



AT&T

999-801-305IS
For use with 3.51 Software

AT&T UNIX[®] PC
Virtual Device Interface
Programmer's Guide

© 1985 AT&T
All Rights Reserved
Printed in USA

NOTICE

The information in this document is subject to change without notice. AT&T assumes no responsibility for any errors that may appear in this document.

GSS-TOOLKIT; GSS-DRIVERS; GSS-TERMINAL; and GSS-CHART are trademarks of Graphic Software Systems, Inc.

Epson MX-series is a trademark of Epson America, Inc.

Microline 84, 92, and 93 are trademarks of Okidata Corp.

VT100 is a trademark of Digital Equipment Corporation.

Retro-Graphics is a trademark of Digital Engineering Corp.

Tektronix is a trademark of Tektronix, Inc.

Diablo is a registered trademark of XEROX CORPORATION.

HILOT DMP-29 Plotter is a registered trademark of Houston Instruments.

ZETA 8 Plotter is a trademark of Nicolet Instrument Corp.

Printronic is a registered trademark of Printronix, Inc.

STROBE is a registered trademark of STROBE, Inc.

Summagraphics is a registered trademark and Summatablet is a trademark of Summagraphics Corporation.

CIT-101e Terminal, CIT-161 Terminal, CIG-101e Graphics Card and CIG-267 Graphics Card are trademarks of CIE Terminals, Inc.

Contents

1	Overview	1-1
	Background	1-1
	GSS-DRIVERS Features	1-5
	GSS-DRIVERS Functions	1-7
	Creating Graphics Applications	1-9
	Error Codes	1-10

2	The GSS Graphics Reference Model	2-1
	The Reference Model	2-1
	GSS-DRIVERS Graphics Functions	2-4
	Programming Tips	2-10
	Pseudocode Example	2-16

3	System Architecture	3-1
	The Components of GSS-DRIVERS	3-1
	Error Codes	3-6

4	Installation and Operation	4-1
	Overview	4-1
	Installing the VDI Software	4-2
	Setting the Environment	4-10
	Linking Applications	4-11
	Testing the System	4-12

5	Control Functions	5-1
	Control Functions	5-1
	Output Functions	5-37
	Attribute Functions	5-61
	Input Functions	5-99
	Inquiry Functions	5-115
	Input/Output Functions	5-139

Glossary

Appendix A: C Conventions and Example	A-1
--	-----

Appendix B: Error Codes	B-1
--------------------------------	-----

Appendix C: Device- Dependent Information	C-1
--	-----

Appendix D: ASCII Code Chart	D-1
---	-----

Appendix E: Bibliography	E-1
---------------------------------	-----

Index

List of Figures

FIGURE 1-1	Graphics Standards	1-3
FIGURE 1-2	GSS Product Line	1-4

FIGURE 2-1	Graphics Program Flow	2-10
FIGURE 2-2	Gantt Chart	2-18

FIGURE 5-1	Angle Specification	5-41
FIGURE 5-2	Cell Array	5-48
FIGURE 5-3	Filled Area	5-52
FIGURE 5-4	Polyline	5-57
FIGURE 5-5	Polymarkers	5-59
FIGURE 5-6	Text Alignment	5-93



List of Tables

TABLE 1-1	Text Options	1-6
-----------	--------------	-----

TABLE 5-1	Control Functions	5-4
TABLE 5-2	Output Functions	5-39
TABLE 5-3	Attribute Functions	5-62
TABLE 5-4	Input Functions	5-100
TABLE 5-5	Inquiry Functions	5-116
TABLE 5-6	Input/Output Functions	5-142

TABLE B-1	Generic Codes	B-2
TABLE B-2	Specific Codes	B-4

1 Overview

Background	1-1
Graphics Standards	1-1
Standards Implementation	1-3

GSS-DRIVERS Features	1-5
-----------------------------	-----

GSS-DRIVERS Functions	1-7
Device Driver Management	1-7
Coordinate Transformation	1-8
Character and File I/O	1-8
Graphics Emulation	1-8

Creating Graphics Applications	1-9
---------------------------------------	-----

Error Codes	1-10
--------------------	------

Overview

This section is an overview of the features and benefits of GSS-DRIVERS. It explains how you can use it to create a device-independent as well as an operating system-independent graphics environment for the AT&T UNIXTM PC.

Background

Let's begin with a little background on computer graphics standards and the products related to GSS-DRIVERS to clarify its role on the UNIX PC.

Graphics Standards

Standards organizations in both the United States and in Europe have been working for over a decade to generate both a graphics model for understanding and communicating graphics information, as well as a formal specification for the implementation of computer graphics.

At first, progress was slow due to the complexity of the subject of computer graphics and its many specific applications. Also, at that time, computer graphics was expensive and primarily relegated to institutions with large budgets—aerospace firms or prestigious universities. Since the market for computer graphics was limited, and there were only a few manufacturers of graphics equipment, the need for standardization was satisfied by “de facto” standards established by the dominant manufacturers.

With the emergence of integrated circuits powerful enough to support graphics and inexpensive enough to be available to the masses came the widespread use of computer graphics. Now graphics is employed in virtually every application area that can

use a computer because it is a much more effective and pleasant way to represent information. In some cases, entirely new applications are enabled by computer graphics, such as computer-aided drafting and engineering.

The result of this development is a strong desire for standardization to create order in an industry rapidly expanding with new products and suppliers. Standardization of computer graphics makes programmers more productive by providing them with a stable graphics programming model. It also provides the opportunity to develop application programs that are independent of the computer system and the graphics input and output devices employed.

The Association for Computing Machinery (ACM) and specifically SIGGRAPH (Special Interest Group on Computer Graphics) became involved early in the standardization effort. They published a pseudo-standard in 1979 called the CORE System paving the way for later work by American and international standards organizations.

The efforts of the American National Standards Institute (ANSI) and the International Standards Organization (ISO) are now resulting in standards that create a uniform conceptual model of computer graphics for system designers, programmers and users, making device-independent graphics a viable goal. The standards are structured around three main interfaces—one at the programmer level, one at the device level and one for communications between computers (see Figure 1-1).

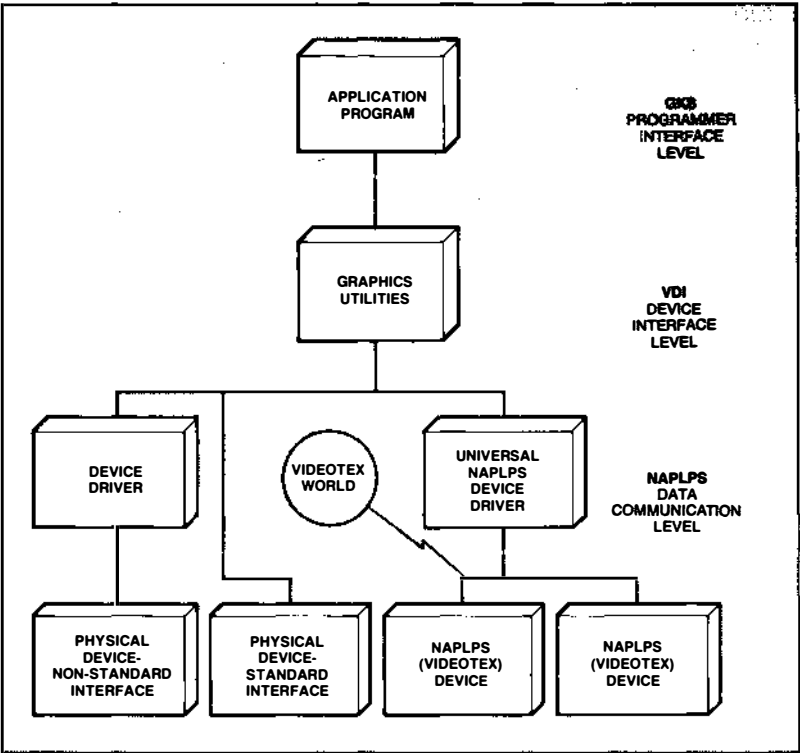
The Graphical Kernel System, or GKS, defines a programmer interface to graphics. It provides source code portability by standardizing the functions and calling conventions of graphics subroutine libraries used by programmers to develop graphics applications.

At the device level, the Virtual Device Interface, or VDI, allows device-independence by creating a logical graphics device interface. This interface allows a computer to control any graphics peripheral without regard for its individual peculiarities. Specific devices can then be matched to the generalized device interface by means of individual device driver programs, one driver for each device.

A similar scheme is employed for graphics data transfer between processors over communications channels. Again, a logical interface allows different computers to communicate by

employing interface programs that are compatible with the standard protocol. In this case, the standards are the Virtual Device Metafile (VDM) and Presentation Level Protocol (PLP).

FIGURE 1-1 Graphics Standards

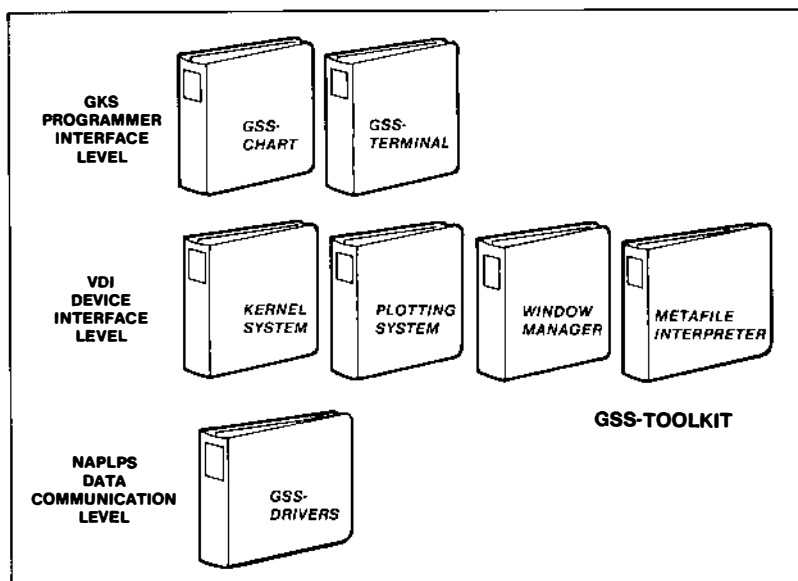


Standards Implementation

GSS-DRIVERS provides device-specific drivers based on the VDI standard for graphics peripherals. Besides standardizing graphics applications, GSS-DRIVERS greatly reduces the programming effort required to create a graphics program. The low level primitive operations and device peculiarities are

transparent to the programmer. Language bindings (interfaces) enable the programmer to incorporate graphics as subroutine calls based on the VDI standard interface.

FIGURE 1-2 GSS Product Line



GSS-DRIVERS Features

GSS-DRIVERS provides the following graphics capabilities through the standard VDI interface:

- Normalized coordinate to device coordinate transformation
- Drawing of graphics primitives (objects)

polylines	circles
polymarkers	arcs
text	pie slices
bars	cells

- Specification of attributes

- color
- fill
- style
- line style
- line width

- Text manipulation (Three text options are available: alpha text, graphics text and cursor text. Manipulations possible with each mode are listed in Table 1-1.)

- location
- size
- rotation
- font
- color
- underlining
- overstrike
- super- and subscripting
- alignment

- Cursor positioning
- Graphics input
- Status inquiry
- Generates standard metafiles

Since capabilities vary among graphics devices, GSS-DRIVERS is designed to emulate certain features. It also provides feedback to the caller about the actual capabilities of the currently open device.

TABLE 1-1 Text Options

<u>Manipulations</u>	<u>Cursor Text</u>	<u>Alpha Text</u>	<u>Graphics Text</u>
Positioning	Character Cell Boundary	Anywhere on display surface	Anywhere on display surface
Scaling	No	Yes*	Yes
Rotation	No	No	Yes
Multiple Fonts	No	Yes	Yes
Bold Text	Yes	Yes	No
Color Selection	Yes	Yes	Yes
Underlining	Yes	Yes	Yes**
Overstriking	No	Yes	No
Super- and Subscripting	No	Yes	No
Quality Levels	No	Yes	No
Line Spacing	No	Yes	Yes
Reverse Video	Yes	No	Yes***
Blink Text	Yes	No	No
Text Alignment	No	No	Yes
Variable Text Height	No	Yes*	Yes
Control Mode	Cursor Addressing Mode	Graphics Mode	Graphics Mode

* The size of alpha text can be changed via the SET ALPHA TEXT FONT AND SIZE function.

** Graphics text can be underlined by specifying polyline primitives underneath the character string.

*** Reverse Video can be selected for graphics text via the SET WRITING MODE function.

GSS-DRIVERS Functions

GSS-DRIVERS provides an environment for the creation of graphics applications that can be transported to any system conforming to graphics standards. GSS-DRIVERS' high-level language bindings incorporate system-independent interfaces for file and character I/O to insure that graphics applications will be completely portable.

The device drivers are stored as executable files on system mass storage. When a workstation is opened, the appropriate driver is spawned as a new process that lives until the workstation is closed.

Specifically, GSS-DRIVERS provides these functions:

- device driver management
- coordinate transformation
- character and file I/O for high-level languages
- emulation of certain graphics primitives
- error reporting

These functions are discussed in more detail in Part 3, "System Architecture."

Device Driver Management

A system may potentially have many graphics input and output peripherals attached, each requiring a specific device driver to interface it to the system. GSS-DRIVERS receives requests for graphics peripherals from an application and spawns the proper device driver process when the workstation is opened.

GSS-DRIVERS also incorporates a metafile driver that generates ANSI standard metafiles. This capability is invoked simply by assigning "METAFIL" as the output device. The metafile device driver creates a metafile that is saved on the system storage device.

Metafiles may be read using a separate product called Metafile Interpreter. See the *AT&T UNIX PC GSS-TOOLKIT Metafile Interpreter Programmer's Guide*.

Coordinate Transformation

Graphics information is passed to GSS-DRIVERS in normalized device coordinates (NDC 0-32767) that are independent of any particular device. The system uses information obtained from the device driver to scale the normalized coordinates to device coordinates that are consistent with the values used by a particular graphics device (for example, raster steps). It also transforms device coordinates into normalized coordinates on input.

Character and File I/O

Even though graphics device calls are standardized, it is still difficult to transport graphics applications across operating systems because of differences in the way a program performs character I/O (for example, when communicating with another computer). Therefore, GSS-DRIVERS' high-level language bindings provide a set of generic character I/O routines that standardize this interface as well.

GSS-DRIVERS also provides a file I/O subsystem. Again, the goal is to make graphics applications completely portable to many systems. The file system is modeled after the UNIX hierarchical file structure including multilevel directories.

Graphics Emulation

The VDI is a robust standard that incorporates many advanced graphics capabilities. Frequently, the capabilities offered by a graphics device are only a subset of the total VDI possibilities. To insure application portability, some functions that are not supported by the device directly are emulated by GSS-DRIVERS.

Creating Graphics Applications

The graphics functions of GSS-DRIVERS can be employed in two ways. The first way is to access them directly through low-level assembly language routine calls with the appropriate parameter lists. However, the calling mechanism is system dependent. The result is applications which are device-independent, but operating system-dependent.

A second, preferred method insures computer-independence. This method uses language bindings that access graphics functions directly as high-level language calls with formal parameters. The C language binding provides an interface between the language source and the VDI interface. Calls in the source conform to the binding function definitions. The binding interface is linked in as an external subroutine library, insuring that your application will be completely portable across any GSS-DRIVERS-compatible system.

After linking, the graphics application may be loaded and run just like any other program.

The exact process for generating a new application is system specific. See Part 4, "Installation and Operation" for detailed information. Appendix A, "Conventions and Example," contains an example program.

Error Codes

GSS-DRIVERS returns a set of error codes to inform the application program of any unusual conditions. A detailed summary and explanation of error messages is contained in Appendix B, "Error Codes."

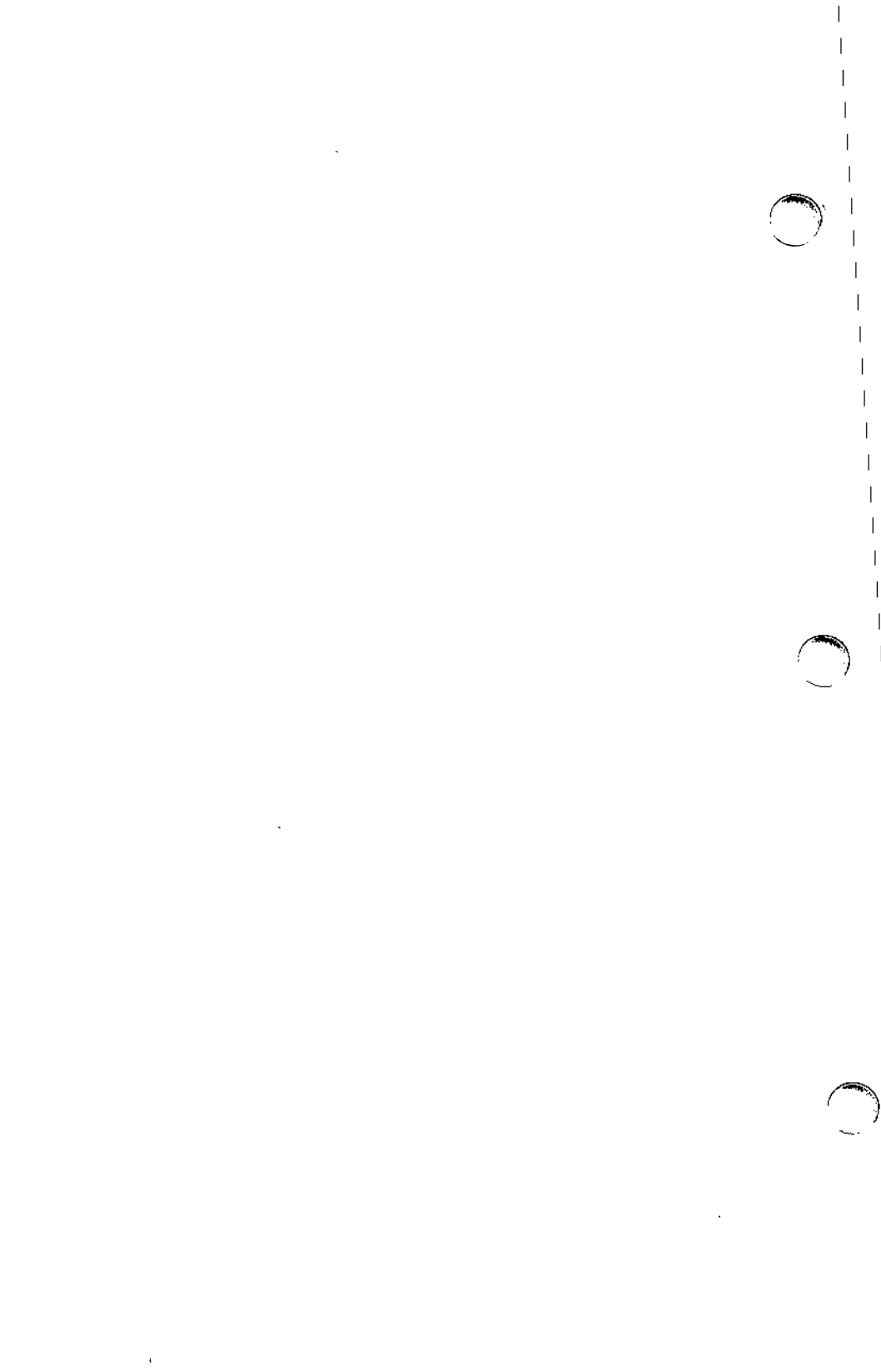
2 The GSS Graphics Reference Model

The Reference Model	2-1
Graphics Output	2-1
Graphics Input	2-2
Transformations	2-2
Workstations	2-2
Additional Elements	2-3

GSS-DRIVERS Graphics Functions	2-4
Control Functions	2-4
Output Functions	2-4
Attribute Functions	2-6
Input Functions	2-7
Inquiry Functions	2-8

Programming Tips	2-10
Control	2-10
Coordinate Transformation Flag	2-11
Prompting Flag	2-11
Device-Specific Data	2-12
Control Mode	2-12
Error Handling	2-13
Setting Attributes	2-13
Color Indices	2-14
Text Rotation	2-14
Graphics Primitive Output	2-14
Input and Inquiry	2-15

Pseudocode Example	2-16
---------------------------	------



The GSS Graphics Reference Model

This part describes the graphics reference model for computer graphics on which GSS-DRIVERS is based. An understanding of this model will clarify the organization of the later parts of this manual and will help you to use GSS-DRIVERS in an application program.

The Reference Model

The Reference Model is a result of the standards effort described in Part 1. It is built on the following basic concepts:

- graphics output
- graphics input
- transformations
- workstations

Graphics Output

Graphics output functions are abstractions of the basic actions an output device can perform, such as drawing a line or displaying text. Output functions include two groups of basic elements called output primitives and output attributes. Primitives result in visible images on the display surface. Attributes modify the appearance of displayed objects, by changing their color, for example. Later in this part how GSS-DRIVERS implements these functions is discussed.

Graphics Input

Information input from a device as a result of an operator action is called graphics input. For example, graphics input occurs when an operator “picks” a location on the display surface with a mouse or crosshair cursor, or types in text at a keyboard.

Transformations

The point coordinates of graphics images must often be translated between different coordinate systems. This occurs at the application level when an operator wishes to scale or rotate an image. It also occurs when coordinates must be modified so that an image can be reproduced on graphics display devices of different sizes.

The VDI standard is based on a special coordinate system called the Normalized Device Coordinate (NDC) System. In NDC space, locations are expressed as Cartesian coordinates with values between 0 and 32767 representing the full coordinate space for all devices. GSS-DRIVERS then transforms the NDC points into the appropriate device coordinates.

Workstations

The graphics reference model employs the concept of a “workstation” to mean zero or more display devices, and zero or more input devices such as a keyboard and a mouse. At upper levels of the graphics system hierarchy (programmer level), workstations can be used to refer to a complete worksite, eliminating the need to reference each device explicitly. The VDI level deals with each individual device explicitly, so “workstation identifiers” are used to refer to a single generic graphics device such as a display, printer or plotter.

Additional Elements

Other concepts included in the reference model such as windows, viewports, clipping and segments are implemented at higher levels (in the Kernel System, for example) and will not be discussed in connection with GSS-DRIVERS. Refer to *AT&T UNIX PC GSS-TOOLKIT Kernel System* for a discussion of these standard graphics functions and their implementation.

GSS-DRIVERS Graphics Functions

The graphics functions, supported by GSS-DRIVERS, can be divided into five functional areas:

- Control functions
- Output functions
- Attribute functions
- Input functions
- Inquiry functions

Control Functions

The control functions allow the application program to control various aspects of the graphics subsystem. The major control functions are:

Open Workstation	Initializes a graphics device and sets defaults for attributes. It also returns information to the caller about the characteristics and capabilities of the device. This must be the first graphics operation performed in a program.
Close Workstation	Terminates graphics operations to a device. This must be the last graphics operation performed in a program.
Clear Workstation	Clears the surface of the workstation. It clears a CRT screen, prompts for new paper on a plotter, or displays all pending graphics to a printer and advances to top-of-form.
Update Workstation	Displays all pending graphics.

Output Functions

Output primitives are functions that generate graphics objects on the display surface. Point locations are specified by giving X,Y coordinates in Normalized Device Coordinate (NDC) space—a

Cartesian space with values ranging from 0 to 32767 on each axis. (Locations outside this range may generate unpredictable and device-dependent results.) Output primitives include:

 Polyline	Draws single vector or series of connected vectors specified by their vertices. A polyline must have at least a beginning and end point specified.
Polymarker	Draws a marker symbol at specified locations.
Graphics Text	Displays a text string at a specified location. Characters come in multiple fonts, and may be rotated or scaled (unlike alpha or cursor text).
Cursor Text	Displays a text string on a fixed rectangular grid. For form fill-out, cursor text is available in only one size and font, is not rotatable, and cannot be combined with graphics text, alpha text, or graphics. Cursor text is typically positioned on a discrete, device-dependent grid of rows and columns. (Typical values are 24 rows by 80 columns.)
 Alpha Text	Displays a text string at a specified location. Alpha text has multiple fonts, variable interline spacing, underlining capabilities and other attributes used for generating formal documents.
Filled Area	Causes an area bounded by a specified set of vertices to be “painted” with a fill pattern.
Bar	Draws a bar (rectangular area) defined by two diagonally opposite vertices. The interior is painted with a fill pattern.
Arc	Draws an arc defined by a center point, the radius, and the starting and ending angles.
Pie Slice	Draws a pie slice defined by the center, radius and starting and ending angles. The interior is painted with a fill pattern.
 Circle	Draws a circle defined by a center point and the radius. The interior is painted with a fill pattern.
Cell Array	Draws a rectangular array of pixels of a specified size at a specified position. The pixel colors that make up the array are defined by a set of color indices.

Attribute Functions

Primitive attributes modify the appearance of output primitives. GSS-DRIVERS provides the following groups of primitive attribute functions:

Character Attributes Character attributes fall into two groups: graphics text attributes and alpha text attributes.

Graphics text attributes control:

- ☐ height
- ☐ base line rotation
- ☐ color
- ☐ font
- ☐ alignment

Alpha text attributes control:

- ☐ position
- ☐ line space
- ☐ font and size
- ☐ color
- ☐ subscript/superscript
- ☐ underline
- ☐ overstrike
- ☐ output quality (draft or final)

Polyline Attributes Polyline control:

- ☐ type
- ☐ width
- ☐ color

Polymarker Attributes Polymarker control:

- ☐ type
- ☐ size
- ☐ color

Fill Attributes	Fill control: □ type □ style □ color
Color Representation	Assigns color values specified in red, green and blue primaries to color index numbers. Color attributes are then selected by index number.
Background Index	Sets color index for display background.

Input Functions

Input functions return information from the operator. Input functions operate in two modes: sample and request mode. In the sample mode, the input, if any, is returned immediately. In the request mode, the operator must complete the input function by actuating some control on the graphics input device (such as a button on a mouse). There are five types of graphics input:

Input Locator	The Input Locator Function returns the point location of the graphics input device (mouse, crosshair, joystick, trackball, cursor, etc.) in NDC units.
Input Valuator	The Input Valuator Function returns a scalar value between 0 and 32767, corresponding to the status of a valuator device (potentiometer, slide control, etc.).
Input Choice	The Input Choice Function returns the status of a choice device (switch, function key, etc.) as an integer between 0 and 32767.
Input String	The Input String Function allows text input from the keyboard.
Read Cursor Movement Keys	The Read Cursor Movement Keys function returns the direction of cursor movement.

Inquiry Functions

Inquiry functions return information about the current state of the graphics subsystem including device capabilities and current attributes. Inquiry functions are provided to determine:

- | | |
|------------------------------|--|
| Graphics primitives | <ul style="list-style-type: none">□ current polyline attributes□ current polymarker attributes□ current fill area attributes□ current graphics text attributes |
| Cursor addressable text | <ul style="list-style-type: none">□ the number of addressable character cells□ the current cursor address |
| Alpha text | <ul style="list-style-type: none">□ alpha text capabilities□ alpha text position□ alpha text font availability□ alpha text string length |
| Additional functions control | <ul style="list-style-type: none">□ cursor text position (direct cursor text addressing—relative and absolute cursor text movement)□ cursor text string and screen erase□ reverse video□ graphics input cursor display□ hard copy output□ plotter pen speed□ raster writing mode (sixteen Boolean operations between source and destination are supported) |

The details of all the GSS-DRIVERS graphics functions are provided in the subsequent reference parts.

Besides creating a basis for development of graphics standards, the Reference Model provides the programmer with a conceptual framework to aid in organizing an application. It also aids the programmer in communicating with other programmers and users and to interpret graphics tools, such as GSS-DRIVERS, by providing a structural context. The rest of this manual is based on the organization defined by the Graphics Reference Model.

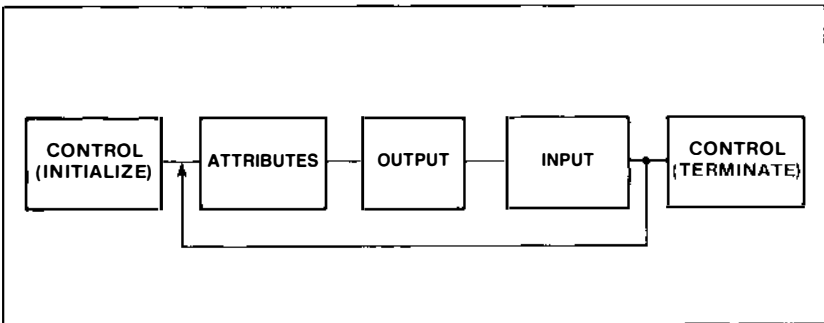
Programming Tips

A typical graphics application will consist of the following five steps:

- Set up graphics control
- Define graphics primitive attributes
- Output graphics primitives
- Input primitives or inquire on status
- Terminate graphics

The intermediate three steps may be repeated numerous times prior to the last step. The general flow of a graphics application program is illustrated in Figure 2-1.

FIGURE 2-1 Graphics Program Flow



Control

The first command the application should invoke is Open Workstation. This command defines the type of device to be used, loads the device into memory if necessary, and returns information to the user regarding the capabilities of the device.

Coordinate Transformation Flag

A key parameter specified by Open Workstation is the coordinate transformation flag. This flag indicates the aspect ratio of the display surface. The user may specify a display surface with 32767 addressable locations in both directions or one with 32767 addressable locations on the longest side (axis) and a lesser amount on the shorter side (axis). The number of addressable locations on the shorter side will be proportional to the actual aspect ratio of the physical display.

The benefit of having the number of addressable locations match the aspect ratio of the display surface is that a “unit distance” is the same length along both axes. The disadvantage is that not all devices have the same display surface aspect ratio, and portions of the output may not fit on all displays.

The advantage of specifying 32767 locations along both axes is that the aspect ratio of the working display surface is always the same, no matter what the shape of the physical display surface. A programmer’s application output will always fit on the display surface if his parameters stay within the range 0 to 32767. This provides for device-independent graphics applications.

Prompting Flag

Another parameter the user must provide to the Open Workstation command is the prompting flag. The use of this flag is important when the graphics output device is a plotter, printer, or other device possibly requiring user attention. For example, on a two-pen plotter, the user may wish to display a chart containing four colors. When colors three and four are needed, the device driver will prompt the user to change the pens. The prompting flag indicates whether the user is prompted or whether the device driver ignores the color change commands. This may be useful for preliminary debugging runs of a program when the actual number of colors output is not important.

Device-Specific Data

Open Workstation also returns an array of information detailing the capabilities of the device requested. This information includes the number of colors available, the number of text sizes available, the number of line styles available, etc. The application programmer can apply these facts to utilizing a device to its best advantage. For example, if a device has multiple colors, different lines of data on a chart can be represented with solid lines in different colors. If black and white are the only colors available, different line styles (combinations of dashes, dots and spaces) can be used to differentiate the lines of data on the same chart output to a different device.

Another important device-specific parameter returned by Open Workstation is the number of different text sizes available. A few devices allow for text to be continuously scaled from one unit high to 32767 units high. Most other devices only provide four to six different text sizes. The programmer should consider the number of scaling options and the actual text sizes needed for the application.

Control Mode

After opening the workstation, the programmer should make sure the device is in the proper mode. The default condition sets Graphics Mode to "on" and Cursor Addressing Mode to "off." Certain functions only work when Cursor Addressing Mode is on, and other functions only when Cursor Addressing Mode is off. Cursor Addressing Mode is only applicable to CRT devices.

Cursor-addressable text is mutually exclusive with graphics text and alpha text. Graphics text and alpha text are compatible and can be displayed simultaneously on the same display surface. The user should remember to call Enter Cursor Addressing Mode prior to executing any cursor addressable commands.

If the device is not in Cursor Addressing Mode when a cursor-addressing command is executed, the exact results are not defined. This may "hang" the device and require restarting. In some instances, the device may ignore the cursor addressing command. On rare occasions, it may actually execute the



desired function. All possible side effects are not listed here since the actual effect depends on the state of the device prior to the cursor addressing command. Similarly, if graphics commands are issued when Cursor Addressing Mode is on, the results are not well defined.

Error Handling

Open Workstation and all other routines also return a value that indicates whether the requested command was completed successfully. A zero or positive value indicates a success; a negative value indicates unsuccessful. When using GSS-DRIVERS high-level language bindings, if a negative value is returned after issuing a call to Inquire VDI Error can be invoked to identify the specific error. Errors are described in Part 3, "System Architecture," and a table of error values is provided in Appendix B.



Device drivers seldom return errors. They are programmed to perform the operation requested to the best of their ability. However, they will return file and communication I/O errors. Such errors prevent the driver from completing an operation.

Cursor Addressing Mode has no effect on printers. It does not return errors but rather the functions are ignored. Inquire Current Cursor Text Address will return that the cursor text is not available.

Setting Attributes



Attribute setting routines always return the attribute value selected. This is either the closest value to the one requested or the specified default in cases where the requested value is out of range. For example, when the default line style is index 1 (solid), out of range line types are mapped to 1.

Color Indices

Color index attribute setting routines select the closest index to the one requested. This can have interesting side effects. If a negative color index is requested, the closest index that will be selected is 0. However, the default value for color index 0 is black. If the CRT background color is black, all subsequent primitives with the default index are invisible on the CRT.

If the programmer wants a specific color associated with a particular pen station, he must inform the user. This can be done several different ways, depending on the device. The programmer could document desired colors in the application manual. Alternatively, a message could be sent to the CRT at the beginning of the program informing the user of the colors expected for each color index and pen station.

Text Rotation

Another attribute, text rotation, is specified via the function Set Graphics Text String Baseline Rotation.

Graphics Primitive Output

Output functions are available to define graphics primitives including polylines, polymarkers, arcs, circles, bars and pie slices. Functions are also provided to describe alpha, graphics and cursor text. Locations are specified in Normalized Device Coordinates. In general, GSS-DRIVERS will honor any graphics primitive output specifications supplied by the programmer.

However, when using the aspect ratio preservation mode, scaling primitives may be partially or totally clipped. For text and markers, the entire character or marker must be visible for it to be displayed.

Also, when specifying fill, remember that filled areas are not outlined unless the fill interior style is hollow. To display a filled area (polygon, bar, circle, pie slice) with an outline, first fill the area with the fill style of solid, pattern or hatch. Then specify fill interior style of hollow for the same area.



Another point to remember is that the radius specified for circles and arcs is assumed to be along the x-axis. The specified radius takes priority over the radius that is determined by the center point and an arbitrary point of the arc/circle. Since circles and arcs are specified by a centerpoint and a radius, a circle cannot be displayed if its center point is not on the display surface.

Input and Inquiry



Functions that return arrays of data may overwrite application data, if the amount of space allocated is smaller than the size given to the VDI. For example, when invoking either Input String (request mode) or Input String (sample mode), the user may specify a different value via the maximum-length parameter than required by the defined array. If the maximum-length parameter indicates that forty characters can be input and the user only defines an array that is ten elements long, an additional thirty characters may overwrite a portion of memory not intended by the user. The results of this situation depends on the application. In some cases, the error may never be detected.



Pseudocode Example

To help you understand the logical sequence of creating a graphics application with GSS-DRIVERS, we have included a programming example. The object of the example is to create a Gantt Chart representation of a processing plant construction project.

A Gantt Chart shows the component activities of a project by functional area along with their relationship in time. They are conveniently shown in a horizontal bar format where each bar shows the duration of a particular activity. Figure 2-2 shows the desired graphics output. The required GSS-DRIVERS calls are described in pseudocode below.

```
{ This is a pseudocode program to generate a Gantt Chart. Lines within
the braces are comments. }
program gantt;
{ Initialize and dimension variables for use in the rest of the program.
This is done as in any other program. The required data types and
allocations are indicated in the function specifications in the
subsequent reference Part.}
OPEN.WORKSTATION;
{ Opening the workstation sets defaults and returns information like size
of plotting surface, pixel heights and colors that can be used by the
application program to best draw the chart. This is device-specific
information reflecting the characteristics of the device being opened. }

{ Note that OPEN.WORKSTATION returns a "device handle," a number that
uniquely identifies the open device. This is used in subsequent calls
to the device by including it as the first argument in the parameter list.
For simplicity, this will not be shown explicitly in the following
pseudocode. }

for (every grid line)
begin;
    { Create an array that contains the points that define a vertical grid line
    in NDC space (beginning and end points of each grid line). The maximum
    extent in each axis is 32767 assuming we opened the device in coordinate
    transformation mode 0. }
    POLYLINE(grid line);
    { Use POLYLINE to draw the line }
end;
SET.ALIGNMENT(top center);
{ SET.ALIGNMENT changes the text attribute defining how GSS-DRIVERS
places text with respect to the points specified in following graphics
output routines. We use this call to center the text below the point
specified. }
for (every horizontal axis tick)
begin;
    GRAPHIC.TEXT(x position,y position,month[i]);
    { Output the text for the horizontal axis using the default character
    size. Month[i] represents the character string identifying each month. }
end;
```

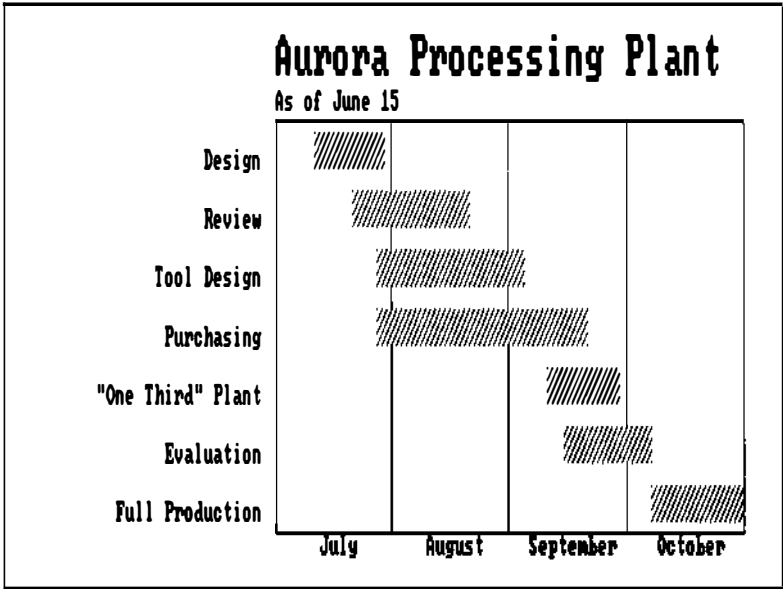
```

SET.ALIGNMENT(right center);
  { We use SET.ALIGNMENT here to position text's right side center line
    at the graphics text position. }
for (every vertical axis label)
  begin;
    GRAPHIC.TEXT(x position,y position,ylabel[i]);
    { Output the text for the vertical axis at the specified points. }
  end;
SET.ALIGNMENT(bottom left);
  { Set the alignment point to the lower left of the text string }
GRAPHIC.TEXT(x position,y position,"As of June 15");
  { Write the subtitle out }
SET.TEXT.HEIGHT(size of title text);
  { Set the height for the title }
GRAPHIC.TEXT(x position,y position,"Aurora Processing Plant");
  { GRAPHIC.TEXT is used to write out the title text }
SET.FILL.INTERIOR(hatch patterns);
  { SET.FILL.INTERIOR is an attribute routine that determines how any area
    is filled. Valid fill types are empty, solid, hatch and pattern.
    Hatch is used to fill the Gantt bars. }
SET.FILL.STYLE(45% narrow);
  { Choose 45% narrow shading }
for (every bar)
  begin;
    BAR(lower left,upper right);
    { Draw the bars }
    POLYLINE(frame);
    { Draw the frame around the axis system }
    POLYLINE(border);
    { Draw the border around the page }
    REQUEST.STRING(2 characters,echo,echo location,input string);
    { Wait before we close work station so viewer has a chance to see picture
      if output device is a CRT. Program continues when carriage return or at
      least two characters are entered. }
    CLOSE.WORK.STATION;
    { Close down workstation }
  end.

