Credits and Trademarks

Sun Workstation® is a registered trademark of Sun Microsystems, Inc.
SunStation®, Sun Microsystems®, SunCore®, SunWindows®, DVMA®, and the combination of Sun with a numeric suffix are trademarks of Sun Microsystems, Inc.
UNIX, UNIX/32V, UNIX System III, and UNIX System V are trademarks of AT&T Bell Laboratories.
Intel® and Multibus® are registered trademarks of Intel Corporation.
DEC®, PDP®, VT®, and VAX® are registered trademarks of Digital Equipment Corporation.

Copyright © 1986 by Sun Microsystems.

This publication is protected by Federal Copyright Law, with all rights reserved. No part of this publication may be reproduced, stored in a retrieval system, translated, transcribed, or transmitted, in any form, or by any means manual, electric, electronic, electro-magnetic, mechanical, chemical, optical, or otherwise, without prior explicit written permission from Sun Microsystems.
Preface .................................................................................................................................................... xv

Chapter 1 Introduction .......................................................................................................... 3
1.1. Using SunCGI ................................................................................................................... 3
1.2. The SunCGI Lint Library ........................................................................................... 5
1.3. Overview of SunCGI ..................................................................................................... 5
      Initialization and Termination .................................................................................. 5
      Output Primitives ........................................................................................................ 6
      Attributes ..................................................................................................................... 6
      Input .................................................................................................................................. 6
      Errors .................................................................................................................................. 6
      Programming Tips ......................................................................................................... 7
      Appendices ..................................................................................................................... 7
1.4. References ............................................................................................................................. 7

Chapter 2 Initializing and Terminating SunCGI .............................................. 11
2.1. View Surface Initialization and Selection ......................................................... 11
      Open CGI (SunCGI Extension) ............................................................................... 12
      Open View Surface (SunCGI Extension) .............................................................. 13
      Activate View Surface (SunCGI Extension) .......................................................... 16
      Deactivate View Surface (SunCGI Extension) ....................................................... 16
      Close View Surface (SunCGI Extension) .............................................................. 16
      Close CGI (SunCGI Extension) ................................................................................ 16
2.2. View Surface Control ..................................................................................................... 17
      VDC Extent .................................................................................................................... 17
Device Viewport ........................................................................................................... 19
Clip Indicator .................................................................................................................. 19
Clip Rectangle ................................................................................................................ 20
Hard Reset ....................................................................................................................... 20
Reset to Defaults ........................................................................................................... 20
Clear View Surface ........................................................................................................ 21
Clear Control ................................................................................................................... 21
Set Error Warning Mask ........................................................................................... 22

2.3. Running SunCGI with SunView ............................................................................. 22
Set Up SIGWINCH (SunCGI Extension) ..................................................................... 23

2.4. Interface Negotiation ............................................................................................. 24
Inquire Device Identification ...................................................................................... 25
Inquire Device Class .................................................................................................... 25
Inquire Physical Coordinate System ........................................................................ 25
Inquire Output Function Set .................................................................................... 26
Inquire VDC Type .......................................................................................................... 26
Inquire Output Capabilities ..................................................................................... 27

2.5. Input Capability Inquiries ....................................................................................... 27
Inquire Input Capabilities ........................................................................................ 27
Inquire LID Capabilities ........................................................................................... 28
Inquire Trigger Capabilities .................................................................................... 25

Chapter 3  Output ....................................................................................................... 33
3.1. Geometrical Output Primitives ........................................................................... 32
Polyline ....................................................................................................................... 34
Disjoint Polyline ....................................................................................................... 34
Polymarker ............................................................................................................... 35
Polygon ...................................................................................................................... 35
Partial Polygon ......................................................................................................... 36
Rectangle .................................................................................................................. 36
Circle ......................................................................................................................... 36
Circular Arc Center ................................................................................................. 36
Circular Arc Center Close ....................................................................................... 35
Circular Arc 3pt ................................................................. 40
Circular Arc 3pt Close .................................................. 41
Ellipse .................................................................................. 41
Elliptical Arc ........................................................................ 41
Elliptical Arc Close ............................................................. 42

3.2. Raster Primitives ........................................................................................................... 42
Text .................................................................................................................. 42
VDM Text ........................................................................................................ 43
Append Text .................................................................................................... 43
Inquire TextExtent ...................................................................................... 43
Cell Array ................................................................................................. 44
Pixel Array ............................................................................................... 44
BitBlt Source Array ................................................................................. 45
BitBlt Pattern Array ................................................................................. 46
BitBlt Patterned Source Array .............................................................. 46
Inquire Cell Array ...................................................................................... 47
Inquire Pixel Array .................................................................................... 47
Inquire Device Bitmap .............................................................................. 48
Inquire BitBlt Alignments ................................................................. 48

3.3. Drawing Modes ............................................................................................................ 48
Set Drawing Mode .................................................................................... 49
Set Global Drawing Mode (SunCGI Extension) ........................................ 50
Inquire Drawing Mode ............................................................................... 50

Chapter 4 Attributes ........................................................................................................... 53

4.1. Bundled Attribute Functions ...................................................................................... 54
Set Aspect Source Flags ............................................................................... 56
Define Bundle Index (SunCGI Extension) ................................................ 56

4.2. Line Attributes ............................................................................................................ 57
Polyline Bundle Index ................................................................................... 57
Line Type ....................................................................................................... 58
Line Type (SunCGI Extension) .................................................................... 58
Line Endstyle (SunCGI Extension) ............................................................ 58
Line Width Specification Mode ............................................................... 59
Line Width ................................................................. 59
Line Color .................................................................... 59
4.3. Polymarker Attributes ............................................. 60
    Polymarker Bundle Index ........................................... 60
    Marker Type ............................................................ 60
    Marker Size Specification Mode ................................. 60
    Marker Size ............................................................ 61
    Marker Color .......................................................... 61
4.4. Solid Object Attributes ............................................ 61
    Fill Area Bundle Index ............................................. 62
    Interior Style .......................................................... 62
4.5. Solid Interior Fill Attribute ....................................... 62
    Fill Color ............................................................... 63
4.6. Hatch and Pattern Attributes ..................................... 63
    Hatch Index ........................................................... 64
    Pattern Index ........................................................ 65
    Pattern Table ........................................................ 65
    Pattern Reference Point .......................................... 65
    Pattern Size .......................................................... 66
    Pattern with Fill Color (SunCGI Extension) .................. 66
4.7. Perimeter Attributes ................................................ 66
    Perimeter Type ....................................................... 66
    Perimeter Width ..................................................... 67
    Perimeter Width Specification Mode ............................ 67
    Perimeter Color ...................................................... 68
4.8. Text Attributes ...................................................... 68
    Text Bundle Index .................................................. 68
    Text Precision ........................................................ 68
    Character Set Index ................................................ 69
    Text Font Index ....................................................... 69
    Character Expansion Factor ...................................... 70
    Character Spacing ................................................... 70
    Character Height ..................................................... 70
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed Font (SunCGI Extension)</td>
<td>71</td>
</tr>
<tr>
<td>Text Color</td>
<td>71</td>
</tr>
<tr>
<td>Character Orientation</td>
<td>71</td>
</tr>
<tr>
<td>Character Path</td>
<td>72</td>
</tr>
<tr>
<td>Text Alignment</td>
<td>72</td>
</tr>
<tr>
<td>4.9. Color Attributes</td>
<td>74</td>
</tr>
<tr>
<td>Color Table</td>
<td>74</td>
</tr>
<tr>
<td>4.10. Inquiry Functions</td>
<td>75</td>
</tr>
<tr>
<td>Inquire Line Attributes</td>
<td>75</td>
</tr>
<tr>
<td>Inquire Marker Attributes</td>
<td>75</td>
</tr>
<tr>
<td>Inquire Fill Area Attributes</td>
<td>76</td>
</tr>
<tr>
<td>Inquire Pattern Attributes</td>
<td>76</td>
</tr>
<tr>
<td>Inquire Text Attributes</td>
<td>77</td>
</tr>
<tr>
<td>Inquire Aspect Source Flags</td>
<td>78</td>
</tr>
<tr>
<td>Chapter 5 Input</td>
<td>81</td>
</tr>
<tr>
<td>5.1. Input Device Initialization</td>
<td>84</td>
</tr>
<tr>
<td>Initialize LID</td>
<td>84</td>
</tr>
<tr>
<td>Release Input Device</td>
<td>85</td>
</tr>
<tr>
<td>Associate</td>
<td>85</td>
</tr>
<tr>
<td>Set Default Trigger Associations</td>
<td>86</td>
</tr>
<tr>
<td>Dissociate</td>
<td>86</td>
</tr>
<tr>
<td>Set Initial Value</td>
<td>87</td>
</tr>
<tr>
<td>Set VALUATOR Range</td>
<td>87</td>
</tr>
<tr>
<td>Track On</td>
<td>88</td>
</tr>
<tr>
<td>Track Off</td>
<td>89</td>
</tr>
<tr>
<td>5.2. Synchronous Input</td>
<td>90</td>
</tr>
<tr>
<td>Request Input</td>
<td>91</td>
</tr>
<tr>
<td>5.3. Asynchronous Input</td>
<td>92</td>
</tr>
<tr>
<td>Initiate Request</td>
<td>92</td>
</tr>
<tr>
<td>5.4. Event Queue Input</td>
<td>93</td>
</tr>
<tr>
<td>Enable Events</td>
<td>95</td>
</tr>
<tr>
<td>Await Event</td>
<td>95</td>
</tr>
</tbody>
</table>
Flush Event Queue ................................................................. 96
Selective Flush of Event Queue ........................................... 96

5.5. Miscellaneous Input Functions ........................................ 97
    Sample Input ........................................................................ 97
    Get Last Requested Input ....................................................... 97
    Disable Events ..................................................................... 98

5.6. Status Inquiries ................................................................. 98
    Inquire LID State List ............................................................ 98
    Inquire LID State ................................................................. 99
    Inquire Trigger State ............................................................ 99
    Inquire Event Queue State ..................................................... 99

Appendix A  Differences between SunCore and SunCGI .......... 103
    A.1. Output Primitives .......................................................... 103
        Output Aspects of SunCore not Supported by SunCGI .......... 104
        Output Features of SunCGI not Available in SunCore ......... 104
    A.2. Segmentation .................................................................. 104
    A.3. Differences in Input Functions between SunCore and SunCGI 104

Appendix B  Unsupported Aspects of CGI ............................... 107

Appendix C  Type and Structure Definitions ......................... 111

Appendix D  Error Messages .................................................... 123
    D.1. Successful Return (0) .................................................... 123
    D.2. State Errors (1-5) ......................................................... 123
    D.3. Control Errors (10-16) .................................................. 124
    D.4. Coordinate Definition (20-24) ....................................... 124
    D.5. Output Attributes (30-51) .............................................. 125
    D.6. Output Primitives (60-70) .............................................. 128
    D.7. Input (80-97) .............................................................. 129
    D.8. Implementation Dependent (110-112) ............................ 131
    D.9. Possible Causes of Visual Errors ................................. 131
Appendix E  Sample Programs ......................................................... 137
   E.1. Martini Glass ........................................................................ 137
   E.2. Tracking Box ........................................................................ 138

Appendix F  Using SunCGI and Pixwins ........................................... 143
   F.1. cgipw Functions ................................................................. 143
       Open Pixwin CGI ................................................................. 143
       Open a CGI Pixwin ............................................................ 143
       Close a CGI Pixwin ............................................................ 144
       Close Pixwin CGI ............................................................... 144
   F.2. Using cgipw .......................................................................... 144
   F.3. cgipw Functions ................................................................. 145
   F.4. Example Program ............................................................... 147

Appendix G  Using SunCGI with Fortran Programs ......................... 151
   G.1. Programming Tips ............................................................. 151
   G.2. Example Program ............................................................. 152
   G.3. FORTRAN Interfaces to SunCGI ....................................... 154

Appendix H  Short C Binding ............................................................ 173
Tables

Table 2-1  SunCGI Default States ........................................................................ 13
Table 2-2  Available View Surfaces ................................................................... 15
Table 2-3  View Surface Default States ............................................................... 15
Table 2-4  Error Warning Masks ........................................................................ 22
Table 2-5  Class Dependent Information ............................................................. 29

Table 4-1  Default Attributes ............................................................................ 54
Table 4-2  Attribute Source Flag Numbers .......................................................... 56
Table 4-3  Available Fonts ................................................................................. 70
Table 4-4  Normal Alignment Values ................................................................. 74
Table 4-5  Default Color Lookup Table .............................................................. 74

Table 5-1  Input Devices Offered by SunCGI ...................................................... 82
Table 5-2  Default Trigger Associations .............................................................. 86
Table 5-3  Available Track Types ....................................................................... 89

Table A-1  Difference in Output Primitives ......................................................... 103

Table B-1  Unsupported Control Functions ......................................................... 107
Table B-2  Unsupported Input Functions ............................................................. 107
Table B-3  Non Standard Control Functions ...................................................... 108
Table B-4  Non Standard Attribute Functions ................................................... 108

Table D-1  Possible Causes of Visual Errors ....................................................... 132
Table D-2  Primitive-Specific Errors ................................................................ 133
Table D-3 Attribute Errors .................................................. 134
Table D-4 Input-specific Errors ........................................... 134

Table F-1 List of cgipw Functions ...................................... 145
Table F-2 SunCGI Functions not Compatible with cgipw Mode 147

Table G-1 SunCGI Fortran Binding – Part I ......................... 154
Table G-2 SunCGI Fortran Binding – Part II ......................... 157
Table G-3 SunCGI Fortran Binding – Part III ....................... 160
Table G-4 SunCGI Fortran Binding – Part IV ....................... 166
Table G-5 SunCGI Fortran Binding – Part V ......................... 168

Table H-1 Correspondence Between Long and Short C Names 173
Figures

Figure 1-1  Simple Example Program ................................................................. 4
Figure 2-1  Example Program with Multiple Workstations .......................... 12
Figure 2-2  Example Program with Multiple Normalization
Transformations .................................................................................................. 18
Figure 2-3  Example Program with set_up_sigwinch Function ............ 24
Figure 3-1  Example Program with Polygons .................................................. 37
Figure 3-2  Example Program with Four Circle Quadrants in Different
Colors .................................................................................................................. 40
Figure 4-1  Example Program with Bundled Attributes ............................. 55
Figure 4-2  Example Program with Bundled Attributes ............................. 64
Figure 5-1  CGI Input State Model .................................................................... 84
Figure 5-2  Example Program with LOCATOR Input Device ..................... 91
Figure 5-3  Example Program with STRING Input Device ......................... 94
Figure E-1  Martini Glass Example Program .................................................. 138
Figure E-2  Tracking Box Example Program .................................................. 139
Figure F-1  Example cgipw Program ............................................................... 148
Figure G-1  Example FORTRAN Program ....................................................... 153
Preface

This document describes SunCGI, an implementation of the ANSI Computer Graphics Interface (CGI) by Sun Microsystems, Inc. Previously, CGI was known as the Virtual Device Interface (VDI) standard. Appendix B summarizes the differences between SunCGI and ANSI CGI.

The CGI standard is currently under development. Future releases of SunCGI will reflect changes in ANSI CGI.

Controlling Document
The following document was used in interpreting the CGI standard:


Audience
The intended reader of this document is an applications programmer who is familiar with interactive computer graphics and the C programming language. This manual contains several example programs that can be used as templates for larger SunCGI applications.

Documentation Conventions
*Italic font* is used to indicate file names, function arguments, variables and internal states of SunCGI. Italics are also used in the conventional manner (to emphasize important words and phrases). *ALL CAPS* is used to indicate values in enumerated types. *Bold font* is used for the names of Sun software packages. Function names are printed with *constant width font.*
# Introduction

## 1. Using SunCGI

## 1.2. The SunCGI Lint Library

## 1.3. Overview of SunCGI

### Initialization and Termination

### Output Primitives

### Attributes

### Input

### Errors

### Programming Tips

### Appendices

## 1.4. References
Introduction

SunCGI provides access to low-level graphics device functions without the restrictions, benefits, or overhead of higher-level graphics packages like SunCore. SunCGI is useful for 2D graphics programs which do not require segmentation or transformations. The absence of segmentation from SunCGI makes drawing diagrams faster and simpler, but does not provide automatic picture regeneration. SunCGI programs are usually smaller and more efficient than SunCore programs with similar functionality. In addition, SunCGI programs will run on Sun devices without explicitly specifying the device at compile time. SunCGI provides output primitives (for example, circles), attributes (for example, sophisticated pattern filling), and input primitives which are not offered by SunCore. The CGI standard is currently under development, and therefore, CGI has not been accepted by the X3H3 committee, ANSI, or the computer graphics community. Only certain models within CGI are supported by SunCGI. Specifically SunCGI implements input option sets 1, 2, 3, 4, and 6 and output option sets 1 through 6 of the CGI standard. CGI does not support 3D output primitives.

SunCGI does provide output primitives, attribute selection, and input device management, at a level which is close to the actual device driver; thus affording speed and flexibility not offered by higher-level graphics packages like SunCore. SunCGI provides output primitives which are not provided by any of the other Sun graphics packages: for example disjoint polygons, circles, ellipses, and cell arrays (which can be thought of as scaled and transformed pixel arrays). CGI also provides a larger vocabulary of attributes than SunCore. SunCGI also provides facilities for explicitly binding virtual input devices to physical input devices as well as explicit management of an event queue.

1.1. Using SunCGI

Here is a SunCGI example application program written in C:
```c
#include <cgidefs.h>

Ccoor box[5] = { 10000,10000 ,
    10000,20000 ,
    20000,20000 ,
    20000,10000 ,
    10000,10000 };

main ()
{
    Ccoorlist boxlist;
    Cint name;
    Cvwsurf device;

    boxlist.n = 5;
    boxlist.ptlist = box;
    NORMAL_VWSURF(device, PIXWINDD);

    open_cgi();
    open_vws(&name, &device);

    polyline(&boxlist);
    sleep(10);

    close_vws(name);
    close_cgi();
}
```

Figure 1-1  Simple Example Program

SunCGI uses a variety of structures and enumerated types shown in Appendix C. The file <cgidefs.h> should be included in each SunCGI application program to provide necessary definitions and constants.

Here is an example of a command line for compiling box.c to run in the SunView environment:

```
% cc box.c -o box -lcgi -lsunwindow -lpixrect -lm
```

The order in which the libraries are linked to the program is important.

All SunCGI functions can be called by one of two names: the expanded name (default) or the C language binding name. See Appendix H for information on the list of names for the shorter C language binding.

As a final note, do not name any user-defined function or variable starting with the letters _cgi because doing so may disrupt the internal workings of SunCGI.

FORTRAN programmers can access SunCGI functions by using the include file in cgidefs77.h and using the /usr/lib/libcgi77.a library to link with. Details of the FORTRAN interface to SunCGI are provided in Appendix G.
1.2. The SunCGI Lint Library

SunCGI provides a lint library which provides type checking beyond the capabilities of the C compiler. For example, you could use the SunCGI lint library to check a program called glass.c with command like this:

% lint glass.c -lcgi

Note that the error messages that lint generates are mostly warnings, and may not necessarily have any effect on the operation of the program. For a detailed explanation of lint, see the lint chapter in the Programming Tools manual.

1.3. Overview of SunCGI

This section provides an overview of the substance of this manual. The four major sections of the manual (which correspond to chapters) are:

1) view surface initialization and termination (control),
2) output primitives,
3) attributes, and
4) input.

The overview of these chapters contains a brief introduction to the basic concepts of CGI. The appendices at the end of this manual provide quick reference tables and descriptions of the interfaces between SunCGI and

1) SunView and
2) FORTRAN.

Initialization and Termination

Chapter 2 describes functions for

1) initializing and terminating the entire SunCGI package and individual view surfaces,
2) defining the coordinate systems,
3) interface negotiation, and
4) signal trapping.

The first section Chapter 2 describes functions for opening and closing view surfaces (which are either windows or screens). SunCGI provides facilities for writing primitives to multiple view surfaces. Output primitives can be written to a selected subset of the open view surfaces by using the activate_vws and deactivate_vws functions (which turn a view surface on or off without closing the view surface or affecting the display). The functions discussed in Chapter 2 also define the range of virtual device coordinates (VDC space) and device coordinates (screen space). The coordinates of most SunCGI functions are expressed in terms of VDC space. The limits of both VDC space and screen space can be defined by the application program.

If you are attempting to run an application program developed on another vendor’s version of CGI, negotiation functions are provided which describe the capabilities of SunCGI. The application program can use the information obtained by using the negotiation functions to call appropriate functions in...
SunCGI to make the application program run correctly. Finally, Chapter 2 describes SunCGI's option for trapping SIGWINCH signals (generated by manipulating the window environment which the application program is using).

Output Primitives

SunCGI provides functions for drawing geometrical output primitives (for example, polygons, circles, and ellipses) as well as functions for performing raster operations. The coordinates of output primitives are specified in VDC space (with the exception of some raster functions). Geometrical output primitives include rectangles, polymarkers, circular and elliptical arcs. Geometrical output primitives are affected by attributes described in Chapter 4 (like fill style and line width). All output primitives are affected by the drawing mode which determines how an output primitive is affected by pixels which have been previously drawn on the screen.

Attributes

Attribute functions control the appearance of output primitives. Attributes can be set individually, or in groups which are called bundles. The use of most attributes is fairly straightforward; fill textures require a word of explanation. Geometrical output primitives can be filled with textures called hatches or patterns. Hatches are simply arrays of color values with each element of the array corresponding to a pixel. Patterns are arrays of color values which can be scaled and translated.

Input

SunCGI offers a standard interface for receiving input from the mouse and the keyboard. The CGI input model is based on the logical input device model in GKS. In this system, a logical input device (for example, a LOCATOR device), is bound to a physical device (for example, the x-y position of the mouse) called a trigger. Triggers may be associated with logical input devices by the application program. Each logical input device has an associated measure (for example, the measure of a LOCATOR device is the mouse position on the screen). Each logical input device also has a state which determines how a device handles input. Each logical input device can be in one of five states:

1) RELEASED (uninitialized),
2) NO_EVENTS (initialized but unable to receive input),
3) REQUEST_EVENT (waiting for one event),
4) RESPOND_EVENT (report one event asynchronously), and
5) QUEUE_EVENT (put each event at the end of the event queue).

Errors

Errors are reported in SunCGI by setting the return value of the function to a nonzero result and echoing an error message and number on the terminal. However, error trapping can be controlled by the set_error_warning_mask function. An explanation of each error message (and suggestions for how to eliminate them) is presented in Appendix D.
Programming Tips

For novice C language users, the syntax of SunCGI may pose some initial difficulties. When a pointer is specified as an argument to a SunCGI function, SunCGI usually expects space to be allocated by the application program and the function argument to be preceded by an ampersand (&). SunCGI uses many enumerated types. These types are printed by the printf function as integers. If you want to print out these values in English, you should use the enumerated types as indices into a character array which contains appropriate English equivalents of the enumerated types. Finally, if you are a novice programmer, copy the example programs in Appendix E and use them as templates to build your own program with. Further help can be obtained by referring to the tables at the end of Appendix D. These tables list commonly encountered problems and how to solve them.

Appendices

The first five appendices are intended to make SunCGI easier to understand. This information will probably be particularly useful to novice users. The last two appendices describe the interfaces:

1. between SunCGI and SunView, and
2. between SunCGI and the FORTRAN programming language.

Appendix A explains the difference between SunCGI and SunCore. Appendix B lists the ANSI CGI standard functions which are not implemented by SunCGI and the SunCGI functions which are not part of the ANSI CGI standard. Appendix C provides the type definitions used by the SunCGI functions. Appendix D lists the error messages and possible strategies for eliminating them. Appendix D also lists possible causes of simple run-time errors. Appendix E describes sample programs.

The final two appendices describe the interfaces between SunCGI and other Sun software packages: SunView and FORTRAN. The first of the two interface appendices explains how to call SunCGI from application programs written on top of SunView. This interface allows SunCGI to write output primitives in different windows using different attributes. This interface is useful for application programs which wish to control different areas of the view surface independently. Appendix G describes the interface to the FORTRAN programming language. The behavior of each SunCGI function is the same in both C and FORTRAN.

1.4. References


Initializing and Terminating SunCGI

Initializing and Terminating SunCGI ................................................................. 11

2.1. View Surface Initialization and Selection .................................................. 11
    Open CGI (SunCGI Extension) ................................................................. 12
    Open View Surface (SunCGI Extension) ................................................ 13
    Activate View Surface (SunCGI Extension) .......................................... 16
    Deactivate View Surface (SunCGI Extension) ..................................... 16
    Close View Surface (SunCGI Extension) ............................................. 16
    Close CGI (SunCGI Extension) .............................................................. 16

2.2. View Surface Control ................................................................................ 17
    VDC Extent ......................................................................................... 17
    Device Viewport .................................................................................. 19
    Clip Indicator ...................................................................................... 19
    Clip Rectangle ...................................................................................... 20
    Hard Reset ........................................................................................... 20
    Reset to Defaults ................................................................................ 20
    Clear View Surface ............................................................................. 21
    Clear Control ....................................................................................... 21
    Set Error Warning Mask .................................................................... 22

2.3. Running SunCGI with SunView ................................................................. 22
    Set Up SIGWINCH (SunCGI Extension) .............................................. 23

2.4. Interface Negotiation ................................................................................ 24
    Inquire Device Identification ............................................................ 25
    Inquire Device Class .......................................................................... 25
Inquire Physical Coordinate System ............................................................... 25
Inquire Output Function Set ............................................................................. 26
Inquire VDC Type ............................................................................................... 26
Inquire Output Capabilities .............................................................................. 27

2.5. Input Capability Inquiries .......................................................................... 27
   Inquire Input Capabilities ............................................................................... 27
   Inquire LID Capabilities .................................................................................. 28
   Inquire Trigger Capabilities ............................................................................ 29
2. Initializing and Terminating SunCGI

The current CGI standard does not provide functions for initializing and terminating devices. ANSI CGI is intended to provide an interface for a single view surface (one per CGI instance). SunCGI extends CGI into the window environment by allowing a single CGI process to control multiple view surfaces. Six nonstandard functions open.cgi, close.cgi, open_vws, close_vws, activate_vws, and deactivate_vws are included in SunCGI.

open.cgi and close.cgi initialize and terminate the operation of the SunCGI package. A view surface is initialized and terminated with open_vws and close_vws. A view surface is automatically activated when it is opened. SunCGI is capable of handling more than one view surface at once. Output primitives can be restricted from a view surface with deactivate_vws.

2.1. View Surface Initialization and Selection

A view surface is automatically activated when it is opened. However, a view surface can be deactivated (with the deactivate_vws function) when the output stream is not intended to appear on all view surfaces. Subsequent calls to SunCGI output functions will not apply to deactivated view surfaces until activate_vws is called again (see the following example).

---

1 However, inputs can be received on deactivated view surfaces.
```c
#include <cgidefs.h>

main()
{
    Coor bot, top, center;
    Cint name1, name2, radius;
    Cvwsurf device1, device2;

    bot.x = 5000;
    bot.y = 5000;
    center.x = 10000;
    center.y = 10000;
    radius = 5000;
    top.x = 15000;
    top.y = 15000;

    open_cgi();
    NORMAL_VWSURF(device1, PIXWINDD);
    open_vws(&name1, &device1);
    NORMAL_VWSURF(device2, PIXWINDD);
    open_vws(&name2, &device2);

    rectangle(&bot, &top);
    deactivate_vws(name2);
    circle(&center, radius);
    activate_vws(name2);
    circle(&center, 2*radius);

    sleep(20);

    close_vws(name1);
    close_vws(name2);
    close_cgi();
}
```

**Figure 2-1** Example Program with Multiple Workstations

**Open CGI (SunCGI Extension)**

Cerror open_cgi()

open_cgi initializes the state of SunCGI to CGOP (CGi OPen). open_cgi does not initialize input devices but does initialize the event queue. No other CGI functions can be used without generating an error if open_cgi has not been called. SunCGI traps various signals as described in Section 2.3.

**Errors**

ENOTCGCL [1] CGI not in proper state: CGI shall be in state CGCL.
You may be unfamiliar with some of the entries discussed in Table 2-1. However, these concepts are explained in the course of this chapter. Further, each of these concepts are referenced in the index.

Open View Surface (SunCGI Extension)

Cerror open_vws(name, devdd)
Cint *name; /* name assigned to cgi view surface */
Cvwsurf *devdd; /* view surface descriptor */

open_vws initializes a view surface. The list of available view surfaces is described below in Table 2-2. open_vws initializes the attributes to their default values (listed in Table 2-3). The returned argument name is the identifier which is used to refer this view surface in other SunCGI functions. To reinitialize the state of the view surface without reopening it, use the hard_reset function.

More than one view surface can be open at one time. Output primitives are displayed on all active view surfaces (view surfaces must be opened before they are activated). However, input is only echoed on the view surface which is pointed to by the mouse. Most of the Cvwsurf fields should be zeroed, as by the NORMAL_VWSURF macro. Set the view surface type by assigning the dd (device driver) element of the devdd argument to the name of the appropriate device driver as in this example:

Cvwsurf device;
NORMAL_VWSURF(device, BW2DD);
open_vws(&name, &device);

Note: The NORMAL_VWSURF macro initializes the dd element of the Cvwsurf structure and guarantees that the view surface will be opened in the normal fashion. However, to open a window with some nonstandard parameters, or open a second window from a graphics tool read the following paragraphs. To use an existing pixwin, then skip the following paragraphs and read Appendix F instead.

---

2 Notice that when SunCGI specifies a pointer it usually requires that the argument is prefaced by a & character when the argument is actually used.
If the view surface of the specified type has been previously initialized and the type of view surface is a window (PIXWINDD or CGPIXWINDD), a CGI tool (a window with the name CGI Tool) is opened. Other characteristics of the view surface can be defined by setting the other elements of the of the devdd argument (which is of type Cvwsurf).

```c
typedef struct {
    char screenname[DEVNAMESIZE]; /* physical screen */
    char windowname[DEVNAMESIZE]; /* window */
    int windowfd; /* window file descriptor */
    int retained; /* retained flag */
    int dd; /* device */
    int cmapsize; /* color map size */
    char cmapname[DEVNAMESIZE]; /* color map name */
    int flags; /* new flag */
    char **ptr; /* CGI tool descriptor */
} Cvwsurf;
```

The elements screenname and windowname specify alternate screens (for example, /dev/cgone0) or alternate window (for example, /dev/win10). If these elements are left blank, the current screen and the current window are used, unless the dd field implicitly specifies a device (for example CG1DD). The element windowfd is the window file descriptor for the current device. The current implementation of SunCGI ignores this element.

If the element retained is nonzero, then the view surface created by open_vws has a retained window associated with it (that is, if the window is covered up by another window and then revealed, the picture present before the window was covered-up will be redisplayed. By default the window created by open_vws is non-retained. That is, if the window is covered-up and then revealed the covered-portion will be redisplayed as white. However, drawing in non-retained windows is twice as fast as drawing in retained windows, so the choice of which type of view surface to open should be carefully considered.

The dd element specifies the view surface type. The cmapsize and the cmapname elements determine the size and the name of the colormap. No colormap is enabled for monochrome devices. The colormap determines the mapping between color indices and red, green, and blue values. If the colormap specified by the cmapname element of the devdd argument is the same as a colormap segment which already exists, then the colormap segment is shared. cmapsize should be a power of two, less than or equal to 256. Refer to the SunView Programmer’s Guide for more information about colormaps.

When the flags element is nonzero, no attempt is made to take over the current graphics subwindow (if one exists). If this flag is set or the graphics subwindow has already been taken over by SunCGI, then a CGI Tool (a window with the name View Surface Tool) is created. The ptr element specifies the size and placement of the CGI Tool. ptr is a pointer to an array of characters which should consist of nine decimal numbers separated by commas. The array takes the following form:

"nl,nt,nw,nh,il,it,iw,ih,I"
Each element of the array should be filled with an integer. The first two elements specify the $x$ and $y$ coordinates of the upper left-hand corner of the CGI Tool. The third and fourth elements specify the width and height of the CGI Tool. The fifth through eighth elements specify the position and size of the iconic form of the CGI Tool. If the ninth element is nonzero, the tool is displayed in its iconic form.

**Errors**

- **ENOTOPPOP [5]** CGI not in proper state CGI shall be either in state CGOP, VSOP, or VSAC.
- **ENOWSTYP [11]** Specified view surface type does not exist.
- **EMAXVSOP [12]** Maximum number of view surfaces already open.
- **EMEMSPAC [110]** Space allocation has failed.
- **ENOTCCPW [112]** Function or argument not compatible with standard CGI.

**Table 2-2** Available View Surfaces

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIXWINDD</td>
<td>SunView on a monochrome display</td>
</tr>
<tr>
<td>CGPIXWINDD</td>
<td>SunView on a color display</td>
</tr>
<tr>
<td>BW1DD</td>
<td>Full screen on a Sun-1 monochrome display</td>
</tr>
<tr>
<td>BW2DD</td>
<td>Full screen on a Sun-2 or Sun-3 monochrome display</td>
</tr>
<tr>
<td>CG1DD</td>
<td>Full screen on a Sun-1 color display</td>
</tr>
<tr>
<td>CG2DD</td>
<td>Full screen on a Sun-2 or Sun-3 color display</td>
</tr>
<tr>
<td>GP1DD</td>
<td>Full screen on a Sun-2/160 or Sun-3/160 with optional Graphics Processor</td>
</tr>
</tbody>
</table>

**Table 2-3** View Surface Default States

<table>
<thead>
<tr>
<th>State</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>View Surface</td>
<td>Cleared</td>
</tr>
<tr>
<td>Device Viewport</td>
<td>View Surface</td>
</tr>
</tbody>
</table>

*Note:* most failures during the opening of a view surface result in error ENOWSTYP [11]. The most common reason is missetting (or failing to set) the $dd$ element of the Cvwsurf structure. For example, opening a device surface type PIXWINDD instead of CGPIXWINDD on a color pixwin, or using CG2DD when the /dev/cg2w* surface is being used by suntools. The NORMAL_VWSURF macro should be used to initialize this structure.
Activate View Surface (SunCGI Extension)

Cerror activate_vws(name)
Cint name; /* view surface name */

activate_vws activates the view surface specified by name. Subsequent SunCGI calls affect this view surface. Nothing is displayed on a view surface unless that view surface is active. Since a view surface is active as soon as it is opened, activate_vws is only need to reactivate a deactivated view surface. Note that activating a view surface may reset the state of SunCGI.

Errors
ENOTOPOP [5] CGI not in proper state CGI shall be either in state CGOP, VSOP, or VSAC.
EVSSIDINV [10] Specified view surface name is invalid.
EVSSISACT [14] Specified view surface is active.

Deactivate View Surface (SunCGI Extension)

Cerror deactivate_vws(name)
Cint name; /* view surface name */
deactivate_vws prevents calls to SunCGI functions from having an effect on this view surface. The view surface may be reactivated by activate_vws at a later time without having to be reopened. Note that deactivating a view surface may reset the state of SunCGI.

Errors
ENOTVSAC [4] CGI not in proper state: CGI shall be in state VSAC.
EVSSIDINV [10] Specified view surface name is invalid.
EVSSNTACT [15] Specified view surface is not active.

Close View Surface (SunCGI Extension)

Cerror close_vws(name)
Cint name; /* view surface name */
close_vws terminates a view surface. Future SunCGI calls have no effect on this view surface. The view surface cannot be reactivated without being reope ned.

Errors
ENOTOPOP [5] CGI not in proper state CGI shall be either in state CGOP, VSOP, or VSAC.
EVSSIDINV [10] Specified view surface name is invalid.
ENOTCCPW [112] Function or argument not compatible with standard CGI.

Close CGI (SunCGI Extension)

Cerror close_cgi()
close_cgi terminates all open view surfaces, and restores the state of the SunView to the state that it was in before SunCGI was opened. Future SunCGI calls will have no effect and will generate errors.

© sun microsystems

Version B of 17 February 1986
A call to `close_cgi` should be included in the exit routines of an application program to guarantee leaving the SunView and SunCGI in a stable state.

### Errors

- **ENOTOPPOP [5]**
  
  CGI not in proper state. CGI shall be either in state CGOP, VSOP, or VSAC.

- **ENOTCCPW [112]**
  
  Function or argument not compatible with standard CGI.

### 2.2. View Surface Control

The functions described in this section

1. define the range of world and device coordinates,
2. control clipping, and
3. reset selected aspects of the view surface and the internal state of SunCGI.

