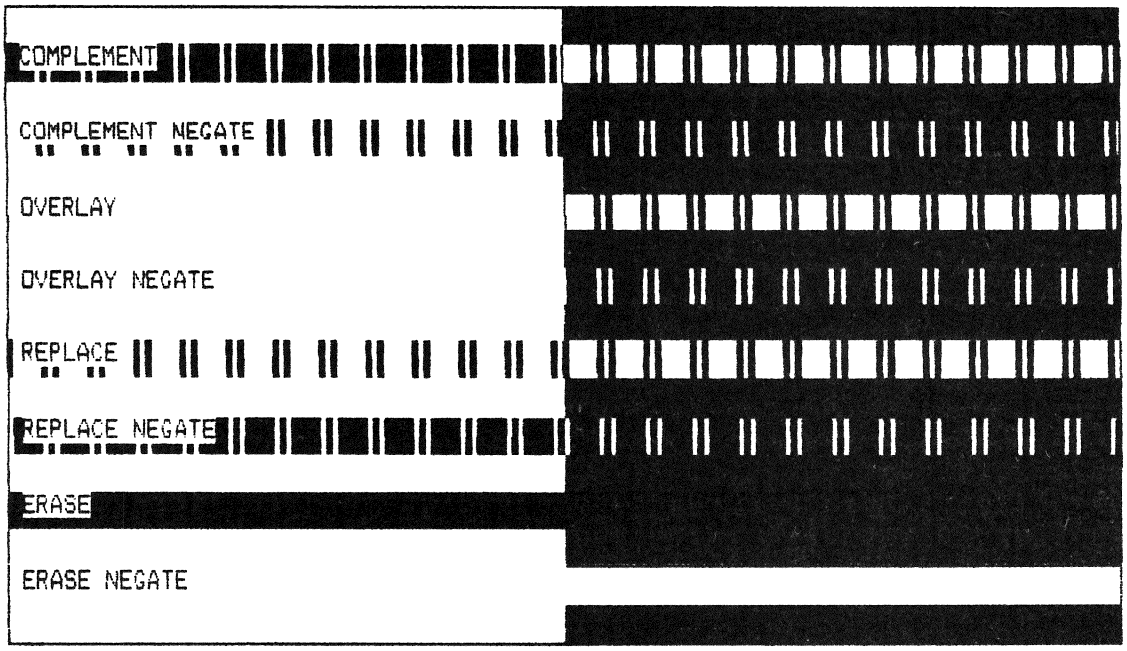


Figure 1-8: The Writing Modes Shown with Line Texture

The writing modes are:

- TRANSPARENT

  In transparent mode, no actual drawing is done. Otherwise, all other drawing processes are exercised and the state information (particularly, the current position) is updated. The texture is ignored. Transparent mode for determining the current position after drawing without actually drawing the image.

- TRANSPARENT NEGATE

  This mode is identical to transparent mode.

- COMPLEMENT

  In complement mode, wherever the current pattern contains set (1) bits, PRO/GIDIS performs a bitwise inversion of the pixel

value (color index). For example, a pixel containing 101 (color map entry 5) changes to 010 (color map entry 2). Wherever the current pattern contains clear (0) bits, no modifications are made.

The purpose of complement mode is to make images "stand out" from whatever has already been drawn. If the appropriate color map entries contain complementary colors, the image in the display area is effectively reversed. The original image can be restored by repeating the process.

Complement mode is likely to produce seams when filled figures overlap. Since filled figures include their borders, areas with a common border are considered to overlap.

- **COMPLEMENT NEGATE**

  Complement negate mode is identical to complement mode except that PRO/GIDIS negates the current pattern. Wherever the current pattern contains clear (0) bits, PRO/GIDIS performs a bitwise inversion of the pixel value (color index). Wherever the current pattern contains set (1) bits, no modifications are made.

- **OVERLAY**

  In overlay mode, wherever the current pattern contains set (1) bits, PRO/GIDIS draws in the current primary color. Wherever the current pattern contains clear (0) bits, no drawing occurs.

  The purpose of overlay mode is to draw images "on top of" whatever is already on the view surface.

- **OVERLAY NEGATE**

  Overlay negate mode is identical to overlay mode except that PRO/GIDIS negates the current pattern. Wherever the current pattern contains clear (0) bits, PRO/GIDIS draws in the current primary color. Wherever the current pattern contains set (1) bits, no drawing occurs.

- **REPLACE**

  In replace mode, wherever the current pattern contains set (1) bits, PRO/GIDIS draws in the primary color. Wherever the current pattern contains clear (0) bits, PRO/GIDIS draws in the secondary color.

  The purpose of replace mode is to draw images that completely replace whatever is already on the view surface.

- **REPLACE NEGATE**

  Replace negate mode is identical to replace mode except that PRO/GIDIS negates the current pattern. Wherever the current pattern contains clear (0) bits, PRO/GIDIS draws in the primary color. Wherever the current pattern contains set (1) bits, PRO/GIDIS draws in the secondary color.

- **ERASE**

  In erase mode, PRO/GIDIS sets all pixels to the secondary color. The purpose of erase mode is to draw images by erasing what is already on the view surface.

- **ERASE NEGATE**

  Erase negate mode is identical to erase mode except that PRO/GIDIS sets all pixels to the primary color.

1.11 LINE AND CURVE ATTRIBUTES

The following sections describe the PRO/GIDIS state variables that determine the appearance of lines and curves.

1.11.1 Line Texture

The SET_LINE_TEXTURE instruction specifies a linear bit pattern used for drawing lines and arcs. The line texture is used in conjunction with the writing mode and the primary and secondary colors to determine the value for all pixels in a line or arc. The line texture is wrapped around corners to provide a smoothly textured pattern.

For example, the character "X" represents pixels drawn in the primary color and "-" represents pixels drawn in the secondary color. The pattern 0100110 produces a line (drawn from left to right) that looks like:

```
- X - - X X -
```

Although you specify the size of a single repetition of the line texture in GIDIS Output Space coordinates, that size is only an approximation. PRO/GIDIS actually uses the largest integral multiple of the pattern length that is less than or equal to the specified size.

For example, assume that the line above has a size of seven and was drawn exactly once in seven pixels. Changing the size to 14 uses each bit twice in succession:

```
- - X X - - - - X X X X - -
```

When you use a large drawing pixel, a number of pixels are written multiple times, potentially with both primary and secondary colors in either order (see the SET_PIXEL_SIZE instruction.) The same example with a pixel size of two by two (pattern size = 14 pixels) would produce:

```
- -- -X XX X- -- -- -- -X XX XX XX X- -- -
- -- -X XX X- -- -- -- -X XX XX XX X- -- -
```

where the symbols mean:

-	Secondary color written once
--	Secondary color written twice
-X	Secondary color written once, then primary once
X-	Primary color written once, then secondary once
XX	Primary color written twice

In replace mode, "-X" produces "X", "X-" produces "-", and so forth, so the final result would be:

```
- - X X - - - - X X X X - - -
- - X X - - - - X X X X - - -
```

In complement mode, "-" does nothing and "X" complements the pattern. Thus "--" does not change the pixel, "-X" and "X-" complements it once, and "XX" complements it twice (returning to the original state). The final result would be:

```
- - X - X - - - X - - - X - -
- - X - X - - - X - - - X - -
```

1.11.2 Pixel Size

The SET_PIXEL_SIZE instruction specifies the size of the logical drawing pixel, the bit pattern used to draw lines and arcs. Increasing the size of the logical drawing pixel widens the lines used to draw images. The default logical drawing pixel is one physical pixel. Figure 1-9 shows several different logical drawing pixels and lines.

1.12 FILLED FIGURE ATTRIBUTES

The following sections describe the PRO/GIDIS state variables that determine the appearance of filled figures.

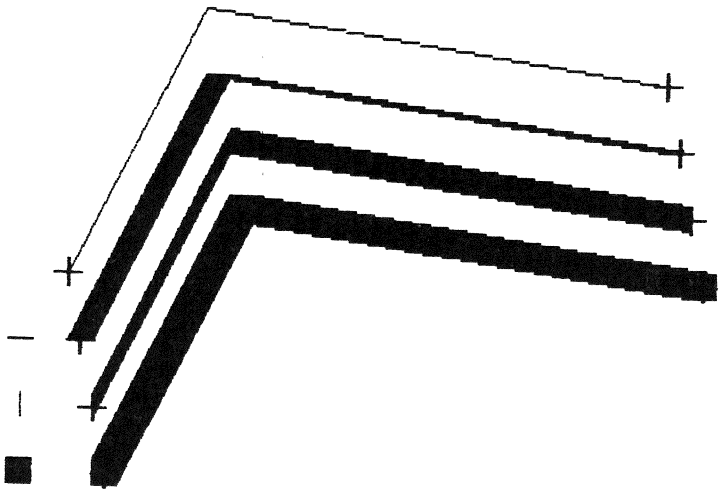


Figure 1-9: The Logical Drawing Pixel

1.12.1 Area Texture

The SET_AREA_TEXTURE instruction specifies a two-dimensional bit pattern that defines the appearance of filled figures. Area texture is specified as a character in an alphabet.

Any arbitrary pattern (up to the maximum character size) can be used for area texture. For example, given alphabet 2, character 23, (the pattern shown here):

```
0   0   0   0
0   1   0   1
0   1   0   0
0   1   0   1
```

With area texture given as alphabet 2, character 23, and the appropriate sizes specified, the pattern drawn is:

```
- - - - - - - - - - - - -
- - - - - - - - - - - - -
- - - X X X - - - X X X
- - - X X X - - - X X X
- - - X X X - - - - - -
- - - X X X - - - - - -
- - - X X X - - - X X X
- - - X X X - - - X X X
```

where the symbols mean:

- \- Secondary color
- X Primary color

Area textures are aligned with a fixed point on the screen so that no seams show if two areas overlap. (Assuming they are drawn with the same pattern, pattern sizes, writing mode, and colors.) The size of area texture cells is defined by the SET_AREA_CELL_SIZE instruction.

1.12.2 Area Texture Size

The SET_AREA_TEXTURE_SIZE instruction specifies the actual size of the area texture character. Although you specify area texture size GIDIS Output Space coordinates, that size is only an approximation. PRO/GIDIS actually uses the largest integral multiple of the texture pattern that is less than or equal to the specified size.

1.12.3 Area Texture Cell Size

The SET_AREA_CELL_SIZE instruction allows you to specify the size of the area texture cell, the source for the area texture. Generally, area texture cell size matches the cell size loaded from a character cell in an alphabet. For those occasions when the loaded cell size is inappropriate for the area texture, this instruction permits clipping the area cell after it is loaded from the character cell.

1.12.4 Shading to a Line or Point

PRO/GIDIS can be used to emulate shading to a line or a point using the filled figure instructions. Your program must calculate the area to be filled and issue the appropriate instructions. One way is to buffer points and generate one fill-area sequence for a number of points. Another is to determine the incremental area to be filled for each point and send a fill area sequence for each point. Because the area includes the border, unusual results (seams) can occur in complement mode when areas are adjacent or overlapping.

1.12.5 Filled Figure Examples

The format for these example programs in described in Chapter 2.

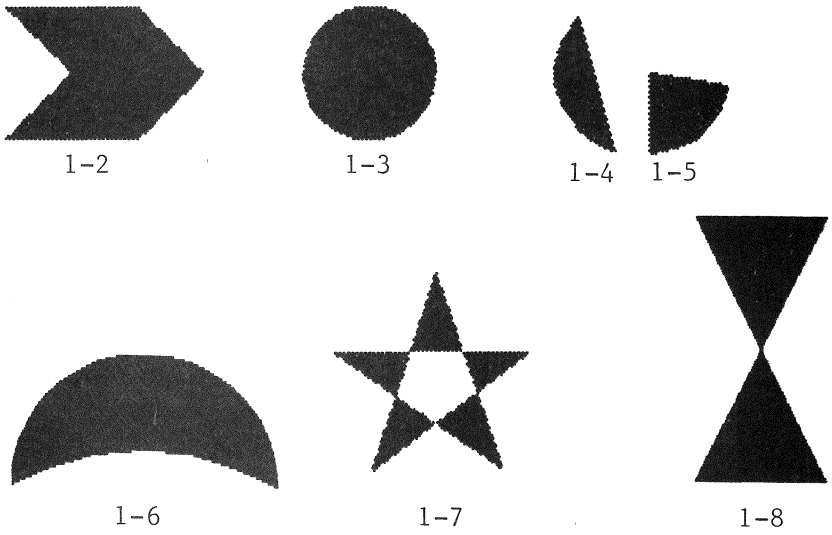


Figure 1-10: Filled Figure Images

```
.BYTE   2.,10.          ; length = 2, opcode = SET_OUTPUT_IDS
.WORD   7680.,4800.     ; width, height
.BYTE   4.,13.          ; length = 4, opcode = SET_OUTPUT_VIEWPORT
.WORD   0.,0.
.WORD   7680.,4800.
.BYTE   4.,9.           ; length = 4, opcode = SET_GIDIS_OUTPUT_SPACE
.WORD   0.,0.
.WORD   7680.,4800.
.BYTE   3.,17.          ; length = 3, opcode = SET_LINE_TEXTURE
.WORD   16.             ; length
.WORD   -1.             ; pattern
.WORD   100.            ; size
.BYTE   2.,14.          ; length = 2, opcode = SET_AREA_TEXTURE
.WORD   -1.             ; special 'alphabet'
.WORD   0.              ; use line texture for area texture
```

Example 1-1: Context for Filled Figure Examples

```
.BYTE   2.,29.          ; length = 2, opcode = SET_POSITION
.WORD   340.,270.
.BYTE   0.,31.          ; length = 0, opcode = BEGIN_FILLED_FIGURE
.BYTE   255.,25.        ; END_LIST terminated, opcode = DRAW_LINES
.WORD     1340,270,
.WORD     1840.,770.
.WORD     1340.,1270.
.WORD      340.,1270.
.WORD      840.,770.
.WORD      340.,270.
.WORD   -32768.
.BYTE   0.,32.          ; length = 0, opcode = END_FILLED_FIGURE
```

Example 1-2: A Filled Figure Using DRAW_LINES

```
.BYTE   2.,29.          ; length = 2, opcode = SET_POSITION
.WORD   3090.,270.
.BYTE   0.,31.          ; length = 0, opcode = BEGIN_FILLED_FIGURE
.BYTE   3.,23.          ; length = 3, opcode = DRAW_ARC
.WORD   3090.,770.
.WORD   360.
.BYTE   0.,32.          ; length = 0, opcode = END_FILLED_FIGURE
```

Example 1-3: A Filled Figure Using DRAW_ARC

```
.BYTE   2.,29.          ; length = 2, opcode = SET_POSITION
.WORD   4660.,340.
.BYTE   0.,31.          ; length = 0, opcode = BEGIN_FILLED_FIGURE
.BYTE   3.,23.          ; length = 0, opcode = DRAW_ARC
.WORD   5090.,770.
.WORD   120.
.BYTE   0.,32.          ; length = 0, opcode = END_FILLED_FIGURE
```

Example 1-4: A Filled Figure Using DRAW_ARC

```
.BYTE   2.,29.          ; length = 2, opcode = SET_POSITION
.WORD   5190.,138.
.BYTE   0.,31.          ; length = 0, opcode = BEGIN_FILLED_FIGURE
.BYTE   3.,27.          ; length = 3, opcode = DRAW_REL_ARC
.WORD   0.,-610.,
.WORD   80.
.BYTE   2.,25.          ; length = 2, opcode = DRAW_LINES
.WORD   5190.,770.
.BYTE   0.,32.          ; length = 0, opcode = END_FILLED_FIGURE
```

Example 1-5: A Filled Figure Using DRAW_REL_ARC

```
.BYTE   2.,29.          ; length = 2, opcode = SET_POSITION
.WORD   7000.,3790.
.BYTE   0.,31.          ; length = 0, opcode = BEGIN_FILLED_FIGURE
.BYTE   3.,23.          ; length = 3, opcode = DRAW_REL_ARC
.WORD   6000.,3800.,180.
.BYTE   3.,23.          ; length = 3, opcode = DRAW_REL_ARC
.WORD   6000.,5532.,-60.
.BYTE   0.,32.          ; length = 0, opcode = END_FILLED_FIGURE
```

Example 1-6: A Filled Figure Using DRAW_REL_ARC

```
.BYTE   2.,29.          ; length = 2, opcode = SET_POSITION
.WORD   890.,3770.
.BYTE   0.,31.          ; length = 0, opcode = BEGIN_FILLED_FIGURE
.BYTE   8.,25.          ; length = 8, opcode = DRAW_LINES
.WORD     1890.,3770.
.WORD      890.,1770.
.WORD     1890.,1770.
.WORD      890.,3770.
.BYTE   0.,32.          ; length = 0, opcode = END_FILLED_FIGURE
```

Example 1-7: A Filled Figure Using DRAW_LINES

```
.BYTE   2.,29.          ; length = 2, opcode = SET_POSITION
.WORD   3090.,3770.
.BYTE   0.,31.          ; length = 0, opcode = BEGIN_FILLED_FIGURE
.BYTE   255.,25.        ; END_LIST terminated, opcode = DRAW_LINES
.WORD     3590.,2270.
.WORD     4090.,3770.
.WORD     2790.,2870.
.WORD     4290.,2870.
.WORD     3090.,3770.
.WORD   -32768.
.BYTE   0.,32.          ; length = 0, opcode = END_FILLED_FIGURE
```

Example 1-8: A Filled Figure Using DRAW_LINES

1.13 TEXT ATTRIBUTES

The following sections describe the PRO/GIDIS state variables that determine the appearance of text.

1.13.1 Alphabets

The SET_ALPHABET instruction specifies the current alphabet, which is used when drawing characters, defining a new character, or erasing an existing alphabet in preparation for establishing a new alphabet. PRO/GIDIS supports up to 16 alphabets.

Each alphabet is a list of characters. Alphabet 0 is the DEC Multinational Character Set (shown in Figure 1-11) and cannot be modified. Alphabets 1 through 15 can contain user-defined characters, which are represented as a two-dimensional raster (bit pattern) with a specific storage size (width and height).

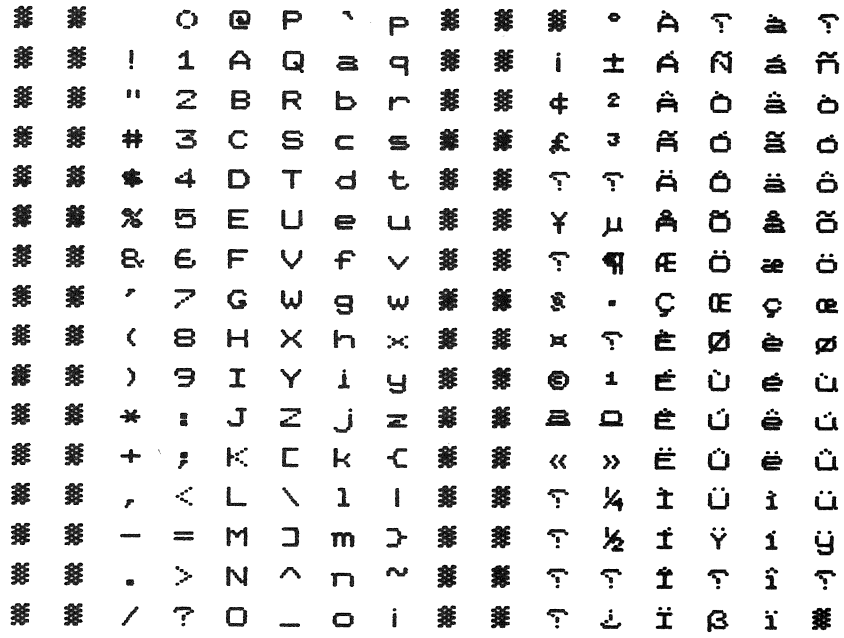


Figure 1-11: Alphabet 0

Characters within alphabets are referenced as a tuple (alphabet, character index). A character index is a value corresponding to the character's position in the alphabet. The first character has a character index of zero. Thus, an alphabet with 26 characters would have index numbers in the range 0 to 25. Character indexes are unsigned, 16-bit integers.

To create a new alphabet of user-defined characters:

1. Use the SET_ALPHABET instruction to specify the current alphabet.

2. Use the CREATE_ALPHABET instruction to clear any existing characters and to allocate storage for the new alphabet. No character definition can take place until CREATE_ALPHABET executes.

3. Use the LOAD_CHARACTER_CELL instruction to add characters to the alphabet.

For example (format described in Chapter 2):

```
.BYTE   1.,38.  ;length = 2, opcode = SET_ALPHABET
.WORD   1.

.BYTE   4.,46.  ;length = 4, opcode = CREATE_ALPHABET
.WORD   8.      ;width (storage pixels)
.WORD   10.     ;height (storage pixels)
.WORD   96.     ;extent
.WORD   0.      ;width-type
```

These instructions erase alphabet 1 and create a new alphabet of 96 characters (numbered 0 through 95) with a storage size of 8 by 10. The last parameter value (0) is required for PRO/GIDIS on P/OS 1.7.