```

Language-specific code for this example is shown in Appendix A.

FIGURE 2-2 Gantt Chart



3 System Architecture

The Components of GSS-DRIVERS	3-1
Device-driver Management	3-2
Coordinate Transformation	3-2
Transformation Modes	3-2
Character I/O	3-3
File I/O	3-4
Directories and File Names	3-4
Graphics Emulation	3-4
Error Reporting	3-4

Error Codes	3-6
--------------------	-----



System Architecture

This part is a description of the architecture of GSS-DRIVERS and a summary of the graphics functions available. A detailed listing of each graphics call, its arguments and operation is contained in the subsequent reference part.

The Components of GSS-DRIVERS

GSS-DRIVERS consists of a set of executable device drivers that form the interface between the VDI and specific graphics peripheral devices.

It allows your program to access the hardware on the system in a standard way, thereby eliminating operating system dependencies. This is true of character and file I/O as well as graphics functions. Your application programs will be truly portable between various systems employing GSS-DRIVERS as the graphics subsystem.

GSS-DRIVERS provides several functions for the graphics subsystem:

- device driver management
- coordinate transformation
- character and file I/O
- graphics emulation
- error reporting

GSS-DRIVERS on UNIX is part of the application and is loaded when the application is executed.

Device-driver Management

The UNIX PC software provides drivers for several graphics devices. Select how to install the necessary drivers with the procedure explained in your *AT&T UNIX PC Owner's Guide* for the Set 1 device drivers, and *AT&T UNIX PC GSS-DRIVERS+ User's Guide* for the optional Set 2 device drivers.

The device driver file names must be unique, but there are no other constraints other than making sure they are descriptive. For example, the device driver file name for an AT&T Model 455 Printer is "att455."

Under the UNIX operating system, GSS-DRIVERS is part of the application and is loaded when the application is executed. When GSS-DRIVERS receives a request for a specific device driver via an Open Workstation call, it spawns the appropriate device driver process. This process will remain available to the application until Close Workstation is invoked.

Coordinate Transformation

All graphics coordinates are passed to GSS-DRIVERS in Normalized Device Coordinates (NDC). Here all locations are measured in Cartesian coordinates between 0 and 32767. This lets graphics information be passed to all devices in an identical way regardless of the device size or coordinate system.

GSS-DRIVERS uses information returned from the device driver when the workstation is opened to transform the normalized coordinates into device-specific coordinates (such as raster lines, plotter steps, etc.). This frees an application from performing any device-dependent transforms.

Transformation Modes

There are two user-selectable ways the NDC-to-device coordinate transformation can take place. In the first mode the NDC range of 0-32767 is mapped to the full extent of the physical display surface in each direction. Using this mode insures that all the graphics information will appear on the display surface since all NDC points are displayable.

However, distortion will occur if the display device does not map NDC units to equal physical distances in both directions. (This happens on devices with non-unity aspect ratios.) A result of this situation is squares that turn into rectangles.

To avoid this distortion, use the second transformation mode. It preserves the aspect ratio of the image by mapping NDC units to equal physical distances in both directions. To do this, the full NDC space is mapped to the longest axis of the device. The other axis displays as much of the NDC space as possible, but some information at the edges will be lost. Compensation is provided for devices with non-square pixels so that circles appear as circles and squares look like squares. The application program is responsible in this case for sending only displayable NDC units to the system (otherwise they will be lost). This can result in device- and system-dependencies. However, drivers automatically take device dependencies into account when using the bar, pie slice, arc and circle GDP's.

The first mode unburdens the application from doing a specific device-dependent transform. The advantage of the second mode is that pictures can be easily transported between devices with the assumption that a unity (square) aspect ratio is used. GSS-DRIVERS will make the adjustment for the actual aspect ratio of the display device.

Character I/O

Although the VDI standardizes access to graphics facilities, various operating systems have different ways to access the ports that control graphics peripherals. To insure that graphics applications written using GSS-DRIVERS are truly portable, the C language bindings include functions that provide a standardized way to do character I/O. The character I/O facility includes functions to open, initialize and close the I/O system, to obtain status, and to read and send characters. Characters can be read and written with or without waiting for completion.

File I/O

A system-independent way of accessing files is also needed to make applications portable. GSS-DRIVERS provides a generic file I/O system based on the UNIX file model. A pointer associated with the file indicates the next byte position to be read or written. Each operation (read or write) causes the pointer to be updated. A seek operation allows the current position in the file to be set without reading or writing data.

Directories and File Names

All files reside in distinct areas called directories. A particular directory is specified by name—a null terminated string of bytes. A file is also referred to by name. A fully qualified file name consists of a directory name and a file name with a maximum of eighty characters.

Functions are provided to associate a file name with a file descriptor—an integer number that uniquely identifies an open file. Up to sixteen file descriptors (open files) may exist simultaneously. All file operations require a file descriptor to identify the object file. A file's descriptor must be obtained before any operations can be performed on it.

Graphics Emulation

The VDI is a robust standard that incorporates many advanced graphics capabilities. Often the capabilities offered by a graphics device are only a subset of the total VDI possibilities. To insure application portability, some functions that are not supported by the device directly are emulated by GSS-DRIVERS.

Error Reporting

GSS-DRIVERS functions always return to the caller whether or not the requested operation was successful. Each function returns an error status that indicates the results of the request. However, no errors are displayed by GSS-DRIVERS. The

application program is made aware of the condition of the graphics subsystem and can take appropriate action without losing control of the system. This places a responsibility on the application program for checking error status and attempting error recovery, or at least informing the user.

An explanation of the status codes returned is included in the detailed descriptions of GSS-DRIVERS functions in the subsequent reference part.

Error Codes

If a function returns an error status, the Inquire VDI Error function can be invoked to obtain the applicable error code. Messages associated with each error value are provided in Appendix B, "Error Codes." Error codes are organized into generic and system-dependent information based on the following rules:

- A negative return from a function always implies an error
- An error code greater than or equal to zero indicates no error occurred
- The generic part of the code is in the form "GGxx" (in decimal)
- The system-dependent part is always in the form "xxDD" (in decimal).

4 Installation and Operation

Overview	4-1
Distribution Files	4-1
Installing the VDI Software	4-2
Setting the Environment	4-10
Linking Applications	4-11
Testing the System	4-12



Installation and Operation

This part describes how to install GSS-DRIVERS on the UNIX PC. This part also discusses how to link your application programs with the GSS-DRIVERS application library.

Overview

The GSS-DRIVERS installation procedure requires you to define the following system environmentals:

- Set the default directory for the device driver files to be used by GSS-DRIVERS
- Select which device drivers correspond to logical device names
- Assign logical graphics devices to physical channels

Once these environmentals are defined, you can incorporate graphics operations into your application programs by calling GSS-DRIVERS functions in your source code. The operations and parameters for all GSS-DRIVERS graphic functions are detailed in Part 5. These are referred to as the language “bindings” because they show explicitly how GSS-DRIVERS interfaces to the C programming language. The bindings for the character and file I/O functions are also provided.

Distribution Files

GSS-DRIVERS distribution files contain the C language bindings library, a header include file, and test files.

- | | |
|------------------|--|
| libcvdi.a | C language binding library. |
| types.h | Header include file. |
| vditest | Test programs that verify proper operation. Both the source and the executable files are included. |

Installing the VDI Software

This section contains a guide to installing the software for the AT&T UNIX PC Virtual Device Interface C Binding. Depending upon the version of the operating system being used, some procedures and screen displays may vary from those shown here. Typical variations in the procedure are addressed within this guide, although others may exist.

Before you begin the installation procedure, locate the following floppy disk:

GSS-DRIVERS C Binding Version 1.01

1 When prompted by **Please login:**

- ☐ Type your user login or you can type *install*.
- ☐ Press **(Enter)**.

If a password is required for system access, the **Password:** prompt will appear and you must continue with the following instructions. Otherwise proceed with the instructions in step 2.

- ☐ Type the password exactly as it was defined to the system. As a security measure, the password will not be displayed on the screen when it is typed.
- ☐ Press **(Enter)**.

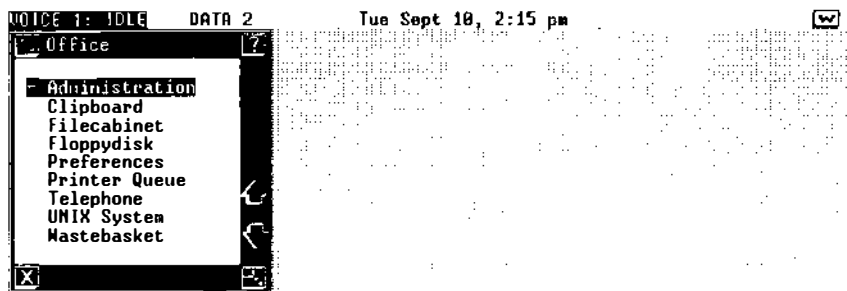
2 The system briefly displays a message that shows the amount of available storage space.

If there is less than 15% of the storage space available, you may want to choose one of the following options:

- Delete unnecessary files from the Wastebasket.
- Backup files from the hard disk onto floppy disks and delete these files from the hard disk.
- Remove software that is no longer required.

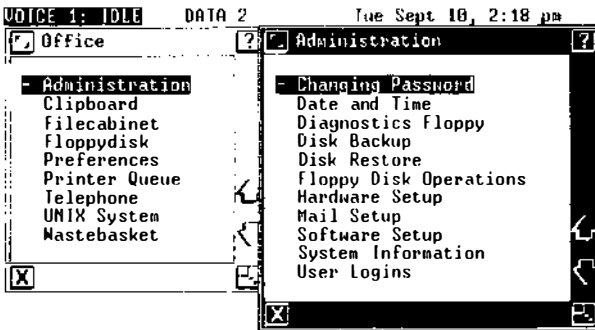
See the AT&T UNIX PC Owner's Manual for details on performing these operations.

- 3 After a few seconds, the system displays the following Office Window with the **Administration** selection highlighted.



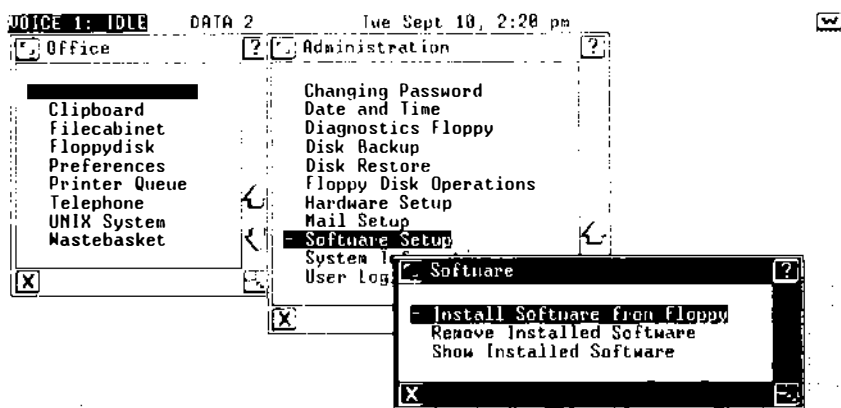
□ Press **(Enter)**.

4 The system displays the Administration Window.



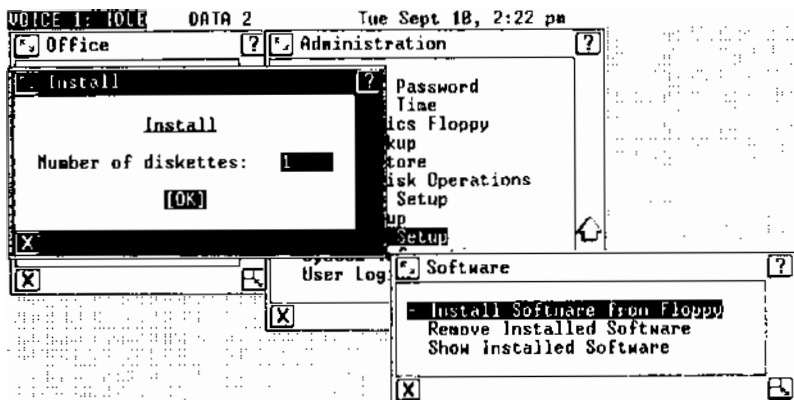
- Press **(Next)** until you have highlighted **Software Setup**.
- Press **(Enter)**.

- 5 The System displays the Software Window with the **Install Software from Floppy** selection highlighted.



- Press **(Enter)**.

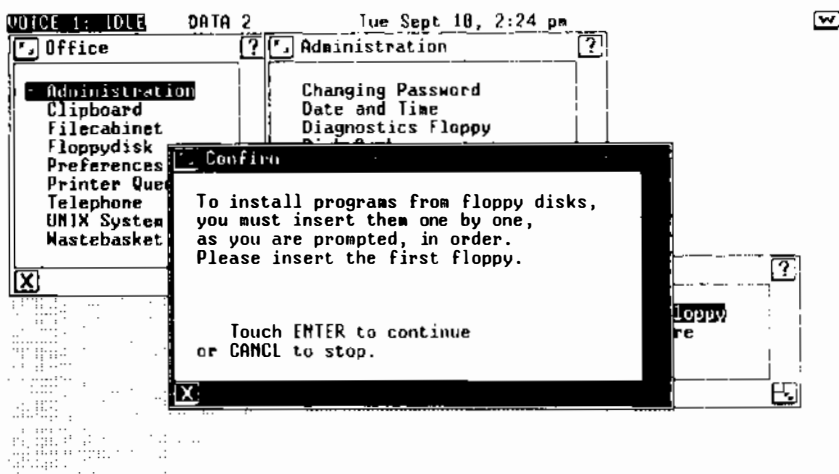
- 6 At this point, depending upon the version of the operating system being used, the system may display the Install Window with the cursor positioned at the prompt **Number of diskettes:** as shown below. If so, continue with the following instructions. Otherwise, proceed with the instructions in step 7.



Type number of floppy diskettes to be installed.

- Enter 1 for the number of diskettes that are to be installed.
- Press **(Enter)**.

- 7 The system displays the Confirm Window with the following message or a similar message instructing you to insert the diskette:

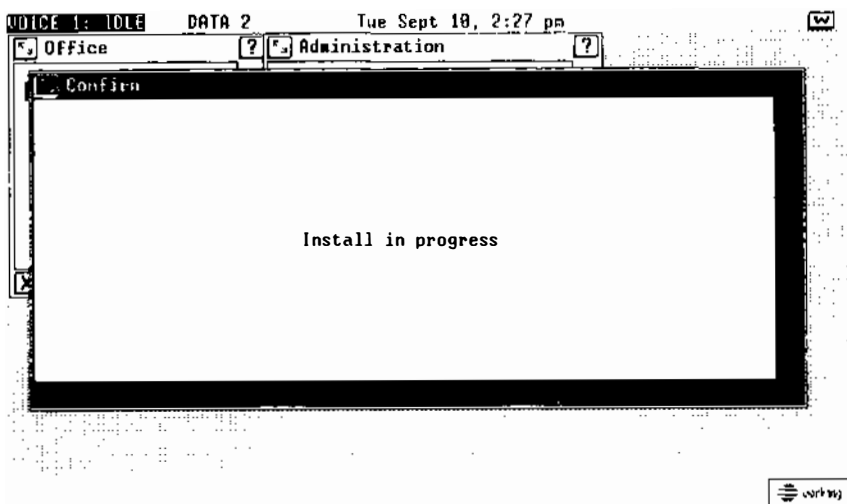


- Insert the **GSS-DRIVERS C Binding Version 1.01** into the diskette drive with the label facing up.
- Close the drive by flipping the lever down.
- Press **(Enter)**.

Note

If an error message appears on the screen during installation, check that the diskette is in good working condition and inserted correctly in the drive.

- 8 Depending upon the version of the operating system being used, the system may display the Confirm Window with the **Install in progress** message as shown below. If so, continue with the following instructions. Otherwise, a screen message informs you of the storage space available on your system and offers you the opportunity to continue by pressing **(Enter)** or to cancel the installation by pressing **(Cancel)**. If not enough space is available, see the options listed under step 2.



- Do not open the diskette drive until it is safe or data can be lost or destroyed.

- 9 When the installation is complete, the system displays the following message:

**The installation of the GSS-DRIVERS C
C Language Binding Version 1.01 package
is now complete.**

Please press Enter to continue.

- Press **(Enter)** as indicated on the screen.
- 10 The system removes the Confirm Window from the screen and displays the Software Window. It is now safe to remove the floppy diskette.
- Open the drive by flipping the drive lever up.
 - Remove the diskette.
 - Press **(Exit)** to return to the Administration Window.
 - Press **(Exit)** again to return to the Office Window.

Setting the Environment

There are several environmental variables that should be set before running your application. Since there is a possibility that several drivers may be resident on your UNIX PC for that device type, you must indicate which driver should be used by GSS-DRIVERS functions. When `v_opnwk` (Open Workstation) is called, you specify a logical device name such as `PRINTER`, `PLOTTER`, `DISPLAY`, etc. The `v_opnwk` then searches the environment for a variable by the same name as the device. This variable holds the file name of the driver.

You must also specify which physical device a driver is to use. This is accomplished by setting an environmental variable with the same name as the device driver, containing the name of the physical device. The variable `VDIPATH` holds the appropriate pathname for GSS-DRIVERS.

These assignments can be made through the "GSS-Drivers Setup" menu. They are inserted in the environment by executing the file

```
/usr/lib/GSS_Drivers/Environment
```

If the GSS-DRIVERS pathname is not provided, the current working directory is used.

The default output metafile name is **metafile.met**. This can be changed with the following commands:

```
METAOUTPUT=<filename>  
export METAOUTPUT
```

where **<filename>** is the name to be used for subsequently created metafiles.

Linking Applications

The code that implements the interface between GSS-DRIVERS and your programming language is contained in the `libcvdi.a` binding library included on the distribution diskette. After your program is compiled, it must be linked with this binding library and other modules and libraries required by the application. In general, the binding should be linked after your application code and before any language support libraries.

The C language binding library, `libcvdi.a`, is located in the system library directory:

`/usr/lib/libcvdi.a`

It can be accessed from any directory by the command:

cc picture.c -lcvdi -ltam -ltermcap -o picture

The option `-lcvdi` is an abbreviation for `/usr/lib/libcvdi.a`. It informs the linker that a library called `/usr/lib/libcvdi.a` is required.

Note

The `-l` switch can be used to associate an argument with an exact library file within `/usr/lib`. The linker will look for the file name that begins with "lib" and ends with the string that follows the `-l` switch. For example, `-ltester` would be the library `/usr/lib/libtester.a`, and `-lm` would be the library `/usr/lib/libm.a`.

Testing the System

The test programs provided in the distribution files can be used to verify that the configuration shell script is correct and that your application is properly linked. If the test programs do not execute successfully, retrace your installation steps to discover the source of the error. Make sure that you have made the appropriate physical device (port) assignment and baud rate selection by exercising your peripherals with previously operating programs.

The executable program module is executed and debugged in the normal manner using the tools provided by the operating system environment. Since GSS-DRIVERS functions always return to the calling program with status information rather than displaying error messages, we recommend that you include a very simple status checking routine in your program that is called after each graphics function. Its purpose is to display any error codes that arise before you have finished debugging your application error handling routines.

5 Control Functions

Control Functions	5-1
Initialization and Termination	5-1
Device Control	5-1
Workstation Control	5-1
Cursor Control	5-2
Metafile VDI Functions	5-3
A Note on Binding Conventions	5-3
v_appl ()	5-5
v_clrwk ()	5-6
v_clswk ()	5-7
v_curdown ()	5-8
v_curhome ()	5-9
v_curleft ()	5-10
v_currright ()	5-11
v_curup ()	5-12
v_dspcur ()	5-13
v_eeol ()	5-14
v_eeos ()	5-15
v_enter_cur ()	5-16
v_exit_cur ()	5-17
v_hardcopy ()	5-18
v_msg ()	5-19
v_opnwk ()	5-20
Color Indices	5-26
Opening an Output Device	5-27
Opening an Input Device	5-27
Code Example—Open Workstation	5-29
v_rmcur ()	5-30
v_updwk ()	5-31
vs_curaddress ()	5-32
vs_penspeed ()	5-33
vsa_position ()	5-34
vswr_mode ()	5-35

Output Functions	5-37
Primitive Functions	5-37
Polyline Function	5-37
Polymarker Function	5-38
Text Function	5-38
Fill Area Function	5-38
Cell Array Function	5-38
Generalized Drawing Primitives	5-38
v_arc ()	5-40
v_atext ()	5-42
v_bar ()	5-44
v_cellarray ()	5-45
v_circle ()	5-49
v_curtext ()	5-50
v_fillarea ()	5-51
v_gtext ()	5-53
v_pieslice ()	5-54
v_pline ()	5-56
v_pmarker ()	5-58
 Attribute Functions	 5-61
Attribute Functions	5-61
v_rvoff ()	5-64
v_rvon ()	5-65
vcur_att ()	5-66
vcur_color ()	5-68
vs_color ()	5-69
vs_editchars ()	5-71
vsa_color ()	5-72
vsa_font ()	5-73
vsa_overstrike ()	5-76
vsa_passthru ()	5-77
vsa_quality ()	5-78
vsa_spacing ()	5-79
vsa_supersub ()	5-80
vsa_underline ()	5-81
vsb_color ()	5-82
vsf_color ()	5-83
vsf_interior ()	5-84
vsf_style ()	5-85
vsl_color ()	5-86
vsl_type ()	5-87
vsl_width ()	5-88

vsm_color ()	5-89
vsm_height ()	5-90
vsm_type ()	5-91
vst_alignment ()	5-92
vst_color ()	5-94
vst_font ()	5-95
vst_height ()	5-96
Code Example—Set Character Height	5-97
vst_rotation ()	5-98

Input Functions	5-99
vrđ_curkeys ()	5-101
vrq_choice ()	5-103
vrq_locator ()	5-104
vrq_string ()	5-106
vrq_valuator ()	5-108
vsm_choice ()	5-109
vsm_locator ()	5-110
vsm_string ()	5-111
vsm_valuator ()	5-113

Inquiry Functions	5-115
vq_cellarray ()	5-117
vq_chcells ()	5-119
vq_color ()	5-120
vq_curaddress ()	5-121
vq_error	5-122
vqa_cap ()	5-123
vqa_cell ()	5-126
vqa_font ()	5-128
vqa_length ()	5-131
vqa_position ()	5-132
vqf_attributes ()	5-133
vql_attributes ()	5-134
vqm_attributes ()	5-135
vqt_attributes ()	5-136

Input/Output Functions	5-139
Communications (Port) I/O	5-140
Disk I/O	5-141
cm_close ()	5-143
cm_inq ()	5-144

cm_open ()	5-145
cm_start	5-146
cm_stop	5-147
cmrx_now ()	5-148
cmrx_wait ()	5-149
cmtx_now ()	5-150
cmtx_wait ()	5-151
fd_close ()	5-152
fd_connect ()	5-153
fd_copy ()	5-154
fd_delete ()	5-155
fd_directory ()	5-156
fd_disconnect ()	5-157
fd_inq ()	5-158
fd_open ()	5-159
fd_parse ()	5-160
fd_read ()	5-161
fd_rename ()	5-162
fd_seek ()	5-163
fd_size (fd)	5-164
fd_write ()	5-165
fdp_read ()	5-166
fdp_write ()	5-167

Control Functions



This section covers the control functions provided by GSS-DRIVERS that allow:

- initialization and termination
- device control
- workstation control
- cursor control

Initialization and Termination



When you initialize GSS-DRIVERS, default values are set for all attributes and the device capabilities are determined. When you terminate the program, control returns to the keyboard. Open Workstation must be the first routine called and Close Workstation, the last.

Device Control

For ultimate device control, GSS-DRIVERS allows you to set the pen speed on a plotter, set the raster writing mode and generate hard copy.

Workstation Control



Control functions are also provided that allow you to update a workstation by making all output current and to clear the workstation.

Cursor Control

You will remember from the Graphics Model discussed in Part 2 that GSS-DRIVERS supplies three types of text:

- Alpha Text used for creating word processor quality text on CRTs or printers
- Cursor Text used primarily for creating forms
- Graphics Text.

To control both Alpha and Graphics Text, you must be in the **Graphics Mode**.

To control the cursor for cursor text you must be in the **Cursor Addressing Mode**. This gives you access to functions that are used for positioning the standard CRT cursor and placing text on the screen.

Note

Cursor addressing only applies to CRT devices.

Cursor addressing is defined on a character cell grid of rows and columns. Rows are the number of lines of text on the screen, and columns are the number of character cells per line. The upper left corner of the screen is row 1, column 1. Typical screen formats are 24 rows by 80 columns.

Metafile VDI Functions



Two special VDI functions are available in GSS-DRIVERS for use with metafiles. The Message function allows message text that is not part of a picture to be placed in the metafile. This text is simply passed on to the metafile interpreter and displayed to the operator at interpretation time. It is intended to permit the display of special device-dependent information needed to process a VDM. There is no control over the position or appearance of the text.

The Application Data function allows an application program to store and access private data in a metafile. When retrieved, this data is not processed in any way by the metafile interpreter, but is available to the application.

A Note on Binding Conventions



In the C bindings, INT16 refers to signed 16-bit integers and INT8 refers to signed 8-bit integers. FD refers to a signed 16-bit integer. INT32 refers to a long integer (32 bit). Scalar input arguments are passed by value, array input arguments are passed by address. All output arguments are passed by address.



TABLE 5-1 Control Functions

<u>Function</u>	<u>Routine</u>	<u>Description</u>	<u>Page</u>
Initialization	v_opnwk	Open Workstation	5-20
Termination	v_clswk	Close Workstation	5-7
Device Control	v_hardcopy	Hard copy	5-18
	vs_penspeed	Set Pen Speed	5-33
	vswr_mode	Set Writing Mode	5-35
Workstation Control	v_clrwk	Clear Workstation	5-6
	v_updwk	Update Workstation	5-31
Alpha Text Control	vsa_position	Set Alpha Text Position	5-34
Cursor Text Control	v_enter_cur	Enter Cursor Addressing Mode	5-16
	v_curup	Cursor Up	5-12
	v_curdown	Cursor Down	5-8
	v_currigh	Cursor Right	5-11
	v_curleft	Cursor Left	5-10
	v_curhome	Cursor Home	5-9
	v_eeol	Erase to End of Line	5-14
	v_eeos	Erase to End of Screen	5-15
	vs_curaddress	Direct Cursor Address	5-32
	v_exit_cur	Exit Cursor Addressing Mode	5-17
	v_dspcur	Display Graphics Input Cursor	5-13
		Remove Graphics Input Cursor	5-30
	v_rmcur		
Metafile	v_appl	Application Data	5-5
	v_msg	Message	5-19

v_appl ()

Purpose	Metafile application data
Syntax	v_appl (dev_handle, function, data_cnt, app_data)
Data Types	INT16 v_appl (); INT16 dev_handle; INT8 function[]; INT16 data_cnt; INT16 app_data[];
Input	dev_handle Metafile device handle function Text string indicating function name data_cnt Number of integers of application data app_data Name of array containing application data
Function Returns	Function v_appl returns error state. 0 if no error -1 if error Actual error can be retrieved by invoking error inquiry function.
Description	This routine allows the metafile generator to place application specific data into the metafile. The function name is a user-defined title for whatever the application data element represents.

v_clrwk ()

Purpose	Clear workstation
Syntax	v_clrwk (dev_handle)
Data Types	INT16 v_clrwk (); INT16 dev_handle;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Function	Function v_clrwk returns error state.
Returns	0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine clears CRT screens, prompts for new paper on plotters, or displays all pending graphics and advances to top-of-form on printers. Often, this call is preceded by an input routine when using a CRT so that output will not be cleared from the screen before the user has a chance to view the image. Prompts can be controlled by Open Workstation.

v_clswk ()

Purpose	Close workstation
Syntax	v_clswk (dev_handle)
Data Types	INT16 v_clswk (); INT16 dev_handle;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Function	Function v_clswk returns error state.
Returns	0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine displays any pending graphics, then stops all graphics output to the specified workstation. It must be called to terminate a program and should be the last graphics routine called.

v_curdown ()

Purpose	Cursor down
Syntax	v_curdown (dev_handle)
Data Types	INT16 v_curdown (); INT16 dev_handle;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Function Returns	Function v_curdown returns error state. 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	When you are in Cursor Addressing Mode, this routine moves the cursor down one row without altering its horizontal position. If the cursor is already at the bottom margin, the screen will scroll up one line. When the cursor is on the bottom line of the screen and characters are subsequently sent to the screen, any characters which exceed the right margin will wrap around to the left margin. Any characters already on the last line will be overwritten. This function is only applicable to CRT devices.

v_curhome ()

Purpose	Cursor home
Syntax	v_curhome (dev_handle)
Data Types	INT16 v_curhome (); INT16 dev_handle;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Function Returns	Function v_curhome returns error state. 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	When you are in Cursor Addressing Mode, this routine moves the cursor to the home position (upper left corner of the screen). It is only applicable to CRT devices.

v_curleft ()

Purpose	Cursor left
Syntax	v_curleft (dev_handle)
Data Types	INT16 v_curleft (); INT16 dev_handle;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Function Returns	Function v_curleft returns error state. 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	When you are in Cursor Addressing Mode, this routine moves the cursor left one column without altering its vertical position. No action occurs if the cursor is already at left margin. This function is only applicable to CRT devices.

v_currright ()

Purpose	Cursor right
Syntax	v_currright (dev_handle)
Data Types	INT16 v_currright (); INT16 dev_handle;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Function Returns	Function v_currright returns error state. 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	When you are in Cursor Addressing Mode, this routine moves the cursor right one column without altering its vertical position. No action occurs if the cursor is already at right margin. This function is only applicable to CRT devices.

v_curup ()

Purpose	Cursor up
Syntax	v_curup (dev_handle)
Data Types	INT16 v_curup (); INT16 dev_handle;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Function Returns	Function v_curup returns error state. 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	When you are in Cursor Addressing Mode, this routine moves the cursor up one row without altering its horizontal position. No action occurs if the cursor is already at the top margin. This function is only applicable to CRT devices.

v_dspcur ()

Purpose	Display graphics input cursor
Syntax	v_dspcur (dev_handle, x, y)
Data Types	INT16 v_dspcur (); INT16 dev_handle; INT16 x,y;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. x x-coordinate of location to place cursor. y y-coordinate of location to place cursor.
Function Returns	Function v_dspcur returns error state 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	When you have exited Cursor Addressing Mode, this routine displays a graphics input cursor centered about the specified location. The graphics cursor is the same as the one used for feedback by the Input Locator function (crosshairs, arrow, etc.). The Input Locator function will automatically display a cursor when needed. This function does not need to be referenced when using Input Locator. It is only applicable to CRT devices.

v_eeol ()

Purpose	Erase to end of line
Syntax	v_eeol (dev_handle)
Data Types	INT16 v_eeol (); INT16 dev_handle;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Function	Function v_eeol returns error state
Returns	0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	When you are in Cursor Addressing Mode, this routine erases from the current cursor position to the end of the line. It is only applicable to CRT devices.

v_eeos ()

Purpose	Erase to end of screen
Syntax	v_eeos (dev_handle)
Data Types	INT16 v_eeos (); INT16 dev_handle;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Function Returns	Function v_eeos returns error state 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	When you are in Cursor Addressing Mode, this routine erases from the current cursor to the end of the screen. It is only applicable to CRT devices.

v_enter_cur ()

Purpose	Enter cursor addressing mode
Syntax	v_enter_cur (dev_handle)
Data Types	INT16 v_enter_cur (); INT16 dev_handle;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Function Returns	Function v_enter_cur returns error state 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine exits Graphics Mode if different from Cursor Addressing Mode. It must precede all other cursor addressing functions, such as Cursor Up, etc. This routine "homes" the cursor. Cursor addressing is only meaningful on CRT devices. It is defined on a character cell grid of rows and columns (rows equals the number of lines on a screen, and columns equals the number of character cells per line). The upper left-hand corner of the screen is row 1, column 1.

v_exit_cur ()

Purpose	Exit cursor addressing mode
Syntax	v_exit_cur (dev_handle)
Data Types	INT16 v_exit_cur (); INT16 dev_handle;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Function Returns	Function v_exit_cur returns error state. 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine exits Cursor Addressing Mode if different from Graphics Mode. For applications to work properly, this function must be used to enter Graphics Mode from Cursor Addressing Mode.

v_hardcopy ()

Purpose	Hardcopy
Syntax	v_hardcopy (dev_handle)
Data Types	INT16 v_hardcopy (); INT16 dev_handle;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Function	Function v_hardcopy returns error state
Returns	0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine generates a hard copy. It is very device-specific and may involve copying the screen to a printer (doing a "screen dump") or other attached hard copy device.

v_msg ()

Purpose	Metafile message
Syntax	v_msg (dev_handle, message, wait)
Data Types	INT16 v_msg (); INT16 dev_handle; INT8 message[]; INT16 wait;
Input	dev_handle Metafile device handle. message Text string. wait Pause indicator. 0 = if no response required 1 = if pause after issuing message and wait for a response.
Function Returns	Function v_msg returns error state. 0 if no error -1 if error Actual error can be retrieved by invoking error inquiry function.
Description	This function places a text string in the metafile which will be displayed by the interpreter as an operator message. It appears on the console in a device-dependent position. The pause indicator controls whether the interpreter will pause for a (device-dependent) response or continue.

v_opnwk ()

Purpose	Open workstation
Syntax	v_opnwk (work_in, &dev_handle, work_out)
Data Types	INT16 v_opnwk (); INT16 work_in[19]; INT16 dev_handle; INT16 work_out[66];
Input	work_in[0] Coordinate transformation mode flag. Determines how to transform NDC space to device coordinate. 0 = Map NDC space to full extent of each axis. This mode does not preserve aspect ratio. Picture will completely fill screen. 1 = Map NDC space to full extent of longest axis only; map subset of NDC space to shorter axis. This mode preserves unity aspect ratio. Using this technique and the appropriate scaling factor results in a picture with the same aspect ratio. work_in[1] Polyline line type. work_in[2] Polyline color index. work_in[3] Polymarker type. work_in[4] Polymarker color index. work_in[5] Graphics text font. work_in[6] Graphics text color index. work_in[7] Fill interior style. work_in[8] Fill style index.

work_in[9]

Fill color index.

work_in[10]

Prompting flag for controlling screen prompts. Typical prompts are for paper and pen changes on plotters.

0 = Do not display device-dependent prompts to the logical message device

1 = Display device-dependent prompts to the logical message device

work_in[11-18]

Workstation identifier (device driver logical name). This is an ADE form that is used to determine which driver to dynamically bring into memory, but is not used by the driver itself. The following names should be used for logical device names unless they are superseded by an environment variable: DISPLAY, PLOTTER, PRINTER, METAFIL, CAMERA, JOYSTIK, MOUSE, GRAFOUT, GRAFIN.



Output

dev_handle

Device handle associated with the workstation identifier. (work_in_[11-18])

work_out[0]

Maximum addressable width of screen/plotter in rasters/steps assuming a 0 start point (e.g. a resolution of 640 implies an addressable area of 0-639, so work_out[0]=639)

work_out[1]

Maximum addressable height of screen/plotter in rasters/steps assuming a 0 start point (e.g. a resolution of 480 implies an addressable area of 0-479, so work_out[1]=479)

work_out[2]

Device coordinate units flag.

0 = Device capable of producing precisely scaled image (typically plotters and printers)

1 = Device not capable of precisely scaled image (CRTs)

work_out[3]

Width of one pixel (plotter step) in micrometers.

work_out[4]

Height of one pixel (plotter step) in micrometers.

work_out[5]

Number of character heights.

0 = continuous scaling

work_out[6]

Number of line types.

0 = device is not capable of graphics

work_out[7]

Number of line widths.

work_out[8]

Number of marker types.

work_out[9]

Number of marker sizes.

0 = continuous scaling

work_out[10]

Number of graphics text fonts.

work_out[11]

Number of patterns.

work_out[12]

Number of hatch styles.

work_out[13]

Number of predefined colors.

This is the number of colors that can be displayed on the device simultaneously when in Graphics Mode. The number of colors in Cursor Addressing Mode may be different.

Note

There must be at least two colors, even for monochrome devices.

work_out[14]

Number of Generalized Drawing Primitives (GDP).

work_out[15-24]

List of available GDPs. (Up to ten allowed)

- 1 = bar
- 2 = arc
- 3 = pie slice
- 4 = circle
- 1 = GDP does not exist

The list can be specified in any order.

work_out[25-34]

Attribute set associated with each GDP.

- 1 = GDP does not exist
- 0 = Polyline
- 1 = Polymarker
- 2 = Text
- 3 = Fill area
- 4 = None
- 5 = Other

work_out[35]

Color capability flag.

- 0 = no
- 1 = yes

work_out[36]

Text rotation capability flag.

- 0 = no
- 1 = yes

work_out[37]

Fill area capability flag.

- 0 = no
- 1 = yes

work_out[38]

Pixel operation capability flag

- 0 = no
- 1 = yes

work_out[39]

Number of available colors.

Total number of colors in color palette.

- 0 = Continuous device
- 2 = Monochrome (black and white)
- >2 = Number of colors available

work_out[40]

Locator capability flag.

0 = no

1 = yes

work_out[41]

Valuator capability flag.

0 = no

1 = yes

work_out[42]

Choice capability flag.

The value returned is the number of CHOICE indicators available. For example, if five function keys are used as CHOICE devices, the value returned would be 5.

work_out[43]

String input capability flag.

0 = no

1 = yes

work_out[44]

Workstation type.

0 = Output only

1 = Input only

2 = Input/Output

3 = Device-independent segment storage

4 = Metafile output

5 = Other

work_out[45]

Device type.

0 = CRT

1 = Plotter

2 = Printer

3 = Camera/film recorder

4 = Metafile output

5 = Other

work_out[46]

Number of writing modes available.

work_out[47]

Highest level of input mode available.

- 0 = none
- 1 = request
- 2 = sample

work_out[48]

Text alignment capability flag.

- 0 = no
- 1 = yes

work_out[49]

Inking capability flag as output echo device.

- 0 = no
- 1 = yes

work_out[50]

Rubberbanding capability flag as output echo device.

- 0 = No rubberband capability
- 1 = Capable of rubberband lines
- 2 = Capable of rubberband lines and rectangles

work_out[51]

Maximum addressable NDC space coordinate on x-axis. This value is filled in based on the coordinate transformation mode selected.

work_out[52]

Maximum addressable NDC space coordinate on y-axis. This value is filled in based on the coordinate transformation mode selected.

work_out[53-57]

Version of the driver.

This is an ADE character string that represents the version of the driver in the following form:

vv.ll
where vv is the actual version
and ll is the level.

work_out[58-59]

Reserved for future use.

work_out[60]

Minimum graphics character height in NDC units.

work_out[61]

Maximum graphics character height in NDC units.

work_out[62]

Minimum line width in NDC units.

work_out[63]

Maximum line width in NDC units.

work_out[64]

Minimum marker height in NDC units.

work_out[65]

Maximum marker height in NDC units.

Function	Function v_opnwk returns error state.
Returns	0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine initializes a workstation (a graphics device). It sets all defaults and returns device information. Both the alpha and graphics display surfaces are cleared by this function.

Color Indices

The following lists are the color index values for the Open Workstation routine.