Most functions in SunCGI express coordinates in VDC space (Virtual Device Coordinate space). In conventional computer graphics terms, VDC space corresponds to world coordinate space. The mapping between VDC space and screen space is determined by the physical size of the screen in pixels. Screen space is set by default to the entire size of the screen or the graphics window depending on the device type. The mapping from VDC space to screen space is always isotropic (the shape of the rectangle defining screen space is the same shape as VDC space). Therefore, VDC space defines the shape of the active view surface. The portion of screen space which does not correspond to VDC space is ignored. The aspect ratio (the ratio between the height and width) is therefore, defined by VDC space and not screen space.

#### VDC Extent

```c
Cerror vdc_extent(cl, c2)
Ccoor *cl, *c2; /* bottom left-hand and */
/* top right-hand corner of VDC space */
```

`vdc_extent` defines the limits of VDC space. The range of the coordinates must be between $-32767$ and $32767$ (or an error is generated). VDC space can be set by the application program, but it ranges from $0$ to $32767$ in both the $x$ and the $y$ directions by default. Resetting VDC space impacts the display of output primitives on all view surfaces.

Resetting the limits of VDC space automatically redefines the clipping rectangle to the new limits of VDC space, regardless of the value of the `clip indicator`.

Changing the mapping from screen space to VDC space allows for translation (move) or scaling (zoom in/zoom out) of output primitives. However, no rotation functions are provided by SunCGI, and therefore, must be supplied in the application program. The code fragment below translates and zooms in on a rectangle:
```c
#include <cgidefs.h>

main()
{
    Cvwsurf device;
    Cint name;
    Ccoor dv1, dv2, lower, upper;

    NORMAL_VWSURF(device, PIXWINDD);
    dv1.x = 0;
    dv1.y = 0;
    dv2.x = 200;
    dv2.y = 200;
    lower.x = 30;    /* rectangle coordinates */
    lower.y = 30;
    upper.x = 70;
    upper.y = 70;

    open_cgi();
    open_vws(&name, &device);
    vdc_extent(&dv1, &dv2);

    rectangle(&upper, &lower); /* draw initial rectangle */
    sleep(4);
    dv1.x = 0;
    dv1.y = 0;
    dv2.x = 100;
    dv2.y = 100;
    vdc_extent(&dv1, &dv2); /* center rectangle */
    rectangle(&upper, &lower);
    sleep(4);
    dv1.x = 20;
    dv1.y = 20;
    dv2.x = 80;
    dv2.y = 80;
    vdc_extent(&dv1, &dv2); /* enlarge rectangle */
    rectangle(&upper, &lower);
    sleep(20);

    close_vws(name);
    close_cgi();
}
```

Figure 2-2  Example Program with Multiple Normalization Transformations

Errors

- **ENOTOPPOP [5]**  CGI not in proper state  CGI shall be either in state CGOP, VSOP, or VSAC.
- **EBADRCTD [20]**  Rectangle definition is invalid.
Device Viewport

**Errors**

- **ENOTOPOP [5]** - CGI not in proper state. CGI shall be either in state `CGOP`, `VSOP`, or `VSAC`.
- **EVSIDINV [10]** - Specified view surface name is invalid.
- **EVSNOTOP [13]** - Specified view surface not open.
- **EBADRCTD [20]** - Rectangle definition is invalid.
- **EBDVIEWP [21]** - Viewport is not within Device Coordinates.
- **ENOTCCPW [112]** - Function or argument not compatible with standard CGI.

**Clip Indicator**

**Errors**

- **ENOTOPOP [5]** - CGI not in proper state. CGI shall be either in state `CGOP`, `VSOP`, or `VSAC`.
- **EVSIDINV [10]** - Specified view surface name is invalid.
- **EVSNOTOP [13]** - Specified view surface not open.
- **EBADRCTD [20]** - Rectangle definition is invalid.
- **EBDVIEWP [21]** - Viewport is not within Device Coordinates.
- **ENOTCCPW [112]** - Function or argument not compatible with standard CGI.

**Device Viewport**

```c
Cerror device_viewport(name, cl, c2)
Cint name; /* name assigned to cgi view surface */
Ccoor *cl, *c2; /* bottom left-hand and top right-hand */
    /* corner of view surface to map device onto */
    /* (expressed in pixels) */
```

device_viewport redefines the limits of screen space. If the new limits are not less than or equal to the size of the current screen or window size, an error is returned. Although `device_viewport` does not redefine the aspect ratio, it may redefine which areas of the screen are unused.

**Clip Indicator**

```c
Cerror clip_indicator(cflag)
Cclip cflag; /* CLIP, NOCLIP or CLIP_RECTANGLE */
```

For some application programs, it is desirable to clip explicitly within the viewport, while other applications may seek to increase efficiency by not checking if the coordinates are within the bounds of the clipping area.

All SunCGI application programs will run faster if clipping is turned off. However, clipping is turned on by default to prevent SunCGI from drawing outside of the bounds of the window.

The extent of VDC may be set with the `vdc_extent` function.

The value of the argument `cflag` determines whether output primitives are clipped before they are displayed. The default state is `CLIP`. The advantage of turning clipping off is that it improves the speed of drawing primitives. However, if clipping is set to `NOCLIP`, SunCGI may draw output primitives outside of the window or within the bounds of an overlapping window. If clipping is not `NOCLIP`, output primitives are clipped to either the clip rectangle (if `cflag` equals `CLIP_RECTANGLE`), or the full extent of VDC space (if `cflag` equals `CLIP`).

```c
typedef enum {
    CLIP,
    NOCLIP,
    CLIP_RECTANGLE
} Cclip;
```
Errors

ENOTOPPOP [5]  CGI not in proper state CGI shall be either in state CGOP, VSOP, or VSAC.
ENOTCCPW [112] Function or argument not compatible with standard CGI.

Clip Rectangle

Cerror clip_rectangle(xmin, xmax, ymin, ymax)  
Cint xmin, xmax, ymin, ymax; /* bottom left-hand */
  /* and top right-hand corner of clipping rectangle */

clip_rectangle defines the clipping rectangle in VDC Coordinates. By default, the clipping rectangle is set to the borders of VDC space. The clip_rectangle function defines the clipping rectangle in VDC space, to be used when clipping is set to CLIP_RECTANGLE. The clipping rectangle is automatically reset by vdc_extent.

Errors

ENOTOPPOP [5]  CGI not in proper state CGI shall be either in state CGOP, VSOP, or VSAC.
EBADRCTD [20]  Rectangle definition is invalid.
ECLIPPTOL [22]  Clipping rectangle is too large.
ECLIPPTOS [23]  Clipping rectangle is too small.
ENOTCCPW [112] Function or argument not compatible with standard CGI.

Hard Reset

Cerror hard_reset()  

Device control functions restore the view surface and the internal state of SunCGI to a known state. The individual aspects of the device which can be reset are the output attributes, the view surface (screen), and the error reporting.

hard_reset returns the output attributes to their default values; terminates all input devices, and empties the event queue and clears all view surfaces. VDC space is reset to its default values and the clip indicator is set to CLIP. This function should be used sparingly because most control, attribute, and input functions called before this function will not have any effect on functions called after hard_reset is called.

Errors

ENOTOPPOP [5]  CGI not in proper state CGI shall be either in state CGOP, VSOP, or VSAC.

Reset to Defaults

Cerror reset_to_defaults()  

reset_to_defaults returns output attributes to defaults (see Table 4-1). reset_to_defaults does not clear the screen, reset the input devices, or reset the character set index.

Errors

ENOTOPPOP [5]  CGI not in proper state CGI shall be either in state CGOP, VSOP, or VSAC.
EVSIDINV [10]  Specified view surface name is invalid.
Clear View Surface

Cerror clear_view_surface(name, defflag, index)
Cint name; /* name assigned to cgi view surface */
Cflag defflag; /* default color flag */
Cint index; /* color of cleared screen */
clear_view_surface changes all pixels in the relevant area of the view surface specified by name to the color specified by the index argument, unless the defflag argument is set to OFF. If defflag is equal to OFF, the view surface is cleared to color zero. The area of the view surface which is actually cleared is determined by the clear_control function. clear_view_surface also resets the internal state of SunCGI according to previous calls to the clear_control function. clear_view_surface resets the current background color to the color of the cleared view surface.

Errors

ENOTVSAC [4] CGI not in proper state: CGI shall be in state VSAC.
EVSIDINV [10] Specified view surface name is invalid.
EVSNCTAT [15] Specified view surface is not active.
ECINDXLZ [35] Color index is less than zero.
EBADCOLX [36] Color index is invalid.

Clear Control

Cerror clear_control(soft, hard, intern, extent)
Cacttype soft, hard; /* soft and hard copy actions */
Cacttype intern; /* internal action */
Cexttype extent; /* clear extent */
clear_control determines the action taken when clear_view_surface is called. The argument soft can be set to either NO_OP or CLEAR. The argument hard which regulates clearing rules for plotters is ignored (because SunCGI does not currently support hard-copy devices) and is included only for ANSI CGI compatibility. The argument intern is set to either RETAIN or CLEAR. This parameter was included to support segmentation storage which is not currently a part of ANSI CGI. Therefore, the intern argument is ignored. The argument extent determines what area of the screen is cleared. It is set to one of the values in the Cexttype enumerated type:

typedef enum {
    CLIP_RECT,
    VIEWPORT,
    VIEWSURFACE
} Cexttype;

Errors

ENOTOP [5] CGI not in proper state CGI shall be either in state CGOP, VSOP, or VSAC.
ENOTCCPW [112] Function not compatible with CGIPW mode.

Set Error Warning Mask
errors

set_error_warning_mask\(^3\) determines the action taken by SunCGI when an error occurs. Three types of action are possible: NO\_ACTION, POLL, INTERRUPT. If the action argument is set to NO\_ACTION, errors are detected internally, but not reported. The error number is returned to the caller of a CGI routine. The user is advised not to set the action argument to NO\_ACTION.

POLL and INTERRUPT actions print an error message on the terminal, but also return the error number (see Appendix D) so the program can perform exception handling. The default error_warning_mask is INTERRUPT.

Enotopop [5] CGI not in proper state CGI shall be either in state CGOP, VSOP, or VSAC.

<table>
<thead>
<tr>
<th>Error Warning Mask</th>
<th>Message Printed</th>
<th>Program Aborted</th>
<th>Error Number Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO_ACTION</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>POLL</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>INTERRUPT</td>
<td>Yes</td>
<td>FATAL errors†</td>
<td>Non-FATAL errors</td>
</tr>
</tbody>
</table>

† SunCGI defines no errors as FATAL. All errors are non-fatal so the application has complete control to abort or perform other processing as desired. Therefore, POLL and INTERRUPT are the same in SunCGI.

2.3. Running SunCGI with SunView

SunCGI always traps five signals: SIGINT, SIGCHLD, SIGIO, SIGHUP and SIGWINCH. The first four of these cause SunCGI cleanup and program termination. When using a Graphics Processor option, SunCGI also traps SIGXCPU. Previous signal handlers, if any, are saved. When one of these signals occurs, SunCGI's signal handler will call the previous signal handler as well as performing its own processing. The actions of the previous (user installed) signal handler may interfere with SunCGI's signal responses, and are hence unsupported.

Unless a SunCGI application program has opened a retained view surface, overlapping another window onto a graphics subwindow will destroy the picture below. SunCGI programs can regenerate a display surface by trapping the SIGWINCH (SIGnal WINdow CHange) signal.

It is possible (though unsupported) to install a signal handler for signals after calling open\_pw\_cgi (see Appendix F). Since these signal handlers replace SunCGI's handler, the application should save SunCGI's signal handler (returned by signal), and call the saved handler when the signal occurs (amid the user's own processing). Because the response of the program to the signal then depends on the place in the user's own signal handling that SunCGI's handler is

\(^3\) The syntax of set_error_warning_mask in SunCGI is slightly different from the proposed ANSI standard in that the ANSI definition allows different actions for different classes of errors.
Set Up SIGWINCH (SunCGI Extension)

Called, results are unpredictable, and may change with a new version of SunCGI.

Note that it is not necessary for an application to catch a SIGWINCH signal, since SunCGI's set_up_sigwinch routine offers an easier interface. A user's sig_function has a different calling semantics from a SIGWINCH in that pw_damaged and pw_donedamaged have already been invoked.

When a window's contents needs regeneration during execution time, the process associated with a window receives a SIGWINCH signal. The application can use this signal to determine when a view surface needs to be regenerated. Note: Under no circumstances will the user be able to access the SIGWINCH signals generated when a view surface is initialized.

When a window obstructs a SunCGI view surface, output to that view surface is normally clipped to the exposed portion only (unless the clip indicator is NOCLIP). When the obstruction is removed, unless the window is RETAINED, the picture must be regenerated by re-running the output generation of the applications, for that view surface at least. An application's SIGWINCH handling function is called for this purpose.

When a SunCGI window's size changes during execution, the picture must be regenerated. But first, SunCGI updates the transformation used to map VDC space into screen space. Then, if the affected view surface is RETAINED, the retained copy is rewritten onto the view surface. (Because of the size change, this may not repair the damage satisfactorily.) Lastly, the application's SIGWINCH function is called.

Cerror set_up_sigwinch(name, sig_function)
Cint name;
Cint (*sig_function)(); /* signal handling function */

set_up_sigwinch allows the application program to trap SIGWINCH signals for view surface name. sig_function is a pointer to a function returning an integer. If sig_function is nonzero, all SIGWINCH signals which are not trapped by the internals of SunCGI (from view surface initialization) are passed to the function specified by sig_function.

The sig_function is called when the SIGWINCH signal is received. It is the programmer's responsibility to use a flag to determine if it is safe to process the signal at this time, or to set a flag indicating that signal processing has been put off until later. See the SunView Programmer's Guide for information on SIGWINCH handling.

The sig_function argument is called with a single argument: the name of the view surface with which it is associated by the call to set_up_sigwinch. This allows more than one view surface to share the same sig_function, and differentiate which view surface needs redisplay.

Here is an example of a program that uses set_up_sigwinch.
Figure 2-3  Example Program with set_up_sigwinch Function

Errors

ENOTOPPOP [5]  CGI not in proper state CGI shall be either in state CGOP, VSOP, or VSAC.

2.4. Interface Negotiation

CGI is intended to support a 'negotiated device interface' which permits programs written on a specific type of hardware to run on other machines. SunCGI only allows inquiry of most of the settable modes.\(^4\) For example the user may want to find out which types of input devices are supported. However, functions for setting color precision, coordinate type, specification mode, and color specification are not provided because SunCGI only supports one type of color precision (8-

\(^4\) The functions which are not supported by SunCGI are classified as non-required by the March 1984 ANSI C header standard. See Appendix B.
bit), coordinate type (integers), and color specification (indexed). The width and size specification modes are settable, but the functions which set them are described in Chapter 4. However, the inquiry negotiation functions are supported so that an application program written for a CGI on another manufacturers’ workstation can find out whether the SunCGI is capable of running that application.

**Inquire Device Identification**

```c
Cerror inquire_device_identification(name, devid)
Cint name; /* device name */
Cchar devid[DEVMESIZE]; /* workstation type */
```

inquire_device_identification reports which type of Sun Workstation view surface `name` is associated with. The argument `devid` may be set to one of the Sun Workstation types described in Table 2-2. The inclusion of the `name` argument deviates from the ANSI standard, but is necessary so that the characteristics of individual view surfaces may be inquired.

**Errors**

- ENOTOP [5] CGI not in proper state CGI shall be either in state CGOP, VSOP, or VSAC.
- EVSIDINV [10] Specified view surface name is invalid.

**Inquire Device Class**

```c
Cerror inquire_device_class(output, input)
Cint *output, *input; /* output and input abilities */
```

inquire_device_class describes the capabilities of Sun Workstations in terms of the CGI functions they support. Each of the two returned values reports the number of functions of each of the two classes which are supported in SunCGI. These numbers (the values of `input` and `output`) are used to make more detailed inquiries by using functions inquire_input_capabilities and inquire_output_capabilities.

**Errors**

- ENOTOP [5] CGI not in proper state CGI shall be either in state CGOP, VSOP, or VSAC.

**Inquire Physical Coordinate System**

```c
Cerror inquire_physical_coordinate_system(name, xbase, ybase, xext, yext, xunits, yunits)
Cint name; /* name assigned to cgi view surface */
Cint *xbase, *ybase; /* base coordinates */
Cint *xext, *yext; /* pixels in x and y directions */
Cfloat *xunits, *yunits; /* number of pixels per mm. */
```

inquire_physical_coordinate_system reports the physical dimensions of the coordinate system of view surface `name` in pixels and millimeters. inquire_physical_coordinate_system is provided to permit the drawing of objects of a known physical size.

inquire_physical_coordinate_system is also provided to assist in

---

5 The `output` argument does not include the non-standard CGI functions.
the computation of parameters for the `device_viewport` function. *xext* and *yext* describe the maximum extent of the window in which the application program is run. (The window may or may not cover the entire screen.) The number of pixels per millimeter is always set to 0 because the actual screen size of device varies between individual monitors. The actual size of the screen may be obtained from the number of pixels in the *x* and *y* directions from the monitor specifications and perform the division in an application program.

**Errors**

- **ENOTOPROP [5]** CGI not in proper state CGI shall be either in state CGOP, VSOP, or VSAC.
- **EVSIDINV [10]** Specified view surface name is invalid.
- **EVSNOTOP [13]** Specified view surface not open.

**Inquire Output Function Set**

```c
Cerror inquire_output_function_set(level, support)
Cint level; /* level of output */
Csuptype *support; /* amount of support */
```

`inquire_output_function_set` reports the extent to which each level of the output portion of the ANSI CGI standard is supported.

```c
typedef enum {
    NONE,
    REQUIRED_FUNCTIONS_ONLY,
    SOME_NON_REQUIRED_FUNCTIONS,
    ALL_NON_REQUIRED_FUNCTIONS
} Csuptype;
```

The standard requires that the `level` argument be an enumerated type; however, for reasons of simplicity only the level number is used by SunCGI. Levels 1-6 are supported completely (that is, both required and non-required functions are implemented. Level 7 is not supported at all. Refer to the ANSI standard for the precise definition of each level.

**Errors**

- **ENOTOPROP [5]** CGI not in proper state CGI shall be either in state CGOP, VSOP, or VSAC.

**Inquire VDC Type**

```c
Cerror inquire_vdc_type(type)
Cvdctype *type; /* type of VDC space */
```

`inquire_vdc_type` reports the type of coordinates used by SunCGI in the returned argument `type`.

```c
typedef enum {  
    INTEGER,
    REAL,
    BOTH
} Cvdctype;
```

`type` is always set to INTEGER (32-bit). SunCore is a higher-level graphics system with coordinate space expressed in real numbers.

**Errors**
ENOTOPPOP [5] CGI not in proper state CGI shall be either in state CGOP, VSOP, or VSAC.

**Inquire Output Capabilities**

Cerror inquire_output_capabilities(first, num, list)
Cint first; /* first element */
Cint num; /* number of elements in list to be returned */
Cchar *list[]; /* returned list */

`inquire_output_capabilities` lists the output functions in the returned argument `list`. The range of the `first` and `num` arguments is determined by the returned argument `output` from the `inquire_device_class` function.

**Errors**

ENOTOPPOP [5] CGI not in proper state CGI shall be either in state CGOP, VSOP, or VSAC.

EINQLTL [16] Inquiry arguments are longer than list.

### 2.5. Input Capability Inquiries

Input devices have a separate class of negotiation functions. Input capability inquiries report qualitative abilities as well as quantitative abilities of input devices. The `inquire_input_capabilities` function reports which devices and overall features are supported by SunCGI. The remaining functions report the capabilities of individual devices or features. Input devices are virtual devices which must be associated with physical triggers (such as mouse buttons). Initializing an input device defines the measure used by a device, for example initializing a LOCATOR device defines the measure as x-y coordinates. In addition to being associated with a trigger, each device has selectable screen echoing capabilities. Association and echoing capabilities for each input device are reported by the functions described in this section.

**Inquire Input Capabilities**

Cerror inquire_input_capabilities(valid, table)
Clogical *valid; /* device state */
Ccgidesctab *table; /* CGI input description table */

`inquire_input_capabilities` reports the total number of input devices of each class that are supported. The argument `valid` returns the value L_TRUE if SunCGI is initialized, and L_FALSE otherwise. If `valid` is set to L_TRUE, the elements of `table` are set to the quantity and quality of inputs supported. All Sun Workstations support input at the same level.
typedef struct {
    Cint numloc;
    Cint numval;
    Cint numstrk;
    Cint numchoice;
    Cint numstr;
    Cint numtrig;
    Csuptype event_queue;
    Csuptype asynch;
    Csuptype coord_map;
    Csuptype echo;
    Csuptype tracking;
    Csuptype prompt;
    Csuptype acknowledgement;
    Csuptype trigger_manipulation;
} Ccgidesctab;

Elements of type Cint report how many of each type device is supported, as well as how many types of triggers are supported. Elements of type Csuptype report how many of the functions of each class are supported. All functions except the tracking functions are fully supported.

Errors

ENOTOPPOP [5]     CGI not in proper state CGI shall be either in state CGOP, VSOP, or VSAC.

Inquire LID Capabilities

Cerror inquire_lid_capabilities(devclass, devnum, valid, table)
Cdevoff devclass;
Cint devnum; /* device number */
Clogical *valid; /* device supported at all */
Cliddescript *table; /* table of descriptors */

inquire_input_device_capabilities describes the capabilities of a specific input device (hereafter, specified device). The input arguments devclass and devnum refer to a specific device type and number. The argument valid reports whether CGI is initialized.

typedef struct {
    Clogical sample;
    Cchangetype change;
    Cint numassoc;
    Cint *trigassoc;
    Cliddescript prompt;
    Cliddescript acknowledgement;
    Cechoav *echo;
    Cchar *classdep;
    Cstatelist state;
} Cliddescript;

The elements of table which are of type Clogical indicate whether an ability is present in the specified logical input device. The change element reports whether associations are changeable at all (all input devices except string are changeable). The numassoc and trigassoc elements of table report how many
and which triggers may be associated with the specified logical input device. The *echo* argument describes which echo types are supported (see Chapter 5 for a list of echo types). The *classdep* argument provides class dependent information in character form (the type of information is given in Table 2-3). If more than one piece of class dependent information is returned, then the pieces of information are separated by commas. The *state* argument reports the initial state of the specified device. See the *inquire_state_list* function.

<table>
<thead>
<tr>
<th>Device Class</th>
<th>Information</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC_LOCATOR</td>
<td>Coordinate Mapping</td>
<td>Yes, No, Partial</td>
</tr>
<tr>
<td></td>
<td>Native Range</td>
<td>xmin, xmax,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ymin, ymax</td>
</tr>
<tr>
<td>IC_VALUATOR</td>
<td>Set Valuator Range</td>
<td>yes/no</td>
</tr>
<tr>
<td>IC_STROKE</td>
<td>Time Increment Settable</td>
<td>yes/no</td>
</tr>
<tr>
<td></td>
<td>Minimum Distance</td>
<td>yes/no</td>
</tr>
<tr>
<td>IC_CHOICE</td>
<td>Range</td>
<td>min/max</td>
</tr>
<tr>
<td>IC_STRING</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

**Errors**

ENOTOPPOP [5] CGI not in proper state CGI shall be either in state CGOP, VSOP, or VSAC.

**Inquire Trigger Capabilities**

Cerror inquire_trigger_capabilities(trigger, valid, tdis)
Cint trigger; /* trigger number */
Clogical *valid; /* trigger supported at all */
Ctrigdis *tdis; /* trigger description table */

inquire_trigger_capabilities describes how a particular *trigger* can be associated. The argument *valid* reports whether the device supports input at all.

typedef struct {
    Cchangetype change;
    Cassociclid *numassoc;
    Cint maxassoc;
    Cpromstate prompt;
    Cackstate acknowledgement;
    Cchar *name;
    Cchar *description;
} Ctrigdis;

The *change* element of *tdis* reports whether the specified trigger can be associated with a logical input device. The *numassoc* element of *tdis* gives supported LID associations for this trigger. This consists of *n*, the number of LID classes which can be associated with the trigger, a pointer to an array of *n* entries telling which *n* device classes can be associated with the trigger, and how many of each

---

6 Note that inquire_lid_capabilities returns an enumerated type whereas track_on accepts integers. Therefore these values may be different.
device class is defined. The \textit{maxassoc} field gives the number of LID's which can be concurrently associated with this trigger. SunCGI does not support either prompt or acknowledgement for any input device. The \textit{name} element is simply a character form of the trigger name (for example, \texttt{LEFT MOUSE BUTTON}). The \textit{description} element is never filled and is included for standards compatibility.

\begin{tabular}{ll}
\textbf{Errors} & \\
ENOTOPOP [5] & CGI not in proper state CGI shall be either in state CGOP, VSOP, or VSAC. \\
EINTRNEX [86] & Trigger does not exist. \\
\end{tabular}
Output

3.1. Geometrical Output Primitives

- Polyline ........................................................................................................... 33
- Disjoint Polyline ............................................................................................... 34
- Polymarker ......................................................................................................... 34
- Polygon .............................................................................................................. 34
- Partial Polygon ................................................................................................. 35
- Rectangle .......................................................................................................... 35
- Circle ................................................................................................................ 35
- Circular Arc Center ........................................................................................... 36
- Circular Arc Center Close ................................................................................ 37
- Circular Arc 3pt ............................................................................................... 37
- Circular Arc 3pt Close ..................................................................................... 37
- Ellipse ............................................................................................................. 37
- Elliptical Arc ................................................................................................... 37
- Elliptical Arc Close ......................................................................................... 38

3.2. Raster Primitives

- Text ................................................................................................................. 38
- VDM Text ......................................................................................................... 38
- Append Text .................................................................................................... 39
- Inquire Text Extent .......................................................................................... 39
- Cell Array ........................................................................................................ 39
- Pixel Array ...................................................................................................... 40
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BitBlt Source Array</td>
<td>45</td>
</tr>
<tr>
<td>BitBlt Pattern Array</td>
<td>46</td>
</tr>
<tr>
<td>BitBlt Patterned Source Array</td>
<td>46</td>
</tr>
<tr>
<td>Inquire Cell Array</td>
<td>47</td>
</tr>
<tr>
<td>Inquire Pixel Array</td>
<td>47</td>
</tr>
<tr>
<td>Inquire Device Bitmap</td>
<td>48</td>
</tr>
<tr>
<td>Inquire BitBlt Alignments</td>
<td>48</td>
</tr>
</tbody>
</table>

3.3. Drawing Modes

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set Drawing Mode</td>
<td>49</td>
</tr>
<tr>
<td>Set Global Drawing Mode (SunCGI Extension)</td>
<td>50</td>
</tr>
<tr>
<td>Inquire Drawing Mode</td>
<td>50</td>
</tr>
</tbody>
</table>
SunCGI supports two classes of output primitives: geometrical output primitives and raster primitives.

**Geometrical Output Primitives**
include arcs, circles, polylines, and polygons. The position of geometrical output primitives are always specified in absolute VDC coordinates.7

**Raster Primitives**
draw text and scaled and unscaled 2D arrays. The coordinate system for raster primitives depends on the type of primitive. The drawing mode determines how output primitives are drawn on top of other output primitives or the background.

Geometrical output primitives are divided into two classes: polygonal primitives and conical primitives. Geometrical output primitives are all 2D in keeping with the CGI standard. However, polygons with holes (via the partial_polygon function) are provided in order to support 3D graphics packages.

Geometrical primitives (except polymarker) are considered either closed or not closed. Polymarker uses its own attributes (see Section 4.3). Non-closed figures (polylines, circular arcs, or elliptical arcs) are drawn with a style, width and color determined from line attributes (see Section 4.2). Closed figures (polygons, rectangles, circles, ellipses, and circular and elliptical closed arcs) use the solid object attributes (see Section 4.4). The geometrical information specifies the boundary of a closed figure. The interior of this boundary is filled using fill area attributes. The boundary may be surrounded with a line, drawn with perimeter attributes, not the line attributes. For example, a circle of radius 1000 and a perimeter width of 100 VDC units has its perimeter between the circle of radius 1000 and a concentric circle of radius 1100 (not from 950 through 1050).

Most polygonal primitives (polyline, polymarker, polygon, and partial_polygon) take one argument of type Coorlist:

---

7 SunCGI (unlike SunCore) maintains no concept of current position.
typedef struct {
    Cint x;
    Cint y;
} Ccoor;

typedef struct {
    Ccoor *ptlist;
    Cint n;
} Ccoorlist;

The element ptlist is really a pointer to an array of type Ccoor which contains
the n coordinates of the points defining the primitive. The style, color, and other
features of lines, markers, and fill patterns used by geometrical output primitives
are set by the attribute functions described in Chapter 4.

The polygons generated by SunCGI may or may not be closed. SunCGI
automatically assumes the polygon is closed for the purpose of filling. However,
a polygon must be explicitly closed in order to get all of its edges drawn, so take
care to generate explicitly closed polygons. The rectangle function implicitly
generates closed objects.

SunCGI has two classes of conical primitives: circular and elliptical. Each class
has functions for drawing solid objects, arcs, and closed arcs. Drawing of conical
primitives is regulated by the same attributes that regulate the drawing of
polygons and polylines.

Polyline

cerror polyline(polycoors)
Ccoorlist *polycoors; /* list of points */

polyline draws lines between the points specified by the ptlist element of
polycoors. polyline does not draw a line between the first and last element of the
point list. To generate a closed polyline, the last point on the list must have the
same coordinates as the first point on the list. The style, color, and width of the
lines are set by the polyline_bundle_index, line_type,
line_color, line_width and line_width_specification_mode
functions. If a line segment of a polyline has a length of zero, the line is not
drawn. To draw a point, use the circle function. If you specify a polyline
that has less than two points, an error is generated. Similarly, if the number of
points specified is greater than the maximum number of points (MAXPTS) an error
is generated.

Errors

ENOTVSAC [4] CGI not in proper state: CGI shall be in state VSAC.
ENMPTSTL [60] Number of points is too large.
EPLMTWPT [61] polylines must have at least two points.

Disjoint Polyline

---

8 A closed portion of a closed figure boundary will not be drawn if it exceeds a clipping boundary.
Chapter 3 — Output

Cerror disjoint_polyline(polycoors)
Ccoorlist *polycoors; /* list of points */

disjoint_polyline draws lines between pairs of elements in ptiist. The
line attributes described in Section 4.2 determine the appearance of the
disjoint_polyline function. If polycoors contains an odd number of
points, the last point is ignored. As with polyl ine, if the number of points is
less than two or greater than MAXPTS, an error is generated.
disjoint_polyline is typically used to implement scan-line polygon filling
algorithms.

Errors

ENOTVSAC [4] CGI not in proper state: CGI shall be in state VSAC.
ENMPTSTL [60] Number of points is too large.
EPLMTWPT [61] polylines must have at least two points.

Polymarker

Cerror polymarker(polycoors)
Ccoorlist *polycoors; /* list of points */

polymarker draws a marker at each point. The type, color, and size of marker
are set by the polymarker_bundle_index,marker_type,
marker_color,marker_size, and
marker_size_specification_mode functions. If the number of points
specified is greater than the maximum number of points, an error is generated.
polymarker is useful for making graphs such as scatter plots.

Errors

ENOTVSAC [4] CGI not in proper state: CGI shall be in state VSAC.
ENMPTSTL [60] Number of points is too large.

Polygon

Cerror polygon(polycoors)
Ccoorlist *polycoors; /* list of points */

polygon displays the polygon described by the points in polycoors. In addition,
any points added to the global_polygon list by the partial_polygon func-
tion are also displayed. The polygon is filled between edges. Polygons are
allowed to be self-intersecting. The visibility of individual edges can only be set
by the partial_polygon function. The style and color used to fill the
polygon are set by the solid object attribute functions described in Chapter 4.
The characteristics of the edges are controlled by the perimeter attribute func-
tions. The number of points in the polygon used to determine the error condition
of too few or too many points is the total number of points on the global_polygon
list, not the number of points specified in polycoors. After the polygon is drawn,
the global_polygon list is emptied.

Errors

ENOTVSAC [4] CGI not in proper state: CGI shall be in state VSAC.
ENMPTSTL [60] Number of points is too large.
EPGMTHPT [62] Polygons must have at least three points.
Global polygon list is full.

Partial Polygon

Cerror partial_polygon(polycoors, cflag)
Ccoorlist *polycoors; /* list of points */
Ccflag cflag; /* CLOSE previous polygon? */

partial_polygon adds elements to the global polygon list without displaying the polygon. The partial_polygon function provides the capability of drawing multiple-boundary polygons, including polygons with holes. The drawing is actually performed when polygon is called. polygon will close the last boundary on the global polygon list and add the coordinate list it is passed as the final polygon boundary before drawing.

cflag controls whether the last polygon in the global polygon list is open or closed. If cflag is set to CLOSE, the last polygon on the global polygon list will be closed by drawing a visible perimeter edge between the last and the first points of the last polygon on the global polygon list. If the cflag is set to OPEN, the points in polycoors are appended to the last polygon on the global polygon list, but an invisible perimeter edge will be drawn between the last point currently on the global polygon list and the first point in the Ccoorlist. The visibility of polygon edges can be individually controlled by calling partial_polygon with cflag set to OPEN for each invisible edge and with cflag set to CLOSE for each new boundary. The interpretation of cflag is slightly different than the pseudocode given in the CGI standard. Future versions of CGI may use a different syntax to offer the capabilities of multiple-boundary polygons and invisible edges.

The CGI standard specifies that circle, rectangle, ellipse and close_arc are primitives that may use the global polygon list for filling. SunCGI does not use the global polygon list in these functions, and therefore leaves it untouched. These SunCGI routines do not empty the global polygon list.
Figure 3-1  Example Program with Polygons

An error is detected if the number of points on the global polygon list exceeds MAXPTS. In this case, the polygon on the global polygon list is drawn, and the new information is not added. The same error handling applies to polygon.

Errors
Rectangle

Cerror rectangle(rbc, ltc)
Ccoor *rbc, *ltc; /* corners defining rectangle */

rectangle displays a box with its lower right-hand corner at point rbc and its upper left-hand corner at point ltc. Calls to rectangle do not affect the global polygon list. The interior of the rectangle (the filled portion) is defined by rbc and ltc. The perimeter is drawn outside of this region. The appearance of the rectangle is determined by the fill area and perimeter attributes. A rectangle with one side coincident with a clipping boundary specifies an interior extending to the boundary. Hence, a portion of the perimeter is outside the clipping boundary and is not drawn.

If the arguments to rectangle would result in a point or a line, the point or line is drawn. However, if the arguments to rectangle determine a point, the point is drawn with width zero, regardless of the current value of perimeter width. If the values of rbc and ltc are reversed, the points are automatically reversed and the rectangle is drawn normally.

Errors

ENOTVSAC [4]  CGI not in proper state: CGI shall be in state VSAC.

Circle

Cerror circle(cl, rad)
Ccoor *cl; /* center */
Cint rad; /* radius */

circle draws a circle of radius rad centered at cl. The argument rad is expressed in terms of VDC space. The color, form, and visibility of the interior and perimeter are controlled by the same solid object attributes which control the drawing of polygons and rectangles.

The argument rad determines the size of the interior of the circle. Therefore, a circle with a thick perimeter may be larger than expected. If the radius is zero, a point is drawn, and no textured perimeter is drawn, even if the perimeter width is large. If the radius is negative, the absolute value of the radius is used.

Textured circles may possibly contain an incorrect element at one point because the digital circumference may not be exactly divisible by the length of the texture element.

Errors

ENOTVSAC [4]  CGI not in proper state: CGI shall be in state VSAC.

Circular Arc Center

Cerror circular_arc_center(cl, c2x, c2y, c3x, c3y, rad)
Ccoor *cl; /* center */
Cint c2x, c2y, c3x, c3y; /* endpoints */
Cint rad; /* radius */
circular_arc_center draws a circular arc between points \(c2x, c2y\) and \(c3x, c3y\) with circle of radius \(rad\) at center \(cl\). Point \(c2x, c2y\) is the starting point and point \(c3x, c3y\) is the ending point. Circular arcs are drawn in a counterclockwise manner. This convention is used to determine the difference between the arc formed by the smaller angle determined by \(c2x, c2y, cl\) and \(c3x, c3y\) and the larger angle specified by these same points. Therefore switching the values of \(c2x, c2y\) and \(c3x, c3y\) will produce arcs which total 360 degrees. If \(rad\) is negative, the points 180 degrees opposite from \(c2x, c2y\) and \(c3x, c3y\) are used as the endpoints of the arc.

If the \(rad\) is zero, a point is drawn at \(cl\). If either \(c2x, c2y\) or \(c3x, c3y\) are not on the circumference of the circle determined by \(cl\) and \(rad\), an error is generated and the arc is not drawn. The attributes which determine the style, width, and color of the arc are the same functions which regulate the drawing of polylines.

Errors

- ENOTVSAC [4] CGI not in proper state: CGI shall be in state VSAC.
- EARCPNCl [64] Arc points do not lie on circle.