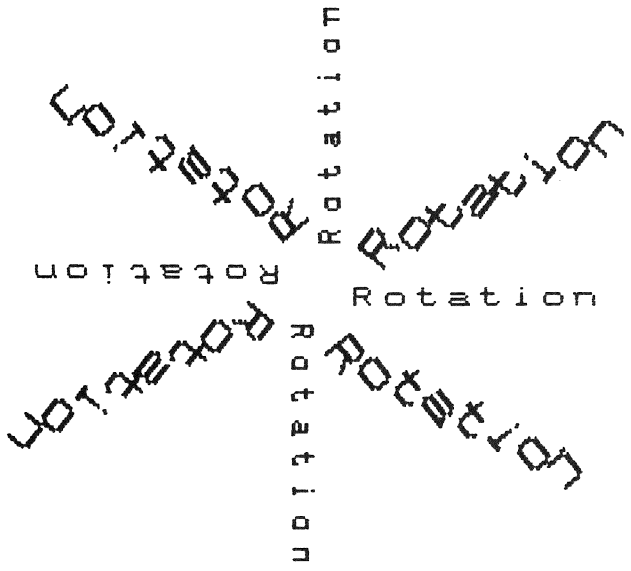


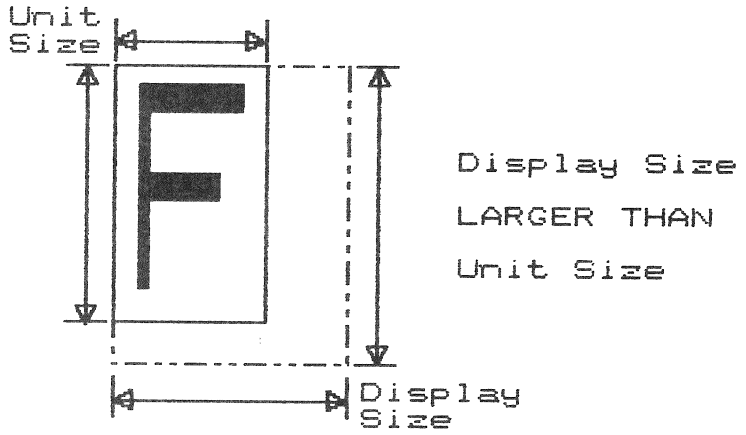
Figure 1-12: Character Cell Rotation

1.13.2 Cell Rendition

The SET_CELL_RENDITION instruction specifies variations on characters that can be performed without selecting a new alphabet, yet are not related to writing colors or writing modes. The renditions defined for the Professional are backslant and italics.

1.13.3 Cell Rotation

The SET_CELL_ROTATION instruction specifies the angle at which characters are to be drawn, as shown in Figure 1-12. Actual rotation is the nearest possible angle to that requested.



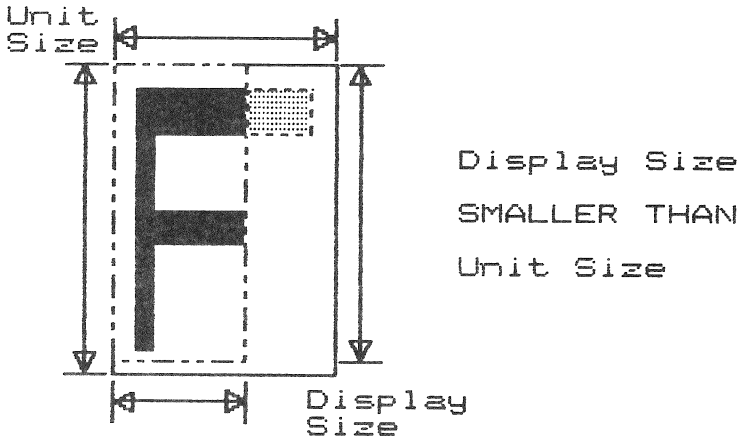


Figure 1-13: Character Unit Cell and Display Cell

1.13.4 Cell Oblique

The SET_CELL_OBLIQUE instruction specifies the angle between the width and the height of the display cell for a non-rotated character. When the angle is non-zero, display cells are parallelograms, rather than rectangles.

1.13.5 Cell Unit Size

The SET_CELL_UNIT_SIZE instruction specifies the actual size of the character pattern. Although you specify cell unit size in GIDIS Output Space coordinates, that size is only an approximation. PRO/GIDIS actually uses the largest integral multiple of the character pattern that is less than or equal to the specified size.

1.13.6 Cell Display Size

The SET_CELL_DISPLAY_SIZE instruction specifies the size in GIDIS Output Space coordinates of the rectangle that contains the character pattern.

The unit cell and the display cell always are aligned at their upper left corners (see Figure 1-13). If the unit cell is larger than display cell, only a portion of the character is shown. If the unit cell is smaller than the display cell, PRO/GIDIS draws the unused portion of the display cell as if the pattern specified clear bits.

1.13.7 Cell Movement

Cell movement attributes specify how the current position changes after each character is drawn. The SET_CELL_MOVEMENT_MODE and the SET_CELL_EXPLICIT_MOVEMENT instructions specify these attributes.

1.13.7.1 Movement mode - There are two ways to specify cell movement: implied and explicit, as shown in Figure 1-14.

- Implied movement means that the current position moves a distance equal to the display cell width in the direction of the cell rotation. If the display cell width value is negative, the current position moves in the direction opposite to the cell rotation.

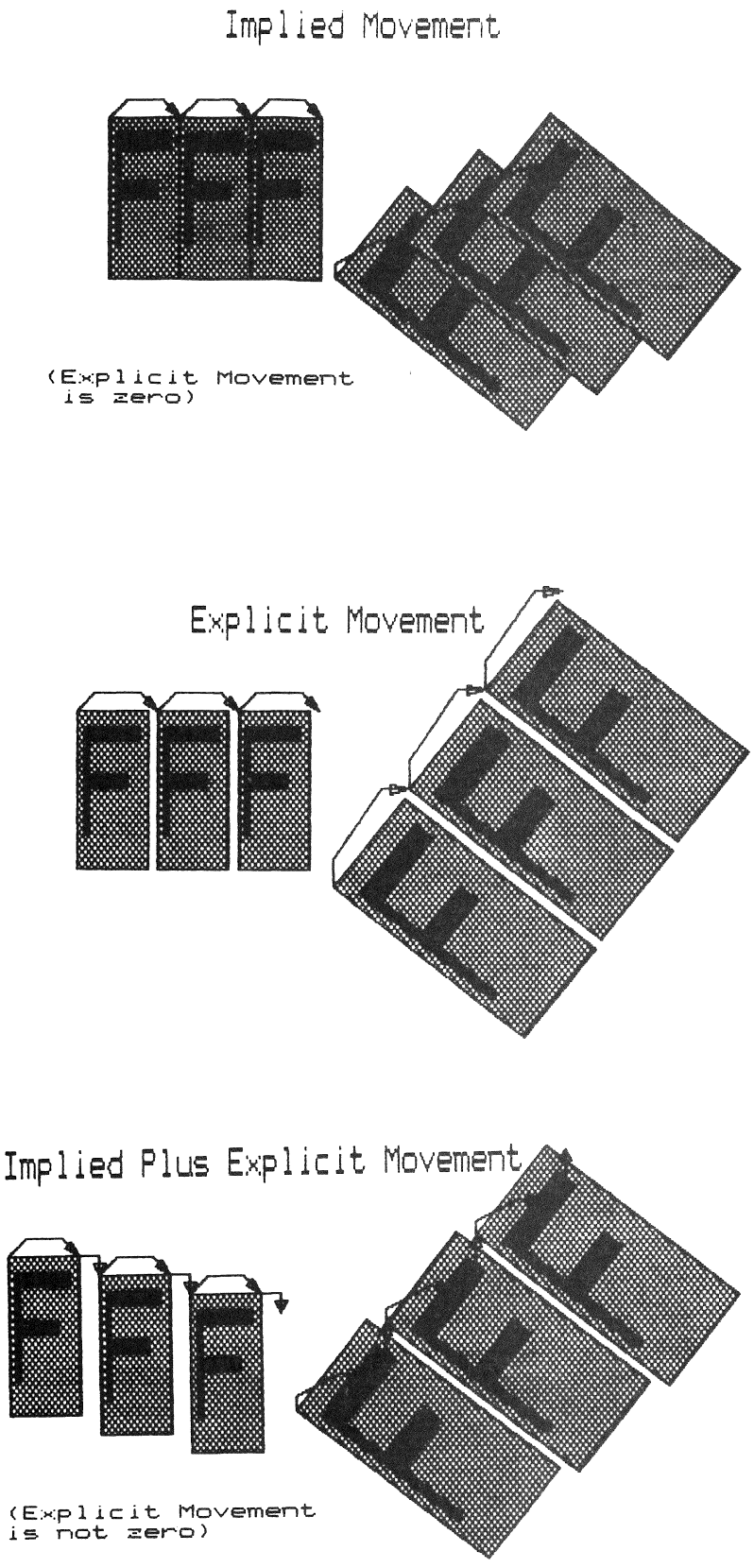


Figure 1-14: Character Cell Movement

- Explicit movement means that the current position moves exactly as specified by your program in GIDIS Output Space coordinates. Explicit movement is not affected by the cell rotation.

Implied movement can be disabled; explicit movement cannot. Thus, if implied movement is enabled, the cell movement is equal to the sum of the implied and explicit movements.

PRO/GIDIS for P/OS V1.7 always uses local symmetry for drawing text and cell movement. Local symmetry preserves sizes and distances in the Hardware Address Space to ensure that all cells and spacings are the same. It also implies that the realized (actual) angle at which the character is drawn is used for implied movement.

1.14 AREA OPERATIONS INSTRUCTIONS

These instructions perform operations on areas defined by the viewing transformation instructions.

- **ERASE_CLIPPING_REGION**

  Changes entire output clipping region to current secondary color.

- **PRINT_SCREEN**

  Prints a portion of the bitmap at the printer connected to the printer port.

1.15 REPORT HANDLING INSTRUCTIONS

Report handling instructions return information about the current PRO/GIDIS state as well as success/failure reports for the immediately preceding PRO/GIDIS instructions.

The report path from the PRO/GIDIS interpreter to your program can be viewed as a data stream. It is possible to queue several pending reports. (P/OS 1.7 imposes a buffer limit of 18 words.)

Your program sends a request instruction to the interpreter, which creates a report and puts it in the queue. Your program then reads the report queue (with a Read Special Data QIO system directive) in the order in which the requests are made. For an example of a report-reading routine, refer to Chapter 2.

A report is a variable length block of words. The first word is a tag specifying the type of report and the number of words in the report. Your program must keep in synchronization with the report queue so that it is not reading a data word and interpreting it as a tag word.

Your program also can set up an asynchronous system trap (AST) to be executed when a report is placed in the report queue.

The following instructions request reports:

- **REQUEST_CURRENT_POSITION**

  PRO/GIDIS reports the X and Y coordinates of the current position.

- **REQUEST_STATUS**

  PRO/GIDIS returns a success or failure code for the last instruction executed.

- **REQUEST_CELL_STANDARD**

  PRO/GIDIS returns the standard character parameters (unit width, unit height, display cell width, and display cell height) for the current alphabet at the current rotation angle.

CHAPTER 2

INTERACTING WITH THE PRO/GIDIS INTERPRETER

The P/OS Terminal Driver provides software access to PRO/GIDIS via the Queue I/O Request (QIO) and Queue I/O Request and Wait (QIOW) system directives. This chapter contains descriptions of the directive formats and PRO/GIDIS instruction syntax rules. QIO error messages are listed at the end of each description.

Figure 2-1 depicts the instruction and parameter data path between your program and PRO/GIDIS.

You can use PRO/GIDIS from MACRO-11 or any supported Tool Kit high-level language that supports external MACRO-11 routines. The recommended method is to write callable MACRO-11 routines that issue QIO and QIOW directives. Tool Kit FORTRAN-77 provides its own callable QIO and WTQIO routines in SYSLIB.

For information on calling a MACRO-11 routine from one of the Tool Kit high-level languages, refer to the documentation for your programming language.

2.1 THE PRO/GIDIS INTERFACE

PRO/GIDIS instructions are sent to the graphics device with a QIO system directive that specifies the Write Special Data (IO.WSD) I/O function code. For low-overhead, high-speed, device interaction, a number of PRO/GIDIS instructions can be passed to the graphics device at one time. Status information returns with the Read Special Data (IO.RSD) I/O function call. P/OS transfers the instruction data to and from the graphics device according to the request priority and device availability.

Programs that use PRO/GIDIS also can use the Professional VT102 terminal emulator. Normal QIO directives (IO.WLB, IO.WVB, and so forth) are passed to the VT102 emulator. For more information, refer to the description of the Terminal Driver in the P/OS System Reference Manual.

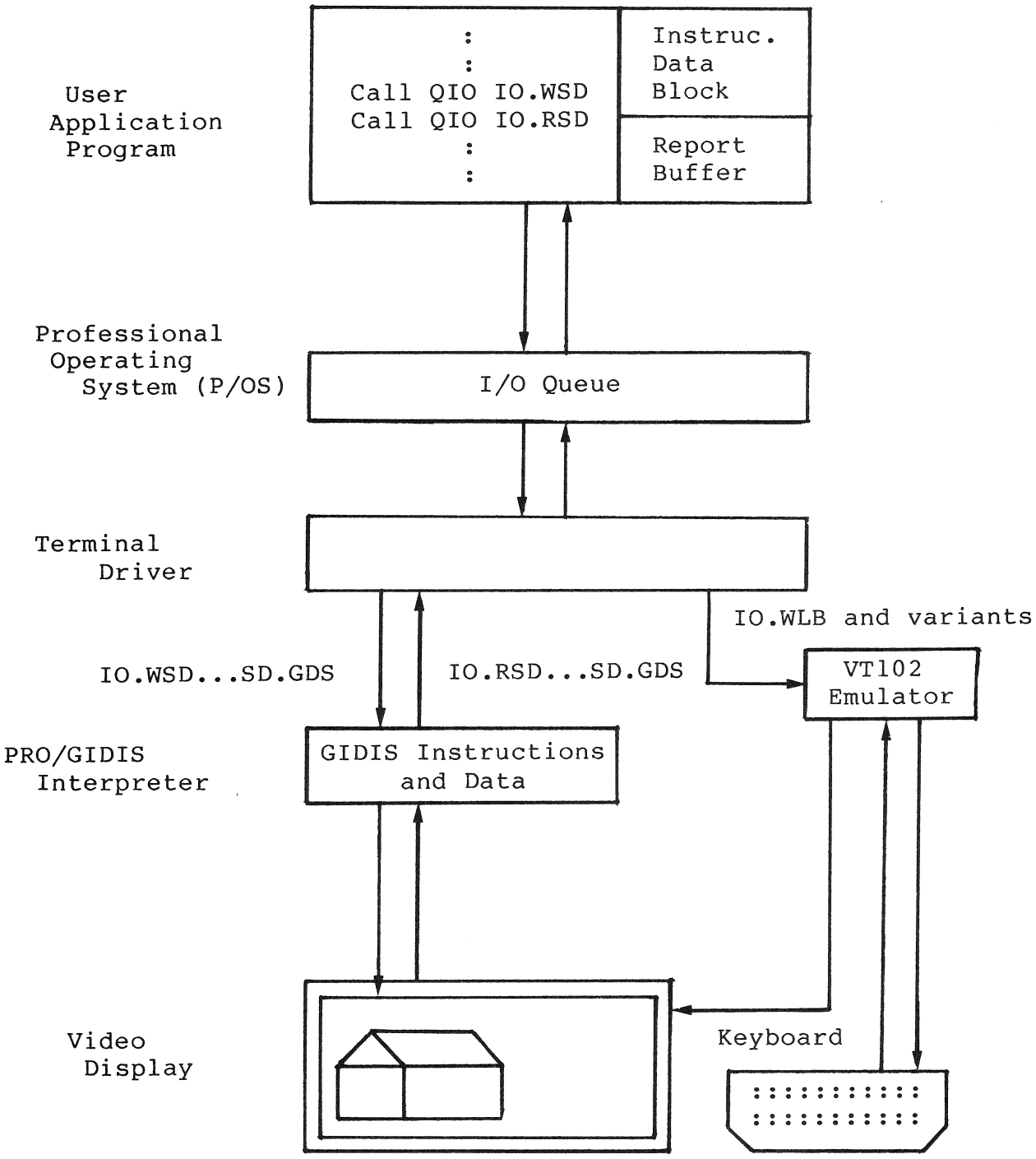


Figure 2-1: PRO/GIDIS Data Path

On a single-plane system, both GIDIS and the VT102 emulator draw on the same plane and overwrite each other's data. On a three-plane system, the VT102 emulator only draws on plane three. Thus, if PRO/GIDIS modifies only planes one and two, there will

be minimal interference. The SET_PLANE_MASK instruction specifies which planes PRO/GIDIS can modify.

The VT102 emulator scrolls all three planes when the scrolling region is set to the entire screen. Any graphics information in any plane scrolls with the text. If the scrolling region is smaller than the entire screen, the VT102 emulator redraws the characters in their new positions. This does not scroll or otherwise affect graphics information in planes one and two but it erases graphics information in plane three.

You can send an RIS (Reset to Initial State - <ESC>c) escape sequence to the VT102 emulator in order to reset both the VT102 emulator and PRO/GIDIS to their initial states. PRO/GIDIS immediately performs an "INITIALIZE -1" instruction, clears the bitmap, and expects an opcode as the next word in the instruction/data stream. Thus, you can use RIS to ensure that your program and PRO/GIDIS are "in synch" when your program starts up. You cannot use it arbitrarily in the middle of picture generation because of the global initialization effect.

The QIO and QIOW directives are described in detail in the P/OS System Reference Manual. The examples in this manual show the $S forms for clarity. The $C and $ forms can be used as well.

IO.WSD and IO.RSD are resolved in the normal manner for system symbols: the Application Builder gets them from module QIOSYM in SYSLIB.OLB. This works correctly in MACRO-11 but may not work with languages that have symbol naming restrictions. For example, FORTRAN-77 does not permit periods in symbol names.

2.1.1 Write Special Data (IO.WSD)

The write-special-data QIO function directs one or more instructions to PRO/GIDIS. The instructions and their associated parameter values are passed in a buffer that must have an even address.

The MACRO-11 format for the Write Special Data QIO (or QIOW) call is shown below.

NOTE

The punctuation marks and the items in bold are mandatory; non-bold items are optional. Items in upper-case letters must be used exactly as shown. Items in lower-case letters must be replaced as described.

QIOW$S #IO.WSD,lun,efn,pri,isb,ast,<buffer,length,,#SD.GDS>

lun is a logical unit number assigned to the terminal.

efn is an event flag number (required with the synchronous wait form QIOW).

pri is the priority (ignored but must be present).

isb is the address of the I/O status block.

ast is the address of the AST service routine entry point.

buffer is the address of the buffer containing PRO/GIDIS instructions and parameters.

length is the length of the PRO/GIDIS instruction/parameter buffer (specified as an even number of bytes in the range 2 to 8128).

SD.GDS is a data type parameter that indicates PRO/GIDIS output.

The QIO system directive returns status in a special global variable called $DSW. Some possible values are:

IS.SUC	Successful completion
IE.ILU	Invalid logical unit number
IE.IEF	Invalid event flag number

For a full list of error codes, refer to the QIO directive description and the terminal driver section of the P/OS System Reference Manual.

When the QIO directive is successful, it can return the following status codes in the I/O status block.

IO.SUC	Successful completion
IS.PND	I/O request pending
IE.ABO	Operation aborted
IE.DNR	Device not ready

2.1.2 Read Special Data (IO.RSD)

The read-special-data QIO function reads reports placed in the report queue by PRO/GIDIS report-request instructions: REQUEST_CURRENT_POSITION, REQUEST_STATUS, and REQUEST_CELL_STANDARD. These instructions are detailed in Chapter 10.

The MACRO-11 format for the Read Special Data QIO or QIOW call is shown below.

NOTE

The punctuation marks and the items in bold are mandatory. Non-bold items are optional. Items in upper-case letters must be used exactly as shown. Items in lower-case letters must be replaced as described.

QIOW$S #IO.RSD,lun,efn,pri,isb,ast,<buffer,length,,#SD.GDS>

lun is a logical unit number assigned to the terminal.

efn is an event flag number (required with the synchronous wait form QIOW).

pri is the priority (ignored but must be present).

isb is the address of the I/O status block.

ast is the address of the AST service routine entry point.

buffer is the address of the buffer to contain PRO/GIDIS report data.

length is the length of the PRO/GIDIS report buffer (specified as an even number of bytes in the range 2 to 8128).

SD.GDS is a data type parameter that indicates PRO/GIDIS output.

If there is no data available, the QIO waits until enough data to fill the buffer becomes available. During this wait, no IO.WSD (write special data) is peformed, even if the no-wait form was used. To avoid deadlock, the preferred method is to issue a QIO$W for the exact number of bytes expected after the request instruction is sent to PRO/GIDIS.

The QIO system directive returns status in a special global variable called $DSW. Some possible values are:

IS.SUC	Successful completion
IE.ILU	Invalid logical unit number
IE.IEF	Invalid event flag number

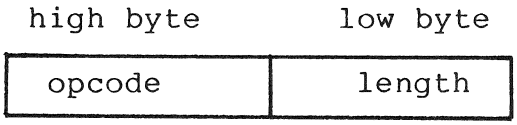
For a full list of error codes, refer to the QIO directive description and the terminal driver section of the P/OS System Reference Manual.

When the QIO directive is successful, it can return the following status codes in the I/O status block.

IO.SUC	Successful completion
IS.PND	I/O request pending
IE.ABO	Operation aborted
IE.DNR	Device not ready

2.2 PRO/GIDIS INSTRUCTION SYNTAX

The PRO/GIDIS interpreter accepts a stream of PRO/GIDIS instructions consisting of operation code (opcode) words and associated parameter blocks. Specific delimiters are not required between PRO/GIDIS instructions. To ensure that the PRO/GIDIS interpreter handles each individual instruction properly, the opcode word for each instruction contains a length value that specifies the number of parameter words that follow it.



2.2.1 Operation Codes

PRO/GIDIS instruction names map to specific numeric operation codes. For example, the INITIALIZE instruction has an opcode of 1, while the SET_PRIMARY_COLOR instruction opcode is 21. (Appendix A provides a list of PRO/GIDIS instructions and their corresponding numeric opcodes.)