Monochrome Devices Index

0 = Black for CRTs & White for printers/plotters

1 = White for CRTs & Black for printers/ plotters

Color Devices Index

0 = Black for CRTs & White for printers/ plotters

1 = White for CRTs & Black for printers/ plotters

2 = Red

3 = Green

4 = Blue

5 = Yellow

6 = Cyan

7 = Magenta

8-n = Device-dependent

Opening an Output Device

Each time **v_opnwk** is invoked to open an output device, the values specified in the input array, **work_in**, are implemented. In addition, the following default values take effect:

Graphics character height = largest size that allows 80 characters horizontally and 24 characters vertically

Character baseline rotation = 0 degrees rotation

Line width = 1 device unit (raster, plotter step)

Marker height = minimum marker height

Writing mode = 4 (replace)

Input mode = request for all input classes (locator, valuator, choice, string)

Text alignment = bottom for vertical alignment; left for horizontal alignment

Cursor addressing mode = off

Alpha text position = uppermost left corner of display surface to allow for one default alpha text character to appear but not be off the display surface

Alpha text line spacing = single spacing

Alpha text font = standard font that has 80 characters horizontally across the display surface

Alpha text subscripting and superscripting = off

Alpha text underlining = off

Alpha text overstriking = off

Alpha text pass through = off

Alpha text quality = high quality

Alpha text color index = 1

Line delete character = `^U` (NAK)

Char delete character = `^H` (Backspace)

Opening an Input Device

When Open Workstation (**v_opnwk**) is called to open an input device, the input array (**work_in**) is implemented as follows: **work_in[0]**, the coordinate transformation mode flag, specifies which aspect ratio to use. If 0 is chosen for this flag for both the input device and the output echo device, the full extents of the graphics input device will map to the full extents of the output echo device when echoing input. If 1 is chosen for the flag for either the input device or the output echo device, the full extents

of the two devices may not be the same. For example, the two devices could possibly overlap.

work_in[1] to work_in[10] are not applicable for an input device
work_in[11] to work_in[18] specify the logical name to be used to refer to the device.

The output array (work_out) for a typical input device would contain the following elements:

work_out[0], the addressable width of input device in rasters =
Device-dependent value

work_out[1], the addressable height of input device in rasters =
Device-dependent value

work_out[2], device coordinates in raster units flag = 1

work_out[3], width of a pixel in micrometers =
Device-dependent value

work_out[4], height of a pixel in micrometers =
Device-dependent value

work_out[5], number of character heights = 0

work_out[6], number of line types = 0

work_out[7], number of line widths = 0

work_out[8], number of marker types = 0

work_out[9], number of marker sizes = 0

work_out[10], number of graphics fonts = 0

work_out[11], number of patterns = 0

work_out[12], number of hatch styles = 0

work_out[13], number of predefined colors = 0

work_out[14], number of GDPs = 0

work_out[15] to work_out[34], no GDPs or associated bundle
tables available = -1

work_out[35], color capability flag = 0

work_out[36], text rotation capability flag = 0

work_out[37], fill area capability flag = 0

work_out[38], pixel operation capability flag = 0

work_out[39], number of available colors = 0

work_out[40], locator capability flag = 1

work_out[41], valuator capability flag = 0

work_out[42], number of choices devices available = 0

work_out[43], string input capability flag = 0

work_out[44], workstation type = 1

work_out[45], device type is other = 5

work_out[46], number of writing modes available = 0

work_out[47], highest level of input mode available—request = 1

work_out[48], text alignment capability flag = 0

work_out[49], inking capability flag = 0
 work_out[50], rubberbanding capability flag = 0
 work_out[51], max x in NDC = Device-dependent value
 work_out[52], max y in NDC = Device-dependent value
 work_out[53] to work_out[57], Version of driver = 2.00
 work_out[58], reserved = 0
 work_out[59], reserved = 0
 work_out[60], minimum character height in NDC space = 0
 work_out[61], maximum character height in NDC space = 0
 work_out[62], minimum line width in NDC space = 0
 work_out[63], maximum line width in NDC units = 0
 work_out[64], minimum marker height in NDC units = 0
 work_out[65], maximum marker height in NDC units = 0

Code Example—Open Workstation

The following program is a code example for the `v_opnwk` (Open Workstation) function:

```

static int16 nominate[] = {1, /* coordinate transformation mode,
    mode 0 is to map chart to entire display surface,
    mode 1 is to map 32767 to longest axis */
    1, /* line type to use, 1 indicates solid */
    1, /* line color index to use */
    3, /* marker type, 3 is a star */
    1, /* marker color index */
    1, /* text font to use */
    1, /* text color */
    0, /* fill interior style, 0 indicates hollow */
    0, /* fill style index */
    1, /* fill color index */
    1, /* device messages to screen */
    'D','I','S','P','L','A','Y', /* device name to use */
}
int16 device_info[66], device_handle, err_report, xscale, yscale, scale;
err_report = v_opnwk (nominate, &device_handle, device_info);
/* Using the aspect ratio returned from the GSS VDI,
    determine the scaling in the x direction */
xscale = device_info[51] / 300;
/* Using the aspect ratio returned from the GSS VDI, determine
    the scaling in the y direction */
yscale = device_info[52] / 200;
/* determine the smaller of the two scaling factors and use it
    for transformation from user units (300 cm by 200 cm) to
    normalized device coordinates */
scale = (xscale > yscale) ? yscale : xscale;
  
```

v_rmcure ()

Purpose	Remove graphics input cursor
Syntax	v_rmcure (dev_handle)
Data Types	INT16 v_rmcure (); INT16 dev_handle;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Function	Function v_rmcure returns error state.
Returns	0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine removes the graphics input cursor from its current location on the screen. It does not need to be referenced for performing Locator Input.

v_updwk ()

Purpose	Update workstation
Syntax	v_updwk (dev_handle)
Data Types	INT16 v_updwk (); INT16 dev_handle;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Function Returns	Function v_updwk returns error state. 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine displays all pending graphics on the workstation. For printers, this causes the current picture to be output and the printer to advance to top-of-form.

vs_curaddress ()

Purpose	Direct cursor address
Syntax	vs_curaddress (dev_handle, row, column)
Data Types	INT16 vs_curaddress (); INT16 dev_handle ; INT16 row ; INT16 column ;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. row Row number. 1 to number of rows column Column number. 1 to number of columns
Function Returns	Function vs_curaddress returns error state. 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	When you are in Cursor Addressing Mode, this routine moves the cursor to a specified position. The position is specified in cursor space with row 1 column 1 being the top left corner of the screen. If you specify a position off the screen, the cursor will not move. It is only applicable to CRT devices.

vs_penspeed ()

Purpose	Set pen speed
Syntax	vs_penspeed (dev_handle, speed)
Data Types	INT16 vs_penspeed (); INT16 dev_handle; INT16 speed;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. speed Pen speed as percentage of maximum speed. 0 to 100
Function Returns	Function vs_penspeed returns ≥0 if pen speed -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine sets the plotter pen speed to a percentage of the maximum speed between 0 and 100; it only affects plotter devices. This call can be used to slow down a plotter when using nonstandard inks or media.

vsa_position ()

Purpose	Set alpha text position
Syntax	vsa_position (dev_handle, x_in, y_in, &x_out, &y_out)
Data Types	INT16 vsa_position (); INT16 dev_handle; INT16 x_in; INT16 y_in; INT16 x_out; INT16 y_out;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. x_in x-coordinate of text position in NDC units. y_in y-coordinate of text position in NDC units.
Output	x_out x-coordinate of text position selected in NDC units. y_out y-coordinate of text position selected in NDC units.
Function Returns	Function vsa_position returns 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	When you have exited Cursor Addressing Mode, this routine sets the alpha text position to the location specified. This specifies the position of the lower left-hand corner of the alpha text string. It is assumed that (0,0) is at the lower left-hand corner of the display surface. The alpha position is updated only when the position is set or when the Output Alpha Text function is invoked. If the position is set at the maximum x or y extent, display of alpha text is device-dependent since characters positioned at that point would be off the display surface.

vswr_mode ()

Purpose	Set writing mode
Syntax	vswr_mode (dev_handle, mode_in)
Data Types	INT16 vswr_mode (); INT16 dev_handle; INT16 mode_in;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. mode_in Writing mode requested. Key: s = color index of source pixel d = color index of destination pixel 1-d = 0 (all bits of color index off) 2-d = d and s 3-d = (not d) and s 4-d = s 5-d = d and (not s) 6-d = d 7-d = d xor s 8-d = d or s 9-d = not (d or s) 10-d = not (d xor s) 11-d = not d 12-d = (not d) or s 13-d = not s (reverse video) 14-d = d or (not s) 15-d = not (d and s) 16-d = all bits of color index on
Function Returns	Function vswr_mode returns ≥ 0 if mode selected -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine sets the current writing mode. It specifies the Boolean operation that is performed between the color indices of the source and destination pixels when lines, text, filled areas, etc. are placed on the display. Mode 4 "replace"

is the default for printers and screen devices.
Mode 8, "overstrike" is the default for plotters.

This function refers only to Graphics Mode, and does not affect cursor text.

Writing modes will work on the printers in a manner similar to screen device operation. If two objects are displayed on top of each other, the result will be determined by the current writing mode.

Output Functions

This section covers the output functions that allow you to describe objects in Normalized Device Coordinates. They provide the capabilities to:

- place polymarkers
- display alpha, cursor and graphics text
- fill a defined area
- output a cell array
- draw special geometric primitives such as bars, circles, and arcs.

Primitive Functions

The graphical world that the programmer describes to the system consists of one or more objects. Each graphical object is described, in turn, by output primitives which have specific attributes.

Polyline Function

The polyline is the fundamental line drawing primitive and generates a set of connected lines given an array of points as a parameter. Since polyline is the basic primitive, attributes such as line color or line style apply to the complete polyline rather than to a segment.

Polymarker Function

Markers are special symbols that can be placed on the display surface to provide a method for identifying two-dimensional positions on the output display surface. The basic primitive is a polymarker that outputs a sequence of markers, each centered on a specified position.

Text Function

The text primitives display a string of either alpha, cursor or graphics text characters at a specified location.

Fill Area Function

This primitive defines the boundary of a polyline that can be hollow, filled in solidly or filled with either a pixel pattern or a hatching pattern.

Cell Array Function

The cell array function provides a means of specifying an array of colors or intensities. This is particularly useful in image processing applications.

Generalized Drawing Primitives

GSS-DRIVERS also supports special geometric primitives.

Bars Bars are a special type of filled area defined by opposite corners, so filled area attributes affect their appearance. Bars can be displayed with control over the interior style (HOLLOW, SOLID, PATTERN, HATCH), fill style (type of pattern or hatch) and color.

- Circles** Circles are also a special type of filled area defined by a center point and radius, so filled area attributes affect their appearance. They too can be displayed with control over the interior style and fill style.
- Pie Slices** Pie slices are a special type of filled area defined by a center point and the two points of the arc. Like bars and circles, they are affected by filled area attributes, including interior and fill style.
- Arcs** Arcs are a special type of polyline defined by a center point and the two end points of the arc. They are affected by the polyline attributes of color, line style and line width.

TABLE 5-2 Output Functions

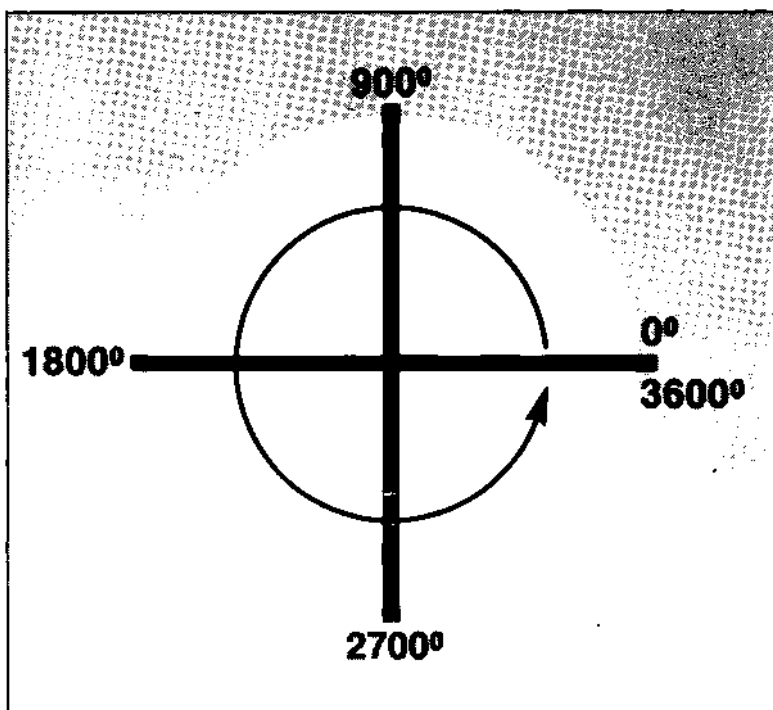
<u>Function</u>	<u>Routine</u>	<u>Description</u>	<u>Page</u>
Polyline	v_pline	Output Polyline	5-56
Polymarker	v_pmarker	Output Polymarker	5-58
Alpha Text	v_atext	Output Alpha Text	5-42
Cursor Addressable Text	v_curtext	Output Cursor	5-50
Graphics Text	v_gtext	Output Graphics Text	5-53
Fill Area	v_fillarea	Output Filled Area	5-51
Cell Array	v_cellarray	Output Cell Array	5-45
Generalized Drawing Primitives	v_arc v_circle v_bar v_pieslice	Output Arc Output Circle Output Bar Output Pie Slice	5-40 5-49 5-44 5-54

v_arc ()

Purpose	Output arc
Syntax	v_arc (dev_handle, x, y, radius, begang, endang)
Data Types	INT16 v_arc(); INT16 dev_handle; INT16 x, y; INT16 radius; INT16 begang; INT16 endang;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. x x-coordinate of center point of arc. y y-coordinate of center point of arc. radius Radius. begang Start angle in tenths of degrees. 0 to 3600 endang End angle in tenths of degrees. 0 to 3600
Function Returns	Function v_arc returns error state. 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine draws arcs using current line attributes. Arcs are defined by the center point and two end points of the arc. The radius is assumed to be measured along the x-(horizontal) axis. All angle specifications assume that 0 degrees is 90 degrees to the right of vertical (or 3:00), with values increasing in the counterclockwise direction.

Arcs and pie slices are always drawn in a counterclockwise direction. The start angle does not need to be larger than the ending angle. If a start angle of 40 degrees and an ending angle of 15 degrees are given, an arc would be drawn counter-clockwise from the 40 degree angle to the 15 degree angle.

FIGURE 5-1 Angle Specification



v_atext ()

Purpose	Output alpha text
Syntax	v_atext (dev_handle, string, &x_out, &y_out)
Data Types	INT16 v_atext (); INT16 dev_handle; INT8 string[]; INT16 x_out; INT16 y_out;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. string Text string. Passed as a contiguous stream of bytes.
Output	x_out x-coordinate of text position (in NDC units) after the text string has been output. This is the same value that is returned if Inquire Alpha Text Position were invoked. y_out y-coordinate of text position (in NDC units) after the text string has been output. This is the same value that is returned if Inquire Alpha Text Position were invoked.
Function Returns	Function v_atext returns error state. 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine outputs text at the current alpha text position, honoring all current alpha text attributes (subscripting, under-line, alpha text font, etc.), and the cursor is positioned at the end of the string. The alpha text position is updated appropriately after outputting the text string. Receipt of the ASCII character <CR> (carriage return) causes the alpha text position to be set to the beginning of the line (x = 0). Receipt of a LF

(line feed) causes the alpha text position to be advanced by the current line spacing ($y = y\text{-line spacing}$). All other control characters/nonprintable characters (ASCII characters 0-31) are not output. Attempting to display characters past the X or Y maximum of the display surface produces device-dependent results.

Alpha text is useful for outputting word processor quality text display on CRTs, printers, etc., and is displayed to the best resolution and accuracy of the hardware.

Note

In the C language the `'\n'` character is interpreted as a line feed character. Use `'\n\r'` to get a line feed, carriage return combination.

v_bar ()

Purpose	Output bar
Syntax	v_bar (dev_handle, xy)
Data Types	INT16 v_bar (); INT16 dev_handle; INT16 xy[4];
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. xy[0] x-coordinate of lower left-hand corner of bar xy[1] y-coordinate of lower left-hand corner of bar xy[2] x-coordinate of upper right-hand corner of bar xy[3] y-coordinate of upper right-hand corner of bar
Function Returns	Function v_bar returns error state. 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine draws a rectangular area using current filled area attributes of interior style, fill style and color.

v_cellarray ()

Purpose Output cell array

Syntax `v_cellarray (dev_handle, xy, row_length, el_per_row,
num_rows, wr_mode, colors)`

Data Types `INT16 v_cellarray ();
INT16 dev_handle;
INT16 xy[4];
INT16 row_length;
INT16 el_per_row;
INT16 num_rows;
INT16 wr_mode;
INT16 colors[];`

Input **dev_handle**
Device handle returned from `v_opnwk`.
Refers to a specific graphics device when multiple workstations are open.

xy[0]
x-coordinate of lower left-hand corner (NDC units)

xy[1]
y-coordinate of lower left-hand corner (NDC units)

xy[2]
x-coordinate of upper right-hand corner (NDC units)

xy[3]
y-coordinate of upper right-hand corner (NDC units)

row_length
Length of each row in color index array

el_per_row
Number of elements used in each row of color index array.

num_rows
Number of rows in color index array.

wr_mode
Pixel operation to be performed.
See Set Writing Mode function for list of operations.

colors
Color index array (stored one row at time).

Function	Function v_cellarray returns error state.
Returns	0 if no error -1 if error Actual error can be retrieved by invoking vq_error.
Description	This routine outputs pixels to the device and can be used for imaging. Each row in the color index array is expanded evenly to fill the entire width of the rectangle specified by pixel replication. Each row in the color index array is also replicated the appropriate number of times to fill the entire height of the rectangular area evenly with the color pattern, starting from the top of the rectangular area and filling downward. For example, if there are two rows and three elements per row, the vertical dimension of the rectangle is divided into two equal parts, and the horizontal dimension is divided into three equal parts. The rectangle is filled in the upper left-hand corner of the area with the color index specified in the first element of the color index array, etc. If the device can't do cell arrays, the area is outlined in the current line color. The input array to the cell array function is a list of color indices. Each of these indices indicates the color to use to fill one of the cells of the cell array. The "XY" array defines the corners of the rectangle. "Elements per row" and "number of rows" define how many sections (cells) to divide the rectangle into. For example, to create the cell array shown in Figure 5-2, the following values would be entered: <pre> xy(0) = 24 xy(1) = 300 xy(2) = 50 xy(3) = 500 row_length = 3 el_per_row = 3 num_rows = 2 wr_mode = 4 </pre>


```
colors(0) = 2
colors(1) = 4
colors(2) = 5
colors(3) = 3
colors(4) = 1
colors(5) = 4
```

We could also have used the following set of values:

```
xy(0) = 24
xy(1) = 300
xy(2) = 50
xy(3) = 500
row_length = 5
el_per_row = 3
num_rows = 2
wr_mode = 4
colors(0) = 2
colors(1) = 4
colors(2) = 5
colors(3) = data value not used
colors(4) = data value not used
colors(5) = 3
colors(6) = 1
colors(7) = 4
colors(8) = data value not used
colors(9) = data value not used
```

Note in the second example, that the “row length” is set to 5 and the colors array has been increased. Since the “elements per row” value is 3, we only use the first three of the five elements per row. This mechanism is useful when the user only wants to use a portion of a large array of data.

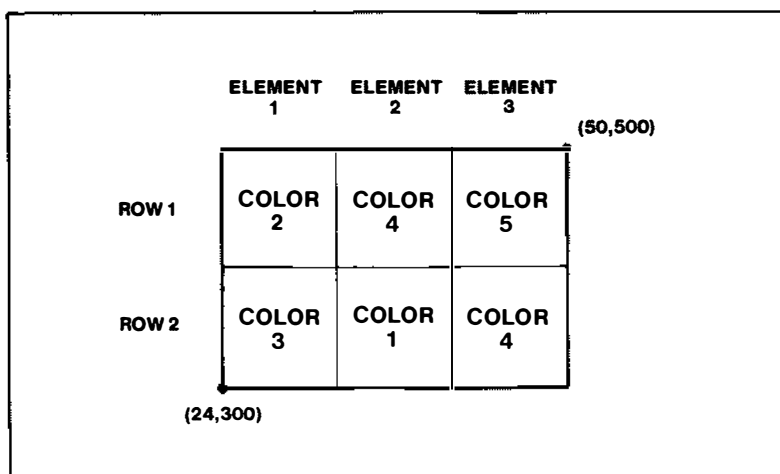
The “colors” values correspond to entries from the currently defined color table.

In this example, we used a writing mode value of 4. However, any of the sixteen writing mode values could have been used.

The term “pixel” is synonymous with the words “raster” and “pel”. The current standards documents use the word “pixel”. Each “pixel” within a cell is displayed with the indicated color.

By specifying an array that is 320×200 pixels long and indicating the colors of the display, the user could set the color of each individual pixel on the screen using the current color graphics card.

FIGURE 5-2 Cell Array



v_circle ()

Purpose	Output circle
Syntax	v_circle (dev_handle, x, y, radius)
Data Types	INT16 v_circle (); INT16 dev_handle; INT16 x; INT16 y; INT16 radius;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. x x-coordinate of center point of circle y y-coordinate of center point of circle radius Radius
Function Returns	Function v_circle returns error state. 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine draws a circle, specified by a center point and filled radius. The radius is assumed to be measured along the x-axis (horizontal). Since circles are a special type of filled area, they are affected by filled area attributes including interior style, fill style and color.

v_curtext ()

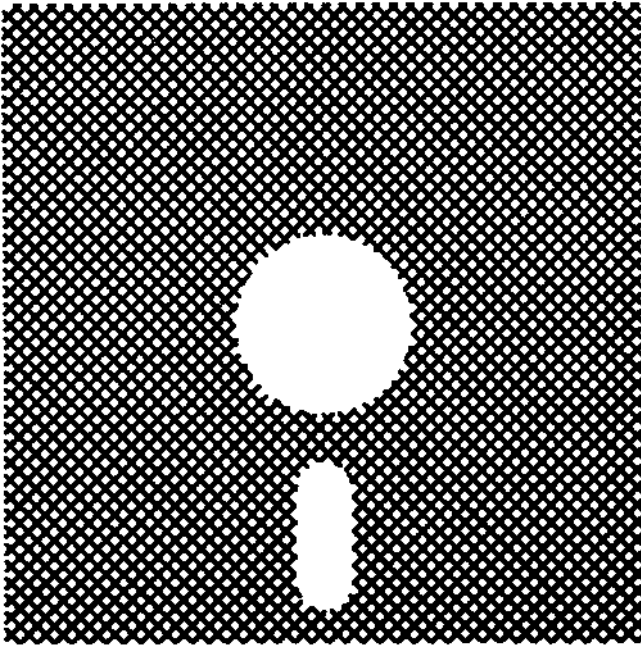
Purpose	Output cursor addressable text
Syntax	v_curtext (dev_handle, string)
Data Types	INT16 v_curtext (); INT16 dev_handle; INT8 string[];
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. string Text string. Passed as a contiguous stream of bytes.
Function Returns	Function v_curtext returns error state. 0 if no error -1 if error Actual error can be retrieved by invoking vq_error . You must be in Cursor Addressing Mode to output cursor addressable text.
Description	For CRTs, this routine outputs text at the current cursor position, honoring all the cursor text attributes (such as color, reverse video, etc.). Note that new text replaces (overwrites) old text at the same location. Only text that will fit onto the line of the cursor will be displayed, and it will not wrap onto the next line at the left edge.

v_fillarea ()

Purpose	Output filled area
Syntax	v_fillarea (dev_handle, count, xy)
Data Types	INT16 v_fillarea (); INT16 dev_handle; INT16 count; INT16 xy[2n];
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. count Number of vertices in polyline. xy Array of coordinates of polyline (NDC units) xy[0] x-coordinate of first point xy[1] y-coordinate of first point xy[2] x-coordinate of second point xy[3] y-coordinate of second point . . . xy[2n-2] x-coordinate of last point xy[2n-1] y-coordinate of last point
Function Returns	Function v_fillarea returns error state. 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .

Description This routine outputs a filled area to the device using current fill area attributes of fill color index, fill interior style index, fill style index. **HOLLOW** filled areas are outlined with a solid border using the current fill color. **SOLID**, **HATCH**, and **PATTERN** filled areas are not outlined.

FIGURE 5-3 Filled Area



v_gtext ()

Purpose	Output graphics text
Syntax	v_gtext (dev_handle, x, y, string)
Data Types	INT16 v_gtext (); INT16 dev_handle; INT16 x,y; INT8 string[];
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. x x-coordinate of alignment point of text (NDC units) y y-coordinate of alignment point of text (NDC units) string Text string. Passed as a pointer to characters or array. Characters must be printable characters, ASCII <space> and above.
Function Returns	Function v_gtext returns error state. 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine outputs graphics text with current attributes to the device. The X, Y location is the position the text will be aligned to, taking into account the current graphics text alignment values. Any text that is outside of the display surface is not displayed.

v_pieslice ()

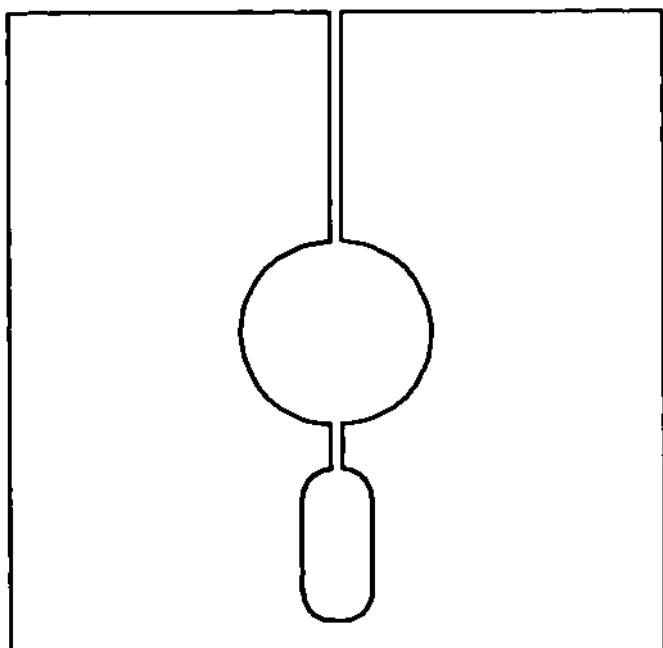
Purpose	Output pie slice
Syntax	v_pieslice (dev_handle, x, y, radius, begang, endang)
Data Types	INT16 v_pieslice (); INT16 dev_handle; INT16 x; INT16 y; INT16 radius; INT16 begang; INT16 endang;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. x x-coordinate of center point of arc (NDC units) y y-coordinate of center point of arc (NDC units) radius Radius (NDC units) begang Start angle in tenths of degrees 0 to 3600 endang End angle in tenths of degrees 0 to 3600
Function Returns	Function v_pieslice returns error state. 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine draws pie slices by specifying the center point and two points on the arc. The radius is assumed to be measured along the x- (horizontal) axis. Since pie slices are a special type of filled area, they are affected by filled area attributes including interior style, fill style and color.

All angle specifications assume that 0 degrees is 90 degrees to the right of vertical, with values increasing in the counter-clockwise direction. (See Figure 5-1.)

v_pline ()

Purpose	Output polyline
Syntax	v_pline (dev_handle, count, xy)
Data Types	INT16 v_pline (); INT16 dev_handle; INT16 count; INT16 xy[2n];
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. count Number of vertices (x,y pairs) in polyline xy Array of coordinates of polyline (NDC units) xy[0] x-coordinate of first point xy[1] y-coordinate of first point xy[2] x-coordinate of second point xy[3] y-coordinate of second point xy[2n-2] x-coordinate of last point xy[2n-1] y-coordinate of last point
Function Returns	Function v_pline returns error state. 0 if no error -1 if error Actual error can be retrieved by invoking vg_error .
Description	This routine outputs a polyline (with the current polyline attributes of line style, width and color) to the device. It moves to the first point and draws a line between subsequent points.

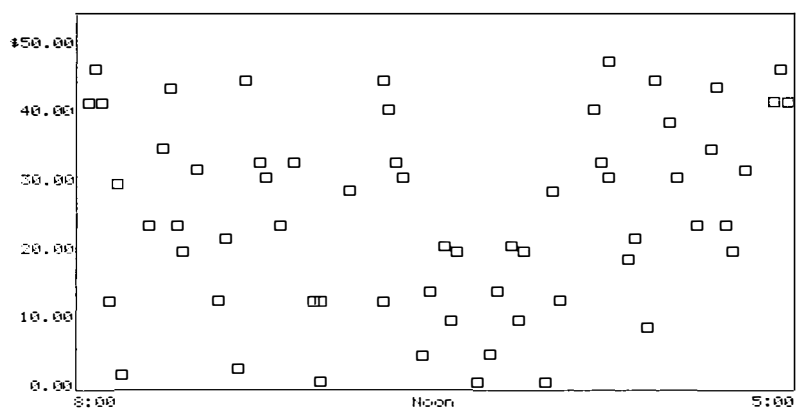
FIGURE 5-4 Polyline



v_pmarker ()

Purpose	Output polymarker
Syntax	v_pmarker (dev_handle, count, xy)
Data Types	INT16 v_pmarker (); INT16 dev_handle; INT16 count; INT16 xy[2n];
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. count Number of markers. xy Array of coordinates (NDC units) xy[0] x-coordinate of first marker. xy[1] y-coordinate of first marker. xy[2] x-coordinate of second marker. xy[3] y-coordinate of second marker. xy[2n-2] x-coordinate of last marker. xy[2n-1] y-coordinate of last marker.
Function Returns	Function v_pmarker returns error state. 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine outputs markers (with the current polymarker attributes of scale, type and color) to the device. At least six marker types are provided as specified in the Set Polymarker Type routine.

FIGURE 5-5 Polymarkers





Attribute Functions

This section discusses primitive attributes, the general characteristics of a display primitive assigned at the time it is defined by the application program. You can interrogate current attribute values and change them at any time after GSS-DRIVERS is initiated.

Attribute Functions

The purpose of the attribute functions is to specify general characteristics for output primitives. Some attributes control the geometric aspects of the primitives. These are aspects that affect the shape or size of a primitive (for example, character height). Geometric attributes are expressed in NDC units. A second type of attribute controls the non-geometric aspects of primitives; these modify the appearance of primitives without changing their shape (for example, line style).

TABLE 5-3 Attribute Functions

<u>Function</u>	<u>Routine</u>	<u>Description</u>	<u>Page</u>
Polyline Attributes	vsl_color	Set Polyline Color Index	5-86
	vsl_type	Set Polyline Line Type	5-87
	vsl_width	Set Polyline Line Width	5-88
Polymarker Attributes	vsm_color	Set Polymarker Color Index	5-89
	vsm_height	Set Polymarker Height	5-90
	vsm_type	Set Polymarker Type	5-91
Alpha Text Attributes	vsa_color	Set Alpha Text Color Index	5-72
	vsa_font	Set Alpha Text Font and Size	5-73
	vsa_spacing	Set Alpha Text Line Spacing	5-79
	vsa_overstrike	Set Alpha Text Overstrike Mode	5-76
	vsa_passthru	Set Alpha Text Pass Through Mode	5-77
	vsa_quality	Set Alpha Text Quality	5-78
	vsa_supersub	Set Alpha Text Subscript/Superscript Mode	5-80
	vsa_underline	Set Alpha Text Underline Mode	5-81
Cursor Text Attributes	vcur_color	Set Cursor Text Color Index	5-68
	vcur_att	Set Cursor Text Attributes	5-66
	v_rvoff	Reverse Video Off	5-64
	v_rvon	Reverse Video On	5-65

(continued)

TABLE 5-3 (continued)

Function	Routine	Description	Page
Graphics Text Attributes	vst_height	Set Character Height	5-96
	vst_alignment	Set Graphics Text Alignment	5-92
	vst_rotation	Set Graphics Text String Baseline Rotation	5-98
	vst_color	Set Graphics Text Color Index	5-94
	vst_font	Set Graphics Text Font	5-95
Color Attributes	vsb_color	Set Background Color Index	5-82
	vs_color	Set Color Representation	5-69
Fill Attributes	vsf_color	Set Fill Color Index	5-83
	vsf_interior	Set Fill Interior Style	5-84
	vsf_style	Set Fill Style Index	5-85
Input Attributes	vs_editchars	Set Line Edit Characters	5-71

v_rvoff ()

Purpose	Reverse video off
Syntax	v_rvoff (dev_handle)
Data Types	INT16 v_rvoff (); INT16 dev_handle;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Function Returns	Function v_rvoff returns error state. 0 if no error -1 if error Actual error can be retrieved by invoking vg_error .
Description	When you are in Cursor Addressing Mode, this routine displays subsequent cursor addressable text in standard video. It is only applicable to CRT devices.

v_rvon ()

Purpose	Reverse video on
Syntax	v_rvon (dev_handle)
Data Types	INT16 v_rvon (); INT16 dev_handle;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Function Returns	Function v_rvon returns error state. 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	When you are in Cursor Addressing Mode, this routine displays subsequent cursor addressable text in reverse video. It is only applicable to CRT devices.

vcur_att ()

Purpose	Set cursor text attributes
Syntax	vcur_att (dev_handle, req_att, sel_att)
Data Types	INT16 vcur_att (); INT16 dev_handle; INT16 req_att[4]; INT16 sel_att[4];
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. req_att[0] Requested Reverse Video Mode. 0 = Disable Reverse Video 1 = Enable Reverse Video 2 = Do not change current state 3 = Toggle Reverse Video Status req_att[1] Requested Underline Cursor Text Mode. 0 = Disable Underline Cursor Text 1 = Enable Underline Cursor Text 2 = Do not change current state 3 = Toggle Underline Cursor Text Status req_att[2] Requested Blink Text Mode. 0 = Disable Blink Cursor Text 1 = Enable Blink Cursor Text 2 = Do not change current state 3 = Toggle Blink Cursor Text Status req_att[3] Requested Bold Cursor Text Mode. 0 = Disable Bold Cursor Text 1 = Enable Bold Cursor Text 2 = Do not change current state 3 = Toggle Bold Cursor Text Status

Output

sel_att[0]

Selected Reverse Video Mode.

0 = disabled

1 = enabled

sel_att[1]

Selected Underline Cursor Text Mode.

0 = disabled

1 = enabled

sel_att[2]

Selected Blink Text Mode.

0 = disabled

1 = enabled

sel_att[3]

Selected Bold Cursor Text Mode.

0 = disabled

1 = enabled

Function

Function **vcur_att** returns error state

Returns

0 if no error

-1 if error

Actual error can be retrieved by invoking **vq_error**.

Description

This routine allows the attributes of Reverse Video, Underline, Blink, and Bold to be set for subsequent cursor addressable text. The Reverse Video Mode can be set in this routine or in the Reverse Video On and Reverse Video Off functions. Whichever was called last will be used as the reverse video attribute for subsequent cursor addressable text.

This function can be used to do an inquiry of the current status by setting all modes to DO NOT CHANGE STATUS. The current state would then be returned in **sel_att**.

vcur_color ()

Purpose	Set cursor text color index
Syntax	vcur_color (dev_handle, fore_requested, back_requested, &fore_selected, &back_selected)
Data Types	INT16 vcur_color (); INT16 dev_handle; INT16 fore_requested; INT16 back_requested; INT16 fore_selected; INT16 back_selected;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. fore_requested Color index of the foreground of subsequent output cursor text. default = 1 back_requested. Color index of the background of subsequent output cursor text. default = 0
Output	fore_selected Color index selected for cursor text foreground. back_selected Color index selected for cursor text background.
Function Returns	Function vcur_color returns error state 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine sets the foreground and background colors for cursor addressable text. If an invalid color index is specified, it is mapped to the index closest to the current device's capabilities.

vs_color ()

Purpose	Set color representation
Syntax	vs_color (dev_handle, ind_in, rgb_in, rgb_out)
Data Types	INT16 vs_color (); INT16 dev_handle; INT16 ind_in; INT16 rgb_in[3]; INT16 rgb_out[3];
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. ind_in Color index requested. rgb[0] Red color intensity. 0 to 1000 (tenths of percent) rgb[1] Green color intensity. 0 to 1000 (tenths of percent) rgb[2] Blue color intensity. 0 to 1000 (tenths of percent)
Output	rgb_out[0] Red color intensity selected. 0 to 1000 (tenths of percent) rgb_out[1] Green color intensity selected. 0 to 1000 (tenths of percent) rgb_out[2] Blue color intensity selected. 0 to 1000 (tenths of percent)
Function Returns	Function vs_color returns ≥0 if index selected -1 if error Actual error can be retrieved by invoking vq_error .

Description This routine is used to map indices to new colors. At least two color indices are provided. If the desired index is outside of the device's capabilities, then the closest device index is set. If a color intensity of less than 0 is requested, it is mapped to 0. If a color intensity greater than 1000 is requested, it is mapped to 1000.

To change the appearance of a color, you must select the desired levels of the three color components (red, green and blue) that make up the index. This can be used to create nondefault colors such as brown or orange. The new color will only be visible on devices that support color definition.

If the value of the function `vs_color` is negative then an error occurred. If the function returns a value greater than or equal to zero, then it is equal to the color index selected, and no error occurred.

vs_editchars ()

Purpose	Set line edit characters
Syntax	vs_editchars (dev_handle , line_del , char_del)
Data Types	INT16 vs_editchars (); INT16 dev_handle ; INT8 line_del ; INT8 char_del ;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. line_del Character to use to delete previous line. Default = ^U (NAK) char_del Character to use to delete previous character. Default = ^H (Backspace)
Function Returns	Function vs_editchars returns error state 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine sets the current line editing characters. They are applied to the Input String function only.

vsa_color ()

Purpose	Set alpha text color index
Syntax	vsa_color (dev_handle, ind_in)
Data Types	INT16 vsa_color (); INT16 dev_handle; INT16 ind_in;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. ind_in Requested text color index. 1 to device maximum
Function Returns	Function vsa_color value returned is: ≥0 if color index selected -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine selects the alpha text color index for subsequent output. At least two color indices are provided. Color indices range from 0 to a device-dependent maximum. If the color index requested is not valid, the closest value within the range of the current device's capabilities is selected.

vsa_font ()

Purpose Set alpha text font and size

Syntax **vsa_font (dev_handle, font_in, size_in, font_cap)**

Data Types **INT16 vsa_font ();**
INT16 dev_handle;
INT16 font_in;
INT16 size_in;
INT16 font_cap[8];

Input **dev_handle**
Device handle returned from **v_opnwk**.
Refers to a specific graphics device when multiple workstations are open.

font_in
Requested font.
If requested font is not available, font 1 (standard font) is used. Fonts 1 to 3 are fixed-space fonts.

- 1 = Normal/standard font (default)
- 2 = Bold (always provided for printers)
- 3 = Italics
- 4 = Proportionally spaced normal font
- 5 = Proportionally spaced bold
- 6 = Proportionally spaced italics
- >6 = Device-dependent

size_in
Requested text size.
1 to device maximum
where size n+ 1 is larger (occupies more area)
than size n.

Output **font_cap[0]**
Text size index selected.

font_cap[1]
Number of horizontal character cell positions across the display surface in this font. This is -1 if the font selected is a proportional font, since the character cell size is not constant.

font_cap[2]
Number of vertical character cell positions down the display surface in this font.

font_cap[3]

Number of horizontal character cell positions represented by the distance specified in font_cap[6]. The ratio ($\text{font_cap}[6] / \text{font_cap}[3]$) can be used to determine the width of a character cell, including any roundoff error.

font_cap[4]

Number of vertical character cell positions represented by the distance specified in font_cap[7]. The ratio ($\text{font_cap}[7] / \text{font_cap}[4]$) can be used to determine the height of a character cell, including any roundoff error.

font_cap[5]

Proportional spacing flag.

0 = No

1 = Yes

If this value is 1, then the size represented by font_cap[6] and font_cap[7] may not represent the selected font. This is the case if the desired font is proportionally spaced.

font_cap[6]

Width in NDC units of the number of character cells in the selected font specified in font_cap[3]. The ratio ($\text{font_cap}[6] / \text{font_cap}[3]$) can be used to determine the width of a character cell, including any roundoff error. This value is not accurate if the proportional spacing flag is set, since the character cell size is not constant.

font_cap[7]

Height in NDC units of the number of character cells in the selected font specified in font_cap[5]. The ratio ($\text{font_cap}[7] / \text{font_cap}[4]$) can be used to determine the height of a character cell including any roundoff error.