Circular Arc Center Close

```c
Cerror circular_arc_center_close(cl, c2x, c2y, c3x, c3y, rad, close)
Ccoor *cl; /* center */
Cint c2x, c2y, c3x, c3y; /* endpoints */
Cint rad; /* radius */
Cclosetype close; /* PIE or CHORD */
```

circular_arc_center_close draws a closed arc centered at \(cl\) with radius \(rad\) and endpoints \(c2x, c2y\) and \(c3x, c3y\). Arcs are closed with either the PIE or CHORD algorithm. The PIE algorithm draws a line from each of the endpoints of the arc to the center point of the circle. SunCGI then fills this region as it would any other solid object. The CHORD algorithm draws a line connecting the endpoints of the arc and then fills this region using solid object attributes.

circular_arc_center_close is useful for drawing pie charts (see following example):
```c
#include <cgidefs.h>

main() /* draws four quadrants in different colors */
{
    Ccoor c1;
    Cint name, radius;
    Cvwsurf device;

    c1.x = 16000; /* center */
    c1.y = 16000;
    NORMAL_VWSURF(device, CGPIXWINDD);
    radius = 8000; /* radius */

    open_cgi();
    open_vws(&name, &device);

    interior_style(SOLIDI, OFF);
    fill_color(1); /* color of quadrant 1 */
    circular_arc_center_close(&c1, 24000, 16000, 16000, 24000, radius, PIE);
    fill_color(2); /* color of quadrant 2 */
    circular_arc_center_close(&c1, 16000, 24000, 8000, 16000, radius, PIE);
    fill_color(3); /* color of quadrant 3 */
    circular_arc_center_close(&c1, 8000, 16000, 16000, 8000, radius, PIE);
    fill_color(4); /* color of quadrant 4 */
    circular_arc_center_close(&c1, 16000, 8000, 24000, 16000, radius, PIE);

    sleep(10);
    close_vws(name);
    close_cgi();
}
```

Figure 3-2  
**Example Program with Four Circle Quadrants in Different Colors**

**Errors**

- ENOTVSAC [4]: CGI not in proper state: CGI shall be in state VSAC.
- EARCPNCI [64]: Arc points do not lie on circle.

**Circular Arc 3pt**

Cerror circular_arc_3pt(c1, c2, c3)

Ccoor *c1, *c2, *c3; /* starting, intermediate and ending points */

circular_arc_3pt draws a circular arc starting at point c1 and ending at point c3 which is guaranteed to pass through point c2. The line attributes functions described in Section 4.2 determine the appearance of the circular_arc_3pt function. If the circular arc is textured (for example, dotted) then the intermediate point may not be displayed. However, if the arc is solid, the intermediate point is always drawn. If the three points are colinear, a
line is drawn. If two of the three points are coincident, a line is drawn between
the two distinct points. Finally, if all three points are coincident, a point is
drawn. \texttt{circular\_arc\_3pt} is considerably slower than
\texttt{circular\_arc\_center}, therefore, you are advised to
\texttt{circular\_arc\_center} if both functions can meet your needs.

**Errors**

\texttt{ENOTVSAC [4]} \hspace{1em} CGI not in proper state: CGI shall be in state VSAC.

**Circular Arc 3pt Close**

\texttt{Cerror\ circular\_arc\_3pt\_close(cl, c2, c3, close)}
\texttt{Ccoor *cl, *c2, *c3; /* starting, intermediate
and ending points */
Cclosetype close; /* PIE or CHORD */
circular\_arc\_3pt\_close draws a circular arc starting at point \texttt{cl} and
ending at point \texttt{c3} which is guaranteed to pass through point \texttt{c2}. The solid object
attributes described in Section 4.4 determine the appearance of the
circular\_arc\_3pt\_close function. As with \texttt{circular\_arc\_3pt},
circular\_arc\_3pt\_close is considerably slower than
circular\_arc\_center\_close; therefore, you are advised to use
circular\_arc\_center\_close if both functions meet your needs.

If the three points are colinear, a line is drawn. If two of the three points are
coincident, a line is drawn between the two distinct points. Finally, if all three
points are coincident, a point is drawn. In none of these cases will any region be
filled.

**Errors**

\texttt{ENOTVSAC [4]} \hspace{1em} CGI not in proper state: CGI shall be in state VSAC.

**Ellipse**

\texttt{Cerror\ ellipse(cl, majx, miny)}
\texttt{Ccoor *cl; /* center */
Cint majx; /* length of x and y axes */
ellipse} draws an ellipse centered at point \texttt{cl} with major (x) and minor (y) axes of
length \texttt{majx} and \texttt{miny}.\footnote{Although the axes are called the major and minor axes by the standard they are really the x and y axes. In fact, the x axis can either be the major or minor axis, depending on the relative length of the y axis.} If either \texttt{majx} or \texttt{miny} are zero, a line is drawn. If both
\texttt{majx} and \texttt{miny} are zero, a point is drawn. The attributes which control the draw­
ing of ellipses are the solid object attributes described in Section 4.4.

**Errors**

\texttt{ENOTVSAC [4]} \hspace{1em} CGI not in proper state: CGI shall be in state VSAC.

**Elliptical Arc**

\texttt{Cerror\ elliptical\_arc(cl, sx, sy, ex, ey, majx, miny)}
\texttt{Ccoor *cl; /* center */
Cint sx, sy; /* starting point of arc */
Cint ex, ey; /* ending point of arc */
Cint majx, miny; /* endpoints of major and minor axes */
elliptical\_arc draws an elliptical arc centered at \texttt{cl} with major (x) and
minor (y) axes of length \texttt{majx} and \texttt{miny}. \texttt{sx}, \texttt{sy} and \texttt{ex}, \texttt{ey} are the starting and
Errors

ENOTVSAC [4]  CGI not in proper state: CGI shall be in state VSAC.
EARCPNEL [65]  Arc points do not lie on ellipse.

Elliptical Arc Close

cerr error elliptical arc close(cl, sx, sy, ex, ey, majx, miny, close)
ccoor *cl; /* center */
cint sx, sy; /* starting point of arc */
cint ex, ey; /* ending point of arc */
cint majx, miny; /* endpoints of major and minor axes */
cclosetype close; /* PIE or CHORD */

elliptical arc close draws an elliptical arc specified by sx, sy, ex, ey
and majx, miny The arc is closed with either the PIE or CHORD algorithm. The
same restrictions on sx, sy, ex, and ey are applied to
elliptical arc close as to elliptical arc. However,
elliptical arc close uses the fill area and perimeter attributes, whereas
elliptical arc uses the line attributes.

If either majx or miny are zero, a line is drawn. If both majx and miny are zero, a
point is drawn. In neither of these cases will any region be filled.

Errors

ENOTVSAC [4]  CGI not in proper state: CGI shall be in state VSAC.
EARCPNEL [65]  Arc points do not lie on ellipse.

3.2. Raster Primitives

Raster primitives include text, cell arrays, pixel arrays, and bitblts (bit block
transfer). Bitblts are pixel arrays (bitmaps) which can be drawn using the various
drawing modes. The current drawing mode determines how bitblt primitives are
affected by information which is already on the screen. Raster primitives differ
from geometrical primitives because their dimensions are not necessarily
expressed in VDC space. Therefore, you must be careful to consider whether
position arguments are expressed in VDC space or screen coordinates.

Text

cerror text(cl, tstring)
ccoor *cl; /* starting point of text (in VDC space) */
cchar *tstring; /* text */

text displays the text contained in tstring at point cl (expressed in VDC space).
The appearance of text is controlled by the text attributes described in Section
4.8. Control characters are displayed as blanks, except in the SYMBOL font where they may be drawn as pictures of bugs.

Errors
ENOTVSAC [4] CGI not in proper state: CGI shall be in state VSAC.

VDM Text
Cerror vdm_text(cl, flag, tstring)
Ccoor *cl; /* starting point of text (in VDC space) */
Ctextfinal flag; /* final text for alignment */
Cchar *tstring; /* text */

vdm_text displays the text contained in tstring at point cl (expressed in VDC space). The intended difference between text and vdm_text is that vdm_text allows control characters; however, SunCGI does not handle control characters so text drawn with vdm_text will appear identical to text drawn with the text function. If the flag argument is equal to FINAL, the previous text and the appended text are aligned separately. However, if the flag argument is equal to NOT_FINAL, the appended and previous text are aligned together.

Errors
ENOTVSAC [4] CGI not in proper state: CGI shall be in state VSAC.

Append Text
Cerror append_text(flag, tstring)
Ctextfinal flag; /* final text for alignment */
Cchar *tstring; /* text */

append_text displays the text contained in tstring after the end of the most recently written text. The type of text written depends on the same attributes which control the display of text. The flag argument determines whether the appended text is aligned with the previous text if the alignment is CONTINUOUS. If the flag argument is equal to FINAL, then the previous text and the appended text are aligned separately. However, if the flag argument is equal to NOT_FINAL, the appended and previous text are aligned together.

Errors
ENOTVSAC [4] CGI not in proper state: CGI shall be in state VSAC.

Inquire Text Extent
Cerror inquire_text_extent(tstring, nextchar, concat, lleft, uleft, uright)
Cchar *tstring; /* text */
Cchar nextchar; /* next character (for kerning) */
Ccoor *concat; /* concatenation point */
Ccoor *lleft, *uleft, *uright;
/* coordinates of text bounding box */
inquire_text_extent determines how large text tstring would be and where it would be placed if it were drawn using the current text attributes. The nextchar parameter is used to determine the point where text would start if more text (starting with nextchar) were appended to the text specified by tstring. If nextchar equals 'single space', the last point of the current character is used. The argument concat returns the coordinates of the point where appended text

\[10\] This is a method for accounting for proportional spacing.
would start. The arguments \texttt{lleft}, \texttt{uleft}, and \texttt{uright} return three of the four corners of the bounding box of text contained in \texttt{tstring}.

The bounding box is a parallelogram (a rectangle if the character up vector and the character base vector are orthogonal). The names of the parallelogram corners are correct if no rotation is applied to the text. For some character orientations, the implied relationships do not hold. For example, \texttt{lleft} may not be the lowest. The fourth corner may be easily calculated from the three returned:

\begin{align*}
\texttt{uright->x} + \texttt{lleft->x} - \texttt{uleft->x} \\
\texttt{uright->y} + \texttt{lleft->y} - \texttt{uleft->y}
\end{align*}

The concatenation point and text alignment parallelogram are returned in VDC space, but assume a text position of (0, 0). If the text is to be drawn at a position (x,y) then (x,y) must be added to each point to yield the true locations.

The values of \texttt{lleft}, \texttt{uleft}, and \texttt{uright} are defined by the bounding box of the character and therefore may not be at the exact pixel where the character ends or begins.

\textbf{Errors}

\begin{itemize}
  \item ENOTVSAC [4] \quad \text{CGI not in proper state: CGI shall be in state VSAC.}
\end{itemize}

\textbf{Cell Array}

\begin{verbatim}
cerror \ cell\ array(p, q, r, dx, dy, colorind)
   Ccolor *p, *q, *r;
   /* corners of parallelogram (in VDC space) */
   Cint dx, dy; /* dimensions of color array */
   Cint *colorind; /* array of color values */

   cell\ array draws a scaled and skewed pixel array on the view surface(s).
   Points \texttt{p}, \texttt{q}, and \texttt{r} (expressed in VDC space) define a parallelogram. Line \texttt{p-q} is a diagonal and \texttt{p} is the lower left-hand corner. \texttt{r} is one of the remaining two corners. \texttt{dx} and \texttt{dy} define the width and the height of the array \texttt{colorind} which is mapped onto the parallelogram defined by \texttt{p}, \texttt{q}, and \texttt{r}.

   cell\ array is one of the few primitives which depends on the actual size of
   the view surface. Cell arrays are not drawn if the elements of the array would be
   smaller than one pixel. However, because different view surfaces may have different
   dimensions, a cell array might be drawn on one view surface, but not on
   another smaller view surface. Finally, all cells composing the cell array are the
   same size; therefore, the upper left hand corner of the cell array might be down
   and to the right of point \texttt{q} because of the accumulated error of making all of the
   cells slightly smaller than their floating point size. For example if each cell of a
   3 \times 3 cell array is supposed to be 3.333 pixels wide, the actual cell array will be
   nine pixels wide instead of ten.
\end{verbatim}

\textbf{Errors}

\begin{itemize}
  \item ENOTVSAC [4] \quad \text{CGI not in proper state: CGI shall be in state VSAC.}
  \item ECELLATS [66] \quad \text{Cell array dimensions \texttt{dx}, \texttt{dy} are too small.}
  \item ECELLPOS [67] \quad \text{Cell array dimensions must be positive.}
\end{itemize}

\textbf{Pixel Array}
Cerror pixel_array(pcell, m, n, colorind)
Coor *pcell; /* base of array in VDC space */
Cint m, n; /* dimensions of color array in screen space */
Cint *colorind; /* array of color values */

pixel_array draws array colorind starting at point pcell (expressed in VDC space). m and n (expressed in screen space) define the x and y dimensions of the array. Therefore, pixel arrays always have a constant physical size, independent of the dimensions of VDC space. The pixel array is drawn down and to the right from point pcell. If either m or n are not positive, the absolute value of m and n are used. pixel_array is not affected by the current drawing mode.

Errors

ENOTVSAC [4] CGI not in proper state: CGI shall be in state VSAC.
EVALOVWS [69] Value outside of view surface.

BitBlt Source Array

Cerror bitblt_source_array(pixsource, xo, yo, xe, ye, pixtarget, xt, yt, name)
Cpixrect *pixsource, *pixtarget;
/* source and target pixel arrays */
Cint xo, yo;
/* coordinates of source array (in VDC space) */
Cint xe, ye;
/* dimensions of source array (in screen space) */
Cint xt, yt;
/* coordinates of target pixel array (in VDC space) */
Cint name; /* view surface name */

bitblt_source_array moves a pixel array from point (xo, yo) to point (xt, yt) using the current drawing mode. Both of these points are expressed in VDC space. The size of the pixel array is determined by the xe and ye arguments which are expressed in screen space. pixsource and pixtarget are pointers to pixrects which must already be created by mem_create.11 These pixrects must be the same depth as the view surface: 1-bit deep on a monochrome device, 8-bit on a color device. The source area of the view surface associated with name is saved into pixsource (at 0,0). The target area, after pixsource is applied to it, is read into pixtarget pixrect (at 0,0).

An error is detected if either xe or ye are not positive. If the replicated pattern array overlaps with the source array on the screen, the visual result depends on the current drawing mode. pixsource and pixtarget may have different contents depending on the screen drawing mode (see the set_drawing_mode function).

Multiple view surfaces and bitblt's are incompatible, so a name argument must be specified.

Errors

ENOTVSAC [4] CGI not in proper state: CGI shall be in state VSAC.

11 Refer to the Pixrect Reference Manual for more information about pixrects.
BitBlt Pattern Array

Cerror bitblt_pattern_array(pixpat, px, py, pixtarget, rx, ry, ox, oy, dx, dy, name)
Cpixrect *pixpat; /* pattern source array */
Cint px, py; /* pattern extent */
Cpixrect *pixtarget; /* destination pattern array */
Cint rx, ry; /* pattern reference point */
Cint ox, oy; /* destination origin */
Cint dx, dy; /* destination extent */
Cint name; /* view surface name */

bitblt_pattern_array replicates the pattern (using the current drawing mode) stored in pixpat to fill the area of the view surface which is determined by ox, oy and dx, dy. The pattern reference point determines the offset of the pattern array from the point zero. The resultant pattern array is displayed at ox, oy. The visual result depends on the current drawing mode.

pixpat is a pointer to a pixrect which must be created and initialized with the pattern by the application program. pixtarget is a pointer to a pixrect (with same depth as the device) which must already be created by the user, using mem_create. The target area, after pixpat is applied to it, is read into the pixtarget pixrect (at 0,0).

Multiple view surfaces and bitblt's are incompatible, so a name argument must be specified.

Errors

ENOTVSAC [4] CGI not in proper state: CGI shall be in state VSAC.
EVALOVWS [69] Value outside of view surface.
EPXNOTCR [70] Pixrect not created.

BitBlt Patterned Source Array

Cerror bitblt_patterned_source_array(pixpat, px, py, pixtarget, rx, ry, pixsource, sx, sy, ox, oy, dx, dy, name)
Cpixrect *pixpat; /* pattern source array */
Cint px, py; /* pattern extent */
Cpixrect *pixsource; /* source array */
Cint sx, sy; /* source origin */
Cpixrect *pixtarget; /* destination pattern array */
Cint rx, ry; /* pattern reference point */
Cint ox, oy; /* destination origin */
Cint dx, dy; /* destination extent */
Cint name; /* view surface name */

bitblt_patterned_source_array replicates (using the current drawing mode) the pattern stored in pixpat to fill the area of the view surface determined by ox, oy and dx, dy. The source area of the view surface is read into the pixrect pointed to by pixsource (which must already be created by the user with same depth as the device) at 0,0. The source area is stenciled through the replicated pattern onto the view surface at ox, oy, using the current drawing mode. The target area, after the copy, is read into the pixtarget pixrect. If
the replicated pattern array overlaps with the source array on the screen, the visual result depends on the current drawing mode.

Multiple view surfaces and bitblt’s are incompatible, so a name argument must be specified.

Errors

ENOTVSAC [4] CGI not in proper state: CGI shall be in state VSAC.
EVALOVWS [69] Value outside of view surface.
EPXNOTCR [70] Pixrect not created.

Inquire Cell Array

Cerror inquire_cell_array(name, p, q, r, dx, dy, colorind)
Cint name; /* view surface name */
Ccoor *p, *q, *r;
    /* corners of parallelogram (in VDC space) */
Cint dx, dy; /* dimensions of color array */
Cint *colorind; /* array of color values */

Points \( p, q \) and \( r \) (in VDC space) define a parallelogram with line \( p-q \) as the diagonal where \( p \) is the lower left-hand corner. \( r \) is one of the remaining two corners. \( dx \) and \( dy \) define the width and the height of the array \( colorind \) which contains the colors of the pixels on the screen which lie within the parallelogram defined by \( p, q, \) and \( r \). Notice that a view surface identifier, \( name \), must be specified because the result of this function is highly dependent on the dimensions and contents of the view surface.

The area of the screen corresponding to the parallelogram is assumed to contain a regular grid of points. However, if each element of the grid is larger than one pixel, the color of the pixel at lower left-hand corner of each element of the grid is defined to be the color of the grid element. Therefore, the values contained in \( colorind \) are highly dependent on the size of the view surface. An error is produced if the elements of the grid are smaller than one pixel.

Errors

ENOTVSAC [4] CGI not in proper state: CGI shall be in state VSAC.
EVSSIDINV [10] Specified view surface name is invalid.
EVSNACT [15] Specified view surface is not active.
ECELLATS [66] Cell array dimensions \( dx, dy \) are too small.
ECELLPOS [67] Cell array dimensions must be positive.

Inquire Pixel Array

Cerror inquire_pixel_array(p, m, n, colorind, name)
Ccoor *p; /* base of array in VDC space */
Cint m, n; /* dimensions of color array in screen space */
Cint *colorind; /* array of color values */
Cint name; /* view surface name */

inquire_pixel_array fills array \( colorind \) with the values of pixels in the area of the screen defined by point \( p \) (expressed in VDC space) and \( m \) and \( n \) (expressed in screen space). The array is filled down and to the right from point...
p. If either $m$ or $n$ are not positive, the absolute value of these arguments is used.

Multiple view surfaces and bitblt's are incompatible, so a name argument must be specified.

Errors

ENOTVSAC [4] CGI not in proper state: CGI shall be in state VSAC.
EVALOVWS [69] Value outside of view surface.
EPXNOTCR [70] Pixrect not created.

Inquire Device Bitmap

Cpixrect *inquire_device_bitmap(name)
Cint name; /* name assigned to cgi view surface */

inquire_device_bitmap returns the pixrect which corresponds to the view surface. The pixrect describes the entire device, even if the view surface is a smaller pixwin. If you want to use subareas of this pixrect or manipulate it any other way, refer to the Pixrect Reference Manual.

Errors

ENOTOPOP [5] CGI not in proper state CGI shall be in in state VDOP, VSOP, or VSAC.

Inquire BitBlt Alignments

Cerror inquire_bitblt_alignments(base, width, px, py, maxpx, maxpy, name)
Cint *base; /* bitmap base alignment */
Cint *width; /* width alignment */
Cint *px, *py; /* pattern extent alignment */
Cint *maxpx, *maxpy; /* maximum pattern size */
Cint name; /* name assigned to cgi view surface */

inquire_bitblt_alignments reports the alignment criteria which are necessary for some implementations. These factors are not critical for SunCGI. However, you should keep in mind the appropriate depth for the pixrect when talking to a specific device. Therefore the arguments base, width, px, and py are always set to zero. The arguments maxpx and maxpy are device dependent and determine the maximum size of a pattern for bitblt_pattern_array and bitblt_patterned_source_array.

Multiple view surfaces and bitblt's are incompatible, so a name argument must be specified.

Errors

ENOTVSAC [4] CGI not in proper state: CGI shall be in state VSAC.
EVSIDINV [10] Specified view surface name is invalid.
EVSNTACT [15] Specified view surface is not active.

3.3. Drawing Modes

Drawing modes determine the result of drawing any output primitive on the clear screen (background) or on top of a previously drawn object. Drawing modes only affect the drawing of bitblt primitives. However, a non-standard set_global_drawing_mode function is provided, which affects all output
primitives except bitblt's. Resetting the drawing mode in the middle of an application program only affects those output primitives drawn after the mode is reset.
The novice user is advised not to reset the drawing mode until the user has written at least one application program using SunCGI.

Set Drawing Mode

Cerror set_drawing_mode(visibility, source, destination, combination)
Cbmode visibility; /* transparent or opaque */
Cbitmaptype source; /* NOT source bits */
Cbitmaptype destination; /* NOT destination bits */
Ccombtype combination; /* combination rules */

set_drawing_mode determines the current drawing mode which in turn determines how bitblt primitives are displayed. The visibility argument determines how pixels with index zero are treated.

typedef enum {
    TRANSPARENT,
    OPAQUE
} Cbmode;

typedef enum {
    BITNOT,
    BITTRUE
} Cbitmaptype;

typedef enum {
    REPLACE,
    AND,
    OR,
    NOT,
    XOR
} Ccombtype;

If visibility is set to TRANSPARENT, all source pixels with index zero leave the destination pixel unchanged, regardless of the operation, whereas if visibility is set to OPAQUE, all pixels are treated normally. The arguments source and destination determine whether the contents of the source and destination pixrects are NOTted before the bitblt operation is performed.

The combination argument determines how the source and destination pixrects are combined. If combination is equal to REPLACE, the source pixrect (after optionally being NOT-ted) replaces the destination pixrect. If combination is equal to AND, OR, or XOR the source pixrect and the destination pixrect are combined in the indicated Boolean fashion. If combination is equal to NOT, then the destination is set to a bitwise NOT operation of the source pixrect.

Errors

ENOTOPRO [5] CGI not in proper state CGI shall be in in state VDOP, VSOP, or VSAC.
Cerror set_global_drawing_mode(combination)
Ccombype combination; /* combination rules */

set_global_drawing_mode determines the current global drawing mode which in turn determines how all output primitives except bitblt's are displayed. The combination argument determines how the source and destination pixrects are combined. If combination is equal to REPLACE (the default value) the output primitive replaces the destination background. If combination is equal to AND, OR, or XOR the output primitive and the information on the screen are combined in the indicated Boolean fashion. If combination is equal to NOT, then the destination is set to a bitwise NOT operation of the source pixrect.

Errors

ENOTOPPOP [5] CGI not in proper state CGI shall be in in state VDOP, VSOP, or VSAC.

Inquire Drawing Mode

Cerror inquire_drawing_mode(visibility, source, destination, combination)
Cbmode *visibility; /* transparent or opaque */
Cbitmaptype *source; /* NOT source bits */
Cbitmaptype *destination; /* NOT destination bits */
Ccombype *combination; /* combination rules */

The inquire_drawing_mode returns the values of the four components of the current drawing mode.

Errors

ENOTOPPOP [5] CGI not in proper state CGI shall be in in state VDOP, VSOP, or VSAC.
Attributes

Attributes ............................................................................................................................................ 53

4.1. Bundled Attribute Functions ..................................................................................... 54
   Set Aspect Source Flags ............................................................................................................. 56
   Define Bundle Index (SunCGI Extension) ............................................................................. 56
4.2. Line Attributes .................................................................................................................. 57
   Polyl ine Bundle Index ........................................................................................................... 57
   Line Type .................................................................................................................................... 58
   Line Endstyle (SunCGI Extension) ......................................................................................... 58
   Line Width Specification Mode ................................................................................................. 59
   Line Width .................................................................................................................................. 59
   Line Color ................................................................................................................................. 59
4.3. Polymarker Attributes .................................................................................................... 60
   Polymarker Bundle Index .......................................................................................................... 60
   Marker Type ................................................................................................................................ 60
   Marker Size Specification Mode ................................................................................................. 60
   Marker Size ................................................................................................................................. 61
   Marker Color ............................................................................................................................... 61
4.4. Solid Object Attributes .................................................................................................... 61
   Fill Area Bundle Index ............................................................................................................. 62
   Interior Style .............................................................................................................................. 62
4.5. Solid Interior Fill Attribute ........................................................................................... 62
   Fill Color .................................................................................................................................... 63
4.6. Hatch and Pattern Attributes ........................................................................................... 63
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hatch Index</td>
<td>64</td>
</tr>
<tr>
<td>Pattern Index</td>
<td>65</td>
</tr>
<tr>
<td>Pattern Table</td>
<td>65</td>
</tr>
<tr>
<td>Pattern Reference Point</td>
<td>65</td>
</tr>
<tr>
<td>Pattern Size</td>
<td>66</td>
</tr>
<tr>
<td>Pattern with Fill Color (SunCGI Extension)</td>
<td>66</td>
</tr>
<tr>
<td>4.7. Perimeter Attributes</td>
<td>66</td>
</tr>
<tr>
<td>Perimeter Type</td>
<td>66</td>
</tr>
<tr>
<td>Perimeter Width</td>
<td>67</td>
</tr>
<tr>
<td>Perimeter Width Specification Mode</td>
<td>67</td>
</tr>
<tr>
<td>Perimeter Color</td>
<td>68</td>
</tr>
<tr>
<td>4.8. Text Attributes</td>
<td>68</td>
</tr>
<tr>
<td>Text Bundle Index</td>
<td>68</td>
</tr>
<tr>
<td>Text Precision</td>
<td>68</td>
</tr>
<tr>
<td>Character Set Index</td>
<td>69</td>
</tr>
<tr>
<td>Text Font Index</td>
<td>69</td>
</tr>
<tr>
<td>Character Expansion Factor</td>
<td>70</td>
</tr>
<tr>
<td>Character Spacing</td>
<td>70</td>
</tr>
<tr>
<td>Character Height</td>
<td>70</td>
</tr>
<tr>
<td>Fixed Font (SunCGI Extension)</td>
<td>71</td>
</tr>
<tr>
<td>Text Color</td>
<td>71</td>
</tr>
<tr>
<td>Character Orientation</td>
<td>71</td>
</tr>
<tr>
<td>Character Path</td>
<td>72</td>
</tr>
<tr>
<td>Text Alignment</td>
<td>72</td>
</tr>
<tr>
<td>4.9. Color Attributes</td>
<td>74</td>
</tr>
<tr>
<td>Color Table</td>
<td>74</td>
</tr>
<tr>
<td>4.10. Inquiry Functions</td>
<td>75</td>
</tr>
<tr>
<td>Inquire Line Attributes</td>
<td>75</td>
</tr>
<tr>
<td>Inquire Marker Attributes</td>
<td>75</td>
</tr>
<tr>
<td>Inquire Fill Area Attributes</td>
<td>76</td>
</tr>
<tr>
<td>Inquire Pattern Attributes</td>
<td>76</td>
</tr>
<tr>
<td>Inquire Text Attributes</td>
<td>77</td>
</tr>
<tr>
<td>Inquire Aspect Source Flags</td>
<td>78</td>
</tr>
</tbody>
</table>
Attributes

The current attributes determine how output primitives are displayed. Attributes are not specific to any view surface, but affect all view surfaces. The default attributes are defined in Table 4-1. The current attributes may be set either individually or in groups (by changing the index into the bundle table). Example programs illustrating these methods of changing attributes are given in Figures 4-1 and 4-2.

Each entry in the bundle table specifies a set of attributes for a particular type of primitive (for example, solid objects). The method for setting the current attributes depends on the state of the ASF (aspect source flag) for each attribute. For individual attribute functions to have an effect, the ASF must be set to INDIVIDUAL. If the ASF is set to BUNDLED, the current attribute is defined by the entry in the bundle table pointed to by the bundle index. The actual appearance of objects also depend on the global drawing mode described in Chapter 3.

The majority of this chapter is devoted to individual attribute functions. Individual attribute functions are grouped according to the output primitives they effect: polylines, polymarkers, filled objects, and text. The color_table function (which redefines color table entries) is also included in this chapter. Finally, functions for obtaining the values of the current attributes are discussed.
Table 4-1  Default Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>All ASF's</td>
<td>INDIVIDUAL</td>
<td>All Bundle Indices</td>
<td>1</td>
</tr>
<tr>
<td>Line Color</td>
<td>1</td>
<td>Line Width</td>
<td>0.0</td>
</tr>
<tr>
<td>Line Endstyle</td>
<td>BEST_FIT</td>
<td>Line Width</td>
<td>SCALED</td>
</tr>
<tr>
<td>Line Type</td>
<td>SOLID</td>
<td>Specification Mode</td>
<td></td>
</tr>
<tr>
<td>Marker Color</td>
<td>1</td>
<td>Marker Size</td>
<td>4.0</td>
</tr>
<tr>
<td>Marker Size</td>
<td>SCALED</td>
<td>Marker Type</td>
<td>DOT</td>
</tr>
<tr>
<td>Perimeter Width</td>
<td>0.0</td>
<td>Specification Mode</td>
<td></td>
</tr>
<tr>
<td>Fill Color</td>
<td>1</td>
<td>Number of Pattern</td>
<td>2</td>
</tr>
<tr>
<td>Fill Hatch Index</td>
<td>0</td>
<td>Table Entries</td>
<td></td>
</tr>
<tr>
<td>Fill Pattern Index</td>
<td>1</td>
<td>Pattern Size</td>
<td>300,300</td>
</tr>
<tr>
<td>Interior Style</td>
<td>HOLLOW</td>
<td>Pattern Reference Point</td>
<td>0,0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pattern with Fill Color</td>
<td>OFF</td>
</tr>
<tr>
<td>Perimeter Color</td>
<td>1</td>
<td>Perimeter Width</td>
<td>SCALED</td>
</tr>
<tr>
<td>Perimeter Type</td>
<td>SOLID</td>
<td>Specification Mode</td>
<td></td>
</tr>
<tr>
<td>Perimeter Visibility</td>
<td>ON</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fontset</td>
<td>1</td>
<td>Text Font</td>
<td>STICK</td>
</tr>
<tr>
<td>Fixed Font</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Character Base.x</td>
<td>1.0</td>
<td>Character Spacing</td>
<td>0.1</td>
</tr>
<tr>
<td>Character Base.y</td>
<td>0.0</td>
<td>Character Up.x</td>
<td>0.0</td>
</tr>
<tr>
<td>Character Expansion Factor</td>
<td>1.0</td>
<td>Character Up.y</td>
<td>1.0</td>
</tr>
<tr>
<td>Character Height</td>
<td>1000</td>
<td>Text Color</td>
<td>1</td>
</tr>
<tr>
<td>Character Path</td>
<td>RIGHT</td>
<td>Text Precision</td>
<td>STRING</td>
</tr>
<tr>
<td>Horizontal Text</td>
<td>NRMAL</td>
<td>Text Continuous</td>
<td>1.0</td>
</tr>
<tr>
<td>Alignment</td>
<td></td>
<td>Alignment.y</td>
<td></td>
</tr>
<tr>
<td>Text Continuous</td>
<td>1.0</td>
<td>Vertical Text</td>
<td>NORMAL</td>
</tr>
<tr>
<td>Alignment.x</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.1. Bundled Attribute Functions

The attribute environment selector functions determine if the current attributes are defined individually or by using a set of attributes (bundles). Bundles are defined by entries in the bundle table. The CGI standard specifies the bundle table as read-only but SunCGI allows user-definition of entries in the bundle table. Each type of primitive has its own index into the bundle table, described with its specific attribute functions.

The following example program illustrates how to change the appearance with bundled attributes. The program draws a polyline with a different line style and line width.
#include <cgidefs.h>

Ccoor box[5] = {
10000,10000,
10000,20000,
20000,20000,
20000,10000,
10000,10000
};

Cbunatt bundle = {
DASHED_DOTTED, 1., 4,
X, 6., 4,
PATTERN, 1, 1, 2,
DOTTED, 1.5, 1,
STICK, CHARACTER,
1.3, 0.05, 1
};

main()
{
Ccoorlist boxlist;
Cint i, line_bundle = 2, name;
Cflaglist flags;
Cvwsurf device;

boxlist.ptlist = box;
boxlist.n = 5;
NORMAL_VWSURF(device, PIXWINDD);

open_cgi();
open_vws(&name, &device);

flags.value = (Casptype *) malloc(18*sizeof(Casptype));
flags.num = (Cint *) malloc(18*sizeof(Cint));
for (i = 0; i < 18; i++) {
    flags.value[i] = BUNDLED;
    flags.num[i] = i;
}
flags.n = 18;

define_bundle_index(2, &bundle);
set_aspect_source_flags(&flags);
polyline_bundle_index(line_bundle);
polyline(&boxlist);

sleep(10);
close_vws(name);
close_cgi();
}

Figure 4-1 Example Program with Bundled Attributes
Set Aspect Source Flags

```c
Cerror set_aspect_source_flags(flags)
Cflaglist *flags; /* list of ASFs */
```

set_aspect_source_flags determines whether individual attributes are set individually or from bundle table entries.

```c
typedef struct {
    Cint n;
    Cint num[];
    Casptype value[];
} Cflaglist;
```

The `n` element of the `flags` argument determines how many flags are to be set. The `num` array of the `flags` argument determines which flags are to be set. Flag numbers are provided in Table 4-2. Finally, the `value` array of the `flags` argument determines the values of the flags specified in `num`. If a value is assigned to INDIVIDUAL, the individual attribute functions affect the current attribute. If the value of index is BUNDLED, calls to individual attribute functions have no effect. The default `bundle index` is set to 1 (which initially contains the default value for the attributes specified in Table 4-1). The default value of all `aspect source flags` is INDIVIDUAL.

Errors

```c
ENOTOPPOP [5]  CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.
```

Table 4-2 Attribute Source Flag Numbers

<table>
<thead>
<tr>
<th>Flag</th>
<th>Attribute</th>
<th>Flag</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>line type</td>
<td>9</td>
<td>fill color</td>
</tr>
<tr>
<td>1</td>
<td>line width</td>
<td>10</td>
<td>perimeter type</td>
</tr>
<tr>
<td>2</td>
<td>line color</td>
<td>11</td>
<td>perimeter width</td>
</tr>
<tr>
<td>3</td>
<td>marker type</td>
<td>12</td>
<td>perimeter color</td>
</tr>
<tr>
<td>4</td>
<td>marker width</td>
<td>13</td>
<td>text font index</td>
</tr>
<tr>
<td>5</td>
<td>marker color</td>
<td>14</td>
<td>text precision</td>
</tr>
<tr>
<td>6</td>
<td>interior style</td>
<td>15</td>
<td>character expansion factor</td>
</tr>
<tr>
<td>7</td>
<td>hatch index</td>
<td>16</td>
<td>character spacing</td>
</tr>
<tr>
<td>8</td>
<td>pattern index</td>
<td>17</td>
<td>text color</td>
</tr>
</tbody>
</table>

Define Bundle Index (SunCGI Extension)

```c
Cerror define_bundle_index(index, entry)
Cint index; /* entry in attribute environment table */
Cbunatt *entry; /* new attribute values */
```

define_bundle_index defines an entry in the bundle table. The type `Cbunatt` is a structure which contains elements corresponding to all the attributes. If the contents of a bundle table entry are changed, all subsequently drawn primitives use the information in the new entry, depending on the relevant aspect source flags. You should keep this fact in mind if you are designing display list traversal algorithms using SunCGI.

---

12 In fact, SunCGI currently produces error 30 when these individual attribute function is called while the corresponding ASF is BUNDLED.
typedef struct {
    Clintype line_type;
    Cfloat line_width;
    Cint line_color;
    Cmartype marker_type;
    Cfloat marker_size;
    Cint marker_color;
    Cintertype interior_style;
    Cint hatch_index;
    Cint pattern_index;
    Cint fill_color;
    Clintype perimeter_type;
    Cfloat perimeter_width;
    Cint perimeter_color;
    Cint text_font;
    Cprectype text_precision;
    Cfloat character_expansion;
    Cfloat character_spacing;
    Cint text_color;
} Cbunatt;

In addition to the errors listed below, other errors can be detected if any of the attribute values are invalid, as specified in later sections. Results are undefined if an error occurs.

