PRO/GIDIS Manual

Order No. AA-Y660A-TK
AD-Y660A-T1

April 1984

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 V2.0 PRO/Tool Kit V2.0
SOFTWARE VERSION:	PRO/GIDIS V2.0

DIGITAL EQUIPMENT CORPORATION
Maynard, Massachusetts 01754

First Printing, December 1983
Updated, April 1984

4/84-15

CONTENTS

CHAPTER 7 FILLED FIGURE INSTRUCTIONS

CHAPTER 8 TEXT INSTRUCTIONS

CHAPTER 9 AREA OPERATION INSTRUCTIONS

CHAPTER 10 REPORT HANDLING

APPENDIX A PRO/GIDIS INSTRUCTION SUMMARIES

APPENDIX B DEC MULTINATIONAL CHARACTER SET

EXAMPLES

FIGURES

TABLES

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 -- Alphabet Data Structure

Appendix D -- Glossary

RELATED DOCUMENTATION

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

- **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).

- **SET_OUTPUT_CURSOR_RENDITION**

  The SET_OUTPUT_CURSOR_RENDITION instruction specifies whether the cursor blinks or is continuous.

- **SET_OUTPUT_RUBBER_BAND**

  The SET_OUTPUT_RUBBER_BAND instruction specifies if a rubber band is to be generated along with the cursor, and where the base of the rubber band is.

- **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.

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_ARCS**

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

- **DRAW_REL_ARCS**

  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.

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 visible writing modes.

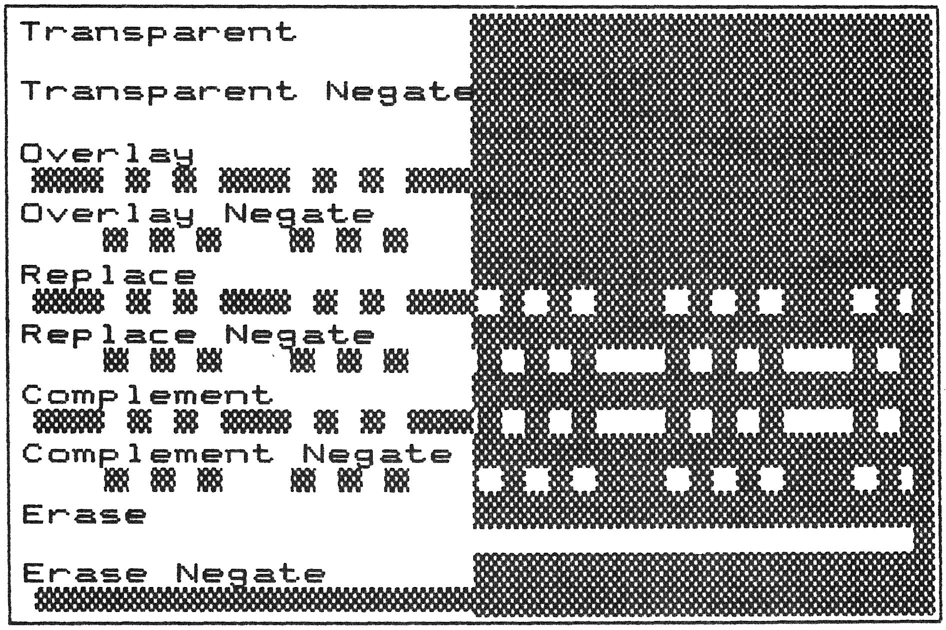

Figure 1-8: The Writing Modes Shown with Line Texture

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.

```
.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_ARCS
.WORD   6000.,3800.,180.
.BYTE   3.,23.            ; length = 3, opcode = DRAW_REL_ARCS
.WORD   6000.,5532.,-60.
.BYTE   0.,32.            ; length = 0, opcode = END_FILLED_FIGURE
```