Function
Returns

Function **vsa_font** returns

≥0 if font selected

-1 if error

Actual error can be retrieved by invoking **vq_error**.

Description This routine sets the hardware alpha text font and size for subsequent output alpha text functions.

On printers, the resident font capability is used. Unlike graphics text, alpha text capabilities do not include font emulation.

vsa_overstrike ()

Purpose	Set alpha text overstrike mode
Syntax	vsa_overstrike (dev_handle, mode_in)
Data Types	INT16 vsa_overstrike (); INT16 dev_handle; INT16 mode_in;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. mode_in Overstrike mode requested. 0 = off (default) 1 = on If an invalid mode is requested, the default is selected.
Function Returns	Function vsa_overstrike returns ≥0 if mode selected -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine turns overstriking on or off. The default is overstriking off. When this mode is on, the alpha text position is not automatically advanced after each character is output; however, carriage return and line feed can still modify the current alpha text position.

vsa_passthru ()

Purpose	Set alpha text pass through mode
Syntax	vsa_passthru (dev_handle, mode_in)
Data Types	INT16 vsa_passthru (); INT16 dev_handle; INT16 mode_in;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. mode_in Pass through mode requested. 0 = off (default) 1 = on If an in valid mode is requested, the default is selected.
Function Returns	Function vsa_passthru returns ≥0 if mode selected -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine turns pass through mode on or off. Pass through mode enables all text to be output. Attributes such as font, color, superscripting, etc. may not be honored. When this mode is in effect, any text displayed does not modify the alpha text position. All characters, including control characters are sent directly to the device. This routine may be used to send device-dependent set-up strings to a particular device. The default is pass through OFF.

vsa_quality ()

Purpose	Set alpha text quality
Syntax	vsa_quality (dev_handle, mode_in)
Data Types	INT16 vsa_quality (); INT16 dev_handle; INT16 mode_in;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. mode_in Text quality requested. 0 to 100 where 0 = Lowest (draft) quality 100 = Highest quality (default) If an invalid mode is requested, the default (high quality) is selected. The number of quality levels is device-dependent.
Function Returns	Function vsa_quality returns ≥0 if mode selected -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine sets the alpha text quality level to some level between draft quality and high quality. The default is high quality. In draft quality range, small imperfections due to bidirectional printing or print head speed are acceptable. In high quality range the output is the best possible. Alpha text quality is a device-dependent attribute normally associated with printers. CRTs often may have multiple alpha fonts, but do not differentiate by quality levels. On dot matrix printers, quality usually affects the number of dots used to display a character.

vsa_spacing ()

 Purpose	Set alpha text line spacing
Syntax	vsa_spacing (dev_handle, spac_in)
Data Types	INT16 vsa_spacing (); INT16 dev_handle; INT16 space_in;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. spac_in Line spacing requested. positive value in NDC units
Function Returns	Function vsa_spacing returns ≥0 if spacing selected -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine sets the vertical spacing between lines of alpha text. It determines the amount of movement down the page when it receives a line feed control character in an output alpha text string. The default is single spacing; that is, the amount of spacing between lines of alpha text is the same as the default character cell height. Line spacing must always be a positive value. It specifies a decrement in the absolute vertical position when a line feed is encountered. You will need to update the line spacing to the character cell height of the new font (or whatever is desired) whenever fonts are changed.

vsa_supersub ()

Purpose	Set alpha text subscript/superscript mode
Syntax	vsa_supersub (dev_handle, mode_in)
Data Types	INT16 vsa_supersub (); INT16 dev_handle; INT16 mode_in;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. mode_in Mode requested. 0 = Subscripting and superscripting off (default) 1 = Subscripting on 2 = Superscripting on If an invalid mode is requested, mode = 0 is selected.
Function Returns	Function vsa_supersub returns ≥0 if mode selected -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine sets subscripting or superscripting for subsequent alpha text. It causes output to be offset above or below the line and is useful for footnotes, etc. The default is subscripting and superscripting OFF.

vsa_underline ()

Purpose	Set alpha text underline mode
Syntax	vsa_underline (dev_handle , mode_in)
Data Types	INT16 vsa_underline (); INT16 dev_handle ; INT16 mode_in ;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. mode_in Underline mode requested. 0 = off (default) 1 = on If an invalid mode is requested, the default is selected.
Function Returns	Function vsa_underline returns ≥0 if color index selected -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine turns alpha text underlining ON or OFF. The default is underlining OFF.

vsb_color ()

Purpose	Set background color index
Syntax	vsb_color (dev_handle, ind_in)
Data Types	INT16 vsb_color (); INT16 dev_handle; INT16 ind_in;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. ind_in Background color index. 0 to device maximum
Function Returns	Function vsb_color returns ≥0 if color index selected -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine sets the device's background color to the desired index. On some devices, this change may not appear until the next Clear Workstation function is called. If the index is not valid, no change will be made in the background index. However, the color index selected is always returned. This function is not applicable on plotters or printers.

vsf_color ()

Purpose	Set fill color index
Syntax	vsf_color (dev_handle, ind_in)
Data Types	INT16 vsf_color (); INT16 dev_handle; INT16 ind_in;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. ind_in Requested fill color index. 0 to device maximum
Function Returns	Function vsf_color returns ≥0 if color index selected -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine determines the color to be used for filling polygons, bars, pie slices and circles. At least two color indices are provided. Color indices range from 0 to a device-dependent maximum. If the color specified is invalid, the closest value in range is chosen. However, the color index selected is returned.

vsf_interior ()

Purpose	Set fill interior style
Syntax	vsf_interior (dev_handle, styl_in)
Data Types	INT16 vsf_interior (); INT16 dev_handle; INT16 styl_in;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. styl_in Requested fill interior style. 0 = HOLLOW 1 = SOLID 2 = PATTERN 3 = HATCH
Function Returns	Function vsf_interior returns ≥0 if style selected -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine sets the style of fill to be used for filled areas, bars, pie slices and circles. When you select HOLLOW style, the area is outlined in the current fill color. A SOLID area is filled in the current color. PATTERN and HATCH are determined by SET FILL STYLE INDEX. If the requested style is invalid, then HOLLOW is used. SOLID, HATCH, and PATTERN filled areas are not outlined.

vsf_style ()

Purpose	Set fill style index
Syntax	vsf_style (dev_handle, ind_in)
Data Types	INT16 vsf_style (); INT16 dev_handle ; INT16 ind_in ;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. ind_in Requested fill style index for PATTERN or HATCH fill.
Function Returns	Function vsf_style returns ≥0 if index selected -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine selects a fill style based on the fill interior style. This index has no effect if the interior style is either HOLLOW or SOLID. If the requested index is not available, index 1 is used. The index references a HATCH style if the fill interior style is HATCH, or a PATTERN if the fill interior style is PATTERN. At least six HATCH styles are provided: 1 = Narrow spaced +45 degree lines 2 = Medium spaced +45 degree lines 3 = Widely spaced +45 degree lines 4 = Narrow spaced +45 & -45 degree lines 5 = Medium spaced +45 & -45 degree lines 6 = Widely spaced +45 & -45 degree lines >6 = Device-dependent There is no difference between HATCH and PATTERN styles on many devices. For example on some devices, asking for a HATCH style of 3 will result in the same output as asking for a PATTERN style of 3.

vsl_color ()

Purpose	Set polyline color index
Syntax	vsl_color (dev_handle, ind_in)
Data Types	INT16 vsl_color (); INT16 dev_handle; INT16 ind_in;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. ind_in Requested color index. 0 to device maximum
Function Returns	Function vsl_color returns ≥0 if color index selected -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine sets the color index for subsequent polylines and arcs. At least two color indices are provided. Color indices range from 0 to a device-dependent maximum. If the color specified is invalid, the closest value in range is chosen. However, the color index selected is always returned. To change the appearance of a color, you must select the desired levels of the three color components (red, green and blue) that make up the index by calling the Set Color Representation routine. This can be used to create non-default colors such as brown or orange. Then Set Polyline Color Index may be called with this index. The new color will only be visible on devices that support color definition.

vsl_type ()

Purpose	Set polyline line type
Syntax	vsl_type (dev_handle, type_in)
Data Types	INT16 vsl_type (); INT16 dev_handle; INT16 type_in;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. type_in Requested line style. 1 = Solid 2 = Long dashed 3 = Dotted 4 = Dashed-dotted 5 = Medium dashed 6 = Dashed with two dots >6 = device-dependent
Function Returns	Function vsl_type returns ≥0 if line type selected -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine sets the line style (dash pattern) for subsequent polylines and arcs. The total number of line styles available is device-dependent, however, the above six line styles are provided. If the requested line style is out of range, then line style 1 is used.

vsl_width ()

Purpose	Set polyline line width
Syntax	vsl_width (dev_handle, wid_in)
Data Types	INT16 vsl_width (); INT16 dev_handle; INT16 wid_in;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. wid_in Requested line width in NDC units.
Function	Function vsl_width returns
Returns	≥ 0 if line width selected -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine sets the width for subsequent polylines and arcs. If the requested line width is outside of the device's capabilities, the line width is set to one device unit and returned in NDC units.

vsm_color ()

Purpose	Set polymarker color index
Syntax	vsm_color (dev_handle, ind_in)
Data Types	INT16 vsm_color (); INT16 dev_handle; INT16 ind_in;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. ind_in Requested polymarker color index. 0 to device maximum
Function Returns	Function vsm_color returns ≥0 if color index selected -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine sets the color index in which subsequent markers will be displayed. At least two color indices are provided. Color indices range from 0 to a device-dependent maximum. If the color specified is invalid, the closest value in range is chosen. However, the color index selected is always returned. To change the appearance of a color, you must select the desired levels of the three color components (red, green and blue) that make up the index. This can be used to create non-default colors such as brown or orange. The new color will only be visible on devices that support color definition.

vsm_height ()

Purpose	Set polymarker height
Syntax	vsm_height (dev_handle, hgt_in)
Data Types	INT16 vsm_height (); INT16 dev_handle; INT16 hgt_in;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. hgt_in Requested polymarker height in NDC units.
Function Returns	Function vsm_height returns ≥ 0 if height selected in NDC units -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine sets the size of subsequent polymarkers. If the requested marker height is outside of the capabilities of the device, the marker height is set to the closest device size. If the requested marker height does not exactly map to a device-supported size, then the largest device size that is not greater than the requested marker height is used. Marker sizes, just like graphics text sizes are specified by NDC values.

vsm_type ()

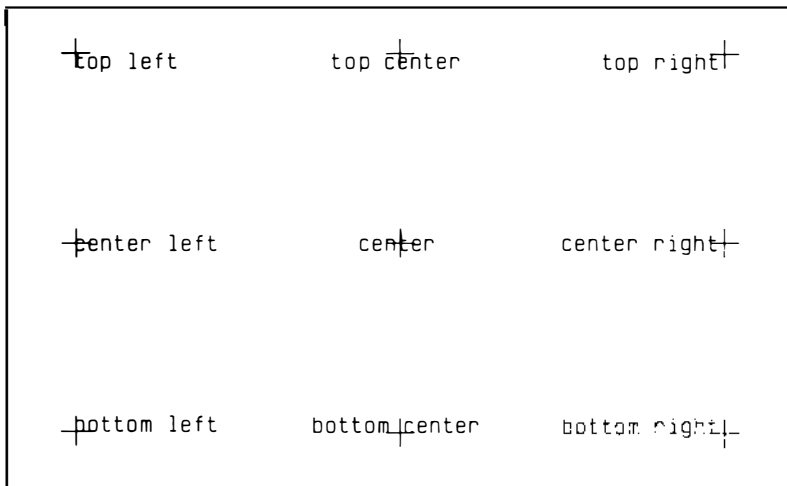
Purpose	Set polymarker type
Syntax	vsm_type (dev_handle, type_in)
Data Types	INT16 vsm_type (); INT16 dev_handle; INT16 type_in;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. type_in Requested polymarker type. 1 = . 2 = + 3 = * 4 = O 5 = X 6 = Diamond >6 = device-dependent
Function Returns	Function vsm_type returns ≥0 if marker type selected -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine sets the polymarker (symbol) type for subsequent polymarker operations. The total number of markers available is device-dependent, however, the above six marker types are provided. If the requested marker type is out of range, type 3 is used.

vst_alignment ()

Purpose	Set graphics text alignment
Syntax	vst_alignment (dev_handle, hor_in, vert_in, &hor_out, &vert_out)
Data Types	INT16 vst_alignment (); INT16 dev_handle; INT16 hor_in; INT16 vert_in; INT16 hor_out; INT16 vert_out;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. hor_in Horizontal alignment requested. Horizontal alignment applies to the width of the character body not the character cell. 0 = Left justified (default) 1 = Center justified 2 = Right justified If an invalid horizontal alignment is requested, the default of left justified is selected. vert_in Vertical alignment requested. Vertical alignment applies to the height of the character body not the character cell. 0 = Bottom justified (default) 1 = Center justified 2 = Top justified If an invalid vertical alignment is requested, the default of bottom is selected.
Output	hor_out Horizontal alignment selected. vert_out Vertical alignment selected.

Function	Function vst_alignment returns error state.
Returns	0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine sets graphics text horizontal and vertical alignment. This controls the positioning of the text extent rectangle in relation to the graphics text position. The default alignment places the bottom left-hand corner of the character (not the character cell) at the graphics text position.

FIGURE 5-6 Text Alignment



vst_color ()

Purpose	Set graphics text color index
Syntax	vst_color (dev_handle, ind_in)
Data Types	INT16 vst_color (); INT16 dev_handle; INT16 ind_in;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. ind_in Requested text color index. 0 to device maximum
Function Returns	Function vst_color returns ≥0 if color index selected -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine sets the graphics text color index. At least two color indices are provided. Color indices range from 0 to a device-dependent maximum. However, the color index selected is always returned. If the color index requested is not valid, the closest value within the range of the current device's capabilities is selected.

vst_font ()

Purpose	Set graphics text font
Syntax	vst_font (dev_handle, font_in)
Data Types	INT16 vst_font (); INT16 dev_handle; INT16 font_in;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. font_in Requested hardware graphics text font number. 1 to device-dependent maximum
Function Returns	Function vst_font returns ≥0 if font selected -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine sets the hardware text font for graphics text. Fonts are device-dependent and are specified from 1 to a device-dependent maximum. If a font is selected outside of device capability, font 1 is used.

vst_height ()

Purpose	Set character height
Syntax	vst_height (dev_handle, rq_height, &char_width, &cell_width, &cell_height)
Data Types	INT16 vst_height (); INT16 dev_handle; INT16 rq_height; INT16 char_width; INT16 cell_width; INT16 cell_height;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. rq_height Requested character height in NDC units.
Output	char_width Actual character width selected in NDC units. cell_width Character cell width in NDC units. cell_height Character cell height in NDC units.
Function Returns	Function vst_height returns error state. ≥0 if height selected in NDC units -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine sets the size of subsequent graphics text. The height specified is the height of the actual character (baseline to top of tallest character), not the character cell. If the requested size is outside of device capabilities, the closest available size on the device is used. If the desired character height does not map exactly to a device size, then the largest character height that does not exceed the desired size is used.

The default size permits at least eighty characters to be displayed horizontally across the display surface and twenty-four characters to be displayed vertically down the display surface.

Code Example—Set Character Height

```
titl_size = 160;
xheight = vst_height(dev_handle, title_size, &xwidth, &cwidth, &cheight);
/* set a new character height */
line_length = 800;
/* we want to make sure that the text we write will not
   be longer than this line */
while ((cwidth * 12) >= line_length && titl_size >= 20) {
/* loop making sure that a string 12 characters long
   won't be longer than line_length */
    title_size -= 10;
    xheight = vst_height(dev_handle, title_size, &xwidth, &cwidth, &cheight)
}
```

vst_rotation ()

Purpose	Set graphics text string baseline rotation
Syntax	vst_rotation (dev_handle, ang_in)
Data Types	INT16 vst_rotation (); INT16 dev_handle; INT16 ang_in;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. angin Requested angle of rotation of character base line. 0 to 3600 (tenths of degrees)
Function Returns	Function vst_rotation returns ≥0 if angle selected -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine sets the base line rotation of graphic text. The entire string of text is rotated (rather than each character separately) specified by the angle of rotation. The angle specification assumes that 0 degrees is 90 degrees to the right of vertical (east on a compass), with angles increasing in the counter-clockwise direction. If the desired angle is outside of the range (0-3600), then a character baseline of 0 degrees is used.

Input Functions

This section describes the input functions. An application program obtains graphical input from an operator by controlling the activity of one or more logical input devices that deliver input values to the program.

The input class determines the type of logical input value the device delivers. The four classes and the values they provide are:

- | | |
|----------|---|
| Locator | Locator input reports coordinate information from a device representing a location on the display surface. An example device is a mouse. |
| Valuator | Valuator input returns scalar values in the range that is proportional (0 to 32767) to the valuator position. An example device is a control dial. |
| Choice | Choice input returns an integer value indicating one of a set of alternatives. An example device is a set of function keys, and the value returned is the function key pressed. |
| String | String input returns text strings from the console keyboard. |

The input functions operate in two modes: sample and request. In sample mode, the input value (locator, valuator, choice or string) is returned immediately. In request mode, the input device is activated and waits for the user to terminate the input process with a device-specific action. Then the input value is returned. For example, a sample locator input returns the current location of the graphics cursor immediately. In request mode, the location of the graphics cursor is returned when the user types an alphanumeric key.

TABLE 5-4 Input Functions

<u>Function</u>	<u>Routine</u>	<u>Description</u>	<u>Page</u>
Locator	vrq_locator	Input Locator (request mode)	5-104
	vsm_locator	Input Locator (sample mode)	5-110
Valuator	vrq_valuator	Input Valuator (request mode)	5-108
	vsm_valuator	Input Valuator (sample mode)	5-113
Choice	vrq_curkeys	Read Cursor Movement Keys	5-101
	vrq_choice	Input Choice (request mode)	5-103
	vsm_choice	Input Choice (sample mode)	5-109
String	vrq_string	Input String (request mode)	5-106
	vsm_string	Input String (sample mode)	5-111

vr_d_curkeys ()

Purpose	Read cursor movement keys
Syntax	vr_d_curkeys (dev_handle, input_mode, &direction,&key)
Data Types	INT16 vr_d_curkeys (); INT16 dev_handle; INT16 input_mode; INT16 direction; INT8 key;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. input_mode Input mode. 1 = Request 2 = Sample
Output	direction Direction indicated. -1 = No keystroke occurred (sample mode only) 0 = Key was pressed, but not a cursor movement key 1 = Down and left 2 = Down 3 = Down and right 4 = Left 6 = Right 7 = Up and left 8 = Up 9 = Up and right key Key identification value. ASCII Decimal Equivalent value of pressed key. -1 if cursor movement key was pressed, or no key was pressed.
Function Returns	Function vr_d_curkeys returns >0 if request successful 0 if request unsuccessful -1 if error Actual error can be retrieved by invoking vq_error .

Description This routine determines if a cursor movement (arrow) key was pressed and returns the resultant direction. It can be used in either the Graphics Mode or Cursor Addressing Mode to input user cursor movement.

vrq_choice ()

Purpose	Input choice (request mode)
Syntax	vrq_choice (dev_handle , ch_in , & ch_out)
Data Types	INT16 vrq_choice (); INT16 dev_handle ; INT16 ch_in ; INT16 ch_out ;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. ch_in Initial choice number.
Output	ch_out Choice number.
Function Returns	Function vrq_choice returns >0 if request successful 0 if request unsuccessful -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine activates the choice device associated with a specified workstation. You must make a selection for the function to terminate. See Appendix C for information on the choice device for the desired workstation. The initial choice number, ch_in , is the value returned as the selected choice value if a non-choice device button/key is pressed. The initial choice number must be a valid value for it to be returned when the user does not press a valid choice input.

vrq_locator ()

Purpose	Input locator (request mode)
Syntax	vrq_locator (dev_handle , xy_in , ink , rubberband , echo_handle , xy_out , & terminator)
Data Types	INT16 vrq_locator (); INT16 dev_handle ; INT16 xy_in [2]; INT16 ink ; INT16 rubberband ; INT16 echo_handle ; INT16 xy_out [2]; INT8 terminator ;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. xy_in [0] Initial x-coordinate of locator in NDC units. xy_in [1] Initial y-coordinate of locator in NDC units. ink Inking status. 0 = off 1 = on If on, a line is drawn between the initial locator position and the final locator position, honoring the current line attributes, such as color and line type. If an invalid status is specified, then inking is turned off. rubberband Rubberbanding status. 0 = off 1 = rubber band line 2 = rubberband rectangle If rubberband line is specified, a line is drawn between the initial locator position and the current position of the locator device as it is moved. The line changes dynamically as the input device changes position. When the locator event is

complete the last rubberband line is removed from the display surface.

If rubberband rectangle is specified, a rectangle is drawn using the initial locator position as one corner and the current position of the locator device as the opposite corner. The rectangle changes dynamically as the input device changes position. When the locator event is complete the last rubberband rectangle is removed from the display surface. If an invalid status is specified, then rubberbanding is turned off. Rubberbanding honors current line attributes, such as color and line style.

echo_handle

The device handle of the device where the echoed output from the input operation will be displayed.

Output

xy_out[0]

Final x-coordinate of locator in NDC units.

xy_out[1]

Final y-coordinate of locator in NDC units.

terminator

Locator terminator.

For keyboard terminated locator input, this is the byte value of the key pressed to terminate input. For non-keyboard terminated input (tablet, mouse, etc.) valid locator terminators begin with <space> and increase from there. For instance, if the puck on a tablet has four buttons, the first button may generate a <space> as a terminator, the second a <!>, the third a <">, and the fourth a <#>.

Function Returns

Function **vrq_locator** returns

>0 if request successful

0 if request unsuccessful

-1 if error

Actual error can be retrieved by invoking **vrq_error**.

Description

This routine causes the graphics cursor to be displayed on the output echo device until the function is terminated. See Appendix C for information on the locator device for the desired workstation.

vrq_string ()

Purpose	Input string (request mode)
Syntax	vrq_string (dev_handle , max_length , echo_mode , echo_xy , string)
Data Types	INT16 vrq_string (); INT16 dev_handle ; INT16 max_length ; INT16 echo_mode ; INT16 echo_xy [2]; INT8 string [];
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. max_length Maximum string length. echo_mode Echo mode. 0 = Don't echo input characters 1 = Echo input characters echo_xy [0] x-coordinate of echo position in NDC units. echo_xy [1] y-coordinate of echo position in NDC units.
Output	string Output string. Passed as a contiguous stream of bytes.
Function Returns	Function vrq_string returns >0 if request successful (length of string) 0 if request unsuccessful -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine activates the keyboard. Any characters up to (but not including) the Return terminator are returned. Line editing characters have their normal effect and can be used if errors are made. The maximum string length must be ≥ 1 . This routine terminates when the "maximum

length'' number of characters or a line terminator has been entered. Since strings in C are null-terminated, the length of array string must be one larger than max_length.

vrq_valuator ()

Purpose	Input valuator (request mode)
Syntax	vrq_valuator (dev_handle, val_in, &val_out)
Data Types	INT16 vrq_valuator (); INT16 dev_handle; INT16 val_in; INT16 val_out;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. val_in Initial value.
Output	val_out Output value.
Function Returns	Function vrq_valuator returns status >0 if request successful 0 if request unsuccessful -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine activates the valuator device and the user sets it to the desired value. When ready, the user terminates the process with a device-specific action (for example, pushing a function key). See Appendix C for information on the valuator device for the desired workstation.

vsm_choice ()

Purpose	Input choice (sample mode)
Syntax	vsm_choice (dev_handle, &ch_out)
Data Types	INT16 vsm_choice (); INT16 dev_handle; INT16 ch_out;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Output	ch_out Choice. Choice number if sample successful 0 if sample unsuccessful
Function Returns	Function vsm_choice returns >0 if sample successful 0 if sample unsuccessful -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine polls the choice device and if a choice was pending, it is returned. See Appendix C for information on the choice function for the desired workstation.

vsm_locator ()

Purpose	Input locator (sample mode)
Syntax	vsm_locator (dev_handle , xy_out)
Data Types	INT16 vsm_locator (); INT16 dev_handle; INT16 xy_out[2];
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Output	xy_out[0] Current x-coordinate of locator in NDC units. xy_out[1] Current y-coordinate of locator in NDC units.
Function Returns	Function vsm_locator returns 1 if sample successful 0 if sample unsuccessful -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine returns the current position of the graphics input cursor without waiting for operator interaction. See Appendix C for information on the locator device for the desired workstation.

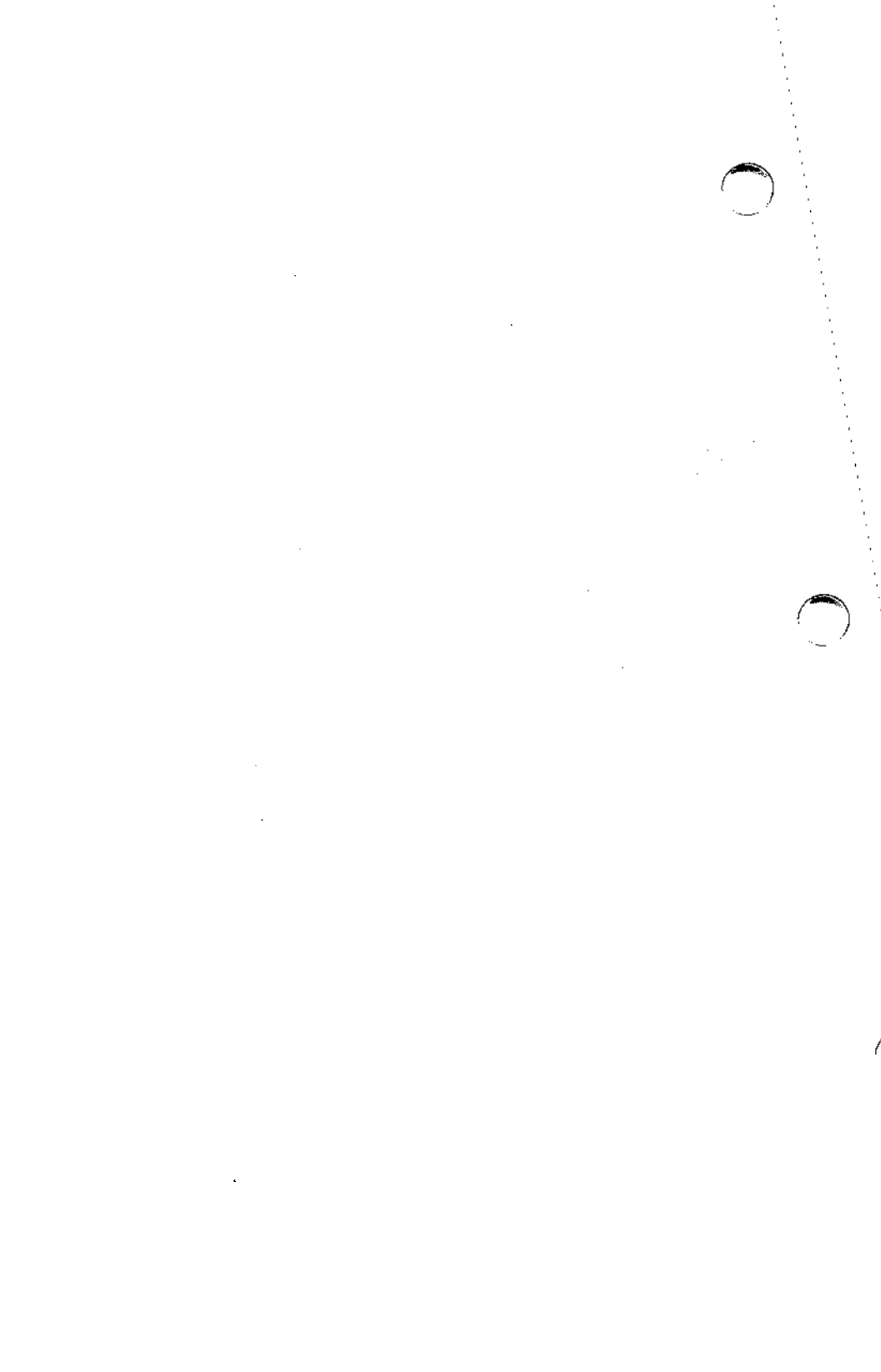
vsm_string ()

Purpose	Input string (sample mode)
Syntax	vsm_string (dev_handle , max_length , echo_mode , echo_xy , string)
Data Types	INT16 vsm_string (); INT16 dev_handle ; INT16 max_length ; INT16 echo_mode ; INT16 echo_xy[2] ; INT8 string[] ;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. max_length Maximum string length. echo_mode Echo mode. 0 = Don't echo input characters 1 = Echo input characters echo_xy[0] x-coordinate of echo position in NDC units. echo_xy[1] y-coordinate of echo position in NDC units.
Output	string Output string. Passed as a contiguous stream of bytes.
Function Returns	Function vsm_string returns >0 if sample successful (value = length of string) 0 if sample unsuccessful (value = characters not available) -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine polls the keyboard of the desired device. If there is any pending input, it is returned until the queue is empty, a carriage return or line feed is encountered or the input

maximum string length is exceeded. The line terminators themselves are not returned. See Appendix C for information on the string device for the desired workstation. Since strings in the C language are null-terminated, the length of array string must be one larger than max_length.

vsm_valuator ()

Purpose	Input valuator (sample mode)
Syntax	vsm_valuator (dev_handle, &val_out)
Data Types	INT16 vsm_valuator (); INT16 dev_handle; INT16 val_out;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Output	val_out Current valuator value, if sample successful.
Function Returns	Function vsm_valuator returns >0 if sample successful 0 if sample unsuccessful -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine returns the current value of the valuator device without waiting for operator interaction. See Appendix C for information on the valuator device for the desired workstation.



Inquiry Functions

This section describes the inquiry functions provided. These routines allow you to inquire on primitive attributes and device capabilities.

TABLE 5-5 Inquiry Functions

<u>Function</u>	<u>Routine</u>	<u>Description</u>	<u>Page</u>
Device Capabilities	vq_chcells	Inquire Addressable Character Cells	5-119
Primitive Attributes	vq_cellarray	Inquire Cell Array	5-117
	vq_color	Inquire Color Representation	5-120
	vqf_attributes	Inquire Current Fill Area Attributes	5-136
	vql_attributes	Inquire Current Polyline Attributes	5-134
	vqm_attributes	Inquire Current Polymarker Attributes	5-135
Text Attributes	vqa_cell	Inquire Alpha Text Cell Location	5-126
	vqa_font	Inquire Alpha Text Font Capability	5-128
	vqa_position	Inquire Alpha Text Position	5-132
	vqa_length	Inquire Alpha Text String Length	5-131
	vq_curaddress	Inquire Current Cursor Text Address	5-121
	vqt_attributes	Inquire Current Graphics Text Attributes	5-133
	vqa_cap	Inquire Alpha Text Capabilities	5-123
Errors	vq_error	Inquire VDI Error	5-122

vq_cellarray ()

Purpose	Inquire cell array
Syntax	vq_cellarray (dev_handle, xy, row_length, num_rows, &el_per_row, &rows_used, &status, colors)
Data Types	INT16 vq_cellarray (); INT16 dev_handle; INT16 xy[4]; INT16 row_length; INT16 num_rows; INT16 el_per_row; INT16 rows_used; INT16 status; INT16 colors[row_length * num_rows];
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. xy[0] x-coordinate of lower left-hand corner in NDC units. xy[1] y-coordinate of lower left-hand corner in NDC units. xy[2] x-coordinate of upper right-hand corner in NDC units. xy[3] y-coordinate of upper right-hand corner in NDC units. row_length Length of each row in color index array. num_rows Number of rows in color index array.
Output	el_per_row Number of elements used in each row of color index array. rows_used Number of rows used in color index array.

status

Invalid value flag.

0 if no errors

1 if a color value could not be determined
for some pixel

colors

Color index array (stored one row at a time).

-1 indicates that a color index could not be
determined for that particular pixel

Function

Function **vq_cellarray** returns error state

Returns

0 if no error

-1 if error

Actual error can be retrieved by invoking **vq_error**.

Description

This routine returns color indices one row at a time, starting from the top of the rectangular area, proceeding downward. See the Cell Array routine for information regarding how the rectangular area is divided.

vq_chcells ()

Purpose	Inquire addressable character cells
Syntax	vq_chcells (dev_handle, &rows, &columns)
Data Types	INT16 vq_chcells (); INT16 dev_handle; INT16 rows; INT16 columns;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Output	rows Number of addressable rows on the screen. -1 indicates cursor addressing is not possible. columns Number of addressable columns on the screen. -1 indicates cursor addressing is not possible.
Function Returns	Function vq_chcells returns error state 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine returns the number of cursor addressable columns in a row and the number of cursor addressable rows on the screen. It is only applicable to CRT devices and is useful for determining addressable page size.

vq_color ()

Purpose	Inquire color representation
Syntax	vq_color (dev_handle, in_in, set_flag, rgb)
Data Types	INT16 vq_color (); INT16 dev_handle; INT16 ind_in; INT16 set_flag; INT16 rgb[3];
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. ind_in Requested color index. set_flag Set or realized flag. 0 = Set (return color values requested) 1 = Realized (return color values realized on device)
Output	rgb[0] Red intensity. 0 to 1000 (tenths of percent) rgb[1] Green intensity. 0 to 1000 (tenths of percent) rgb[2] Blue intensity. 0 to 1000 (tenths of percent)
Function Returns	Function vq_color returns ≥0 if color index selected -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine allows inquiry of the color associated with a given index. If an index outside of the device's capability is requested, the index closest to it is used for the inquiry.

vq_curaddress ()

Purpose	Inquire current cursor text address
Syntax	vq_curaddress (dev_handle, &row, &column)
Data Types	INT16 vq_curaddress (); INT16 dev_handle; INT16 row; INT16 column;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Output	row Row number. 1 to number of rows column Column number. 1 to number of columns
Function Returns	Function vq_curaddress returns error state 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	When you are in Cursor Addressing Mode, this routine returns the current cursor position. It is only applicable to CRT devices.

vq_error

Purpose	Inquire VDI error
Data Types	INT16 vq_error() ;
Function	Function vq_error returns last error encountered.
Returns	0 if no error <0 actual error last encountered
Description	This routine returns the number of the actual error last encountered. It is called when the function value returned by any other function is -1. See Appendix B for the current list of error numbers.

vqa_cap ()

Purpose	Inquire alpha text capabilities
Syntax	vqa_cap (dev_handle, alph_cap)
Data Types	INT16 vqa_cap (); INT16 dev_handle; INT16 alph_cap[15];
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Output	alph_cap[0] Superscript capability flag. 0 = no 1 = yes alph_cap[1] Subscript capability flag. 0 = no 1 = yes alph_cap[2] Underline capability flag. 0 = no 1 = yes alph_cap[3] Overstrike capability flag. 0 = no 1 = yes alph_cap[4] Number of discrete alpha text sizes. where size 2 is larger (occupies more area) than size 1, etc. At least one size must be present. alph_cap[5] Discrete size index of the default font. This size is dependent on the number of font sizes supported.

alph_cap[6]

Character positioning capability flag.

0 = Characters positionable on cell boundaries only

1 = Characters positionable on a finer grid than a character cell, but not necessarily the same grid as graphics.

alph_cap[7]

Number of horizontal character cell positions across the display surface in the default font. For a typical CRT or printer that can only place text on cell boundaries, this value is 80.

alph_cap[8]

Number of vertical character cell positions down the display surface in the default font. This value is 24 for a typical CRT and 66 for a typical printer that can only place text on cell boundaries.

alph_cap[9]

Number of horizontal character cell positions represented by distance specified in `alph_cap[13]`. The ratio `alph_cap[13]/alph_cap[9]` can be used to determine the width of a character cell, including any roundoff error.

alph_cap[10]

Number of vertical character cell positions represented by distance specified in `alph_cap[14]`. The ratio `alph_cap[14]/alph_cap[10]` can be used to determine the height of a character cell, including any roundoff error.

alph_cap[11]

Number of horizontal alpha text grids represented by distance specified in `alph_cap[13]`. The ratio `alph_cap[13]/alph_cap[11]` can be used to determine the width of an alpha text grid, including roundoff error.

alph_cap[12]

Number of vertical alpha text grids represented by distance specified in `alph_cap[14]`. The ratio `alph_cap[14]/alph_cap[12]` can be used to determine the height of an alpha text grid, including roundoff error.

alph_cap[13]

Width in NDC units of the number of character cells (in the default font) specified in `alph_cap[9]`. This ratio (`alph_cap[13]/ alph_cap[9]`) can be used to determine the width of a character cell, including any roundoff error.

alph_cap[14]

Height in NDC units of the number of character cells (in the default font) specified in `alph_cap[10]`. This ratio (`alph_cap[14]/alph_cap[10]`) can be used to determine the height of a character cell, including any roundoff error.

Function	Function vqa_cap returns error state
Returns	0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine returns information regarding the alpha text features of the device, such as subscripting, superscripting and default character width and height.

vqa_cell ()

Purpose	Inquire alpha text cell location
Syntax	vqa_cell (dev_handle, row, column, &propflag, &x_out, &y_out)
Data Types	INT16 vqa_cell (); INT16 dev_handle; INT16 row; INT16 column; INT16 propflag; INT16 x_out; INT16 y_out;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. row Row number of character cell. 1 to number of rows column Column number of character cell. 1 to number of columns
Output	propflag Proportional spacing flag. 0 = no 1 = yes If this value is 1, then the size represented by x_out and y_out may not represent the selected font. This is the case if the desired font is proportionally spaced. x_out x-coordinate of lower left-hand corner of character cell in NDC units. Value may not be accurate if the proportional spacing flag is set to 1, since the character cell size is not constant. y_out y-coordinate of lower left-hand corner of character cell in NDC units. Value may not be accurate if the proportional spacing flag is set to 1, since the character cell size is not constant.

Function	Function vqa_cell returns
Returns	0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine returns the Normalized Device Coordinates of the lower left-hand corner of the character cell position specified, based on the current font. This allows text to be positioned in a specific column on the output device. Column 1 implies a 0 x position on the display surface, and row 1 implies the maximum y position on the display surface. This is not applicable if the current font is a proportional font since the size of a character cell is not constant.

vqa_font ()

Purpose	Inquire alpha text font capability
Syntax	vqa_font (dev_handle, font_in, size_in, font_status)
Data Types	INT16 vqa_font (); INT16 dev_handle; INT16 font_in; INT16 size_in; INT16 font_status[7];
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. font_in Requested font. Fonts 1 to 3 are fixed-space fonts. 1 = Normal/standard font (default) 2 = Bold (always provided for printers) 3 = Italics 4 = Proportionally spaced normal font 5 = Proportionally spaced bold 6 = Proportionally spaced italics >6 = Device-dependent size_in Requested text size. 1-device maximum where size n+1 is larger (occupies more area) than size n, etc.
Output	font_status[0] Number of horizontal character cell positions across the display surface in this font. This is -1 if the font selected is a proportional font, since the character cell size is not constant. It is 0 if the requested font is not available. font_status[1] Number of vertical character cell positions down the display surface in this font. This is 0 if the requested font is not available.

font_status[2]

Number of horizontal character cell positions represented by the distance specified in font_status[5]. This ratio $(\text{font_status}[5]/\text{font_status}[2])$ can be used to determine the width of a character cell, including any roundoff error. This is 0 if the requested font is not available.

font_status[3]

Number of vertical character cell positions represented by the distance specified in font_status[6]. This ratio $(\text{font_status}[6]/\text{font_status}[3])$ can be used to determine the height of a character cell, including any roundoff error. This is 0 if the requested font is not available.

font_status[4]

Proportional spacing flag.