Errors

ENOTOPOP [5]  CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.

EBBDTBDI [31] Bundle table index out of range.

4.2. Line Attributes

SunCGI provides for specifying the style, width and color of lines which constitute polylines, circular arcs, and elliptical arcs. The functions do not affect the drawing of the perimeter of solid objects which are set by the perimeter functions.

Polyline Bundle Index

Cerror polyline_bundle_index(index)
Cint index; /* polyline bundle index */

polyline_bundle_index sets the current polyline bundle index to the value of index. The contents of the polyline bundle index are line type, line width and line color. The line width specification mode and the line endstyle attributes are not included in the polyline bundle. If index is not defined, an error is generated, and the polyline_bundle_index does not change. If the ASF's for any of these attributes is set to BUNDLED, the current values of these attributes are set to the contents of the bundle.

Errors

ENOTOPOP [5]  CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.

EBADLINX [33] Polyline index is invalid.
Line Type

```c
Cerror line_type(tttyp)
Clintype tttyp; /* style of line */
```

`line_type` defines the line type for polylines. The enumerated type `Clintype` contains values that correspond to valid line types.

```c
typedef enum {
    SOLID,
    DOTTED,
    DASHED,
    DASHED_DOTTED,
    DASH_DOT_DOTTED,
    LONG_DASHED
} Clintype;
```

The default line style is `SOLID`. The actual representation of a line on the screen is affected by the `line endstyle`. `DASH_DOT_DOTTED` actually has three dots between dashes.

Errors

```c
ENOTOPOP [5]      CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.
EBTBUNDL [30]     ASF is BUNDLED.
```

Line Endstyle (SunCGI Extension)

```c
Cerror line_endstyle(tttyp)
Cendstyle tttyp; /* style of line */
```

`line_endstyle` determines how a textured (non-SOLID) line terminates. The enumerated type `Cendstyle` contains values that correspond to valid line end styles.

```c
typedef enum {
    NATURAL,
    POINT,
    BEST_FIT
} Cendstyle;
```

If the endstyle selected is `NATURAL`, the last component of the line texture (for example, a dash or a dot) which can be completely drawn is drawn. Blank space at the end of the line may cause the line to not appear as long as specified by the starting and ending coordinates. If the endstyle selected is `POINT`, the last point of the line is drawn whether it is appropriate or not. In this case, the endpoints of the line always appear on the screen. If the endstyle selected is `BEST_FIT`, the last point is always drawn but is extended as far back as the last space if appropriate. However, the `BEST_FIT` endstyle may shorten the space between the last element of the line and the element preceding the last element by one in order to guarantee that the line ends on a drawn point. The default endstyle is `BEST_FIT`.

Errors

```c
ENOTOPOP [5]      CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.
```

Line Width Specification

Mode
Chapter 4 — Attributes

Cerror line_width_specification_mode(mode)
Cspecmode mode; /* pixels or percent */

line_width_specification_mode allows the line_width to be specified in pixels or as a percentage of VDC space according to the value of mode. The enumerated type Cspecmode contains values that correspond to line width specification modes.

typedef enum {
    ABSOLUTE,
    SCALED
} Cspecmode;

If the line_width specification mode is changed from ABSOLUTE to SCALED, the change in the line width will probably be dramatic. The default line_width specification mode is SCALED.

If multiple view surfaces are active, the line width is scaled separately for each view surface.

Errors

ENOTOPPOP [5] CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.

Line Width

Cerror line_width(index)
Cfloat index; /* line width */

line_width determines the width of the lines composing polylines, circular arcs, etc. If the line_width specification mode is SCALED, index is expressed in percent of VDC space and if the x and y dimensions are different, the width is calculated on the basis of the range of the x coordinate of VDC space. If the parameter setting would result in a line less than one pixel wide, the line width is displayed as one pixel wide. The default line_width is 0.0 (SCALED).

Errors

ENOTOPPOP [5] CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.

EBTBUNDL [30] ASF is BUNDLED.

EBDWIDTH [34] Width must be nonnegative.

Line Color

Cerror line_color(index)
Cint index; /* line color */

line_color determines the color of the lines. index selects an entry in the color lookup table. The default value of index is 1. An error is detected if index is not between 0 and 255.

Errors

ENOTOPPOP [5] CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.

EBTBUNDL [30] ASF is BUNDLED.

ECINDXLZ [35] Color index is less than zero.
4.3. Polymarker Attributes

The type, size and color of markers (the components of polymarkers) are controlled by the following functions.

Polymarker Bundle Index

Cerror polymarker_bundle_index(index)
Cint index; /* polymarker bundle index */

polymarker_bundle_index sets the current polymarker bundle index to the value of index. The contents of a polymarker bundle are marker type, marker size and marker color. The marker size specification mode function is not included in the polymarker bundle. If index is not defined, an error is generated, and the polymarker bundle index does not change. If the ASP's for any of these attributes is set to BUNDLED, the current values of these attributes are set to the values of the corresponding attribute in the bundle.

Errors

ENOTOPOP [5] CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.
EBADMRKX [37] Polymarker index is invalid.

Marker Type

Cerror marker_type(ttyp)
Cmartype ttyp; /* style of marker */

marker_type sets the marker type. The enumerated type Cmartype contains values that correspond to valid marker types.

typedef enum {
    DOT,
    PLUS,
    ASTERISK,
    CIRCLE,
    X
} Cmartype;

Note that all marker types appear as a point when the marker size is very small. The default marker type is DOT.

Errors

ENOTOPOP [5] CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.

Marker Size Specification Mode

Cerror marker_size_specification_mode(mode)
Cspecmode mode; /* pixels or percent */

marker_size_specification_mode allows the marker size to be specified in pixels or as a percentage of VDC space according to the value of mode. The enumerated type Cspecmode contains values that correspond to valid marker size specifications.
typedef enum {
    ABSOLUTE,
    SCALED
} Cspecmode;

The default marker size specification mode is SCALED.

Errors

ENOTOPOP [5] CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.

Marker Size

Cerror marker_size(index)
Cfloat index; /* marker size */

marker_size sets the size of the marker height and marker width. index is expressed in percent of VDC space. The default marker size is 4.0 percent of VDC space. If the marker size becomes very small, markers of all types are displayed as points. An error is detected if index is negative.

Errors

ENOTOPOP [5] CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.
EBADSIZE [38] Size must be nonnegative.

Marker Color

Cerror marker_color(index)
Cint index; /* marker color */

marker_color determines the color of the markers. index selects an entry in the color lookup table. An error is detected if index is not between 0 and 255. The default marker color is 1.

Errors

ENOTOPOP [5] CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.
EBTBUNDL [30] ASF is BUNDLED.
ECINDXLZ [35] Color index is less than zero.
EBADCOLX [36] Color index is invalid.

4.4. Solid Object Attributes

The solid object attribute functions describe how all solid object primitives are filled (colored-in). There are three sets of solid object attribute functions:

fill area attributes
The fill area attribute functions determine the general method for filling solid geometrical objects.

hatch and pattern attributes
determines a pixel array for filling a polygon if the fill style is set to PATTERN.

perimeter attributes
determine how the boundary of a geometrical object is displayed if the perimeter visibility is ON.
Fill Area Bundle Index

```c
Cerror fill_area_bundle_index(index)
Cint index; /* fill area bundle index */
```

`fill_area_bundle_index` sets the current `fill area bundle index` to the value of `index`. The contents of the `fill area bundle` are `interior style`, `fill color`, `hatch index`, `pattern index`, `perimeter type`, `perimeter width` and `perimeter color`. The `perimeter width specification mode` and the pattern attributes are not included in the definition of the fill area bundle. If `index` is not defined, an error is generated, and the fill area bundle index does not change. If the ASF's for any of these attributes is set to BUNDLED, the current value of the attribute is set to the value of the corresponding attribute in the bundle.

Errors

```c
ENOTOPOP [5] // CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.
```

```c
EBADFABX [39] // Fill area index is invalid.
```

Interior Style

```c
Cerror interior_style(istyle, perimvis)
Cintertype istyle; /* fill style */
Cflag perimvis; /* perimeter visibility */
```

`interior_style` sets the `fill style` for solid objects. The enumerated type `Cintertype` contains values that correspond to valid line types.

```c
typedef enum {
    HOLLOW,
    SOLIDI,
    PATTERN,
    HATCH
} Cintertype;
```

If the `fill style` is set to SOLIDI, the solid object is filled with the current `fill color`. If `istyle` is set to PATTERN or HATCH, the solid object is filled with the current PATTERN or HATCH style. The PATTERN and HATCH styles are explained in the pattern attributes section. The default `fill style` is HOLLOW.

`interior_style` also determines whether the perimeter of the solid object is visible according to the value of `perimvis` (which must be ON or OFF). If `perimvis` is OFF, the perimeter attributes have no effect. The default value of `perimeter visibility` is ON.

Be careful when using the `interior style` function to explicitly specify the `perimvis` argument. If you do not specify it, or set it to OFF, the geometrical output primitive may not be displayed because the `interior style` is HOLLOW.

Errors

```c
ENOTOPOP [5] // CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.
```

4.5. Solid Interior Fill Attribute

The following section contains the description of a function that determines the color of an interior region if the `fill style` is not HOLLOW.
Fill Color

Cerror fill_color(color)
Cint color; /* color for solid object fill */

fill_color determines the color for filling solid objects, if the fill style is not set to HOLLOW.

The default fill style is HOLLOW, so changing the fill color will not have an effect without changing the interior style first. The default fill color is 1. An error is detected if fill color is not between 0 and 255.

Errors

ENOTOPOP [5] CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.
ECINDXZLZ [35] Color index is less than zero.
EBADCOLX [36] Color index is invalid.

4.6. Hatch and Pattern Attributes

Geometrical primitives can be filled with 2D arrays of color values called patterns. SunCGI supports pre-defined as well as user-defined patterns. The definition of patterns is stored in the pattern table. Each entry in the pattern table consists of a 2D array of color values and the x and y dimensions of the array. The starting position (upper left-hand corner) of the pattern is determined by the pattern reference point.

Two types of patterns are available: PATTERNS and HATCHES. PATTERNS can be scaled and translated. HATCHES can't and simply fill the geometrical output primitives with pixel arrays.

The following example program illustrates how to change the appearance with the individual attribute functions. The program draws a polygon and fills it with a pattern.
#include <cgidefs.h>

Ccoor box[5] = {
    10000,10000,
    10000,20000,
    20000,20000,
    20000,10000,
    10000,10000
};

Cint pattern[16] = {
    50, 75, 100, 125,
    150, 0, 0, 175,
    200, 0, 0, 225,
    250, 275, 300, 325
};

main()
{
    Ccoorlist boxlist;
    Cint dx = 250, dy = 250, index = 2, name;
    Cvwsurf device;

    boxlist.n = 5;
    boxlist.ptlist = box;
    NORMAL_VWSURF(device, PIXWINDD);

    open_cgi();
    open_vws(&name, &device);

    interior_style(PATTERN, ON);
    pattern_table(index, 4, 4, pattern);
    pattern_index(index);
    pattern_size(dx, dy);
    polygon(&boxlist);

    sleep(10);

    close_vws(name);
    close_cgi();
}

Figure 4-2 Example Program with Bundled Attributes

Hatch Index

Cerror hatch_index(index)
Cint index; /* HATCH index in the pattern table */

hatch_index determines which entry in the pattern table is used to fill solid
goals when the fill style is set to HATCH. The default hatch index is 0. An error
is generated if index points to an undefined entry in the pattern table.

Errors

ENOTOP [5]          CGI not in proper state CGI shall be in state VDOP,
                     VSOP, or VSAC.
Pattern Index

Cerror pattern_index(index)
Cint index; /* PATTERN index in the pattern table */

pattern_index determines which index in the pattern table is used to fill solid objects when the fill style is set to PATTERN. The default pattern index is 1. An error is generated if index points to an undefined entry in the pattern table.

Pattern Table

Cerror pattern_table(index, m, n, colorind)
Cint index; /* entry in table */
Cint m, n; /* number of rows and columns */
Cint *colorind; /* array containing pattern */

pattern_table defines an entry in the pattern table. index defines the entry in the table (which must be less than 50). An error is generated if index is outside the bounds of the pattern table. m and n define the height and width of the pattern (in pixels). The array pointed to by the argument colorind contains the actual pattern row-wise from the upper left. For monochrome view surfaces, all nonzero entries in colorind are treated as 1 when used. The maximum number of elements in a pattern \((m \times n)\) is MAXPATSIZE.

Pattern 0 is initially defined to be a 3 \(\times\) 3 matrix which is set to zero at the corners and one elsewhere. Pattern 1 (which produces a polka-dot pattern) is initially defined to be a 3 \(\times\) 3 matrix which is set to 1 at the center and 0 elsewhere.

Errors

ENOTOPOP [5] CGl not in proper state CGI shall be in state VDOP, VSOP, or VSAC.
EBTBUNDL [30] ASF is BUNLED.
ESTYLLEZ [42] Style (pattern or hatch) index is less than zero.
ENOPATNX [43] Pattern table index not defined.

Pattern Reference Point

Cerror pattern_reference_point(begin)
Ccoor *begin;

pattern_reference_point defines the point in VDC space where the
Pattern box begins. The pattern is then replicated over all VDC space. The upper left-hand corner of the pattern box is determined by begin. The default pattern reference point is \((0, 0)\). pattern_reference_point has no effect if the interior style is not set to PATTERN.

Errors

ENOTOPPOP [5] CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.

Pattern Size

Cerror pattern_size(dx, dy)
Cint dx, dy; /* size of pattern in VDC space */

pattern_size defines the size of the pattern array in VDC coordinates. \(dx\) and \(dy\) determine the size of an element of the pattern in VDC space. pattern_size therefore allows you to ‘stretch’ the pattern to a certain size. If \(dx\) or \(dy\) would result in pattern elements less than one pixel wide, 1 is used. If the pattern size is larger than the bounds of screen space, the effective pattern size is the size of VDC space. The default pattern size is \((300, 300)\).

Errors

ENOTOPPOP [5] CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.

Pattern with Fill Color
(SunCGI Extension)

Cerror pattern_with_fill_color(flag)
Cflag flag; /* ON to use nonzero pattern elements as fill color */

Binary patterns allow the same pattern to be applied in different colors, without redefining the pattern array. pattern_with_fill_color sets a nonstandard CGI state pattern with fill color. The default pattern with fill color is OFF and each color value in a pattern table entry is used verbatim, as in standard CGI. When a pattern is used while flag is ON, the pattern is considered to be a 2D array of flags: where the pattern element is nonzero, the current fill color is used, instead of the actual value of the pattern element. (When pattern with fill color is zero, a zero color index is used, just as when the flag is OFF.)

4.7. Perimeter Attributes

The following sections contain descriptions of functions that determine the perimeter attributes perimeter type, perimeter width, perimeter width specification mode and perimeter color.

Perimeter Type

Cerror perimeter_type(ttyp)
Clintype ttyp; /* style of perimeter */

perimeter_type defines the perimeter type for solid objects. The enumerated type Clintype contains values that correspond to valid perimeter types.
typedef enum {
    SOLID,
    DOTTED,
    DASHED,
    DASHED_DOTTED,
    DASH_DOT_DOTTED,
    LONG_DASHED
} Clintype;

The default perimeter style is SOLID. Notice that there is no ending style for perimeter. The endstyle is controlled by the line_endstyle function.

As mentioned previously, control of the drawing of the borders of solid objects is under the control of the perimeter attribute functions, not the line attribute functions. However, the two sets of functions take the same values. The perimeter attributes are essentially the same as the line attributes except that they affect the borders of solid attributes. The appearance of a perimeter can be similar to a line especially if interior style is set to HOLLOW. Perimeter attribute functions have no effect if the perimeter visibility is set to OFF.

Errors

ENOTOPOP [5] CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.
EBTBUNDL [30] ASF is BUNDLED.

Perimeter Width

Cerror perimeter_width(width) Cfloat width; /* perimeter width */

perimeter_width determines the width of the perimeters of solid objects. index can be expressed in percent of VDC space or pixels. If the perimeter width specification mode is set to SCALED and the x and y dimensions are different, the perimeter width is calculated on the basis of the range of the x coordinate of VDC space. If the parameter setting would result in a perimeter less than one pixel wide, the perimeter width is displayed as one pixel wide. The default perimeter width is 0.0 (SCALED).

Errors

ENOTOPOP [5] CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.
EBTBUNDL [30] ASF is BUNDLED.
EBDWIDTH [34] Width must be nonnegative.

Perimeter Width Specification Mode

Cerror perimeter_width_specification_mode(mode) Cspecmode mode; /* pixels or percent */

perimeter_width_specification_mode allows the perimeter_width to be specified in pixels or as a percentage of VDC space according to the value of mode (which can either be ABSOLUTE or SCALED). If the perimeter_width_specification_mode is changed from ABSOLUTE to SCALED, the change in the line width will probably be dramatic. The default perimeter width specification mode is SCALED.
Errors

ENOTOPPOP [5] CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.

Perimeter Color

Cerror perimeter_color(index)
Cint index; /* perimeter color */

perimeter_color determines the color of the perimeters. index selects an entry in the color lookup table. The default value of index is 1. An error is detected if index is not between 0 and 255.

Errors

ENOTOPPOP [5] CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.
EBTBUNDL [30] ASF is BUNDLED.
ECINDXLZ [35] Color index is less than zero.
EBADCOLX [36] Color index is invalid.

4.8. Text Attributes

SunCGI provides a variety of functions for determining how text is written to the screen. The most important text attribute is text precision. If text precision is set to STRING, firmware characters are used. The fonts, size, spacing, and alignment of firmware are more limited than characters drawn with text precision set to a value other than STRING. Therefore, calls to text attribute functions regulating these aspects of text drawing have no effect when text precision is set to STRING.

Text Bundle Index

Cerror text_bundle_index(index)
Cint index; /* text bundle index */

text_bundle_index sets the current text bundle index to the value of index. The contents of the text bundle index are text font text precision, character expansion factor, character spacing, and text color. The character height character orientation character path text alignment and fixed font are not included in the definition of the text bundle. If index is not defined, an error is generated, and the text bundle index does not change. If the ASF’s for any of these attributes are set to BUNDLED, the current values of these attributes are set to the contents of the bundle.

Errors

ENOTOPPOP [5] CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.
EBADTXTX [45] Text index is invalid.

Text Precision

Cerror text_precision(ttyp)
Cprectype ttyp; /* text type */

text_precision controls the precision with which text is displayed. The enumerated type Cprectype contains values that correspond to valid text precisions.
typedef enum {
    STRING,
    CHARACTER,
    STROKE
} Cprectype;

If the text precision is set to STRING, the firmware character set is used. Note: firmware characters cannot be scaled or rotated.

Characters are clipped, but not in parts (that is, if any portion of the character exceeds the clipping boundary the whole character is clipped). If the text precision is set to CHARACTER, software generated characters are employed and characters are clipped, but not in parts. All text attributes have a visible effect on software generated characters. If the text precision is set to STROKE, the CHARACTER precision capabilities are enabled and characters are clipped in parts. The default text precision is STRING.

**Character Set Index**

Cerror character_set_index(index)
Cint index; /* font set */

character_set_index selects a set of fonts. Although SunCGI supports this function, only set number 1 is defined. Calls to character_set_index with index assigned to a value other than 1 are ignored.

**Text Font Index**

text_font_index determines the current font. A list of available fonts and their availability when text precision is set to STRING is given in Table 4-3. A warning about the SYMBOL font: undefined characters are displayed as bugs (the six-legged kind). The default font is STICK.
Table 4-3  Available Fonts

<table>
<thead>
<tr>
<th>Font</th>
<th>String Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROMAN</td>
<td>Yes</td>
</tr>
<tr>
<td>GREEK</td>
<td>Yes†</td>
</tr>
<tr>
<td>SCRIPT</td>
<td>Yes</td>
</tr>
<tr>
<td>OLDENGLISH</td>
<td>No</td>
</tr>
<tr>
<td>STICK</td>
<td>Yes</td>
</tr>
<tr>
<td>SYMBOLS</td>
<td>No</td>
</tr>
</tbody>
</table>

† displayed as STICK font.

Character Expansion Factor

```c
Cerror character_expansion_factor(efac)
Cfloat efac; /* width factor */
```

determines the width-to-height ratio of characters. If efac is greater than 1 the characters appear fatter than they are wide. If efac is less than 1 the characters appear slimmer than they are wide. The default character expansion factor is 1.0. An error is generated if efac is less than 0.01 or greater than 10.

Errors

- ENOTOPOP [5]: CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.
- EBBTUNDL [30]: ASF is BUNDLED.
- ECXFOOR [48]: Expansion factor is out of range.

Character Spacing

```c
Cerror character_spacing(spcratio)
Cfloat spcratio; /* spacing ratio */
```

sets the spacing between characters based on the height of the characters. The amount of space between characters is obtained by multiplying the character height by spcratio. The default character spacing factor is 0.1. An error is generated if spcratio is less than -10 or greater than 10.

Errors

- ENOTOPOP [5]: CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.
- EBBTUNDL [30]: ASF is BUNDLED.
- ECXFOOR [48]: Expansion factor is out of range.

Character Height

```c
Cerror character_height(height)
Cint height; /* height in VDC */
```

The character_height function determines the height of text in VDC units. The height is defined as the distance from the top to the bottom of the character.

Notice that changing the character height implicitly changes the character spacing.
The default character height is 1000. This may result in huge characters if VDC space is reset from its default range (0-32767). If the x and y dimensions of VDC space are different, the height is calculated on the basis of the range of the x coordinate of VDC space.

**Errors**

- ENOTOPPop [5]  CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.
- EBTBUNdL [30]  ASF is BUNDLED.
- ECHHTLEZ [49]  Character height is less than or equal to zero.

**Fixed Font (SunCGI Extension)**

The function `fixed_font(flag)` allows characters to be of fixed or variable size. If `flag` is nonzero, the characters are of uniform size, otherwise the characters are packed proportional to their actual sizes. If the `character precision` is STRING, this function has no effect. By default SunCGI supports variable width characters.

**Errors**

- ENOTOPPop [5]  CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.

**Text Color**

The function `text_color(index)` determines the color of the text. `index` selects an entry in the color lookup table. The default value of `index` is 1. An error is detected if `index` is not between 0 and 255.

**Errors**

- ENOTOPPop [5]  CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.
- EBTBUNdL [30]  ASF is BUNDLED.
- ECINDXLZ [35]  Color index is less than zero.
- EBADCOLX [36]  Color index is invalid.

**Character Orientation**

The function `character_orientation(xbase, ybase, xup, yup)` specifies the skew and direction of text. The left side of the character box lies on an invisible line called the `character up vector` whose slope is determined by `xup` and `yup`. The bottom of the character box lies on an invisible line called the `character base vector` whose slope is determined by `xbase` and `ybase`.

If the `character up vector` and the `character base vector` are not orthogonal, the text is distorted. Calls to `character_orientation` have no effect if `text precision` is set to STRING. The default values for the `character up vector` and the `character base vector` are `xbase = 1.0`, `ybase = 0.0`, `xup = 0.0`, and `yup = 1.0`. 
The character up vector and the character base vector influence the character path and the character alignment. For example, if $x_{base} = -1.0$ and the character path is RIGHT, the text is written to the left.

**Errors**

ENOTOP[5] CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.

ECHRUPVZ[50] Length of character up vector or character base vector is zero.

**Character Path**

```c
typedef enum {
    RIGHT,
    LEFT,
    UP,
    DOWN
} Cpathtype;
```

The actual effect of character path depends on the character up vector and the character base vector. RIGHT specifies that the text is written in the direction of the character base vector. For example, if the direction of the character base vector points left instead of right ($x_{up} = -1.0$ instead of 1.0), the text will be written right-to-left instead of left-to-right which is the usual interpretation of RIGHT. LEFT specifies that the text is written in the opposite direction of the character base vector. The character up vector and character base vector essentially change functions when the character direction is set to UP or DOWN. UP specifies that the text is written in the direction of the character up vector. DOWN specifies that the text is written in the opposite direction of the character up vector. The default character path is RIGHT.

**Text Alignment**

```c
typedef enum {
    LEFT,
    RIGHT,
    CENTER
} Chaligntype;
```

The enumerated type Chaligntype contains values that correspond to valid horizontal alignments.

```c
typedef enum {
    TOP,
    CENTER,
    BOTTOM
} Cvaligntype;
```

The enumerated type Cvaligntype contains values that correspond to valid vertical alignments.

```c
Cfloat hcalind, vcalind;
```

The text_alignment determines where the text is positioned relative to the starting point specified by the $cl$ argument of the text or vdm_text function. halign determines where the character is placed in relation to the x component of the starting coordinate of the text position (specified by the $cl$ argument of text). The default character alignment is CENTER.
typedef enum {
    LFT,
    CNTER,
    RGHT,
    NRMAL,
    CNT
} Chaligntype;

If the value of halign is LFT, the horizontal position of the text will begin at the left edge of the box enclosing the text. Similarly, if the value of halign is RGHT, the horizontal position of the text will begin at the right edge of the box enclosing the text. If the value of halign is CNTER the horizontal position of the text will begin equidistant from the right and the left edges of the text box. NRMAL assigns the alignment based on the value of the character path (see Table 4-4). If the value of halign is CNT (continuous) the horizontal position of the text is determined by the argument hcalind. In this case, the text will begin hcalind fraction of the width of the text box from the left edge of the character box. The default value of halign is NRMAL.

valign specifies where the character is placed in relation to the y component of the text position. The enumerated type Cvaligntype contains values that correspond to valid vertical alignments.

typedef enum {
    TOP,
    CAP,
    HALF,
    BASE,
    BOTTOM,
    NORMAL,
    CONT
} Cvaligntype;

If the value of valign is TOP, the vertical position of the text will begin at the top edge of the character box. If the value of valign is CAP, the vertical position of the text will begin at the cap line of the character. Similarly, if the value of valign is BOTTOM, the vertical position of the text will begin at the bottom edge of the character box. If the value of valign is BASE, the vertical position of the text will begin at the baseline of the character. If the value of valign is HALF the vertical position of the text will begin equidistant from the top and the bottom edges of the character box. NORMAL assigns the alignment based on the value of the character path (see Table 4-4). If the value of valign is assigned to CONT (continuous), the vertical position of the text is determined by the argument vcalind and will begin vcalind fraction of the height of the character box from the bottom edge of the character box. The default value of valign is NORMAL.

13 The cap line is defined as the invisible line corresponding to the top of the average character within a font.
14 The baseline is defined as the invisible line corresponding to the bottom of the average character within a font. The baseline does not necessarily correspond to the bottom of a character. For example, a the tail of a lower-case g extends below the baseline.
Table 4-4  Normal Alignment Values

<table>
<thead>
<tr>
<th>Character Path</th>
<th>Horizontal Normal</th>
<th>Vertical Normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>RIGHT</td>
<td>LEFT</td>
<td>BASELINE</td>
</tr>
<tr>
<td>LEFT</td>
<td>RIGHT</td>
<td>BASELINE</td>
</tr>
<tr>
<td>UP</td>
<td>CENTER</td>
<td>BASELINE</td>
</tr>
<tr>
<td>DOWN</td>
<td>CENTER</td>
<td>TOP</td>
</tr>
</tbody>
</table>

Errors

ENOTOP [5]  CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.

4.9. Color Attributes

SunCGI supports only one color specification mode — INDEXED. This color specification mode means that the red, green, and blue values (hereafter referred to as RGB values) are obtained from a table known as the color lookup table. The initial values of the color lookup table are provided in Table 4-5. If the device is monochrome, nonzero color values are displayed as black; zero is displayed as white.

Table 4-5  Default Color Lookup Table

<table>
<thead>
<tr>
<th>Index</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>black</td>
</tr>
<tr>
<td>1</td>
<td>red</td>
</tr>
<tr>
<td>2</td>
<td>yellow</td>
</tr>
<tr>
<td>3</td>
<td>green</td>
</tr>
<tr>
<td>4</td>
<td>cyan</td>
</tr>
<tr>
<td>5</td>
<td>blue</td>
</tr>
<tr>
<td>6</td>
<td>magenta</td>
</tr>
<tr>
<td>7</td>
<td>white</td>
</tr>
</tbody>
</table>

Color Table

Cerror color_table(istart, clist)
Cint istart; /* starting address */
Centry *clist; /* color triples and number of entries */

color_table defines RGB entries into the color lookup table. The color lookup table is initialized based on the depth of the display frame buffer and the cmapsize field provided in the Cvwsurf structure provided to open_vws. A monochrome device has an unwritable color map; non-zero color indices are displayed as black, zero is displayed as white. A color device gets a color map segment with 8 entries if the cmapsize field is zero upon opening the view surface. The 8 default color values are given in Table 4-5. Larger color maps are also initialized to evenly spaced RGB values.

The structure Centry contains elements that describe a color map entry.
The minimum and maximum color table entries are treated specially by Pixwins and hence by SunCGI. If they are set to be the same value, the user's values for these two entries are both ignored. They revert to the inverse of the normal values; entry 0 becomes white, the maximum entry becomes black.

The argument istart determines the first entry in the color lookup table to be modified. The argument clist contains the color information for entry istart in terms of triples of values of numbers ranging between 0 and 255. The last field of clist reports how many entries are to be modified. An error is generated if either the indices to the color lookup table are out of range.

4.10. Inquiry Functions

The attribute inquiry functions permit examination of the current attributes. Attributes are reported in groups corresponding to the class of output primitive which they modify. The argument to each inquiry function has its own structure type which has an element for each of the individual attributes (see Appendix D).

Inquire Line Attributes

`Cmimage *inquire_line_attributes()

/* returns a pointer to line attribute structure */`

`inquire_line_attributes` reports the current line style, line width, line color, and polyline bundle index in the appropriate elements of the returned value of the function.

typedef struct {
  Cntype `style`;
  Cfloat `width`;
  Cint `color`;
  Cint `index`;
} Cimage;

`inquire_line_attributes` returns a NULL (not an error number) in case of errors. Errors are printed if the error warning mode is not set to NO_ACTION.
inquire_marker_attributes reports the current marker style, marker width, marker color, and polymarker bundle index in the appropriate elements of the returned value of the function.

typedef struct {
    Cmartye type;
    Cfloat size;
    Cint color;
    Cint index;
} Cmarkatt;

inquire_marker_attributes returns a NULL (not an error number) in case of errors. Errors are printed if the error warning mode is not set to NO_ACTION.

Errors

ENOTOPOP [5] CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.

Inquire Fill Area Attributes

Cfillatt *inquire_fill_area_attributes()

The current interior style, perimeter visibility, fill color, hatch index, pattern index, fill area bundle index, perimeter style, perimeter width, and perimeter color can be obtained by using the inquire_fill_attributes function.

typedef struct {
    Cintertype style;
    Cflagtype visible;
    Cint color;
    Cint hatch_index;
    Cint pattern_index;
    Cint index;
    Cintype pstyle;
    Cfloat pwidth;
    Cint pcolor;
} fillatt;

inquire_fill_area_attributes returns a NULL (not an error number) in case of errors. Errors are printed if the error warning mode is not set to NO_ACTION.

Errors

ENOTOPOP [5] CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.

Inquire Pattern Attributes

Cpatternatt *inquire_pattern_attributes()

/* returns a pointer to pattern attribute structure */

inquire_pattern_attributes reports the current pattern index, row count, column count, color list, pattern reference point, and pattern size.
typedef struct {
    Cint cur_index;
    Cint row;
    Cint column;
    Cint *colorlist;
    Ccoor *point;
    Cint dx;
    Cint dy;
} patternatt;

inquire_pattern_attributes returns a NULL (not an error number) in case of errors. Errors are printed if the error warning mode is not set to NO_ACTION.

Inquire Text Attributes

Ctextatt *inquire_text_attributes()
    /* returns a pointer to text attribute structure */

inquire_text_attributes reports the current font set, text bundle index, font, text precision, character expansion factor, character spacing, text color, character height, character base vector, character up vector, character path, and text alignment.

typedef struct {
    Cint fontset;
    Cint index;
    Cint current_font;
    Cprectype precision;
    Cfloat exp_factor;
    Cfloat space;
    Cint color;
    Cint height;
    Cfloat basex;
    Cfloat basey;
    Cfloat upx;
    Cfloat upy;
    Cpathtype path;
    Chaligntype halign;
    Cvaligntype valign;
    Cfloat hcalind;
    Cfloat vcalind;
} textatt;

inquire_text_attributes returns a NULL (not an error number) in case of errors. Errors are printed if the error warning mode is not set to NO_ACTION.

Errors

ENOTOPPOP[5] CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.
Inquire Aspect Source Flags

Cflaglist *inquire_aspect_source_flags()
    /* returns a pointer to text attribute structure */

inquire_aspect_source_flags reports whether attributes are set individually by returning all of the values of the ASFs. The element n of the flaglist struct is set to 18. The definitions of each flag are in Table 4-2.

typedef struct {
    Cint n;
    Cint *num;
    Casptype *value;
} Cflaglist;

inquire_aspect_source_flags returns a NULL (not an error number) in case of errors. Errors are printed if the error warning mode is not set to NO_ACTION.

Errors

ENOTOPPOP [5]  CGI not in proper state CGI shall be in state VDOP, VSOP, or VSAC.
Input

5.1. Input Device Initialization ................................................................. 84
  Initialize LID ................................................................................. 84
  Release Input Device ........................................................................ 85
  Associate ......................................................................................... 85
  Set Default Trigger Associations ..................................................... 86
  Dissociate ......................................................................................... 86
  Set Initial Value .............................................................................. 87
  Set VALUATOR Range .................................................................... 87
  Track On ........................................................................................... 88
  Track Off .......................................................................................... 89

5.2. Synchronous Input .......................................................................... 90
  Request Input ................................................................................... 91

5.3. Asynchronous Input ......................................................................... 92
  Initiate Request .............................................................................. 92

5.4. Event Queue Input .......................................................................... 93
  Enable Events .................................................................................. 95
  Await Event .................................................................................... 95
  Flush Event Queue .......................................................................... 96
  Selective Flush of Event Queue ....................................................... 96

5.5. Miscellaneous Input Functions ....................................................... 97
  Sample Input .................................................................................... 97
  Get Last Requested Input ............................................................... 97
Disable Events ................................................................................................................ 98
5.6. Status Inquiries ........................................................................................................ 98
  Inquire LID State List ................................................................................................. 98
  Inquire LID State ........................................................................................................ 99
  Inquire Trigger State ................................................................................................. 99
  Inquire Event Queue State ....................................................................................... 99
CGI has a collection of functions for managing input devices. The design of these functions has two purposes: provide an interface close to the actual input device and maintain portability of applications. CGI accomplishes the first goal with different input device classes and methods of extracting input values. The second goal is achieved through CGI's model of logical input devices (LID), an abstraction whereby logical input devices required by the CGI standard are mapped onto the physical devices available to a CGI implementation. This section will introduce some of the terms used in describing the functionality of the CGI input primitives.

A CGI input device consists of a measure associated with a trigger. A measure is the current value of a logical input device. For example, the IC_LOCATOR device reports an x-y position. This device is useful for determining a position on the screen. A trigger is a physical device used by an operator to accept a current value. A trigger fire corresponds to an event on a physical input device. At the request of the application program, SunCGI associates a measure with a trigger. Table 5-1 has a list of the five logical input devices available to SunCGI application programs and the available triggers. For example, a mouse button on a Sun workstation is a trigger that can be associated with an IC_LOCATOR device. When the mouse button is pressed, the x-y position of the mouse is returned as the measure of the IC_LOCATOR input device.

An input event is the information saved when a trigger fires. This includes the measure of a logical input device associated with a trigger.
Table 5-1 \textit{Input Devices Offered by SunCGI}

<table>
<thead>
<tr>
<th>Device Class</th>
<th>Measure</th>
<th>Trigger Number</th>
<th>Trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC_LOCATOR</td>
<td>$x$-$y$ position in VDC space.</td>
<td>2</td>
<td>Left mouse button</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>Middle mouse button</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>Right mouse button</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>Mouse movement†</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>Mouse still‡</td>
</tr>
<tr>
<td>IC_STROKE</td>
<td>Array of $x$-$y$ points in VDC space.</td>
<td>2</td>
<td>Left mouse button</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>Middle mouse button</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>Right mouse button</td>
</tr>
<tr>
<td>IC_VALUATOR</td>
<td>Normalized $x$ position.</td>
<td>2</td>
<td>Left mouse button</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>Middle mouse button</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>Right mouse button</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>Mouse movement</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>Mouse still</td>
</tr>
<tr>
<td>IC_CHOICE</td>
<td>A non-negative integer which represents a selection from a number of choices. Zero represents &quot;no choice&quot;.</td>
<td>2</td>
<td>Left mouse button</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>Middle mouse button</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>Right mouse button</td>
</tr>
<tr>
<td>IC_STRING</td>
<td>Character string.</td>
<td>1</td>
<td>Keyboard input terminated a carriage return.</td>
</tr>
</tbody>
</table>

† The \textit{Mouse Movement} trigger fires when the mouse moves.