Example 1-6: A Filled Figure Using DRAW_REL_ARCS

```
.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).

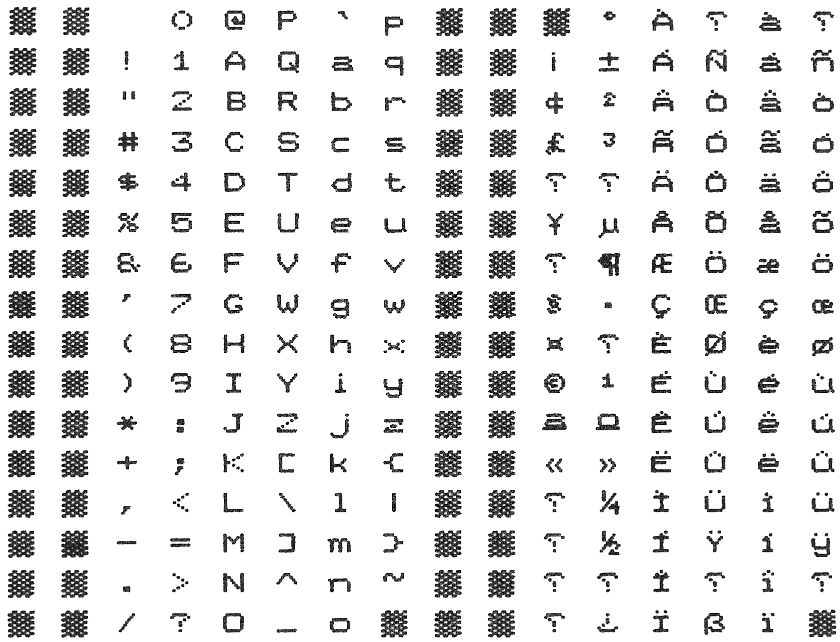

Figure 1-11: Alphabet 0

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.

You can create a new alphabet of user-defined characters in two ways:

1. Created Alphabets

   - Use the SET_ALPHABET instruction to specify the current alphabet.

   - 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.

- Use the LOAD_CHARACTER_CELL instruction to add characters to the alphabet. Or use BEGIN_DEFINE_CHARACTER and END_DEFINE_CHARACTER instructions.

2. Loaded Alphabets

- Create a named region using P/OS facilities (see Tool Kit Manual, P/OS Directives).
- Use SET_ALPHABET
- Use LOAD_BY_NAME

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 width-type parameter must be 0 for PRO/GIDIS on P/OS 2.0.

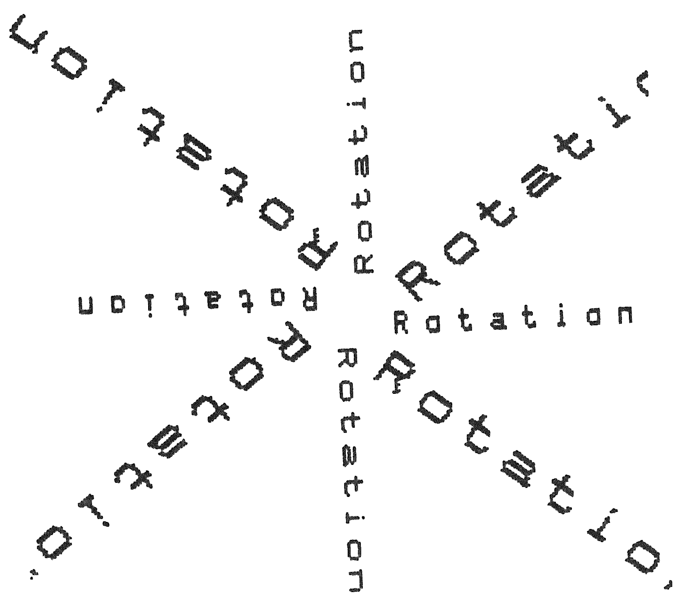

Figure 1-12: Character Cell Rotation

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.

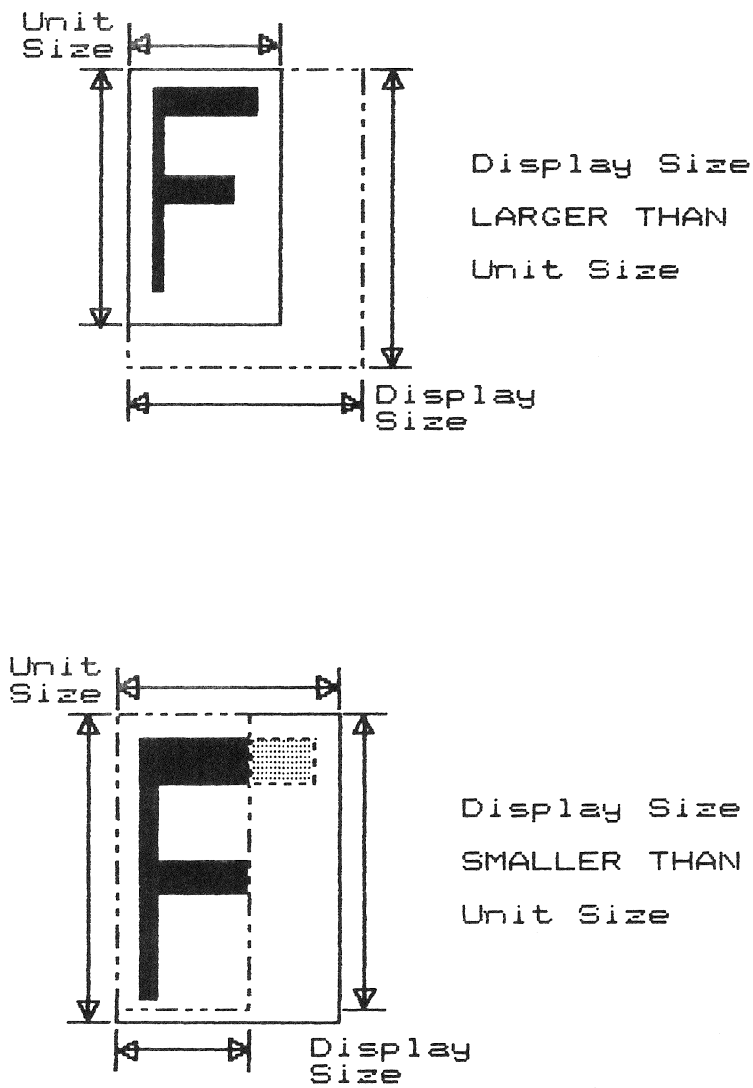

Figure 1-13: Character Unit Cell and Display Cell

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.

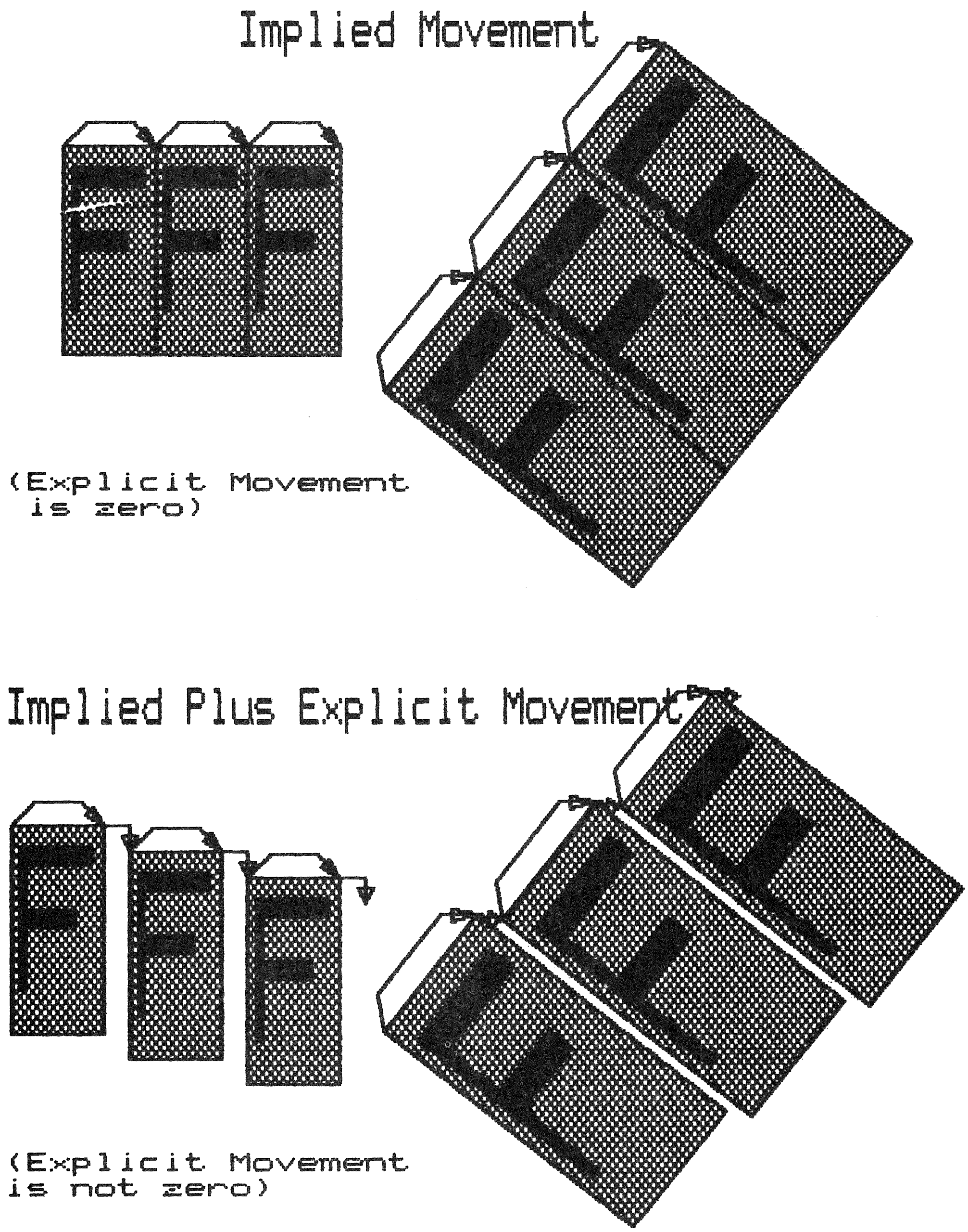

Figure 1-14: Character Cell Movement

- Explicit movement means that the current position moves 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 always draws characters in local symmetry. This means that characters with the same unit and display sizes always cover the same number of pixels.

Cell movement can exhibit either local or global symmetry. When local symmetry is enabled (the default), all cell movements will be the same number of physical pixels. This means that all spaces will look the same on the screen but the "hard" position and the "soft" position can drift apart. Global symmetry keeps the "soft" position and the "hard" position together but allows 1 pixel wide gaps between every few characters. (The "soft" position is calculated in GOS coordinates and the "hard" position in physical pixels.)

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.

- **SCROLL_CLIPPING_REGION**

  Moves the data within the output clipping region.

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 2.0 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.

- **REQUEST_OUTPUT_SIZE**

  Reports the parameters of the size, shape and resolution of the video hardware used.

- **REQUEST_VERSION_NUMBER**

  Reports the version number of GIDIS.

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.

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

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.