Your program can define PRO/GIDIS instruction names as numeric constants. For example, in MACRO-11, this could be:

```
G$INIT = 1.
G$PRIM = 21.
```

In FORTRAN, this could be:

```
INTEGER*2 GINIT,GPRIM
PARAMETER (GINIT = 1, GPRIM = 21)
```

In PASCAL, this could be:

```
CONST
  INITIALIZE = 1;
  SET_PRIMARY_COLOR = 21;
```

2.2.2 Parameter Blocks

Most PRO/GIDIS instructions require a specific number of parameters. For example, SET_POSITION needs exactly two parameters.

A length value in the range 0 to 254 indicates a fixed-length parameter block. For example, if you specify a length value of two, PRO/GIDIS expects two parameter words as shown in Example 2-1.

Some PRO/GIDIS instructions accept a variable number of parameters. The instructions in this category are DRAW_LINES, DRAW_REL_LINES, DRAW_CHARACTERS, and LOAD_CHARACTER_CELL.

```
.BYTE   2.,29.  ;Instruction data block length = 2
                ;Opcode for SET_POSITION instruction = 29
.WORD   100.    ;Horizontal coordinate for current pos.
.WORD   350.    ;Vertical coordinate for current pos.
                ;Following execution of this instruction,
                ;the current position is 100,550.
```

Example 2-1: Instruction with Fixed-Length Parameter Block

A length value of 255 indicates that the parameter block contains a variable number of parameter words terminated by an END_LIST instruction word (-32768), as shown in example 2-2. Thus, in a variable-length parameter block, a parameter word cannot contain the value -32768.

```
.BYTE   255.,26.;END-LIST-term.,opcode=DRAW_REL_LINES
.WORD   10.     ;dx1
.WORD   -30.    ;dy1
.WORD   20.     ;dx2
.WORD   +60.    ;dy2
.WORD   -32768. ;END_LIST instruction opcode word
```

Example 2-2: Instruction with Variable-Length Parameter Block

This technique permits the PRO/GIDIS interpreter to handle erroneous or unsupported opcodes without aborting the program or misinterpreting subsequent opcodes. If the interpreter does not recognize an instruction, that instruction and any data following

are ignored. After counting and discarding the erroneous instruction's data, PRO/GIDIS proceeds to the next sequential instruction.

NOTE

PRO/GIDIS on P/OS V1.7 sets the status flag to SUCCESS when it fails to recognize an instruction. Future versions may set the status flag to FAILURE.

An instruction with insufficient parameters is not executed at all. An instruction with extra parameters is executed as though the extra arguments do not exist; the extra parameters are skipped and not interpreted as new instructions. For example, a SET_POSITION instruction with only one argument is ignored, while a SET_POSITION with three arguments uses the first two arguments and discards the third.

2.3 SAMPLE MACRO-11 PROGRAM

```
IOSB:   .BLKW   2.
OBUF:   .BYTE   0.,55.  ;Length=0 REQUEST_CURRENT_POSITION
RBUF:   .BLKW   3.
                        ;SEND INSTRUCTION TO PRO/GIDIS
        QIOW$S #IO.WSD,#5,#1,,#IOSB,,<#OBUF,#2,,#SD.GDS>
        BCS     ERROR   ; DIRECTIVE FAILED
        TSTB    IOSB
        BLE     ERROR   ;OPERATION FAILED
                        ;READ THE REPORT
        QIOW$S #IO.RSD,#5,#1,,#IOSB,,<#RBUF,#6,,#SD.GDS>
        BCS     ERROR   ;BRANCH IF DIRECTIVE FAILED
        TSTB    IOSB
        BLE     ERROR   ;BRANCH IF OPERATION FAILED
                        ;
                        ; NEW CONTENTS OF RBUF:
                        ; BYTE AT RBUF    2.  (LENGTH)
                        ; BYTE AT RBUF+1  1.
                        ;   (CURRENT POSITION REPORT TAG)
                        ; RBUF+2:  CURRENT X POSITION
                        ; RBUF+4:  CURRENT Y POSITION

ERROR:                  ; Error handling routine
```

2.4 SAMPLE FORTRAN PROGRAM

```
      INTEGER*2   SDGDS, IOWSD, IORSD, ISSUC
      PARAMETER  (SDGDS=1), (IOWSD="5410), (IORSD="6030), (ISSUC=1)
      INTEGER*2  IOSB(2), IDS, OBUF
      INTEGER*2  RBUF(3),PARLST(6)

      OBUF = 55*256+0    !OPCODE 55=REQUEST_CURRENT_POSITION
                         !LENGTH=0


      CALL GETADR(PARLST(1),OBUF)          ! ADDRESS

      PARLST(2) = 2                        !LENGTH=2 BYTES
      PARLST(4) = SDGDS

      CALL WTQIO(IOWSD,5,1,0,IOSB,PARLST,IDS)
      IF (IDS.NE.ISSUC) GO TO 999          !DIRECTIVE FAILED
      IF (IOSB(1).NE.ISSUC) GO TO 999      !I/O REQUEST FAILED

      CALL GETADR(PARLST(1),RBUF)

      PARLST(2) = 6                  !EXPECTED LENGTH OF REPORT IN BYTES

      CALL WTQIO(IORSD,5,1,0,IOSB,PARLST,IDS)
      IF (IDS.NE.ISSUC) GO TO 999          !DIRECTIVE FAILED
      IF (IOSB(1).NE.ISSUC) GO TO 999      !I/O REQUEST FAILED

            ! NEW CONTENTS OF RBUF:
            ! RBUF(1):  258
            ! REPORT TAG = 1*256+2
            ! 1 = THE REPORT TAG AND 2 = LENGTH OF DATA FOLLOWING
            ! RBUF(2):  CURRENT X POSITION IN GIDIS OUTPUT SPACE
            ! RBUF(3):  CURRENT Y POSITION IN GIDIS OUTPUT SPACE

999         ! ERROR FOUND
```

CHAPTER 3

CONTROL INSTRUCTIONS

This chapter contains a detailed description of each control instruction. Table 3-1 lists the instructions covered in the chapter.

Table 3-1: Control Instructions Summary Chart

Opcode/Length	Instruction	Arguments
1/1	INITIALIZE	subsystem-mask-value
6/0	NEW_PICTURE	
24/0	END_PICTURE	
28/0	FLUSH_BUFFERS	
5/6	SET_OUTPUT_CURSOR	alphabet, index, width, height, offset_x, offset_y
0/0	NOP	
128/0	END_LIST	

3.1 INITIALIZE

The initialize instruction restores PRO/GIDIS characteristics to their power-on default states. Specifically, this instruction sets up PRO/GIDIS device subsystem characteristics such as the color map, the primary color, secondary color, addressing, writing modes, and so forth.

Opcode: 1 **Length:** 1

Format: INITIALIZE subsystem-mask-value

subsystem-mask-value is a word that specifies a set of graphics subsystems as shown in Figure 3-1.

Status: SUCCESS

The INITIALIZE instruction represents device subsystems by specific bits in a mask value passed to PRO/GIDIS with the instruction. For example, a parameter value of 4 (bit 2 set) restores a device's text subsystem's power-on default conditions. The initialize instruction mask bit settings for the various subsystems are shown in Figure 3-1.

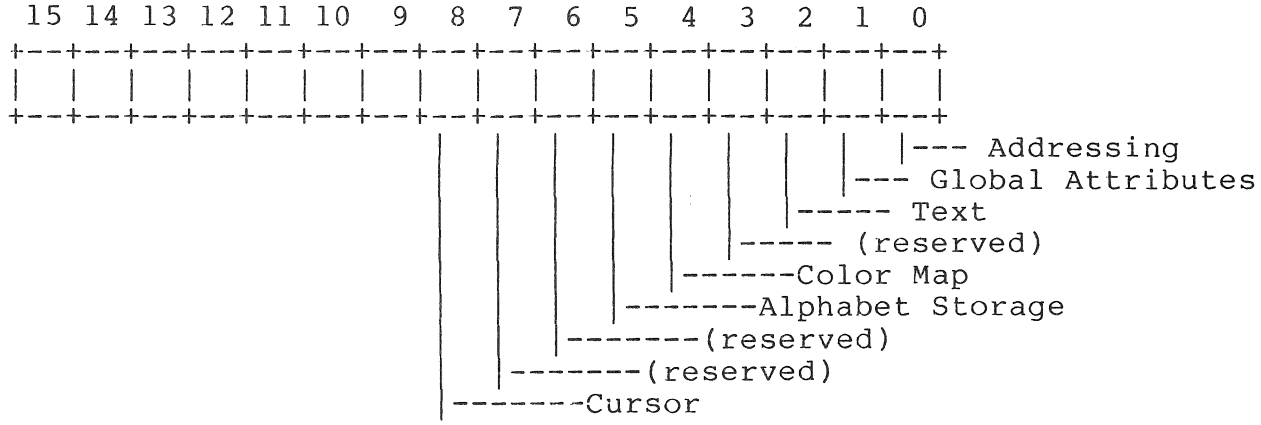


Figure 3-1: INITIALIZE Subsystem Initialization Bit Mask

The PRO/GIDIS subsystems that can be selected for initialization are listed in Table 3-2.

Table 3-2: Initialization Subsystems

Subsystem		Description	Bit
Addressing	-	Resets the viewing transformation	0
Global Attributes	-	Reinitializes writing mode, primary color, secondary color, line and area texture, planes selected, and pixel size	1

Text	-	Resets the current alphabet, unit size, display size, cell rotation, cell rendition, implicit cell movement flag, and explicit cell movement	2
Color map	-	Reinitializes the color map	4
Alphabet	-	Clears all user-defined alphabets	5
Cursor	-	Resets the output cursor	8

Notes:

- You can OR (logical inclusive) mask values together to initialize multiple subsystems in one instruction.
- A mask of -1 decimal (177777 octal) explicitly initializes all subsystems.
- The order of initialization is: (1) addressing, (2) global attributes, (3) text, (4) color map, (5) alphabet storage, and (6) cursor.
- Regardless of the mask word, the INITIALIZE instruction aborts any blocks begun with BEGIN_FILLED_FIGURE.
- Table 3-3 lists all of the state variables affected and their values after initialization.
- Some state variables are included in more than one subsystem.

Example:

```
        .BYTE   1.,1.           ;length=1,opcode for INITIALIZE
        .WORD   1.!2.!4.        ;addressing, global attributes,
                                ;and text subsystems mask bits
```

Table 3-3: Initialization Variable States

Variable	Reset State
Addressing Subsystem	
output ids width	default ids width (960)
output ids height	default ids height (600)

output viewport x origin	0
output viewport y origin	0
output viewport width	default ids width (960)
output viewport height	default ids height (600)
gidis output space x origin	0
gidis output space y origin	0
gidis output space width	default ids width (960)
gidis output space height	default ids height (600)
output clipping x origin	0
output clipping y origin	0
output clipping width	default ids width (960)
output clipping height	default ids height (600)
current position x	0
current position y	0
output cursor width	standard unit width
output cursor height	standard unit height
output cursor x offset	1/2 standard unit width
output cursor y offset	1/2 standard unit height
line texture size	line texture length *
area texture width	standard unit width
area texture height	standard unit height
logical pixel width	0 (1 hardware pixel)
logical pixel height	0 (1 hardware pixel)
logical pixel x offset	0
logical pixel y offset	0
cell movement mode flag	implied movement
cell explicit movement dx	0
cell explicit movement dy	0
cell display size width	standard display width
cell display size height	standard display height
cell unit size width	standard unit width
cell unit size height	standard unit height

* converted to new GOS coordinates

Global Attributes Subsystem

primary color	7
secondary color	0
plane mask	7
writing mode	overlay
logical pixel width	0 (1 hardware pixel)
logical pixel height	0 (1 hardware pixel)
logical pixel x offset	0
logical pixel y offset	0
line texture pattern	solid (all ones)
line texture length	16 (bits)
line texture size	16 hardware pixels *
area texture alphabet	-1
area texture char	0
area texture width	standard unit width
area texture height	standard unit height

* converted to new GOS coordinates

Text Subsystem

current alphabet	0
cell display size width	standard display width
cell display size height	standard display height
cell unit size width	standard unit width
cell unit size height	standard unit height
cell rotation	0
cell oblique	0
cell rendition	none
cell movement mode flag	implied movement
cell explicit movement dx	0
cell explicit movement dy	0

Color Map Subsystem

	R	G	B	M	name
color map [0]	.0,	.0,	.0,	.0	black (dark)
color map [1]	.2,	.2,	.6,	.2	blue (dark gray)
color map [2]	.7,	.2,	.2,	.3	red (light gray)
color map [3]	.2,	.7,	.2,	.4	green (light)
color map [4]	.6,	.6,	.6,	.7	white (light)
color map [5]	.6,	.6,	.6,	.7	white (light)
color map [6]	.6,	.6,	.6,	.7	white (light)
color map [7]	.6,	.6,	.6,	.7	white (light)

Alphabet Storage Subsystem

alphabet extent [1 to 16]	0
alphabet height [1 to 16]	0
alphabet width [1 to 16]	0

Cursor Subsystem

output cursor alphabet	-1
output cursor character index	0
output cursor width	standard unit width
output cursor height	standard unit height
output cursor x offset	1/2 standard unit width
output cursor y offset	1/2 standard unit height

3.2 NEW_PICTURE

The NEW_PICTURE instruction prepares the video display for a new picture by clearing the screen. This instruction sets the entire screen to the current secondary color. (The SET_SECONDARY_COLOR instruction is described in the chapter on global attributes.)

Opcode: 6 **Length:** 0

Format: NEW_PICTURE

Status: SUCCESS

For device-independent programs, it is recommended that you use NEW_PICTURE and END_PICTURE to enclose any picture-drawing instructions. Place instructions for pausing or requesting operator interaction after the END_PICTURE instruction and before another NEW_PICTURE instruction.

Notes:

- PRO/GIDIS ignores any arguments specified with a NEW_PICTURE instruction.
- The secondary color is written to the bitmap subject to the plane mask in effect at the time NEW_PICTURE executes. (Refer to the SET_PLANE_MASK instruction in the chapter on global attributes.)
- NEW_PICTURE clears all pixels in the bitmap including those outside the clipping region. In particular, it clears the 32-pixel bands on both sides of the bitmap that are not part of HAS and not normally used.

Example:

```
        .BYTE   0.,6.           ;length=0,opcode for NEW_PICTURE
```

3.3 END_PICTURE

The END_PICTURE instruction causes all internally buffered data to be sent to the output device. (Refer to the FLUSH_BUFFERS instruction). Any required device output processing of a picture occurs here (top-of-form, wait for interactive response, and so forth).

Opcode: 24 **Length:** 0

Format: END_PICTURE

Status: SUCCESS

For device-independent programs, it is recommended that you use NEW_PICTURE and END_PICTURE to enclose any picture-drawing instructions. Place instructions for pausing or requesting operator interaction after the END_PICTURE instruction and before another NEW_PICTURE instruction.

Notes:

- END_PICTURE's effect on the video display is the same as the FLUSH_BUFFERS instruction (for P/OS V1.7).

Example:

```
        .BYTE   0.,6.           ;length=0,opcode for NEW_PICTURE
          .                     ;
          .                     ;drawing instructions
          .                     ;
        .BYTE   0.,24.          ;length=0,opcode for END_PICTURE
          .                     ;
          .                     ;wait for operator response
          .                     ;
        .BYTE   0.,6.           ;length=0,opcode for NEW_PICTURE
          .                     ;
          .                     ;more drawing instructions
          .                     ;
```

3.4 FLUSH_BUFFERS

FLUSH_BUFFERS forces execution of all buffered instructions.

Opcode: 28 **Length:** 0

Format: FLUSH_BUFFERS

Status: SUCCESS

Notes:

- PRO/GIDIS ignores any arguments specified with a FLUSH_BUFFERS instruction.
- Issue a FLUSH_BUFFERS instruction prior to any delay or wait for operator response so that all drawing instructions are executed before the wait.

Example:

```
        .BYTE   0.,28.  ;length=0,opcode for FLUSH_BUFFERS
```

3.5 SET_OUTPUT_CURSOR

The SET_OUTPUT_CURSOR instruction selects a specific character to be used as the program's output cursor. The output cursor is a visible indication of the current output position.

Opcode: 5 **Length:** 6

Format: SET_OUTPUT_CURSOR alphabet, index,
width, height,
offset-x, offset-y

alphabet specifies the alphabet containing the character or the special cursors indicator (-1).

index specifies the character or special cursor.

width specifies the width of the cursor in GIDIS Output Space coordinates (greater than or equal to zero).

height specifies the height of the cursor in GIDIS Output Space coordinates (greater than or equal to zero).

offset-x specifies the X offset from the top left corner of the cursor to the current position (range 0 to width).

offset-y specifies the Y offset from the top left corner of the cursor to the current position (range 0 to height).

Status: SUCCESS if the requested character (alphabet, index) is defined, and width and height and coordinates are in range; otherwise FAILURE.

Notes:

- The width and height (in PRO/GIDIS Output Space) are treated as a unit cell size; there is no equivalent of a display cell. When width and height are adjusted automatically to an integral multiple of the storage size of the character, the x and y offsets are adjusted by the same ratio.

- An alphabet code of -1 specifies that one of the special built-in cursors is to be used. For P/OS 1.7 these cursors are:

-1	No cursor
0	Implementation default (same as 1)
1	Tracking Cross (small cross)
2	Crosshairs (full screen width and height)

User-specified width and height are ignored when the tracking is used. All other values are reserved.

- If the chosen cursor is not predefined (either a special cursor or a character in alphabet 0, your program must first define the character and then execute a SET_OUTPUT_CURSOR instruction. If the character is redefined after the SET_OUTPUT_CURSOR, the appearance of the cursor is unchanged until another SET_OUTPUT_CURSOR executes.

- Once the SET_OUTPUT_CURSOR instruction executes, the appearance of the cursor changes immediately.

Example:

```
        .BYTE   6.,5.   ;length=6,opcode for SET_OUTPUT_CURSOR
        .WORD   1.      ;Alphabet 1 (user-defined alphabet)
        .WORD   2.      ;Character index value
                        ;(Assume that Alphabet 1, character-index
                        ;2, is defined as an arrow pointing
                        ;straight upward
        .WORD   30.     ;Width of 30
        .WORD   30.     ;Height of 30
        .WORD   15.     ;x offset
        .WORD   0.      ;y offset
                        ;The arrow is the new cursor with the tip
                        ;  being at the current position.
```

Example:

```
        .BYTE   6.,5.   ;length=6,opcode for SET_OUTPUT_CURSOR
        .WORD   -1.     ;PRO/GIDIS Cursor Alphabet
        .WORD   -1.     ;No cursor
        .WORD   0.      ;Width value of zero  (ignored)
        .WORD   0.      ;Height value of zero  (ignored)
        .WORD   3.      ;x offset (ignored)
        .WORD   4.      ;y offset (ignored)
```

3.6 NOP

The NOP instruction performs no operation. Execution of a NOP has no effect on the current state of PRO/GIDIS, other than to set the status flag to SUCCESS.

Opcode: 0 **Length:** 0

Format: NOP

Status: SUCCESS

Notes:

- PRO/GIDIS ignores any arguments included with a NOP instruction.

Example:

```
        .BYTE   0.,0.   ;length=0,opcode for NOP
```

Example:

```
        .BYTE   2.,0.   ;length=2,opcode for NOP
        .WORD   1540.   ;private data (ignored by PRO/GIDIS)
        .WORD   71.     ;private data (ignored by PRO/GIDIS)
```

3.7 END_LIST

The END_LIST instruction indicates the end of a variable argument list. This instruction follows the last argument in the list. Those PRO/GIDIS instructions often used with a variable-length argument list that terminates with an END_LIST instruction include the following: DRAW_LINES, DRAW_REL_LINES, DRAW_CHARACTERS, and LOAD_CHARACTER_CELL.

Opcode: 128 **Length:** 0

Format: END_LIST

Status: SUCCESS

Notes:

- PRO/GIDIS ignores any arguments specified with an END_LIST instruction.
- The code for this instruction (-32768) is invalid for any data in any variable argument list. Your program must ensure this value is never sent in a variable argument list; however, the code is valid as an argument for fixed-length instructions. When this value appears as an opcode, no operation is performed. For example, the point [-32768,0] could not be sent in a DRAW_LINES instruction that terminates with an END_LIST instruction, but could be sent in a DRAW_LINES instruction with counted arguments.

Example:

```
        .BYTE   255.,25. ;length=END_LIST,opcode for DRAW_LINES
        .WORD   100.     ;DRAW_LINES data
        .WORD   110.     ;DRAW_LINES data
        .WORD   -32768.  ;END_LIST
                         ;Note that the length = 255 is a
                         ;  special value and does not indicate
                         ;  255 data words following, but that
                         ;  there is an unknown number of words
                         ;  following, to be terminated with
                         ;  the END_LIST code
```

CHAPTER 4

VIEWING TRANSFORMATION INSTRUCTIONS

This chapter contains a detailed description of each of the viewing transformation instructions. Table 4-1 lists the instructions covered in the chapter.

Table 4-1: Viewing Transformation Instructions Summary Chart

Opcode/Length	Instruction	Arguments
12/2	SET_OUTPUT_IDS	width, height
13/4	SET_OUTPUT_VIEWPORT	ulx, uly, width, height
9/4	SET_GIDIS_OUTPUT_SPACE	ulx, uly, width, height
4/4	SET_OUTPUT_CLIPPING_REGION	ulx, uly, width, height

4.1 SET_OUTPUT_IDS

The SET_OUTPUT_IDS instruction specifies the width and height of Imposed Device Space (IDS). It also sets the window, clipping region, and viewport to be identical with IDS, and resets all global attributes to their default values as shown in Table 4-2.

Opcode: 12 **Length:** 2

Format: SET_OUTPUT_IDS width, height

The parameters are integers greater than or equal to zero.

width specifies the width of IDS.

height specifies the height of IDS.