‡ The \textit{Mouse Still} trigger fires when the mouse does not move for one fifth of a second or more.

The graphical method with which the measure of an input device is displayed is called \textit{tracking}. SunCGI provides several methods of tracking for each input device. Table 5-3 has a list of track types available for each input device class. Tracking must be explicitly enabled for each device.

Each input device can be in one of the five states described pictorially in Figure 5-1. The state of an input device determines the manner in which the application program retrieves the measure of the input device. The input functions that allow a change of state are listed next to the arrows indicating the state change.

\textbf{RELEASED}

Before an input device is initialized it is in the RELEASED state. Any input function (except initialization) will generate an error in this state.

\textbf{NO_EVENTS}

After an input device has been initialized it is in the NO_EVENTS state. An application program can extract an input value of an input device in NO_EVENTS state. This will result in either the value that the device was
initialized with or the value the device had when it was in a state where it could process events. This is not necessarily the current measure of the device and does not change while the device is in this state.

RESPOND_EVENT

The RESPOND_EVENT state corresponds with synchronous communication between the process that controls the input device and the application program. When an application program requests the measure of an input device in RESPOND_EVENT state, SunCGI blocks program execution until it can fulfill the request. The request_input function will return when the trigger fires and the input request is satisfied or after a timeout period. The input device then reverts to NO_EVENTS state.

The function that requests input and puts the input device in RESPOND_EVENT state is request_input. When the trigger associated with an input device in RESPOND_EVENT state fires, the measure of that input device is then stored in the request register as well as returned by the request_input function.

REQUEST_EVENT

The REQUEST_EVENT state corresponds with asynchronous communication between the process that controls the input device and the application program. When an application samples an input device, input handling and program execution continue in parallel. Either the requested trigger fires or an explicit request is made to disable event processing and return the device to NO_EVENTS state.

When the trigger associated with an input device in REQUEST_EVENT state fires, the measure of that input device is then stored in the request register, a buffer with one element per device. The request register can be then be read with get_last_requested_event.

QUEUE_EVENT

When a device is in QUEUE_EVENT mode, events associated with the indicated device are appended to the event queue, a first-in, first-out (FIFO) buffer shared by all input devices. After calling enable_events, the SunCGI application retains program control. While an input device is in QUEUE_EVENT mode, events are simultaneously added to the event queue when the program executes.

await_event returns the event at the head of the event queue. If the queue is empty, await_event will wait for the designated trigger to fire or a timeout. The application program must process this queue in a timely fashion or it will overflow. The event queue can be flushed completely or for a specific device. The application program must make an explicit request to disable event queue processing and return an input device to NO_EVENTS state.
5.1. Input Device Initialization

Before input can be processed, an input device must be initialized and associated with a trigger. Input device initialization requires at least one active view surface. Typically, the procedure for initializing an input device includes calls to the initialize_lid and associate functions which turn on an input device and associate it with a specific trigger.

Initialize LID

Cerror initialize_lid(devclass, devnum, ival)
Cdevoff devclass; /* device type */
Cint devnum; /* device number */
Cinrep *ival; /* initial value of device measure */

initialize_lid initializes an input device and changes its state from RELEASED to NO_EVENTS. This function must be called for an input device before it can be referenced by any other input function. The argument devclass specifies the desired type of input value. devnum indicates the number of the device within that class. The argument ival sets the initial measure of the device.

The Cinrep structure contains different elements for each type of measure. The appropriate element of Cinrep must be set or an error will be generated.
typedef struct {
    Ccoor *xypt; /* LOCATOR */
    Ccoorlist *points; /* STROKE devices */
    Cfloat val; /* VALUATOR device */
    Cint choice; /* CHOICE devices */
    Cchar *string; /* STRING device */
    Cpick *pick; /* PICK devices (unsupported) */
} Cinrep;

For example, in a LOCATOR device initialization, the xypt field of Cinrep must be set to the address of a Ccoor allocated by the application program before the x and y elements can be set. See the example program in Figure 5-2.

Notice that whenever a device is initialized, no associations with triggers are made. This must be done by having the application program call the appropriate functions. An error is generated by initialize_lid if the device does not exist, if it is already initialized, or if the initial value is out of range.

### Errors

- **ENOTVSAC [4]**: CGI not in proper state: CGI shall be in state VSAC.
- **EINDNOEX [80]**: Input device does not exist.
- **EINDALIN [82]**: Input device already initialized.\(^\text{15}\)
- **EBADDATA [95]**: Contents of input data record are invalid.
- **ESTRSIZE [96]**: Length of initial string is greater than the implementation defined maximum.

### Release Input Device

```c
Cerror release_input_device(devclass, devnum)
Cdevoff devclass; /* device type */
Cint devnum; /* device number */
```

release_input_device releases all associations between a device and its triggers, and removes all pending events for the device from the event queue. release_input_device changes the state of the specified input device from NO_EVENTS to RELEASED. An error is produced if devclass and devnum does not refer to an existing and initialized device.

### Errors

- **ENOTVSAC [4]**: CGI not in proper state: CGI shall be in state VSAC.
- **EINDNOEX [80]**: Input device does not exist.
- **EINDINIT [81]**: Input device not initialized.

### Associate

```c
Cerror associate(trigger, devclass, devnum)
Cint trigger; /* trigger number */
Cdevoff devclass; /* device type */
Cint devnum; /* device number */
```

\(^{15}\) The ANSI standard allows initialized input devices to be re-initialized. SunCGI does not because it is felt that re-initialization is usually a mistake.
associate links a trigger with a specific device. The trigger numbers available for each device are listed in Table 5-1. Multiple associations are allowed; however, some associations are not allowed (for example, IC_LOCATOR may not be associated with the keyboard).

The interaction between an IC_STROKE device and the trigger requires some additional explanation. IC_STROKE can only be associated with the mouse buttons. The first coordinate in the IC_STROKE array is entered when the mouse button is initially pressed, the last coordinate is entered when the mouse button is released. For IC_LOCATOR and IC_VALUATOR devices, the measure is reported when the mouse button is pressed.

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENOTVSAC [4]</td>
<td>CGI not in proper state: CGI shall be in state VSAC.</td>
</tr>
<tr>
<td>EINDNOEX [80]</td>
<td>Input device does not exist.</td>
</tr>
<tr>
<td>EINDINIT [81]</td>
<td>Input device not initialized.</td>
</tr>
<tr>
<td>EINASAEX [83]</td>
<td>Association already exists.</td>
</tr>
<tr>
<td>EINAIIMP [84]</td>
<td>Association is impossible.</td>
</tr>
<tr>
<td>EINTRNEX [86]</td>
<td>Trigger does not exist.</td>
</tr>
</tbody>
</table>

Set Default Trigger Associations

```
cerror set_default_trigger_associations(devclass, devnum)
Cdevoff devclasSi /* device type */
Cint devnumi /* device number */
```

set_default_trigger_associations associates a device with a default trigger. The default associations are listed in Table 5-2. The rules for trigger association are the same as those for the associate function.

Table 5-2 Default Trigger Associations

<table>
<thead>
<tr>
<th>Device Class</th>
<th>Trigger Number</th>
<th>Trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC_LOCATOR</td>
<td>5</td>
<td>Mouse position</td>
</tr>
<tr>
<td>IC_STROKE</td>
<td>4</td>
<td>Right mouse button</td>
</tr>
<tr>
<td>IC_VALUATOR</td>
<td>3</td>
<td>Middle mouse button</td>
</tr>
<tr>
<td>IC_CHOICE</td>
<td>2</td>
<td>Left mouse button</td>
</tr>
<tr>
<td>IC_STRING</td>
<td>1</td>
<td>Keyboard</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENOTVSAC [4]</td>
<td>CGI not in proper state: CGI shall be in state VSAC.</td>
</tr>
<tr>
<td>EINDNOEX [80]</td>
<td>Input device does not exist.</td>
</tr>
<tr>
<td>EINDINIT [81]</td>
<td>Input device not initialized.</td>
</tr>
<tr>
<td>EINASAEX [83]</td>
<td>Association already exists.</td>
</tr>
<tr>
<td>EINTRNEX [86]</td>
<td>Trigger does not exist.</td>
</tr>
</tbody>
</table>

Dissociate
dissociate removes the association between a trigger and a specified device. If dissociate is called while there are events pending in the event queue for the dissociated device, the pending events are discarded.

### Errors

- **ENOTVSAC [4]**: CGI not in proper state: CGI shall be in state VSAC.
- **EINDNOEX [80]**: Input device does not exist.
- **EINDINIT [81]**: Input device not initialized.
- **EINNTASD [85]**: association does not exist.
- **EINTRNEX [86]**: Trigger does not exist.

### Set Initial Value

`set_initial_value(devclass, devnum, value)`

`devclass` /* device type */

`devnum` /* device number */

`value` /* device value */

set_initial_value sets the current measure of a specified device. This function resets the position of the track, if the track is appropriate and activated. set_initial_value also resets the request register.

A pointer element of the `Cinrep` structure must be set to the address of an application program allocated area before the values can be set. For example, in Figure 5-2 the following statements were necessary before an initial value could be assigned to the LOCATOR device.

```c
Cinrep ivalue;
point.x = 16384;
point.y = 16384;
ivalue.xypt = &point;
```

### Errors

- **ENOTVSAC [4]**: CGI not in proper state: CGI shall be in state VSAC.
- **EINDNOEX [80]**: Input device does not exist.
- **EINDINIT [81]**: Input device not initialized.
- **EBADDATA [95]**: Contents of input data record are invalid.
- **ESTRSIZE [96]**: Length of initial string is greater than the implementation defined maximum.

### Set VALUATOR Range

`set_valuator_range(devnum, vmin, vmax)`

`devnum` /* device number */

`vmin`, `vmax` /* limits of VALUATOR */

set_valuator_range specifies the limits of the IC_VALUATOR. Device coordinates are mapped into the IC_VALUATOR range. IC_VALUATOR events
which are already on the event queue are not rescaled. These events must be dequeued with either the \texttt{selective\_flush\_of\_event\_queue} function or \texttt{flush\_event\_queue}.

\textbf{Errors}

- \texttt{ENOTVSAC} [4] \quad CGI not in proper state: CGI shall be in state VSAC.
- \texttt{EINDNOEX} [80] \quad Input device does not exist.
- \texttt{EINDINIT} [81] \quad Input device not initialized.

\textbf{Track On}

\begin{verbatim}
Cerror track_on(devclass, devnum, tracktype, trackregion, value)
Cdevoff devclass; /* device type */
Cint devnum; /* device number */
Cint tracktype; /* track number */
Ccoorpair *trackregion; /* window for tracking */
Cinrep *value; /* device value */
\end{verbatim}

Tracking functions determine how the measure of an input device is displayed on the view surface. Each class of devices has its own set of possible tracks (given in Table 5-3). Although SunCGI allows certain classes of devices to track simultaneously, all types of input devices are not allowed to track at once. Tracking is not provided in the \texttt{NO\_EVENTS} state unless the track type is \texttt{PRINTERS\_FIST}.

\texttt{track\_on} initiates track (or echo) for a specific device. The \texttt{tracktype} argument specifies the type of track to be used. The \texttt{trackregion} argument is not used; the device tracks in all areas of the view surface. The argument \texttt{value} is used to initialize tracking. The track is initially displayed on the first view surface opened.

The \texttt{xypt} element of the \texttt{Cinrep} structure must be set to the address of an application allocated \texttt{Ccoor} and the \texttt{Ccoor's x} and \texttt{y} fields are set to position the cursor. The reference point for \texttt{IC\_STROKE} echos 2 through 5 is the first point in the \texttt{STROKE} array. The reference point for \texttt{STRING\_TRACK} echo is the \texttt{append\_text} concatenation point, and can be changed by calling \texttt{text} or \texttt{append\_text}.

\textbf{Errors}

- \texttt{ENOTVSAC} [4] \quad CGI not in proper state: CGI shall be in state VSAC.
- \texttt{ENECHON} [88] \quad Track already on.
- \texttt{EINETNSU} [91] \quad Track type not supported.
- \texttt{EBADDATA} [95] \quad Contents of input data record are invalid.
- \texttt{ESTRSIZE} [96] \quad Length of initial string is greater than the implementation defined maximum.

\hspace{1cm}

\textcopyright\ microsystems

Version B of 17 February 1986
Table 5-3  Available Track Types

<table>
<thead>
<tr>
<th>Device Class</th>
<th>Number</th>
<th>Track Type†</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC_LOCATOR</td>
<td>≤0</td>
<td>NO_ECHO</td>
<td>Default cursor.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>PRINTER'S_FIST</td>
<td>Designate the current position of the IC_LOCATOR device with a printer's fist cursor.</td>
</tr>
<tr>
<td>IC_STROKE</td>
<td>≤0</td>
<td>NO_ECHO</td>
<td>Default cursor.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>PRINTER'S_FIST</td>
<td>Designate the current position of the IC_STROKE device with a printer's fist cursor.</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>SOLID_LINE</td>
<td>Draw a line from the origin to the current position in the STROKE array.</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>X_LINE</td>
<td>Draw a line from the x-axis to the current position in the STROKE array.</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Y_LINE</td>
<td>Draw a line from the y-axis to the current position in the STROKE array.</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>RUBBER_BAND_BOX</td>
<td>Designate the current position of the IC_STROKE device with a rubber band line connecting the initial position and the current position in the STROKE array.</td>
</tr>
<tr>
<td>IC_VALUATOR</td>
<td>≤0</td>
<td>NO_ECHO</td>
<td>Default cursor.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>PRINTER'S_FIST</td>
<td>Indicate the state of the IC_VALUATOR device with a printer's fist cursor.</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>STRING_TRACK</td>
<td>Display a digital representation of the current IC_VALUATOR value.</td>
</tr>
<tr>
<td>IC_CHOICE</td>
<td>≤0</td>
<td>NO_ECHO</td>
<td>Default cursor.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>PRINTER'S_FIST</td>
<td>Indicate the state of the IC_CHOICE device with a printer's fist cursor.</td>
</tr>
<tr>
<td>IC_STRING</td>
<td>≤0</td>
<td>NO_ECHO</td>
<td>Default cursor.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>PRINTER'S_FIST</td>
<td>Indicate the state of the IC_STRING device with a printer's fist cursor.</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>STRING_TRACK</td>
<td>Display the current STRING value.</td>
</tr>
</tbody>
</table>

† The values listed in the Track Type column in Table 5-3 are contained in the enumerated type Cechotype returned in the Cstatelist structure by inquire_lid_state_list. They are not used by track_on to define a track type.

Track Off

```c
Cerror track_off(devclass, devnum, tracktype, action)
Cdevoff devclass; /* device type */
Cint devnum; /* device number */
Cint tracktype;
Cfreeze action;
```

track_off terminates tracking for a specified input device. The tracktype and the action arguments are always ignored.

Errors

- **ENOTVSAC** [4]  CGI not in proper state: CGI shall be in state VSAC.
- **EINDNOEX** [80]  Input device does not exist.
5.2. Synchronous Input

The synchronous input function `request_input` allows the application program to obtain the current measure of an input device. This function requires explicit identification of an input device (through the `associate` function).

Figure 5-2 contains an example program that illustrates how to use the synchronous input functions to get information from an input device. First, a `IC_LOCATOR` device is initialized and associated with a trigger (the left mouse button). The tracking method for the `IC_LOCATOR` is defined to be a printer's fist. Then measure of the `IC_LOCATOR` is requested with a timeout period of ten seconds. If the trigger is activated during this period, `request_input` returns a valid measure in `ivalue`. Finally, the `IC_LOCATOR` is dissociated from the mouse button and released. The program exits.
```c
#include <cgidefs.h>
#define TEN SECONDS (10 * 1000 * 1000)

main ()
{
    Cawresult stat;
    Ccoor point;
    Cinrep ivalue;
    Cint name;
    Cint trigger;
    Cvwsurf device;

    NORMAL_VWSURF(device, PIXWINDD);
    point.x = 16384;
    point.y = 16384;
    ivalue.xypt = &point;

    open cgi();
    open_vws(&name, &device);

    initialize_lid(IC LOCATOR, 1, &ivalue);
    associate(2, IC LOCATOR, 1);
    track_on(IC LOCATOR, 1, 1, (Ccoorpair *)0, &ivalue);
    request_input(IC LOCATOR, 1, TEN SECONDS,
                   &stat, &ivalue, &trigger);
    if (stat == VALID_DATA)
        printf("trigger activated at %d %d \n",
               ivalue.xypt->x, ivalue.xypt->y);
    else
        printf("trigger not activated \n");
    dissociate(2, IC LOCATOR, 1);
    release_input_device(IC LOCATOR, 1);
    close_vws(name);
    close.cgi();
}
```

**Figure 5-2  Example Program with LOCATOR Input Device**

**Request Input**

Cerror request_input(devclass, devnum, timeout, 
valid, sample, trigger)
Cdevoff devclass; /* device type */
Cint devnum; /* device number */
Cint timeout; /* amount of time to wait for input */
Cawresult *valid; /* device status */
Cinrep *sample; /* device value */
Cint *trigger; /* trigger number */

request_input waits `timeout` microseconds for activation of a trigger associated with a specific device. If `timeout` is negative, the request will wait forever.
request_input puts the input device in the RESPOND_EVENT state. If a trigger is activated within this period, the activating trigger and the device measure are returned in the trigger and sample arguments respectively. If the trigger is not activated within this period, the current device measure is returned in the sample argument and trigger is set to zero. Before returning, the input device is reset to NO_EVENTS state.

request_input returns a device status in the argument valid. This argument uses the enumerated type Cawresult (AWait Result) which contains values describing the state of an input device.

typedef enum {
    VALID_DATA,
    TIMED_OUT,
    DISABLED,
    WRONG_STATE,
    NOT_SUPPORTED
} Cawresult;

VALID_DATA indicates a trigger is activated within the specified timeout period. TIMED_OUT indicates that a trigger was not activated with a specified period. WRONG_STATE indicates SunCGI is not in state VSAC. NOT_SUPPORTED indicates the requested device is not a legal device.

If the appropriate field of the sample argument is a pointer, it must be set to an application program allocated area.

Errors

ENOTVSAC [4] CGI not in proper state: CGI shall be in state VSAC.
EINDNOEX [80] Input device does not exist.
EINDINIT [81] Input device not initialized.
EINEVNEN [94] Events not enabled.

5.3. Asynchronous Input

This section explains the asynchronous method of input device management where the application process and the input device process operate simultaneously. The designated input device is sampled with initiate_request and the measure of the input device is read with get_last_requested_input. Alternatively, the current measure of a device may be read with sample_input.

The example program in Figure E-2 demonstrates how to use the asynchronous input functions.

Initiate Request

Cerror initiate_request(devclass, devnum)
Cdevoff devclasSi /* device type */
Cint devnum; /* device number */

initiate_request sets up a device so that the measure resulting from the next trigger activation will be placed in the request register.
initiate_request puts the device in the REQUEST_EVENT state. It then returns to the calling function without waiting for a trigger activation. The value caused by the trigger activation can be obtained by the
The event queue is a single FIFO buffer that holds events from input devices. Since the event queue has a fixed length, it must be processed in a timely fashion or it will overflow. Events can be removed from the event queue in three ways: the event at the head of the event queue can be processed with `await_event`; the entire event queue can be emptied with `flush_event_queue`; and the events from a particular device can be removed from the event queue with `selective_flush_of_event_queue`.

Figure 5-3 contains an example program that illustrates how to use the event queue input functions to get information from an input device. First, a `IC_STRING` device is initialized and associated with a trigger (the keyboard). The tracking method for the `IC_STRING` is defined to be a string that echos the keyboard input on the bottom of the viewport. The `IC_STRING` is put into the `QUEUE_EVENT` state with `enable_events`. After the trigger fires, the measure of the `IC_STRING` device is determined with `await_event`. Finally, the `LOCATOR` is dissociated from the mouse button and released. The program then exits.
main()
{
    Cawresult valid;
    Ccoor point;
    Cdevoff devclass = IC_STRING;
    Ceqflow overflow;
    Cinrep ivalue;
    Cint devnum = 1;
    Cint name;
    Cint replost;
    Cint time_stamp;
    Cint timeout = (10 * 1000 * 1000); /* ten seconds */
    Cint tracktype = 2;
    Cint trigger = 1;
    Cmesstype message_link;
    Cqtype qstat;
    Cvwsurf device;

    NORMAL_VWSURF(device, PIXWINDD);
    point.x = 16384;
    point.y = 16384;
    ivalue.xyp = &point;
    ivalue.string = "This is a string";

    open_cgi();
    open_vws(&name, &device);

    initialize_lid(devclass, devnum, &ivalue);
    associate(trigger, devclass, devnum);
    track_on(devclass, devnum, tracktype,
            (Ccoorpair *)0, &ivalue);
    enable_events(devclass, devnum);
    await_event(timeout, &valid, &devclass, &devnum,
            &ivalue, &message_link, &replost, &time_stamp,
            &qstat, &overflow);
    printf("%s
", ivalue.string);
    disable_events(IC_STRING, devnum);
    dissociate(trigger, IC_STRING, devnum);
    release_input_device(IC_STRING, devnum);

    close_vws(name);
    close_cgi();
}
Enable Events

Cerror enable_events(devclass, devnum)
Cdevoff devclass; /* device type */
Cint devnum; /* device number */

enable_events allows a device in NO_EVENTS state to put events on the event queue. enable_events puts the input device in the QUEUE_EVENT state. An error is generated if the device specified by devclass or devnum does not exist or is not initialized.

Errors

ENOTVSAC [4]  CGI not in proper state: CGI shall be in state VSAC.
EINDNOEX [80] Input device does not exist.
EINDINIT [81] Input device not initialized.
EIAEVENEN [93] Events already enabled.

Await Event

Cerror await_event(timeout, valid, devclass, devnum, measure, message_link, replost, time_stamp, qstat, overflow)
Cint timeout; /* input timeout period */
Cawresult *valid; /* status */
Cdevoff *devclass; /* device type */
Cint *devnum; /* device number */
Cinrep *measure; /* device value */
Cmesstype *message_link; /* type of message */
Cint *replost; /* reports lost */
Cint *time_stamp; /* time_stamp */
Cqtype *qstat; /* queue status */
Ceqflow *overflow; /* event queue status */

await_event processes the event at the head of the event queue. valid is set to WRONG_STATE if SunCGI is not in state VSAC. If the event queue is EMPTY, then await_event waits timeout microseconds for a trigger to be activated. If timeout is less than 0, SunCGI waits until a trigger is activated. valid is set to VALID_DATA if a trigger is activated within the specified timeout period and TIMED_OUT otherwise.

If either the event queue is not empty or a trigger is activated, the class, number and value of the device generating the event are reported in the returned arguments devclass, devnum and measure. If the appropriate field of the measure argument is a pointer, it must be set to an application program allocated area.

If two events on the event queue have the same trigger but different values, the argument message_link is assigned to SIMULTANEOUS_EVENT_FOLLOWS; otherwise the argument message_link is set to SINGLE_EVENT. The enumerated type Cmesstype contains the following values:

typedef enum {
    SIMULTANEOUS_EVENT_FOLLOWS,
    SINGLE_EVENT
} Cmesstype;

The replost and time_stamp arguments should be ignored and are always zero. The returned argument qstat reports the queue status after an event is removed.
from the head of the event queue.

typedef enum {
    NOT_VALID,
    EMPTY,
    NON_EMPTY,
    ALMOST_FULL,
    FULL
} Cqtype;

qstat is set to EMPTY if the event queue has no pending events. qstat is set to
NON_EMPTY if the event queue has events pending, but is not FULL or
ALMOST_FULL. qstat is set to ALMOST_FULL if there is room for only one more
event on the event queue. qstat is set to FULL if there is no room for more events
on the event queue.

The argument overflow indicates whether the event queue has overflowed or not.
The enumerated type Ceqflow contains the following values:

typedef enum {
    NO_OFLO,
    OFLO
} Ceqflow;

Errors

ENOTVSAC [4]     CGI not in proper state: CGI shall be in state VSAC.

EINQOVFL [97]    Input queue has overflowed.

Flush Event Queue

Cerror flush_event_queue()

flush_event_queue discards all events in the event queue. The purpose of
flush_event_queue is to return the event queue to a stable state (NO_OFLO).
flush_event_queue does not affect the state of input devices. This function
should be used carefully to avoid throwing away mouse-ahead or type-ahead
inputs.

Errors

ENOTOPOP [5]     CGI not in proper state CGI shall be in either in state
                  VDOP, VSOP, or VSAC.

Selective Flush of Event
Queue

Cerror selective_flush_of_event_queue(devclass, devnum)

Cdevoff devclass; /* device type */
Cint devnum; /* device number */

selective_flush_of_event_queue discards all events in the event
queue which were generated by a specified device.
selective_flush_of_event_queue does not affect the state of the
specified input device. devclass and devnum must refer to an existing and ini-
tialized device or an error is produced. However, no error is returned if no events
from the specified device are pending.

Errors

ENOTOPOP [5]     CGI not in proper state CGI shall be in either in state
                  VDOP, VSOP, or VSAC.
5.5. Miscellaneous Input Functions

The functions described in this section can be used with several of the input device management techniques described in the previous sections. For example, sample_input can be used when a device is in either RESPOND_EVENT or QUEUE_EVENT state. Likewise, disable_events can be used in either of these states.

Sample Input

Cerror sample_input(devclass, devnum, valid, sample)  
Cdevoff devclass; /* device type */  
Cint devnum; /* device number */  
Clogical *valid; /* device status */  
Cinrep *sample; /* device value */

sample_input reports the current measure of the specified input device in the returned argument sample. The returned argument valid reports whether the device is initialized and prepared to receive an input. The current measure of the device may be set by a queued event, a requested event, or a device initialization depending on the state of the input device and the most recent trigger activation(s). See the introduction of this chapter for an explanation of the relationship between the measure of an input device and the state of an input device. If the appropriate field of the sample argument is a pointer, it must be set to an application program allocated area.

Errors

ENOTVSAC [4]  CGI not in proper state: CGI shall be in state VSAC.
EINDNOEX [80]  Input device does not exist.
EINDINIT [81]  Input device not initialized.

Get Last Requested Input

Cerror get_last_requested_input(devclass, devnum, valid, sample)  
Cdevoff devclass; /* device type */  
Cint devnum; /* device number */  
Clogical *valid; /* device status */  
Cinrep *sample; /* device value */

get_last_requested_input returns the contents of the request register. get_last_requested_input is usually used with initiate_request, but request_input also changes the contents of the request register. The returned argument valid indicates whether the device exists and is initialized. The returned argument sample reports the event in the request register. If no event is in the request register, the initial device value is reported. If the appropriate field of the sample argument is a pointer, it must be set to an application program allocated area.

Errors

ENOTVSAC [4]  CGI not in proper state: CGI shall be in state VSAC.
EINDNOEX [80]  Input device does not exist.
5.6 Status Inquiries

The current state of the input devices, triggers, and the event queue can be obtained by using the functions discussed in this section.

Inquire LID State List

Cerror inquire_lid_state_list(devclass, devnum, 
   valid, list)
Cdevoff devclass; /* device type */
Cint devnum; /* device number */
Clogical *valid; /* device supported at all */
Cstatelist *list; /* table of descriptors */

inquire_lid_state_list reports the status of a specific input device specified by devclass and devnum. The argument valid reports whether the device is supported at all. The list argument reports the track, associations, state and measure of the device in the appropriate elements of list. When checking the elements of list, first check the state element — if state is RELEASED, the other elements of list are undefined.

typedef struct {
   Clidstate state;
   Cpromstate prompt;
   Cackstate acknowledgement;
   Cinrep *current;
   Cint n;
   Cint *triggers;
   Cechotype echotyp;
   Cechostate echosta;
   Cint echodat;
} Cstatelist;

Errors

EINDEININIT [81]   Input device not initialized.

Disable Events

Cerror disable_events(devclass, devnum)
Cdevoff devclass; /* device type */
Cint devnum; /* device number */

disable_events puts the input device in the NO_EVENTS state. If the device is in RESPOND_EVENT state, the specified device is returned to NO_EVENTS state; the measure of the device is not changed by disable_events. If the device is in QUEUE_EVENT state, disable_events stops the specified device from putting events on the event queue. However, existing entries on the event queue are not removed and existing associations remain. devclass and devnum must refer to an existing and initialized device or an error is produced.

Errors

ENOTVSAC [4]   CGI not in proper state: CGI shall be in state VSAC.
EINDNOEX [80]   Input device does not exist.
EINDEININIT [81]   Input device not initialized.
EINEVNEN [94]   Events not enabled.
Inquire LID State

Cerror inquire_lid_state(devclass, devnum, valid, state)
Cdevoff devclass; /* device type */
Cint devnum; /* device number */
Clogical *valid; /* device supported at all */
Clidstate *state; /* table of descriptors */

inquire_lid_state reports the status of a specific input device specified by devclass and devnum. The argument valid reports whether the device is supported at all. The state argument (of type Clidstate) reports the current state of the specified input device.

typedef enum {
    RELEASE,
    NO_EVENTS,
    REQUEST_EVENT,
    RESPOND_EVENT,
    QUEUE_EVENT
} Clidstate;

Errors

ENOTVSAC [4]   CGI not in proper state: CGI shall be in state VSAC.
EINDNOEX [80]  Input device does not exist.

Inquire Trigger State

Cerror inquire_trigger_state(trigger, valid, list)
Cint trigger; /* trigger number */
Clogical *valid; /* trigger state */
Ctrigstate *list; /* trigger description table */

inquire_trigger_state describes the binding between a trigger and an input device. If the state element of the returned argument list is INACTIVE, no associations have been made with the trigger. An error is generated if the trigger does not exist.

typedef struct {
    Cactstate state; /* state */
    Cassocclid *assoc; /* list of associations */
} Ctrigstate;

Errors

ENOTVSAC [4]   CGI not in proper state: CGI shall be in state VSAC.
EINTRNEX [86]  Trigger does not exist.

Inquire Event Queue State

Cerror inquire_event_queue_state(qstat, qflow)
Cqtype * qstat; /* queue state */
Ceqflow * qflow; /* overflow indicator */

inquire_event_queue_state reports the status of the event queue. qstat indicates whether any events are pending. The argument qflow reports if the event queue is overflowing.
typedef enum {
    NOT_VALID,
    EMPTY,
    NON_EMPTY,
    ALMOST_FULL,
    FULL
} Cqtype;

typedef enum {
    NO_OFLO,
    OFLO
} Ceqflow;

Errors

ENOTVSAC [4]  CGI not in proper state: CGI shall be in state VSAC.
Differences between SunCore and SunCGI

Differences between SunCore and SunCGI .............................................. 103

A.1. Output Primitives ................................................................................... 103
    Output Aspects of SunCore not Supported by SunCGI ...................... 104
    Output Features of SunCGI not Available in SunCore ..................... 104

A.2. Segmentation ....................................................................................... 104

A.3. Differences in Input Functions between SunCore and SunCGI ....... 104
Differences between SunCore and SunCGI

This appendix provides an introduction to SunCGI for programmers who have programming experience with SunCore or graphics packages based on the ACM Core Graphics Specification. The three major differences between SunCore and SunCGI are in the areas of output primitives, segmentation, and input. While SunCore is generally a 'higher-level' package, SunCGI has capabilities which are not available in SunCore.

A.1. Output Primitives

The major differences in drawing objects to the screen between SunCore and SunCGI are that

1. SunCGI does not support 3D primitives, and
2. SunCGI does not have floating-point world coordinates or image transforms, and,
3. SunCGI does not support the concept of current position, and
4. SunCGI does not support textured color lookup table for monochrome devices.

However, SunCGI provides a wider variety of geometrical and raster primitives, and more control over the drawing of text. These differences are summarized in Table A-1.

<table>
<thead>
<tr>
<th>Feature</th>
<th>SunCore</th>
<th>SunCGI</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D Output Primitives</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Current Position</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Textured Color Lookup Tables</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Polygons with Invisible Edges</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Circles and Ellipses</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Cell Arrays</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Character Clipping</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Output Aspects of SunCore not Supported by SunCGI

SunCGI does not support 3D output primitives, current position, or textured color lookup tables for monochrome devices. Since 3D output primitives are not supported, no shading or lighting functions are provided either. Furthermore, no rotation or translation functions are provided. Therefore, if you want to rotate a geometrical output primitive, these operations must be done by your application program.

Since SunCGI does not maintain the current position of the output ‘cursor’, relative drawing functions such as polygon_rel_3 are not supported. However, the application programmer can implement this function by specifying all coordinates as a base register plus a constant. The base register can be used by the application program to maintain the value of the current position.

For monochrome devices, SunCore interprets the entries in the color lookup table with indices greater than one as patterns. SunCGI interprets all color lookup table entries greater than zero as black. Patterns in SunCGI are explicitly specified in the pattern table and invoked by using the PATTERN or HATCH interior styles. In addition, while patterns in SunCore are all 4×4 matrices, patterns in SunCGI have variable dimensions.

Output Features of SunCGI not Available in SunCore

SunCGI offers geometrical and raster primitives not available in SunCore, as well as increased control over the drawing of text. SunCGI provides circles and ellipses. SunCGI also supports the cell array which is a raster array whose element size is a function of the screen size. SunCGI clips characters in parts if the text precision is set to STROKE.

A.2. Segmentation

SunCGI does not support segmentation. This effect influences the effect of attribute calls. In SunCore, some attributes (for example, highlighting) apply to entire segments. Since no concept of segmentation exists in SunCGI, these attributes are not offered. Furthermore, SunCGI does not allow the saving or restoring of segments to the screen, so screen repainting functions must be completely defined by the application program, unless the view surface is initialized as a retained view surface and is not resized.

A.3. Differences in Input Functions between SunCore and SunCGI

SunCore provides device-specific functions for setting input device parameters and reading input from them. SunCGI provides no device dependent calls. SunCGI has three methods for obtaining the measure of input devices

1. by first activation (REQUEST EVENT),
2. by most recent activation (RESPOND EVENT), or
3. by mediating input requests through the event queue (QUEUE EVENT).

Furthermore, SunCGI allows the explicit binding of triggers (physical input devices) to logical input devices.
Unsupported Aspects of CGI
Unsupported Aspects of CGI

SunCGI does not support certain optional aspects of the proposed draft ANSI CGI standard. Most notably SunCGI does not support the full constellation of negotiation functions or tracking. SunCGI does not allow the resetting of coordinate type, coordinate precision or color specification mode because to do so would greatly reduce the speed of application programs written in SunCGI. Furthermore, SunCGI does not support echoing functions for input, but provides the tracking functions instead.