```
.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

Either type of parameter block can be used with any instruction. 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. Version 2.0 and future versions set the status flag to FAILURE in this case and when not enough parameters are given.

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
```

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
72/1	SET_OUTPUT_CURSOR_RENDITION	mask
53/3	SET_OUTPUT_RUBBER_BAND	type, base_x, base_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.

```
 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
                      |  |  |  |  |  |  |  |  |--- Addressing
                      |  |  |  |  |  |  |  |--- Global Attributes
                      |  |  |  |  |  |  |----- Text
                      |  |  |  |  |  |----- (reserved)
                      |  |  |  |  |------Color Map
                      |  |  |  |--------Alphabet Storage
                      |  |  |-------(reserved)
                      |  |-------(reserved)
                      |-------Cursor

            (Bits 8-15 are reserved)
```

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 and output rubber band	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
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

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 2.0 these cursors are:

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

User-specified width and height are ignored when the tracking cross or crosshairs are 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.

- SET_OUTPUT_CURSOR changes only the graphics mode cursor. However, turning on or off the VT102 emulator cursor has the side effect of turning on or off the graphics cursor.

- 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 SET_OUTPUT_CURSOR_RENDITION

The SET_OUTPUT_CURSOR_RENDITION instruction determines if the cursor blinks or is continuous.

Opcode: 72 **Length:** 1

Format: SET_OUTPUT_CURSOR_RENDITION mask

mask is a word that specifies whether the cursor should blink or be continuous as shown in Figure 3-2.