0 = no

1 = yes

If this value is 1, then the size represented by font_status[5] and font_status[6] may not represent the selected font. This is the case if the desired font is proportionally spaced.

font_status[5]

Width in NDC units of the number of character cells (in the selected font) specified in font_status[2]. This ratio $(\text{font_status}[5]/\text{font_status}[2])$ can be used to determine the width of a character cell, including any roundoff error. This value is not accurate if the proportional spacing flag is set to 1, since the character cell size is not constant. It is 0 if the font is not available.

font_status[6]

Height in NDC units of the number of character cells (in the selected font) specified in font_status[3]. This ratio $(\text{font_status}[6]/\text{font_status}[3])$ can be used to determine the height of a character cell, including any roundoff error. It is 0 if the font is not available.

Function	Function vqa_font returns
Returns	>0 if font and size available 0 if font and size not available -1 if error Actual error can be retrieved by invoking vq_error.
Description	This routine inquires the attributes of a particular alpha text font and size, such as availability on this device and height and width. It can be used to determine the font which best fits specific size requirements without having to alter the currently set text font.

vqa_length ()

Purpose	Inquire alpha text string length
Syntax	vqa_length (dev_handle, string)
Data Types	INT16 vqa_length (); INT16 dev_handle; INT8 string[];
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open. string Text string. Passed as a contiguous stream of bytes. All control characters (ASCII 0 to 31) are ignored and not taken into account in computing the length.
Function Returns	Function vqa_length returns ≥ 0 if string length in NDC units -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine returns the length of the text string specified in NDC units, based on the current font in use. If a control character (ASCII 0-31) appears in the string, it terminates the string length, and the string length up to that point is returned. This routine is useful when using proportional fonts since each character is not the same width. It is also useful for doing microjustification between words since multiplication of the width of a character cell in NDC space may produce inaccurate results due to the inherent roundoff error in the character cell size reported back to the user.

vqa_position ()

Purpose	Inquire alpha text position
Syntax	vqa_position (dev_handle, &x_out, &y_out)
Data Types	INT16 vqa_position (); INT16 dev_handle ; INT16 x_out ; INT16 y_out ;
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Output	x_out x-coordinate of text position in NDC units. y_out y-coordinate of text position in NDC units.
Function Returns	Function vqa_position returns 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine reports the current alpha text position (returned in 0-32767 NDC units). It is assumed that (0,0) is at the lower left-hand corner of the display surface.

vqf_attributes ()

Purpose	Inquire current fill area attributes
Syntax	vqf_attributes (dev_handle, attrib)
Data Types	INT16 vqf_attributes (); INT16 dev_handle; INT16 attrib[4];
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Output	attrib[0] Current fill area interior style. 0 = HOLLOW 1 = SOLID 2 = PATTERN 3 = HATCH attrib[1] Current fill area color index. attrib[2] Current fill area style index. 1 = Narrow spaced +45 degree lines 2 = Medium spaced +45 degree lines 3 = Widely spaced +45 degree lines 4 = Narrow spaced +45 & -45 degree lines 5 = Medium spaced +45 & -45 degree lines 6 = Widely spaced +45 & -45 degree lines >6 = Device-dependent attrib[3] Current writing mode. (See Set Writing Mode function for description)
Function Returns	Function vqf_attributes returns 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	Reports current setting of all attributes that affect filled areas, bars, pie slices and circles, including interior style, fill color and fill style index.

vql_attributes ()

Purpose	Inquire current polyline attributes
Syntax	vql_attributes (dev_handle, attrib)
Data Types	INT16 vql_attributes (); INT16 dev_handle; INT16 attrib[4];
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Output	attrib[0] Current polyline line type. 1 = Solid 2 = Long dashed 3 = Dotted 4 = Dashed-dotted 5 = Medium dashed 6 = Dashed with two dots >6 = Device-dependent attrib[1] Current polyline line color index. attrib[2] Current writing mode. (See Set Writing Mode function for description.) attrib[3] Current line width in NDC units.
Function Returns	Function vql_attributes returns 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine reports the current setting of all attributes that affect polylines and arcs, such as line type, line color, line width and writing mode.

vqm_attributes ()

Purpose	Inquire current polymarker attributes
Syntax	vqm_attributes (dev_handle, attrib)
Data Types	INT16 vqm_attributes (); INT16 dev_handle; INT16 attrib[4];
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Output	attrib[0] Current polymarker marker type. 1 = . 2 = + 3 = * 4 = O 5 = X 6 = Diamond >6 = Device-dependent attrib[1] Current polymarker marker color index. attrib[2] Current writing mode. (See Set Writing Mode function for description.) attrib[3] Current polymarker height in NDC units. 0 if no error -1 if error Actual error can be retrieved by invoking vq_error .
Description	This routine reports the current setting of all attributes that affect polymarkers, such as marker type, marker color and marker height.

vqt_attributes ()

Purpose	Inquire current graphics text attributes
Syntax	vqt_attributes (dev_handle, attrib)
Data Types	INT16 vqt_attributes (); INT16 dev_handle; INT16 attrib[10];
Input	dev_handle Device handle returned from v_opnwk . Refers to a specific graphics device when multiple workstations are open.
Output	attrib[0] Current graphics text font. attrib[1] Current graphics text color. attrib[2] Current angle of rotation of text base line. 0 to 3600 (tenths of degrees) attrib[3] Current horizontal alignment. 0 = Left justified (default) 1 = Center justified 2 = Right justified attrib[4] Current vertical alignment. 2 = Top justified 1 = Center justified 0 = Bottom justified (default) attrib[5] Current writing mode. (See Set Writing Mode function for description.) attrib[6] Current character width in NDC units. attrib[7] Current character height in NDC units. attrib[8] Current character cell width in NDC units.

attrib[9]

Current character cell height in NDC units.

Function

Function **vqt_attributes** returns

Returns

0 if no error

-1 if error

Actual error can be retrieved by invoking **vq_error**.

Description

This routine reports the current setting of all attributes that affect graphics text, such as text size, text color, text font and text rotation.

Input/Output Functions



This section covers character and file I/O functions. Though these routines are not required in a graphics program, we have included them to insure computer independence for your programs.

All graphics calls are functions that return an integer value which may be the error status or other requested information.

The Open Workstation call initializes a device and returns a “device handle,” an integer number used to refer to that device. All other calls require the device handle as an input argument to identify the object device.



Communications (Port) I/O

Character I/O (cm) functions always return status. The status is in a 16-bit word where bit 0 is the least significant bit and bit 15 is the most significant bit. Not all cm functions return all the status fields. The fields that a function returns are noted in the descriptions of the individual functions.

Bit(s)	Meaning	Yes	No
15	Error	1	0
14	Always reset to 0		
13	Always reset to 0		
12	Receive ready	1	0
11	Transmit ready	1	0
10	EOF	1	0
9	Comm initialized	1	0
8	(Unused)		
7	Framing error	1	0
6	Overrun error	1	0
5	Parity error	1	0
0-4	Lower five bits of VDI error codes		

The high bit ON (bit 15 set to 1) indicates that an error has occurred. The lower five bits of such a negative return function value correspond to one of the VDI error codes listed in Appendix B, "Error Codes."

Disk I/O



The disk I/O (fd) functions treat a file as a linear sequence of bytes. The length of the file is the number of bytes in the file. There is a current file position associated with each open file which is the next byte to be read or written in the file.

Each read/write operation implicitly advances the current file position forward by the number of bytes read/written. A seek operation is provided to set the current file position without reading or writing any bytes. A directory is an area in which a file resides. It is specified as a null-terminated string of bytes. The syntax is dependent on the operating environment.



A file name is specified as a null-terminated string of bytes. The syntax is dependent on the operating environment. A fully qualified file name consists of a directory and simple file name. All file operations take a file descriptor (fd) as an argument. This descriptor is used to identify a connection to a directory or file. A maximum of sixteen file descriptors (fd's), open files, may exist at one time. There may be a maximum of eighty characters in a fully qualified file name.

Unless otherwise stated (as in `fd_parse`), directories may be specified relative to the current directory or be fully qualified (with a leading slash).

VDI error codes contain both generic and system-dependent information based on the following rules:

- A negative return from a function always implies an error.
- An error return greater than or equal to zero indicates that no error occurred.
- The generic part of the code is of the form-**XX00**
- The system-dependent part of the code is of the form-**00YY**

Error codes are listed in Appendix B, "Error Codes."



TABLE 5-6 Input/Output Functions

<u>Function</u>	<u>Routine</u>	<u>Description</u>	<u>Page</u>
Character I/O	cm_start	Initialize I/O System	5-146
	cm_stop	Close I/O System	5-147
	cm_open	Initialize Logical Channel	5-145
	cm_close	Close Logical Channel	5-143
	cm_inq	Inquire Status of Channel	5-144
	cmrx_wait	Read Character From Channel With Wait	5-149
	cmrx_now	Read Character From Channel Without Wait	5-148
	cmtx_wait	Send Character With Wait	5-151
	cmtx_now	Send Character Without Wait	5-150
File I/O	fd_connect	Connect Directory	5-153
	fd_disconnect	Disconnect Directory	5-157
	fd_copy	Copy Directory	5-154
	fd_open	Open File	5-159
	fd_close	Close File	5-152
	fd_read	Read File Wait	5-161
	fd_write	Write File Wait	5-165
	fdp_read	Read File Proceed	5-166
	fdp_write	Write File Proceed	5-167
	fd_inq	Inquire File Status	5-158
	fd_seek	Seek File	5-163
	fd_size	File Size	5-164
	fd_delete	Delete File	5-155
	fd_rename	Rename File	5-161
	fd_directory	Read Directory	5-156
	fd_parse	Parse File Name	5-160

cm_close ()

Purpose	Close logical channel
Syntax	cm_close (channel)
Data Types	INT16 cm_close (); INT16 channel;
Input	channel Logical channel number.
Function Returns	Function cm_close returns status of operation.
Description	This routine flushes the output buffer associated with the I/O channel and marks the logical channel as uninitialized. Default parameters will be reset to their original values (prior to being set by cm_open). Bits 15, 9 and 0-4 are valid in the returned status.

cm_inq ()

Purpose	Inquire status of channel
Syntax	cm_inq (channel)
Data Types	INT16 cm_inq (); INT16 channel;
Input	channel Logical channel number.
Function Returns	Function cm_inq returns status of operation.
Description	This routine returns the status of the specified channel. All fields are valid except bits 5-7. If bit 11 is 1 and you attempt to send a character, the write will be successful. If bit 12 is 1 and you attempt to read a character, the read will be successful.

cm_open ()

Purpose Initialize logical channel

Syntax **cm_open (channel)**

Data Types **INT16 cm_open ();**
INT16 channel;

Input **channel**
Logical channel number.
0 = Alpha console/CRT
1 = Graphics CRT
2 = Printer
3 = Plotter
4 = Messages
5 = Host communications
6 = Auxiliary graphics input

Function Returns Function **cm_open** returns status of operation.

Description This routine performs the environment-dependent actions necessary to prepare the logical channel for input/output. Baud rate, stop bits, and parity must be set by the user through standard operating system commands prior to using the channel. All other terminal port parameters will be set as required by GSS-DRIVERS. Sixteen logical channels are provided to allow for expansion. In practice, many logical channels may map the same physical channel. All fields are valid in the returned status.

These routines may be nested but the opened channels must be closed in the reverse order to the order in which they were opened. Failure to close channels in this reverse order may result in parameters being reset to incorrect values.

cm_start

Purpose	Initialize I/O system
Data Types	INT16 cm_start ();
Function Returns	Function cm_start returns status of operation.
Description	This routine performs the actions necessary to start the I/O system, such as setting all logical channels to the uninitialized state. Bits 15, 9 and 0-4 are valid in the returned status.

cm_stop

Purpose	Close I/O system
Data Types	INT16 cm_stop ();
Function	Function cm_stop returns status of operation.
Returns	
Description	This routine flushes all output buffers associated with all I/O channels. Default parameters will be reset to their original values (prior to being set by cm_open). Bits 15, 9 and 0-4 are valid in the returned status.

cmrx_now ()

Purpose	Read character from channel without wait
Syntax	cmrx_now (channel, &cp)
Data Types	INT16 cmrx_now (); INT16 channel; INT8 cp;
Input	channel Logical channel number.
Output	cp Character read.
Function Returns	Function cmrx_now returns status of operation.
Description	If a character is available on the specified channel, this routine places it in the byte pointed to by the second parameter. Bits 15, 12, 10, 9 and 0-7 are valid in the returned status.

cmrx_wait ()

Purpose	Read character from channel with wait
Syntax	cmrx_wait (channel, &cp)
Data Types	INT16 cmrx_wait (); INT16 channel; INT8 cp;
Input	channel Logical channel number.
Output	cp Character read.
Function Returns	Function cmrx_wait returns status of operation.
Description	This routine waits until a character has been received on the specified channel and then places it in the byte pointed to by the second parameter. Bits 15, 12, 10, 9 and 0-7 are valid in the returned status.

cmtx_now ()

Purpose	Send character without wait
Syntax	cmtx_now (channel, c)
Data Types	INT16 cmtx_now (); INT16 channel; INT8 c;
Input	channel Logical channel number. c Character sent.
Function Returns	Function cmtx_now returns status of operation.
Description	If the specified channel is ready, then this routine sends the character specified as the second parameter. If a no-wait function is not available on the channel, then this routine is identical to Send Character With Wait. Bits 15, 12, 9 and 0-4 are valid in the returned status.

cmtx_wait ()

Purpose	Send character with wait
Syntax	cmtx_wait (channel, c)
Data Types	INT16 cmtx_wait (); INT16 channel; INT8 c;
Input	channel Logical channel number. c Character sent.
Function Returns	Function cmtx_wait returns status of operation.
Description	This routine waits until the specified channel is ready, then sends a character specified as the second parameter. Bits 15, 12, 9 and 0-4 are valid in the returned status.

fd_close ()

Purpose	Close file
Syntax	fd_close (fd)
Data Types	INT16 fd_close (); FD fd;
Input	fd File descriptor.
Function Returns	Function fd_close returns status of operation. <0 if error -6YY if unable to close the file associated with the fd.
Description	This routine closes the file specified by the input file descriptor. This includes writing any pending data buffers to disk. The directory associated with that fd is the one associated with the fd_connect call. The file descriptor remains active until fd_disconnect is called.

fd_connect ()

Purpose	Connect directory
Syntax	fd_connect (directory)
Data Types	FD fd_connect (); INT8 directory[];
Input	directory Directory name. string
Function Returns	Function fd_connect returns status of operation. ≥0 if fd associated with the directory <0 if error -2YY if unable to connect to directory
Description	This routine returns a file descriptor that is associated with the specified directory. This must be done before a file can be opened and any I/O operations performed. Note that the current working directory can be specified with a null string ("") or a dot (.). The string cannot exceed eighty characters. Additional characters beyond eighty will be truncated.

fd_copy ()

Purpose	Copy directory
Syntax	fd_copy (fd)
Data Types	FD fd_copy (); FD fd;
Input	fd File descriptor.
Function Returns	Function fd_copy returns status of operation. <0 if error -4YY if unable to create a new fd
Description	This routine returns a unique file descriptor that is associated with the same directory of the file descriptor specified. If the fd is connected to a file, the function returns an error.

fd_delete ()

Purpose	Delete file
Syntax	fd_delete (fd, name)
Data Types	INT16 fd_delete (); FD fd; INT8 name[];
Input	fd File descriptor. name File name.
Function	Function fd_delete returns status of operation.
Returns	<0 if error -14YY if unable to delete file
Description	This routine deletes the file specified. The full path name (directory name plus file name) cannot exceed eighty characters.

fd_directory ()

Purpose	Read directory
Syntax	fd_directory (fd, name, buffer, space_available)
Data Types	INT16 fd_directory (); FD fd; INT8 name[]; INT8 buffer[]; INT16 space_available;
Input	fd File descriptor. name File name qualifier. space_available Number of bytes available in buffer.
Output	buffer Directory buffer location.
Function Returns	Function fd_directory returns amount of space used to hold both directory name and file name. Space needed is returned if space needed is greater than space available. (Only space available bytes of buffer were filled.) -16YY returned if unable to get directory list on fd
Description	This function returns the names of all files matching the specified name in the directory associated with the specified fd. File names are specified as null-terminated byte strings which are placed in the buffer. Input name descriptions are specified as null-terminated byte strings. Wildcards are allowed in input name descriptions: * matches any string of characters in name ? matches any individual character in name A null name specifies that the names of all files in the directory are to be returned. The end of the buffer is marked by an additional null following the final name/null entry.

fd_disconnect ()

Purpose	Disconnect directory
Syntax	fd_disconnect (fd)
Data Types	INT16 fd_disconnect (); FD fd;
Input	fd File descriptor.
Function	Function fd_disconnect returns status of operation.
Returns	<0 if error -300 if unable to disconnect the fd
Description	This routine disassociates the specified fd from a directory.

fd_inq ()

Purpose	Inquire file status
Syntax	fd_inq (fd)
Data Types	INT16 fd_inq (); FD fd;
Input	fd File descriptor.
Function	Function fd_inq returns status of operation.
Returns	≥ 0 if number of bytes read/written by the last fdp_read or fdp_write . -11YY if unable to get read/write status for the fd.
Description	This routine returns the number of bytes read or written by the last fdp_read or fdp_write routine, unless an illegal fd is specified in which case it returns an error.

fd_open ()

Purpose Open file

Syntax **fd_open (fd, name, mode)**

Data Types **INT16 fd_open ();**
FD fd;
INT8 name[];
INT16 mode;

Input **fd**
File descriptor.

name
File name.

mode
Access mode.

Function Returns Function **fd_open** returns status of operation.
-5YY if unable to open the requested file.

Description This routine opens the specified file in the directory associated with the specified fd. The file descriptor (fd) will be ignored if the file name (name) is fully qualified. Mode is a 16-bit word where bit 0 is the least significant and bit 15 is the most significant bit.

Bit	Meaning
0	Read access 0 = no 1 = yes
1	Write access 0 = no 1 = yes
2	ASCII/binary flag 0 = ASCII text 1 = Binary (8-bit) data

The current file position is initialized to 0, that is, before the first byte of the file.

fd_parse ()

Purpose	Parse file name
Syntax	fd_parse (qualified_name, buffer, space_available)
Data Types	INT16 fd_parse (); INT8 qualified_name[]; INT8 buffer[]; INT16 space_available;
Input	qualified_name Fully qualified file name to be parsed.
Output	buffer Parsing buffer location. space_available Buffer space available (size).
Function Returns	Function fd_parse returns amount of space used to hold both the directory name and file name. If space needed is greater than the space available, then space needed is returned. (Only space available bytes of the buffer are filled.)
Description	This routine returns the directory and name portions of a fully qualified file name. The buffer contains two null-terminated byte strings (directory and file name). This routine is useful for separating the directory from the file name.

fd_read ()

Purpose	Read file wait
Syntax	fd_read (fd, buffer, count)
Data Types	INT16 fd_read (); FD fd; INT8 buffer[]; INT16 count;
Input	fd File descriptor. count Desired number of bytes.
Output	buffer Read buffer location.
Function Returns	Function fd_read returns number of bytes read. <0 if error -7YY if error reading file
Description	This routine reads the specified number of bytes from the file associated with the specified file descriptor, starting at the current file position. It returns the number of bytes actually read and increments the current file position by the number of bytes read. If the number of bytes requested is greater than the number of bytes currently in the buffer, all bytes in the buffer will be returned. If the file position is at end of file, then the routine reports that zero bytes were read.

fd_rename ()

Purpose	Rename file
Syntax	fd_rename (fd, old_name, new_name)
Data Types	INT16 fd_rename (); FD fd; INT8 old_name[]; INT8 new_name[];
Input	fd File descriptor. old_name Old file name. new_name New file name.
Function	Function fd_rename returns status of operation.
Returns	<0 if error -15YY if unable to rename file
Description	This routine changes the name of file old_name to be new_name.

fd_seek ()

 Purpose Seek file

Syntax **fd_seek (fd, position, offset)**

Data Types **INT32 fd_seek ();**
 FD fd;
 INT32 position;
 INT16 offset;

Input **fd**
 File descriptor.

position
 Specifies new file position.

offset
 Offset mode.
 0 = Absolute
 1 = Relative

 Function Returns Function **fd_seek** returns status of operation.
 ≥ 0 if updated position
 < 0 if error
 -12YY if unable to seek on requested fd

Description This routine sets the current file position of the file associated with fd to the position specified in the position argument. The position may be specified either as an absolute offset from the beginning of the file (offset = 0) or an offset relative to the current position (offset = 1). Both the position requested and the updated position are long integers (32 bits).

fd_size (fd)

Purpose	File size
Syntax	fd_size (fd)
Data Types	INT32 fd_size (); FD fd;
Input	fd File descriptor.
Function	Function fd_size returns status of operation.
Returns	The size returned is a long integer (32 bits): ≥ 0 if file size in bytes < 0 if error -13YY if unable to determine file size on requested fd.
Description	This routine returns the current length of the file associated with the specified file descriptor in bytes. This value can be used to set the file position to the end of the file and append to the file. To obtain the size of an open file, first call fd_close to close the file.

fd_write ()

Purpose	Write file wait
Syntax	fd_write (fd, buffer, count)
Data Types	INT16 fd_write (); FD fd; INT8 buffer[]; INT16 count;
Input	fd File descriptor. buffer Write buffer location. count Desired number of bytes.
Function Returns	Function fd_write returns actual number of bytes written. <0 if error -10YY if error writing file
Description	This routine writes the specified number of bytes to the file associated with the specified file descriptor, starting at the current file position. It returns the number of bytes actually written and increments the current file position by the number of bytes written.

fdp_read ()

Purpose	Read file proceed
Syntax	fdp_read (fd, buffer, count)
Data Types	INT16 fdp_read (); FD fd; INT8 buffer[]; INT16 count;
Input	fd File descriptor. count Desired number of bytes.
Output	buffer Read buffer location.
Function Returns	Function fdp_read returns status. >0 if number of bytes read 0 if read operation in progress -7YY if error reading file
Description	This routine reads the specified number of bytes from the file associated with the specified file descriptor, starting at the current file position. It increments the current file position by the number of bytes read. This routine does not wait for the read to be complete to return. The Inquire File Status routine can be used to determine if file operation has been completed. This operation is the same as a Read File.

fdp_write ()

Purpose	Write file proceed
Syntax	fdp_write (fd, buffer, count)
Data Types	INT16 fdp_write (); FD fd; INT8 buffer[]; INT16 count;
Input	fd File descriptor. buffer Write buffer location. count Desired number of bytes.
Function Returns	Function fdp_write returns status. >0 if number of bytes written 0 if write operation in progress -8YY if unable to write to requested file
Description	This routine writes the specified number of bytes to the file associated with the specified file descriptor, starting at the current file position. It increments the current file position by the number of bytes written. This routine does not wait for the write to be complete to return. The Inquire File Status routine can be used to determine if file operation has been completed. This operation is the same as a Write File.



Glossary

ADE	ASCII Decimal Equivalents are decimal numbers used in code to represent ASCII characters. Examples are 65=A, 66=B. ADE character parameters are passed and returned as integers.
Argument	One of the independent variables that the action or output of a routine depends on. Arguments are enclosed in parentheses in the routine call.
Array	Series of related items (data) arranged in a meaningful pattern.
Aspects of Primitives	Ways in which the appearance of a primitive can vary. Aspects are controlled directly by primitive attributes.
Attribute Functions	Primitive attributes affect the appearance of objects created with primitive functions. (Examples: character height, line style)
Binding	Language binding refers to the exact calling syntax and data type specification for arguments to be used when calling GSS-DRIVERS routines from a specific programming language.
Cartesian Coordinate System	Coordinate system composed of an X-axis (horizontal) increasing positively towards the right, and a Y-axis (vertical) increasing positively upwards. The axes are positioned at right angles, and the point of intersection is the origin (0., 0.). The position of any point is defined by the displacement from the origin along first the X- and then the Y-axis.

Cell Array GSS-DRIVERS output primitive consisting of a rectangular grid of equal size rectangular cells, each having a single color.

Note

These cells may not map one-to-one with frame buffer pixels.

Choice Input Device A logical input device that offers the user a set of alternatives and returns an integer value indication of the option selected. An example device is a set of function keys.

Clipping When you set a window in the world coordinate space, part of an object may lie outside the window. In this case, the part lying outside the window will be clipped; that is, it will not be projected onto the viewport.

Color Map Table designed to provide a range of colors by defining different mixtures of the color components. A desired color is referenced by its assigned number. The identifying numbers with their assigned colors are called the color map. Changing colors assigned to the identifying number changes the map.

Color Table Workstation-dependent table in which the entries specify the values of the red, green and blue intensities defining a particular color. Control Functions These facilities allow you to exercise control over certain aspects of the system and the display device. GSS-DRIVERS provides a means to access the non-standard capabilities of your display device through an escape mechanism invoked with the escape function.

Coordinate Graphics Computer graphics in which display images are generated from display commands and coordinate data.



Coordinate Scaling

Coordinate scaling transforms points from one "space" to another. In GSS-DRIVERS all point coordinates must be specified in Normalized Device Coordinates with values between 0 and 32767. These coordinates are then scaled into values which are appropriate for your graphics device.

Default

A value assigned to a parameter by GSS-DRIVERS and used when you do not specify a value.

Device Coordinate

A coordinate expressed in a coordinate system that is device-dependent.

Device Driver

Device-dependent software that generates instructions specifying items to be drawn on the display surface from the invocations of GSS-DRIVERS.



Device Handle

Number returned that represents a unique device.

Device-Independent

The ability to be used on more than one type of graphics display device.

Device Space

The space defined by the addressable points of a display device.

Display Device

A device (for example, refresh display, storage tube display, plotter) on which display images can be represented.

Display Surface

In a display device, that medium on which display images may appear.

Echo

The immediate notification of the current value provided by an input device to the operator at the display console.



Environmentals

Environmentals are attributes, selected by the programmer, that affect the appearance of some aspect of the displayed graph. For example, there is a call to set the line type for polyline output. Environmentals have default values which are in effect after initialization of GSS-DRIVERS. If the

	programmer alters an environmental, it retains its new value until it is changed by another attribute call or GSS-DRIVERS is reinitialized.
Escape	A function within GSS-DRIVERS which is the only access to implementation-dependent or device-dependent support for nonstandard functionality other than graphics output.
Fill Area	A GSS-DRIVERS output primitive consisting of a polygon (closed boundary) which may be hollow or may be filled with a uniform color, a pattern, or a hatch style.
Generalized Drawing Primitive (GDP)	A display element (output primitive) used to address special geometrical workstation capabilities such as curve drawing.
GKS	Graphical Kernel System.
Graphical Kernel System	The Graphical Kernel System (GKS) is an international standard for the programmer's interface to graphics.
Graphics Primitives	Graphics primitives are the basic graphics operations performed by GSS-DRIVERS; for example, drawing lines, markers and text strings.
GSS-DRIVERS	GSS-DRIVERS is a host- and device-independent graphics subsystem that serves as an environment for graphics applications as well as application development.
Host-Independent	Capable of running on a number of operating systems.
Input Functions	GSS-DRIVERS allows you to obtain the value of an NDC point from an interactive graphics device. The method for specifying the point is device-dependent.

Inquiry Functions

GSS-DRIVERS provides inquiry facilities that allow your program to determine the present state of the system. You may determine the current value of the following:

- primitive attributes
- device capabilities
- device state

Integer

A whole number; that is, a number with no fractional part.

I/O System

Host-dependent part of GSS-DRIVERS which allows GSS-DRIVERS to communicate with your graphics devices using the standard host hardware.

Locator Input Device

A GSS-DRIVERS logical input device providing a position in world coordinates and a normalization transformation number. Examples include cursors, mice and joysticks.

NDC

Normalized Device Coordinates.

Normalized Device Coordinate Space

Normalized Device Coordinate Space is a uniform virtual space by which a graphics application program passes graphics information to a device. GSS-DRIVERS translates between NDC space and the display coordinates of a particular device.

Normalized Device Coordinates

GSS-DRIVERS introduces the concept of a Normalized Device Coordinate (NDC) Space in which the full extent of the device axes are assigned values between 0 and 32767. This convention provides improved device independence for a graphics system by allowing the viewing operations to be carried out without regard for device coordinate specifics. The NDC coordinates are then converted to specific device coordinates by GSS-DRIVERS.

Null-Terminated String	A one-dimensional array or list of characters. The end of a string is indicated by the NULL character (ADE 0).																																			
Output Primitives	The graphical world which the programmer describes consists of one or more objects. Objects are created and modified by invocations of graphics primitive functions provided by GSS-DRIVERS. These functions describe polylines, polymarkers, text strings, pixel arrays, fill areas and generalized drawing primitives. The invocation of an output primitive function results in an output primitive, such as a sequence of markers or polylines. The appearance of output primitives is affected by the values of primitive attributes.																																			
Pixel	The smallest element of a display surface that can be independently assigned a color or intensity.																																			
Polyline	A GSS-DRIVERS output primitive consisting of a set of connected lines.																																			
Polymarker	A GSS-DRIVERS output primitive consisting of a set of locations to be indicated by a marker.																																			
Raster	A field of closely spaced lines on the face of a video terminal that defines an image. The spacing between raster lines defines the resolution of a display.																																			
Real	A number which contains a fractional part expressed as a decimal. For example, 23.56.																																			
RGB	An additive method for defining color, in which colors are produced by adding percentages of the primaries: <table><tr><td></td><td></td><td>red</td><td>+</td><td>green</td><td>+</td><td>blue</td></tr><tr><td>white</td><td>=</td><td>100%</td><td>+</td><td>100%</td><td>+</td><td>100%</td></tr><tr><td>yellow</td><td>=</td><td>100%</td><td>+</td><td>100%</td><td>+</td><td>0%</td></tr><tr><td>magenta</td><td>=</td><td>100%</td><td>+</td><td>0%</td><td>+</td><td>100%</td></tr><tr><td>cyan</td><td>=</td><td>0%</td><td>+</td><td>100%</td><td>+</td><td>100%</td></tr></table>			red	+	green	+	blue	white	=	100%	+	100%	+	100%	yellow	=	100%	+	100%	+	0%	magenta	=	100%	+	0%	+	100%	cyan	=	0%	+	100%	+	100%
		red	+	green	+	blue																														
white	=	100%	+	100%	+	100%																														
yellow	=	100%	+	100%	+	0%																														
magenta	=	100%	+	0%	+	100%																														
cyan	=	0%	+	100%	+	100%																														



Transformation

The mapping of objects from one coordinate space to another.

Valuator Input Device

A logical input device that returns scalar values in the range that is proportional (0 to 32767) to the valuator position. An example device is a control dial.

VDI

Virtual Device Interface.

Virtual Device Interface

The Virtual Device Interface is a standard interface between device-dependent and device-independent code in a graphics environment. VDI makes all device drivers appear identical to the calling program. GSS-DRIVERS is based on VDI and all device drivers written for GSS-DRIVERS must conform to the VDI specification.



Workstation

GSS-DRIVERS is based on the concept of abstract graphical workstations which provide the logical interface through which the applications program controls physical devices.





Appendices

A	C Conventions and Example	A-1
	Using the Bindings	A-1
	A Note on Binding Conventions	A-1
	Programming Example	A-2

B	Error Codes	B-1
----------	--------------------	------------

C	Device-Dependent Information	C-1
	Device Information Categories	C-2
	AT&T Model 470 Graphics Printer	C-4
	AT&T Model 455 Letter Quality Printer	C-8
	AT&T UNIX PC Model 7300 Display	C-12
	CIT-101e Terminal™ with CIG-101e Graphics Card™	C-17
	CIT-161 Terminal™ with CIG-267 Graphics Card™	C-22
	Diablo® C150 InkJet Color Printer	C-27
	Epson MX™-100	C-31
	Epson MX-80 with Graphtrax Plus	C-35
	Hewlett-Packard 7470A Plotter	C-39
	Hewlett-Packard 7475A Plotter	C-43
	HILOT DMP-29 Plotter®	C-47
	NEC Model 7730 Letter Quality Printer	C-51
	NEC Model 3550 Letter Quality Printer	C-55
	Nicolet Instrument Corporation ZETA™-8 Plotter	C-59
	Okidata Microline™ 93	C-63
	Okidata Microline 84 with Step 2 Support	C-63
	Okidata Microline 92	C-67
	Printronix® MVP™ Printer	C-71
	Printronix P300 Printer	C-71
	Printronix P600 Printer	C-71
	Strobe® Model 100—Single pen Plotter	C-75
	Strobe Model 200—Single pen Plotter	C-75
	Strobe Model 260—Eight pen Plotter	C-75

Summagraphics® Summatablet™	C-79
Tektronix™ 4105 Color Graphics Terminal	C-81
Virtual Device Metafile (VDM)	C-87
VT100™ Terminal with Retro-Graphics™ Card	C-92

D	ASCII Code Chart	D-1
----------	-------------------------	-----

E	Bibliography	E-1
----------	---------------------	-----

C Conventions and Example

GSS-DRIVERS C language binding provides an interface between VDI-compatible device drivers and your C compiler. This allows the programmer to specify graphics operations as subroutine calls with the native syntax of the language environment. By hiding many of the details of the device-driver interface, the language binding eases programming tasks and allows the programmer to concentrate on his application.

Using the Bindings

The C binding is implemented as a linkable library containing all the functions described in Part 5. This library must be included in the list of relocatable modules during the linking process. The normal operating procedures for your linker should be followed when linking GSS-DRIVERS bindings to an application.

A Note on Binding Conventions

In the C bindings, INT16 refers to signed 16-bit integers and INT8 refers to signed 8-bit integers. FD refers to signed 16-bit integers. INT32 refers to long integer (32 bit). Scalar input arguments are passed by value; array input arguments are passed by address. All output arguments are passed by address.

All graphics calls are functions that return an integer value which may be the error status or other requested information.

The Open Workstation call initializes a device and returns a "device handle," an integer number used to refer to that device. All other calls require the device handle as an input argument to identify the object device.

Programming Example

The following C programming example accompanies the pseudocode in Part 2, producing the Gantt Chart in Figure 2-2.

/ This is a program to use the VDI C binding to draw a Gantt chart */*

```
main()
{
    static char *tasks[ ] = {"Full Production", "Evaluation", "One Third",
        "Plant", "Purchasing", "Tool Design", "Review", "Design"};
    static char *title = {"Aurora Processing Plant"};
    static char *y_label = {"As of June 15"};
    static char *y_ticks[ ] = {"July", "August", "September", "October"};

    extern box();
    extern short pto32k();

    short dev_handle,
        xheight,
        istring[2],
        gdms_err,
        xy[10],
        savary[64],
        xwidth,
        cwidth,
        i, j,
        cheight;

    static short echo_xy[2] = {0,0};
    static short nominate[ ] = {0, 1, 1, 3, 1, 1, 1, 0, 0, 1, 1,
        'D', 'I', 'S', 'P', 'L', 'A', 'Y', ' '};
    static short start_dates[ ] = {83, 72, 70, 48, 48, 45, 40};
    static short end_dates[ ] = {95, 83, 79, 75, 67, 60, 49};
    gdms_err = v_opnwk(nominate,&dev_handle,savary);
    /* nominate the device */
    xy[1] = pto32k(10); xy[3] = pto32k(80);
    /* set the constants for the grid */
    for (i=50;i<=80;i+=15){
        xy[0] = pto32k(i); xy[2] = xy[0];
        /* set variable elements in array for grid */
        gdms_err = v_pline(dev_handle,2,xy);
        /* draw the line */
    }

    xheight = vst_height(dev_handle,pto32k(4),&xwidth,&cwidth,&cheight);
    /* set character height */
    gdms_err = vst_alignment(dev_handle,1,2,&i,&j);
    /* set text alignment */
    j = 0;
    /* Index into tick labels */
    for (i=43;i<=88;i+=15){
        gdms_err = v_gtext(dev_handle,pto32k(i),pto32k(10),y_ticks[ j++]);
        /* write text */
    }

    gdms_err = vst_alignment(dev_handle,2,1,&i,&j);
    /* set text alignment */
```

```

j = 0;
    /* index into y axis labels */
    for (i=15;i<=75;i+=10){
        gdms_err = v_gtext(dev_handle,pto32k(33),(short) (pto32k(i)-xheight/2.0),
            tasks[ j++ ]);
        /* write out text */
    }
    gdms_err = vst_alignment(dev_handle,0,0,&i,&j);
    /* set text alignment */
    gdms_err = v_gtext(dev_handle,pto32k(35),pto32k(82),y_label);
    /* write out the y axis label */
    gdms_err = vst_height(dev_handle,pto32k(9),&xwidth,&cwidth,&cheight);
    /* set new character height */
    gdms_err = v_gtext(dev_handle,pto32k(35),pto32k(88),title);
    /* write out title text */

    gdms_err = vsf_style(dev_handle,2);
    gdms_err = vsf_interior(dev_handle,3);
    /* set fill pattern */
    j = 0;
    /* set index into data arrays */
    for (i=12;i<=72;i+=10) {
        xy[1] = pto32k(i); xy[3] = pto32k(i+6);
        xy[0] = pto32k(start_dates[ j]);
        xy[2] = pto32k(end_dates[ j++]);
        /* set dimensions for bars */
        gdms_err = v_bar(dev_handle,xy);
        /* draw the bars */
    }

    box(35,95,10,80,xy);
    /* create box points */
    gdms_err = v_pline(dev_handle,5,xy);
    /* draw frame around chart */
    box(0,100,0,100,xy);
    /* create page border */
    gdms_err = v_pline(dev_handle,5,xy);
    /* draw page border */
    gdms_err = vrq_string(dev_handle,2,0,echo_xy,istring);
    /* wait for <CR> */
    gdms_err = v_clswk(dev_handle);
}

box(xmin,xmax,ymin,ymax,xyout)
short xmin, xmax, ymin, ymax, xyout[ ];
{
    extern short pto32k();
    xyout[0] = pto32k(xmin); xyout[1] = pto32k(ymin);
    xyout[2] = pto32k(xmax); xyout[3] = xyout[1];
    xyout[4] = xyout[2]; xyout[5] = pto32k(ymax);
    xyout[6] = xyout[0]; xyout[7] = xyout[5];
    xyout[8] = xyout[0]; xyout[9] = xyout[1];
}

short pto32k(percent)
short percent;
{
    return((float) percent / 100.0 * 32767);
}

```



Error Codes

This appendix lists the error codes returned by GSS-DRIVERS functions.

GSS-DRIVERS functions always return to the caller whether or not the requested operation was successful. Each function returns an error status that indicates the results of the request. In this way the application program is made aware of the condition of the graphics subsystem and can take appropriate action without losing control of the system. This also places a responsibility for checking error status and attempting error recovery, or at least informing the user, on the applications program.

No error messages are displayed on the system console by GSS-DRIVERS.

Error codes contain both generic and system-dependent information based on the following conventions:

- A negative return from a function always implies an error. An error return greater than or equal to zero indicates that no error occurred.
- The error code has two parts: generic and system-dependent.
- The generic part of the code is in the form -XX00.
- The system-dependent part of the code is in the form 00YY.

For example error code -1405 indicates:

- 1400 => error in fd delete error
- 0005 => access denied

TABLE B-1 Generic Codes

<u>Code</u>	<u>Cause</u>
-100	VDI SWAP DRIVER ERROR An error has been encountered while swapping device drivers. The proper driver may not be resident in the specified directory area or the driver name may not be included in the system environment.
-200	FD CONNECT ERROR Error during the file connect operation.
-300	FD DISCONNECT ERROR Error during the file disconnect operation.
-400	FD COPY ERROR Error during the copy file descriptor operation.
-500	FD OPEN ERROR Error while opening a file.
-600	FD CLOSE ERROR Error while closing a file.
-700	FD READ ERROR Error while reading a file in wait mode.
-800	FD WRITE ERROR Error while writing to a file in wait mode.
-900	FDP READ ERROR Error while reading file in proceed mode.
-1000	FDP WRITE ERROR Error while writing to a file in proceed mode.
-1100	FD INQUIRE ERROR Error while inquiring about file status.
-1200	FD SEEK ERROR Error during seek operation.
-1300	FD SIZE ERROR Error while inquiring current file size.
-1400	FD DELETE ERROR Error during file delete operation.