Table B-1  Unsupported Control Functions

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>vdc_type</td>
</tr>
<tr>
<td>vdc_precision_for_integer_points</td>
</tr>
<tr>
<td>vdc_precision_for_real_points</td>
</tr>
<tr>
<td>integer_precision</td>
</tr>
<tr>
<td>real_precision</td>
</tr>
<tr>
<td>index_precision</td>
</tr>
<tr>
<td>color_selection_mode</td>
</tr>
<tr>
<td>color_precision</td>
</tr>
<tr>
<td>color_index_precision</td>
</tr>
<tr>
<td>viewport_specification_mode</td>
</tr>
<tr>
<td>make_picture_current</td>
</tr>
</tbody>
</table>

Table B-2  Unsupported Input Functions

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>set_prompt_state</td>
</tr>
<tr>
<td>set_acknowledgement_state</td>
</tr>
<tr>
<td>echo_on</td>
</tr>
<tr>
<td>echo_off</td>
</tr>
<tr>
<td>echo_update</td>
</tr>
</tbody>
</table>

The following SunCGI functions are nonstandard (that is, are not in the standards document) and are included to make CGI easier to use. In addition, SunCGI has non-standard view surface arguments for certain control functions.
Table B-3  *Non Standard Control Functions*

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>open_cgi</td>
</tr>
<tr>
<td>open_vws</td>
</tr>
<tr>
<td>activate_vws</td>
</tr>
<tr>
<td>deactivate_vws</td>
</tr>
<tr>
<td>close_vws</td>
</tr>
<tr>
<td>close_cgi</td>
</tr>
</tbody>
</table>

Table B-4  *Non Standard Attribute Functions*

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>define_bundle_index</td>
</tr>
<tr>
<td>line_endstyle</td>
</tr>
<tr>
<td>set_globalDrawing_mode</td>
</tr>
<tr>
<td>pattern_with_fill_color</td>
</tr>
<tr>
<td>fixed_font</td>
</tr>
</tbody>
</table>

The Cinrep structure contains a presently unsupported *pick* field, for compatibility with future segment manipulation capabilities.
Type and Structure Definitions
This appendix provides a list of the structures and enumerated types used by SunCGI functions. In addition, a list of useful constants defined in `<cgiconstants.h>` is given.

```c
/*devices*/
#define BW1DD 1
#define BW2DD 2
#define CG1DD 3
#define PIXWINDD 4
#define CGPIXWINDD 5
#define GP1DD 6
#define CG2DD 7
#define VWSURF_NEWFLG 1

/* limits */
#define MAXVWS 5
#define MAXTRIG 6
#define MAXASSOC 5
#define MAXEVENTS 1024
#define MAXAESIZE 10 /* maximum number of AES table entries */
#define MAXNUMPATS 50 /* maximum number of pattern table entries */
#define MAXPATSIZE 256 /* maximum pattern size */
#define MAXPTS 1024 /* maximum number of pts per polygon */
#define MAXCHAR 256 /* maximum number of chars in a string */
#define OUTFUNS 67 /* number of output functions */
#define INFUNS 22 /* number of input functions */
#define SMALL_CHAR 6 /* minimum character size */
#define DEVNAMESIZE 20
```

The type and structure definitions that follow can be found in the header file `<cgidefs.h>`.

```c
typedef enum {
    ACK_ON,
    ACK_OFF
} Cackstate;

typedef enum {
    ACTIVE,
    INACTIVE
```
typedef enum {
  CLEAR,
  NO_OP,
  RETAIN
} Cactstate;

typedef enum {
  INDIVIDUAL,
  BUNDLED
} Casptype;

typedef struct {
  Cint n;
  Cdevoff *class;
  Cint *assoc;
} Cassoclid;

typedef enum {
  VALID_DATA,
  TIMED_OUT,
  DISABLED,
  WRONG_STATE,
  NOT_SUPPORTED
} Cawresult;

typedef enum {
  BITNOT,
  BITTRUE
} Cbitmaptype;

typedef enum {
  TRANSPARENT,
  OPAQUE
} Cbmode;

typedef struct {
  Clintype line_type;
  Cfloat line_width;
  Cint line_color;
  Clmartype marker_type;
  Cfloat marker_size;
  Cint marker_color;
  Clintype interior_style;
  Cint hatch_index;
  Cint pattern_index;
  Cint fill_color;
  Clintype perimeter_type;
  Cfloat perimeter_width;
  Cint perimeter_color;
  Cint text_font;
  Cprectype text_precision;
} Cstruct;
Appendix C — Type and Structure Definitions

Cfloat character_expansion;
Cfloat character_spacing;
Cint text_color;
} Cbunatt;

typedef struct {
    unsigned char *ra;
    unsigned char *ga;
    unsigned char *ba;
    Cint n;
} Ccentry;

typedef enum {
    OPEN,
    CLOSE
} Ccflag;

typedef struct {
    Cint numloc;
    Cint numval;
    Cint numstrk;
    Cint numchoice;
    Cint numstr;
    Cint numtrig;
    Csuptype event_queue;
    Csuptype asynch;
    Csuptype coord_map;
    Csuptype echo;
    Csuptype tracking;
    Csuptype prompt;
    Csuptype acknowledgement;
    Csuptype trigger_manipulation;
} Ccidesctab;

typedef enum {
    YES,
    NO
} Cchangetype;

typedef enum {
    CLIP,
    NOCLIP,
    CLIP_RECTANGLE
} Cclip;

typedef enum {
    CHORD,
    PIE
} Cclosetype;

typedef enum {
    REPLACE,
    AND,
typedef struct {
    Cint x;
    Cint y;
} Ccoor;

typedef struct {
    Ccoor *ptlist;
    Cint n;
} Ccoorlist;

typedef struct {
    Ccoor *upper;
    Ccoor *lower;
} Ccoorpair;

typedef enum {
    IC_LOCATOR,
    IC_STROKE,
    IC_VALUATOR,
    IC_CHOICE,
    IC_STRING,
    IC_PICK
} Cdevoff;

typedef enum {
    E_TRACK,
    E_ECHO,
    E_TRACK_OR_ECHO,
    E_TRACK_AND_ECHO
} Cechoav;

typedef struct {
    Cinrep *echos;
    Cint n;
} Cechodatalst;

typedef enum {
    ECHO_OFF,
    ECHO_ON,
    TRACK_ON
} Cechostate;

typedef struct {
    Cechostate *echos;
    Cint n;
} Cechostatelist;

typedef enum {

typedef struct {
    Cint n;
    Cechoav *elements;
    Cechotype *echos;
} Cechotypelist;

typedef enum {
    NATURAL,
    POINT,
    BEST_FIT
} Cendstyle;

typedef enum {
    NO_OFLO,
    OFLO
} Ceqflow;

typedef enum {
    NO_OFLO,
    OFLO
} Ceqflow;

typedef Cint Cerror;

typedef enum {
    INTERRUPT,
    NO_ACTION,
    POLL
} Cerrtype;

typedef enum {
    CLIP_RECT,
    VIEWPORT,
    VIEWSURFACE
} Cexttype;

typedef struct {
    Cintertype style;
    Cflag visible;
    Cint color;
    Cint hatch_index;
    Cint pattern_index;
} PrintersFist;
Cint index;
Cintype pstyle;
Cfloat pwidth;
Cint pcolor;
} Cfillatt;

typedef enum {
  OFF,
  ON
} Cflag;

typedef struct {
  Cint n;
  Cint *num;
  CsetType *value;
} Cflaglist;

typedef char Cchar;

typedef float Cfloat;

typedef enum {
  FREEZE,
  REMOVE
} Cfreeze;

typedef enum {
  LFT,
  CNTER,
  RGHT,
  NRMAL,
  CNT
} Chaligntype;

typedef enum {
  NO_INPUT,
  ALWAYS_ON,
  SETTABLE,
  DEPENDS_ON_LID
} Cinputability;

typedef struct {
  Ccoor *xypt;
  Ccoorlist *points;
  Cfloat val;
  Cint choice;
  Cchar *string;
  Cpick *pick;
} Cinrep;

typedef float Cfloat;

typedef int Cint;
typedef enum {
    HOLLOW,
    SOLIDI,
    PATTERN,
    HATCH
} Cintertype;

typedef struct {
    Clogical sample;
    Cchangetype change;
    Cint numassoc;
    Cint *trigassoc;
    Cliddescriptor prompt;
    Cliddescriptor acknowledgement;
    Cechotypelst *echo;
    Cchar *classdep;
    Cstatelist state;
} Cliddescript;

typedef enum {
    RELEASE,
    NO_EVENTS,
    REQUEST_EVENT,
    RESPOND_EVENT,
    QUEUE_EVENT
} Clidstate;

typedef struct {
    Cintype style;
    Cfloat width;
    Cint color;
    Cint index;
} Clinatt;

typedef enum {
    SOLID,
    DOTTED,
    DASHED,
    DASHED_DOTTED,
    DASH_DOT_DOTTED,
    LONG_DASHED
} Cintype;

typedef enum {
    L_FALSE,
    L_TRUE
} Clogical;

typedef struct {
    Cmartype type;
    Cfloat size;
    Cint color;
    Cint index;
} Cmartype;
typedef enum {
    DOT,
    PLUS,
    ASTERISK,
    CIRCLE,
    X
} Cmartype;

typedef enum {
    SIMULTANEOUS_EVENT_FOLLOWS,
    SINGLE_EVENT
} Cmesstype;

typedef enum {
    RIGHT,
    LEFT,
    UP,
    DOWN
} Cpathtype;

typedef struct {
    Cint cur_index;
    Cint row;
    Cint column;
    Cint *colorlist;
    Ccoor *point;
    Cint dx;
    Cint dy;
} Cpatternatt;

typedef struct {
    int segid;
    int pickid;
} Cpick;

typedef struct pixrect Cpixrect;

typedef enum {
    STRING,
    CHARACTER,
    STROKE
} Cprectype;

typedef enum {
    PROMPT_OFF,
    PROMPT_ON
} Cpromstate;

typedef enum {
    NOT_VALID,
    EMPTY,
}
typedef enum {
    NON_EMPTY, 
    ALMOST_FULL, 
    FULL 
} Cqtype;

typedef enum {
    ABSOLUTE, 
    SCALED 
} Cspecmode;

typedef struct {
    Clidstate state; 
    Cpromstate prompt; 
    Cackstate acknowledgement; 
    Cinrep *current; 
    Cint n; 
    Cint *triggers; 
    Cechotype echotyp; 
    Cechostate echosta; 
    Cint echodat; 
} Cstatelist;

typedef enum {
    NONE, 
    REQUIRED_FUNCTIONS_ONLY, 
    SOME_NON_REQUIRED_FUNCTIONS, 
    ALL_NON_REQUIRED_FUNCTIONS 
} Csuptype;

typedef struct {
    Cint fontset; 
    Cint index; 
    Cint current_font; 
    Cprectype precision; 
    Cfloat exp_factor; 
    Cfloat space; 
    Cint color; 
    Cint height; 
    Cfloat basex; 
    Cfloat basey; 
    Cfloat upx; 
    Cfloat upy; 
    Cpathtype path; 
    Chaligntype halign; 
    Cvaligntype valign; 
    Cfloat hcalind; 
    Cfloat vcalind; 
} Ctextatt;

typedef enum {
    NOT_FINAL, 
    FINAL 
} Ctextfinal;
typedef struct {
    Cchangetype change;
    Cassoclid *numassoc;
    Cint maxassoc;
    Cpromstate prompt;
    Cackstate acknowledgement;
    Cchar *name;
    Cchar *description;
} Ctrigdis;

typedef struct {
    Cactstate state;
    Cassoclid *assoc;
} Ctrigstate;

typedef enum {
    TOP,
    CAP,
    HALF,
    BASE,
    BOTTOM,
    NORMAL,
    CONT
} Cvaligntype;

typedef enum {
    INTEGER,
    REAL,
    BOTH
} Cvdctype;

typedef struct {
    Cchar screenname[DEVNAMESIZE];
    Cchar windowname[DEVNAMESIZE];
    Cint windowfd;
    Cint retained;
    Cint dd;
    Cint cmapsize;
    Cchar cmapname[DEVNAMESIZE];
    Cint flags;
    Cchar **ptr;
} Cvwsurf;
Error Messages

Error Messages ............................................................................................................................... 123

D.1. Successful Return (0) ............................................................................................................. 123
D.2. State Errors (1-5) ................................................................................................................ 123
D.3. Control Errors (10-16) ....................................................................................................... 124
D.4. Coordinate Definition (20-24) ........................................................................................ 124
D.5. Output Attributes (30-51) ............................................................................................... 125
D.6. Output Primitives (60-70) ............................................................................................... 128
D.7. Input (80-97) .................................................................................................................... 129
D.8. Implementation Dependent (110-112) ........................................................................... 131
D.9. Possible Causes of Visual Errors ................................................................................... 131
D.3. Control Errors (10-16)  

ENOTOPPOP [5]  
CGI not in proper state. CGI should be in state CGOP, VSOP, or VSAC. The function which generated the error requires that SunCGI is at least initialized. If this error is received, make sure that your application program has called open_cgi, or that it has not recently called close_cgi.

EVSIDINV [10]  
Specified view surface name is invalid. The view surface name specified by the name argument has never been opened or if it has been opened, it has since been closed. Corrective action involves opening the view surface or changing the value of the name argument.

ENOWSTYP [11]  
Specified view surface type does not exist. The application program has specified a type of view surface which is not supported by SunCGI. Corrective action involves changing the type of view surface.

EMAXVSOP [12]  
Maximum number of view surfaces already open. An attempt was made to open a view surface when the maximum number of view surfaces is already open. Corrective action involves removing one call to open_vws.

EVSNOTOP [13]  
Specified view surface not open. An attempt was made to close a view surface which is already closed. Corrective action involves removing one call to close_vws.

EVSISACT [14]  
Specified view surface is active. An attempt was made to activate a view surface which is already activated. Corrective action involves removing one call to activate_vws.

EVSNTACT [15]  
Specified view surface is not active. An attempt was made to deactivate a view surface which has already been deactivated. Corrective action involves removing one call to deactivate_vws.

EINQALTL [16]  
Inquiry arguments are longer than list. A call to inquiry negotiation function with indices greater than the number of supported functions was made. The returned list is always empty. Corrective action may be facilitated by obtaining the size of the list by using the inquire_device_class function.

D.4. Coordinate Definition (20-24)  

EBADRCTD [20]  
Rectangle definition is invalid. The application program has made a call to vdc_extent or device_viewport with the coordinates of both corners equal in the x or y dimensions or both. Corrective action involves changing one of the arguments to the function which generated the error so that the values of the two
arguments are different in both the $x$ and $y$ dimensions.

**EBDVIEW [21]**  
Viewport is not within Device Coordinates. A call to `device_viewport` has been made which specifies a viewport which is larger than the view surface. Corrective action involves making the arguments to `device_viewport` less than the view surface size. The size of the view surface can be obtained by calling the `inquire_physical_coordinate_system` function.

**ECLIPTOL [22]**  
Clipping rectangle is too large. The clipping rectangle would exceed the boundaries of VDC space. Corrective action involves resetting the clipping rectangle to be within limits of VDC space.

**ECLIPTOS [23]**  
Clipping rectangle is too small. The clipping rectangle would define an area of screen space smaller than one pixel. The clipping rectangle remains unchanged. Since the occurrence of this error is partially a function of the size of the view surface, changing the size of the view surface may be a viable alternative to changing the size of the clipping rectangle.

**EVDCSDIL [24]**  
VDC space definition is illegal. One or more of the arguments to the `vdc_extent` function exceeds the acceptable limits (-32767 to 32767) or coordinates of the lower-left hand corner are greater than the coordinates of the upper-right hand corner. Corrective action involves changing the arguments to `vdc_extent`.

**ASF is BUNDLED.** Error 16 is generated when attempting to call an individual attribute function when the attributes are specified by entries in the `attribute environment table`. Calls to these functions have no effect on the current attributes. Corrective action includes resetting the `attribute environment selector` to BUNDLED by using the `set_attribute_environment_selector` function.

**EBDBTBDI [31]**  
Bundle table index out of range. The entry in the `bundle table` exceeds the size of the table. The only corrective action is to change the value of the `index` argument.

**EBTUNDEF [32]**  
Bundle table index is undefined. The entry in the `attribute environment table` specified by the most recent call to `set_attribute_environment_table_index` has not been defined by SunCGI or the application program.
EBADLINX [33] Polyline index is invalid. The polyline bundle is not defined. Corrective action involves changing the index argument to polyline_bundle_index, or by defining the polyline bundle index.

EBDWIDTH [34] Width must be nonnegative. The width of a perimeter or line must be greater than or equal to zero. The current value of the perimeter width or line width remains unchanged. Changing the value of the width argument to a non-negative value will correct this error.

ECINDXLZ [35] Color index is less than zero. The value of the index argument to one of the attribute functions or the color entry in one of the bundles is negative. Corrective action involves changing the value of the color.

EBADCOLX [36] Color index is invalid. The color index argument to one of the attribute functions or the color entry in one of the bundles is not defined in the colormap. Indices in the color lookup table must be between 0 and 255 for the Sun 8-bit per pixel frame buffer. Any color specification outside of this range is ignored. Corrective action involves changing the value of the color.

EBADMRKX [37] Polymarker index is invalid. The polymarker bundle is not defined. Corrective action involves changing the index argument to polymarker_bundle_index, or by defining the polymarker bundle index.

EBADSIZE [38] Size must be nonnegative. The size of a marker or line must be greater or equal to zero. The current value of the marker size remains unchanged. Changing the value of the size argument to a non-negative value will correct this error.

EBADFABX [39] Fill area index is invalid. The fill area bundle is not defined. Corrective action involves changing the index argument to fill_area_bundle_index, or by defining the polymarker bundle index.

EPATARTL [40] Pattern array too large. The pattern array must contain less than 257 elements. The pattern is not entered into the pattern table. Corrective action involves designing a new pattern.

EPATSZTS [41] Pattern size too small. The pattern size must be at least two-by-two. The pattern is not entered into the pattern table. Corrective action could include designing a new pattern which includes several replications of the original pattern.
ESTYLLEZ [42] Style (pattern or hatch) index is less than zero. All indices in the pattern table must be positive. To fix this mistake, change the argument to the pattern_index or the hatch_index or the entries in the bundle table.

ENOPATNX [43] Pattern table index not defined. The argument to the hatch_index or pattern_index function or the entry bundle table should be reset to correspond to a defined value.

EPATITOL [44] Pattern table index too large. The index argument to pattern_table exceeded the bounds of the pattern table. The pattern is not entered into the pattern table. Redefining the pattern index to be between one and ten will eliminate the error.

EBADTXTX [45] Text index is invalid. The text bundle is not defined. Corrective action involves changing the index argument to text_bundle_index, or by defining the text bundle index.

EBDCHRIX [46] Character index is undefined. All other character indices besides 1 are undefined in SunCGI. The new character index is simply ignored. You are advised to ignore the character_index function entirely.

ETXTFLIN [47] Text font is invalid. The text fonts range from 1 to 6. All other integers do not correspond to actual fonts. Corrective action involves changing the argument to the text_font_index function or resetting the font index in the text bundle.

ECSEXFOOR [48] Expansion factor is out of range. The character expansion factor or the character space expansion factor would result in a character or a space which would exceed the bounds of the screen or would result in a character smaller than the limitations of the character drawing software. To eliminate this error, reset the offending value to within an acceptable range (0.1-2.0 are reasonable guidelines).

ECHHTLEZ [49] Character height is less than or equal to zero. The character height must be positive. Corrective action involves changing the argument to the character height function or the element of the text bundle.

ECHRUPVZ [50] Length of character up vector or character base vector is zero. Both the character up vector and the character base vector must be nonzero. Corrective action involves changing the arguments to the character_orientation function or the element of...
the text bundles.

ECOLRNGE [51] RGB values must be between 0 and 255. The red, green, and blue values are only defined between 0 and 255. The call to color_table which produced the error is ignored. Corrective action requires respecifying the values of the arguments to color_table.

ENMPTSTL [60] Number of points is too large. The number of points exceeds 255. Change the n element of the Coorlist structure to a value less than or equal to 255.

EPLMTWPT [61] Polygons must have at least two points. Change the n element of the Coorlist structure to a value greater than or equal to 2 and add the corresponding points to the ptlist element.

EPLMTWPT [62] Polygons must have at least three points. Change the n element of the Coorlist structure to a value greater than or equal to 3 and add the corresponding points to the ptlist element.

EGPLISFL [63] Global polygon list is full. The number of points on the global polygon list exceeds 256. The points which exceed 256 are ignored. This error can be corrected by inserting a call to polygon (which clears the global polygon list by displaying its contents) before the call to partial_polygon which caused the overflow.

EARCPNCI [64] Arc points do not lie on circle. The starting and ending points of either an open or close circular arc do not lie on the perimeter of the circle described by the arguments cl and rad. If this error occurs, the arc is not drawn. Corrective action may include determination of the endpoints with the application program (for example c2.x = rad*cos(start_angle);).

EARCPNEL [65] Arc points do not lie on ellipse. The starting and ending points of either an open or close elliptical arc do not lie on the perimeter of the ellipse described by the arguments c1,c2, and c3. If this error occurs, the arc is not drawn. Corrective action may include determination of the endpoints with the application program (see error 11).

ECELLATS [66] Cell array dimensions dx,dy are too small. The dimensions of the cell array are too small for a cell array element to be mapped onto one pixel of the view surface. The cell array is not drawn. This error depends on the physical size of the view surface as well as the limits of VDC space. Therefore, corrective action might require changing the size of the view surface, VDC.
Appendix D — Error Messages

D.7. Input (80-97)

**ECELLPOS [67]**  
Cell array dimensions must be positive. Negative cell array dimensions are not permitted. Corrective action requires changing the parameters to the cell array function.

**ECELLTLS [68]**  
Is not used.

**EVALOVWS [69]**  
Value outside of view surface. A coordinate of a pixel array is outside the physical range of the view surface. The pixel array is not drawn. Change the arguments to the pixel_array or bitblt_source_array.

**EPXNOTCR [70]**  
Pixrect not created. One of the BitBlt functions required a user-defined pixrect, and that pixrect had not been created. Corrective action involves creating a pixrect in your application program before calling the offending BitBlt function.

**EINDNOEX [80]**  
Input device does not exist. The input device specification (specified by the devclass and devnum arguments of most input functions) does not exist. Corrective action involves resetting the device specification to a valid device.

**EINDINIT [81]**  
Input device not initialized. A call to an input device function specified a device which was not initialized. Calls which generate this error have no effect. A call to initialize_input_device should be inserted before the call generating the error.

**EINDALIN [82]**  
Input device already initialized. An attempt to initialize a device which has previously been initialized. The parameters to the offending call to initialize_input_device are ignored. Removing the offending call to initialize_input_device will correct this error.

**EINASAEX [83]**  
Association already exists. An attempt is being made to bind the input device to a trigger to which it has been previously bound. The status of the input device trigger are unchanged. This error is purely informational and no corrective action is required.

**EINAIIMP [84]**  
Association is impossible. An attempt is being made to bind the input device to a trigger to which it cannot be bound. For example a IC_STRING device cannot be bound to a mouse button. To eliminate this error, change the arguments to the offending call of the associate function.
Association does not exist. An attempt to set-up call an input function which specifies a device with no associated triggers was made. The offending call is ignored. Corrective action involves calling associate before the offending call is issued.

Trigger does not exist. An attempt was made to associate or inquire about a trigger which has a number less than one or greater than five. The offending call is ignored. To eliminate the error, change the trigger number.

Input device does not echo. CHOICE devices do not support echo. Corrective action requires removing the call to echo_on from the application program.

Echo already on. A call to echo_on has been made to a device whose echoing ability has already been activated. To stop generation of the error either remove the offending call or change the arguments to specify a device whose echo is currently off.

Echo incompatible with existing echos. Although SunCGI can support certain combinations of echos (such as IC_STRING and IC_LOCATOR), not all combinations are supported. The easiest remedy is to remove the most recent call to echo_on from the application program.

Echoregion larger than view surface. Error 91 is generated when the rectangle defined by the echoregion argument exceeds the limits of VDC space. To eliminate this error, change the values to the echoregion argument to be within the confines of VDC space.

Echo type not supported. All devices except the IC_STROKE device only support one type of echo. Therefore, assigning a value to echotype other than zero or one will produce an error for any device except IC_STROKE. Corrective action involves changing the value of the echotype argument.

Echo not on. The device echoing has not been turned on. Either remove the call to echo_off, turn the echo on, or change the device specification.

Events already enabled. Events have already been enabled for the specified device. The solution is to remove the offending call to enable_events.

Events not enabled. Events have not been enabled for the specified device. The solution is to include a call to enable_events before a call to the
D.8. Implementation Dependent (110-112)

await_event, sample_event, or request_event function is made with the specified device as input parameter.

**EBADDATA [95]** Contents of input data record are invalid. The value argument of initialize_lid function is out of range or is the wrong type. The solution is to change the contents value argument.

**ESTRSIZE [96]** Length of initial string is greater than the implementation defined maximum. The initial string in the value argument is greater than 80 characters. Shorten the string.

**EINQOVFL [97]** Input queue has overflowed. The event queue can no longer record input events. Solutions include flushing the event queue or dequeueing events with the await_event, sample_event, or request_event function.

**EMEMSPAC [110]** Space allocation has failed. A function which was supposed to work has failed. The only action which you can take is to eliminate other processes which may be using memory. If you have eliminated all other processes, and this error is still generated, please contact SUN Microsystems.

**ENOTCSTD [111]** Function or argument not compatible with standard CGI. A function call is not supported by the CGI library.

**ENOTCCPW [112]** Function or argument not compatible with CGIPW mode. A function call is not supported by the cgipw library.

D.9. Possible Causes of Visual Errors
Table D-1  Possible Causes of Visual Errors

<table>
<thead>
<tr>
<th>Behavior</th>
<th>Possible Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segmentation fault for <code>open_vws</code></td>
<td><code>devdd</code> argument for</td>
</tr>
<tr>
<td></td>
<td><code>open_vws</code> is declared as a pointer (the address of <code>devdd</code> should be passed).</td>
</tr>
<tr>
<td>No primitives displayed</td>
<td>View surface not initialized.</td>
</tr>
<tr>
<td></td>
<td>View surface not active.</td>
</tr>
<tr>
<td></td>
<td>VDC to device coordinate mapping makes objects too small.</td>
</tr>
<tr>
<td></td>
<td>Clipping rectangle is too small and clipping is ON.</td>
</tr>
<tr>
<td></td>
<td>Perimeter visibility is set to OFF and interior style is set to HOLLOW.</td>
</tr>
<tr>
<td></td>
<td><code>line color or fill color</code> is set to background color.</td>
</tr>
<tr>
<td>Primitives displayed on undesired view surfaces</td>
<td>Undesired view surfaces have not been deactivated.</td>
</tr>
<tr>
<td>Segmentation fault for inquiry functions</td>
<td>passing variable instead of address (&amp;) of variable.</td>
</tr>
</tbody>
</table>

- `devdd`: device driver
- `open_vws`: open view surface

Version B of 17 February 1991
<table>
<thead>
<tr>
<th>Behavior</th>
<th>Possible Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polylines or polylines aren’t displayed.</td>
<td>Width or size is zero.</td>
</tr>
<tr>
<td>Polygon borders aren’t displayed.</td>
<td>Width is zero.</td>
</tr>
<tr>
<td>Circles aren’t displayed.</td>
<td>Width or size is zero.</td>
</tr>
<tr>
<td>Ellipses aren’t displayed.</td>
<td>Width or size is zero.</td>
</tr>
<tr>
<td>Text isn’t displayed.</td>
<td>Width or size is zero.</td>
</tr>
<tr>
<td>Cell arrays aren’t displayed.</td>
<td>$dx$ or $dy$ arguments are too small.</td>
</tr>
<tr>
<td>Cell arrays aren’t displayed on all active view surfaces.</td>
<td>Mapping from cell size to view surface for smaller view surfaces is too small.</td>
</tr>
<tr>
<td>Pixel arrays aren’t displayed.</td>
<td>Location is outside of view surface or clipping rectangle.</td>
</tr>
<tr>
<td>BitBlts aren’t displayed.</td>
<td>Width or size is zero.</td>
</tr>
</tbody>
</table>

*Primitive-Specific Errors*
### Table D-3  **Attribute Errors**

<table>
<thead>
<tr>
<th>Behavior</th>
<th>Possible Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute setting has no effect</td>
<td>attribute ASF is set to BUNDLED.</td>
</tr>
<tr>
<td>Text attributes have no effect</td>
<td>text precision is set to CHARACTER.</td>
</tr>
<tr>
<td>PATTERN fill is the same as HATCH</td>
<td>attribute ASF is set to BUNDLED.</td>
</tr>
<tr>
<td>PATTERN fill is different on different view surfaces.</td>
<td>pattern index and hatch index are identical pattern size is too small</td>
</tr>
</tbody>
</table>

### Table D-4  **Input-specific Errors**

<table>
<thead>
<tr>
<th>Behavior</th>
<th>Possible Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input device does not report</td>
<td>device not initialized</td>
</tr>
<tr>
<td>Input device does not echo</td>
<td>echo not initialized</td>
</tr>
<tr>
<td>Input device does not echo on whole view surface</td>
<td>echo region not set to whole view surface.</td>
</tr>
</tbody>
</table>
Sample Programs

Sample Programs ................................................................. 137

E.1. Martini Glass ................................................................. 137
E.2. Tracking Box ................................................................. 138
Sample Programs

E.1. Martini Glass

The following program draws a martini glass. The program exits after 10 seconds.
```c
#include <cgidefs.h>

Ccoorlist martinilist;
Ccoor glass_coords[10] = { 0,0,  
                           -10,0,  
                           -1,1,  
                           -1,20,  
                           -15,35,  
                           15,35,  
                           1,20,  
                           1,1,  
                           10,0,  
                           0,0 };  

Ccoor water_coords[2] = { -12,33,  
                           12,33 };  

Ccoor vpll = { -50,-10 };  
Ccoor vpur = { 50,80 };  

main()
{
  Cvwsurf device;
  Cint name;

  NORMAL_VWSURF(device, PIXWINDD);

  open_cgi();
  open_vws(&name, &device);
  vdc_extent(&vpll, &vpur);

  martinilist.ptlist = glass_coords;  
  martinilist.n = 10;
  polyline(&martinilist);
  martinilist.ptlist = water_coords;  
  martinilist.n = 2;
  polyline(&martinilist);

  sleep(10);
  close_vws(name);
  close_cgi();
}
```

**Figure E-1**  
*Martini Glass Example Program*

**E.2. Tracking Box**  
The following program demonstrates the use of the CGI input functions. A square is displayed on the screen and moved with the mouse. The program exits if the mouse is still for five seconds.
```c
#include <cgidefs.h>
#define DEVNUM 1 /* device number */
#define MOUSE_POSITION 5 /* trigger number */
#define TIMEOUT (5 * 1000 * 1000) /* timeout in microseconds */

Ccoor ulc = {1000, 2000};
Ccoor lrc = {2000, 1000};

main()
{
    Cint name;
    Cvwsurf device;
    Cawresult stat;
    Cinrep sample; /* device measure value */
    Ccoor samp; /* LOCATOR's x,y position */
    Cint trigger; /* trigger number */

    NORMAL_VWSURF(device, PIXWINDD);
    sample.xypt = &samp;
    samp.x = 0;
    samp.y = 27000;

    open_cgi();
    open_vws(&name, &device);
    set_global_drawing_mode(XOR);
    initialize_lid(IC_LOCATOR, DEVNUM, &sample);
    associate(MOUSE_POSITION, IC_LOCATOR, DEVNUM);
    rectangle(&lrc, &ulc); /* draw first rectangle */
    /* wait TIMEOUT micro-seconds for input and check the status */
    while (request_input(IC_LOCATOR, DEVNUM, TIMEOUT, &stat, &sample, &trigger), (stat == VALID_DATA)) {
        if ((sample.xypt->x != ulc.x) || (sample.xypt->y != lrc.y)) {
            rectangle(&lrc, &ulc);
            lrc.y = sample.xypt->y; /* move to new location */
            lrc.x = (sample.xypt->x + 1000);
            ulc.x = sample.xypt->x;
            ulc.y = (sample.xypt->y + 1000);
            rectangle(&lrc, &ulc);
        }
    }
    dissociate(MOUSE_POSITION, IC_LOCATOR, DEVNUM);
    release_input_device(IC_LOCATOR, DEVNUM);
    close_vws(name);
    close_cgi();
}
```

Figure E-2  Tracking Box Example Program
Using SunCGI and Pixwins

F.1. cgipw Functions ................................................................. 143
    Open Pixwin CGI .............................................................. 143
    Open a CGI Pixwin .......................................................... 143
    Close a CGI Pixwin ......................................................... 144
    Close Pixwin CGI ............................................................. 144
F.2. Using cgipw ................................................................. 144
F.3. cgipw Functions ............................................................. 145
F.4. Example Program ............................................................ 147
F

Using SunCGI and Pixwins

The CGI standard does not provide facilities for dealing with multiple overlapping windows. An application program can use SunCGI and Pixwins features through the cgipw functions. These functions combine the richness of CGI's primitives with the ability of Pixwins to manage multiple (potentially overlapping) windows.

This appendix assumes familiarity with both SunCGI and Pixwins. See SunView Programmer's Guide for more information on Pixwins. An example program is included at the end of this appendix in Figure F-1.

If you decide to use CGI and Pixwins, you may not use the standard SunCGI calls. Instead you must use cgipw calls. For example, cgipw polyline replaces polyline. The first argument of each cgipw function is a pixwin descriptor of type Ccgiwin. The file <cgipw.h> must be included in the cgipw application program instead of <cgefis.h>.

F.1. cgipw Functions

The four functions open_pw_cgi, open_cgi_pw, close_cgi_pw and close_pw_cgi are necessary for managing the SunCGI–Pixwins interface.

Open Pixwin CGI

Cerror open_PW_cgi()

open_PW_cgi initializes CGI by setting the attributes to the default values and setting the VDC to device coordinate mapping to 1:1. Therefore, all input and output primitives will use device coordinates. The origin of the device coordinates is in the upper left-hand corner instead of the lower left-hand corner. The entire window is used, not just a square region within it. No standard errors are specified for open_PW_cgi. If open_PW_cgi returns a nonzero result, then the initialization failed. open_PW_cgi corresponds to open_cgi.

Open a CGI Pixwin

Cerror open_cgi_pw(pw, desc, name)
struct pixwin *pw; /* pixwin */
Ccgiwin *desc; /* CGI pixwin descriptor */
Cint *name;

open_cgi_pw informs CGI of the pixwin pointed to by pw. Calls to CGI primitives may then reference this pixwin. However, CGI does not guarantee that a pixwin exists or is in any other way properly initialized. desc is a pointer to a CGI pixwin descriptor allocated by the application program and defined by open_cgi_pw. It will be used as the first argument to cgipw functions. Calls
may also be made to any pixwin function (see example program). Multiple calls to open_cgi_pw with pointers to different Ccgiwin structures will allow primitives to be displayed on multiple view surfaces by repeating calls to cgipw functions with different Ccgiwin descriptors. Attributes are local to the pixwin associated with the CGI descriptor passed to the cgipw attribute functions. open_cgi_pw corresponds to open_vws. open_pw_cgi must be called prior to open_cgi_pw; otherwise, error 111 is returned. Other errors (as with open_vws may also be detected.

Errors
ENOTOP [5] CGI not in proper state CGI shall be either in state CGOP, VSOP, or VSAC.
EMEMSPAC [110] Space allocation has failed.

Close a CGI Pixwin
Cerror close_cgi_pw(desc)
Ccgiwin *desc; /* CGI pixwin descriptor */
close_cgi_pw takes the CGI pixwin descriptor desc as an argument and removes it from the list of pixwins that CGI writes to. The pixwin is not closed.
close_cgi_pw corresponds to close_vws, and may return any of the errors close_vws detects (except 112).

Errors
ENOTOP [5] CGI not in proper state CGI shall be either in state CGOP, VSOP, or VSAC.
EVSIDINV [10] Specified view surface name is invalid.

Close Pixwin CGI
Cerror close_pw_cgi()
close_pw_cgi takes care of leaving CGI in an orderly state. This function should be called before exiting the application program. close_pw_cgi corresponds to close_cgi.

Errors
ENOTOP [5] CGI not in proper state CGI should be in state CGOP, VSOP, or VSAC.

F.2. Using cgipw
After calling the two initialization functions (open_pw_cgi and open_cgi_pw) the application program may call functions from both the Pixwins and cgipw libraries. Figure F-1 contains an example program that uses cgipw functions.

Since cgipw functions use a 1:1 mapping from VDC to device coordinates, attributes in VDC units (such as pattern size and character height) will be huge unless they are reset. And because the cgipw origin is the device coordinate origin, the upper left-hand corner, attributes with direction or position (e.g., pattern reference point and character orientation) have their meaning reversed in
the y dimension.

Most cgipw functions do not print error messages even if the error warning mask is INTERRUPT or POLL. They all return error codes which may be tested. The application program should not use both SunCGI and window system input functions, since both SunCGI and the window system share a common event queue. For example, events handled by a SunCGI function will not be handled by a window system call after the SunCGI call.

A list of the cgipw functions and their corresponding SunCGI functions is given in Table F-1 below. If a function is not included in this table, then use the normal SunCGI function except as described below in Table F-2. Most of the functions listed below are output and attribute functions; however, the tracking functions are listed so that you can control which surfaces input devices echo on. The arguments of the cgipw functions are the same as those of the SunCGI functions except that the first argument is always a desc argument of type Ccgiwin. desc is a pointer to a pixwin descriptor filled in by the open_cgi_pw function.

Table F-1 contains a list of functions available in cgipw mode. SunCGI functions incompatible with cgipw mode are given in Table F-2. partial_polygon may be used with cgipw_polygon, but the global polygon list is freed after use by cgipw_polygon, so calls to partial_polygon must be repeated prior to use of cgipw_polygon on another view surface.