Status: SUCCESS if width and height are greater than 0, FAILURE otherwise.

Notes:

- The upper left corner of IDS is always [0,0]. The coordinates of the lower-right corner is [width - 1, height - 1].
- When the picture aspect ratio of IDS space is not equal to the picture aspect ratio of the hardware address space, only a portion of the view surface is used.
- No drawing is done by the SET_OUTPUT_IDS instruction.
- It is recommended that width and height be less than or equal to 16384 (2 to the 14th power). This limitation ensures sufficient off-screen address space for accurate clipping.
- Table 4-1 lists all of the state variables affected and their reset values after the SET_OUTPUT_IDS executes. All sizes are reset.

Example: See SET_GIDIS_OUTPUT_SPACE description.

Table 4-2: State Variables Affected by SET_OUTPUT_IDS

Variable	Reset State
IDS width	as specified
IDS height	as specified
viewport	same as IDS
GIDIS output space	same as IDS
clipping region	same as IDS
current position x	0
current position y	0
cursor width	standard unit width
cursor height	standard unit height
cursor x offset	1/2 standard unit width
cursor y offset	1/2 standard unit height
line texture size	line texture length *
area texture width	standard unit width
area texture height	standard unit height

logical pixel x offset	0
logical pixel y offset	0
logical pixel width	0 (1 hardware pixel)
logical pixel height	0 (1 hardware pixel)
cell movement mode flag	implicit, local
cell explicit movement dx	0
cell explicit movement dy	0
cell display size width	standard display width
cell display size height	standard display height
cell unit size width	standard unit width
cell unit size height	standard unit height

* converted to new GOS coordinates

4.2 SET_OUTPUT_VIEWPORT

The SET_OUTPUT_VIEWPORT instruction specifies the viewport.

Opcode: 13 **Length:** 4

Format: SET_OUTPUT_VIEWPORT ulx, uly, width, height

The parameters are integer values representing IDS coordinates.

ulx specifies the x (horizontal) address of the origin of the viewport.

uly specifies the y (vertical) address of the origin of the viewport.

width specifies the width of the viewport (value must be greater than zero).

height specifies the height of the viewport (value must be greater than zero).

Status: SUCCESS if width and height are greater than 0, FAILURE otherwise.

Notes:

- No drawing is done by the SET_OUTPUT_VIEWPORT instruction.
- If the picture aspect ratios of the window and viewport are not equal, only a portion of the viewport is used.

- Unlike SET_OUTPUT_IDS and SET_GIDIS_OUTPUT_SPACE, this instruction does not change any of the state variables that depend on the definition of GIDIS Output Space (cell unit size, line texture size, and so forth). The GOS values are preserved.

Example: See SET_GIDIS_OUTPUT_SPACE description.

4.3 SET_GIDIS_OUTPUT_SPACE

The SET_GIDIS_OUTPUT_SPACE instruction specifies the bounds of the window in GIDIS Output Space. It also sets the output clipping region to coincide with the window and resets all global attributes to their default values as shown in Table 4-3.

Opcode: 9 **Length:** 4

Format: SET_GIDIS_OUTPUT_SPACE ulx, uly, width, height

The parameters are integer values representing GIDIS Output Space coordinates.

ulx specifies the x (horizontal) address of the origin of the window.

uly specifies the y (vertical) address of the origin of the window.

width specifies the width of the window (value must be greater than zero).

height specifies the height of the window (value must be greater than zero).

Status: SUCCESS if width and height are greater than zero, FAILURE otherwise

Notes:

- No drawing is done when SET_GIDIS_OUTPUT_SPACE executes.
- It is recommended that a maximum absolute value of 16384 (2 to the 14th power) be used for the following: ulx, uly, ulx + width, and uly + height. This will allow sufficient off-screen address space for accurate clipping.

- Table 4-3 lists all of the state variables affected and their reset values after the SET_GIDIS_OUTPUT_SPACE instruction executes. All sizes are reset.

Example:

```
.BYTE   2.,12.  ;length=2,opcode for SET_OUTPUT_IDS
.WORD   960.    ;width
.WORD   600.    ;height (upper left corner is [0,0] and
                ;  lower right corner is [959,599])

.BYTE   4.,13.  ;length=4,opcode for SET_OUTPUT_VIEWPORT
.WORD   0.      ;
.WORD   0.      ;Sets the viewport to the left half
.WORD   480.    ;   of the screen
.WORD   600.    ;

.BYTE   4.,9.   ;length=4,opcode SET_GIDIS_OUTPUT_SPACE
.WORD   0.      ;
.WORD   0.      ;Sets GOS to 0-to-2399 in X
.WORD   2400.   ;  and 0-to-3999 in Y  (all within the
.WORD   4000.   ;      left half of the screen)
```

Table 4-3: State Variables Affected by SET_GIDIS_OUTPUT_SPACE

Variable	Reset State
GIDIS output space	as specified
clipping region	same as window
current position x	0
current position y	0
output cursor width	standard unit width
output cursor height	standard unit height
output cursor x offset	1/2 standard unit width
output cursor y offset	1/2 standard unit height
line texture size	line texture length *
area texture width	standard unit width
area texture height	standard unit height
logical pixel x offset	0
logical pixel y offset	0
logical pixel width	0 (1 hardware pixel)
logical pixel height	0 (1 hardware pixel)

cell movement mode flag	implied
cell explicit movement dx	0
cell explicit movement dy	0
cell display size width	standard display width
cell display size height	standard display height
cell unit size width	standard unit width
cell unit size height	standard unit height

* converted to new GOS coordinates

4.4 SET_OUTPUT_CLIPPING_REGION

The SET_OUTPUT_CLIPPING_REGION instruction specifies the output clipping region in PRO/GIDIS output space coordinates. The output clipping region is the rectangle within which the PRO/GIDIS interpreter is permitted to draw on the view surface.

Opcode: 4 **Length:** 4

Format: SET_OUTPUT_CLIPPING_REGION ulx, uly, width, height

The parameters are GIDIS Output Space coordinates.

ulx specifies the X (horizontal) coordinate of the upper left corner of the clipping region.

uly specifies the Y (vertical) coordinate upper left corner of the clipping region.

width specifies the width of the clipping region.

height specifies the height of the clipping region

Status: SUCCESS if width and height are greater than or equal to 0, FAILURE otherwise

Notes:

- The SET_OUTPUT_CLIPPING_REGION instruction does not do any drawing.
- You cannot set the clipping region to an area larger than the output device. An attempt to do so reduces the clipping region to the portion that maps to the output device. If no portion maps, the clipping region width and height are set to 0, and all drawing is clipped to nothing.

- Clipping affects all drawing primitives, including those generating intermediate points that might be outside the spaces specified. For example, circles are drawn only within the specified clipping region.
- If width or height are zero, all drawing instructions are clipped to nothing.
- The clipping region includes the borders. Therefore, the following instructions:

```
SET_OUTPUT_CLIPPING_REGION 100, 100, 300, 300
SET_POSITION 50, 100
DRAW_LINES 100, 100
```

  draw the pixel at [100,100], assuming this point is within the visible area.

```
SET_POSITION 100, 150
DRAW_LINES 500, 150
```

  draws pixels from [100,150] to [400,150] inclusive.

Example:

```
.BYTE   4.,4.   ;length=4,SET_OUTPUT_CLIPPING_REGION
.WORD   100.    ;Sets output clipping region to rectangle
.WORD   100.    ;with the upper left corner at 100,100
.WORD   400.    ;and the lower right corner at 500,200.
.WORD   100.
```

CHAPTER 5

GLOBAL ATTRIBUTES INSTRUCTIONS

This chapter contains a detailed description of each global attribute instruction. Table 5-1 lists the instructions covered in the chapter.

Table 5-1: Global Attribute Instructions Summary Chart

Opcode/Length	Instruction/Arguments	
21/1	SET_PRIMARY_COLOR	color-value
15/1	SET_SECONDARY_COLOR	color-value
16/6	SET_COLOR_MAP_ENTRY	map, index, red, green, blue, mono
20/1	SET_PLANE_MASK	plane-mask
22/1	SET_WRITING_MODE	mode-value
19/4	SET_PIXEL_SIZE	width, height, offset-x, offset-y
17/3	SET_LINE_TEXTURE	length, pattern, size
14/2	SET_AREA_TEXTURE	alphabet, char-index
3/2	SET_AREA_TEXTURE SIZE	width, height
69/2	SET_AREA_CELL_SIZE	width, height

5.1 SET_PRIMARY_COLOR

The primary color generally indicates the presence of an image. On a three-plane bitmap system, SET_PRIMARY_COLOR specifies an

index into the color map. On a single-plane system, SET_PRIMARY_COLOR instruction specifies light or dark.

Opcode: 21 **Length:** 1

Format: SET_PRIMARY_COLOR color-value

color-value A word that specifies a color map entry (range 0 to 7) or a monochrome value (0 = dark, any other value = white).

Status: SUCCESS

Notes:

- Refer to the INITIALIZE instruction in Chapter 3 for a list of the power-on default colors.
- This instruction does not draw anything on the screen or affect the bitmap. It affects objects drawn in primary color after the instruction executes.

Example:

```
.BYTE   1.,21.  ;length=1,opcode for SET_PRIMARY_COLOR
.WORD   4.      ;defines primary color as color number 4
```

5.2 SET_SECONDARY_COLOR

The secondary color generally indicates the absence of an image. On a three-plane bitmap system, SET_SECONDARY_COLOR specifies an index into the color map. On a single-plane system, SET_SECONDARY_COLOR instruction specifies light or dark.

Opcode: 15 **Length:** 1

Format: SET_SECONDARY_COLOR color-value

color-value A word that specifies a color map entry (range 0 to 7) or a monochrome value (0 = dark, any other value = white).

Status: SUCCESS

Notes:

- Refer to the SET_COLOR_MAP_ENTRY description in this chapter for a list of the power-on default colors.
- This instruction does not draw anything on the screen or affect the bitmap. It affects objects drawn in the secondary color after the instruction executes.

Example:

```
.BYTE   1.,15.  ;length=1,opcode for SET_SECONDARY_COLOR
.WORD   1.      ;defines secondary color as number 1
```

5.3 SET_COLOR_MAP_ENTRY

The SET_COLOR_MAP_ENTRY instruction specifies the contents of a particular color map entry, assuming that the EBO is present. All images that were drawn using that color map entry are immediately affected.

Opcode: 16 **Length:** 6

Format: SET_COLOR_MAP_ENTRY map, index, red, green, blue, mono

map — an integer representing a specific color map. For the Professional, this value must be 0 to represent the Professional's only map.

index — an integer in the range 0 to 7 representing one of the eight color map entries.

red — an integer in the range of 0 to 65535 representing the proportion of red.

green — an integer in the range of 0 to 65535 representing the proportion of green.

blue — an integer in the range of 0 to 65535 representing the proportion of blue.

mono — an integer in the range of 0 to 65535 representing the proportion of monochrome.

Status: SUCCESS if map = 0 and index is in the range 0 to 7.

For convenience, the following table shows proportional values in the form of octal fractions. Because programming languages vary, the equivalents are shown as octal, unsigned decimal, and signed decimal constants.

Table 5-2: Color Map Values for the Professional

Octal Fraction*	Octal Integer	Decimal Integer (Unsigned)	Decimal Integer (Signed)
0.0	0	0	0
0.1	20000	8192	8192
0.2	40000	16384	16384
0.3	60000	24576	24576
0.4	100000	32768	-32768
0.5	120000	40960	-24576
0.6	140000	49152	-16384
0.7	160000	57344	-8192

* Only the even numbers are useful for blue values: (0.0,0.2,0.4, and 0.6).

Notes:

- This instruction is not useful with a single-plane bitmap (non-EBO) device.
- No change occurs in the bitmap when SET_COLOR_MAP_ENTRY executes, but every pixel drawn with the specified index changes color instantly.

Example:

```
.BYTE   6.,16.  ;length=6,opcode for SET_COLOR_MAP_ENTRY
.WORD   0.      ;PRO's bitmap (value must be 0)
.WORD   1.      ;Color index 1
.WORD   20000   ;Red is one-eighth (octal)
.WORD   40960.  ;Green is five-eighths (unsigned decimal)
.WORD   16384.  ;Blue is one-quarter
.WORD   -32768. ;Monochrome is one-half (signed decimal)
```

5.4 SET_PLANE_MASK

The SET_PLANE_MASK instruction specifies which bitmap planes are accessible to the PRO/GIDIS interpreter. The interpreter cannot modify a plane unless that plane has been designated modifiable with a SET_PLANE_MASK instruction.

Opcode: 20 **Length:** 1

Format: SET_PLANE_MASK plane-mask

plane-mask An integer bit mask representing a combination of bitmap planes (as shown below). A set (1) bit indicates an accessible plane.

Status: SUCCESS

The following table shows all combinations that can be represented by plane mask (n) values. It also shows the effective writing indexes for each combination, assuming that all "write-locked" planes are clear.

Table 5-3: Plane Mask Values

n	Plane 3	Plane 2	Plane 1	Writing Indexes
0	-	-	-	none
1	-	-	X	0, 1
2	-	X	-	0, 2
3	-	X	X	0, 1, 2, 3
4	X	-	-	0, 4
5	X	-	X	0, 1, 4, 5
6	X	X	-	0, 2, 4, 6
7	X	X	X	0, 1, 2, 3, 4, 5, 6, 7

X = accessible to PRO/GIDIS
- = not accessible to PRO/GIDIS

Notes:

- SET_PLANE_MASK does not affect the video display.
- Write-locked planes change the effects of the writing modes. See SET_WRITING_MODE for details.
- The Professional VT102 emulator uses plane 3 for text. Refer to Chapter 2 of this manual and the Terminal Subsystem Manual for more information.

- The color map can be used in combination with the color mask to prepare separate images in separate planes for switching back and forth quickly. For example:

```
        .                   ;Set all color map entries to dark
        .
        .

    .BYTE   1.,20.  ;len=1,opcode for SET_PLANE_MASK
    .WORD   1.      ;plane 1
        .
        .           ;draw image A in plane 1
        .
        .           ;Set map entries 1, 3, 5, and 7 to light
        .           ;Image A appears
        .

    .BYTE   1.,20.  ;len=1,opcode for SET_PLANE_MASK
    .WORD   2.      ;plane 2
        .
        .           ;draw image B in plane 2
        .
        .           ;Set map entries 1, 3, 5, and 7 to dark
        .           ;Image A disappears
        .
        .           ;Set map entries 2, 3, 6, and 7 to light
        .           ;Image B appears
        .
```

Although drawing image B might take a while, depending on the complexity of the image, it will appear all at once. You can continue flipping images A and B very quickly. In other words, you can draw B while A is being viewed and so forth.

This technique requires careful coordination of the color map. In order for the contents of one or more planes to have no effect on the visible image, certain color map entries must be set identically:

"Invisible" Planes	Color Map Entry Sets
none	(0) (1) (2) (3) (4) (5) (6) (7)
1	(0,1) (2,3) (4,5) (6,7)
2	(0,2) (1,3) (4,6) (5,7)
3	(0,4) (1,5) (2,6) (3,7)
1,2	(0,1,2,3) (4,5,6,7)
1,3	(0,1,4,5) (2,3,6,7)
2,3	(0,2,4,6) (1,3,5,7)
1,2,3	(0,1,2,3,4,5,6,7)

Example:

```
.BYTE   1.,20.  ;length=1,opcode for SET_PLANE_MASK
.WORD   ^B011   ;Enables GIDIS access to planes 1 and 2
                ;Plane 3 is write-protected
```

5.5 SET_WRITING_MODE

The SET_WRITING_MODE instruction defines how PRO/GIDIS writes images to the bitmap. See Chapter 1 for a description of the writing modes.

Opcode: 22 **Length:** 1

Format: SET_WRITING_MODE mode-code

mode-code Specifies one of the following integer values:

```
0  =  TRANSPARENT
1  =  TRANSPARENT NEGATE
2  =  COMPLEMENT
3  =  COMPLEMENT NEGATE
4  =  OVERLAY
5  =  OVERLAY NEGATE
6  =  REPLACE
7  =  REPLACE NEGATE
8  =  ERASE
9  =  ERASE NEGATE
```

Status: SUCCESS if a valid mode is requested; FAILURE otherwise.

Notes:

- No drawing is done when the SET_WRITING_MODE instruction executes.

Example:

```
.BYTE   1.,22.  ;length=1,opcode for SET_WRITING_MODE
.WORD   6.      ;Specifies REPLACE writing mode
```

5.6 SET_PIXEL_SIZE

The SET_PIXEL_SIZE instruction permits you to increase the size of the logical pixel used for drawing lines.

Opcode: 19 **Length:** 4

Format: SET_PIXEL_SIZE width, height, offset-x, offset-y

The parameters are GIDIS Output Space coordinates.

width specifies the logical drawing pixel width.

height specifies the logical drawing pixel height.

offset-x specifies the X offset from the top left corner of the pixel to the current position.

offset-y specifies the Y offset from the top left corner of the pixel to the current position.

Status: SUCCESS if width and height are greater than or equal to zero, offset x is greater than or equal to zero and not less than width, and offset y is greater than or equal to zero and not greater than height.

Notes:

- The default drawing point is one physical pixel, and, by default, lines are drawn one pixel wide.
- Any value that maps to a size smaller than a hardware pixel is set to a hardware pixel size. In particular, width = 0 and height = 0 always sets the drawing pixel to the hardware pixel size.
- The drawing pixel is always a rectangle orthogonal to the X and Y axes.
- Complement writing mode can produce unexpected results, as can replace mode using line texture.
- No drawing is done when the SET_PIXEL_SIZE function executes.
- SET_PIXEL_SIZE is used in drawing lines and arcs and in the WRITE_PIXEL_ARRAY instruction.

Example:

```
.BYTE   4.,19.  ;length=4,opcode for SET_PIXEL_SIZE
.WORD   6.      ;width in GIDIS output space units
.WORD   6.      ;Height in GIDIS output space units
.WORD   3.      ;Centered horizontally
.WORD   3.      ;Centered vertically
```

5.7 SET_LINE_TEXTURE

The SET_LINE_TEXTURE instruction defines the line texture, a bit pattern that is repeated in drawing lines and arcs.

Opcode: 17 **Length:** 3

Format: SET_LINE_TEXTURE length, pattern, size

length the length (in bits) of the selected pattern. The length parameter must be less than or equal to 16.

pattern a binary line pattern (ones and zeroes). PRO/GIDIS begins the pattern by using the low-order (rightmost) bit 0 first.

size specifies the length (in GIDIS output space coordinates) of one repetition of the pattern. Size must be greater than zero.

Status: SUCCESS if length in a range of 1 to 16, and if size is greater than 0, FAILURE otherwise.

Notes:

- Size is used in a similar manner as the unit size for characters is used for text. Effective size is an integral multiple of the pattern length in physical pixels.
- The pattern length is repeated every size units in the display. The low-order bit is used first, then successively higher order bits. The pattern is rotated as lines are drawn, so that the pattern is preserved around corners and bends.
- For the Professional, the size given is used for horizontal and vertical lines. Diagonal lines have a size that is larger by a factor of the square root of 2 (1.414...)
- Pixel size does not change the 'size' in which the pattern repeats, although the appearance of the line does change.
- The only way to reset the texture to the original is to issue another SET_LINE_TEXTURE instruction.
- No drawing is done by the SET_LINE_TEXTURE.

Example:

```
.BYTE   3.,17   ;length=3., opcode for SET_LINE_TEXTURE
.WORD   8.      ;Length of pattern
.WORD   ^B10001111 ;ON, ON, ON, ON, OFF, OFF, OFF, ON
                ;Bits are used in order low to high
.WORD   100.    ;Size of one repetition of pattern in
                ;GIDIS output space
```

5.8 SET_AREA_TEXTURE

SET_AREA_TEXTURE specifies a two-dimensional bit pattern that defines the appearance of filled figures. Area texture is specified as a character in an alphabet. Any arbitrary pattern (up to the maximum character size) can be used for area texture.

Opcode: 14 **Length:** 2

Format: SET_AREA_TEXTURE alphabet, char-index

alphabet Specifies the number of the alphabet containing the texture character. It can be the DEC Multinational character set (alphabet 0), a user-defined alphabet, or the special texture indicator "-1".

char-index The index of the specific character or special texture.

Status: SUCCESS if the specified character (alphabet, char-index) is defined.

Notes

- The alphabet number is explicitly specified and does not depend on the current alphabet selected by the SET_ALPHABET instruction.
- If the character is re-defined after this function, the appearance of future filled areas is according to the old definition. In other words, the current pattern is copied to a special area texture cell at the time SET_AREA_TEXTURE executes.
- Alphabet -1 is reserved for implementation-defined area textures. Alphabet -1 char-index 0 is derived from the line texture (the pattern is drawn vertically and replicated horizontally) when the instruction executes. The texture size is taken from the line texture size, not from the area texture width and height.

- The pattern always appears orthogonal to the axes; there is no way to rotate the pattern.
- Area textures are self-aligning. When two adjacent or overlapping areas are filled, the appearance shows no seams. An exception occurs in complement and complement-negate modes where overlapping areas collide. Areas that overlap are drawn twice in these modes.
- Pixel size is not used when filling areas.
- No drawing is done when SET_AREA_TEXTURE executes.

Example:

```
.BYTE   2.,14   ;length=2., opcode for SET_AREA_TEXTURE
.WORD   8.      ;User-defined alphabet 2
.WORD   23.     ;23rd character
```

5.9 SET_AREA_TEXTURE_SIZE

The SET_AREA_TEXTURE_SIZE instruction specifies the actual size of texture character displayed; the size is specified in GIDIS output space. Character pattern is mapped completely to this size.

Opcode: 3 **Length:** 2

Format: SET_AREA_TEXTURE_SIZE width, height

The parameters are GIDIS Output Space coordinates.

width specifies the width of one repetition of the area texture pattern.

height specifies the height of one repetition of the area texture pattern.