(continued)

TABLE B-1 (continued)

Code	Cause
-1500	FD RENAME ERROR Error while renaming a file.
-1600	FD DIRECTORY ERROR Error during directory list operation.
-1700	FD PARSE ERROR Error while parsing filename.
-1800	EM DASH ERROR Error during dash output.
-1900	EM MARKER ERROR Error during marker output.
-2000	EM ALIGNMENT ERROR Error during text alignment output.
-2100	EM POLYGON ERROR Error during polygon output.
-2200	EM BAR ERROR Error during bar output.
-2300	EM ARC ERROR Error during arc output.
-2400	EM PIE ERROR Error during pie output.
-2500	EM CIRCLE ERROR Error during circle output.
-2600	GIN INIT ERROR Error initializing input device.
-2700	GIN POINT ERROR Error moving cursor during GIN.
-2800	GIN TERMINATOR Error terminating GIN function.
-3000	DEVICE DRIVER ERROR Error while calling device driver.
-5000	NOT CAPABLE ERROR Specified device is not capable of requested function.

TABLE B-2 Specific Codes

<u>Code</u>	<u>Cause</u>
-1	INVALID FUNCTION Operating system error—invalid function code.
-2	FILE NOT FOUND Operating system error—named file not found.
-3	PATH NOT FOUND Operating system error—named path not found.
-4	TOO MANY OPEN FILES Operating system error—too many files open.
-5	ACCESS DENIED Operating system error. Named file not accessible (protected).
-6	INVALID HANDLE Operating system error—invalid device handle.
-8	INSUFFICIENT MEMORY Operating system error. Insufficient memory for requested operation.
-16	RM CURRENT DIRECTORY Operating system error. Attempt to remove the current directory.
-18	NO MORE FILES Operating system error. No more space on device.
-78	CANT DELETE DIRECTORY Attempted to delete a directory.
-79	DEVICE BUSY Device currently in use.
-80	HARDWARE NOT PRESENT Device or hardware not present.
-81	ALL SLOTS USED Cannot open another UNIX driver.
-82	VDI CANT START Cannot get VDI going.

(continued)

TABLE B-2 (continued)

<u>Code</u>	<u>Cause</u>
-84	CANT START DRIVER Fork of driver failed.
-85	EOF ERROR End of file found.
-86	ALREADY OPEN Open Workstation attempted when workstation already open.
-88	CANNOT START READ FORK Reader not executed.
-89	NO DRIVER FILE Filename not found.
-90	UNKNOWN DRIVER Driver file does not exist.
-92	ILLEGAL HANDLE Illegal device handle.
-94	FD ILLEGAL OFFSET Illegal byte offset within file.
-96	FD NOT CONNECTED The file description is not connected.
-97	FD FILE IS OPEN File currently open.
-98	FD NOT AVAILABLE File not available.
-99	FD NOT DIRECTORY FILE File not within current directory.



Device-Dependent Information

This appendix contains information about specific graphics peripheral devices that are supported by GSS-DRIVERS. It describes the capabilities and limitations of devices, the index assignments for selectable functions such as polymarker types and color, and other special information.

Specific information is provided for the following devices:

AT&T Model 470 Graphics Printer	C-4
AT&T Model 455 Letter Quality Printer	C-8
AT&T UNIX PC Model 7300 Display	C-12
CIT-101e Terminal with CIG-101e Graphics Card	C-17
CIT-161 Terminal with CIG-267 Graphics Card	C-22
Diablo C150 Ink Jet Color Printer	C-27
Epson MX-100 Printer	C-31
Epson MX-80 Printer with Graphtrax Plus	C-35
Hewlett-Packard 7470A Plotter	C-39
Hewlett-Packard 7475A Plotter	C-43
HILOT DMP-29 Plotter	C-47
NEC Model 7730 Letter Quality Printer	C-51
NEC Model 3550 Letter Quality Printer	C-55
Nicolet Zeta Corporation Zeta 8 Plotter	C-59
Okidata Microline 93 and 84 Printer with Step 2 Support	C-63
Okidata Microline 92 Printer	C-67
Printronix Printers	C-71
Strobe Plotters	C-75
Summagraphics Summatablet	C-79
Tektronix 4105 Color Graphics Terminal	C-81
Virtual Device Metafile (VDM)	C-87
VT100 Terminal with Retro-Graphics Card	C-92

Device Information Categories

The following information is given for each graphics devices:

File Name	The name of the device driver file.								
Device Type	The logical name of the device. This name must be included as the workstation identifier in the Open Workstation call that initializes the device. It is also used when setting your graphics environment to redefine the logical to physical assignments.								
Communications	Specific information about the data interface to the device such as flagging protocols and strapping options.								
Features Supported	Describes the features contained within the VDI Specification that are supported on this device. Subcategories include: <table><tr><td>Polylines</td><td>Describes the available line styles and their index numbers.</td></tr><tr><td>Graphics Text</td><td>Describes the graphics text capabilities of the device such as character size and rotation and their index numbers.</td></tr><tr><td>Graphics Markers</td><td>Describe the marker symbols available, their size variations and their index numbers.</td></tr><tr><td>Filled Areas</td><td>Describes the fill styles available for area fills and their corresponding indices.</td></tr></table>	Polylines	Describes the available line styles and their index numbers.	Graphics Text	Describes the graphics text capabilities of the device such as character size and rotation and their index numbers.	Graphics Markers	Describe the marker symbols available, their size variations and their index numbers.	Filled Areas	Describes the fill styles available for area fills and their corresponding indices.
Polylines	Describes the available line styles and their index numbers.								
Graphics Text	Describes the graphics text capabilities of the device such as character size and rotation and their index numbers.								
Graphics Markers	Describe the marker symbols available, their size variations and their index numbers.								
Filled Areas	Describes the fill styles available for area fills and their corresponding indices.								
Color	Describes the color capabilities of the device, default colors, color selection, color indices and color index to pen correspondence in the case of plotters.								
Graphics Input	Specific information about the operation of graphics input such as the operator action required to initiate and terminate input.								
Alpha Text	Describes the text capabilities of the device such as fonts, sizes, color and other								

attributes. On some devices alpha text is distinct from graphics text.

Request Locator	Describes how the Request Locator function is implemented on the device.
Request String	Describes how the Request String function is implemented on the device.
Raster Writing	Describes pixel (pel) operations.
Cursor Addressable Text	Lists the attributes available to cursor addressable text on the device. Cursor text attributes include reverse video, underline, blink, bold, and color.
Special Information	Contains information unique to the device which is not described under another heading.
VDI Opcode Summary	Lists the VDI operations that are supported by the device.

AT&T Model 470 Graphics Printer

File Name	att470	
Device Type	PRINTER	
Communications	Parallel The standard Centronics parallel cable will operate the Model 470 printer properly.	
Features Supported	Polylines	Lines and arcs can be drawn with the seven line styles listed below: 1 _____ Solid 2 _____ Long Dash 3 Dotted 4 _____ . _____ Dash Dotted 5 _____ Medium Dashed 6 _____ . . _____ Dash Two Dots 7 _____ Short Dash
	Graphics Text	This printer supports twelve character sizes with a Preserve Aspect Ratio and twelve sizes in a Non-preserve Aspect Ratio. Note that these graphics text characters can be rotated on 0, 90, 180, and 270 degree base lines.
	Graphic Markers	The driver supports six kinds of Graphic Markers. Each marker has five possible sizes in Preserve Aspect Ratio and five sizes in Non-preserve Aspect Ratio. The six kinds of markers are shown in the following list: 1 = Dot 2 = Cross 3 = Star 4 = Circle 5 = X 6 = Diamond

Filled Areas Filled areas, bars, pie slices and circles are all displayed using the current fill area attributes of color, interior style, and style index. The interior styles of HOLLOW, SOLID, and HATCH are all supported by this device. The interior style of PATTERN is mapped to the HATCH interior style. Six different styles are supported for the HATCH interior style:

- 1 = Narrow diagonal
- 2 = Medium diagonal
- 3 = Wide diagonal
- 4 = Narrow crosshatch
- 5 = Medium crosshatch
- 6 = Wide crosshatch

Color The Model 470 printer supports two colors: Index 1 is displayed in black ink and index 0 is not displayed. These colors can not be redefined.

Alpha Text Alpha text can be positioned anywhere on the output page. The following text capabilities are available on the Model 470 printer:

Fonts: 1 = Normal (default)
2 = Bold
4 = Proportional spaced
5 = Bold proportional spaced

Sizes: 1 = 17 characters per inch
2 = 12 characters per inch
3 = 10 characters per inch
4 = 8.5 characters per inch
5 = 6 characters per inch
6 = 5 characters per inch

Note

Text size is ignored when selecting a proportional font.

VDI Opcode Summary

Superscript and Subscript

Underline

Line Spacing

Overstrike Mode

Pass Through Mode

The GSS VDI functions available for the Model 470 driver include:

Clear Workstation—Causes picture generation; advances paper

Close Workstation—Causes picture generation; resets printer to default state

Open Workstation —Initializes printer for output

Set Alpha Text Position

Set Writing Mode—All sixteen Boolean writing modes are supported

Update Workstation—Causes picture generation; advances paper; subsequent output is overlaid

Primitives

Arc (uses polyline attributes)

Bar (uses filled area attributes)

Cell Array—Outlines the area, no other action

Circle (uses filled area attributes)

Filled Area

Graphics Text

Output Alpha Text

Output Cursor Addressable Text—Transmits string to printer immediately

Pie Slice (uses filled area attributes)

Polyline

Polymarker

Attributes

Set Alpha Text Color

Set Alpha Text Font and Size

Set Alpha Text Line Spacing

Set Alpha Text Overstrike Mode

Set Alpha Text Pass Through Mode

Set Alpha Text Quality—Returns default value
Set Alpha Text Script Mode
Set Alpha Text Underline Mode
Set Graphics Text String Baseline Rotation
Set Character Height
Set Color Representation—Returns default settings
Set Fill Color Index
Set Fill Interior Style
Set Fill Style Index
Set Graphics Text Alignment
Set Graphics Text Color Index
Set Graphics Text Font—Returns default setting
Set Polyline Color Index
Set Polyline Type
Set Polyline Width—Returns default setting
Set Polymarker Type
Set Polymarker Scale
Set Polymarker Color Index

Inquiries

Inquire Alpha Text Capabilities
Inquire Alpha Cell Location
Inquire Alpha Font Availability
Inquire Alpha Text Position
Inquire Alpha String Extent
Inquire Addressable Character Cells—Will return (-1, -1) to indicate no cursor addressing functions
Inquire Color Representation
Inquire Current Fill Area Attributes
Inquire Current Graphics Text Attributes
Inquire Current Polyline Attributes
Inquire Current Polymarker Attributes

AT&T Model 455 Letter Quality Printer

File Name	att455		
Device Type	PRINTER		
Communications	Parallel A Centronics standard parallel cable will operate the AT&T 455 printer properly.		
Features Supported	Polylines	Lines and arcs can be drawn with the seven line styles listed below:	
	1	_____	Solid
	2	_____	Long Dash
	3		Dotted
	4	_____ . _____	Dash Dotted
	5	____ _	Medium Dashed
	6	____ _ . . _____	Dash Two Dots
	7	- - - - -	Short Dash
Graphics Text	This printer supports twelve character sizes with a Preserve Aspect Ratio and twelve sizes in a Non-preserve Aspect Ratio. Note that these graphics text characters can be rotated on 0, 90, 180, and 270 degree base lines.		
Graphic Markers	The driver supports six kinds of Graphic Markers. Each marker has five possible sizes in Preserve Aspect Ratio and five sizes in Non-preserve Aspect Ratio. The six kinds of markers are shown in the following list:		
	1	= Dot	
	2	= Cross	
	3	= Star	
	4	= Circle	
	5	= X	
	6	= Diamond	

Filled Areas Filled areas, bars, pie slices and circles are all displayed using the current fill area attributes of color, interior style, and style index. The interior styles of HOLLOW, SOLID, and HATCH are all supported by this device. The interior style of PATTERN is mapped to the HATCH interior style. Six different styles are supported for the HATCH interior style:

- 1 = Narrow diagonal
- 2 = Medium diagonal
- 3 = Wide diagonal
- 4 = Narrow crosshatch
- 5 = Medium crosshatch
- 6 = Wide crosshatch

Color The AT&T Model 455 printer supports two colors. Index 1 is displayed with the black ribbon and index 0 is not displayed. These colors cannot be redefined.

Alpha Text Alpha text can be positioned anywhere on the output page. The following text capabilities are available on the AT&T 455 printer:

Fonts: 1 = Normal (default)
2 = Bold
4 = Proportionally spaced
5 = Bold proportionally spaced

Sizes: One size supported.
10 cpi print wheel for constant spacing, or a WPS print wheel for proportional spacing.

Superscript and Subscript

Underline

Overstrike Mode

Pass Through Mode

Alpha Text Quality:

<50 : Turn on bi-direction printing

≥50 : Turn off bi-direction printing

Special Information

Prior to running an application, the constant spacing print wheel must be installed. When a proportionally spaced alpha text font is desired, the user will be prompted to place the proportional spacing print wheel into the AT&T 455. If the alpha text font is later changed back to a constant spacing font, the user will be prompted to insert it back into the printer. Note that changing the print wheel will cause the READY light to blink. Press the PAUSE button to reactivate the printer.

VDI Opcode Summary

The functions available in the GSS VDI for the AT&T 455 printer are:

Clear Workstation—Causes picture generation and advances paper

Close Workstation—Causes picture generation

Open Workstation—Advances the paper

Set Alpha Text Position

Set Writing Mode—Mode 4 (replace) and mode 8 (overstrike) are supported

Update Workstation—Causes picture generation; subsequent output is overlaid

Output

Arc—uses polyline attributes

Bar—uses filled area attributes

Cell Array—Outlines the area, no other action

Circle—uses filled area attributes

Filled Area

Graphics Text

Output Alpha Text

Output Cursor Addressable Text—Transmits string to printer immediately

Pie Slice—uses filled area attributes

Polyline

Polymarker

Attributes

Set Alpha Text Color
Set Alpha Text Font and Size
Set Alpha Text Line Spacing
Set Alpha Text Overstrike Mode
Set Alpha Text Pass Through Mode
Set Alpha Text Quality
Set Alpha Text Script Mode
Set Alpha Text Underline Mode
Set Graphics Text String Baseline Rotation
Set Character Height
Set Color Representation—Returns default settings
Set Fill Color Index
Set Fill Interior Style
Set Fill Style Index
Set Graphics Text Alignment
Set Graphics Text Color Index
Set Graphics Text Font—Returns default setting
Set Polyline Color Index
Set Polyline Type
Set Polyline Width—Returns default setting
Set Polymarker Type
Set Polymarker Scale
Set Polymarker Color Index

Inquiries

Inquire Alpha Text Capabilities
Inquire Alpha Cell Location
Inquire Alpha Font Availability
Inquire Alpha Text Position
Inquire Alpha String Extent
Inquire Addressable Character Cells—Will return (-1, -1) to indicate no cursor addressing functions
Inquire Color Representation
Inquire Current Fill Area Attributes
Inquire Current Graphics Text Attributes
Inquire Current Polyline Attributes
Inquire Current Polymarker Attributes

AT&T UNIX PC Model 7300 Display

File Name	pc7300	
Device Type	DISPLAY	
Features Supported	Polylines	Lines and arcs can be drawn with the seven line styles listed below: 1 2 3 4 5 6 7 Solid Long Dash Dotted Dash Dotted Medium Dashed Dash Two Dots Short Dash
	Graphics Text	This printer supports five character sizes with a Preserve Aspect Ratio and five sizes in a Non-preserve Aspect Ratio. Note that these graphics text characters can be rotated on 0, 90, 180, and 270 degree base lines.
	Graphic Markers	The driver supports six kinds of Graphic Markers. Each marker has five possible sizes in Preserve Aspect and five sizes in Non-preserve Aspect Ratio. The six kinds of markers are shown in the following list: 1 = Dot 2 = Cross 3 = Star 4 = Circle 5 = X 6 = Diamond
	Filled Areas	Filled areas, bars, pie slices and circles are all displayed using the current fill area attributes of color, interior style, and style index. The interior styles of HOLLOW, SOLID, and HATCH

are all supported by this device. The interior style of PATTERN is mapped to the HATCH interior style. Six different styles are supported for the HATCH interior style:

- 1 = Narrow diagonal
- 2 = Medium diagonal
- 3 = Wide diagonal
- 4 = Narrow crosshatch
- 5 = Medium crosshatch
- 6 = Wide crosshatch

Color

In graphics mode, there are two available colors that can be used to display graphic primitives. Color index 0 is displayed as Black, and color index 1 is displayed as white. These colors can not be redefined.

Request Locator

When locator is invoked on the UNIX PC, a tracking cross appears on the display at the initial locator position. The cross can be moved by moving the attached mouse. When the cross is at the desired location, the point can be selected by pressing the left mouse button. This causes the coordinates of the point to be transmitted back to the application program.

If desired, the device will perform an inking function. When the locator is terminated, a line from the initial position to the desired position is drawn honoring the current line attributes such as color and line style.

Also the device performs rubberbanding if desired. There are two types of rubberbanding supported: lines and rectangles. If rubberbanding lines are desired, a line will be drawn from the initial locator position to the current position of the graphic cursor. The line changes dynamically as the cursor is moved. When the locator is terminated, the line is removed.

	If rubberband rectangle is specified, a rectangle is displayed with one corner at the initial locator position and the opposite at the current position of the graphic cursor. The rectangle changes dynamically as the cursor is moved. When the locator is terminated, the rectangle is removed from the display.
Request Choice	The function keys (F1) to (F8) on the keyboard are used to enter choice input.
Request String	The keyboard is used to enter strings. The string is terminated by the Return key.
Cursor Addressable Text	Cursor addressable text is supported. The device Addressable Text must be in Cursor Addressing Mode before it can perform any cursor control functions. To display graphic primitives, the device must be removed from Cursor Addressing Mode.
Alpha Text	Alpha text can be positioned anywhere on the output page. The following text capabilities are available on the UNIX PC Display: One Font One Size Superscript and Subscript Underline Line spacing
VDI Opcode Summary	The functions available in the GSS VDI for the AT&T UNIX PC Model 7300 driver are: Clear Workstation Close Workstation Cursor Down Cursor Left Cursor Right Cursor Up Direct Cursor Address Enter Cursor Addressing Mode Erase to End of Line Erase to End of Screen

Exit Cursor Addressing Mode
Home Cursor
Open Workstation
Graphic Input Cursor
Set Alpha Text Position
Set Input Mode
Set Line Edit Characters
Set Writing Mode—All sixteen Boolean
writing modes are supported
Update Workstation—No action is performed

Primitives

Arc (uses polyline attributes)
Bar (uses filled area attributes)
Cell Array
Circle (uses filled area attributes)
Filled Area
Graphics Text
Output Alpha Text
Output Cursor Addressable Text
Pie Slice (uses filled area attributes)
Polyline
Polymarker

Attributes

Reverse Video Off
Reverse Video On
Set Alpha Text Color
Set Alpha Text Font and Size
Set Alpha Text Line Spacing
Set Alpha Text Overstrike Mode
Set Alpha Text Pass Through Mode—Returns
default value
Set Alpha Text Quality—Returns default
value
Set Alpha Text Script Mode
Set Alpha Text Underline Mode
Set Background Color Index
Set Graphics Text String Baseline Rotation
Set Character Height
Set Color Representation—Returns default
settings
Set Cursor Text Attributes
Set Cursor Text Color Indices—Only the
foreground color can be selected

Set Fill Color Index
Set Fill Interior Style
Set Fill Style Index
Set Graphics Text Alignment
Set Graphics Text Color Index
Set Graphics Text Font—Returns default
setting
Set Polyline Color Index
Set Polyline Type
Set Polyline Width—Returns default setting
Set Polymarker Type
Set Polymarker Scale
Set Polymarker Color Index

Input

Input Locator—Request Mode
Input Choice—Request Mode
Input String—Request Mode
Read Cursor Movement Keys

Inquiries

Inquire Alpha Text Capabilities
Inquire Alpha Cell Location
Inquire Alpha Font Availability
Inquire Alpha Text Position
Inquire Alpha String Extent
Inquire Addressable Character Cells
Inquire Color Representation
Inquire Cell Array
Inquire Current Cursor Address
Inquire Current Fill Area Attributes
Inquire Current Graphics Text Attributes
Inquire Current Polyline Attributes
Inquire Current Polymarker Attributes

CIT-101e Terminal with CIG-101e Graphics Card

File Name	cit101	
Device Type	DISPLAY	
Communications	Serial (RS-232)	
Features Supported	Polylines	Lines and arcs can be drawn with the seven line styles listed below: 1 _____ Solid 2 _____ Long Dash 3 Dotted 4 _____ . _____ Dash Dotted 5 _____ Medium Dashed 6 _____ . . _____ Dash Two Dots 7 _____ Short Dash
	Graphics Text	This device driver supports four character sizes with a Preserve Aspect Ratio and four sizes in a Non-preserve Aspect Ratio. Note that these graphics text characters cannot be rotated.
	Graphic Markers	The driver supports six kinds of Graphic Markers. Each marker has five possible sizes in Preserve Aspect Ratio and five sizes in Non-preserve Aspect Ratio. The six kinds of markers are shown in the following list: 1 = Dot 2 = Cross 3 = Star 4 = Circle 5 = X 6 = Diamond
	Filled Areas	Filled areas, bars, pie slices and circles are all displayed using the current fill area attributes of color, interior style, and style

index. The interior styles of HOLLOW, SOLID, and HATCH are all supported by this device. The interior style of PATTERN is mapped to the HATCH interior style. Six different styles are supported for the HATCH interior style:

- 1 = Narrow diagonal
- 2 = Medium diagonal
- 3 = Wide diagonal
- 4 = Narrow crosshatch
- 5 = Medium crosshatch
- 6 = Wide crosshatch

Color	The CIG-101e graphics card supports two simultaneously displayable colors. These colors cannot be remapped.
Request Locator	When locator is invoked, a tracking cross appears on the display at the initial locator position. The cross can be moved by pressing one of the four arrow keys on the keyboard. Initially, the cross moves in small increments. Holding an arrow key down, will cause the cross to move faster after a slight delay. When the cross is at the desired location, that point can be selected by pressing any alpha key on the keyboard. This causes the coordinates of point to be transmitted back to the user program. If desired, the device will perform an inking function. When the locator is terminated, a line from the initial position to the desired position is drawn honoring the current line attributes such as color and line style.
Request Choice	The numeric keys 0 to 9 on the keypad are used to enter choice input.
Request String	The keyboard is used to enter strings. All strings are terminated by the Return key.



Cursor Addressable Text

Cursor addressable text is supported. The terminal must be in Cursor Addressing Mode before it can perform any cursor control functions. The following attributes are supported:

Reverse Video

Blink

Underline

To display graphics primitives, the device must be removed from Cursor Addressing Mode.

Alpha Text

Alpha text can be positioned anywhere on the output page. The following text capabilities are available:

One Font

One Size



Superscript and Subscript

Underline

Line spacing

VDI Opcode Summary

The functions available in the GSS VDI for the Cit101 driver are:

Clear Workstation

Close Workstation

Cursor Down

Cursor Left

Cursor Right

Cursor Up

Direct Cursor Address

Enter Cursor Addressing Mode

Erase to End of Line

Erase to End of Screen

Exit Cursor Addressing Mode



Home Cursor

Open Workstation

Remove Graphics Input Cursor

Set Alpha Text Position

Set Input Mode

Set Line Edit Characters

Set Writing Mode—Modes supported include:

1—Clear (dots off)

7—XOR

8—Overstrike (OR) Update

Workstation—No action is performed

Primitives

Arc (uses polyline attributes)

Bar (uses filled area attributes)

Cell Array

Circle (uses filled area attributes)

Filled Area

Graphics Text

Output Alpha Text

Output Cursor Addressable Text

Pie Slice (uses filled area attributes)

Polyline

Polymarker

Attributes

Reverse Video Off

Reverse Video On

Set Alpha Text Color

Set Alpha Text Font and Size

Set Alpha Text Line Spacing

Set Alpha Text Overstrike Mode

Set Alpha Text Pass Through Mode—Returns default value

Set Alpha Text Quality—Returns default value

Set Alpha Text Script Mode

Set Alpha Text Underline Mode

Set Background Color Index

Set Graphics Text String Baseline Rotation

Set Character Height

Set Color Representation

Set Cursor Text Attributes

Set Cursor Text Color Indices

Set Fill Color Index

Set Fill Interior Style

Set Fill Style Index

Set Graphics Text Alignment

Set Graphics Text Color Index

Set Graphics Text Font—Returns default setting
Set Polyline Color Index
Set Polyline Type
Set Polyline Width—Returns default setting
Set Polymarker Type
Set Polymarker Scale
Set Polymarker Color Index

Input

Input Locator—Request Mode
Input Choice—Request Mode
Input String—Request Mode
Read Cursor Movement Keys

Inquiries

Inquire Alpha Text Capabilities
Inquire Alpha Cell Location
Inquire Alpha Font Availability
Inquire Alpha Text Position
Inquire Alpha String Extent
Inquire Addressable Character Cells
Inquire Color Representation
Inquire Current Cursor Address
Inquire Current Fill Area Attributes
Inquire Current Graphics Text Attributes
Inquire Current Polyline Attributes
Inquire Current Polymarker Attributes

CIT-161 Terminal with CIG-267 Graphics Card

File Name	cit161	
Device Type	DISPLAY	
Communications	Serial (RS-232)	
Features Supported	Polylines	Lines and arcs can be drawn with the seven line styles listed below: 1 _____ Solid 2 _____ Long Dash 3 Dotted 4 _____ . _____ Dash Dotted 5 _____ Medium Dashed 6 _____ . . _____ Dash Two Dots 7 - - - - - Short Dash
	Graphics Text	This device driver supports four character sizes with a Preserve Aspect Ratio and four sizes in a Non-preserve Aspect Ratio. Note that these graphics text characters cannot be rotated.
	Graphic Markers	The driver supports six kinds of Graphic Markers. Each marker has five possible sizes in Preserve Aspect Ratio and five sizes in Non-preserve Aspect Ratio. The six kinds of markers are shown in the following list: 1 = Dot 2 = Cross 3 = Star 4 = Circle 5 = X 6 = Diamond
	Filled Areas	Filled areas, bars, pie slices and circles are all displayed using the current fill area attributes of color, interior style, and style

index. The interior styles of HOLLOW, SOLID, and HATCH are all supported by this device. The interior style of PATTERN is mapped to the HATCH interior style. Six different styles are supported for the HATCH interior style:

- 1 = Narrow diagonal
- 2 = Medium diagonal
- 3 = Wide diagonal
- 4 = Narrow crosshatch
- 5 = Medium crosshatch
- 6 = Wide crosshatch

Color	The CIG-267 graphics card supports eight simultaneously displayable colors. These colors cannot be remapped.
Request Locator	When locator is invoked, a tracking cross appears on the display at the initial locator position. The cross can be moved by pressing one of the four arrow keys on the keyboard. Initially, the cross moves in small increments. Holding an arrow key down, will cause the cross to move faster after a slight delay. When the cross is at the desired location, that point can be selected by pressing any alpha key on the keyboard. This causes the coordinates of the point to be transmitted back to the user program. If desired, the device will perform an inking function. When the locator is terminated, a line from the initial position to the desired position is drawn honoring the current line attributes such as color and line style.
Request Choice	The numeric keys 0 to 9 on the keypad are used to enter choice input.
Request String	The keyboard is used to enter strings. All strings are terminated by the Return key.

**Cursor
Addressable
Text**

Cursor addressable text is supported. The terminal must be in Cursor Addressing Mode before it can perform any cursor control functions. The following attributes are supported:

Reverse Video

Blink

Underline

Bold Intensity

Colors:

0 = Black

1 = White

2 = Red

3 = Green

4 = Blue

5 = Yellow

6 = Cyan

7 = Magenta

To display graphics primitives, the device must be removed from Cursor Addressing Mode.

Alpha Text

Alpha text can be positioned anywhere on the output page. The following text capabilities are available:

One Font

One Size

Superscript and Subscript

Bold

Underline

Line Spacing

Eight Colors

**VDI Opcode
Summary**

The functions available in the GSS VDI for the cit161 driver are:

Clear Workstation

Close Workstation

Cursor Down

Cursor Left
Cursor Right
Cursor Up
Direct Cursor Address
Enter Cursor Addressing Mode
Erase to End of Line
Erase to End of Screen
Exit Cursor Addressing Mode
Home Cursor
Open Workstation
Remove Graphics Input Cursor
Set Alpha Text Position
Set Input Mode
Set Line Edit Characters
Set Writing Mode—Only REPLACE mode is supported
Update Workstation—No action is performed

Primitives

Arc (uses polyline attributes)
Bar (uses filled area attributes)
Cell Array
Circle (uses filled area attributes)
Filled Area
Graphics Text
Output Alpha Text
Output Cursor Addressable Text
Pie Slice (uses filled area attributes)
Polyline
Polymarker

Attributes

Reverse Video Off
Reverse Video On
Set Alpha Text Color
Set Alpha Text Font and Size
Set Alpha Text Line Spacing
Set Alpha Text Overstrike Mode
Set Alpha Text Pass Through Mode—Returns default value
Set Alpha Text Quality—Returns default value
Set Alpha Text Script Mode
Set Alpha Text Underline Mode
Set Background Color Index

Set Graphics Text String Baseline Rotation
Set Character Height
Set Color Representation
Set Cursor Text Attributes
Set Cursor Text Color Indices
Set Fill Color Index
Set Fill Interior Style
Set Fill Style Index
Set Graphics Text Alignment
Set Graphics Text Color Index
Set Graphics Text Font—Returns default
 setting
Set Polyline Color Index
Set Polyline Type
Set Polyline Width—Returns default setting
Set Polymarker Type
Set Polymarker Scale
Set Polymarker Color Index

Input

Input Locator—Request Mode
Input Choice—Request Mode
Input String—Request Mode
Read Cursor Movement Keys

Inquiries

Inquire Alpha Text Capabilities
Inquire Alpha Cell Location
Inquire Alpha Font Availability
Inquire Alpha Text Position
Inquire Alpha String Extent
Inquire Addressable Character Cells
Inquire Color Representation
Inquire Current Cursor Address
Inquire Current Fill Area Attributes
Inquire Current Graphics Text Attributes
Inquire Current Polyline Attributes
Inquire Current Polymarker Attributes

Diablo C150 Ink Jet Color Printer

File Name diab150

Device Type PRINTER

Communications Parallel
A Centronics standard parallel cable will operate the Diablo printer properly.

Features Supported Polylines Lines and arcs can be drawn with the seven line styles listed below:

- | | | |
|---|-----------------|---------------|
| 1 | _____ | Solid |
| 2 | _____ | Long Dash |
| 3 | | Dotted |
| 4 | _____ . _____ | Dash Dotted |
| 5 | ____ _ | Medium Dashed |
| 6 | _____ . . _____ | Dash Two Dots |
| 7 | __ _ | Short Dash |

Graphics Text This printer supports twelve character sizes with a Preserve Aspect Ratio and twelve sizes in a Non-preserve Aspect Ratio. Note that these graphics text characters can be rotated on 0, 90, 180, and 270 degree base lines.

Graphic Markers The driver supports six kinds of Graphic Markers. Each marker has five possible sizes in Preserve Aspect Ratio and five sizes in Non-preserve Aspect Ratio. The six kinds of markers are shown in the following list:

- 1 = Dot
- 2 = Cross
- 3 = Star
- 4 = Circle
- 5 = X
- 6 = Diamond

Filled Areas Filled areas, bars, pie slices and circles are all displayed using the current fill area attributes of color, interior style, and style index. The interior styles of HOLLOW, SOLID, and HATCH are all supported by this device. The interior style of PATTERN is mapped to the HATCH interior style. Six different styles are supported for the HATCH interior style:

- 1 = Narrow diagonal
- 2 = Medium diagonal
- 3 = Wide diagonal
- 4 = Narrow crosshatch
- 5 = Medium crosshatch
- 6 = Wide crosshatch

Color The printer supports eight colors. They are defined as follows:

- 0 = Background (not displayed)
- 1 = Black
- 2 = Red
- 3 = Green
- 4 = Blue
- 5 = Yellow
- 6 = Violet
- 7 = Orange

These colors cannot be redefined.

Alpha Text Alpha text can be positioned anywhere on the output page. The following text capabilities are available on the Diablo printer:

Fonts: 1—Normal

Sizes: 1—12 characters per inch

Underline

Overstrike Mode

Pass Through Mode

VDI Opcode Summary

The functions available in the GSS VDI for the Diablo Printer are:

Clear Workstation—Causes picture generation; advances paper
Close Workstation—Causes picture generation
Open Workstation—Advances the paper
Set Alpha Text Position
Set Writing Mode—Replace mode only
Update Workstation—Causes picture generation; subsequent output is overlaid.

Output

Arc (uses polyline attributes)
Bar (uses filled area attributes)
Cell Array—Outlines the area, no other action
Circle (uses filled area attributes)
Filled Area
Graphics Text
Output Alpha Text
Output Cursor Addressable Text—Transmits string to printer immediately
Pie Slice (uses filled area attributes)
Polyline
Polymarker

Attributes

Set Alpha Text Color
Set Alpha Text Font and Size
Set Alpha Text Line Spacing
Set Alpha Text Overstrike Mode
Set Alpha Text Pass Through Mode
Set Alpha Text Quality
Set Alpha Text Script Mode
Set Alpha Text Underline Mode
Set Graphics Text String Baseline Rotation
Set Character Height
Set Color Representation—Returns default settings
Set Fill Color Index
Set Fill Interior Style
Set Fill Style Index

Set Graphics Text Alignment
Set Graphics Text Color Index
Set Graphics Text Font—Returns default
setting
Set Polyline Color Index
Set Polyline Type
Set Polyline Width—Returns default setting
Set Polymarker Type
Set Polymarker Scale
Set Polymarker Color Index

Inquiries

Inquire Alpha Text Capabilities
Inquire Alpha Cell Location
Inquire Alpha Font Availability
Inquire Alpha Text Position
Inquire Alpha String Extent
Inquire Addressable Character Cells—Will
return (-1,-1) to indicate no cursor
addressing functions
Inquire Color Representation
Inquire Current Fill Area Attributes
Inquire Current Graphics Text Attributes
Inquire Current Polyline Attributes
Inquire Current Polymarker Attributes

Epson MX-100

File Name epmx100

Device Type PRINTER

Communications Parallel

A Centronics standard parallel cable will operate the Epson MX-100 printer properly.

Features Supported Polylines Lines and arcs can be drawn with the seven line styles listed below:

- | | | |
|---|-----------------|---------------|
| 1 | _____ | Solid |
| 2 | _____ | Long Dash |
| 3 | | Dotted |
| 4 | _____ . _____ | Dash Dotted |
| 5 | _____ | Medium Dashed |
| 6 | _____ . . _____ | Dash Two Dots |
| 7 | _____ | Short Dash |

Graphics Text This printer supports twelve character sizes with a Preserve Aspect Ratio and twelve sizes in a Non-preserve Aspect Ratio. Note that these graphics text characters can be rotated on 0, 90, 180, and 270 degree base lines.

Graphic Markers The driver supports six kinds of Graphic Markers. Each marker has five possible sizes in Preserve Aspect Ratio and five sizes in Non-preserve Aspect Ratio. The six kinds of markers are shown in the following list:

- 1 = Dot
- 2 = Cross
- 3 = Star
- 4 = Circle
- 5 = X
- 6 = Diamond

Filled Areas	Filled areas, bars, pie slices and circles are all displayed using the current fill area attributes of color, interior style, and style index. The interior styles of HOLLOW, SOLID, and HATCH are all supported by this device. The interior style of PATTERN is mapped to the HATCH interior style. Six different styles are supported for the HATCH interior style: 1 = Narrow diagonal 2 = Medium diagonal 3 = Wide diagonal 4 = Narrow crosshatch 5 = Medium crosshatch 6 = Wide crosshatch
Color	The Epson MX-100 printer supports two colors. Index 1 is displayed with the black ribbon and index 0 is not displayed. These colors cannot be redefined.
Alpha Text	Alpha text can be positioned anywhere on the output page. The following text capabilities are available on the Epson MX-100 printer: Fonts: 1 = Normal (default) 2 = Bold Sizes: 1 = 16.5 characters per inch 2 = 10 characters per inch (default) 3 = 5 characters per inch Superscript and Subscript Underline Overstrike Mode Pass through Mode Alpha Text Quality: <50 : Turn on bi-direction printing ≥50 : Turn off bi-direction printing

VDI Opcode Summary

The functions available in the GSS VDI for the Epson MX-100 printer are:

Clear Workstation—Causes picture generation; advances paper

Close Workstation—Causes picture generation

Open Workstation—Advances the paper

Set Alpha Text Position

Set Writing mode—Mode 4 (replace) and mode 8 (overstrike) are supported

Update Workstation—Causes picture generation subsequent output is overlaid.

Output

Arc (uses polyline attributes)

Bar (uses filled area attributes)

Cell Array—Outlines the area, no other action

Circle (uses filled area attributes)

Filled Area

Graphics Text

Output Alpha Text

Output Cursor Addressable Text—Transmits string to printer immediately

Pie Slice (uses filled area attributes)

Polyline

Polymarker

Attributes

Set Alpha Text Color

Set Alpha Text Font and Size

Set Alpha Text Line Spacing

Set Alpha Text Overstrike Mode

Set Alpha Text Pass Through Mode

Set Alpha Text Quality

Set Alpha Text Script Mode

Set Alpha Text Underline Mode

Set Graphics Text String Baseline Rotation

Set Character Height

Set Color Representation—Returns default settings

Set Fill Color Index

Set Fill Interior Style

Set Fill Style Index

Set Graphics Text Alignment
Set Graphics Text Color Index
Set Graphics Text Font—Returns default
setting
Set Polyline Color Index
Set Polyline Type
Set Polyline Width—Returns default setting
Set Polymarker Type
Set Polymarker Scale
Set Polymarker Color Index

Inquiries

Inquire Alpha Text Capabilities
Inquire Alpha Cell Location
Inquire Alpha Font Availability
Inquire Alpha Text Position
Inquire Alpha String Extent
Inquire Addressable Character Cells—Will
return (-1,-1) to indicate no cursor
addressing functions
Inquire Color Representation
Inquire Current Fill Area Attributes
Inquire Current Graphics Text Attributes
Inquire Current Polyline Attributes
Inquire Current Polymarker Attributes

Epson MX-80 with Graphtrax Plus

File Name	epmx80
Device Type	PRINTER
Communications	Parallel A Centronics standard parallel cable will operate the Epson MX-80 printer properly. Serial (RS-232) To communicate with the Epson serially requires an optional serial interface card to be installed in the Epson.

Features Supported	Polylines Lines and arcs can be drawn with the seven line styles listed below:
--------------------	--

1	_____	Solid
2	_____	Long Dash
3		Dotted
4	_____ . _____	Dash Dotted
5	____ _	Medium Dashed
6	_____ . . _____	Dash Two Dots
7	- - - - -	Short Dash

Graphics Text	This printer supports twelve character sizes with a Preserve Aspect Ratio and twelve sizes in a Non-preserve Aspect Ratio. Note that these graphics text characters can be rotated on 0, 90, 180, and 270 degree base lines.
---------------	--

Graphic Markers	The driver supports six kinds of graphic markers, as shown in the following list:
-----------------	---

- 1 = Dot
- 2 = Cross
- 3 = Star
- 4 = Circle
- 5 = X
- 6 = Diamond

Each marker has five possible sizes in Preserve Aspect Ratio and five sizes in Non-preserve Aspect Ratio.