<table>
<thead>
<tr>
<th>SunCGI Function Name</th>
<th>cgipw Function Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>append_text(flag, tstring)</td>
<td>cgipw_append_text(desc, flag, tstring)</td>
</tr>
<tr>
<td>cell_array(p, q, r, dx, dy, colorind)</td>
<td>cgipw_cell_array(desc, p, q, r, dx, dy, colorind)</td>
</tr>
<tr>
<td>character_expansion_factor(sfac)</td>
<td>cgipw_character_expansion_factor(desc, sfac)</td>
</tr>
<tr>
<td>character_height(height)</td>
<td>cgipw_character_height(desc, height)</td>
</tr>
<tr>
<td>character_orientation(xup, yup, xbase, ybase)</td>
<td>cgipw_character_orientation(desc, xup, yup, xbase, ybase)</td>
</tr>
<tr>
<td>character_path(path)</td>
<td>cgipw_character_path(desc, path)</td>
</tr>
<tr>
<td>character_set_index(index)</td>
<td>cgipw_character_set_index(desc, index)</td>
</tr>
<tr>
<td>character_spacing(spcratio)</td>
<td>cgipw_character_spacing(desc, spcratio)</td>
</tr>
<tr>
<td>circle(c1, rad)</td>
<td>cgipw_circle(desc, c1, rad)</td>
</tr>
<tr>
<td>circular_arc_3pt(c1, c2, c3)</td>
<td>cgipw_circular_arc_3pt(desc, c1, c2, c3)</td>
</tr>
<tr>
<td>circular_arc_3pt_close(c1, c2, c3, close)</td>
<td>cgipw_circular_arc_3pt_close(desc, c1, c2, c3, close)</td>
</tr>
<tr>
<td>circular_arc_center(c1, c2x, c2y, c3x, c3y, rad)</td>
<td>cgipw_circular_arc_center(desc, c1, c2x, c2y, c3x, c3y, rad)</td>
</tr>
<tr>
<td>circular_arc_center_close(c1, c2x, c2y, c3x, c3y, rad, close)</td>
<td>cgipw_circular_arc_center_close(desc, c1, c2x, c2y, c3x, c3y, rad, close)</td>
</tr>
<tr>
<td>color_table(istart, clist)</td>
<td>cgipw_color_table(desc, istart, clist)</td>
</tr>
<tr>
<td>define_bundle_index(index)</td>
<td>cgipw_define_bundle_index(desc, index)</td>
</tr>
<tr>
<td>disjoint_polyline(polycoors)</td>
<td>cgipw_disjoint_polyline(desc, polycoors)</td>
</tr>
<tr>
<td>ellipse(c1, majx, miny)</td>
<td>cgipw_ellipse(desc, c1, majx, miny)</td>
</tr>
<tr>
<td>SunCGI Function Name</td>
<td>cgipw Function Name</td>
</tr>
<tr>
<td>----------------------------------------------------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>elliptical_arc(cl, sx, sy, ex, ey, majx, miny)</td>
<td>cgipw_elliptical_arc(desc, cl, sx, sy, ex, ey, majx, miny)</td>
</tr>
<tr>
<td>elliptical_arc_close(cl, sx, sy, ex, ey, majx, miny, close)</td>
<td>cgipw_elliptical_arc_close(desc, cl, sx, sy, ex, ey, majx, miny, close)</td>
</tr>
<tr>
<td>fill_area_bundle_index(index)</td>
<td>cgipw_fill_area_bundle_index(desc, index)</td>
</tr>
<tr>
<td>fill_color(color)</td>
<td>cgipw_fill_color(desc, color)</td>
</tr>
<tr>
<td>fixed_font(index)</td>
<td>cgipw_fixed_font(desc, index)</td>
</tr>
<tr>
<td>hatch_index(index)</td>
<td>cgipw_hatch_index(desc, index)</td>
</tr>
<tr>
<td>inquire_aspect_source_flags()</td>
<td>cgipw_inquire_aspect_source_flags(desc)</td>
</tr>
<tr>
<td>inquire_drawing_mode(visibility, source, destination, combination)</td>
<td>cgipw_inquire_drawing_mode(desc, visibility, source, destination, combination)</td>
</tr>
<tr>
<td>inquire_fill_area_attributes()</td>
<td>cgipw_inquire_fill_area_attributes(desc)</td>
</tr>
<tr>
<td>inquire_line_attributes()</td>
<td>cgipw_inquire_line_attributes(desc)</td>
</tr>
<tr>
<td>inquire_marker_attributes()</td>
<td>cgipw_inquire_marker_attributes(desc)</td>
</tr>
<tr>
<td>inquire_pattern_attributes()</td>
<td>cgipw_inquire_pattern_attributes(desc)</td>
</tr>
<tr>
<td>inquire_pixel_array(p, m, n, colorind)</td>
<td>cgipw_inquire_pixel_array(desc, p, m, n, colorind)</td>
</tr>
<tr>
<td>inquire_text_attributes()</td>
<td>cgipw_inquire_text_attributes(desc)</td>
</tr>
<tr>
<td>inquire_text_extent(tstring, nextchar, concat, lleft, uleft, uright)</td>
<td>cgipw_inquire_text_extent(desc, tstring, nextchar, concat, lleft, uleft, uright)</td>
</tr>
<tr>
<td>interior_style(istyle, perimvis)</td>
<td>cgipw_interior_style(desc, istyle, perimvis)</td>
</tr>
<tr>
<td>line_color(index)</td>
<td>cgipw_line_color(desc, index)</td>
</tr>
<tr>
<td>line_endstyle(ttyp)</td>
<td>cgipw_line_endstyle(desc, ttyp)</td>
</tr>
<tr>
<td>line_type(ttyp)</td>
<td>cgipw_line_type(desc, ttyp)</td>
</tr>
<tr>
<td>line_width(index)</td>
<td>cgipw_line_width(desc, index)</td>
</tr>
<tr>
<td>line_width_specification_mode(mode)</td>
<td>cgipw_line_width_specification_mode(desc, mode)</td>
</tr>
<tr>
<td>marker_color(index)</td>
<td>cgipw_marker_color(desc, index)</td>
</tr>
<tr>
<td>marker_size(index)</td>
<td>cgipw_marker_size(desc, index)</td>
</tr>
<tr>
<td>marker_size_specification_mode(mode)</td>
<td>cgipw_marker_size_specification_mode(desc, mode)</td>
</tr>
<tr>
<td>marker_type(ttyp)</td>
<td>cgipw_marker_type(desc, ttyp)</td>
</tr>
<tr>
<td>pattern_index(index)</td>
<td>cgipw_pattern_index(desc, index)</td>
</tr>
<tr>
<td>pattern_reference_point(open)</td>
<td>cgipw_pattern_reference_point(desc, open)</td>
</tr>
<tr>
<td>pattern_size(dx, dy)</td>
<td>cgipw_pattern_size(desc, dx, dy)</td>
</tr>
<tr>
<td>perimeter_color(index)</td>
<td>cgipw_perimeter_color(desc, index)</td>
</tr>
<tr>
<td>perimeter_type(ttyp)</td>
<td>cgipw_perimeter_type(desc, ttyp)</td>
</tr>
<tr>
<td>perimeter_width(width)</td>
<td>cgipw_perimeter_width(desc, width)</td>
</tr>
<tr>
<td>perimeter_width_specification_mode(mode)</td>
<td>cgipw_perimeter_width_specification_mode(desc, mode)</td>
</tr>
<tr>
<td>pixel_array(pcell, m, n, colorind)</td>
<td>cgipw_pixel_array(desc, pcell, m, n, colorind)</td>
</tr>
<tr>
<td>polygon(polycoors)</td>
<td>cgipw_polygon(desc, polycoors)</td>
</tr>
<tr>
<td>polyline(polycoors)</td>
<td>cgipw_polyline(desc, polycoors)</td>
</tr>
<tr>
<td>polyline_bundle_index(index)</td>
<td>cgipw_polyline_bundle_index(desc, index)</td>
</tr>
<tr>
<td>polymarker(polycoors)</td>
<td>cgipw_polymarker(desc, polycoors)</td>
</tr>
<tr>
<td>polymarker_bundle_index(index)</td>
<td>cgipw_polymarker_bundle_index(desc, index)</td>
</tr>
<tr>
<td>rectangle(lrc, ulc)</td>
<td>cgipw_rectangle(desc, lrc, ulc)</td>
</tr>
<tr>
<td>set_aspect_source_flags(flags)</td>
<td>cgipw_set_aspect_source_flags(desc, flags)</td>
</tr>
<tr>
<td>text(cl, tstring)</td>
<td>cgipw_text(desc, cl, tstring)</td>
</tr>
</tbody>
</table>
Table F-1  *List of cgipw Functions—Continued*

<table>
<thead>
<tr>
<th>SunCGI Function Name</th>
<th>cgipw Function Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>text_alignment(halign, valign, hcalind, vcalind)</td>
<td>cgipw_text_alignment(desc, halign, valign, hcalind, vcalind)</td>
</tr>
<tr>
<td>text_bundle_index(index)</td>
<td>cgipw_text_bundle_index(desc, index)</td>
</tr>
<tr>
<td>text_color(index)</td>
<td>cgipw_text_color(desc, index)</td>
</tr>
<tr>
<td>text_font_index(index)</td>
<td>cgipw_text_font_index(desc, index)</td>
</tr>
<tr>
<td>text_precision(ttyp)</td>
<td>cgipw_text_precision(desc, ttyp)</td>
</tr>
<tr>
<td>vdm_text(cl, flag, tstring)</td>
<td>cgipw_vdm_text(desc, cl, flag, tstring)</td>
</tr>
</tbody>
</table>

Table F-2  *SunCGI Functions not Compatible with cgipw Mode*

<table>
<thead>
<tr>
<th>Function</th>
<th>Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>clear_control</td>
<td>All clear extents are identical</td>
</tr>
<tr>
<td>clip_indicator</td>
<td>when cflag is CLIP_RECTANGLE</td>
</tr>
<tr>
<td>clip_rectangle</td>
<td>Instead, use pw_region prior to open_cgi_pw</td>
</tr>
<tr>
<td>close_cgi</td>
<td>Use close_pw_cgi</td>
</tr>
<tr>
<td>close_vws</td>
<td>Use close_cgi</td>
</tr>
<tr>
<td>device_viewport</td>
<td>use pw_region prior to open_cgi_pw</td>
</tr>
<tr>
<td>open_cgi</td>
<td>Use open_pw_cgi</td>
</tr>
<tr>
<td>open_vws</td>
<td>Use open_cgi</td>
</tr>
<tr>
<td>partial_polygon</td>
<td><em>global polygon list</em> is freed after cgipw_polygon</td>
</tr>
<tr>
<td>vdc_extent</td>
<td>cgipw’s VDC space is identical to screen space</td>
</tr>
</tbody>
</table>

F.4. Example Program

Figure F-1 contains an example program that uses cgipw functions. This example uses retained pixwins to ease redisplay after window obstruction (see Section 2.3). This makes the program slower during image generation, because it writes both on the screen and onto a copy retained in memory.
#include <cgipw.h>
#include <suntool/gfxsw.h>

struct pixwin *mypw;
struct gfxsubwindow *mine;

main()
{
    Ccgiwin vpw;
    Ccoor bottom;
    Ccoor top;
    int name;
    int op;

    mine = gfxsw_init(0, 0);
gfxsw_getretained(mine);
    mypw = mine->gfx_pixwin;
pw_writebackground(mypw, 0, 0,
        mypw->pw_prretained->pr_size.x,
        mypw->pw_prretained->pr_size.y, PIX_CLR);

    open_pw.cgi();
    open_cgi_pw(mypw, &vpw, &name);
    op = PIX_COLOR(1) | PIX_SRC;
    pw_write(mypw, 0, 0, 100, 100,
        op, 0, 0, 0);
    bottom.x = 300;
    bottom.y = 100;
    top.x = 200;
    top.y = 0;
    cgipw_interior_style(&vpw, SOLIDI, ON);
    cgipw_rectangle(&vpw, &bottom, &top);
    sleep(10);

    close_cgi_pw(&vpw);
    close_pw.cgi();
}

Figure F-1  Example cgipw Program
Using SunCGI with Fortran Programs

Using SunCGI with Fortran Programs ......................................................... 151

G.1. Programming Tips .................................................................................. 151
G.2. Example Program ................................................................................... 152
G.3. FORTRAN Interfaces to SunCGI ......................................................... 154
Using SunCGI with Fortran Programs

All functions provided in SunCGI may be called from FORTRAN programs by linking them with the libcgi77.a library. This is done by using the f77 compiler with a command line like:

```
$ f77 -o box box.f -lcgi77 -lcgi -lsunwindow -lpixrect -lm
```

where box.f is the FORTRAN source program. Note that libcgi.a must be linked with the program (the -lcgi option), and libcgi77.a must precede it (the -lcgi77 option).

Defined constants may be referenced in source programs by including cgidefs77.h. In a FORTRAN program, this must be done via a source statement like:

```
include 'cgidefs77.h'
```

This include statement must be in each FORTRAN program unit which uses the defined constants, not just once in each source program file.

In the Sun release of FORTRAN, names are restricted to sixteen characters in length and may not contain the underline character. For this reason, FORTRAN programs must use abbreviated names to call the corresponding SunCGI functions. The correspondence between the full SunCGI names and the FORTRAN names appears later in this appendix. In addition, FORTRAN declarations for all SunCGI functions appear at the end of this appendix.

G.1. Programming Tips

- The abbreviated names of the SunCGI functions are less readable than the full length names because the underline character cannot be used in the FORTRAN names. However, since FORTRAN doesn't distinguish between upper-case and lower-case letters in names, upper-case characters can be used to improve readability. There is an example of this later in this appendix.

- Character strings passed from FORTRAN programs to SunCGI cannot be longer than 256 characters.

- Pointers returned by C functions are handled in FORTRAN as integer*4 values, and exist solely to be passed to other Sun graphics functions.

- FORTRAN passes all arguments by reference. Although some SunCGI functions receive arguments by value, the FORTRAN programmer need not worry
about this. The interface routines in /usr/lib/libcgi77.a handle this situation correctly. When in doubt, look at the FORTRAN declarations for SunCGI functions at the end of this appendix.

- Some SunCGI functions have structures as arguments or return values. These are handled in FORTRAN by unbundling the structures into separate arguments. In general, these will be in the same order, and have the same names, as the members of the C structures. One exception is the Ccoorlist structure, which is replaced in FORTRAN with an array of x’s, and one of y’s, rather than an array of x-y pairs. You may need to consult both the C and FORTRAN documentation to determine which FORTRAN arguments are input values, and which are output.

- Since FORTRAN does not distinguish between upper-case letters and lower-case letters in identifiers, any FORTRAN program unit which includes the cgidefs77.h header file cannot use the same spelling as any constant defined in that header file, regardless of case.

- The function cfqoutcap returns the FORTRAN binding names of the output capabilities, rather than the C bindings. This is an exception to the rule that the FORTRAN library provides a transparent interface to the C functions.

G.2. Example Program

This example is the FORTRAN equivalent of the very simple program for drawing a martini glass.
program test

parameter (ibignum=256)

integer name
character screenname* (ibignum)
integer screenlen
character windowname* (ibignum)
integer windowlen
integer windowfd
integer retained
integer dd
integer cmapsize
character cmapname* (ibignum)
integer cmaplen
integer flags
character ptr* (ibignum)
integer noargs

coordinates of glass
integer xc(10), yc(10), n
coordinates of waterline.
integer xc2(2), yc2(2)
data xc /0, -10, -1, -1, -15, 15, 1, 1, 10, 0 /
data yc /0, 0, 1, 20, 35, 35, 20, 1, 0, 0 /
data xc2 /-12, 12 /
data yc2 /33, 33 /

open cgi
  call cfopencgi()
open a pixwin
  dd = 4
  call cfopenvws(name, screenname, screenlen, windowname,
     + windowlen, windowfd, retained, dd, cmapsize,
     + cmapname, cmaplen, flags, ptr, noargs)
reset VDC space
  call cfvdcext(-50, -10, 50, 80)
draw martini glass and waterline
  n = 10
  call cfpolyline(xc, yc, n)
  n = 2
  call cfpolyline(xc2, yc2, n)
sleep for 10 seconds
  call sleep(10)
close and exit
  call cfclosecgi()
call exit()
end
Note: Although all SunCGI procedures are declared here as functions, each may also be called as a subroutine if the user does not want to check the returned value.

### Table G-1  
*SunCGI Fortran Binding – Part I*

<table>
<thead>
<tr>
<th>CGI Specification Name</th>
<th>Fortran Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Activate View Surface</strong></td>
<td>integer function cfactvws(name)</td>
</tr>
<tr>
<td><em>(SunCGI Extension)</em></td>
<td>integer name</td>
</tr>
<tr>
<td><strong>Append Text</strong></td>
<td>integer function cfaptext(flag, string)</td>
</tr>
<tr>
<td></td>
<td>integer flag</td>
</tr>
<tr>
<td></td>
<td>character*(<em>)(</em>) string</td>
</tr>
<tr>
<td><strong>Associate</strong></td>
<td>integer function cfassoc(trigger, devclass, devnum)</td>
</tr>
<tr>
<td></td>
<td>integer trigger</td>
</tr>
<tr>
<td></td>
<td>integer devclass</td>
</tr>
<tr>
<td></td>
<td>integer devnum</td>
</tr>
<tr>
<td><strong>Await Event</strong></td>
<td>integer function cfawaitev(timeout, valid, devclass, devnum, x, y, xlist, ylist, n, val, choice, string, segid, pickid, message_link, replost, time_stamp, qstat, overflow)</td>
</tr>
<tr>
<td></td>
<td>integer timeout</td>
</tr>
<tr>
<td></td>
<td>integer valid</td>
</tr>
<tr>
<td></td>
<td>integer devclass</td>
</tr>
<tr>
<td></td>
<td>integer devnum</td>
</tr>
<tr>
<td></td>
<td>integer x, y</td>
</tr>
<tr>
<td></td>
<td>integer xlist(*)</td>
</tr>
<tr>
<td></td>
<td>integer ylist(*)</td>
</tr>
<tr>
<td></td>
<td>integer n</td>
</tr>
<tr>
<td></td>
<td>real val</td>
</tr>
<tr>
<td></td>
<td>integer choice</td>
</tr>
<tr>
<td></td>
<td>character*(<em>)(</em>) string</td>
</tr>
<tr>
<td></td>
<td>integer segid</td>
</tr>
<tr>
<td></td>
<td>integer pickid</td>
</tr>
<tr>
<td></td>
<td>integer message_link</td>
</tr>
<tr>
<td></td>
<td>integer replost</td>
</tr>
<tr>
<td></td>
<td>integer time_stamp</td>
</tr>
<tr>
<td></td>
<td>integer qstat</td>
</tr>
<tr>
<td></td>
<td>integer overflow</td>
</tr>
</tbody>
</table>
### Table G-1  SunCGI Fortran Binding — Part I — Continued

<table>
<thead>
<tr>
<th>CGI Specification Name</th>
<th>Fortran Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BitBlt Pattern Array</strong></td>
<td>integer function cfbtblpatarr(pixpat, px, py, pixtarget, rx, ry, ox, oy, dx, dy, name)</td>
</tr>
<tr>
<td></td>
<td>integer pixpat</td>
</tr>
<tr>
<td></td>
<td>integer px, py</td>
</tr>
<tr>
<td></td>
<td>integer pixtarget</td>
</tr>
<tr>
<td></td>
<td>integer rx, ry</td>
</tr>
<tr>
<td></td>
<td>integer ox, oy</td>
</tr>
<tr>
<td></td>
<td>integer dx, dy</td>
</tr>
<tr>
<td></td>
<td>integer name</td>
</tr>
<tr>
<td><strong>BitBlt Patterned Source Array</strong></td>
<td>integer function cfbtblpatsouarr(pixpat, px, py, pixsource, sx, sy, pixtarget, rx, ry, ox, oy, dx, dy, name)</td>
</tr>
<tr>
<td></td>
<td>integer pixpat</td>
</tr>
<tr>
<td></td>
<td>integer px, py</td>
</tr>
<tr>
<td></td>
<td>integer pixsource</td>
</tr>
<tr>
<td></td>
<td>integer sx, sy</td>
</tr>
<tr>
<td></td>
<td>integer pixtarget</td>
</tr>
<tr>
<td></td>
<td>integer rx, ry</td>
</tr>
<tr>
<td></td>
<td>integer ox, oy</td>
</tr>
<tr>
<td></td>
<td>integer dx, dy</td>
</tr>
<tr>
<td></td>
<td>integer name</td>
</tr>
<tr>
<td><strong>BitBlt Source Array</strong></td>
<td>integer function cfbtblsouarr(bitsource, xo, yo, xe, ye, bittarget, xt, yt, name)</td>
</tr>
<tr>
<td></td>
<td>integer*4 bitsource, bittarget</td>
</tr>
<tr>
<td></td>
<td>integer xo, yo, xe, ye, xt, yt</td>
</tr>
<tr>
<td></td>
<td>integer name</td>
</tr>
<tr>
<td><strong>Cell Array</strong></td>
<td>integer function cfcellarr(px, qx, rx, py, qy, ry, dx, dy, colorind)</td>
</tr>
<tr>
<td></td>
<td>integer px, py</td>
</tr>
<tr>
<td></td>
<td>integer qx, qy</td>
</tr>
<tr>
<td></td>
<td>integer rx, ry</td>
</tr>
<tr>
<td></td>
<td>integer dx, dy</td>
</tr>
<tr>
<td></td>
<td>integer colorind(*)</td>
</tr>
<tr>
<td><strong>Character Expansion Factor</strong></td>
<td>integer function cfcharexpfac(efac)</td>
</tr>
<tr>
<td></td>
<td>real efac</td>
</tr>
<tr>
<td><strong>Character Height</strong></td>
<td>integer function cfcharheight(height)</td>
</tr>
<tr>
<td></td>
<td>integer height</td>
</tr>
<tr>
<td><strong>Character Orientation</strong></td>
<td>integer function cfcharorient(bx, by, dx, dy)</td>
</tr>
<tr>
<td></td>
<td>real bx, by, dx, dy</td>
</tr>
<tr>
<td><strong>Character Path</strong></td>
<td>integer function cfcharpath(path)</td>
</tr>
<tr>
<td></td>
<td>integer path</td>
</tr>
<tr>
<td><strong>Character Set Index</strong></td>
<td>integer function cfcharsetix(index)</td>
</tr>
<tr>
<td></td>
<td>integer index</td>
</tr>
</tbody>
</table>
### Table G-1  SunCGI Fortran Binding – Part I—Continued

<table>
<thead>
<tr>
<th>CGI Specification Name</th>
<th>Fortran Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character Spacing</td>
<td>integer function cfcharspacing(efac)</td>
</tr>
<tr>
<td></td>
<td>real efac</td>
</tr>
<tr>
<td>Circle</td>
<td>integer function cfcircle(x, y, rad)</td>
</tr>
<tr>
<td></td>
<td>integer x</td>
</tr>
<tr>
<td></td>
<td>integer y</td>
</tr>
<tr>
<td></td>
<td>integer rad</td>
</tr>
<tr>
<td>Circular Arc 3pt Close</td>
<td>integer function cfcircarcthreecl(clx, cly, c2x, c2y,</td>
</tr>
<tr>
<td></td>
<td>1 c3x, c3y, close)</td>
</tr>
<tr>
<td></td>
<td>integer clx, cly, c2x, c2y, c3x, c3y</td>
</tr>
<tr>
<td></td>
<td>integer close</td>
</tr>
<tr>
<td>Circular Arc 3pt</td>
<td>integer function cfcircarcthree(clx, cly, c2x, c2y,</td>
</tr>
<tr>
<td></td>
<td>1 c3x, c3y)</td>
</tr>
<tr>
<td></td>
<td>integer clx, cly, c2x, c2y, c3x, c3y</td>
</tr>
<tr>
<td>Circular Arc Center Close</td>
<td>integer function cfcircarccentcl(clx, cly, c2x, c2y,</td>
</tr>
<tr>
<td></td>
<td>1 c3x, c3y, rad, close)</td>
</tr>
<tr>
<td></td>
<td>integer clx, cly, c2x, c2y, c3x, c3y</td>
</tr>
<tr>
<td></td>
<td>integer close</td>
</tr>
<tr>
<td>Circular Arc Center</td>
<td>integer function cfcircarccent(clx, cly, c2x, c2y,</td>
</tr>
<tr>
<td></td>
<td>1 c3x, c3y)</td>
</tr>
<tr>
<td></td>
<td>integer clx, cly, c2x, c2y, c3x, c3y</td>
</tr>
<tr>
<td></td>
<td>integer rad</td>
</tr>
<tr>
<td>Clear Control</td>
<td>integer function cfclrcont(soft, hard, intern, extent)</td>
</tr>
<tr>
<td></td>
<td>integer soft, hard</td>
</tr>
<tr>
<td></td>
<td>integer intern</td>
</tr>
<tr>
<td></td>
<td>integer extent</td>
</tr>
<tr>
<td>Clear View Surface</td>
<td>integer function cfclrvws(name, defflag, color)</td>
</tr>
<tr>
<td></td>
<td>integer name</td>
</tr>
<tr>
<td></td>
<td>integer defflag</td>
</tr>
<tr>
<td></td>
<td>integer color</td>
</tr>
<tr>
<td>Clip Indicator</td>
<td>integer function cfclipind(flag)</td>
</tr>
<tr>
<td></td>
<td>integer flag</td>
</tr>
<tr>
<td>Clip Rectangle</td>
<td>integer function cfcliprect(xmin, xmax, ymin, ymax)</td>
</tr>
<tr>
<td></td>
<td>integer xmin, xmax, ymin, ymax</td>
</tr>
<tr>
<td>Close CGI (SunCGI Extension)</td>
<td>integer function cfclosecgi()</td>
</tr>
<tr>
<td>Close View Surface (SunCGI Extension)</td>
<td>integer function cfclosevws(name)</td>
</tr>
<tr>
<td></td>
<td>integer name</td>
</tr>
<tr>
<td>CGI Specification Name</td>
<td>Fortran Binding</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td><strong>Color Table</strong></td>
<td>integer function cfcotable(istart, ra, ga, ba, n)</td>
</tr>
<tr>
<td></td>
<td>integer istart</td>
</tr>
<tr>
<td></td>
<td>integer ra(<em>), ga(</em>), ba(*)</td>
</tr>
<tr>
<td></td>
<td>integer n</td>
</tr>
<tr>
<td><strong>Deactivate View Surface</strong></td>
<td>integer function cfdeactvws(name)</td>
</tr>
<tr>
<td>(SunCGI Extension)</td>
<td>integer name</td>
</tr>
<tr>
<td><strong>Define Bundle Index</strong></td>
<td>integer function cfdefbundix(index, linetype, linewidth, linecolor, marktype, marksize, markcolor, intstyle, batchindex, pattindex, fillcolor, perimtype, perimwidth, perimcolor, t3extfont, textprec, charexpand, charspace, textcolor)</td>
</tr>
<tr>
<td>(SunCGI Extension)</td>
<td>integer index</td>
</tr>
<tr>
<td></td>
<td>integer linetype</td>
</tr>
<tr>
<td></td>
<td>real linewidth</td>
</tr>
<tr>
<td></td>
<td>integer linecolor</td>
</tr>
<tr>
<td></td>
<td>integer marktype</td>
</tr>
<tr>
<td></td>
<td>real marksize</td>
</tr>
<tr>
<td></td>
<td>integer markcolor</td>
</tr>
<tr>
<td></td>
<td>integer intstyle</td>
</tr>
<tr>
<td></td>
<td>integer batchindex</td>
</tr>
<tr>
<td></td>
<td>integer pattindex</td>
</tr>
<tr>
<td></td>
<td>integer fillcolor</td>
</tr>
<tr>
<td></td>
<td>integer perimtype</td>
</tr>
<tr>
<td></td>
<td>real perimwidth</td>
</tr>
<tr>
<td></td>
<td>integer perimcolor</td>
</tr>
<tr>
<td></td>
<td>integer t3extfont</td>
</tr>
<tr>
<td></td>
<td>integer textprec</td>
</tr>
<tr>
<td></td>
<td>real charexpand</td>
</tr>
<tr>
<td></td>
<td>real charspace</td>
</tr>
<tr>
<td></td>
<td>integer textcolor</td>
</tr>
<tr>
<td><strong>Device Viewport</strong></td>
<td>integer function cfdevvpt(name, xbot, ybot, xtop, ytop)</td>
</tr>
<tr>
<td></td>
<td>integer name</td>
</tr>
<tr>
<td></td>
<td>integer xbot, ybot, xtop, ytop</td>
</tr>
<tr>
<td><strong>Disable Events</strong></td>
<td>integer function cfdaevents(devclass, devnum)</td>
</tr>
<tr>
<td></td>
<td>integer devclass</td>
</tr>
<tr>
<td></td>
<td>integer devnum</td>
</tr>
<tr>
<td><strong>Disjoint Polyline</strong></td>
<td>integer function cfdpolyline(xcoors, ycoors, n)</td>
</tr>
<tr>
<td></td>
<td>integer xcoors(*)</td>
</tr>
<tr>
<td></td>
<td>integer ycoors(*)</td>
</tr>
<tr>
<td></td>
<td>integer n</td>
</tr>
<tr>
<td>CGI Specification Name</td>
<td>Fortran Binding</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><em>Dissociate</em></td>
<td>integer function cfdissoc(trigger, devclass, devnum) integer trigger integer devclass integer devnum</td>
</tr>
<tr>
<td><em>Ellipse</em></td>
<td>integer function cfellipse(x, y, majx, miny) integer x, y integer majx, miny</td>
</tr>
<tr>
<td><em>Elliptical Arc Close</em></td>
<td>integer function cfelliparcc1(x, y, sx, sy, ex, ey, 1 majx, miny, close) integer x, y integer sx, sy integer ex, ey integer majx, miny integer close</td>
</tr>
<tr>
<td><em>Elliptical Arc</em></td>
<td>integer function cfelliparc(x, y, sx, sy, ex, ey, majx, 1 miny) integer x, y integer sx, sy integer ex, ey integer majx, miny</td>
</tr>
<tr>
<td><em>Enable Events</em></td>
<td>integer function cfenevents(devclass, devnum) integer devclass integer devnum</td>
</tr>
<tr>
<td><em>Fill Area Bundle Index</em></td>
<td>integer function cfflareabundix(index) integer index</td>
</tr>
<tr>
<td><em>Fill Color</em></td>
<td>integer function cfflcolor(color) integer color</td>
</tr>
<tr>
<td><em>(SunCGI Extension)</em></td>
<td>integer function cffixedfont(index) integer index</td>
</tr>
<tr>
<td><em>Flush Event Queue</em></td>
<td>integer function cfflusheventqu()</td>
</tr>
</tbody>
</table>
### Table G-2  SunCGI Fortran Binding – Part II—Continued

<table>
<thead>
<tr>
<th>CGI Specification Name</th>
<th>Fortran Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get Last Requested Input</td>
<td>integer function cfgetlastreqinp(devclass, devnum, valid, x, y, xlist, ylist, n, val, choice, string, segid, pickid)</td>
</tr>
<tr>
<td></td>
<td>integer devclass</td>
</tr>
<tr>
<td></td>
<td>integer devnum</td>
</tr>
<tr>
<td></td>
<td>integer valid</td>
</tr>
<tr>
<td></td>
<td>integer x, y</td>
</tr>
<tr>
<td></td>
<td>integer xlist(*)</td>
</tr>
<tr>
<td></td>
<td>integer ylist(*)</td>
</tr>
<tr>
<td></td>
<td>integer n</td>
</tr>
<tr>
<td></td>
<td>real val</td>
</tr>
<tr>
<td></td>
<td>integer choice</td>
</tr>
<tr>
<td></td>
<td>character*(*) string</td>
</tr>
<tr>
<td></td>
<td>integer segid</td>
</tr>
<tr>
<td></td>
<td>integer pickid</td>
</tr>
<tr>
<td>Hard Reset</td>
<td>integer function cfhardrst()</td>
</tr>
<tr>
<td>Hatch Index</td>
<td>integer function cfhatchix(index)</td>
</tr>
<tr>
<td>Initialize LID</td>
<td>integer function cfinitlid(devclass, devnum, x, y, xlist, ylist, n, val, choice, string, segid, pickid)</td>
</tr>
<tr>
<td></td>
<td>integer devclass</td>
</tr>
<tr>
<td></td>
<td>integer devnum</td>
</tr>
<tr>
<td></td>
<td>integer x, y</td>
</tr>
<tr>
<td></td>
<td>integer xlist(*)</td>
</tr>
<tr>
<td></td>
<td>integer ylist(*)</td>
</tr>
<tr>
<td></td>
<td>integer n</td>
</tr>
<tr>
<td></td>
<td>real val</td>
</tr>
<tr>
<td></td>
<td>integer choice</td>
</tr>
<tr>
<td></td>
<td>character*(*) string</td>
</tr>
<tr>
<td></td>
<td>integer segid</td>
</tr>
<tr>
<td></td>
<td>integer pickid</td>
</tr>
<tr>
<td>Initiate Request</td>
<td>integer function cfinitreq(devclass, devnum)</td>
</tr>
<tr>
<td></td>
<td>integer devclass</td>
</tr>
<tr>
<td></td>
<td>integer devnum</td>
</tr>
<tr>
<td>Inquire Aspect Source Flags</td>
<td>integer function cfqasfs(n, num, vals)</td>
</tr>
<tr>
<td></td>
<td>integer n</td>
</tr>
<tr>
<td></td>
<td>integer num(*)</td>
</tr>
<tr>
<td></td>
<td>integer vals(*)</td>
</tr>
</tbody>
</table>
Table G-2  SunCGI Fortran Binding – Part II—Continued

<table>
<thead>
<tr>
<th>CGI Specification Name</th>
<th>Fortran Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inquire BitBlt Alignments</td>
<td>integer function cfqbtbltaln(base, width, px, py, l maxpx, maxpy, name)</td>
</tr>
<tr>
<td></td>
<td>integer base</td>
</tr>
<tr>
<td></td>
<td>integer width</td>
</tr>
<tr>
<td></td>
<td>integer px</td>
</tr>
<tr>
<td></td>
<td>integer py</td>
</tr>
<tr>
<td></td>
<td>integer maxpx</td>
</tr>
<tr>
<td></td>
<td>integer maxpy</td>
</tr>
<tr>
<td></td>
<td>integer name</td>
</tr>
<tr>
<td>Inquire Cell Array</td>
<td>integer function cfqcellarr(name, px, qx, rx, py, qy, l ry, dx, dy, colorind)</td>
</tr>
<tr>
<td></td>
<td>integer name</td>
</tr>
<tr>
<td></td>
<td>integer px, py</td>
</tr>
<tr>
<td></td>
<td>integer qx, qy</td>
</tr>
<tr>
<td></td>
<td>integer rx, ry</td>
</tr>
<tr>
<td></td>
<td>integer dx, dy</td>
</tr>
<tr>
<td></td>
<td>integer colorind(*)</td>
</tr>
<tr>
<td>Inquire Device Bitmap</td>
<td>integer function cfqdevbtmp(name, map)</td>
</tr>
<tr>
<td></td>
<td>integer name</td>
</tr>
<tr>
<td></td>
<td>integer*4 map</td>
</tr>
<tr>
<td>Inquire Device Class</td>
<td>integer function cfqdevclass(output, input)</td>
</tr>
<tr>
<td></td>
<td>integer output, input</td>
</tr>
</tbody>
</table>

Table G-3  SunCGI Fortran Binding – Part III

<table>
<thead>
<tr>
<th>CGI Specification Name</th>
<th>Fortran Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inquire Device Identification</td>
<td>integer function cfqdevid(name, devid)</td>
</tr>
<tr>
<td></td>
<td>integer name</td>
</tr>
<tr>
<td></td>
<td>character*(*) devid</td>
</tr>
<tr>
<td>Inquire Drawing Mode</td>
<td>integer function cfqdrawmode(visibility, source, l destination, combination)</td>
</tr>
<tr>
<td></td>
<td>integer visibility</td>
</tr>
<tr>
<td></td>
<td>integer source</td>
</tr>
<tr>
<td></td>
<td>integer destination</td>
</tr>
<tr>
<td></td>
<td>integer combination</td>
</tr>
<tr>
<td>Inquire Event Queue State</td>
<td>integer function cfqevque(qstate, qoflow)</td>
</tr>
<tr>
<td></td>
<td>integer qstate</td>
</tr>
<tr>
<td></td>
<td>integer qoflow</td>
</tr>
</tbody>
</table>
### Table G-3  
**SunCGI Fortran Binding — Part III—Continued**

<table>
<thead>
<tr>
<th>CGI Specification Name</th>
<th>Fortran Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inquire Fill Area Attributes</strong></td>
<td>integer function cfqflareaatts(style, vis, color, hindex, 1 pindex, bindex, pstyle, pwidth, pcolor)</td>
</tr>
<tr>
<td></td>
<td>integer style, vis, color</td>
</tr>
<tr>
<td></td>
<td>integer hindex, pindex, bindex</td>
</tr>
<tr>
<td></td>
<td>integer pstyle</td>
</tr>
<tr>
<td></td>
<td>real pwidth</td>
</tr>
<tr>
<td></td>
<td>integer pcolor</td>
</tr>
</tbody>
</table>