Status: SUCCESS if width and height are greater than zero, FAILURE otherwise.

Notes:

- The area size is restricted to an integral multiple of the texture pattern. PRO/GIDIS uses the largest available size that is not larger than the size specified. If no available size is small enough, the smallest available is used.

- This instruction does not affect the pattern used.
- No drawing is done when SET_AREA_TEXTURE_SIZE executes.

Example:

```
.BYTE   2.,3    ;length=2., SET_AREA_TEXTURE_SIZE
.WORD   12.     ;Area texture width = 12 units
.WORD   8.      ;Area texture height = 8 units
```

5.10 SET_AREA_CELL_SIZE

The SET_AREA_CELL_SIZE instruction allows you to specify a particular size for the area texture cell, the source for the area texture.

Opcode: 69 **Length:** 2

Format: SET_AREA_CELL_SIZE width, height

width The width (in storage pixels) for the area cell. The width value must be in a range of 1 to 16.

height The height of the area cell in storage pixels. The height value must be in a range of 1 to 16.

Status: SUCCESS if both width and height are in the 1-16 range, otherwise, FAILURE.

Notes:

- When a character cell is loaded into the area cell, PRO/GIDIS clears the remainder (if any) of the area cell bits. Thus, if a subsequent SET_AREA_CELL_SIZE instruction increases the area cell size, the new pattern is all zeros.
- The SET_AREA_TEXTURE instruction sets the area cell size from that of the character cell, overriding any previous SET_AREA_CELL_SIZE specification.
- No drawing is done when this instruction executes.

Example:

```
        .BYTE   2.,14.  ;Length=2, opcode for SET_AREA_TEXTURE
        .WORD   2.      ;User-defined alphabet 2 (assume 8 by 10)
        .WORD   23      ;Character 23
                        ; Area Cell Size is now 8 by 10
        .BYTE   2.,69.  ;Length=2, opcode for SET_AREA_CELL_SIZE
        .WORD   9.      ;area cell width = 9 storage pixels
        .WORD   9.      ;area cell heighth = 9 storage pixels
                        ;Area cell size is now 9 by 9, padded on
                        ;the right, with one column of OFF's, and
                        ;with the bottom row of the character
                        ;cell removed
```

CHAPTER 6

DRAWING INSTRUCTIONS

This chapter contains a detailed description of each PRO/GIDIS drawing instruction. Table 6-1 lists the instructions covered in the chapter.

Table 6-1: Drawing Instructions Summary Chart

Opcode/Length	Instruction/Arguments	
29/2	SET_POSITION	x, y
30/2	SET_REL_POSITION	dx, dy
25/n	DRAW_LINES	x, y
26/n	DRAW_REL_LINES	dx, dy
23/3	DRAW_ARC	x, y, angle
27/3	DRAW_REL_ARC	dx, dy, angle

6.1 SET_POSITION

The SET_POSITION instruction specifies a new current position as an absolute position in GIDIS Output Space.

Opcode: 29 **Length:** 2

Format: SET_POSITION x, y

x Specifies the new X (horizontal) value of the current position in GIDIS output space

y Specifies the new Y (vertical) value of the current position in GIDIS output space

Status: SUCCESS

Notes:

- No drawing is done when SET_POSITION executes.

Example:

```
.BYTE   2.,29.  ;Length=2, opcode for SET_POSITION
.WORD   100.    ;New current position
.WORD   350.    ;is [100,350]
```

6.2 SET_REL_POSITION

The SET_REL_POSITION instruction specifies a new current position as an offset from the old current position in GIDIS Output Space.

Opcode: 30 **Length:** 2

Format: SET_REL_POSITION dx, dy

dx Specifies an offset from the current X (horizontal) position in GIDIS output space

dy Specifies an offset from the current Y (vertical) position in GIDIS output space

Status: SUCCESS, provided no arithmetic overflow occurs; on overflow, FAILURE (position does not change)

Notes:

- No drawing is done when SET_POSITION executes.
- Global symmetry is preserved; SET_REL_POSITION [Dx,Dy] is always same as a 'SET_POSITION [Current_x + Dx,Current_y + Dy]'.

Example:

```
                ;Current position is [100,350]
.BYTE   2.,30.  ;Length=2, opcode for SET_REL_POSITION
.WORD   100.    ;Relative position is
.WORD   -50.    ;[+100,-50]
                ;New current position is [200,300]
```

6.3 DRAW_LINES

The DRAW_LINES instruction draws a series of straight line segments, starting at the current drawing position. The end point of each line segment is specified as absolute coordinate pairs, expressed in GIDIS output space.

Opcode: 25 **Length:** n

Format: DRAW_LINES x1, y1, x2, y2, ...

x1 Specifies the X (horizontal) value of the first line's end point in GIDIS output space.

y1 Specifies the Y (vertical) value of the first line's end point in GIDIS output space.

x2 Specifies the X (horizontal) value of the second line's end point in GIDIS output space.

y2 Specifies the Y (vertical) value of the second line's end point in GIDIS output space.

Status: SUCCESS, provided no filled figure table overflow occurs; on overflow, FAILURE (position does not change)

Notes:

- The coordinates can be specified either in a counted argument list (with the count supplied with the opcode word, or in an END_LIST terminated list with 255 in the opcode word, as described in Chapter 2 of this manual.
- If there is an X coordinate with no Y coordinate, the lone X coordinate is ignored with no error indication.
- The DRAW_LINES instruction is affected by the following global attributes: writing mode, primary color, plane mask, secondary color, pixel size, line texture, and filled figure flag. (See Note 4.)
- The way the coordinate parameters are used depends on the filled figure flag. When the filled figure flag yields FALSE, this instruction draws a straight line from the current position to the specified point. Then, the current position is changed and the specified point becomes the new current position. The next line is drawn from this new position to the location specified by the next parameter pair.

- In complement and complement negate mode, common points (last pixel in one line, first pixel in the next) are drawn only once. The first pixel of a line is skipped and the last pixel is drawn. If the first pixel is the last pixel, the pixel is drawn.

- When the filled figure flag yields TRUE, this instruction saves the given points in the filled figure table. No drawing is done; however, current position changes and is set to each specified (x, y) point. When the instruction completes, the current drawing position is located at the point indicated by the last parameter pair. When the filled figure table is full, coordinate pairs are ignored, and status is set to FAILURE.

- DRAW_LINES modifies the bitmap only inside the clipping region.

Example:

```
                        ;Not in a filled figure definition
                        ;(filled figure flag is FALSE)
                        ;Current position is [200,300]
        .BYTE   2.,25.  ;Length=2, opcode for DRAW_LINES
        .WORD   150.    ;Draw a line from [200,300]
        .WORD   200.    ;to [150,200]
                        ;New current position is [150,200]
```

Example:

```
                        ;current position is [150,200]
                        ;not in a filled figure definition
        .BYTE   4.,25.  ;Length=4, opcode for DRAW_LINES
        .WORD   600.    ;x1
        .WORD   -10.    ;y1
        .WORD   300.    ;x2
        .WORD   +10.    ;y2
                        ;Draw lines from [150,200] to [600,-10]
                        ;then from [600,-10] to [300,10]
                        ;New current position is [300,10]
                        ;Note that both the -10 and the +10 are
                        ;absolute coordinates.
```

Example:

```
                        ;current position is [300,10]
                        ;not in a filled figure definition
        .BYTE   255.,25.;END_LIST-terminated,opcode = DRAW_LINES
        .WORD   400.    ;x1
        .WORD   40.     ;y1
        .WORD   700.    ;x2
        .WORD   -32768. ;ENDLIST terminator value
                        ;Draws a line from [300,10] to [400,40]
                        ;New current position is [400,40]
                        ;(Note that PRO/GIDIS ignores the x2
                        ;coordinate with no corresponding y2.)
```

Example:

```
                        ;Inside a filled-figure definition
                        ; (Filled figure flag is TRUE)
                        ;Current position is not used
                        ; by this instruction
        .BYTE   4.,25.  ;Length=4, opcode for DRAW_LINES
        .WORD   300.    ;x1
        .WORD   200.    ;y1
        .WORD   300.    ;x2
        .WORD   300.    ;y2
                        ;The points [300,200] and [300,300] are
                        ;added to the filled figure table
                        ;new current position is [300,300]
```

6.4 DRAW_REL_LINES

The DRAW_REL_LINES instruction draws a series of connected line segments, starting at the current drawing position. The end point of the first line segment is specified relative to the current position, and the end points of any following line segments are relative to the preceding segment's end point.

Opcode: 26 **Length:** n

Format: DRAW_REL_LINES dx1, dy1, dx2, dy2, ...

dx1 Specifies the X offset (horizontal) value of the first line's end point in GIDIS output space.

dy1 Specifies the Y offset (vertical) value of the first line's end point in GIDIS output space.

dx2 Specifies X offset (horizontal) value of the second line's end point in GIDIS output space.

dy2 Specifies the Y offset (vertical) value of the second line's end point in GIDIS output space.

... Additional coordinate pairs specify relative end points for additional lines.

Status: SUCCESS, provided no last pair arithmetic overflow or filled figure table overflow occurs; on overflow, FAILURE (position is indeterminate).

Notes:

Please refer to the notes for DRAW_LINES.

Example:

```
                        ;Not in a filled figure definition
                        ;(filled figure flag is FALSE)
                        ;Current position is [100,100]
        .BYTE   4.,26.  ;Length=4, opcode for DRAW_REL_LINES
        .WORD   +10.    :dx1
        .WORD   -10.    ;dy1
        .WORD   +30.    ;dx2
        .WORD   +15     ;dy2
                        ;Draw lines from [100,100] to [110,90]
                        ;and from [110,90] to [140,105]
                        ;New current position is [140,105]
```

Example:

```
                        ;Current position is [140,105]
                        ;not in a filled figure definition
        .BYTE   255.,26. ;END-LIST-term.,opcode=DRAW_REL_LINES
        .WORD   10.     ;dx1
        .WORD   -30.    ;dy1
        .WORD   20.     ;dx2
        .WORD   +60.    ;dy2
        .WORD   -32768. ;END_LIST
                        ;Draw lines from [140,105] to [150,75]
                        ;then to [170,135]
                        ;New current position is [170,135]
```

Example:

```
                    ;Inside a filled-figure definition
                    ; (Filled figure flag is TRUE)
                    ;Current position is [100,100]
.BYTE   5.,26.      ;Length=5, opcode for DRAW_REL_LINES
.WORD   100.        ;dx1
.WORD   0.          ;dy1
.WORD   0.          ;dx2
.WORD   100.        ;dy2
.WORD   79.         ;dx3
                    ;Adds the points [200,100] and [200,200]
                    ;to the filled figure table
                    ;New current position is [200,200]
                    ;since there is no dy3, dx3 is ignored
```

6.5 DRAW_ARC

The DRAW_ARC instruction draws a circular arc from the current position around the specified center. Direction of the arc is determined by the sign of the angle parameter. For example, a DRAW_ARC 105,105,-90 instruction would draw a quarter-circle starting at the current position, using location 105,105 as the center of the arc's circle. Because the angle's sign is negative, the arc drawn would be clockwise from the current position.

Opcode: 23 **Length:** 3

Format: DRAW_ARC x, y, angle

x Specifies the X (horizontal) value of the arc's center point in GIDIS output space.

y Specifies the Y (vertical) value of the arc's center point in GIDIS output space.

angle The angle is given in degrees, with a positive value meaning counter-clockwise with respect to the physical screen.

Status: SUCCESS provided angle is within a range of -360 to +360 and there is no filled figure table overflow, otherwise FAILURE.

Notes:

- An angle of zero means no drawing is done; +/- 360 means a full circle. Values greater than 360 (or less than -360) are errors, and no arc is drawn.
- If the filled figure flag is TRUE then, instead of drawing the arc, all internally calculated interpolation points are added to the filled figure table.
- The current position is left at the end of the arc.
- DRAW_ARC is affected by the following global attributes: writing mode, primary color, plane mask, secondary color, pixel size, line texture, and filled figure flag.
- In P/OS 1.7, the PRO/GIDIS interpreter calculates one interpolation point per 10 degrees of arc (or portion), regardless of the size of the circle.
- Full quadrant arcs always end at the exact point expected. Fractional quadrant arcs end at the closest available point, which might not be precisely correct. Multiple fractional quadrant arcs are not guaranteed to end at the exact point predicted by your program. For example, a full circle of a 103 degree arc and a 257 degree arc is not guaranteed to leave the current position exactly where it started.
- DRAW_ARC modifies the bitmap only inside the clipping region.

Example:

```
                        ;Not in a filled figure definition
                        ;(filled figure flag is FALSE)
                        ;Current position is [500,300]
.BYTE   3.,23.          ;Length=3, opcode for DRAW_ARC
.WORD   400.            :x coordinate of center
.WORD   300.            ;y coordinate of center
.WORD   180.            ;180 degrees is one-half a circle
                        ; (counter-clockwise)
                        ;Draws the top half of the circle
                        ;centered at [400,300] with radius 100
                        ;Middle of the arc is [400,200]
                        ;New current position is [300,300]
```

Example:

```
                        ;Inside a filled-figure definition
                        ; (Filled figure flag is TRUE)
                        ;Current position is [500,300]
        .BYTE   3.,23.  ;Length=3, opcode for DRAW_ARC
        .WORD   400.    ;Center is [400,300]
        .WORD   300.
        .WORD   -90.    ;90 degrees = 1 quadrant
                        ;Adds nine interpolation points
                        ;(internally calculated)
                        ;to the filled figure table
                        ;Last point added is [400,400]
                        ;New current position is [400,400]
```

6.6 DRAW_REL_ARC

The DRAW_REL_ARC instruction draws a circular arc from the current position around the center, specified relative to the current position. Length of the arc is specified by an angle in degrees. Direction of the arc is determined by the sign of the angle parameter.

Opcode: 27 **Length:** 3

Format: DRAW_REL_ARC dx, dy, angle

dx Specifies the offset X (horizontal) value of the arc's center point in GIDIS output space.

dy Specifies the offset Y (vertical) value of the arc's center point in GIDIS output space.

angle The angle is given in degrees, with a positive value meaning clockwise with respect to the physical screen.

Status: SUCCESS, provided angle within a range of -360 to +360 and there is no filled figure table overflow or arithmetic overflow, otherwise FAILURE.

Notes:

Please refer to the notes for DRAW_ARC.

Example:

```
                        ;Current position is [400,300]
                        ;Filled figure flag is FALSE
        .BYTE   3.,27.  ;Length=3,opcode is DRAW_REL_ARC
        .WORD   -100.   ;Center is [-100,+30]
        .WORD   +30.    ; relative to current position
        .WORD   -90.    ;90 degrees = one quadrant (clockwise)
                        ;Draws one quadrant from [400,300] to
                        ;[330,430] centered at [300,330]
                        ;New current position is [330,430]
```

CHAPTER 7

FILLED FIGURE INSTRUCTIONS

This chapter contains a detailed description of each filled figure instruction. Table 7-1 lists the instructions covered in the chapter.

Table 7-1: Filled Figure Summary Chart

Opcode/Length	Instruction/Arguments
31/0	BEGIN_FILLED_FIGURE
32/0	END_FILLED_FIGURE

7.1 BEGIN_FILLED_FIGURE

This instruction starts the definition of a closed figure to be filled by the END_FILLED_FIGURE instruction. Filling a figure occurs according to the current attributes in effect. A corresponding END_FILLED_FIGURE instruction is required to actually fill the figure.

Opcode: 31 **Length:** 0

Format: BEGIN_FILLED_FIGURE

Status: SUCCESS

The following PRO/GIDIS instructions are invalid (set status to failure and do nothing) when used between a BEGIN_FILLED_FIGURE instruction and its corresponding END_FILLED_FIGURE.

```
BEGIN_FILLED_FIGURE
DRAW_CHARACTERS
SET_GIDIS_OUTPUT_SPACE
```

SET_OUTPUT_IDS
SET_OUTPUT_VIEWPORT
SET_POSITION
SET_REL_POSITION

Notes:

- The current position is saved in the filled-figure table as are the first and last point of the figure.
- If a filled figure is in progress, this instruction is invalid.
- BEGIN_FILLED_FIGURE sets the filled-figure flag to TRUE.
- Use DRAW_LINES, DRAW_REL_LINES, DRAW_ARC, and DRAW_REL_ARC to enter positions in the filled-figure table.
- No drawing is done by the BEGIN_FILLED_FIGURE function.
- To abort a filled-figure definition, send the INITIALIZE instruction with any argument (including 0). An INITIALIZE 0 instruction aborts a filled-figure definition without affecting anything else.

Example: (See END_FILLED_FIGURE.)

7.2 END_FILLED_FIGURE

The END_FILLED_FIGURE instruction terminates the definition of a closed figure and fills the figure. This instruction is used in conjunction with the BEGIN_FILLED_FIGURE instruction, described in the preceding paragraphs.

Opcode: 32 **Length:** 0

Format: END_FILLED_FIGURE

Status: SUCCESS if there is a corresponding BEGIN_FILLED_FIGURE, otherwise, FAILURE.

Notes:

- The filled-figure table contains the vertices of the figure to be filled.

- The edges of the filled figure are the mathematically ideal lines through the positions in the filled-figure table, but one "extra" pixel is included to ensure that no gap occurs between two adjacent areas. The edge is not guaranteed to be identical to a line drawn (outside of a filled-figure definition) through the same points due to differences in drawing direction and round-off errors.

- The current position is unchanged by the END_FILLED_FIGURE. Current position is left at the last position given. Note that if table overflow occurred, the last point might not be the last vertex in the filled-figure table.

- This instruction modifies the bitmap inside the clipping region.

- If too many positions are specified, only the first 256 vertices are used, and a straight line connects the 256th point with the first point. (256 is the maximum number of vertices in the filled-figure table for P/OS 1.7.)

- Global attributes used in the fill are: primary writing color, secondary writing color, writing mode, plane mask, area texture cell, area cell size, and area size. Unused global attributes are: pixel size.

- Complement and complement-negate writing modes can give unexpected results when filled figure areas overlap or abut.

Example:

```
        .BYTE   2.,29.  ;Length=2,opcode for SET_POSITION
        .WORD   100.    ;Current position
        .WORD   100.    ;  is [100,100]
        .BYTE   0.,31.  ;Length=0,opcode for BEGIN_FILLED_FIGURE
                        ;Filled-figure table now has [100,100]
        .BYTE   4.,26.  ;Length=4,opcode for DRAW_REL_LINES
        .WORD   +100.   ;dx1
        .WORD   +0.     ;dy1
        .WORD   +0.     ;dx2
        .WORD   +100.   ;dy2
                        ;Adds points [200,100] and [200,200] to
                        ;  the filled-figure table
        .BYTE   255.,25. ;END_LIST terminated,opcode = DRAW_LINES
        .WORD   100.    ;x1
        .WORD   200.    ;x2
        .WORD   -32768. ;END_LIST
                        ;Adds point [100,200] to
                        ;  the filled-figure table
        .BYTE   9.,32.  ;Length=0,opcode for END_FILLED_FIGURE
                        ;The area defined by [100,100],
                        ;  [200,100], [200,200], [100,200], and
                        ;  [100,100] (a square) is filled with
                        ;  the current area texture (modified by
                        ;  whatever current global attributes are
                        ;  in effect.)
```

CHAPTER 8

TEXT INSTRUCTIONS

This chapter contains a detailed description of each text instruction. Table 8-1 lists the instructions covered in the chapter.

Table 8-1: Text Instructions Summary Chart

Opcode/Length	Instruction/Arguments	
38/1	SET_ALPHABET	alphabet-number
46/4	CREATE_ALPHABET	width, height, extent, width-type
34/n	LOAD_CHARACTER_CELL	char-index, width, d0, dl,...,dn
43/1	SET_CELL_RENDITION	cell-rendition
44/1	SET_CELL_ROTATION	dx, dy
65/1	SET_CELL_OBLIQUE	dx, dy
45/2	SET_CELL_UNIT_SIZE	width, height
40/2	SET_CELL_DISPLAY_SIZE	width, height
42/1	SET_CELL_MOVEMENT_MODE	flag
41/2	SET_CELL_EXPLICIT_MOVEMENT	dx, dy
35/n	DRAW_CHARACTERS	char-indexl, char-index2, ...

8.1 SET_ALPHABET

The SET_ALPHABET instruction selects a specific alphabet to be used as the current alphabet. Any alphabet-related operations, except as noted, act on the currently selected alphabet.

Opcode: 38 **Length:** 1

Format: SET_ALPHABET alphabet

alphabet is an integer value in a range of 0 to 15 that specifies the current alphabet.

Status: SUCCESS if the alphabet number is valid (from 0 to 15), FAILURE otherwise.

Notes:

- Valid alphabet numbers for the Professional (P/OS 1.7) are 0 through 15. Alphabet 0 is the DEC Multinational Character Set.
- Alphabet 0, the DEC Multinational font, is a read-only alphabet and cannot be changed through PRO/GIDIS. Control characters (indices 0 through 31 and 128 through 159 decimal) are not included, and, if requested with DRAW_CHARACTERS, appear as an error ("blob" or "checkerboard") character. Some indices are resevered and appear as a reversed question mark.
- The current alphabet can be reset to the power-on default (0) by the PRO/GIDIS INITIALIZE instruction's text subsystem option.
- No drawing is done by the SET_ALPHABET instruction.

Example:

```
        .BYTE   1.,38.  ;Length=1, opcode for SET_ALPHABET
        .WORD   2.      ;Selects alphabet #2 as current alphabet
```

8.2 CREATE_ALPHABET

CREATE_ALPHABET erases the current alphabet and reserves resources for an alphabet with the specified storage size (width by height by extent).