Filled Areas

Filled areas, bars, pie slices and circles are all displayed using the current fill area attributes of color, interior style, and style index. The interior styles of HOLLOW, SOLID, and HATCH are all supported by this device. The interior style of PATTERN is mapped to the HATCH interior style. Six different styles are supported for the HATCH interior style:

- 1 = Narrow diagonal
- 2 = Medium diagonal
- 3 = Wide diagonal
- 4 = Narrow crosshatch
- 5 = Medium crosshatch
- 6 = Wide crosshatch

Color

The MX-80 printer supports two colors. Index 1 is displayed with the black ribbon and index 0 is not displayed. These colors cannot be redefined.

Alpha Text

Alpha Text can be positioned anywhere on the output page. The following text capabilities are available on the Epson MX-80:

Fonts: 1 = Normal (default)
2 = Bold
3 = Italics

Sizes: 1 = 17.16 characters per inch
2 = 10 characters per inch (default)

Superscript and Subscript

Underline

Overstrike Mode

Pass Through Mode



VDI Opcode Summary

Alpha Text Quality:

<50 : Turn on bi-direction printing

≥50 : Turn off bi-direction printing

The functions available in the GSS VDI for the Epson MX-80 printer are:

Clear Workstation—Causes picture generation; advances paper

Close Workstation—Causes picture generation

Open Workstation

Set Alpha Text Position

Set Writing Mode—Mode 4 (replace) and mode 8 (overstrike) are supported

Update Workstation—Causes picture generation subsequent output is overlaid.

Output

Arc (uses polyline attributes)

Bar (uses filled area attributes)

Cell Array—Outlines the area, no other action

Circle (uses filled area attributes)

Filled Area

Graphics Text

Output Alpha Text

Output Cursor Addressable Text—Transmits string to printer immediately

Pie Slice (uses filled area attributes)

Polyline

Polymarker

Attributes

Set Alpha Text Color

Set Alpha Text Font and Size

Set Alpha Text Line Spacing

Set Alpha Text Overstrike Mode

Set Alpha Text Pass Through Mode

Set Alpha Text Quality

Set Alpha Text Script Mode

Set Alpha Text Underline Mode

Set Graphics Text String Baseline Rotation

Set Character Height

Set Color Representation—Returns default settings

Set Fill Color Index
Set Fill Interior Style
Set Fill Style Index
Set Graphics Text Alignment
Set Graphics Text Color Index
Set Graphics Text Font—Returns default
setting
Set Polyline Color Index
Set Polyline Type
Set Polyline Width—Returns default setting
Set Polymarker Type
Set Polymarker Scale
Set Polymarker Color Index

Inquiries

Inquire Alpha Text Capabilities
Inquire Alpha Cell Location
Inquire Alpha Font Availability
Inquire Alpha Text Position
Inquire Alpha String Extent
Inquire Addressable Character Cells—Will
return (-1,-1) to indicate no cursor
addressing functions
Inquire Color Representation
Inquire Current Fill Area Attributes
Inquire Current Graphics Text Attributes
Inquire Current Polyline Attributes
Inquire Current Polymarker Attributes

Hewlett-Packard 7470A Plotter

File Name	hp7470
Device Type	PLOTTER
Communications	Serial (RS-232) To communicate with the Hewlett Packard 7470A requires that the plotter have the serial interface installed.
Features Supported	Polylines Lines and arcs can be drawn with the six line styles listed below: 1 _____ Solid 2 _____ Long Dash 3 Dotted 4 _____ . Dash Dotted 5 _____ Medium Dashed 6 _____ . . Dash Two Dots Graphics Text The plotter supports continuous character scaling. This text can be rotated in increments of one-tenth of a degree. Graphics Markers The driver supports six kinds of graphic markers, as shown in the following list: 1 = Dot 2 = Cross 3 = Star 4 = Circle 5 = X 6 = Diamond Each marker has five possible sizes in Preserve Aspect Ratio and five sizes in Non-preserve Aspect Ratio. Filled Areas Filled areas, bars, pie slices and circles are all displayed using the current fill area attributes of color, interior style, and style index. The interior styles of

HOLLOW, SOLID, and HATCH are all supported by this device. The interior style of PATTERN is mapped to the HATCH interior style. Six different styles are supported for the HATCH interior style:

- 1 = Narrow diagonal
- 2 = Medium diagonal
- 3 = Wide diagonal
- 4 = Narrow crosshatch
- 5 = Medium crosshatch
- 6 = Wide crosshatch

Color Color 1 always is located in pen station 1. It is assumed to be a black pen. By default, color index 2 is located in pen station two. If the user program request is a color index other than what currently is in pen stations one or two, the user is prompted to insert the requested pen into pen station two. Color 0 is not displayed.

Request Locator The pen holder is used to indicate what point is to input. The pen holder is moved by pressing the position keys on the front panel. When the pen holder is at the desired location, the point can be selected by pressing the Enter button. This causes the coordinates of the point to be transmitted back to the user program.

Alpha Text Alpha text can be positioned anywhere on the output page. The following text capabilities are available on the plotter:

Fonts: 1 = Normal (default)
2 = Bold
3 = Italics

Sizes: 1 (default = 66 characters down page)

Color

Superscript and Subscript

VDI Opcode Summary

Overstrike Mode

Underlining

The functions available in the GSS VDI for the Hewlett-Packard 7470A plotter are:

Clear Workstation—Ask for paper change

Close Workstation

Open Workstation

Set Alpha Text Position

Set Input Mode—Request mode is only supported mode

Set Pen Speed

Set Writing Mode—Overstrike mode is the only supported mode

Update Workstation—No action is performed

Output

Arc (uses polyline attributes)

Bar (uses filled area attributes)

Cell Array—Outlines the area, no other action

Circle (uses filled area attributes)

Filled Area

Graphics Text

Output Alpha Text

Output Cursor Addressable Text—Transmits string to plotter with no error checking

Pie Slice (uses filled area attributes)

Polyline

Polymarker

Attributes

Set Alpha Text Color

Set Alpha Text Font and Size

Set Alpha Text Line Spacing

Set Alpha Text Overstrike Mode

Set Alpha Text Pass Through Mode—Returns default value

Set Alpha Text Quality—Returns default value

Set Alpha Text Script Mode

Set Alpha Text Underline Mode

Set Graphics Text String Baseline Rotation

Set Character Height

Set Color Representation—Returns default settings
Set Fill Color Index
Set Fill Interior Style
Set Fill Style Index
Set Graphics Text Alignment
Set Graphics Text Color Index
Set Graphics Text Font—Five fonts supported
Set Polyline Color Index
Set Polyline Type
Set Polyline Width—Returns default setting
Set Polymarker Type
Set Polymarker Scale
Set Polymarker Color Index

Input

Input Locator—Request Mode

Inquiries

Inquire Alpha Text Capabilities
Inquire Alpha Cell Location
Inquire Alpha Font Availability
Inquire Alpha Text Position
Inquire Alpha String Extent
Inquire Addressable Character Cells—
Returns (-1, -1) to indicate no cursor
addressing functions
Inquire Color Representation
Inquire Current Fill Area Attributes
Inquire Current Graphics Text Attributes
Inquire Current Polyline Attributes
Inquire Current Polymarker Attributes

Hewlett-Packard 7475A Plotter

File Name hp7475

Device Type PLOTTER

Communications Serial (RS-232)
To communicate with the Hewlett Packard 7475A requires that the plotter have the serial interface installed.

Features Supported Polylines Lines and arcs can be drawn with the seven line styles listed below:

- 1 _____ Solid
- 2 _____ Long Dash
- 3 Dotted
- 4 _____ . _____ Dash Dotted
- 5 _____ Medium Dashed
- 6 _____ . . _____ Dash Two Dots
- 7 - - - - - Short Dash

Graphics Text The plotter supports continuous character scaling. This text can be rotated in increments of one-tenth of a degree.

Graphics Markers The driver supports six kinds of graphic markers, as shown in the following list:

- 1 = Dot
- 2 = Cross
- 3 = Star
- 4 = Circle
- 5 = X
- 6 = Diamond

Each marker has five possible sizes in Preserve Aspect Ratio and five sizes in Non-preserve Aspect Ratio for A-size and B-size paper.

Filled Areas	Filled areas, bars, pie slices and circles are all displayed using the current fill area attributes of color, interior style, and style index. The interior styles of HOLLOW, SOLID, and HATCH are all supported by this device. The interior style of PATTERN is mapped to the HATCH interior style. Six different styles are supported for the HATCH interior style: 1 = Narrow diagonal 2 = Medium diagonal 3 = Wide diagonal 4 = Narrow crosshatch 5 = Medium crosshatch 6 = Wide crosshatch
Color	Color indices 1 through 6 are mapped to pen stations 1 through 6 of the plotter. Color 0 is not displayed.
Request Locator	The pen holder is used to indicate what point is to be input. The pen holder is moved by pressing the position keys on the front panel. When the pen holder is at the desired location, the point can be selected by pressing the Enter button. This causes the coordinates of the point to be transmitted back to the user program.
Alpha Text	Alpha text can be positioned anywhere on the output page. The following text capabilities are available on the plotter: - Fonts: 1 = Normal (default) 2 = Bold 3 = Italics Sizes: 1 (default = 66 characters down page) Color Superscript and Subscript

Overstrike Mode

Underlining

Special Information

This driver will drive the plotter in both A and B size modes. The paper size can be selected by setting the rear dip switch or by pressing the Size pushbutton simultaneously with the Enter pushbutton. This should be done prior to running your application.

VDI Opcode Summary

The functions available in the GSS VDI for the Hewlett-Packard 7475A plotter are:

Clear Workstation—Ask for paper change

Close Workstation

Open Workstation

Set Alpha Text Position

Set Input Mode—Request mode is only supported mode

Set Pen Speed

Set Writing Mode—Overstrike mode is the only supported mode

Update Workstation—No action is performed

Output

Arc (uses polyline attributes)

Bar (uses filled area attributes)

Cell Array—Outlines the area, no other action

Circle (uses filled area attributes)

Filled Area

Graphics Text

Output Alpha Text

Output Cursor Addressable Text—Transmits string to plotter with no error checking

Pie Slice (uses filled area attributes)

Polyline

Polymarker

Attributes

Set Alpha Text Color

Set Alpha Text Font and Size

Set Alpha Text Line Spacing

Set Alpha Text Overstrike Mode

Set Alpha Text Pass Through Mode—Returns default value

Set Alpha Text Quality—Returns default value
Set Alpha Text Script Mode
Set Alpha Text Underline Mode
Set Graphics Text String Baseline Rotation
Set Character Height
Set Color Representation
Set Fill Color Index
Set Fill Interior Style
Set Fill Style Index
Set Graphics Text Alignment
Set Graphics Text Color Index
Set Graphics Text Font—Five fonts supported
Set Polyline Color Index
Set Polyline Type
Set Polyline Width—Returns default setting
Set Polymarker Type
Set Polymarker Scale
Set Polymarker Color Index

Input

Input Locator—Request Mode

Inquiries

Inquire Alpha Text Capabilities
Inquire Alpha Cell Location
Inquire Alpha Font Availability
Inquire Alpha Text Position
Inquire Alpha String Extent
Inquire Addressable Character Cells—
Returns (-1, -1) to indicate no cursor
addressing functions
Inquire Color Representation
Inquire Current Fill Area Attributes
Inquire Current Graphics Text Attributes
Inquire Current Polyline Attributes
Inquire Current Polymarker Attributes

Houston Instrument DMP-29 Plotter

File Name hipt29

Device Type PLOTTER

Communications Serial (RS-232)
This interface has a maximum baud rate of 9600. The Input Locator function of the driver will function at a maximum of 1200 baud.

Features Supported Polylines Lines and arcs can be drawn with the six line styles listed below:

- | | | |
|---|-----------|---------------|
| 1 | _____ | Solid |
| 2 | _____ | Long Dash |
| 3 | | Dotted |
| 4 | _____ | Dash Dotted |
| 5 | _____ | Medium Dashed |
| 6 | _____ | Dash Two Dots |

Graphics Text The plotter supports continuous character scaling. This text can be rotated in increments of one-tenth of a degree.

Graphics Markers The driver supports six kinds of graphic markers, as shown in the following list:

- 1 = Dot
- 2 = Cross
- 3 = Star
- 4 = Circle
- 5 = X
- 6 = Diamond

Each marker has five possible sizes in Preserve Aspect Ratio and five sizes in Non-preserve Aspect Ratio.

Filled Areas Filled areas, bars, pie slices and circles are all displayed using the current fill area attributes of color, interior style, and style

index. The interior styles of HOLLOW, SOLID, and HATCH are all supported by this device. The interior style of PATTERN is mapped to the HATCH interior style. Six different styles are supported for the HATCH interior style:

- 1 = Narrow diagonal
- 2 = Medium diagonal
- 3 = Wide diagonal
- 4 = Narrow crosshatch
- 5 = Medium crosshatch
- 6 = Wide crosshatch

Color Color indices 1 through 8 correspond to pen stations 1 through 8 of the plotter. Color 0 is not displayed.

Request Locator The pen holder is used to indicate what point is to be input. To input a point, first the Local button must be pressed so that the plotter can be operated manually. Then the pen holder is moved by pressing the position keys on the front panel. When the pen is at the desired location, the point can be selected by pressing the Report Status button. This causes the coordinates of the point to be transmitted back to the user program. Then the Remote button must be pressed to return control back to the user program.

Alpha Text Alpha text can be positioned anywhere on the output page. The following text capabilities are available on the plotter:

- Fonts: 1 = Normal (default)
2 = Bold
3 = Italics

Sizes: 1 (default = 66 characters down page)

Color

Superscript and Subscript



Special Information

VDI Opcode Summary

Overstrike Mode

Underlining

This driver drives the plotter in large paper mode only.

The functions available in the GSS VDI for the Houston Instrument DMP-29 plotter are:

Clear Workstation—Ask for paper change

Close Workstation

Open Workstation

Set Alpha Text Position

Set Input Mode—Request mode is the only supported mode

Set Pen Speed

Set Writing Mode—Overstrike mode is the only supported mode

Update Workstation—No action is performed

Output

Arc (uses polyline attributes)

Bar (uses filled area attributes)

Cell Array—Outlines the area, no other action

Circle (uses filled area attributes)

Filled Area

Graphics Text

Output Alpha Text

Output Cursor Addressable Text—Transmits string to plotter with no error checking

Pie Slice (uses filled area attributes)

Polyline

Polymarker

Attributes

Set Alpha Text Color

Set Alpha Text Font and Size

Set Alpha Text Line Spacing

Set Alpha Text Overstrike Mode

Set Alpha Text Pass Through Mode—Returns default value

Set Alpha Text Quality- Returns default value

Set Alpha Text Script Mode

Set Alpha Text Underline Mode

Set Graphics Text String Baseline Rotation

Set Character Height
Set Color Representation—Returns default settings
Set Fill Color Index
Set Fill Interior Style
Set Fill Style Index
Set Graphics Text Alignment
Set Graphics Text Color Index
Set Graphics Text Font—Five fonts supported
Set Polyline Color Index
Set Polyline Type
Set Polyline Width—Returns default setting
Set Polymarker Type
Set Polymarker Scale
Set Polymarker Color Index

Input

Input Locator—Request Mode

Inquiries

Inquire Alpha Text capabilities
Inquire Alpha Cell Location
Inquire Alpha Font Availability
Inquire Alpha Text Position
Inquire Alpha String Extent
Inquire Addressable Character Cells—
Returns (-1, -1) to indicate no cursor
addressing functions
Inquire Color Representation
Inquire Current Fill Area Attributes
Inquire Current Graphics Text Attributes
Inquire Current Polyline Attributes

NEC Model 7730 Letter Quality Printer

File Name	nec7730
Device Type	PRINTER
Communications	Parallel A Centronics standard parallel cable will operate the NEC printer properly.
Features Supported	Polylines Lines and arcs can be drawn with the seven line styles listed below: 1 _____ Solid 2 _____ Long Dash 3 Dotted 4 _____ . _____ Dash Dotted 5 _____ Medium Dashed 6 _____ . . _____ Dash Two Dots 7 - - - - - Short Dash
Graphics Text	This printer supports twelve character sizes with a Preserve Aspect Ratio and twelve sizes in a Non-preserve Aspect Ratio. Note that these graphics text characters can be rotated on 0, 90, 180, and 270 degree base lines.
Graphic Markers	The driver supports six kinds of Graphic Markers. Each marker has five possible sizes in Preserve Aspect Ratio and five sizes in Non-preserve Aspect Ratio. The six kinds of markers are shown in the following list: 1 = Dot 2 = Cross 3 = Star 4 = Circle 5 = X 6 = Diamond

Filled Areas	Filled areas, bars, pie slices, and circles are all displayed using the current fill area attributes of color, interior style, and style index. The interior styles of HOLLOW, SOLID, and HATCH are all supported by this device. The interior style of PATTERN is mapped to the HATCH interior style. Six different styles are supported for the HATCH interior style: 1 = Narrow diagonal 2 = Medium diagonal 3 = Wide diagonal 4 = Narrow crosshatch 5 = Medium crosshatch 6 = Wide crosshatch
Color	The NEC printer supports two colors. Index 1 is displayed with the black ribbon and index 0 is not displayed. These colors cannot be redefined.
Alpha Text	Alpha text can be positioned anywhere on the output page. The following text capabilities are available on the NEC printer: Fonts: 1 = Normal (default) 2 = Bold Sizes: 1 = Courier 10 Thimble (10 cpi) Superscript and Subscript Underline Overstrike Mode Pass Through Mode Alpha Text Quality: <50 : Turn on bi-direction printing ≥50 : Turn off bi-direction printing

VDI Opcode Summary

The functions available in the GSS VDI for the NEC 7730 Printer are:

Clear Workstation—Causes picture generation; advances paper
Close Workstation—Causes picture generation
Open Workstation—Advances the paper
Set Alpha Text Position
Set Writing Mode—Mode 4 (replace) and mode 8 (overstrike) are supported
Update Workstation—Causes picture generation; subsequent output is overlaid.

Output

Arc (uses polyline attributes)
Bar (uses filled area attributes)
Cell Array—Outlines the area, no other action
Circle (uses filled area attributes)
Filled Area
Graphics Text
Output Alpha Text
Output Cursor Addressable Text—Transmits string to printer immediately
Pie Slice (uses filled area attributes)
Polyline
Polymarker

Attributes

Set Alpha Text Color
Set Alpha Text Font and Size
Set Alpha Text Line Spacing
Set Alpha Text Overstrike Mode
Set Alpha Text Pass Through Mode
Set Alpha Text Quality
Set Alpha Text Script Mode
Set Alpha Text Underline Mode
Set Graphics Text String Baseline Rotation
Set Character Height
Set Color Representation—Returns default settings
Set Fill Color Index
Set Fill Interior Style

Set Fill Style Index
Set Graphics Text Alignment
Set Graphics Text Color Index
Set Graphics Text Font—Returns default
setting
Set Polyline Color Index
Set Polyline Type
Set Polyline Width—Returns default setting
Set Polymarker Type
Set Polymarker Scale
Set Polymarker Color Index

Inquiries

Inquire Alpha Text Capabilities
Inquire Alpha Cell Location
Inquire Alpha Font Availability
Inquire Alpha Text Position
Inquire Alpha String Extent
Inquire Addressable Character Cells—Will
return (-1, -1) to indicate no cursor
addressing functions
Inquire Color Representation
Inquire Current Fill Area Attributes
Inquire Current Graphics Text Attributes
Inquire Current Polyline Attributes
Inquire Current Polymarker Attributes

NEC Model 3550 Letter Quality Printer

File Name nec3550

Device Type PRINTER

Communications Parallel
A Centronics standard parallel cable will operate the NEC printer properly.

Features Supported Polylines Lines and arcs can be drawn with the seven line styles listed below:

- | | | |
|---|---------------------|---------------|
| 1 | _____ | Solid |
| 2 | _____ | Long Dash |
| 3 | | Dotted |
| 4 | _____ . _____ | Dash Dotted |
| 5 | _____ | Medium Dashed |
| 6 | _____ . . _____ . . | Dash Two Dots |
| 7 | _____ | Short Dash |

Graphics Text This printer supports twelve character sizes with a Preserve Aspect Ratio and twelve sizes in a Non-preserve Aspect Ratio. Note that these graphics text characters can be rotated on 0, 90, 180, and 270 degree base lines.

Graphic Markers The driver supports six kinds of Graphic Markers. Each marker has five possible sizes in Preserve Aspect Ratio and five sizes in Non-preserve Aspect Ratio. The six kinds of markers are shown in the following list:

- 1 = Dot
- 2 = Cross
- 3 = Star
- 4 = Circle
- 5 = X
- 6 = Diamond

Filled Areas Filled areas, bars, pie slices and circles are all displayed using the current fill area attributes of color, interior style, and style index. The interior styles of HOLLOW, SOLID, and HATCH are all supported by this device. The interior style of PATTERN is mapped to the HATCH interior style. Six different styles are supported for the HATCH interior style:

- 1 = Narrow diagonal
- 2 = Medium diagonal
- 3 = Wide diagonal
- 4 = Narrow crosshatch
- 5 = Medium crosshatch
- 6 = Wide crosshatch

Color The NEC printer supports two colors. Index 1 is displayed with the black ribbon and index 0 is not displayed. These colors cannot be redefined.

Alpha Text Alpha text can be positioned anywhere on the output page. The following text capabilities are available on the NEC printer:

Fonts: 1 = Normal (default)
2 = Bold
4 = Proportionally spaced
5 = Bold proportionally spaced

Sizes: One size supported—Courier 10
Thimble (10 cpi, constant spacing) or
Bold PS Thimble (proportional spacing)

Superscript and Subscript

Underline

Overstrike Mode

Pass Through Mode

Alpha Text Quality:

- <50 : Turn on bi-direction printing
- ≥50 : Turn off bi-direction printing

Special Information

When a proportionally spaced alpha text font is desired, the user will be prompted to place the proportional space thimble into the NEC 3550, set the front switches correctly, and turn the power to the printer off and on.

VDI Opcode Summary

The functions available in the GSS VDI for the NEC 3550 Printer are:

Clear Workstation—Causes picture generation; advances paper

Close Workstation—Causes picture generation

Open Workstation—Advances the paper

Set Alpha Text Position

Set Writing Mode—Mode 4 (replace) and mode 8 (overstrike) are supported

Update Workstation—Causes picture generation; subsequent output is overlaid.

Output

Arc (uses polyline attributes)

Bar (uses filled area attributes)

Cell Array—Outlines the area, no other action

Circle (uses filled area attributes)

Filled Area

Graphics Text

Output Alpha Text

Output Cursor Addressable Text—Transmits string to printer immediately

Pie Slice (uses filled area attributes)

Polyline

Polymarker

Attributes

Set Alpha Text Color

Set Alpha Text Font and Size

Set Alpha Text Line Spacing

Set Alpha Text Overstrike Mode

Set Alpha Text Pass Through Mode

Set Alpha Text Quality

Set Alpha Text Script Mode

Set Alpha Text Underline Mode

Set Graphics Text String Baseline Rotation

Set Character Height
Set Color Representation—Returns default settings
Set Fill Color Index
Set Fill Interior Style
Set Fill Style Index
Set Graphics Text Alignment
Set Graphics Text Color Index
Set Graphics Text Font—Returns default setting
Set Polyline Color Index
Set Polyline Type
Set Polyline Width—Returns default setting
Set Polymarker Type
Set Polymarker Scale
Set Polymarker Color Index

Inquiries

Inquire Alpha Text Capabilities
Inquire Alpha Cell Location
Inquire Alpha Font Availability
Inquire Alpha Text Position
Inquire Alpha String Extent
Inquire Addressable Character Cells—Will return (-1, -1) to indicate no cursor addressing functions
Inquire Color Representation
Inquire Current Fill Area Attributes
Inquire Current Graphics Text Attributes
Inquire Current Polyline Attributes
Inquire Current Polymarker Attributes

Nicolet Zeta 8 Plotter

File Name	zeta8	
Device Type	PLOTTER	
Communications	Serial (RS-232) Communication with the Nicolet Zeta 8 plotter is accomplished through the modem port J102.	
Features Supported	Polylines	Lines and arcs can be drawn with the seven line styles listed below: 1 _____ Solid 2 _____ Long Dash 3 Dotted 4 _____ . _____ Dash Dotted 5 _____ Medium Dashed 6 _____ . . _____ Dash Two Dots 7 - - - - - Short Dash
	Graphics Text	The plotter supports continuous character scaling. This text can be rotated in increments of one degree.
	Graphics Markers	The driver supports six kinds of graphic markers, as shown in the following list: 1 = Dot 2 = Cross 3 = Star 4 = Circle 5 = X 6 = Diamond Each marker has five possible sizes in Preserve Aspect Ratio and five sizes in Non-preserve Aspect Ratio.
	Filled Areas	Filled areas, bars, pie slices and circles are all displayed using the current fill area attributes of color, interior style, and style

index. The interior styles of HOLLOW, SOLID, and HATCH are all supported by this device. The interior style of PATTERN is mapped to the HATCH interior style. Six different styles are supported for the HATCH interior style:

- 1 = Narrow diagonal
- 2 = Medium diagonal
- 3 = Wide diagonal
- 4 = Narrow crosshatch
- 5 = Medium crosshatch
- 6 = Wide crosshatch

Color Color indices 1 through 8 are mapped to pen stations 1 through 8 of the plotter. Color 0 is not displayed.

Alpha Text Alpha text can be positioned anywhere on the output page. The following text capabilities are available on the plotter:

Fonts: 1 = Normal (default)
2 = Bold

Sizes: 1 (default = 66 characters down page)

Color

Superscript and Subscript

Overstrike Mode

Underlining

Special Information This driver expects the plotter to be set to use mode 0 (GML/RS-232). The switch settings are shown below.

Switch 1

- 1—ON (plot speed fast)
- 2—OFF (non-correct)
- 3—ON (plot on)
- 4—OFF (non-print)

Switch 2

- 1—ON (Inch resolution)
- 2—ON (High resolution)

- 3—OFF (Monitor echo off)
- 4—OFF (No Line Turnaround Delay)
- 5—OFF (No parity)
- 6—Don't care
- 7—OFF (CTS/RTS with sw1-2 set to OFF)
- 8—Reserved

Switch 3

- 1—OFF (GML mode)
- 2—OFF (GML mode)
- 3—OFF (GML mode)
- 4—OFF (select limit)
- 5—ON (virtual limit)
- 6—OFF (multiple pen)
- 7—OFF (ASA page size)
- 8—OFF (auto-sense pen pressure)

Switch 4

- A—8 (9600 baud)
- B—3 (pen pressure)

VDI Opcode Summary

The functions available in the GSS VDI for the Nicolet Zeta 8 plotter are:

- Clear Workstation—Advance paper
- Close Workstation—Advance paper
- Open Workstation
- Set Alpha Text Position
- Set Input Mode—Request mode is only supported mode
- Set Pen Speed
- Set Writing Mode—Overstrike mode is the only supported mode
- Update Workstation—No action is performed

Output

- Arc (uses polyline attributes)
- Bar (uses filled area attributes)
- Cell Array—Outlines the area, no other action
- Circle (uses filled area attributes)
- Filled Area
- Graphics Text
- Output Alpha Text
- Output Cursor Addressable Text—Transmits string to plotter with no error checking

Pie Slice (uses filled area attributes)

Polyline

Polymarker

Attributes

Set Alpha Text Color

Set Alpha Text Font and Size

Set Alpha Text Line Spacing

Set Alpha Text Overstrike Mode

Set Alpha Text Pass Through Mode—Returns default value

Set Alpha Text Quality—Returns default value

Set Alpha Text Script Mode

Set Alpha Text Underline Mode

Set Graphics Text String Baseline Rotation

Set Character Height

Set Color Representation

Set Fill Color Index

Set Fill Interior Style

Set Fill Style Index

Set Graphics Text Alignment

Set Graphics Text Color Index

Set Graphics Text Font—One font supported

Set Polyline Color Index

Set Polyline Type

Set Polyline Width—Returns default setting

Set Polymarker Type

Set Polymarker Scale

Set Polymarker Color Index

Inquiries

Inquire Alpha Text Capabilities

Inquire Alpha Cell Location

Inquire Alpha Font Availability

Inquire Alpha Text Position

Inquire Alpha String Extent

Inquire Addressable Character Cells—
Returns (-1, -1) to indicate no cursor
addressing functions

Inquire Color Representation

Inquire Current Fill Area Attributes

Inquire Current Graphics Text Attributes

Inquire Current Polyline Attributes

Inquire Current Polymarker Attributes

Okidata Microline 93

Okidata Microline 84 with Step 2 Support

File Name	okid84	
Device Type	PRINTER	
Communications	Parallel	A Centronics standard parallel cable will operate the Okidata printer properly.
Features Supported	Polylines	Lines and arcs can be drawn with the seven line styles listed below: 1 _____ Solid 2 _____ Long Dash 3 Dotted 4 _____ . _____ Dash Dotted 5 _____ Medium Dashed 6 _____ . . _____ Dash Two Dots 7 _____ Short Dash
	Graphics Text	This printer supports twelve character sizes with a Preserve Aspect Ratio and twelve sizes in a Non-preserve Aspect Ratio. Note that these graphics text characters can be rotated on 0, 90, 180, and 270 degree base lines.
	Graphic Markers	The driver supports six kinds of Graphic Markers. Each marker has five possible sizes in Preserve Aspect Ratio and five sizes in Non-preserve Aspect Ratio. The six kinds of markers are shown in the following list: 1 = Dot 2 = Cross 3 = Star 4 = Circle 5 = X 6 = Diamond

Filled Areas Filled areas, bars, pie slices and circles are all displayed using the current fill area attributes of color, interior style, and style index. The interior styles of HOLLOW, SOLID, and HATCH are all supported by this device. The interior style of PATTERN is mapped to the HATCH interior style. Six different styles are supported for the HATCH interior style:

- 1 = Narrow diagonal
- 2 = Medium diagonal
- 3 = Wide diagonal
- 4 = Narrow crosshatch
- 5 = Medium crosshatch
- 6 = Wide crosshatch

Color The Okidata printers support two colors. Index 1 is displayed with the black ribbon and index 0 is not displayed. These colors cannot be redefined.

Alpha Text Alpha text can be positioned anywhere on the output page. The following text capabilities are available on the Okidata printers:

Fonts: 1 = Normal (default)
2 = Bold

Sizes: 1 = 17 characters per inch
2 = 12 characters per inch
3 = 10 characters per inch (default)

Superscript and Subscript

Underline

Overstrike Mode

Pass through Mode

Alpha Text Quality:

<50 : Enable Data Processing Mode

≥50 : Enable Correspondence Quality Mode

VDI Opcode Summary

The functions available in the GSS VDI for the Okidata 93 and 84 printers are:

Clear Workstation—Causes picture generation; advances paper
Close Workstation—Causes picture generation
Open Workstation—Advances paper
Set Alpha Text Position
Set Writing Mode—Mode 4 (replace) and mode 8 (overstrike) are supported
Update Workstation—Causes picture generation; subsequent output is overlaid.

Output

Arc (uses polyline attributes)
Bar (uses filled area attributes)
Cell Array—Outlines the area, no other action
Circle (uses filled area attributes)
Filled Area
Graphics Text
Output Alpha Text
Output Cursor Addressable Text—Transmits string to printer immediately
Pie Slice (uses filled area attributes)
Polyline
Polymarker

Attributes

Set Alpha Text Color
Set Alpha Text Font and Size
Set Alpha Text Line Spacing
Set Alpha Text Overstrike Mode
Set Alpha Text Pass Through Mode
Set Alpha Text Quality
Set Alpha Text Script Mode
Set Alpha Text Underline Mode
Set Graphics Text String Baseline Rotation
Set Character Height
Set Color Representation—Returns default settings
Set Fill Color Index
Set Fill Interior Style

Set Fill Style Index
Set Graphics Text Alignment
Set Graphics Text Color Index
Set Graphics Text Font—Returns default
 setting
Set Polyline Color Index
Set Polyline Type
Set Polyline Width—Returns default setting
Set Polymarker Type
Set Polymarker Scale
Set Polymarker Color Index

Inquiries

Inquire Alpha Text Capabilities
Inquire Alpha Cell Location
Inquire Alpha Font Availability
Inquire Alpha Text Position
Inquire Alpha String Extent
Inquire Addressable Character Cells—Will
 return (-1,-1) to indicate no cursor
 addressing functions
Inquire Color Representation
Inquire Current Fill Area Attributes
Inquire Current Graphics Text Attributes
Inquire Current Polyline Attributes
Inquire Current Polymarker Attributes

Okidata Microline 92

File Name	okid92
Device Type	PRINTER
Communications	Parallel A Centronics standard parallel cable will operate the Okidata printer properly.
Features Supported	Polylines Lines and arcs can be drawn with the seven line styles listed below: 1 _____ Solid 2 _____ Long Dash 3 Dotted 4 _____ . Dash Dotted 5 _____ Medium Dashed 6 _____ . . Dash Two Dots 7 _ _ _ _ _ Short Dash
Graphics Text	This printer supports twelve character sizes with a Preserve Aspect Ratio and twelve sizes in a Non-preserve Aspect Ratio. Note that these graphics text characters can be rotated on 0, 90, 180, and 270 degree base lines.
Graphic Markers	The driver supports six kinds of Graphic Markers. Each marker has five possible sizes in Preserve Aspect Ratio and five sizes in Non-preserve Aspect Ratio. The six kinds of markers are shown in the following list: 1 = Dot 2 = Cross 3 = Star 4 = Circle 5 = X 6 = Diamond

Filled Areas Filled areas, bars, pie slices and circles are all displayed using the current fill area attributes of color, interior style, and style index. The interior styles of HOLLOW, SOLID, and HATCH are all supported by this device. The interior style of PATTERN is mapped to the HATCH interior style. Six different styles are supported for the HATCH interior style:

- 1 = Narrow diagonal
- 2 = Medium diagonal
- 3 = Wide diagonal
- 4 = Narrow crosshatch
- 5 = Medium crosshatch
- 6 = Wide crosshatch

Color The Okidata printer supports two colors. Index 1 is displayed with the black ribbon and index 0 is not displayed. These colors cannot be redefined.

Alpha Text Alpha text can be positioned anywhere on the output page. The following text capabilities are available on the Okidata printer:

Fonts: 1 = Normal (default)
2 = Bold

Sizes: 1 = 17 characters per inch
2 = 12 characters per inch
3 = 10 characters per inch (default)

Superscript and Subscript

Underline

Overstrike Mode

Pass Through Mode

Alpha Text Quality:

- <50 : Enable Data Processing Mode
- ≥50 : Enable Correspondence Quality Mode

VDI Opcode Summary

The functions available in the GSS VDI for the Okidata 92 Printer are:

Clear Workstation—Causes picture generation; advances paper

Close Workstation—Causes picture generation

Open Workstation—Advances the paper

Set Alpha Text Position

Set Writing Mode—Mode 4 (replace) and mode 8 (overstrike) are supported

Update Workstation—Causes picture generation; subsequent output is overlaid.

Output

Arc (uses polyline attributes)

Bar (uses filled area attributes)

Cell Array—Outlines the area, no other action

Circle (uses filled area attributes)

Filled Area

Graphics Text

Output Alpha Text

Output Cursor Addressable Text—Transmits string to printer immediately

Pie Slice (uses filled area attributes)

Polyline

Polymarker

Output

Set Alpha Text Color—Returns default setting (1)

Set Alpha Text Font and Size

Set Alpha Text Line Spacing

Set Alpha Text Overstrike Mode

Set Alpha Text Pass Through Mode

Set Alpha Text Quality

Set Alpha Text Script Mode

Set Alpha Text Underline Mode

Set Graphics Text String Baseline Rotation

Set Character Height

Set Color Representation—Returns default settings

Set Fill Color Index

Set Fill Interior Style
Set Fill Style Index
Set Graphics Text Alignment
Set Graphics Text Color Index
Set Graphics Text Font—Returns default
 setting
Set Polyline Color Index
Set Polyline Type
Set Polyline Width—Returns default setting
Set Polymarker Type
Set Polymarker Scale
Set Polymarker Color Index

Inquiries

Inquire Alpha Text Capabilities
Inquire Alpha Cell Location
Inquire Alpha Font Availability
Inquire Alpha Text Position
Inquire Alpha String Extent
Inquire Addressable Character Cells—Will
 return (-1,-1) to indicate no cursor
 addressing functions
Inquire Color Representation
Inquire Current Fill Area Attributes
Inquire Current Graphics Text Attributes
Inquire Current Polyline Attributes
Inquire Current Polymarker Attributes

Printronix MVP Printer

Printronix P300 Printer

Printronix P600 Printer

File Name prntrx

Device Type PRINTER

Communications Serial (RS-232)
To communicate to the Printronix Printers requires that the printers have an RS-232 serial interface installed.

Features Supported Polylines Lines and arcs can be drawn with the seven line styles listed below:

- | | | |
|---|-----------------|---------------|
| 1 | _____ | Solid |
| 2 | _____ | Long Dash |
| 3 | | Dotted |
| 4 | _____ . _____ | Dash Dotted |
| 5 | _____ _____ | Medium Dashed |
| 6 | _____ . . _____ | Dash Two Dots |
| 7 | _____ - - - - - | Short Dash |

Graphics Text This printer supports twelve character sizes with a Preserve Aspect Ratio and twelve sizes in a Non-preserve Aspect Ratio. Note that these graphics text characters can be rotated on 0, 90, 180, and 270 degree base lines.

Graphic Markers The driver supports six kinds of graphic markers, as shown in the following list:

- 1 = Dot
- 2 = Cross
- 3 = Star
- 4 = Circle
- 5 = X
- 6 = Diamond

Each marker has five possible sizes in Preserve Aspect Ratio

and five sizes in Non-preserve Aspect Ratio.

Filled Areas

Filled areas, bars, pie slices and circles are all displayed using the current fill area attributes of color, interior style, and style index. The interior styles of HOLLOW, SOLID, and HATCH are all supported by this device. The interior style of PATTERN is mapped to the HATCH interior style. Six different styles are supported for the HATCH interior style:

- 1 = Narrow diagonal
- 2 = Medium diagonal
- 3 = Wide diagonal
- 4 = Narrow crosshatch
- 5 = Medium crosshatch
- 6 = Wide crosshatch

Color

The Printronix supports two colors. Index 1 is displayed with the black ink and index 0 is not displayed.

These colors cannot be redefined.

Alpha Text

Alpha text is generated in the Printronix Printer PLOT Mode using a 5 x 7 dot matrix pattern character font. Alpha Text can be positioned anywhere on the output page. The following text capabilities are available on the Printronix printers:

Fonts: 1—Normal (default)

Sizes: 1-10 characters per inch

Superscript and Subscript

Underline

Overstrike Mode

Pass Through Mode

Special Information

This driver allows the user to change the paper width independently. The valid options for paper are WIDE (13 inch output) and NARROW (8 inch output). The default is Narrow paper.

The paper width option can be changed via the ENVIRONMENT capability of UNIX. To change the paper size enter one of the following commands at the UNIX command level:

```
PAPER=WIDE  
export PAPER  
PAPER=NARROW  
export PAPER
```

VDI Opcode Summary

The functions available in the GSS VDI for the Printronix Printers are:

Clear Workstation—Causes picture generation; advances paper

Close Workstation—Causes picture generation

Open Workstation—Advances the paper

Set Alpha Text Position

Writing Mode—All sixteen writing modes are supported

Update Workstation—Causes picture generation; subsequent output is overlaid.