```
 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+   1 blinking
                                                | -- 0 continuous

   |  |  |  |  |  |  |  |  |  |  |  |  |  |  | -- (Reserved)
```

Figure 3-2: SET_OUTPUT_CURSOR_RENDITION mask

Status: SUCCESS

Example:

```
.BYTE   1.,72.  ;length=1, opcode for
                ;SET_OUTPUT_CURSOR_RENDITION
.WORD   0.      ;set to continuous mode
```

3.7 SET_OUTPUT_RUBBER_BAND

The SET_OUTPUT_RUBBER_BAND instruction specifies if a rubber band is to be generated along with the output cursor. It also gives the base of the rubber band.

Opcode: 53 **Length:** 3

Format: SET_OUTPUT_RUBBER_BAND type, base_x, base_y

type the type of rubber band to use. (see table 3-4)

base_x the x coordinate (in GOS) of the desired rubber band base

base_y the y coordinate (in GOS) of the desired rubber band base

Status: SUCCESS if the type is legal; otherwise, FAILURE.

Type Code	Rubber Band

-1	no rubber band
0	default (same as -1)
1	rubber band line
2	rubber band rectangle

Table 3-4: Types of Rubber Bands

Notes:

- The rubber band (if selected) will be drawn each time the cursor is drawn.

  If the cursor blinks, then the rubber band also blinks.

- The rubber band line is drawn from the base position to the current position.

- The rubber band rectangle is the rectangle with one corner at the base position and the opposite corner at the current position. The rectangle will degenerate to a line or point if the current position and base position are the same in one or both coordinates.

- Since both the cursor and the rubber band are drawn in complement mode, if they cover the same points, they will both disappear. It may be preferable to turn the cursor off when a rubber band is on.

Example:

```
.BYTE   3.,53.  ;length=3., opcode for
                ;SET_OUTPUT_RUBBER_BAND
.WORD   1.      ;rubber band line
.WORD   50.     ;the base is [50,60]
.WORD   60.

.BYTE   2.,29.  ;length=1., opcode for set_position
.WORD   100.    ;new current position
.WORD   300.    ;is [100,300]

                ;when the cursor appears there
                ;will also be a rubber band line from
                ;[50,60] to [100,300].
```

3.8 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)
```

3.9 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.

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. This means that physical sizes will change when you change the viewport.

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.

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/N	DRAW_ARCS	x, y, angle, ...
27/N	DRAW_REL_ARCS	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.

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

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:

```
                        ;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_ARCS

The DRAW_ARCS 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_ARCS 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:** N

Format: DRAW_ARCS x1, y1, angle1, x2, y2, angle2 ...

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

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

angle1 The angle for the first arc is given in degrees, with a positive value meaning counter-clockwise with respect to the physical screen.

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

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

angle2 The angle for the second arc is given in degrees, with a positive value meaning counter-clockwise with respect to the physical screen.

... Additional coordinate and angle triplets specify center points and angles for additional arcs.

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.
- Each arc starts at the end point of the previous arc.
- The current position is left at the end of the last arc.
- DRAW_ARCS 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 2.0, 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_ARCS 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_ARCS
        .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_ARCS
        .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_ARCS

The DRAW_REL_ARCS 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:** N

Format: DRAW_REL_ARCS dx, dy, angle, 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_ARCS.

Example:

```
                        ;Current position is [400,300]
                        ;Filled figure flag is FALSE
.BYTE   3.,27.          ;Length=3,opcode is DRAW_REL_ARCS
.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]
```

Example:

```
                        ;Current position is [330,430]
                        ;Filled figure flag is FALSE
.BYTE   6.,27.          ;Length=3, opcode is DRAW_REL_ARCS
.WORD     +35.          ;
.WORD     -50.          ;Center is [+35,-50]
.WORD      90.          ;[365,380], 90 degree arc
.WORD     -35.          ;Current position is now [415,415]
.WORD     +50.          ;Center is 380,465]
.WORD      90.          ;90 degrees
                        ;draws a lens shaped object with two
                        ;circular arcs.
```

- 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 2.0.)

- 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   0.,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, d1,...,dn
33/4	BEGIN_DEFINE_CHARACTER	char-index, width, nominal-width, nominal-height
36/0	END_DEFINE_CHARACTER	
37/2	LOAD_BY_NAME	name_0, name_1
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-index1, char-index2, ...
74/N	DRAW_PACKED_CHARACTERS	2charindex, ...

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 2.0) 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 127 through 160 decimal) are not included, and, if requested with DRAW_CHARACTERS, appear as an error ("blob" or "checkerboard") character. Some indices are reserved 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 or 5

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

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.

initialize
: initializes all characters in the newly created alphabet. If 0, initialize to blank. If it is not 0, initialize to solid. If not present, then initialize to solid.

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.

- Since alphabet storage uses dynamic memory, use the REQUEST_STATUS instruction to obtain the status of any CREATE_ALPHABET instruction.

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, d1,...,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.

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 2.0 does not use this value. However, it still must be in the appropriate range.)

d0, d1,... 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 (d1), 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.
- This command should not be used within a character definition block. (See the BEGIN_DEFINE_CHARACTER instruction.)

Example:

```
                        ;Alphabet 2 has width of 4, height of 5,
                        ;  and extent of 10
        .BYTE   6.,34.  ;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 BEGIN_DEFINE_CHARACTER

The BEGIN_DEFINE_CHARACTER instruction starts a character definition block in which all commands draw into the binary storage raster associated with the given character instead of drawing into the entire screen. This command is paired with the END_DEFINE_CHARACTER command.

Opcode: 33 **Length:** 4

Format: BEGIN_DEFINE_CHARACTER char-index, width, nom_width, nom_height

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.

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 2.0 does not use this value. However, it still must be in the appropriate range.)

nomwidth The nominal width specifies the desired width of the character.

nomheight The nominal height specifies the desired height of the character.