Opcode: 46 **Length:** 4

Format: CREATE_ALPHABET width, height, extent, width-type

width is an integer in the range (0 to 16) that specifies the number of horizontal bits in a character pattern.

height is an integer in the range (0 to 16) that specifies the number of vertical bits in a character pattern.

extent is an unsigned integer that specifies the number of characters in the alphabet. Character indices can range from 0 to extent - 1.

width-type is reserved for future use. For the Professional, this value must be 0.

Status: SUCCESS if width and height are greater than or equal to zero, the width type is zero, the current alphabet number is a valid user definable alphabet (not 0), and there are sufficient resources to create the alphabet; FAILURE otherwise.

Notes:

- Character indices are 16-bit numbers and do not necessarily correspond to DEC Multinational codes (except for alphabet 0).
- For the Professional, all characters in an alphabet have the same storage size.
- To reclaim alphabet space, CREATE_ALPHABET 0,0,0,0 erases the existing alphabet and returns all of the alphabet's resources.

Example:

```
                        ;Current alphabet is alphabet number 2
.BYTE   4.,46.          ;Length=4, opcode for CREATE_ALPHABET
.WORD   10.             ;width
.WORD   16.             ;height
.WORD   32.             ;extent
.WORD   0.              ;width-type (Note:  MUST BE ZERO)
                        ;Erases alphabet 2 and
                        ;  creates an empty alphabet with
                        ;  the specified characteristics
```

8.3 LOAD_CHARACTER_CELL

The LOAD_CHARACTER_CELL instruction loads a character cell from the raster data given as parameters. This instruction acts on the currently selected alphabet.

Opcode: 34 **Length:** variable

Format: LOAD_CHARACTER_CELL char-index, width, d0, dl,...,dn

char-index The index of the character cell to be loaded. This value must be in a range of 0 to extent - 1, where extent is the total character count for the alphabet. .b.i-12; width The width value must be in a range of 0 to the width value given with the CREATE_ALPHABET instruction that established the alphabet. (P/OS 1.7 does not use this value. However, it still must be in the appropriate range.)

d0, dl,... Zero to 16 words of data to be loaded into the character cell. The top character cell row is loaded from the first data word (d0), the second row from the second data word (dl), and so forth.

Status: SUCCESS if character index is in a range of 0 to extent - 1, and width is in a range of 0 to alphabet width; otherwise, FAILURE.

Notes:

- The leftmost pixel in a row comes from the low-order bit in the appropriate data word.
- Characters cannot be loaded into any alphabet with an extent value of zero.

Example:

```
                        ;Alphabet 2 has width of 4, height of 5,
                        ;  and extent of 10
        .BYTE   6.,46.  ;Length=6, opcode for LOAD_CHARACTER_CELL
        .WORD   9.      ;Character index (last cell in alphabet)
        .WORD   4.      ;Width
        .WORD   ^B1011  ;Pattern:        ON   ON   OFF  ON
        .WORD   ^B1010  ;                OFF  ON   OFF  ON
        .WORD   ^B1000  ; (Note the      OFF  OFF  OFF  ON
        .WORD   ^B0001  ; bit reversal)  ON   OFF  OFF  OFF
                        ;                OFF  OFF  OFF  OFF
                        ;Last row not given,cleared automatically
```

8.4 SET_CELL_RENDITION

The SET_CELL_RENDITION instruction specifies the renditions to be added to characters written following execution of this instruction. Renditions are variations on characters that could be performed without selecting a new alphabet, yet are not related to writing colors or writing modes. Renditions defined for the Professional are: backslant and italics.

Opcode: 43 **Length:** 1

Format: SET_CELL_RENDITION mask-value

mask-value A mask representing one or more of the cell renditions. The mask value corresponds to one or more rendition bits set. (See Figure 8-1).

Status: SUCCESS if the requested rendition is supported by the implementation, FAILURE otherwise.

This instruction specifies a bit-mask value indicating which renditions are to be added. Certain mask bits are reserved for future PRO/GIDIS enhancements. Figure 8-1 illustrates the defined rendition bits that can be selected.

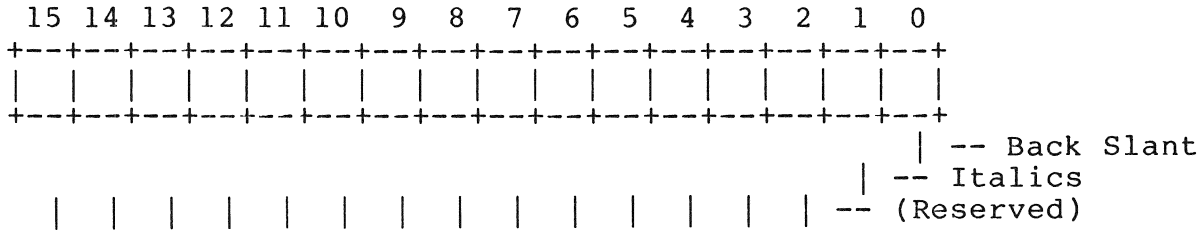


Figure 8-1: SET_CELL_RENDITION Bit Mask

Notes:

- Setting both backslant and italics (mask value ^B00011) is an error. In this case, rendition is undefined.
- Italics and backslant (reverse italics) are generated by oblique angles of -23 and +23 degrees, respectively. Requesting italic or backslant rendition overrides any specific oblique angle requested previously. Requesting an oblique angle after an italic rendition request nullifies the italic request.

Example:

```
.BYTE   1.,43.  ;Length=1, opcode for SET_CELL_RENDITION
.WORD   2.      ;Requests italics rendition
```

8.5 SET_CELL_ROTATION

The SET_CELL_ROTATION instruction defines the angle of rotation at which the character is to be displayed. The character is rotated about the current position (upper left corner of the display cell) to the available rotation angle nearest to the angle specified. Table 8-2 shows the angles available.

Opcode: 44 **Length:** 1

Format: SET_CELL_ROTATION angle

angle The requested angle in degrees. A positive angle value indicates counter-clockwise from normal text.

Status: SUCCESS

Notes:

- No drawing takes place when the SET_CELL_ROTATION instruction executes.

Example:

```
.BYTE   1.,44.  ;Length=1, opcode for SET_CELL_ROTATION
.WORD   -90.    ;Text to go down the screen
```

Requested	Actual
0	0
45	51
90	90
135	129
180	180
225	232
270	270
315	309

Table 8-2: Cell Rotation Angles

8.6 SET_CELL_OBLIQUE

An obliqued character cell is slanted away from a true rectanglular orientation. The SET_CELL_OBLIQUE instruction specifies the angle from vertical of the side of the display cell. A positive angle parameter results in a back-slanted character; a negative angle yields a forward, italic-type slant. Oblique angles always are set for unrotated characters. The oblique angle rotates with the specified rotation.

Opcode: 65 **Length:** 1

Format: SET_CELL_OBLIQUE angle

angle The requested angle in degrees. A positive angle value indicates a backward slant; a negative angle indicates a forward slant.

Status: SUCCESS

Notes:

- No drawing takes place when SET_CELL_OBLIQUE executes.
- PRO/GIDIS (P/OS V1.7) uses an approximation algorithm to oblique text which is accurate for small angles (less than ten degrees), less accurate for medium angles (up to 25 degrees), and inaccurate for large angles. Increasing the argument value will always increase the text oblique.

Example:

```
.BYTE   1.,65.  ;Length=1, opcode for SET_CELL_OBLIQUE
.WORD   -23.    ;Regular italics (slanting right)
```

8.7 SET_CELL_UNIT_SIZE

The SET_CELL_UNIT_SIZE instruction specifies the actual size of character(s) displayed; the size is specified in GIDIS output space. The stored character pattern is mapped completely to the unit size.

Opcode: 45 **Length:** 2

Format: SET_CELL_UNIT_SIZE width, height

width A width value in GIDIS output space. The width must be greater than zero.

height A height value in GIDIS output space. The height must be greater than zero.

Status: SUCCESS if width and height are greater than zero, FAILURE otherwise

For PRO/GIDIS, the unit size is restricted to an integral multiple of the character pattern (storage size). PRO/GIDIS uses the largest available size that is not larger than the size specified. If no available size is small enough, the smallest

available is used. Refer to the SET_CELL_DISPLAY_SIZE to define the size of the area in which the cell unit is displayed.

Notes:

- The requested unit size does not change when the current alphabet changes, but the adjustment described above is recalculated.
- The unit cell and the display cell always are aligned at their upper-left corners.
- No drawing is done when the SET_CELL_UNIT_SIZE executes.

Example:

```
.BYTE   1.,45.  ;Length=1, opcode for SET_CELL_UNIT_SIZE
.WORD   10.     ;Width
.WORD   30.     ;Height
```

8.8 SET_CELL_DISPLAY_SIZE

SET_CELL_DISPLAY_SIZE defines a character's display cell, the rectangular area of the display modified when a character is drawn.

Opcode: 40 **Length:** 2

Format: SET_CELL_DISPLAY_SIZE width, height

width A width value in GIDIS output space.

height A height value in GIDIS output space.

Status: SUCCESS

Notes:

- The origin of the display cell is always the upper left corner of the cell and is aligned with the unit cell at that corner.
- The unit cell can be larger than the display cell; if so, the character is clipped to the display cell. If the unit cell is smaller than the display cell, then all of the character is drawn and the rest of the display cell is treated as if the character pattern specified OFF.

- Negative values in width or height produce a mirroring in X and Y, respectively. This mirroring or reflection always is done about the origin (the upper-left corner of the cell). Implied movement always goes across the display cell, so implied movement for a display cell mirrored in X is in the opposite direction from the rotation angle.
- No drawing is done when the SET_CELL_DISPLAY_SIZE executes.
- If either the width or the height is zero, no portion of a character is drawn.

Example:

```
.BYTE   2.,40.  ;Length=2, opcode=SET_CELL_DISPLAY_SIZE
.WORD   12.     ;Width
.WORD   28.     ;Height
```

8.9 SET_CELL_MOVEMENT_MODE

The SET_CELL_MOVEMENT_MODE instruction specifies the manner in which the current position moves after each character is drawn by a DRAW_CHARACTERS instruction.

Opcode: 42 **Length:** 1

Format: SET_CELL_MOVEMENT_MODE flag

flag specifies one of the following movement modes:

0	Explicit cell movement, local symmetry
2	Explicit and implied movement, local symmetry

(All other values are reserved.)

Status: SUCCESS if Flag is 0 or 2, FAILURE otherwise.

Notes:

- When using local symmetry, the current position after a DRAW_CHARACTERS instruction could be different from that calculated by your program. It is suggested that all DRAW_CHARACTERS instructions be followed with a SET_POSITION instruction or a REQUEST_POSITION instruction unless the exact end of the string is the desired position for the next instruction.

- No drawing occurs when the SET_CELL_MOVEMENT_MODE instruction executes.
- If the flag specified is not 0 or 2, the flag does not change from its previous state.

Example:

```
.BYTE   1.,42.  ;Length=1, opcode=SET_CELL_MOVEMENT_MODE
.WORD   0.      ;Explicit local mode symmetry
```

8.10 SET_CELL_EXPLICIT_MOVEMENT

The SET_CELL_EXPLICIT_MOVEMENT instruction specifies the relative distance that the current position is to move after a character is drawn. The relative distance is specified in GIDIS output space.

Opcode: 41 **Length:** 2

Format: SET_CELL_EXPLICIT_MOVEMENT dx, dy

dx Specifies the horizontal distance in GIDIS output space to move the current position.

dy Specifies the vertical distance in GIDIS output space to move the current position.

Status: SUCCESS

The explicit value is the total movement when the cell movement mode is 0. The value represents the inter-character spacing when the mode is 2. (Refer to the preceding SET_CELL_MOVEMENT_MODE description for details on implied movement.) The default mode is 2, default explicit movement is [0,0].

Notes:

- No drawing occurs when the SET_CELL_EXPLICIT_MOVEMENT instruction executes.
- When using local symmetry, the current position after a DRAW_CHARACTERS instruction could be different from that calculated by your program. It is suggested that all DRAW_CHARACTERS instructions be followed with a SET_POSITION instruction or a REQUEST_POSITION instruction unless the exact end of the string is the desired position for the next instruction.

- The explicit movement is used exactly as specified. It is not adjusted according to the rotation angle.

Example:

```
.BYTE   2.,41.  ;Length=2,SET_CELL_EXPLICIT_MOVEMENT
.WORD   12.     ;dx
.WORD   0.      ;dy
```

8.11 DRAW_CHARACTERS

The DRAW_CHARACTERS instruction displays each of the characters specified by each character index in the parameter list. The characters are taken from the currently selected alphabet.

Opcode: 35 **Length:** n

Format: DRAW_CHARACTERS char-index, ...

char-index an unsigned 16-bit word

Status: SUCCESS if the current alphabet number is valid and if the last character-index is valid in the alphabet, FAILURE otherwise.

Characters can be specified either in a counted argument list (with the count supplied with the opcode word) or in an END_LIST terminated list with 255 in the opcode word. The rules for variable-length argument lists for PRO/GIDIS are described in Chapter 2 of this manual.

Notes:

- The current position is updated after each character display, according to the cell movement controls. (See the descriptions of the SET_CELL_MOVEMENT_MODE and SET_CELL_EXPLICIT_MOVEMENT instructions.)
- This instruction uses implied and explicit cell movement, unit and display size, cell rotation, rendition mask, current alphabet, and writing mode.
- DRAW_CHARACTERS modifies the bitmap (only inside the clipping region) and the current position.

Example:

```
                        ;Current alphabet = 0 (DEC Multinational)
                        ;Unit size, display size, etc. are
                        ;  set up properly
        .BYTE   3.,35.  ;Length=3, opcode for DRAW_CHARACTERS
                        ;Counted form
        .WORD   65.     ; 'A'
        .WORD   66.     ; 'B'
        .WORD   67.     ; 'C'
```

Example:

```
                        ;Current alphabet = 1 (user-defined)
                        ;Unit size, display size, etc. are
                        ;  set up properly
        .BYTE   255.,35. ;END_LIST Term.,opcode = DRAW_CHARACTERS
        .WORD   0.
        .WORD   13.
        .WORD   7.
        .WORD   45.
        .WORD   -32768. ;END_LIST
                        ;Displays 4 characters from alphabet 1,
                        ;  which are user-defined characters
```

CHAPTER 9

AREA OPERATION INSTRUCTIONS

This chapter contains a detailed description of each of the area operation instructions. Table 9-1 lists the instructions covered in the chapter.

Table 9-1: Area Operations Summary Chart

Opcode/Length	Instruction/Arguments
48/0	ERASE_CLIPPING_REGION
141/6	PRINT_SCREEN x, y, width, height, hx, hy

9.1 ERASE_CLIPPING_REGION

The ERASE_CLIPPING_REGION instruction sets every pixel inside the current output clipping region to the secondary color that is currently in effect. This instruction erases data currently displayed within the region. ERASE_CLIPPING_REGION can be used to clear a rectangular area without implying the end of a picture or the beginning of a new picture.

Opcode: 48 **Length:** 0

Format: ERASE_CLIPPING_REGION

Status: SUCCESS

Notes:

- To erase the entire screen, use the END_PICTURE/NEW_PICTURE combination, not the ERASE_CLIPPING_REGION.
- This instruction sets the bitmap in the output clipping region.
- The current writing mode, current area texture, and primary color do not affect this instruction.

Example:

```
.BYTE   0.,48.  ;Length=0, opcode=ERASE_CLIPPING_REGION
```

9.2 PRINT_SCREEN

The PRINT_SCREEN instruction sends a specified portion of the bitmap to a printer connected to the printer port, generally an LA50 or LA100. Parameters permit you to indicate the portion of the data to be printed and the relative point on the paper for the printer to start printing.

Opcode: 141 **Length:** 6

Format: PRINT_SCREEN x, y, width, height, hx, hy

x Specifies the upper left horizontal point of the bitmap data to be printed. This location is given in GIDIS output units.

y Specifies the upper left vertical point of the bitmap data to be printed. This location is given in GIDIS output units.

width Width of the area to be printed, specified in GIDIS output units

height Height of the area to be printed, specified in GIDIS output units

hx Specifies the horizontal offset from the current printhead location to begin printing the screen data. This point corresponds to the upper left horizontal coordinate (x) of the data to be printed. The value is given in GIDIS Output Space coordinates.

hy Specifies the vertical offset from the current printhead location to begin printing the screen data. This point

corresponds to the upper left vertical coordinate (y) of the data to be printed. The value is given in GOS coordinates.

Status: SUCCESS

Notes:

- In a single plane system, a pixel value of 0 is mapped to a skip (leaves paper white) and a 1 is mapped to a strike (prints on the paper). On multi-plane systems, the monochrome value of the color map is tested. If 0, the point is skipped (white), if not zero, the point prints.
- If the printer port does not have an LA50 or LA100 connected, nothing occurs.

Example:

```
.BYTE   6.,141. ;Length=6, opcode for PRINT_SCREEN
.WORD   100.    ;Upper left bitmap corner
.WORD   100.    ;   is [100,100]
.WORD   400.    ;Data to be printed is 400 units wide
.WORD   200.    ;   by 200 units high
.WORD   0.      ;Begin printing at current printhead
.WORD   0.      ;   location
```

CHAPTER 10

REPORT HANDLING

This chapter contains a detailed description of each of the report handling instructions. Table 10-1 lists the instructions and report tags covered in the chapter.

Table 10-1: Report Handling Summary Chart

Opcode/Length	Instruction	Tag Name	Data Record
55/0	REQUEST_CURRENT_POSITION	CURRENT_POSITION_REPORT	x, y
58/0	REQUEST_STATUS	STATUS_REPORT	code
54/0	REQUEST_CELL_STANDARD	CELL_STANDARD_REPORT	uw, uh, dw, dh

10.1 REQUEST_CURRENT_POSITION

The REQUEST_CURRENT_POSITION instruction reports the absolute location of the current position. The current position is the display location at which the next character, line, or arc would be drawn.

Opcode: 55 **Length:** 0

Format: REQUEST_CURRENT_POSITION

Status: SUCCESS

The reported information takes the following form:

CURRENT_POSITION_REPORT, x, y

The X and Y values reported are the PRO/GIDIS output space coordinates of the current position.

Notes:

- The current position is not necessarily the same as the last position given to SET_POSITION or DRAW_LINES; DRAW_CHARACTERS and DRAW_ARC instructions also move the current position.
- The REQUEST_CURRENT_POSITION instruction is most useful following a DRAW_ARC or a DRAW_CHARACTERS (local symmetry), since your program cannot determine precisely where PRO/GIDIS leaves the current position after these instructions.

Example:

```
        .BYTE   0.,55.  ;Lgth=0, opcode=REQUEST_CURRENT_POSITION
                        ;
                        ;This instruction causes the following
                        ;  report to be placed in the report
                        ;  queue if there is sufficient room.
                        ;
                        ; Byte  2. (Data words following)
                        ; Byte  1. (Current Pos. Rpt. Tag)
                        ; Word  x  (PRO/GIDIS coordinates
                        ; Word  y     for current position)
                        ;
                        ;The report can be read using a QIO
                        ;  with the function code IO.RSD.  Refer
                        ;  to Chapter 2 for more information.
```

10.2 REQUEST_STATUS

REQUEST_STATUS reports the success or failure of a PRO/GIDIS instruction. All PRO/GIDIS instructions set the status variable.

Opcode: 58 **Length:** 0

Format: REQUEST_STATUS

Status: SUCCESS

Status is reported in the following format:

```
        STATUS_REPORT,  status
```

where the low-order bit of the variable status is either 1 indicating SUCCESS or 0 indicating FAILURE.

Notes:

- No other codes are defined. (Codes other than 0 or 1 are reserved for future use.)
- FAILURE status is not saved. If your program needs information about the success or failure of every instruction, you must place a REQUEST_STATUS instruction after each PRO/GIDIS instruction.
- Testing is recommended only following major PRO/GIDIS instructions, such as CREATE_ALPHABET.

Example:

```
                    ;assumes previous instruction failed
        .BYTE   0.,58.  ;Lgth=0, opcode=REQUEST_STATUS
                        ;
                        ; Byte  1. (Data words following)
                        ; Byte  4. (Current Stat. Rpt. Tag)
                        ; Word  0  (FAILURE   status)
                        ;For additional examples, refer to
                        ;Chapter 2.
```

10.3 REQUEST_CELL_STANDARD

The REQUEST_CELL_STANDARD instruction reports the current unit cell and display cell sizes.

Opcode: 54 **Length:** 0

Format: REQUEST_CELL_STANDARD

Status: SUCCESS

The report takes the following form:

```
CELL_STANDARD_REPORT, unit-wd, unit-ht,
                      display-wd, display-ht
```

where unit-wd and unit-ht are the unit cell width and height of the standard size character in GIDIS space. Display-wd and display-ht are the display cell width and height.

Notes:

- This instruction takes into account the storage size of the current alphabet and the character rotation currently in effect. The standard size for alphabet 0 (DEC Multinational) is not necessarily the same as the standard size for alphabet 1.
- Rounding could take place converting from device coordinates to GIDIS space. If your program requests 'n' times the size of the standard, the characters actually formed might not be precisely 'n' times the standard.

Example:

```
        .BYTE   0.,54.  ;Lgth=0, opcode=REQUEST_CELL_STANDARD
                        ;
                        ; Byte  4.   (Data words following)
                        ; Byte  5.   (Cell Standard Rpt. Tag)
                        ; Word  9.   (Unit width)
                        ; Word  20.  (Unit height)
                        ; Word  8.   (Display width)
                        ; Word  20.  (Display height)
                        ;
                        ;For additional examples, refer to
                        ;Chapter 2.
```

APPENDIX A

PRO/GIDIS INSTRUCTION SUMMARIES

This chapter contains a PRO/GIDIS instruction summary in three different orders: by function, in ascending opcode order, and in alphabetic order. The opcode and parameter block length values are shown as a word value as well as separate byte values.