Output

Arc (uses polyline attributes)

Bar (uses filled area attributes)

Cell Array—Outlines the area, no other action

Circle (uses filled area attributes)

Filled Area

Graphics Text

Output Alpha Text

Pie Slice (uses filled area attributes)

Polyline

Polymarker

Attributes

Set Alpha Text Color
Set Alpha Text Font and Size
Set Alpha Text Line Spacing
Set Alpha Text Overstrike Mode
Set Alpha Text Pass Through Mode
Set Alpha Text Script Mode
Set Alpha Text Underline Mode
Set Graphics Text String Baseline Rotation
Set Character Height
Set Color Representation—Returns default settings
Set Fill Color Index
Set Fill Interior Style
Set Fill Style Index
Set Graphics Text Alignment
Set Graphics Text Color Index
Set Graphics Text Font—Returns default setting
Set Polyline Color Index
Set Polyline Type
Set Polyline Width—Returns default setting
Set Polymarker Type
Set Polymarker Scale
Set Polymarker Color Index

Inquiries

Inquire Alpha Text Capabilities
Inquire Alpha Cell Location
Inquire Alpha Font Availability
Inquire Alpha Text Position
Inquire Alpha String Extent
Inquire Addressable Character Cells—Will return (-1,-1) to indicate no cursor addressing functions
Inquire Color Representation
Inquire Current Fill Area Attributes
Inquire Current Graphics Text Attributes
Inquire Current Polyline Attributes
Inquire Current Polymarker Attributes

Strobe Model 100—Single pen Plotter
Strobe Model 200—Single pen Plotter
Strobe Model 260—Eight pen Plotter

File Name	strobe
Device Type	PLOTTER
Communications	Serial (RS-232) Communication with the Strobe plotters requires that the serial interface be installed.
Features Supported	Polylines Lines and arcs can be drawn with the seven line styles listed below: 1 _____ Solid 2 _____ Long Dash 3 Dotted 4 _____ . _____ Dash Dotted 5 _____ Medium Dashed 6 _____ . . _____ Dash Two Dots 7 _ _ _ _ _ Short Dash
Graphics Text	The plotter supports continuous character scaling. This text can be rotated in increments of ninety degrees.
Graphics Markers	The driver supports six kinds of graphic markers, as shown in the following list: 1 = Dot 2 = Cross 3 = Star 4 = Circle 5 = X 6 = Diamond Each marker has five possible sizes in Preserve Aspect Ratio and five sizes in Non-preserve Aspect Ratio.

Filled Areas	Filled areas, bars, pie slices and circles are all displayed using the current fill area attributes of color, interior style, and style index. The interior styles of HOLLOW, SOLID, and HATCH are all supported by this device. The interior style of PATTERN is mapped to the HATCH interior style. Six different styles are supported for the HATCH interior style: 1 = Narrow diagonal 2 = Medium diagonal 3 = Wide diagonal 4 = Narrow crosshatch 5 = Medium crosshatch 6 = Wide crosshatch
Color	Nine color indices are allowed. Color indices 1 to 8 refer to pen stations 1 to 8 on the model 260. Models 100 and 200 issue a pen change prompt for color changes and will continue on a carriage return. Color 0 is not displayed.
Request Locator	The pen holder is used to indicate what point is to be input. The pen holder is moved by pressing the position keys on the front panel. When the pen holder is at the desired location, the point can be selected by pressing the START/ENTER button. This causes the coordinates of the point to be transmitted back to the user program.
Alpha Text	Alpha text can be positioned anywhere on the output page. The following text capabilities are available on the plotter: Fonts: 1 = Normal (default) 2 = Bold Sizes: 1 (default = 52 characters down page) Color

**Special
Information**

Superscript and Subscript

Overstrike Mode

Underlining

The pen position must be set at the lower left corner of the plotting surface prior to performing Open Workstation. Performing an Open Workstation on the model 100 and 200 sets the pen origin to the current pen position.

**VDI Opcode
Summary**

The functions available in the GSS VDI for the Strobe plotter are:

Clear Workstation—Ask for paper change

Close Workstation

Open Workstation

Set Alpha Text Position

Set Input Mode—Request mode is only supported mode

Set Pen Speed—One speed only

Set Writing Mode—Overstrike mode is the only supported mode

Update Workstation—No action is performed

Output

Arc (uses polyline attributes)

Bar (uses filled area attributes)

Cell Array—Outlines the area, no other action

Circle (uses filled area attributes)

Filled Area

Graphics Text

Output Alpha Text

Output Cursor Addressable Text—Transmits string to plotter with no error checking

Pie Slice (uses filled area attributes)

Polyline

Polymarker

Attributes

Set Alpha Text Color

Set Alpha Text Font and Size

Set Alpha Text Line Spacing

Set Alpha Text Overstrike Mode

Set Alpha Text Pass Through Mode
Set Alpha Text Quality—Returns default value
Set Alpha Text Script Mode
Set Alpha Text Underline Mode
Set Graphics Text String Baseline Rotation
Set Character Height
Set Color Representation
Set Fill Color Index
Set Fill Interior Style
Set Fill Style Index
Set Graphics Text Alignment
Set Graphics Text Color Index
Set Graphics Text Font—Four fonts supported
Set Polyline Color Index
Set Polyline Type
Set Polyline Width—Returns default setting
Set Polymarker Type
Set Polymarker Scale
Set Polymarker Color Index

Input

Input Locator—Request Mode

Inquiries

Inquire Alpha Text Capabilities
Inquire Alpha Cell Location
Inquire Alpha Font Availability
Inquire Alpha Text Position
Inquire Alpha String Extent
Inquire Addressable Character Cells—
Returns (-1, -1) to indicate no cursor addressing functions
Inquire Color Representation
Inquire Current Fill Area Attributes
Inquire Current Graphics Text Attributes
Inquire Current Polyline Attributes
Inquire Current Polymarker Attributes

Summagraphics Summatablet

File Name	summatb
Device Type	TABLET
Communications	Serial (RS-232)
Request Locator	When locator is invoked via the tablet, a tracking cross appears at the center of the screen of the output echo device. As the tablet's puck is moved across the tablet surface its position is shown by the tracking cross on the display. When the cross is at the desired location, that point can be selected by pressing one of the buttons on the puck. This causes the coordinates of the point to be transmitted back to the user program along with an ASCII character code for the button(s) pressed. The character codes associated with the buttons are as follows:

Character	Left	Middle	Right
<space>	×		
!		×	
"			×
#	×	×	
\$	×		×
%		×	×
'	×	×	×

If desired, the output echo device will perform an inking function. When the locator is terminated, a line from the initial position to the desired position is drawn. The current line attributes of the output echo device, such as color and line style, are honored.

Also the output echo device performs rubberbanding if desired. There are two types of rubberbanding supported, line and rectangle. If rubberbanding lines are desired, then a line will be drawn from the initial

locator position to the current position of the graphics cursor. The line changes dynamically as the cursor is moved. When the locator is terminated, the line is removed.

If rubberband rectangle is specified, a rectangle is displayed with one corner at the initial locator position and the opposite at the current position of the graphics cursor. The rectangle changes dynamically as the cursor is moved. When the locator is terminated, the rectangle is removed from the display.

**Special
Information**

This driver is an input only device and must be used in conjunction with an output echo display device.

**VDI Opcode
Summary**

The functions available in the GSS VDI for the Summatablet are:

Close Workstation

Open Workstation

Set Input Mode—Request mode is the only supported mode

Input

Input Locator—Request Mode

Tektronix 4105 Color Graphics Terminal

File Name tek4105

Device Type DISPLAY

Maximum Baud Rate 38400

Communications Serial (RS-232)

Features Supported Polylines Lines and arcs can be drawn with the seven line styles listed below:

- | | | |
|---|-----------------|---------------|
| 1 | _____ | Solid |
| 2 | _____ | Long Dash |
| 3 | | Dotted |
| 4 | _____ . _____ | Dash Dotted |
| 5 | _____ _____ | Medium Dashed |
| 6 | _____ . . _____ | Dash Two Dots |
| 7 | _____ - - - - - | Short Dash |

Graphics Text This terminal supports 58 character sizes. There is one base cell size for the smallest character. All other character sizes are multiples of the base size. The text characters can be rotated on 0, 90, 180, and 270 degree base lines.

Graphics Markers The driver supports a total of eleven markers, however, only the first six markers map to the VDI. All eleven markers are of a single size, and are shown in the following list:

- 1 = Dot
- 2 = Cross
- 3 = Star
- 4 = Circle
- 5 = X
- 6 = Diamond
- 7 = Period
- 8 = Square

- 9 = Square w/ black dot in center
- 10 = Solid Diamond
- 11 = Square w/ white dot in center

Filled Areas Filled areas, bars, pie slices and circles are all displayed using the current fill area attributes of color, interior style, and style index. The interior styles of HOLLOW, SOLID, PATTERN, and HATCH are all supported by this device. The interior style of HOLLOW is simulated by drawing vectors. The interior style of SOLID causes the solid fills of the device to be mapped to the VDI for color and index number. Sixteen device-dependent fill styles are available for the fill interior style of PATTERN (See Appendix J of the operators manual). Six different fill styles are supported for the HATCH interior style:

- 1 = Narrow diagonal
- 2 = Medium diagonal
- 3 = Wide diagonal
- 4 = Narrow crosshatch
- 5 = Medium crosshatch
- 6 = Wide crosshatch

Color The Tektronix 4105 terminal supports eight simultaneously displayable colors. Each color index can be remapped to a new color via the Set Color Representation command. There are a total of 64 colors available for mapping.

Request Locator When locator is invoked, a tracking cross appears on the display at the initial locator position. The cross can be moved by using the joydisk on the 4105 keyboard.

Initially, the cross moves in large increments. Holding down the shift key will cause the

cross to move slower. When the cross is at the desired location, that point can be selected by pressing any alpha key on the keyboard. This causes the coordinates of the point to be transmitted back to the user program.

If desired, the device will perform an inking function. When the locator is terminated, a line from the initial position to the desired position is drawn honoring the current line attributes such as color and line style.

Request Choice The function keys F1 to F8 on the keyboard are used to enter choice input.

Request String The keyboard is used to enter strings. All strings are terminated with the Return key.

Cursor Addressable Text Cursor addressable text is supported. The terminal must be in Cursor Addressing Mode before it can perform any cursor control functions. The following attributes are supported:

Reverse Video

Blink

Underline

Bold Intensity uses index 2

Color—Foreground and Background

Color Index

0 = Black

1 = White

2 = Red

3 = Green

4 = Blue

5 = Yellow

6 = Cyan

7 = Magenta

To display graphics primitives, the device must be removed from Cursor Addressing Mode.

Alpha Text

Alpha text can be positioned anywhere on the output page. The following text capabilities are available:

One Font

One Size

Superscript and Subscript

Bold

Underline

Line spacing

Eight colors

VDI Opcode Summary

The functions available in the GSS VDI for the Tektronix 4105 terminal driver are:

Clear Workstation

Close Workstation

Cursor Down

Cursor Left

Cursor Right

Cursor Up

Direct Cursor Address

Enter Cursor Addressing Mode

Erase to End of Line

Erase to End of Screen

Exit Cursor Addressing Mode

Hardcopy—issues a screen hard copy
command to the terminal

Home Cursor

Open Workstation

Set Alpha Text Position

Set Input Mode

Set Line Edit Characters

Set Writing Mode—Overstrike and REPLACE
(index 4 and 8) mode are supported

Update Workstation

Primitives

Arc (uses polyline attributes)

Bar (uses filled area attributes)

Cell Array

Circle (uses filled area attributes)

Filled Area

Graphics Text
Output Alpha Text
Output Cursor Addressable Text
Pie Slice (uses filled area attributes)
Polyline
Polymarker

Attributes

Reverse Video Off
Reverse Video On
Set Alpha Text Color
Set Alpha Text Font and Size
Set Alpha Text Line Spacing
Set Alpha Text Overstrike Mode
Set Alpha Text Pass Through Mode—Returns default value
Set Alpha Text Quality—Returns default value
Set Alpha Text Script Mode
Set Alpha Text Underline Mode
Set Background Color Index
Set Graphics Text String Baseline Rotation
Set Character Height
Set Color Representation
Set Cursor Text Attributes
Set Cursor Text Color Indices
Set Fill Color Index
Set Fill Interior Style
Set Fill Style Index
Set Graphics Text Alignment
Set Graphics Text Color Index
Set Graphics Text Font—Returns default setting
Set Polyline Color Index
Set Polyline Type
Set Polyline Width—Returns default setting
Set Polymarker Type
Set Polymarker Scale
Set Polymarker Color Index

Input

Input Locator—Request Mode
Input Choice—Request Mode
Input String—Request Mode
Read Cursor Movement Keys

Inquiries

Inquire Alpha Text Capabilities

Inquire Alpha Cell Location

Inquire Alpha Font Availability

Inquire Alpha Text Position

Inquire Alpha String Extent

Inquire Addressable Character Cells

Inquire Color Representation

Inquire Current Cursor Address

Inquire Current Fill Area Attributes

Inquire Current Graphics Text Attributes

Inquire Current Polyline Attributes

Inquire Current Polymarker Attributes

Virtual Device Metafile (VDM)

File Name	ddmeta (device driver)	
Device Type	METAFIL The default output metafile name is METAFILE.DAT. This name can be changed with the following commands: METAOUTPUT=filename export METAOUTPUT	
Communications	Not applicable	
Maximum Baud Rate	Not applicable	
Features Supported	Polylines	The six line styles listed below are guaranteed by the interpreter. Other index values are device-dependent. 1 _____ Solid 2 _____ Long Dash 3 Dotted 4 _____ . _____ Dash Dotted 5 ____ ____ Medium Dashed 6 ____ . . _____ Dash Two Dots 7 _ _ _ _ _ Short Dash Metafile supports sixteen polyline colors (assigned in a color table) and many (32767) line widths.
	Graphics Text	Metafile supports many fonts, sizes and rotations (32767). Metafile supports 256 text colors, assigned in the color table. When the metafile device is opened via the Open Workstation opcode, the coordinate transform mode should be set to preserve aspect ratio (1) to ensure that the metafile can be transported and displayed correctly on other devices.

Graphics Markers Metafile supports six styles, 256 colors and many (32767) marker sizes.

- 1 = Dot
- 2 = Cross
- 3 = Star
- 4 = Circle
- 5 = X
- 6 = Diamond

Filled Areas Filled areas, bars, pie slices and circles are all displayed using the current fill area attributes of color, interior style, and style index. The interior styles of HOLLOW, SOLID, and HATCH are all supported by this device. The interior style of PATTERN is mapped to the HATCH interior style. Six different styles are supported for the HATCH interior style:

- 1 = Narrow diagonal
- 2 = Medium diagonal
- 3 = Wide diagonal
- 4 = Narrow crosshatch
- 5 = Medium crosshatch
- 6 = Wide crosshatch

Color A color is selected by an index. The user can change the appearance of a color by selecting desired levels of the three color components: red, green and blue, that will make up the index. This can be used to create non-default colors such brown or orange. This new color will only be visible on color devices that allow color definition.

The metafile supports 256 color indices with many levels of red, green and blue.

The default colors are:

0 = Black
1 = White
2 = Red
3 = Green
4 = Blue
5 = Yellow
6 = Cyan
7 = Magenta
8-15 = White

Raster Writing

Raster writing modes define how pixels are MODES set in the bit map. All sixteen raster writing modes (Boolean operations between source and destination pixels) are supported by the metafile.

Cursor Addressable Text

The following attributes are supported:

Reverse Video
Blink
Bold Intensity
Color—Expanded palette
0 = Black
1 = White
2 = Red
3 = Green
4 = Blue
5 = Yellow
6 = Cyan
7 = Magenta
8-15 = White

To display graphics primitives, the device must be removed from Cursor Addressing Mode.

Alpha Text

The following capabilities are supported:

Underline
Line spacing
Superscript and Subscript
Overstrike

VDI Opcode Summary

The functions available in the VDI for the VDM are:

Open Workstation
Close Workstation
Clear Workstation
Update Workstation—No action is performed
Exit Cursor Addressing Mode
Enter Cursor Addressing Mode
Cursor Up
Cursor Down
Cursor Left
Cursor Right
Home Cursor
Erase to End of Screen
Erase to End of Line
Direct Cursor Address
Set Writing Mode
Output Alpha Text
Output Cursor Addressable Text
Set Pen Speed

Primitives

Polyline
Polymarker
Graphics Text
Filled Area
Cell Array
Generalized Drawing Primitives
Bar
Arc
Pie Slice
Circle

Attributes

Set Background Index
Reverse Video On
Reverse Video Off
Set Cursor Text Attributes
Set Cursor Text Color Indices
Set Alpha Text Line Spacing
Set Alpha Text Font and Size
Set Alpha Text Color Index
Set Alpha Text Overstrike Mode
Set Alpha Text Pass Through Mode

Set Alpha Text Position
Set Alpha Text Quality
Set Alpha Text Subscript/Superscript Mode
Set Alpha Text Underline Mode
Set Character Height
Set Graphics Text String Baseline Rotation
Set Color Representation
Set Polyline Line Type
Set Polyline Line Width
Set Polyline Color Index
Set Polymarker Type
Set Polymarker Scale
Set Polymarker Color Index
Set Graphics Text Alignment
Set Graphics Text Font
Set Text Color Index
Set Fill Interior Style
Set Fill Style Index
Set Fill Color Index

Inquiries

Inquire Addressable Character Cells
Inquire Alpha Text Font Availability
Inquire Alpha Text Capabilities
Inquire Alpha Text String Length
Inquire Alpha Text Cell Location
Inquire Alpha Text Position
Inquire Color Representation
Inquire Current Polyline Attributes
Inquire Current Polymarker Attributes
Inquire Current Fill Area Attributes
Inquire Current Graphics Text Attributes

VT100 Terminal with Retro-Graphics Card

File Name	vt100ret	
Device Type	DISPLAY	
Communications	Serial (RS-232)	
Features Supported	Polylines	Lines and arcs can be drawn with the seven line styles listed below:

- | | | |
|---|-----------------|---------------|
| 1 | _____ | Solid |
| 2 | _____ | Long Dash |
| 3 | | Dotted |
| 4 | _____ . _____ | Dash Dotted |
| 5 | ____ _ | Medium Dashed |
| 6 | _____ . . _____ | Dash Two Dots |
| 7 | __ _ | Short Dash |

Graphics Text	This printer supports four character sizes with a Preserve Aspect Ratio and four sizes in a Non-preserve Aspect Ratio. Note that these graphics text characters cannot be rotated.
---------------	--

Graphic Markers	The driver supports six kinds of Graphic Markers. Each marker has five possible sizes in Preserve Aspect Ratio and five sizes in Non-preserve Aspect Ratio. The six kinds of markers are shown in the following list:
-----------------	---

- 1 = Dot
- 2 = Cross
- 3 = Star
- 4 = Circle
- 5 = X
- 6 = Diamond

Filled Areas	Filled areas, bars, pie slices and circles are all displayed using the current fill area attributes of color, interior style, and style index. The interior styles of
--------------	---

HOLLOW, SOLID, and HATCH are all supported by this device. The interior style of PATTERN is mapped to the HATCH interior style. Six different styles are supported for the HATCH interior style:

- 1 = Narrow diagonal
- 2 = Medium diagonal
- 3 = Wide diagonal
- 4 = Narrow crosshatch
- 5 = Medium crosshatch
- 6 = Wide crosshatch

Color The VT100 with Retro-Graphics supports two colors. Index 1 is displayed as the foreground color (white) and index 0 is displayed as the background color (black). These colors cannot be remapped.

Request Locator When locator is invoked on the Retro-Graphics card a tracking cross appears on the display at the initial locator position. The cross can be moved by pressing one of the arrow keys on the keyboard. Initially, the cross moves in small increments. Holding the arrow key down will cause the cross movement to speed up after a short pause.

When the cross is at the desired location, that point can be selected by pressing any alpha key on the keyboard. This causes the coordinates of the point to be transmitted back to the user program. If desired, the device will perform an inking function. When the locator is terminated, a line from the initial position to the desired position is drawn honoring the current line attributes such as color and line style.

Request Choice The numeric keys 0 to 9 on the keyboard are used to enter choice input.

Request String The keyboard is used to enter strings. All strings are terminated by the Enter key.

Cursor
Addressable
Text

Cursor addressable text is supported. The Text device must be in Cursor Addressing Mode before it can perform any cursor control functions. The following attributes are supported:

Reverse Video

Underline

Blink

Note

Underline or Reverse video may not be active at the same time. The block cursor will allow reverse and the bar cursor will allow underline.

Alpha Text

Alpha text can be positioned anywhere on the output page. The following text capabilities are available on the Retro-Graphics card:

One Font

One Size

Superscript and Subscript

Underline

Line spacing

VDI Opcode
Summary

The functions available in the GSS VDI for the VT100 driver are:

Clear Workstation

Close Workstation

Cursor Down

Cursor Left

Cursor Right

Cursor Up

Direct Cursor Address

Enter Cursor Addressing Mode

Erase to End of Line

Erase to End of Screen

Exit Cursor Addressing Mode

Home Cursor

Open Workstation

Remove Graphics Input Cursor
Set Alpha Text Position
Set Input Mode
Set Line Edit Characters
Set Writing Mode—The following modes are supported:
 1—Clear (Color 0)
 7—XOR
 16—All bits on (Color 255)
Update Workstation—No action is performed

Primitives

Arc (uses polyline attributes)
Bar (uses filled area attributes)
Cell Array
Circle (uses filled area attributes)
Filled Area
Graphics Text
Output Alpha Text
Output Cursor Addressable Text
Pie Slice (uses filled area attributes)
Polyline
Polymarker

Attributes

Reverse Video Off
Reverse Video On
Set Alpha Text Color
Set Alpha Text Font and Size
Set Alpha Text Line Spacing
Set Alpha Text Overstrike Mode
Set Alpha Text Pass Through Mode—Returns default value
Set Alpha Text Quality—Returns default value
Set Alpha Text Script Mode
Set Alpha Text Underline Mode
Set Background Color Index
Set Graphics Text String Baseline Rotation
Set Character Height
Set Color Representation
Set Cursor Text Attributes
Set Cursor Text Color Indices
Set Fill Color Index
Set Fill Interior Style

Set Fill Style Index
Set Graphics Text Alignment
Set Graphics Text Color Index
Set Graphics Text Font—Returns default setting
Set Polyline Color Index
Set Polyline Type
Set Polyline Width—Returns default setting
Set Polymarker Type
Set Polymarker Scale
Set Polymarker Color Index

Input

Input Locator—Request Mode
Input Choice—Request Mode
Input String—Request Mode
Input String—Sample Mode
Read Cursor Movement Keys

Inquiries

Inquire Alpha Text Capabilities
Inquire Alpha Cell Location
Inquire Alpha Font Availability
Inquire Alpha Text Position
Inquire Alpha String Extent
Inquire Addressable Character Cells
Inquire Color Representation
Inquire Current Cursor Address
Inquire Current Fill Area Attributes
Inquire Current Graphics Text Attributes
Inquire Current Polyline Attributes
Inquire Current Polymarker Attributes

ASCII Code Chart

The following table is intended as a convenience, to provide you with the ASCII Character Codes that are used throughout the computer industry.

CONTROL		NUMBERS SYMBOLS		UPPER CASE		LOWER CASE	
NUL	DLE	SP	0	@	P	'	p
SOH	DC1	!	1	A	Q	a	q
STX	DC2	"	2	B	R	b	r
ETX	DC3	#	3	C	S	c	s
EOT	DC4	\$	4	D	T	d	t
ENQ	NAK	%	5	E	U	e	u
ACK	SYN	&	6	F	V	f	v
BEL	ETB	'	7	G	W	g	w
BS	CAN	(	8	H	X	h	x
HT	EM	)	9	I	Y	i	y
LF	SUB	*	:	J	Z	j	z
VT	ESC	+	;	K	[	k	{
FF	FS	,	<	L	\	l	
CR	GS	-	=	M	]	m	}
SO	RS	.	>	N	^	n	~
SI	US	/	?	O	_	o	RUBOUT (DEL)



Bibliography

Benest, I.N. "A Review of Computer Graphics Publications,"
Computers and Graphics. 4(2:1979), 95-136.

Bono, P.R., Encarnacao, J.L., Hopgood, F.R.A., ten Hagen,
P.J.W. "GKS--The First Graphics Standard," *IEEE*
Computer Graphics and Applications. 2(5:1982), 9-23.

Foley, J.D. and van Dam, A. *Fundamentals of Interactive Computer*
Graphics. Reading, Massachusetts: Addison-Wesley,
1982.

Giloi, W. K. *Interactive Computer Graphics*. Englewood Cliffs,
New Jersey: Prentice-Hall Inc., 1978.

Graphical Kernel System (GKS) Functional Description, Draft
International Standard ISO/DIS (July 1983).

Newman, W.M. and Sproull, R.P. *Principles of Interactive*
Computer Graphics. New York: McGraw-Hill Book Co.,
1979.

Raster Graphics. Covina, California: Conrac Corporation, 1980.



Index

A

Additional Elements, 2-3
Additional Functions Control,
2-8
ADE, G-1
Alpha Text, 1-6, 2-5, 2-8, 5-42,
5-80, C-2
Alpha Text Font, 5-75, 5-130
Alpha Text Position, 5-34
Alpha Text Size, 5-130
Alpha Text Spacing, 5-79
Angle Specification, 5-41
ANSI, 1-2
Application, 2-10
Arcs, 2-5, 5-39
Argument, G-1
Array, G-1
Aspect Ratio, 2-11, 2-14, 3-3,
5-20, 5-27
Aspects of Primitives, G-1
AT&T Model 455 Letter
Quality Printer, C-8
AT&T Model 470 Graphics
Printer, C-4
AT&T UNIX PC Model 7300
Display, C-12
Attribute Functions, 2-6, 5-61,
5-62, G-1
Attributes, 5-61

B

Background, 1-1
Background Color, 5-68, 5-82

Background Index, 2-7
Bars, 2-5, 5-38
Baseline, 5-96, 5-98
Bindings, 1-3, A-1, G-1
Binding Conventions, 5-3, A-1
Blink, 5-67
Bold, 5-67

C

Cartesian Coordinate System,
G-1
Cell Array, 2-5, 5-38, 5-48,
5-118, G-2
Cell Array Function, 5-38
Character And File I/O, 1-8
Character Attributes, 2-6
Character Cell, 5-96, 5-127
Character Cell Height, 5-79
Character Height, 5-25, 5-96,
5-125
Character I/O, 3-3
Character Width, 5-125
Choice, 2-7, 5-99
Capability Flag, 5-24
Choice Input Device, G-2
Circles, 2-5, 2-15, 5-39
CIT-101e Terminal with CIG-
101e Graphics Card, C-17
CIT-161 Terminal with CIG-267
Graphics Card, C-22
Clear Workstation, 2-4
Clipping, G-2
Close Workstation, 2-4

- Cm_close (), 5-143
- Cm_inq (), 5-144
- Cm_open (), 5-145
- Cmr_x_now (), 5-148
- Cmr_x_wait (), 5-149
- Cm_start, 5-146
- Cm_stop, 5-147
- Cmt_x_now (), 5-150
- Cmt_x_wait (), 5-151
- Code Example—Open Workstation, 5-29
- Code Example—Set Character Height, 5-97
- Color, 2-14, 5-22, 5-23, 5-26, 5-83, C-1, C-2
 - Capability Flag, 5-23
- Color Components, 5-70, 5-86, 5-120
- Color Index, 5-35, 5-46, 5-70, 5-118, 5-120
- Color Indices, 2-14, 5-26
- Color Map, G-2
- Color Representation, 2-7
- Color Table, 5-47, G-2
- Columns, 5-2
- Communications, C-2
- Communications (Port) I/O, 5-140
- Components of GSS-DRIVERS, 3-1
- Control, 2-10
- Control Functions, 2-4, 5-1, 5-4
- Control Mode, 2-12
- Coordinate Graphics, G-2
- Coordinate Scaling, G-3
- Coordinate System, 2-2
- Coordinate Transformation, 1-8, 3-2
- Coordinate Transformation Flag, 2-11
- Coordinate Transformation Mode, 5-25

- Coordinate Transformation
 - Mode Flag, 5-20
- CORE, 1-2
- Creating Graphics
 - Applications, 1-9
- Current Cursor Position, 5-121
- Current File Position, 5-141, 5-163
- Cursor Addressable Text, 2-8, 2-12, 5-119, C-3
- Cursor Addressing Mode, 2-12, 5-2, 5-32
- Cursor Control, 5-2
- Cursor Movement, 5-102
- Cursor Text, 1-6, 2-5

D

- Default, G-3
- Defaults, 5-27
- Device Capabilities, 5-115, C-1
- Device Control, 5-1
- Device Coordinate Units Flag, 5-21
- Device Coordinates, 2-2, 3-2, 5-20, G-3
- Device Drivers, 3-1, C-1, G-3
- Device Driver Management, 1-7, 3-2
- Device Handle, 2-16, 5-21, 5-139, G-3
- Device Independence, 2-11
- Device Information Categories, C-2
- Device Space, G-3
- Device Type, 5-24, C-2
- Device-Independent, G-3
- Device-Specific Data, 2-12
- Diablo C150 Ink Jet Color Printer, C-27
- Directories, 3-4

Directories And File Names,
3-4

Directory, 5-141

Disk I/O, 5-141

Display Device, G-3

Display Surface, G-3

Distribution Files, 4-1

E

Echo, 5-105, G-3

Editing Characters, 5-71, 5-106

Emulation, 1-8, 3-4

Environmentals, G-3

Epson MX-100, C-31

Epson MX-80 with Graphtrax
Plus, C-35

Errors, 1-10, 3-4, 5-122, B-1

Error Codes, 1-10, 3-6

Error Handling, 2-13

Error Reporting, 3-4

Escape, G-4

F

fd, 5-3, 5-141

fd_close (), 5-152

fd_connect (), 5-153

fd_copy (), 5-154

fd_delete (), 5-155

fd_directory (), 5-156

fd_disconnect (), 5-157

fd_inq (), 5-158

fd_open (), 5-159

fd_parse (), 5-160

fdp_read (), 5-166

fdp_write (), 5-167

fd_read (), 5-161

fd_rename (), 5-162

fd_seek (), 5-163

fd_size (fd), 5-164

fd_write (), 5-165

Features Supported, C-2

File Descriptor, 3-4, 5-141,
5-159

File I/O, 3-4

File Names, 3-4, 5-141, C-2

Filled Areas, 2-17, 2-5, 5-38,
5-52, C-2, G-4

Fill Area Attributes, 2-7, 5-133

Fill Area Capability Flag, 5-23

Fill Area Function, 5-38

Fill Interior Style, 5-84

Font, 5-75, 5-79, 5-127

Foreground Color, 5-68

G

Gantt Chart, 2-18

GDP, 5-22, G-4

Generalized Drawing

Primitives, 5-22, 5-38, G-4

Generic Codes, B-2

GKS, 1-2, G-4

Graphical Kernel System, G-4

Graphics Cursor, 5-99, 5-105,
5-110

Graphics Emulation, 1-8, 3-4

Graphics Input, 2-2, C-2

Graphics Input Cursor, 5-30

Graphics Kernel System, 1-2

Graphics Markers, C-2

Graphics Mode, 2-12, 5-36

Graphics Output, 2-1

Graphics Primitive Output,
2-14

Graphics Primitives, 2-8, 2-14,
G-4

Graphics Program Flow, 2-10

Graphics Reference Model, 2-1

Graphics Standards, 1-1, 1-3

Graphics Text, 1-6, 2-5, 5-2,
C-2

Graphics Text Alignment, 5-93
Graphics Text Attributes, 5-137
GSS Product Line, 1-4
GSS-DRIVERS, 1-1, G-4
GSS-DRIVERS Features, 1-5
GSS-DRIVERS Functions, 1-7
GSS-DRIVERS Graphics
 Functions, 2-4

H

Hardware Font, 5-75
Hewlett-Packard 7470A Plotter,
 C-39
Hewlett-Packard 7475A Plotter,
 C-43
Host-Independent, G-4
Houston Instrument DMP-29
 Plotter, C-47

I

Initialization, 5-1
Initialization And Termination,
 5-1
Inking, 5-104
 Capability Flag, 5-25
Input And Inquiry, 2-15
Input Choice, 2-7
Input Functions, 2-7, 5-99,
 5-100, G-4
Input Locator, 2-7
Input String, 2-7
Input Valuator, 2-7
Input/Output Functions, 5-139,
 5-142
Inquiry Functions, 2-8, 5-115,
 5-116, G-5
Installing the VDI Software,
 4-2
INT16, 5-3

INT32, 5-3
INT8, 5-3
Integer, G-5
Interface, 1-3
I/O Channel, 5-143
I/O System, G-5
ISO, 1-2

K

Keyboard, 5-106

L

Language Bindings, A-1
libcvdi.a, 4-1
Line Width, 5-26, 5-88
Linking Applications, 4-11
Locator, 2-7, 5-99
 Capability Flag, 5-24
Locator Input Device, G-5
Logical Channel Number,
 5-145

M

Marker Height, 5-26
Message, 5-3
Metafile, 1-7, 5-3, 5-19
Metafile VDI Functions, 5-3

N

NDC, 2-2, 2-4, 2-16, 3-2, 5-20,
 5-25, 5-61, G-5
NEC Model 3550 Letter Quality
 Printer, C-55
NEC Model 7730 Letter Quality
 Printer, C-51
Nicolet Zeta Corporation Zeta
 8 Plotter, C-59

Normalized Device
Coordinates, 2-2, 2-4, 3-2,
G-5
Normalized Device Coordinate
Space, G-5
Null-Terminated String, G-6

O

Okidata Microline 84 with Step
2 Support, C-63
Okidata Microline 92, C-67
Okidata Microline 93, C-63
Open Workstation, 2-4
Opening An Input Device, 5-27
Opening An Output Device,
5-27
Output Attributes, 2-1
Output Buffer, 5-143, 5-147
Output Functions, 2-4, 5-37,
5-39
Output Primitives, 2-1, 2-4,
5-37, G-6
Overstrike, 5-76
Overview, 4-1

P

Page Size, 5-119
Pass Through Mode, 5-77
Pause Indicator, 5-19
Pel, 5-47
Pie Slices, 2-5, 5-39
Pixel, 5-47, G-6
Capability Flag, 5-23
Plotter Pen Speed, 5-33
Polling, 5-109, 5-111
Polylines, 2-5, 5-37, 5-57, C-2,
G-6
Polyline Attributes, 2-6, 5-134
Polyline Function, 5-37

Polymarkers, 2-5, 5-38, 5-59,
G-6
Polymarker Attributes, 2-6,
5-135
Polymarker Function, 5-38
Polymarker Height, 5-90
Polymarker Type, 5-91, C-1
Predefined Colors, 5-22
Preserve Aspect Ratio Mode,
3-3
Primitive Attributes, 5-115
Primitive Functions, 5-37
Printronix MVP Printer, C-71
Printronix P300 Printer, C-71
Printronix P600 Printer, C-71
Programming Example, A-2
Programming Tips, 2-10
Prompting Flag, 2-11
Proportional Font, 5-127, 5-131
Proportional Spacing Flag,
5-126
Pseudocode Example, 2-16

R

Raster, 1-8, 5-47, G-6
Raster Writing, C-3
Read Cursor Movement Keys,
2-7
Real, G-6
Reference Model, 2-1
Request Input Mode, 2-7, 5-99
Request Locator, C-3
Request String, C-3
Reverse Video, 5-65, 5-67
RGB, G-6
Rotation, 2-2, 2-5
Rows, 5-2
Rubberband, 5-104
Capability Flag, 5-25

S

- Sample Input Mode, 2-7, 5-99
- Scale, 2-2
- Scaling, 2-5, 2-12, 5-20
- Seek Operation, 5-141
- Setting Attributes, 2-13
- Setting the Environment, 4-10
- Special Information, C-3
- Specific Codes, B-4
- Standards, 1-1
- Standards Implementation, 1-3
- String, 2-7, 5-99
 - Capability Flag, 5-24
- Strobe Model 100—Single Pen Plotter, C-75
- Strobe Model 200—Single Pen Plotter, C-75
- Strobe Model 260—Eight Pen Plotter, C-75
- Subscripting, 5-80, 5-125
- Summagraphics Summatablet, C-79
- Superscripting, 5-80, 5-125

T

- Tektronix 4105 Color Graphics Terminal, C-81
- Termination, 5-1
- Terminator, 5-105
- Testing the System, 4-12
- Text, 1-6, 5-38
- Text Alignment, 2-16, 5-93
 - Capability Flag, 5-25
- Text Function, 5-38
- Text Options, 1-6
- Text Quality, 5-78
- Text Rotation, 2-14, 5-98
 - Capability Flag, 5-23
- Text Sizes, 2-12
- Text String Length, 5-131

- Transformations, 1-8, 2-2, G-7
- Transformation Modes, 2-16, 3-2
- Transformation Mode Flag, 5-27
- types.h, 4-1

U

- Underline, 5-67
- Underlining, 5-81
- Unit Distance, 2-11
- Update Workstation, 2-4

V

- Valuator, 2-7, 5-99
 - Capability Flag, 5-24
- Valuator Input Device, G-7
- v_appl (), 5-5
- v_arc (), 5-40
- v_atext (), 5-42
- v_bar (), 5-44
- v_cellarray (), 5-45
- v_circle (), 5-49
- v_clrwk (), 5-6
- v_clswk (), 5-7
- vcur_att (), 5-66
- vcur_color (), 5-68
- v_curdown (), 5-8
- v_curhome (), 5-9
- v_curleft (), 5-10
- v_curright (), 5-11
- v_curtext (), 5-50
- v_curup (), 5-12
- VDI, 1-2, G-7
- VDI Opcode Summary, C-3
- Vdittest, 4-1
- VDM, 1-2, C-87
- v_dspcur (), 5-13
- v_eeol (), 5-14

v_eeos (), 5-15
 v_enter_cur (), 5-16
 v_exit_cur (), 5-17
 v_fillarea (), 5-51
 v_gtext (), 5-53
 v_hardcopy (), 5-18
 Virtual Device Interface, 1-2,
 G-7
 Virtual Device Metafile, 1-2,
 C-87
 v_msg (), 5-19
 v_opnwk (), 5-20
 v_pieslice (), 5-54
 v_pline (), 5-56
 v_pmarker (), 5-58
 vqa_cap (), 5-123
 vqa_cell (), 5-126
 vqa_font (), 5-128
 vqa_length (), 5-131
 vqa_position (), 5-132
 vq_cellarray (), 5-117
 vq_chcells (), 5-119
 vq_color (), 5-120
 vq_curaddress (), 5-121
 vq_error, 5-122
 vqf_attributes (), 5-133
 vql_attributes (), 5-134
 vqm_attributes (), 5-135
 vqt_attributes (), 5-136
 vrd_curkeys (), 5-101
 v_rmcure (), 5-30
 vrq_choice (), 5-103
 vrq_locator (), 5-104
 vrq_string (), 5-106
 vrq_valuator (), 5-108
 v_rvoff (), 5-64
 v_rvon (), 5-65
 vsa_color (), 5-72
 vsa_font (), 5-73
 vsa_overstrike (), 5-76
 vsa_passthru (), 5-77
 vsa_position (), 5-34
 vsa_quality (), 5-78
 vsa_spacing (), 5-79
 vsa_supersub (), 5-80
 vsa_underline (), 5-81
 vsb_color (), 5-82
 vs_color (), 5-69
 vs_curaddress (), 5-32
 vs_editchars (), 5-71
 vsf_color (), 5-83
 vsf_interior (), 5-84
 vsf_style (), 5-85
 vsl_color (), 5-86
 vsl_type (), 5-87
 vsl_width (), 5-88
 vsm_choice (), 5-109
 vsm_color (), 5-89
 vsm_height (), 5-90
 vsm_locator (), 5-110
 vsm_string (), 5-111
 vsm_type (), 5-91
 vsm_valuator (), 5-113
 vs_penspeed (), 5-33
 vst_alignment (), 5-92
 vst_color (), 5-94
 vst_font (), 5-95
 vst_height (), 5-96
 vst_rotation (), 5-98
 vswr_mode (), 5-35
 VT100 Terminal with Retro-
 Graphics Card, C-92
 v_updwk (), 5-31

W

Workstations, 2-2, 5-31, G-7
 Workstation Control, 5-1
 Workstation Identifier, 2-2,
 5-21
 Workstation Type, 5-24
 Writing Modes, 5-24, 5-35