Status: SUCCESS if current alphabet is not equal to 0 and Char_index is less than or equal to alphabet extend (current alphabet) and there are sufficient resources to define this character; otherwise FAILURE.

Notes:

- If the character definition contains a circle, then that character will contain a circle when drawn with unit cell width and height proportional to the nominal width and height. In other words, characters can be drawn with arbitrary scaling in X and Y.

- This command implicitly saves the entire PRO/GIDIS state. That state will be restored by the END_DEFINE_CHARACTER command. The following state will be imposed in place of the active state.

output ids width	Nominal Width
output ids height	Nominal Height
output viewport x origin	0
output viewport y origin	0
output viewport width	Nominal Width
output viewport height	Nominal Height
GIDIS output space x origin	0
GIDIS output space y origin	0
GIDIS output space width	Nominal Width
GIDIS output space height	Nominal Height
output clipping x origin	0
output clipping y origin	0
output clipping width	Nominal Width
output clipping height	Nominal Height
current position x	0
current position y	0
area texture width	Nominal Width
area texture height	Nominal Height
line texture size	GOS Equiv. of 'line texture length'
logical pixel x offset	0
logical pixel y offset	0
logical pixel width	1 hardware pixel
logical pixel height	1 hardware pixel
cell unit size width	Nominal Width
cell unit size height	Nominal Height
cell display size width	Nominal Width
cell display size height	Nominal Height

cell movement mode flag	implicit, local
cell explicit movement dx	0
cell explicit movement dy	0
primary color	1
secondary color	0
character cell	all '0's
plane mask	1
writing mode	overlay

- BEGIN_DEFINE_CHARACTER commands can not be nested.
- Any INITIALIZE command will abort the character definition mode.
- The following commands are illegal inside a character definition block:

  BEGIN_DEFINE_CHARACTER
  LOAD_CHARACTER_CELL
  CREATE_ALPHABET
  LOAD_BY_NAME

- If BEGIN_DEFINE_CHARACTER fails, GIDIS may skip all further commands until an END_DEFINE_CHARACTER. In particular, REQUEST_STATUS may not succeed, and your program may hang. So insure that all BEGIN_DEFINE_CHARACTER commands be followed by an END_DEFINE_CHARACTER.
- Do not allow the VT102 simulator to do a full screen scroll while defining a character. See the END_DEFINE_CHARACTER example.

Example:

See END_DEFINE_CHARACTER example.

8.5 END_DEFINE_CHARACTER

The END_DEFINE_CHARACTER instruction terminates the character definition block and makes that character available and restores previous conditions.

Opcode: 36 **Length:** 0

Format: END_DEFINE_CHARACTER

Status: SUCCESS if character definition flag is TRUE; otherwise, FAILURE.

Example:

```
                        ;assume current alphabet is 1, storage
                        ;size of alphabet 1 is 9 by 9.
        .BYTE   4.,33.  ;length = 4,
                        ; opcode=BEGIN_DEFINE_CHARACTER
        .BYTE   3.      ;defining character 3
        .WORD   9.      ;width
        .WORD   90.     ;nominal width
        .WORD   225.    ;nominal height
                        ;now ready to draw into the characters
                        ;storage area
                        ;IDS is 90 X 235.
        .BYTE   .2,29.  ;length = 2, opcode=SET POSITION
        .WORD   0.      ;[0,100] is moddle of left hand side.

        .BYTE .255.,25. ;end-list terminated, opcode=DRAW_LINES
        .WORD   40.     ;
        .WORD   200.    ;[40,200]
        .WORD   80.     ;
        .WORD   100.    ;[80,100]
        .WORD   40.     ;
        .WORD   0.      ;[40,0]
        .WORD   0.
        .WORD   100.    ;[0,100]
        .WORD   -32768. ;end list

        .BYTE   0.,36.  ;END_DEFINE_CHARACTER
```

Figure 8-1 illustrates some examples of the character defined in the example above.

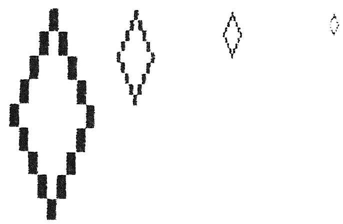

Figure 8-1: End_Define_Character

8.6 LOAD_BY_NAME

The LOAD_BY_NAME instruction makes an alphabet that is defined in a named region available to users. The argument is a pair of 16 bit words which contain a region name in RAD50. Other software must explicitly create the region, either by the CRRG$ directive or by defining the alphabet in a MACRO program and using the INSTALL command on the native tool kit. In either case the region must have the format shown in Appendix C. If no such alphabet (region) can be found then an alphabet containing no characters will be loaded.

Opcode: 37 **Length:** 2

Format: LOAD_BY_NAME name_0, name_1

name_0 , name_1 are arguments that contain a region name in RAD 50.

Status: SUCCESS if the two word argument specifies a valid region, the region has the proper format, the current alphabet number is valid for a user definable alphabet (not 0) AND there are sufficient resources to load the alphabet, FAILURE otherwise.

Notes:

- Gidis does not create a region. The user should create a region and make sure it has the proper format. Use the RSX directive CRRG$ and load it algorithmically or build a common library and install it using the INSTALL command on the native tool kit. In either case the region must have the format shown in Appendix C.

Example:

```
.BYTE   2.,37.  ;length=2, opcode=LOAD__BY__NAME
.RAD50 "BOLD  " ;let MACRO-11 compute the RAD50 for BOLD
```

Example:

```
.BYTE   2.,37.                  ;RAD50 for MYALPH
.WORD   050500+001750+000001    ;MYA
.WORD   045400+001200+000010    ;LPH
```

8.7 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 8-2).