A.1 INSTRUCTIONS GROUPED BY FUNCTION

Opcode	Length	Opcode Word	Instruction and Arguments	Function
0	0	0	NOP -	control
1	1	257	INITIALIZE mask	control
5	6	1286	SET_OUTPUT_CURSOR a, c, w, h, ox, oy	control
6	0	1536	NEW_PICTURE -	control
24	0	6144	END_PICTURE -	control
28	0	7168	FLUSH_BUFFER -	control
128	0	-32768	END_LIST -	control
4	4	1028	SET_OUTPUT_CLIPPING_REGION x, y, w, h	transform
9	4	2308	SET_GIDIS_OUTPUT_SPACE x, y, w, h	transform
12	2	3074	SET_OUTPUT_IDS w, h	transform
13	4	3332	SET_OUTPUT_VIEWPORT x, y, w, h	transform
3	2	770	SET_AREA_TEXTURE_SIZE w, h	attributes
14	2	3586	SET_AREA_TEXTURE a, c	attributes
15	1	3841	SET_SECONDARY_COLOR color	attributes
16	6	4102	SET_COLOR_MAP_ENTRY m, color, r, g, b,	mono "
17	3	4355	SET_LINE_TEXTURE length, pattern, size	attributes
19	4	4868	SET_PIXEL_SIZE w, h, ox, oy	attributes
20	1	5121	SET_PLANE_MASK mask	attributes
21	1	5377	SET_PRIMARY_COLOR color	attributes
22	1	5633	SET_WRITING_MODE mode	attributes
69	2	17666	SET_AREA_CELL_SIZE w, h	attributes

23	3	5891	DRAW_ARC x, y, angle	drawing
25	N	6400+N	DRAW_LINES x, y, ...	drawing
26	N	6656+N	DRAW_REL_LINES dx, dy, ...	drawing
27	3	6915	DRAW_REL_ARC dx, dy, angle	drawing
29	2	7426	SET_POSITION x, y	drawing
30	2	7682	SET_REL_POSITION dx, dy	drawing
31	0	7936	BEGIN_FILLED_FIGURE -	filled figures
32	0	8192	END_FILLED_FIGURE -	filled figures
33	2	8450	BEGIN_DEFINE_CHARACTER c, w	text
34	N	8704+N	LOAD_CHARACTER_CELL c, w, d0, ... d15	text
35	N	8960+N	DRAW_CHARACTERS c, ...	text
36	0	9216	END_DEFINE_CHARACTER -	text
38	1	9729	SET_ALPHABET a	text
40	2	10242	SET_CELL_DISPLAY_SIZE w, h	text
41	2	10498	SET_CELL_EXPLICIT_MOVEMENT dx, dy	text
42	1	10753	SET_CELL_MOVEMENT_MODE flag	text
43	1	11009	SET_CELL_RENDITION flags	text
44	1	11265	SET_CELL_ROTATION angle	text
45	2	11522	SET_CELL_UNIT_SIZE w, h	text
46	4	11780	CREATE_ALPHABET w, h, extent, type	text
65	1	16641	SET_CELL_OBLIQUE angle	text
48	0	12288	ERASE_CLIPPING_REGION -	area
141	6	-29434	PRINT_SCREEN x, y, w, h, hx, hy	area
54	0	13824	REQUEST_CELL_STANDARD -	reports
55	0	14080	REQUEST_CURRENT_POSITION -	reports
58	0	14848	REQUEST_STATUS -	reports

A.2 INSTRUCTIONS IN OPCODE ORDER

Opcode	Length	Opcode Word	Instruction and Arguments	Function
0	0	0	NOP -	control
1	1	257	INITIALIZE mask	control
3	2	770	SET_AREA_TEXTURE_SIZE w, h	attributes
4	4	1028	SET_OUTPUT_CLIPPING_REGION x, y, w, h	transform
5	6	1286	SET_OUTPUT_CURSOR a, c, w, h, ox, oy	control
6	0	1536	NEW_PICTURE -	control
9	4	2308	SET_GIDIS_OUTPUT_SPACE x, y, w, h	transform
12	2	3074	SET_OUTPUT_IDS w, h	transform
13	4	3332	SET_OUTPUT_VIEWPORT x, y, w, h	transform
14	2	3586	SET_AREA_TEXTURE a, c	attributes
15	1	3841	SET_SECONDARY_COLOR color	attributes

```
16   6   4102     SET_COLOR_MAP_ENTRY m, color, r, g, b, mono   "
17   3   4355     SET_LINE_TEXTURE length, pattern, size attributes
19   4   4868     SET_PIXEL_SIZE w, h, ox, oy            attributes
20   1   5121     SET_PLANE_MASK mask                    attributes
21   1   5377     SET_PRIMARY_COLOR color                attributes
22   1   5633     SET_WRITING_MODE mode                  attributes
23   3   5891     DRAW_ARC x, y, angle                   drawing
24   0   6144     END_PICTURE -                          control
25   N   6400+N   DRAW_LINES x, y, ...                   drawing
26   N   6656+N   DRAW_REL_LINES dx, dy, ...             drawing
27   3   6915     DRAW_REL_ARC dx, dy, angle             drawing
28   0   7168     FLUSH_BUFFER -                         control
29   2   7426     SET_POSITION x, y                      drawing
30   2   7682     SET_REL_POSITION dx, dy                drawing
31   0   7936     BEGIN_FILLED_FIGURE -                  filled figures
32   0   8192     END_FILLED_FIGURE -                    filled figures
33   2   8450     BEGIN_DEFINE_CHARACTER c, w            text
34   N   8704+N   LOAD_CHARACTER_CELL c, w, d0, ... d15  text
35   N   8960+N   DRAW_CHARACTERS c, ...                 text
36   0   9216     END_DEFINE_CHARACTER -                 text
38   1   9729     SET_ALPHABET a                         text
40   2   10242    SET_CELL_DISPLAY_SIZE w, h             text
41   2   10498    SET_CELL_EXPLICIT_MOVEMENT dx, dy      text
42   1   10753    SET_CELL_MOVEMENT_MODE flag            text
43   1   11009    SET_CELL_RENDITION flags               text
44   1   11265    SET_CELL_ROTATION angle                text
45   2   11522    SET_CELL_UNIT_SIZE w, h                text
46   4   11780    CREATE_ALPHABET w, h, extent, type     text
48   0   12288    ERASE_CLIPPING_REGION -                area
54   0   13824    REQUEST_CELL_STANDARD -                reports
55   0   14080    REQUEST_CURRENT_POSITION -             reports
58   0   14848    REQUEST_STATUS -                       reports
65   1   16641    SET_CELL_OBLIQUE angle                 text
69   2   17666    SET_AREA_CELL_SIZE w, h                attributes
128  0   -32768   END_LIST -                             control
141  6   -29434   PRINT_SCREEN x, y, w, h, hx, hy        area
```

A.3 INSTRUCTIONS IN ALPHABETIC ORDER

```
Opcode Length Opcode Word   Instruction and Arguments          Function
-------------------------------------------------------------------------
.LT
 33   2   8450     BEGIN_DEFINE_CHARACTER c, w            text
 31   0   7936     BEGIN_FILLED_FIGURE -                  filled figures
 46   4   11780    CREATE_ALPHABET w, h, extent, type     text
 23   3   5891     DRAW_ARC x, y, angle                   drawing
```

35	N	8960+N	DRAW_CHARACTERS c, ...	text
25	N	6400+N	DRAW_LINES x, y, ...	drawing
27	3	6915	DRAW_REL_ARC dx, dy, angle	drawing
26	N	6656+N	DRAW_REL_LINES dx, dy, ...	drawing
36	0	9216	END_DEFINE_CHARACTER -	text
32	0	8192	END_FILLED_FIGURE -	filled figures
128	0	-32768	END_LIST -	control
24	0	6144	END_PICTURE -	control
48	0	12288	ERASE_CLIPPING_REGION -	area
28	0	7168	FLUSH_BUFFER -	control
1	1	257	INITIALIZE mask	control
34	N	8704+N	LOAD_CHARACTER_CELL c, w, d0, ... d15	text
6	0	1536	NEW_PICTURE -	control
0	0	0	NOP -	control
141	6	-29434	PRINT_SCREEN x, y, w, h, hx, hy	area
54	0	13824	REQUEST_CELL_STANDARD -	reports
55	0	14080	REQUEST_CURRENT_POSITION -	reports
58	0	14848	REQUEST_STATUS -	reports
38	1	9729	SET_ALPHABET a	text
69	2	17666	SET_AREA_CELL_SIZE w, h	attributes
14	2	3586	SET_AREA_TEXTURE a, c	attributes
3	2	770	SET_AREA_TEXTURE_SIZE w, h	attributes
40	2	10242	SET_CELL_DISPLAY_SIZE w, h	text
41	2	10498	SET_CELL_EXPLICIT_MOVEMENT dx, dy	text
42	1	10753	SET_CELL_MOVEMENT_MODE flag	text
65	1	16641	SET_CELL_OBLIQUE angle	text
43	1	11009	SET_CELL_RENDITION flags	text
44	1	11265	SET_CELL_ROTATION angle	text
45	2	11522	SET_CELL_UNIT_SIZE w, h	text
16	6	4102	SET_COLOR_MAP_ENTRY m, color, r, g, b,	mono "
9	4	2308	SET_GIDIS_OUTPUT_SPACE x, y, w, h	transform
17	3	4355	SET_LINE_TEXTURE length, pattern, size	attributes
4	4	1028	SET_OUTPUT_CLIPPING_REGION x, y, w, h	transform
5	6	1286	SET_OUTPUT_CURSOR a, c, w, h, ox, oy	control
12	2	3074	SET_OUTPUT_IDS w, h	transform
13	4	3332	SET_OUTPUT_VIEWPORT x, y, w, h	transform
19	4	4868	SET_PIXEL_SIZE w, h, ox, oy	attributes
20	1	5121	SET_PLANE_MASK mask	attributes
29	2	7426	SET_POSITION x, y	drawing
21	1	5377	SET_PRIMARY_COLOR color	attributes
30	2	7682	SET_REL_POSITION dx, dy	drawing
15	1	3841	SET_SECONDARY_COLOR color	attributes
22	1	5633	SET_WRITING_MODE mode	attributes

A.4 REPORT TAGS

Opcode Length	Opcode Word	Report Tag	Arguments	Function
1	2	258	CURRENT_POSITION_REPORT x, y	report tags
4	1	1025	STATUS_REPORT code	report tags
5	4	1284	CELL_STANDARD_REPORT uw, uh, dw, dh	report tags

APPENDIX B

DEC MULTINATIONAL CHARACTER SET

DEC Multinational Character Set (C0 and GL Codes)

ROW	b8 b7 b6 b5 b4 b3 b2 b1	0	1	2	3	4	5	6	7
0	0 0 0 0	NUL 0 0 0	DLE 20 16 10	SP 40 32 20	0 60 48 30	@ 100 64 40	P 120 80 50	` 140 96 60	p 160 112 70
1	0 0 0 1	SOH 1 1 1	DC1 (XON) 21 17 11	! 41 33 21	1 61 49 31	A 101 65 41	Q 121 81 51	a 141 97 61	q 161 113 71
2	0 0 1 0	STX 2 2 2	DC2 22 18 12	" 42 34 22	2 62 50 32	B 102 66 42	R 122 82 52	b 142 98 62	r 162 114 72
3	0 0 1 1	ETX 3 3 3	DC3 (XOFF) 23 19 13	# 43 35 23	3 63 51 33	C 103 67 43	S 123 83 53	c 143 99 63	s 163 115 73
4	0 1 0 0	EOT 4 4 4	DC4 24 20 14	$ 44 36 24	4 64 52 34	D 104 68 44	T 124 84 54	d 144 100 64	t 164 116 74
5	0 1 0 1	ENQ 5 5 5	NAK 25 21 15	% 45 37 25	5 65 53 35	E 105 69 45	U 125 85 55	e 145 101 65	u 165 117 75
6	0 1 1 0	ACK 6 6 6	SYN 26 22 16	& 46 38 26	6 66 54 36	F 106 70 46	V 126 86 56	f 146 102 66	v 166 118 76
7	0 1 1 1	BEL 7 7 7	ETB 27 23 17	' 47 39 27	7 67 55 37	G 107 71 47	W 127 87 57	g 147 103 67	w 167 119 77
8	1 0 0 0	BS 10 8 8	CAN 30 24 18	(50 40 28	8 70 56 38	H 110 72 48	X 130 88 58	h 150 104 68	x 170 120 78
9	1 0 0 1	HT 11 9 9	EM 31 25 19	) 51 41 29	9 71 57 39	I 111 73 49	Y 131 89 59	i 151 105 69	y 171 121 79
10	1 0 1 0	LF 12 10 A	SUB 32 26 1A	* 52 42 2A	: 72 58 3A	J 112 74 4A	Z 132 90 5A	j 152 106 6A	z 172 122 7A
11	1 0 1 1	VT 13 11 B	ESC 33 27 1B	+ 53 43 2B	; 73 59 3B	K 113 75 4B	[133 91 5B	k 153 107 6B	{ 173 123 7B
12	1 1 0 0	FF 14 12 C	FS 34 28 1C	, 54 44 2C	< 74 60 3C	L 114 76 4C	\ 134 92 5C	l 154 108 6C	| 174 124 7C
13	1 1 0 1	CR 15 13 D	GS 35 29 1D	- 55 45 2D	= 75 61 3D	M 115 77 4D	] 135 93 5D	m 155 109 6D	} 175 125 7D
14	1 1 1 0	SO 16 14 E	RS 36 30 1E	. 56 46 2E	> 76 62 3E	N 116 78 4E	^ 136 94 5E	n 156 110 6E	~ 176 126 7E
15	1 1 1 1	SI 17 15 F	US 37 31 1F	/ 57 47 2F	? 77 63 3F	O 117 79 4F	_ 137 95 5F	o 157 111 6F	DEL 177 127 7F

C0 CODES

GL CODES (ASCII GRAPHICS)

KEY

CHARACTER — ESC — 33 OCTAL, 27 DECIMAL, 1B HEX

DEC Multinational Character Set (C1 and GR Codes)

8		9		10		11		12		13		14		15		COLUMN BITS b8 b7 b6 b5 b4 b3 b2 b1	ROW
1 0 0 0		1 0 0 1		1 0 1 0		1 0 1 1		1 1 0 0		1 1 0 1		1 1 1 0		1 1 1 1			
	200 128 80	DCS	220 144 90		240 160 A0	°	260 176 B0	À	300 192 C0		320 208 D0	à	340 224 E0		360 240 F0	0 0 0 0	0
	201 129 81	PU1	221 145 91	¡	241 161 A1	±	261 177 B1	Á	301 193 C1	Ñ	321 209 D1	á	341 225 E1	ñ	361 241 F1	0 0 0 1	1
	202 130 82	PU2	222 146 92	¢	242 162 A2	2	262 178 B2	Â	302 194 C2	Ò	322 210 D2	â	342 226 E2	ò	362 242 F2	0 0 1 0	2
	203 131 83	STS	223 147 93	£	243 163 A3	3	263 179 B3	Ã	303 195 C3	Ó	323 211 D3	ã	343 227 E3	ó	363 243 F3	0 0 1 1	3
IND	204 132 84	CCH	224 148 94		244 164 A4		264 180 B4	Ä	304 196 C4	Ô	324 212 D4	ä	344 228 E4	ô	364 244 F4	0 1 0 0	4
NEL	205 133 85	MW	225 149 95	¥	245 165 A5	µ	265 181 B5	Å	305 197 C5	Õ	325 213 D5	å	345 229 E5	õ	365 245 F5	0 1 0 1	5
SSA	206 134 86	SPA	226 150 96		246 166 A6	¶	266 182 B6	Æ	306 198 C6	Ö	326 214 D6	æ	346 230 E6	ö	366 246 F6	0 1 1 0	6
ESA	207 135 87	EPA	227 151 97	§	247 167 A7	·	267 183 B7	Ç	307 199 C7	Œ	327 215 D7	ç	347 231 E7	œ	367 247 F7	0 1 1 1	7
HTS	210 136 88		230 152 98	¤	250 168 A8		270 184 B8	È	310 200 C8	Ø	330 216 D8	è	350 232 E8	ø	370 248 F8	1 0 0 0	8
HTJ	211 137 89		231 153 99	©	251 169 A9	1	271 185 B9	É	311 201 C9	Ù	331 217 D9	é	351 233 E9	ù	371 249 F9	1 0 0 1	9
VTS	212 138 8A		232 154 9A	a	252 170 AA	o	272 186 BA	Ê	312 202 CA	Ú	332 218 DA	ê	352 234 EA	ú	372 250 FA	1 0 1 0	10
PLD	213 139 8B	CSI	233 155 9B	«	253 171 AB	»	273 187 BB	Ë	313 203 CB	Û	333 219 DB	ë	353 235 EB	û	373 251 FB	1 0 1 1	11
PLU	214 140 8C	ST	234 156 9C		254 172 AC	¼	274 188 BC	Ì	314 204 CC	Ü	334 220 DC	ì	354 236 EC	ü	374 252 FC	1 1 0 0	12
RI	215 141 8D	OSC	235 157 9D		255 173 AD	½	275 189 BD	Í	315 205 CD	Ý	335 221 DD	í	355 237 ED	ý	375 253 FD	1 1 0 1	13
SS2	216 142 8E	PM	236 158 9E		256 174 AE		276 190 BE	Î	316 206 CE		336 222 DE	î	356 238 EE		376 254 FE	1 1 1 0	14
SS3	217 143 8F	APC	237 159 9F		257 175 AF	¿	277 191 BF	Ï	317 207 CF	ß	337 223 DF	ï	357 239 EF		377 255 FF	1 1 1 1	15

C1 CODES

GR CODES
(DEC SUPPLEMENTAL GRAPHICS)

DEC Special Graphics Set Mapped into GL

ROW	COLUMN 0	COLUMN 1	COLUMN 2	COLUMN 3	COLUMN 4	COLUMN 5	COLUMN 6	COLUMN 7
	BITS B7 B6 B5 / B4 B3 B2 B1	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0
0	0 0 0 0	NUL 0 0 0	DLE 20 16 10	SP 40 32 20	0 60 48 30	@ 100 64 40	P 120 80 50	♦ 140 96 60	— 160 112 70 SCAN 3
1	0 0 0 1	SOH 1 1 1	DC1 (XON) 21 17 11	! 41 33 21	1 61 49 31	A 101 65 41	Q 121 81 51	▒ 141 97 61	— 161 113 71 SCAN 5
2	0 0 1 0	STX 2 2 2	DC2 22 18 12	" 42 34 22	2 62 50 32	B 102 66 42	R 122 82 52	H 142 98 62	— 162 114 72 SCAN 7
3	0 0 1 1	ETX 3 3 3	DC3 (XOFF) 23 19 13	# 43 35 23	3 63 51 33	C 103 67 43	S 123 83 53	F 143 99 63	¯ 163 115 73 SCAN 9
4	0 1 0 0	EOT 4 4 4	DC4 24 20 14	$ 44 36 24	4 64 52 34	D 104 68 44	T 124 84 54	G 144 100 64	├ 164 116 74
5	0 1 0 1	ENQ 5 5 5	NAK 25 21 15	% 45 37 25	5 65 53 35	E 105 69 45	U 125 85 55	┤ 145 101 65	┤ 165 117 75
6	0 1 1 0	ACK 6 6 6	SYN 26 22 16	& 46 38 26	6 66 54 36	F 106 70 46	V 126 86 56	° 146 102 66	⊥ 166 118 76
7	0 1 1 1	BEL 7 7 7	ETB 27 23 17	' 47 39 27	7 67 55 37	G 107 71 47	W 127 87 57	± 147 103 67	⊤ 167 119 77
8	1 0 0 0	BS 10 8 8	CAN 30 24 18	(50 40 28	8 70 56 38	H 110 72 48	X 130 88 58	N 150 104 68	┼ 170 120 78
9	1 0 0 1	HT 11 9 9	EM 31 25 19	) 51 41 29	9 71 57 39	I 111 73 49	Y 131 89 59	¶ 151 105 69	≤ 171 121 79
10	1 0 1 0	LF 12 10 A	SUB 32 26 1A	* 52 42 2A	: 72 58 3A	J 112 74 4A	Z 132 90 5A	J 152 106 6A	≥ 172 122 7A
11	1 0 1 1	VT 13 11 B	ESC 33 27 1B	+ 53 43 2B	; 73 59 3B	K 113 75 4B	[133 91 5B	┐ 153 107 6B	π 173 123 7B
12	1 1 0 0	FF 14 12 C	FS 34 28 1C	, 54 44 2C	< 74 60 3C	L 114 76 4C	\ 134 92 5C	┌ 154 108 6C	≠ 174 124 7C
13	1 1 0 1	CR 15 13 D	GS 35 29 1D	- 55 45 2D	= 75 61 3D	M 115 77 4D	] 135 93 5D	└ 155 109 6D	£ 175 125 7D
14	1 1 1 0	SO 16 14 E	RS 36 30 1E	. 56 46 2E	> 76 62 3E	N 116 78 4E	^ 136 94 5E	┼ 156 110 6E	· 176 126 7E
15	1 1 1 1	SI 17 15 F	US 37 31 1F	/ 57 47 2F	? 77 63 3F	O 117 79 4F	(BLANK) 137 95 5F	¯ 157 111 6F SCAN 1	DEL 177 127 7F

C0 CODES — GL CODES (DEC SPECIAL GRAPHICS)

KEY

CHARACTER — ESC 33 (OCTAL) 27 (DECIMAL) 1B (HEX)

APPENDIX C

GLOSSARY

The words in this glossary are used throughout this manual. These definitions are not absolute and might differ somewhat in other contexts. Where possible, the most common computer industry usage is the basis of the definition.