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 8-2 illustrates the defined rendition bits that can be selected.

```
 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
                                             | -- Back Slant
                                          | -- Italics
 |  |  |  |  |  |  |  |  |  |  |  |  |  | -- (Reserved)
```

Figure 8-2: 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
```

8.8 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

8.9 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 2.0) 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)
```

8.10 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
```

8.11 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
```

8.12 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
1	Explicit cell movement, global symmetry
2	Explicit and implied movement, local symmetry
3	Explicit and implied movement, global symmetry
	(All other values are reserved.)

Status: SUCCESS if Flag is 0 to 3, 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 such 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.

- When using global symmetry, the current position is exactly that value that would be calculated by your program. However, character spacing may not always be even due to round-off errors.

- No drawing occurs when the SET_CELL_MOVEMENT_MODE instruction executes.

Example:

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

8.13 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 or 1. The value represents the inter-character spacing when the mode is 2 or 3. (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
```

8.14 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
```

8.15 DRAW_PACKED_CHARACTERS

Opcode: 74 **Length:** N

Format: DRAW_PACKED_CHARACTERS 2charindex, ...

2charindex is two 8 bit quantities which are used as character indexes to write.

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

Notes:

- The arguments are taken as pairs of characters. Each word has 2 byte characters to draw. This command can be used for characters whose indexes are in the range 0-254. The value 255 explicitly performs no operation. If you want to draw an odd number of characters, use 255 as the filler. See the DRAW_CHARACTER command for a description of how characters are actually drawn.

- The low order byte is used first and then the high order byte.

Example:

```
                          ;assume current alphabet is 0
        .BYTE   3.,74.    ;length=3 words,opcode for
                          ;DRAW_PACKED_CHARACTERS
        .BYTE   116.,101. ;'t','e'
```

```
        .BYTE   115.,116. ;'s', 't'
        .BYTE   49.,255.  ; '1', no character
```

Example:

```
                          ;current alphabet 0
        .BYTE   255.,74.  ;end-list-term, opcode=1
                          ;DRAW_PACKED_CHARACTERS
        .ASCII "TEST"     ;test
        .BYTE  '2,255.    ;'2', no character
        .WORD  -32768.
```

Example:

```
        .BYTE  1.,38.     ;length=1, opcode=set-alphabet
        .WORD  1.         ;alphabet 1

        .BYTE  2.,74.     ;length=2,
                          ; opcode=DRAW_PACKED_CHARACTERS
        .BYTE  0.,1.      ;draw characters 0,1
        .BYTE  255., 254. ; and 254 from alphabet 1.
```

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
```

9.3 SCROLL_CLIPPING_REGION

The SCROLL_CLIPPING_REGION instruction moves data within the output clipping region, according to the direction and distance specified by the instruction's accompanying parameters. The vacated display area is reset to the current secondary color, ignoring area texture and writing mode.

Opcode: 52 **Length:** 2

Format: SCROLL_CLIPPING_REGION dx, dy

dx The GIDIS output space distance to move the data horizontally. If dx is positive, the data is shifted right to left; if negative, the shift is to the right.

dy The GIDIS output space distance to move the data vertically. If dy is positive, the data is shifted toward the top of the screen; if negative, the shift is toward the bottom of the screen.

Notes:

- PRO GIDIS will copy the data in the currently selected planes directly to the desired position. Planes not selected are not scrolled or otherwise changed. Hardware assist is used when possible so the appearance of the screen may be different during the scroll operations. Scrolls that include everything on the screen, except the 32 pixel wide bands on either side, might scroll those bands in addition to the clipping region. (See Figure 1-5 for a picture of the screen showing the 32 pixel wide bands).

- The data scrolled out is not saved -- you cannot scroll out a portion of an image and then scroll it back in. Solid secondary color always scrolls in.

- Shaded areas within the clipping region will not necessarily be aligned with shaded areas outside the clipping region after this command.

- Scroll Clipping region does not work in P/OS 1.7 except when the clipping region is set to the entire screen, and the plane mask includes all planes present. It works correctly in P/OS 2.0 for all combinations of plane mask and clipping region.

Example:

```
        .BYTE   2.,52.  ;Length=0, opcode=SCROLL_CLIPPING_REGION
        .WORD   -100.   ;dx
        .WORD   0.      ;dy
                        ;Slides data to the right 100 units
```

Example:

```
        .BYTE   2.,52.
        .WORD   0.      ;Scroll data down
        .WORD   -15.    ;15 units
```

Example:

```
        .BYTE   2.,52.
        .WORD   -30     ;Move data in the clipping region
        .WORD   +30     ;30 units left and 20 units up.
```

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
71/0	REQUEST_VERSION_NUMBER	VERSION_NUMBER_REPORT	dev_code, version number
57/0	REQUEST_OUTPUT_SIZE	OUTPUT_SIZE_REPORT	ulx, uly, screen_width, screen_height, total_width, total_height, resolution_x, resolution_y, total_plane_mask

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_ARCS instructions also move the current position.
- The REQUEST_CURRENT_POSITION instruction is most useful following a DRAW_ARCS 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.
```

10.4 REQUEST_OUTPUT_SIZE

The REQUEST_OUTPUT_SIZE command reports the parameter of the device currently used.