ALPHABET

An alphabet is a collection of characters. The component characters are numbered 0,1,...n-1, where n is the extent of the alphabet.

ALPHABET ATTRIBUTE

An attribute that applies to an entire alphabet. PRO/GIDIS supports storage width and height as its only alphabet attributes.

ANISOTROPIC

Not isotropic. In an anisotropic coordinate space, one unit in the X direction is not equal to one unit in the Y direction.

AREA TEXTURE

A binary pattern used to shade areas. In PRO/GIDIS this is selected from a normal character alphabet. Area texture includes size parameters (specified independently) to determine the appearance of the pattern.

ASPECT RATIO

The ratio of the width of an object to its height. Objects whose aspect ratio are important in graphics include video displays, rectangular extents (picture aspect ratio), and addressing spaces.

ATTRIBUTE

A particular property that applies to a display element (output primitive), such as character height, line texture, and so forth.

BITMAP

The rectangular array of pixels (picture elements) that is displayed on the Professional's video screen. Also known as raster or frame buffer. The Professional has a bitmap 960 pixels wide by 240 pixels high, and either one or three planes deep.

CHARACTER

A character is a two-dimensional pattern made up of two "colors" or pixel states. A character is an element in an alphabet and is specified by an identifying tuple (alphabet number, character number).

CHARACTER CELL

(See display cell or unit cell.)

CLIPPING

The drawing of only those parts of display elements that lie inside a given extent (the clipping region).

COLOR

A "real color" is a particular shade of light described in terms of its red, green, blue, and monochrome components. The color map can contain up to eight different real colors at one time.

A "logical color" is a value that represents an index into the color map. PRO/GIDIS draws images by storing either the primary or secondary logical color in pixels.

COLOR MAP

A table with entries that contain the values of the red, green and blue intensities of a particular color. This table is used to convert logical color to real color.

CURRENT POSITION

The position from which lines, arcs, and characters are to be drawn.

CURSOR

A visual representation of the current position.

DISPLAY CELL

In text processing, the display cell is that area of the screen that a character should take up. The character pattern itself resides within a unit cell; any portion of the display cell not covered by the unit cell is treated as though the pattern is OFF for that area. If the unit cell is larger than the display cell, the unit cell is clipped at the display cell borders.

DISPLAY ELEMENT

A basic graphic element that can be used to construct a display image. (also known as a graphic primitive). The display elements for PRO/GIDIS are: lines, characters, filled figures, and arcs.

FILLED FIGURE

A GIDIS display element consisting of a polygon which is filled with a two-color pattern.

GIDIS OUTPUT SPACE (GOS)

An application-specified coordinate space used by all drawing and report operations in PRO/GIDIS. A location within the PRO/GIDIS output space maps to a location within the viewport on the screen.

GLOBAL SYMMETRY

Preservation of GIDIS Output Space relationships at the expense of Hardware Address Space relationships. For example, suppose that a ten-unit distance in GOS maps to 7.5 units in HAS. With global symmetry, repeatedly moving ten GOS units results in a move of seven HAS units, then eight has units, then seven, and so forth. Local symmetry always would move seven HAS units each time. PRO/GIDIS, version 1.7, supports global symmetry for the SET_REL_POSITION, DRAW_REL_LINES, and DRAW_REL_ARC instructions.

HARDWARE ADDRESS SPACE (HAS)

A coordinate space (possibly anisotropic) used by the graphic hardware device. GIDIS hides this space from your program, and addresses the hardware through an Imposed Device Space or GIDIS Output Space.

IMPOSED DEVICE SPACE (IDS)

A coordinate space imposed on the hardware by your program. An Imposed Device Space requests PRO/GIDIS to simulate the requested device in terms of aspect ratio and addressing space. IDS is used only to set the viewport. All other coordinates and sizes are in GIDIS Output Space.

ISOTROPIC

In an isotropic coordinate space one unit in the X direction is equal to one unit in the Y direction.

LINE TEXTURE

A linear pattern used to help distinguish lines. Examples are solid, dashed, dotted, and so forth. PRO/GIDIS supports a two-color (binary) up to 16 units in length.

LOCAL SYMMETRY

Preservation of Hardware Address Space relationships at the expense of GIDIS Output Space relationships. For example, assume a ten-unit distance in GIDIS output space maps to 7.5 units in hardware coordinate space. Local symmetry always would move seven hardware units each time. With global symmetry, repeatedly moving ten GIDIS output space units results in a move of seven hardware units, then eight hardware units, then seven, and so forth. PRO/GIDIS supports local symmetry for unit cells, display cells, and cell movement (implicit and explicit).

ORIGIN

The origin of an address space is the point [0,0]. In PRO/GIDIS, the origin of IDS space is always the upper left corner of the screen. The origin of GIDIS output space is set by your program.

The origin of a character cell (either display cell or unit cell) is the point in the cell placed over the current position. This is also the point about which the cell rotates. For P/OS, V1.7, the character cell origin always is the upper left corner.

OUTPUT SPACE

(See GIDIS output space.)

PICTURE ASPECT RATIO

The ratio of the width of a picture to the height. This is normally expressed as two small numbers, such as 4:3. The picture aspect ratio on the Professional 350 monitor is 8:5. In this context, "picture" means a rectangular extent in an address space.

PIXEL (PICTURE ELEMENT)

The smallest element of a display surface that can be assigned a color or intensity.

PIXEL ASPECT RATIO

The ratio of the width of a pixel to the height. The width is the horizontal distance between adjacent pixels and the height is the vertical distance. Pixel aspect ratio is normally expressed as two small numbers, e.g. 1:2. The pixel aspect ratio on the Professional 350 monitor is 1:2.5 or 2:5.

PLANE

A plane is a portion of a bitmap that contains one bit for each pixel. The Professional 350 has either one plane (without EBO) or three planes (with EBO).

PRIMARY COLOR

The primary color is that logical color generally used to indicate the presence of an image.

SECONDARY COLOR

The secondary color is that logical color generally used to indicate the absence of an image.

STANDARD DISPLAY SIZE

The standard display size is normally equal to the standard unit size. However, for alphabet 0, rotation 0, the standard display size is slightly smaller (horizontally) than the standard unit size. This is for increased compatibility with the VT125.

STANDARD UNIT SIZE

The standard unit size depends on the alphabet width and height and the rotation angle. It is the size in GIDIS Output Space coordinates of the character displayed when one bit of the character pattern maps to exactly one pixel in the bitmap.

UNIT CELL

In text processing, the unit cell is the area in which a character pattern is drawn. If the display cell is smaller that the unit cell, the unit cell is clipped at the display cell borders. If the unit cell is larger, the remainder of the display cell is treated as though the pattern is specified OFF for that area. PRO/GIDIS automatically shrinks the unit cell to the closest possible size and, if there is no size small enough, uses the smallest possible size.

VIEWING TRANSFORMATION

The process where user-specified coordinates (in GIDIS output space) are changed into hardware coordinates. This is also known as the "graphics pipeline".

VIEWPORT

The viewport is an extent within Imposed Device Space. The viewing transformation maps the window to the viewport.

WINDOW

The window is an extent within GIDIS Output Space. The viewing transformation maps the window to the viewport.

-A-

Address space
in Viewing Transformation, 1-6
Alphabet
and REQUEST_CELL_STANDARD, 10-3
creating, 1-25
definition, C-1
description, 1-24
reset state, 3-5 to 3-6
user-defined, 1-24
Alphabet attribute
definition, C-1
Anisotropic
definition, C-1
Arc
drawing, 6-7, 6-9
Area cell size
setting, 5-12
Area operation
description, 1-29
purpose, 1-5
Area texture
affected by
SET_GIDIS_OUTPUT_SPACE, 4-5
affected by SET_OUTPUT_IDS, 4-2
definition, C-1
description, 1-20
reset state, 3-4 to 3-5
setting, 5-10
taken from line texture, 5-10
Area texture cell size
description, 1-21
Area texture size
description, 1-20
setting, 5-11
Aspect ratio
definition, C-1
description, 1-7
Attribute
definition, C-1

-B-

Backslant
see cell rendition
BEGIN_FILLED_FIGURE
aborted by initialization, 3-3
general description, 1-13
reference description, 7-1
Bitmap
and NEW_PICTURE, 3-7
definition, C-2
description, 1-14

-C-

Cartesian Coordinate System
use of, 1-6
Cell display size
affected by
SET_GIDIS_OUTPUT_SPACE, 4-6
affected by SET_OUTPUT_IDS, 4-3
description, 1-27
reset state, 3-4 to 3-5
Cell movement
affected by
SET_GIDIS_OUTPUT_SPACE, 4-5
affected by SET_OUTPUT_IDS, 4-3
description, 1-27
reset state, 3-4 to 3-5
Cell oblique
description, 1-27
reset state, 3-5
Cell rendition
description, 1-26
reset state, 3-5
Cell rotation
description, 1-26
reset state, 3-5
Cell unit size
affected by
SET_GIDIS_OUTPUT_SPACE, 4-6
affected by SET_OUTPUT_IDS, 4-3
description, 1-27
reset state, 3-4 to 3-5
CGL
relationship to PRO/GIDIS, 1-2
when to use, 1-3
Character
definition, C-2
Character cell
definition, C-2

Character rotation
and REQUEST_CELL_STANDARD, 10-3
Clipping
definition, C-2
Clipping region
affected by
SET_GIDIS_OUTPUT_SPACE, 4-5
affected by SET_OUTPUT_IDS, 4-2
and window, 1-10
description, 1-9
erasing, 9-1
in viewing transformation, 1-7
reset state, 3-4
setting, 4-6
Color
attributes, 1-14
definition, C-2
Color map
and complement mode, 1-17
definition, C-2
description, 1-15
interaction with plane mask, 5-6
reset state, 3-6
setting, 5-3
values, 5-4
Complement mode
description, 1-16
effect on filled figure, 1-21, 7-3
effect on line texture, 1-19
effect on lines, 6-4
effect on pixel size, 5-8
Complement negate mode
description, 1-17
effect on filled figure, 7-3
effect on lines, 6-4
Control instruction
purpose, 1-4
CORE Graphics Library
see CGL
CREATE_ALPHABET
in alphabet creation, 1-25
reference description, 8-2
Current pattern
description, 1-15
Current position
affected by
SET_GIDIS_OUTPUT_SPACE, 4-5
affected by SET_OUTPUT_IDS, 4-2
after DRAW_ARC, 6-8
Current position (Cont.)
definition, C-2
description, 1-12
reporting, 1-30, 10-1
reset state, 3-4
setting, 6-1 to 6-2
Cursor
affected by
SET_GIDIS_OUTPUT_SPACE, 4-5
affected by SET_OUTPUT_IDS, 4-2
definition, C-2
purpose of, 1-12
reset state, 3-4, 3-6
selecting built-in, 3-9
setting, 3-8
Curve attribute
description, 1-18

-D-

DEC Multinational Character Set
as alphabet 0, 1-24
Display cell
definition, C-3
reporting, 10-3
Display element
definition, C-3
DRAW_ARC
and REQUEST_CURRENT_POSITION, 10-2
general description, 1-13
reference description, 6-7
DRAW_CHARACTERS
and END_LIST, 3-11
and REQUEST_CURRENT_POSITION, 10-2
general description, 1-14
invalid in filled figure, 7-2
parameter block, 2-7
reference description, 8-11
DRAW_LINES
and END_LIST, 3-11
general description, 1-13
parameter block, 2-7
reference description, 6-3
DRAW_REL_ARC
general description, 1-13
reference description, 6-9
DRAW_REL_LINES
and END_LIST, 3-11
general description, 1-13

DRAW_REL_LINES (Cont.)
parameter block, 2-7
reference description, 6-5
Drawing instruction
general description, 1-12
purpose, 1-4
$DSW variable
values of, 2-4 to 2-5

-E-

EBO
description, 1-15
END_FILLED_FIGURE
general description, 1-13
reference description, 7-2
END_LIST
and DRAW_LINES, 6-3
function of, 2-7
general description, 1-6
reference description, 3-11
END_PICTURE
general description, 1-5
reference description, 3-7
use of, 3-6
Erase mode
description, 1-18
Erase negate mode
description, 1-18
ERASE_CLIPPING_REGION
general description, 1-29
reference description, 9-1
Error
in instruction stream, 2-7
Explicit movement
description, 1-29
Extended Bitmap Option
see EBO
Extent
and picture aspect ratio, 1-8
description, 1-7

-F-

Filled figure
and DRAW_LINES, 6-3
defining, 7-1
definition, C-3
effect on DRAW_ARC, 6-8
ending, 7-2
in complement mode, 1-17

Filled figure (Cont.)
instructions, 1-13
purpose, 1-5
shading to line or point, 1-21
Filled figure attribute
description, 1-19
FLUSH_BUFFERS
and END_PICTURE, 3-7
general description, 1-5
reference description, 3-8
FORTRAN-77
PRO/GIDIS instruction names in, 2-6
sample program, 2-9
symbol name restrictions, 2-3
use of with PRO/GIDIS, 2-1

-G-

GIDIS Output Space
see GOS
GIGI
and ReGIS, 1-2
Global attribute
affected by SET_GIDIS_OUTPUT_SPACE, 4-5
and SET_OUTPUT_IDS, 4-1
purpose, 1-4
Global symmetry
and SET_REL_POSITION, 6-2
definition, C-3
GOS
affected by SET_OUTPUT_IDS, 4-2
bounds of, 1-8
definition, C-3
description, 1-8
in viewing transformation, 1-6
reset state, 3-4
setting, 4-4
unit aspect ratio, 1-7 to 1-8

-H-

Hardware Address Space
see HAS
HAS
definition, C-3
general description, 1-10
in viewing transformation, 1-6
unit aspect ratio, 1-7

-I-

I/O Status Block
values of, 2-4, 2-6
IDS
definition, C-3
description, 1-9
in viewing transformation, 1-6
reset state, 3-3
setting, 4-1
unit aspect ratio, 1-7, 1-9
Implied movement
description, 1-27
Imposed Device Space
see IDS
INITIALIZE
and RIS, 2-3
effect on filled figure, 7-2
general description, 1-5
reference description, 3-1
Instruction syntax
description, 2-6, 2-8
IO.RSD function code
format, 2-5
in QIO, 2-1
use of, 1-29, 2-4, 2-6
IO.WLB function code
and VT102 Emulator, 2-1
IO.WSD function code
format, 2-4
in QIO, 2-1
use of, 2-3 to 2-4
IO.WVB function code
and VT102 Emulator, 2-1
Isotropic
definition, C-4
Isotropic mapping
description, 1-8
IDS to HAS, 1-9
window to viewport, 1-8
Italic
see cell rendition

-L-

Line
drawing, 6-3, 6-5
Line attribute
description, 1-18
Line texture
affected by
SET_GIDIS_OUTPUT_SPACE, 4-5
affected by SET_OUTPUT_IDS, 4-2
definition, C-4
description, 1-18 to 1-19
reset state, 3-4 to 3-5
setting, 5-9
LOAD_CHARACTER_CELL
and END_LIST, 3-11
in alphabet creation, 1-25
parameter block, 2-7
reference description, 8-3
Local symmetry
definition, C-4
in character cell movement, 1-29

-M-

MACRO-11
PRO/GIDIS instruction names in, 2-6
sample program, 2-8
use of with PRO/GIDIS, 2-1

-N-

NEW_PICTURE
general description, 1-5
reference description, 3-6
NOP
general description, 1-6
reference description, 3-10

-O-

Opcode
function of, 2-6
Opcode word
format, 2-6
Origin
definition, C-4
Output space
definition, C-4
Overflow
avoiding, 1-8
Overlay mode
description, 1-17
Overlay negate mode
description, 1-17

-P-

Parameter block
fixed length, 2-7
format, 2-7
variable length, 2-7
PASCAL
PRO/GIDIS instruction names in, 2-6
Picture aspect ratio
definition, C-4
description, 1-8
Pixel
addressing individual, 1-12
definition, C-4
Pixel aspect ratio
definition, C-5
Pixel size
description, 1-19
setting, 5-7
Plane
definition, C-5
description, 1-14
Plane mask
reset state, 3-5
setting, 5-4
Primary color
definition, C-5
description, 1-15
reset state, 3-5
setting, 5-1
PRINT_SCREEN
general description, 1-29
reference description, 9-2

-Q-

QIO
access to PRO/GIDIS, 2-1, 2-9
expansion forms, 2-3
FORTRAN-77 routine, 2-1
QIOW
see QIO
Queue I/O Request
see QIO

-R-

Read Special Data
see IO.RSD
ReGIS
relationship to PRO/GIDIS, 1-2
when to use, 1-4
Remote Graphics Instruction Set
see ReGIS
Replace mode
description, 1-17
effect on line texture, 1-19
effect on pixel size, 5-8
Replace negate mode
description, 1-17
Report
format, 1-30
Report handling
description, 1-29
Report instruction
purpose, 1-5
REQUEST_CELL_STANDARD
and IO.RSD, 2-4
general description, 1-30
reference description, 10-3
REQUEST_CURRENT_POSITION
and IO.RSD, 2-4
general description, 1-30
reference description, 10-1
REQUEST_STATUS
and IO.RSD, 2-4
general description, 1-30
reference description, 10-2
Reset to Initial State (RIS)
use of, 2-3
RIS (Reset) escape sequence
use of, 2-3

-S-

Screen
printing, 9-2
Scrolling
by VT102 Emulator, 2-3
SD.GDS parameter
use of, 2-4 to 2-5
Secondary color
definition, C-5
description, 1-15
reset state, 3-5
setting, 5-2
Secondary color and NEW_PICTURE, 3-6
SET_ALPHABET
general description, 1-24

SET_ALPHABET (Cont.)
in alphabet creation, 1-25
reference description, 8-2
SET_AREA_CELL_SIZE
general description, 1-21
reference description, 5-12
SET_AREA_TEXTURE
effect on area cell size, 5-12
general description, 1-20
reference description, 5-10
SET_AREA_TEXTURE_SIZE
general description, 1-20
reference description, 5-11
SET_CELL_DISPLAY_SIZE
general description, 1-27
reference description, 8-8
SET_CELL_EXPLICIT_MOVEMENT
general description, 1-27
reference description, 8-10
SET_CELL_MOVEMENT_MODE
general description, 1-27
reference description, 8-9
SET_CELL_OBLIQUE
general description, 1-27
reference description, 8-6
SET_CELL_RENDITION
general description, 1-26
reference description, 8-4
SET_CELL_ROTATION
general description, 1-26
reference description, 8-6
SET_CELL_UNIT_SIZE
general description, 1-27
reference description, 8-7
SET_COLOR_MAP_ENTRY
general description, 1-15
reference description, 5-3
SET_GIDIS_OUTPUT_SPACE
effect on clipping region, 1-9
invalid in filled figure, 7-2
purpose, 1-8
reference description, 4-4
SET_LINE_TEXTURE
general description, 1-18
reference description, 5-9
SET_OUTPUT_CLIPPING_REGION
general description, 1-9
reference description, 4-6
SET_OUTPUT_CURSOR
general description, 1-5
reference description, 3-8
SET_OUTPUT_IDS
general description, 1-9
invalid in filled figure, 7-2
reference description, 4-1
SET_OUTPUT_VIEWPORT
general description, 1-10
invalid in filled figure, 7-2
reference description, 4-3
SET_PIXEL_SIZE
effect on line texture, 1-19
general description, 1-19
reference description, 5-7
SET_PLANE_MASK
and VT102 Emulator, 2-3
general description, 1-15
reference description, 5-4
SET_POSITION
general description, 1-13
invalid in filled figure, 7-2
reference description, 6-1
SET_PRIMARY_COLOR
general description, 1-15
reference description, 5-1
SET_REL_POSITION
general description, 1-13
invalid in filled figure, 7-2
reference description, 6-2
SET_SECONDARY_COLOR
general description, 1-15
reference description, 5-2
SET_WRITING_MODE
reference description, 5-7
Standard display size
definition, C-5
reporting, 1-30
Standard unit size
definition, C-5
reporting, 1-30
Status
in error condition, 2-7
reporting, 1-30, 10-2
SYSLIB
as source of QIO routine, 2-1
module QIOSYM, 2-3

-T-

Terminal emulator
see VT102 Emulator
see VT125 Emulator

Text
  instruction, 1-14
Text attribute
  description, 1-24
Text instruction
  purpose, 1-5
Transparent mode
  description, 1-16
Transparent negate mode
  description, 1-16

-U-

Unit aspect ratio
  description, 1-7
  of GOS, 1-7 to 1-8
  of HAS, 1-7
  of IDS, 1-7
Unit cell
  definition, C-5
  reporting, 10-3

-V-

Video monitor
  color, 1-11
  monochrome, 1-10
Viewing transformation
  definition, C-6
  description, 1-6, 1-12
  process, 1-11
  purpose, 1-4
Viewport
  affected by SET_OUTPUT_IDS, 4-2
  definition, C-6
  description, 1-10
  in viewing transformation, 1-7
  reset state, 3-4
  setting, 4-3
VT102 Emulator
  interaction with PRO/GIDIS, 2-2 to 2-3
  use of planes, 5-5
  use of with PRO/GIDIS, 2-1
VT125 Emulator
  use of, 1-4
VT125 terminal
  and ReGIS, 1-2

-W-

Window
  and clipping region, 1-10
  definition, C-6
  description, 1-8
  in Viewing Transformation, 1-7
Write Special Data
  see IO.WSD
Writing mode
  see also individual modes
  description, 1-15, 1-18
  interaction with plane mask, 5-5
  reset state, 3-5
  setting, 5-7

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized?
Please make suggestions for improvement.

Please cut along this line

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____
or
Country

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

Professional 300 Series Publications
DIGITAL EQUIPMENT CORPORATION
146 MAIN STREET
MAYNARD, MASSACHUSETTS 01754

Do Not Tear - Fold Here

Cut Along Dotted Line