Opcode: 57 **Length:** 0

Format: REQUEST_OUTPUT_SIZE

Status: SUCCESS

The report takes the following form:

OUTPUT_SIZE_REPORT, ulx, uly, screen_width, screen_height, total_width, total_height, resolution_x, resolution_y, Total_plane_mask

where

- [ulx, uly] is the upper left corner (in Output IDS) of the total device executable space;
- Screen_width and Screen_height are the width and height (in Output IDS) of the visible area of the output device;
- Total_width and Total_height are the width and height (in Output IDS) of the total device executable space;
- Resolution_x and Resolution_y are the number of addressable units (pixels) in the total device executable space;
- Total_plane_mask is the plane mask that contains a 1 for every plane accessible.

Example:

```
                        ;Assume IDS is 960 by 600
.BYTE    0.,57.         ;length=0,opcode for REQUEST_OUTPUT_SIZE
.WORD    0.             ;total upper left is [-32,0]
.WORD    1024           ;IDS width and height of visible area
.WORD    600            ;
.WORD    1024           ;IDS width and height of total area
.WORD    600            ;
.WORD    1024           ;number of pixels in total device width
.WORD    240            ;number of pixels in total device height
.WORD    7.             ;total plane mask
.BYTE    9.             ;9 words following output size report tag
.BYTE    2.             ;OUTPUT_SIZE_REPORT_TAG
.WORD    -32.           ;IDS coordinate of
.BYTE    9.             ;data words following
```

10.5 REQUEST_VERSION_NUMBER

The REQUEST_VERSION_NUMBER instruction reports the version number of PRO/GIDIS.

Opcode: 71 **Length:** 0

Format: REQUEST_VERSION_NUMBER

Status: SUCCESS

The reported information takes the following form:
VERSION_NUMBER_REPORT, device_code, version

where

- device_code is 21 for the PRO/VIDEO GIDIS.
- version is the version number.

Notes:

- These numbers should be used with all error reports.

Example:

```
BYTE    0.,71.  ;Lgth=0, opcode=VERSION_NUMBER_REPORT
                ;byte 2. data words following
                ;VERSION_NUMBER_REPORT
                ;byte 7. tag
                ;word 21. device code
                ;word 25. version number
```

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_BUFFERS -	control
53	3	13571	SET_OUTPUT_RUBBER_BAND type, x, y	control
72	1	18433	SET_OUTPUT_CURSOR_RENDITION mask	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_ARCs 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_ARCS 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, nw, nh	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
37	2	9474	LOAD_BY_NAME name_0, name_1	text (RAD50)
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
74	N	18944+N	DRAW_PACKED_CHARACTERS 2charindex, ...	text
48	0	12288	ERASE_CLIPPING_REGION -	area
52	2	13314	SCROLL_CLIPPING_REGION dx, dy	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
57	0	14592	REQUEST_OUTPUT_SIZE	reports
58	0	14848	REQUEST_STATUS -	reports
71	0	18176	REQUEST_VERSION_NUMBER	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_ARCS 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_ARCS 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	4	8452	BEGIN_DEFINE_CHARACTER c, w, nw, nh	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
37	2	9474	LOAD_BY_NAME name_0, name_1	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
52	2	13314	SCROLL_CLIPPING REGION dx, dy	area
53	3	13571	SET_OUTPUT_RUBBER_BAND type, x, y	control
54	0	13824	REQUEST_CELL_STANDARD -	reports
55	0	14080	REQUEST_CURRENT_POSITION -	reports
57	0	14592	REQUEST_OUTPUT_SIZE	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
71	0	18176	REQUEST_VERSION_NUMBER -	report
72	1	18433	SET_OUTPUT_CURSOR_RENDITION mask	control
74	N	18944+N	DRAW_PACKED_CHARACTERS 2charindex, ...	text
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
33	4	8452	BEGIN_DEFINE_CHARACTER c, w, nh, nw	text
31	0	7936	BEGIN_FILLED_FIGURE -	filled figures
46	4	11780	CREATE_ALPHABET w, h, extent, type	text
23	3	5891	DRAW_ARCS x, y, angle	drawing
35	N	8960+N	DRAW_CHARACTERS c, ...	text
25	N	6400+N	DRAW_LINES x, y, ...	drawing
74	N	18944+N	DRAW_PACKED_CHARACTERS 2charindex, ...	text
27	3	6915	DRAW_REL_ARCS 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
37	2	9474	LOAD_BY_NAME name_0, name_1	text
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
57	0	14592	REQUEST_OUTPUT_SIZE	reports
58	0	14848	REQUEST_STATUS -	reports
71	0	18176	REQUEST_VERSION_NUMBER -	report
52	2	13314	SCROLL_CLIPPING_REGION dx, dy	area
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

72	1	18433	SET_OUTPUT_CURSOR_RENDITION mask	control
12	2	3074	SET_OUTPUT_IDS w, h	transform
53	3	13571	SET_OUTPUT_RUBBER_BAND type, x, y	control
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
2	9	521	OUTPUT_SIZE_REPORT	ulx, uly, screen_width, Screen_height, total_width, total_height, resolution_x, resolution_y, total_plane_mask	report tags
4	1	1025	STATUS_REPORT	code	report tags
5	4	1284	CELL_STANDARD_REPORT	uw, uh, dw, dh	report tags
7	2	1794	VERSION_NUMBER_REPORT	code, version	report tags

DEC MULTINATIONAL CHARACTER SET

DEC Multinational Character Set (C1 and GR Codes)

8		9		10		11		12		13		14		15		COLUMN	
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		b8 b7 b6 b5 BITS b4 b3 b2 b1	ROW
	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	Ÿ	335 221 DD	í	355 237 ED	ÿ	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)

APPENDIX C

ALPHABET DATA STRUCTURE

This Appendix describes the Alphabet Data Structure required by the LOAD_BY_NAME instruction.

Key:

name address 'name'

<name> contents of word at address 'name'

1. Header information (word wide)

```
    Header information must start at the beginning of the region.
    Word 0 in the region is al$mag.
    name    description
    ------  ------------

    al_$mag  magic number -- must be 16473 (decimal)
    al_$str  structure version number -- 102 (decimal)
    al_$siz  size of header (bytes) --
    al_$tot  total size of this entire data structure (bytes)
                     (PRO/GIDIS 2.0 -- This must be <= 8KB)
    al_$flg  flags -- reserved

    al_$wid  maximum width (bits)
                     ( <al_$wid> + 7 ) / 8 bytes)
                     (Must be 0 to 16, inclusive.  If this width is 0 to
                 8, then the font information is stored in byte wide
                 cells.   If 9 to 16, the font information
                 is stored in word-wide cells.

    al_$nom  nominal width (bits). Used as the 'alphabet width' for
                     scaling. (Must be equal to <al_$wid>.
    al_$hgt  height (bits)
                 (0 to 16, inclusive)
    al_$fst  index of first character represented in this alphabet
    al_$ext  extent of alphabet -- the number of characters represented i
                     this alphabet.  There is no specific limit,
```

```
                but the entire structure must fit in 1 APR (8KB).
al_$ptr   offset from al_$mag to pointer table.  Pointer table MUST
                 PRESENT.
al_$wdt   offset from al_$mag to actual width table.  Not used, reserved

al_$fnt   offset from al_$mag to start of font data.  MUST BE PRESENT.
al_$orp   offset from <al_$fnt> to out_of_range character font data
                 (optional -- if -1 then PRO/GIDIS substitutes its own
                 out_of_range_character)
al_$orw   actual width of out_of_range character - reserved
```

2. Pointer table (word wide)

```
<al_$pnt>          offset from <al_$fnt> to the font information for
                          the character with index <al_$fst>
<al_$pnt> + 2      offset from <al_$fnt> to the font information for
                          the character with index <al_$fst> + 1
 . . .

<al_$ptr> + 2 * ( <al_$ext> - 1)
                  offset from <al_$fnt> to the font information for
                          the last character in the alphabet
```

If the 'offset' value is -1, then treat the character as if it were out of range.

3. Font information (byte or word wide table, starts on 64

byte boundary)

This data starts at <al$fnt>. All pointers to this block are relative to the start of this block.

Font information is stored for byte wide cells as:

```
                            right portion   left portion

+--------------------->  7  6  5  4  3  2  1  0   top scan line
|                  (+1)  7  6  5  4  3  2  1  0   next to top scan line
|                . . .
|                  (+n)  7  6  5  4  3  2  1  0   bottom scan line
|
+--- pointer in <al_$ptr> ...
```

Font information is stored for word wide cells as:

```
     right portion of character     left portion

+-->  15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0  top scan line
|(+2) 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0  next to top scan
                 . . .                                             lin
|(+2n)15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0  bottom scan line
|
+--- pointer in <al_$ptr> ...
```

Word wide cells MUST BE ON WORD BOUNDARIES. We recommend the font table be set up with all word wide cells before any byte wide cells, especially with an odd height. PRO/GIDIS will only support alphabets with total sizes less than 1 APR's worth (8KB).

Although the header is the only entity required to be in a fixed place, we strongly recommend that the font information be last and on a 64 byte boundary, so that future implementations could use 1 APR to map the header information and 1 APR to map the font data. Our suggested ordering:

- header (at the start of the partition)
- pointer table and actual width table (in either order)
- font table (on a 64 byte boundary) (word wide cells then byte wide cells)

APPENDIX D

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, DRAW_REL_ARC and for cell movement after DRAW_CHARACTERS or DRAW_PACKED_CHARACTERS 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.

RUBBER BAND

There are two types of rubber band available in PRO/GIDIS; the rubber band line and the rubber band rectangle. The line stretches from the base position to the current position. The rectangle has one corner at the base position and the opposite corner at the current position. The rectangle will degenerate to a line or point if the current position and base position are the same in one or both coordinates.

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.

PRO/GIGIS Manual
Order No. AA-Y660A-TK
AD-Y660A-T1

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.

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

Please cut along this line.

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

94-003|115/25

Update Notice Number 1

PRO/GIDIS Manual
AD-Y660A-T1

April 1984

Insert this page in the PRO/GIDIS Manual to maintain an up-to-date record of changes to the manual.

NEW AND CHANGED INFORMATION

This update reflects software changes and additions made in P/OS Version 2.0. Also included are additional corrections to the documentation.

INSTRUCTIONS

Add the following pages to the PRO/GIDIS Manual as replacements for or additions to current pages. The technical changes made on replacement pages are indicated in the outside margin by change bars. Text deleted on replacement pages are indicated in the outside margin by bullets. Changes of an editorial nature are not marked.

OLD PAGE	NEW PAGE
Title page/copyright page	Title page/copyright page
iii through ix	iii through xi
1-5, 1-6	1-5 through 1-6.1
1-13, 1-14	1-13, 1-14
1-17, 1-18	1-17, 1-18
1-25 through 1-32	1-25 through 1-32

2-7, 2-8	2-7, 2-8
3-1 through 3-4	3-1 through 3-4
3-9 through 3-12	3-9 through 3-13
4-3, 4-4	4-3, 4-4
6-1 through 6-4	6-1 through 6-4
7-3, 7-4	7-3, 7-4
Chapter 8	Chapter 8
9-3, 9-4	9-3, 9-4
Chapter 10	Chapter 10
Appendix A	Appendix A
B-3, B-4	B-3, Blank
Appendix C	Appendix C/Appendix D
Index	Index
Reader's Comments/Mailer	Reader's Comments/Mailer