<p>| <strong>Inquire Input Capabilities</strong> | integer function cfqinpcaps(valid, numloc, numval, numstrk, 1 numchoice, numstr, numtrig, evqueue, asynch, coordmap, 2 echo, tracking, prompt, acknowledgement, trigman) |
|                               | integer valid |
|                               | integer numloc |
|                               | integer numval |
|                               | integer numstrk |
|                               | integer numstr |
|                               | integer numchoice |
|                               | integer numstr |
|                               | integer numtrig |
|                               | integer evqueue |
|                               | integer asynch |
|                               | integer coordmap |
|                               | integer echo |
|                               | integer tracking |
|                               | integer prompt |
|                               | integer acknowledgement |
|                               | integer trigman |</p>
<table>
<thead>
<tr>
<th>CGI Specification Name</th>
<th>Fortran Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inquire LID State List</strong></td>
<td>integer function cfqlidstateis(devclass, devnum, valid, \</td>
</tr>
<tr>
<td></td>
<td>1 state, prompt, acknowledgement, x, y, xlist, ylist, n, \</td>
</tr>
<tr>
<td></td>
<td>2 val, choice, string, segid, pickid, n, triggers, \</td>
</tr>
<tr>
<td></td>
<td>3 echotype, echosta, echodat)</td>
</tr>
<tr>
<td></td>
<td>integer devclass</td>
</tr>
<tr>
<td></td>
<td>integer devnum</td>
</tr>
<tr>
<td></td>
<td>integer valid</td>
</tr>
<tr>
<td></td>
<td>integer state</td>
</tr>
<tr>
<td></td>
<td>integer prompt</td>
</tr>
<tr>
<td></td>
<td>integer acknowledgement</td>
</tr>
<tr>
<td></td>
<td>integer x</td>
</tr>
<tr>
<td></td>
<td>integer y</td>
</tr>
<tr>
<td></td>
<td>integer xlist(*)</td>
</tr>
<tr>
<td></td>
<td>integer ylist(*)</td>
</tr>
<tr>
<td></td>
<td>integer n</td>
</tr>
<tr>
<td></td>
<td>real val</td>
</tr>
<tr>
<td></td>
<td>integer choice</td>
</tr>
<tr>
<td></td>
<td>character*(*) string</td>
</tr>
<tr>
<td></td>
<td>integer segid</td>
</tr>
<tr>
<td></td>
<td>integer pickid</td>
</tr>
<tr>
<td></td>
<td>integer n</td>
</tr>
<tr>
<td></td>
<td>integer triggers(*)</td>
</tr>
<tr>
<td></td>
<td>integer echotype</td>
</tr>
<tr>
<td></td>
<td>integer echosta</td>
</tr>
<tr>
<td></td>
<td>integer echodat</td>
</tr>
</tbody>
</table>

| **Inquire LID State**          | integer function cfqlidstate(devclass, devnum, valid, \                     |
|                               | 1 state)                                                                    |
|                               | integer devclass                                                            |
|                               | integer devnum                                                              |
|                               | integer valid                                                               |
|                               | integer state                                                               |
### Table G-3  *SunCGI Fortran Binding – Part III—Continued*

<table>
<thead>
<tr>
<th>CGI Specification Name</th>
<th>Fortran Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inquire LID Capabilities</strong></td>
<td>integer function cfqlidcaps(devclass, devnum, valid, 1 sample, change, numassoc, trigassoc, prompt, 2 acknowledgement, echo, echotype, n, classdep, state)</td>
</tr>
<tr>
<td></td>
<td>integer devclass</td>
</tr>
<tr>
<td></td>
<td>integer devnum</td>
</tr>
<tr>
<td></td>
<td>integer valid</td>
</tr>
<tr>
<td></td>
<td>integer sample</td>
</tr>
<tr>
<td></td>
<td>integer change</td>
</tr>
<tr>
<td></td>
<td>integer numassoc</td>
</tr>
<tr>
<td></td>
<td>integer trigassoc(*)</td>
</tr>
<tr>
<td></td>
<td>integer prompt</td>
</tr>
<tr>
<td></td>
<td>integer acknowledgement</td>
</tr>
<tr>
<td></td>
<td>integer echo(*)</td>
</tr>
<tr>
<td></td>
<td>integer echotype(*)</td>
</tr>
<tr>
<td></td>
<td>integer n</td>
</tr>
<tr>
<td></td>
<td>character(<em>)(</em>) classdep</td>
</tr>
<tr>
<td></td>
<td>integer state(*)</td>
</tr>
<tr>
<td><strong>Inquire Line Attributes</strong></td>
<td>integer function cfqlnatts(style, width, color, index)</td>
</tr>
<tr>
<td></td>
<td>integer style</td>
</tr>
<tr>
<td></td>
<td>real width</td>
</tr>
<tr>
<td></td>
<td>integer color, index</td>
</tr>
<tr>
<td><strong>Inquire Marker Attributes</strong></td>
<td>integer function cfqmkatts(type, size, color, index)</td>
</tr>
<tr>
<td></td>
<td>integer type</td>
</tr>
<tr>
<td></td>
<td>real size</td>
</tr>
<tr>
<td></td>
<td>integer color, index</td>
</tr>
<tr>
<td><strong>Inquire Output Capabilities</strong></td>
<td>integer function cfqoutcap(first, last, list)</td>
</tr>
<tr>
<td></td>
<td>integer first, last</td>
</tr>
<tr>
<td></td>
<td>character<em>80 list(</em>)</td>
</tr>
<tr>
<td><strong>Inquire Output Function Set</strong></td>
<td>integer function cfqoutfunset(level, support)</td>
</tr>
<tr>
<td></td>
<td>integer level</td>
</tr>
<tr>
<td></td>
<td>integer support</td>
</tr>
<tr>
<td><strong>Inquire Pattern Attributes</strong></td>
<td>integer function cfqpatatts(cindex, row, column, colorlis, 1 x, y, dx, dy)</td>
</tr>
<tr>
<td></td>
<td>integer cindex</td>
</tr>
<tr>
<td></td>
<td>integer row</td>
</tr>
<tr>
<td></td>
<td>integer column</td>
</tr>
<tr>
<td></td>
<td>integer colorlis(*)</td>
</tr>
<tr>
<td></td>
<td>integer x</td>
</tr>
<tr>
<td></td>
<td>integer y</td>
</tr>
<tr>
<td></td>
<td>integer dx</td>
</tr>
<tr>
<td></td>
<td>integer dy</td>
</tr>
<tr>
<td>CGI Specification Name</td>
<td>Fortran Binding</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Inquire Physical Coordinate System</strong></td>
<td>integer function cfqphyscsys(name, xbase, ybase, xext, yext, xunits, yunits)</td>
</tr>
<tr>
<td></td>
<td>integer name</td>
</tr>
<tr>
<td></td>
<td>integer xbase, ybase</td>
</tr>
<tr>
<td></td>
<td>integer xext, yext</td>
</tr>
<tr>
<td></td>
<td>real xunits, yunits</td>
</tr>
<tr>
<td><strong>Inquire Pixel Array</strong></td>
<td>integer function cfqpixarr(px, py, m, n, colorind, name)</td>
</tr>
<tr>
<td></td>
<td>integer px, py</td>
</tr>
<tr>
<td></td>
<td>integer m, n</td>
</tr>
<tr>
<td></td>
<td>integer colorind(*)</td>
</tr>
<tr>
<td></td>
<td>integer name</td>
</tr>
<tr>
<td><strong>Inquire Text Attributes</strong></td>
<td>integer function cfqtextatts(fontset, index, cfont, prec, efac, space, color, hgt, bx, by, ux, uy, path, halign, valign, hfac, cfac)</td>
</tr>
<tr>
<td></td>
<td>integer fontset, index, cfont, prec</td>
</tr>
<tr>
<td></td>
<td>real efac, space</td>
</tr>
<tr>
<td></td>
<td>integer color, hgt</td>
</tr>
<tr>
<td></td>
<td>real bx, by, ux, uy</td>
</tr>
<tr>
<td></td>
<td>integer path, halign, valign</td>
</tr>
<tr>
<td></td>
<td>real hfac, cfac</td>
</tr>
<tr>
<td><strong>Inquire Text Extent</strong></td>
<td>integer function cfqtextext(string, nextchar, conx, cony, llpx, llpy, ulpx, ulpy, urpx, urpy)</td>
</tr>
<tr>
<td></td>
<td>character*(<em>)(</em>) string</td>
</tr>
<tr>
<td></td>
<td>character*(<em>)(</em>) nextchar</td>
</tr>
<tr>
<td></td>
<td>integer conx</td>
</tr>
<tr>
<td></td>
<td>integer cony</td>
</tr>
<tr>
<td></td>
<td>integer llpx</td>
</tr>
<tr>
<td></td>
<td>integer llpy</td>
</tr>
<tr>
<td></td>
<td>integer ulpx</td>
</tr>
<tr>
<td></td>
<td>integer ulpy</td>
</tr>
<tr>
<td></td>
<td>integer urpx</td>
</tr>
<tr>
<td></td>
<td>integer urpy</td>
</tr>
<tr>
<td>CGI Specification Name</td>
<td>Fortran Binding</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Inquire Trigger Capabilities</strong></td>
<td>integer function cfqtrigcaps(trigger, valid, change, n,</td>
</tr>
<tr>
<td></td>
<td>1 class, assoc, maxassoc, prompt, acknowledgement,</td>
</tr>
<tr>
<td></td>
<td>2 name, description)</td>
</tr>
<tr>
<td></td>
<td>integer trigger</td>
</tr>
<tr>
<td></td>
<td>integer valid</td>
</tr>
<tr>
<td></td>
<td>integer change</td>
</tr>
<tr>
<td></td>
<td>integer n</td>
</tr>
<tr>
<td></td>
<td>integer class(*)</td>
</tr>
<tr>
<td></td>
<td>integer assoc(*)</td>
</tr>
<tr>
<td></td>
<td>integer maxassoc</td>
</tr>
<tr>
<td></td>
<td>integer prompt</td>
</tr>
<tr>
<td></td>
<td>integer acknowledgement</td>
</tr>
<tr>
<td></td>
<td>character*(*) name</td>
</tr>
<tr>
<td></td>
<td>character*(*) description</td>
</tr>
<tr>
<td><strong>Inquire Trigger State</strong></td>
<td>integer function cfqtrigstate(trigger, valid, state, n,</td>
</tr>
<tr>
<td></td>
<td>1 class, assoc)</td>
</tr>
<tr>
<td></td>
<td>integer trigger</td>
</tr>
<tr>
<td></td>
<td>integer valid</td>
</tr>
<tr>
<td></td>
<td>integer state</td>
</tr>
<tr>
<td></td>
<td>integer n</td>
</tr>
<tr>
<td></td>
<td>integer class(*)</td>
</tr>
<tr>
<td></td>
<td>integer assoc(*)</td>
</tr>
<tr>
<td><strong>Inquire VDC Type</strong></td>
<td>integer function cfqvdctype(type)</td>
</tr>
<tr>
<td><strong>Interior Style</strong></td>
<td>integer function cfintstyle(istyle, perimvis)</td>
</tr>
<tr>
<td></td>
<td>integer istyle</td>
</tr>
<tr>
<td></td>
<td>integer perimvis</td>
</tr>
<tr>
<td><strong>Line Color</strong></td>
<td>integer function cfllncolor(index)</td>
</tr>
<tr>
<td></td>
<td>integer index</td>
</tr>
<tr>
<td><strong>Line Endstyle</strong></td>
<td>integer function cfllnendstyle(ttyp)</td>
</tr>
<tr>
<td>(SunCGI Extension)</td>
<td>integer ttyp</td>
</tr>
<tr>
<td><strong>Line Type</strong></td>
<td>integer function cfllntype(ttyp)</td>
</tr>
<tr>
<td></td>
<td>integer ttyp</td>
</tr>
<tr>
<td><strong>Line Width Specification</strong></td>
<td>integer function cfllnspecmode(mode)</td>
</tr>
<tr>
<td><strong>Mode</strong></td>
<td>integer mode</td>
</tr>
</tbody>
</table>

Version B of 17 February 1986
### Table G-4  SunCGI Fortran Binding – Part IV

<table>
<thead>
<tr>
<th>CGI Specification Name</th>
<th>Fortran Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Line Width</strong></td>
<td><code>integer function cflnwidth(index)</code></td>
</tr>
<tr>
<td></td>
<td><code>real index</code></td>
</tr>
<tr>
<td><strong>Marker Color</strong></td>
<td><code>integer function cfmkcolor(index)</code></td>
</tr>
<tr>
<td></td>
<td><code>integer index</code></td>
</tr>
<tr>
<td><strong>Marker Size</strong></td>
<td><code>integer function cfmspecmode(mode)</code></td>
</tr>
<tr>
<td><strong>Specification Mode</strong></td>
<td><code>integer mode</code></td>
</tr>
<tr>
<td><strong>Marker Size</strong></td>
<td><code>integer function cfmsize(index)</code></td>
</tr>
<tr>
<td></td>
<td><code>real index</code></td>
</tr>
<tr>
<td><strong>Marker Type</strong></td>
<td><code>integer function cfmtree(ttyp)</code></td>
</tr>
<tr>
<td></td>
<td><code>integer ttyp</code></td>
</tr>
<tr>
<td><strong>Open CGI (SunCGI Extension)</strong></td>
<td><code>integer function cfopencgi()</code></td>
</tr>
<tr>
<td><strong>Open View Surface</strong></td>
<td><code>integer function cfopenvws(name, screenname, windowname, windowfd, retained, dd, cmapsize, cmapname, flags, ptr)</code></td>
</tr>
<tr>
<td><em>(SunCGI Extension)</em></td>
<td><code>integer name</code></td>
</tr>
<tr>
<td></td>
<td><code>character*(*) screenname</code></td>
</tr>
<tr>
<td></td>
<td><code>character*(*) windowname</code></td>
</tr>
<tr>
<td></td>
<td><code>integer windowfd</code></td>
</tr>
<tr>
<td></td>
<td><code>integer retained</code></td>
</tr>
<tr>
<td></td>
<td><code>integer dd</code></td>
</tr>
<tr>
<td></td>
<td><code>integer cmapsize</code></td>
</tr>
<tr>
<td></td>
<td><code>character*(*) cmapname</code></td>
</tr>
<tr>
<td></td>
<td><code>integer flags</code></td>
</tr>
<tr>
<td></td>
<td><code>character*(*) ptr</code></td>
</tr>
<tr>
<td><strong>Partial Polygon</strong></td>
<td><code>integer function cfppolygon(xcoors, ycoors, n, flag)</code></td>
</tr>
<tr>
<td></td>
<td><code>integer xcoors(*)</code></td>
</tr>
<tr>
<td></td>
<td><code>integer ycoors(*)</code></td>
</tr>
<tr>
<td></td>
<td><code>integer n</code></td>
</tr>
<tr>
<td></td>
<td><code>integer flag</code></td>
</tr>
<tr>
<td><strong>Pattern Index</strong></td>
<td><code>integer function cfpatix(index)</code></td>
</tr>
<tr>
<td></td>
<td><code>integer index</code></td>
</tr>
<tr>
<td><strong>Pattern Reference Point</strong></td>
<td><code>integer function cfpatrefpt(x, y)</code></td>
</tr>
<tr>
<td></td>
<td><code>integer x, y</code></td>
</tr>
<tr>
<td><strong>Pattern Size</strong></td>
<td><code>integer function cfpatsize(dx, dy)</code></td>
</tr>
<tr>
<td></td>
<td><code>integer dx, dy</code></td>
</tr>
<tr>
<td><strong>Pattern Table</strong></td>
<td><code>integer function cfpatable(index, m, n, colorind)</code></td>
</tr>
<tr>
<td></td>
<td><code>integer index</code></td>
</tr>
<tr>
<td></td>
<td><code>integer m, n</code></td>
</tr>
<tr>
<td></td>
<td><code>integer colorind(*)</code></td>
</tr>
<tr>
<td>CGI Specification Name</td>
<td>Fortran Binding</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Pattern with Fill Color (SunCGI Extension)</td>
<td>integer function cfpatfillcolor(flag) integer flag</td>
</tr>
<tr>
<td>Perimeter Color</td>
<td>integer function cfperimcolor(index) integer index</td>
</tr>
<tr>
<td>Perimeter Type</td>
<td>integer function cfperimtype(ttyp) integer ttyp</td>
</tr>
<tr>
<td>Perimeter Width</td>
<td>integer function cfperimspecmode(mode) integer mode</td>
</tr>
<tr>
<td>Perimeter Width</td>
<td>integer function cfperimwidth(index) real index</td>
</tr>
<tr>
<td>Pixel Array</td>
<td>integer function cfpixarr(px, py, m, n, colorind) integer px, py integer m, n integer colorind(*)</td>
</tr>
<tr>
<td>Polygon</td>
<td>integer function cfpolygon(xcoors, ycoors, n) integer xcoors(<em>) integer ycoors(</em>) integer n</td>
</tr>
<tr>
<td>Polylne Bundle Index</td>
<td>integer function cfpolylnbundix(index) integer index</td>
</tr>
<tr>
<td>Polyline</td>
<td>integer function cfpolyline(xcoors, ycoors, n) integer xcoors(<em>) integer ycoors(</em>) integer n</td>
</tr>
<tr>
<td>Polymarker Bundle Index</td>
<td>integer function cfpolymkbundix(index) integer index</td>
</tr>
<tr>
<td>Polymarker</td>
<td>integer function cfpolymarker(xcoors, ycoors, n) integer xcoors(<em>) integer ycoors(</em>) integer n</td>
</tr>
<tr>
<td>Rectangle</td>
<td>integer function cfrectangle(xbot, ybot, xtop, ytop) integer xbot, ybot, xtop, ytop</td>
</tr>
<tr>
<td>Release Input Device</td>
<td>integer function cfrelidev(devclass, devnum) integer devclass integer devnum</td>
</tr>
</tbody>
</table>
### Table G-5  *SunCGI Fortran Binding – Part V*

<table>
<thead>
<tr>
<th>CGI Specification Name</th>
<th>Fortran Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Request Input</strong></td>
<td>integer function cfreqinp(devclass, devnum, timeout,</td>
</tr>
<tr>
<td></td>
<td>1   valid, x, y, xlist, ylist, n, val, choice, string,</td>
</tr>
<tr>
<td></td>
<td>2   segid, pickid, trigger)</td>
</tr>
<tr>
<td></td>
<td>integer devclass</td>
</tr>
<tr>
<td></td>
<td>integer devnum</td>
</tr>
<tr>
<td></td>
<td>integer timeout</td>
</tr>
<tr>
<td></td>
<td>integer valid</td>
</tr>
<tr>
<td></td>
<td>integer x, y</td>
</tr>
<tr>
<td></td>
<td>integer xlist(*)</td>
</tr>
<tr>
<td></td>
<td>integer ylist(*)</td>
</tr>
<tr>
<td></td>
<td>integer n</td>
</tr>
<tr>
<td></td>
<td>real val</td>
</tr>
<tr>
<td></td>
<td>integer choice</td>
</tr>
<tr>
<td></td>
<td>character(*) string</td>
</tr>
<tr>
<td></td>
<td>integer segid</td>
</tr>
<tr>
<td></td>
<td>integer pickid</td>
</tr>
<tr>
<td></td>
<td>integer trigger</td>
</tr>
<tr>
<td><strong>Reset to Defaults</strong></td>
<td>integer function cfrsttodefs()</td>
</tr>
<tr>
<td><strong>Sample Input</strong></td>
<td>integer function cfsampinp(devclass, devnum, valid, x, y,</td>
</tr>
<tr>
<td></td>
<td>1   xlist, ylist, n, val, choice, string, segid, pickid)</td>
</tr>
<tr>
<td></td>
<td>integer devclass</td>
</tr>
<tr>
<td></td>
<td>integer devnum</td>
</tr>
<tr>
<td></td>
<td>integer valid</td>
</tr>
<tr>
<td></td>
<td>integer x, y</td>
</tr>
<tr>
<td></td>
<td>integer xlist(*)</td>
</tr>
<tr>
<td></td>
<td>integer ylist(*)</td>
</tr>
<tr>
<td></td>
<td>integer n</td>
</tr>
<tr>
<td></td>
<td>real val</td>
</tr>
<tr>
<td></td>
<td>integer choice</td>
</tr>
<tr>
<td></td>
<td>character(*) string</td>
</tr>
<tr>
<td></td>
<td>integer segid</td>
</tr>
<tr>
<td></td>
<td>integer pickid</td>
</tr>
<tr>
<td><strong>Selective Flush of Event</strong></td>
<td>integer function cfsflusheventqu(devclass, devnum)</td>
</tr>
<tr>
<td><strong>Queue</strong></td>
<td>integer devclass</td>
</tr>
<tr>
<td></td>
<td>integer devnum</td>
</tr>
<tr>
<td><strong>Set Aspect Source Flags</strong></td>
<td>integer function cfsaspsouflags(fval, fnum, n)</td>
</tr>
<tr>
<td></td>
<td>integer fval(<em>), fnum(</em>), n</td>
</tr>
<tr>
<td><strong>Set Default Trigger</strong></td>
<td>integer function cfsdefatrigassoc(devclass, devnum)</td>
</tr>
<tr>
<td><strong>Associations</strong></td>
<td>integer devclass</td>
</tr>
<tr>
<td></td>
<td>integer devnum</td>
</tr>
<tr>
<td>CGI Specification Name</td>
<td>Fortran Binding</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Set Drawing Mode</td>
<td>integer function cfsdrawmode(visibility, source, destination, combination)</td>
</tr>
<tr>
<td>Set Error Warning Mask</td>
<td>integer function cfserrwarnmk(action)</td>
</tr>
<tr>
<td>Set Global Drawing Mode</td>
<td>integer function cfsgldrawmode(combination)</td>
</tr>
<tr>
<td></td>
<td>(SunCGI Extension)</td>
</tr>
<tr>
<td>Set Initial Value</td>
<td>integer function cfsinitval(devclass, devnum, x, y, xlist, ylist, n, val, choice, string, segid, pickid)</td>
</tr>
<tr>
<td></td>
<td>(SunCGI Extension)</td>
</tr>
<tr>
<td>Set Up SIGWINCH (SunCGI Extension)</td>
<td>integer function cfsupsig(name, sig_function)</td>
</tr>
<tr>
<td></td>
<td>external sig_function</td>
</tr>
<tr>
<td>Set VALUATOR Range</td>
<td>integer function cfsvalrange(devnum, mn, mx)</td>
</tr>
<tr>
<td>Text Alignment</td>
<td>integer function cftextalign(halign, valign, hcalind, vcalind)</td>
</tr>
<tr>
<td>Text Bundle Index</td>
<td>integer function cftextbundix(index)</td>
</tr>
<tr>
<td>Text Color</td>
<td>integer function cftextcolor(index)</td>
</tr>
<tr>
<td>Text Font Index</td>
<td>integer function cftextfontix(index)</td>
</tr>
</tbody>
</table>
Table G-5  *SunCGI Fortran Binding – Part V—Continued*

<table>
<thead>
<tr>
<th>CGI Specification Name</th>
<th>Fortran Binding</th>
</tr>
</thead>
</table>
| **Text Precision**     | integer function cftextprec(ttyp)  
|                        | integer ttyp     |
| **Text**               | integer function cftext(x, y, string)  
|                        | integer x       
|                        | integer y       
|                        | character*(*) string |
| **Track Off**          | integer function cftrackoff(devclass, devnum, tracktype, action)  
|                        | integer devclass |
|                        | integer devnum  |
|                        | integer tracktype |
|                        | integer action  |
| **Track On**           | integer function cftrackon(devclass, devnum, echotype, exlow, eylow, exup, eyup, x, y, xlist, ylist, n, val, choice, string, segid, pickid)  
|                        | integer devclass |
|                        | integer devnum  |
|                        | integer echotype |
|                        | integer exlow   |
|                        | integer eylow   |
|                        | integer exup    |
|                        | integer eyup    |
|                        | integer x, y    |
|                        | integer xlist(*) |
|                        | integer ylist(*) |
|                        | integer n      |
|                        | real val        |
|                        | integer choice  |
|                        | character*(*) string |
|                        | integer segid   |
|                        | integer pickid  |
| **VDC Extent**         | integer function cfvdcext(xbot, ybot, xtop, ytop)  
|                        | integer xbot, ybot, xtop, ytop |
| **VDM Text**           | integer function cfvdmtext(x, y, flag, string)  
|                        | integer x       |
|                        | integer y       |
|                        | integer flag    |
|                        | character*(*) string |
Short C Binding
Table H-1  Correspondence Between Long and Short C Names—Continued

<table>
<thead>
<tr>
<th>Long Name</th>
<th>Short Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>define_bundle_index</td>
<td>Cdefbundix</td>
</tr>
<tr>
<td>device_viewport</td>
<td>Cdevvpt</td>
</tr>
<tr>
<td>disable_events</td>
<td>Cdaevents</td>
</tr>
<tr>
<td>disjoint_polyline</td>
<td>Cd polyline</td>
</tr>
<tr>
<td>dissociate</td>
<td>Cdissoc</td>
</tr>
<tr>
<td>echo_off</td>
<td>Cechooff</td>
</tr>
<tr>
<td>echo_on</td>
<td>Cechoon</td>
</tr>
<tr>
<td>echo_update</td>
<td>Cechoupd</td>
</tr>
<tr>
<td>ellipse</td>
<td>Cellipse</td>
</tr>
<tr>
<td>elliptical_arc</td>
<td>Celliparc</td>
</tr>
<tr>
<td>elliptical_arc_close</td>
<td>Celliparcc1</td>
</tr>
<tr>
<td>enable_events</td>
<td>Cenevents</td>
</tr>
<tr>
<td>fill_area_bundle_index</td>
<td>Cflareabundix</td>
</tr>
<tr>
<td>fill_color</td>
<td>Cfcolor</td>
</tr>
<tr>
<td>fixed_font</td>
<td>Cfixedfont</td>
</tr>
<tr>
<td>flush_event_queue</td>
<td>Cflusheventqu</td>
</tr>
<tr>
<td>get_last_requested_input</td>
<td>Cgetlastreqinp</td>
</tr>
<tr>
<td>hard_reset</td>
<td>Chardrst</td>
</tr>
<tr>
<td>hatch_index</td>
<td>Chatchix</td>
</tr>
<tr>
<td>initialize_lid</td>
<td>Cinitlid</td>
</tr>
<tr>
<td>initiate_request</td>
<td>Cinitreq</td>
</tr>
<tr>
<td>inquire_aspect_source_flags</td>
<td>Cqasfs</td>
</tr>
<tr>
<td>inquire_bitblt_alignments</td>
<td>Cqbtlalign</td>
</tr>
<tr>
<td>inquire_cell_array</td>
<td>Cqcellarr</td>
</tr>
<tr>
<td>inquire_device_bitmap</td>
<td>Cqdevbtmp</td>
</tr>
<tr>
<td>inquire_device_class</td>
<td>Cqdevclass</td>
</tr>
<tr>
<td>inquire_device_identification</td>
<td>Cqdevid</td>
</tr>
<tr>
<td>inquire_drawing_mode</td>
<td>Cqdrawmode</td>
</tr>
<tr>
<td>inquire_event_queue_state</td>
<td>Cqevquestate</td>
</tr>
<tr>
<td>inquire_fill_area_attributes</td>
<td>Cqflareaatts</td>
</tr>
<tr>
<td>inquire_input_capabilities</td>
<td>Cqinpcaps</td>
</tr>
<tr>
<td>inquire_lid_capabilities</td>
<td>Cqlidcaps</td>
</tr>
<tr>
<td>inquire_lid_state</td>
<td>Cqlidstate</td>
</tr>
<tr>
<td>inquire_lid_state_list</td>
<td>Cqlidstatelis</td>
</tr>
<tr>
<td>inquire_line_attributes</td>
<td>Cqlnatts</td>
</tr>
<tr>
<td>inquire_marker_attributes</td>
<td>Cqmkatts</td>
</tr>
<tr>
<td>inquire_output_capabilities</td>
<td>Cqoutcap</td>
</tr>
<tr>
<td>inquire_output_function_set</td>
<td>Cqoutfunset</td>
</tr>
<tr>
<td>inquire_pattern_attributes</td>
<td>Cqpatatts</td>
</tr>
<tr>
<td>inquire_physical_coordinate_system</td>
<td>Cqphyscsys</td>
</tr>
<tr>
<td>inquire_pixel_array</td>
<td>Cqpixarr</td>
</tr>
<tr>
<td>inquire_text_attributes</td>
<td>Cqtextatts</td>
</tr>
<tr>
<td>inquire_text_extent</td>
<td>Cqtextext</td>
</tr>
<tr>
<td>inquire_trigger_capabilities</td>
<td>Cqtrigcaps</td>
</tr>
<tr>
<td>inquire_trigger_state</td>
<td>Cqtrigstate</td>
</tr>
<tr>
<td>inquire_vdc_type</td>
<td>Cqvdctype</td>
</tr>
</tbody>
</table>
### Table H-1 Correspondence Between Long and Short C Names—Continued

<table>
<thead>
<tr>
<th>Long Name</th>
<th>Short Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>interior_style</td>
<td>Cintstyle</td>
</tr>
<tr>
<td>line_color</td>
<td>Clncolor</td>
</tr>
<tr>
<td>line_endstyle</td>
<td>Clnendstyle</td>
</tr>
<tr>
<td>line_type</td>
<td>Clntype</td>
</tr>
<tr>
<td>line_width</td>
<td>Clnwidth</td>
</tr>
<tr>
<td>line_width_specification_mode</td>
<td>Clnwidthspecmode</td>
</tr>
<tr>
<td>marker_color</td>
<td>Cmkcolor</td>
</tr>
<tr>
<td>marker_size</td>
<td>Cmksizespecmode</td>
</tr>
<tr>
<td>marker_size_specification_mode</td>
<td>Cmksizespecmode</td>
</tr>
<tr>
<td>marker_type</td>
<td>Cmktype</td>
</tr>
<tr>
<td>opencgi</td>
<td>Copencgi</td>
</tr>
<tr>
<td>open_vws</td>
<td>Copenvws</td>
</tr>
<tr>
<td>partial_polygon</td>
<td>Ccppolygon</td>
</tr>
<tr>
<td>pattern_index</td>
<td>Cpatix</td>
</tr>
<tr>
<td>pattern_reference_point</td>
<td>Cpatrefpt</td>
</tr>
<tr>
<td>pattern_size</td>
<td>Cpatsize</td>
</tr>
<tr>
<td>pattern_table</td>
<td>Cpattable</td>
</tr>
<tr>
<td>pattern_with_fill_color</td>
<td>Cpatfillcolor</td>
</tr>
<tr>
<td>perimeter_color</td>
<td>Cperimcolor</td>
</tr>
<tr>
<td>perimeter_type</td>
<td>Cperimtype</td>
</tr>
<tr>
<td>perimeter_width</td>
<td>Cperimwidth</td>
</tr>
<tr>
<td>perimeter_width_specification_mode</td>
<td>Cperimwidthspecmode</td>
</tr>
<tr>
<td>pixel_array</td>
<td>Cpixarr</td>
</tr>
<tr>
<td>polygon</td>
<td>Cpolygon</td>
</tr>
<tr>
<td>polyline</td>
<td>Cpolyline</td>
</tr>
<tr>
<td>polyline_bundle_index</td>
<td>Cpolylnbundix</td>
</tr>
<tr>
<td>polymarker</td>
<td>Cpolymarker</td>
</tr>
<tr>
<td>polymarker_bundle_Index</td>
<td>Cpolymkbundix</td>
</tr>
<tr>
<td>rectangle</td>
<td>Crectangle</td>
</tr>
<tr>
<td>release_input_device</td>
<td>Creleidev</td>
</tr>
<tr>
<td>request_input</td>
<td>Creqinp</td>
</tr>
<tr>
<td>reset_to_defaults</td>
<td>Crsttodefs</td>
</tr>
<tr>
<td>sample_input</td>
<td>Csampling</td>
</tr>
<tr>
<td>selective_flush_of_event_queue</td>
<td>Cselectflushesventqu</td>
</tr>
<tr>
<td>set_aspect_source_flags</td>
<td>Csaaspectsourceflags</td>
</tr>
<tr>
<td>set_default_trigger_associations</td>
<td>Csdefatrigassoc</td>
</tr>
<tr>
<td>set_drawing_mode</td>
<td>Csdrawmode</td>
</tr>
<tr>
<td>set_error_warning_mask</td>
<td>Cserrwarning</td>
</tr>
<tr>
<td>set_global_drawing_mode</td>
<td>Csgldrawmode</td>
</tr>
<tr>
<td>set_initial_value</td>
<td>Csinitval</td>
</tr>
<tr>
<td>set_up_sigwinch</td>
<td>Csupsig</td>
</tr>
<tr>
<td>set_valuator_range</td>
<td>Csvalrange</td>
</tr>
<tr>
<td>text</td>
<td>Ctext</td>
</tr>
<tr>
<td>text_alignment</td>
<td>Ctextalign</td>
</tr>
<tr>
<td>text_bundle_index</td>
<td>Ctextbundle</td>
</tr>
<tr>
<td>text_color</td>
<td>Ctextcolor</td>
</tr>
</tbody>
</table>
Charlotte, 142

Character Expansion Factor, 70, 154
Character Height, 70, 154
Character Orientation, 71, 154
Character Path, 72, 154
Character Set Index, 69, 154
Character Spacing, 70, 154
character_expansion_factor, 70
character_height, 70
character_orientation, 71
character_path, 72
character_set_index, 69
character_spacing, 70
Circle, 38, 154
circle
area of a, 38
perimeter definition, 38
circle, 38
Circular Arc 3pt, 40, 154
Circular Arc 3pt Close, 41, 154
Circular Arc Center, 38, 154
Circular Arc Center Close, 39, 154
circular arcs
center, 39
close, 39
direction of drawing, 39
three-point, 40
circular arc_3pt, 40
circular arc_3pt_close, 41
circular arc_center, 38
circular arc_center_close, 39
Clear Control, 21, 154
Clear View Surface, 21, 154
clear_control, 21
clear_view_surface, 21
Clip Indicator, 19, 154
Clip Rectangle, 20, 154
clip_indicator, 19
clip_rectangle, 20
clipping, 17, 19
Close a CGI Pixwin, 144
Close CGI (SunCGI Extension), 16, 154
Close Pixwin CGI, 144
Close View Surface (SunCGI Extension), 16, 154
close cgi, 16
close cgi_pw, 144
close_vws, 16
color attributes, 74 thru 75
color_table, 74
color_table, 59, 74, 157
color_table, 74
conical output primitives, 33, 33 thru 42
color_table, 74
coordinate definition errors, 124 thru 125
current position, 103
data type definitions, 111 thru 120
dataspace, 16
Deactivate View Surface (SunCGI Extension), 16, 157
deactivate_vws, 16
Define Bundle Index (SunCGI Extension), 56, 157
define_bundle_index, 56
device coordinates (see screen space), 17
Device Viewport, 19, 157
device_viewport, 19
Disable Events, 98, 157
disable_events, 98
Disjoint Polyline, 34, 157
disjoint_polyline, 34
Dissociate, 86, 157
documentation conventions, xv
drawing mode, 6, 42
drawing modes, 48 thru 50
Ellipse, 41, 157
Elliptical Arc, 41, 157
Elliptical Arc Close, 42, 157
efficient arcs, 41
drawing of, 42
efficient_arc, 41
efficient_arc_close, 42
Enable Events, 95, 157
enable_events, 95
errors
color, 124
color_table, 74 thru 75
color_table, 59, 74, 157
color_table, 74
cOLOR_ATTRIBUTES, 74 thru 75
cOLOR_TABLE, 74
cOLOR_TABLE, 59, 74, 157
cOLOR_TABLE, 74
cOORDINATE_DEFINITION_ERRORS, 124 thru 125
cURRENT_POSITION, 103
data type definitions, 111 thru 120
dataspace, 16
deactivate_vws, 16
define_bundle_index, 56
device coordinates (see screen space), 17
device_viewport, 19
disable_events, 98
disjoint_polyline, 34
Dissociate, 86
documentation conventions, xv
drawing mode, 6, 42
drawing modes, 48 thru 50
Ellipse, 41
Elliptical Arc, 41
Elliptical Arc Close, 42
Enable Events, 95
Enable Events, 95
errors
color, 124
color_table, 74 thru 75
color_table, 59, 74, 157
color_table, 74
cOLOR_ATTRIBUTES, 74 thru 75
cOLOR_TABLE, 74
cOLOR_TABLE, 59, 74, 157
cOLOR_TABLE, 74
cOORDINATE_DEFINITION_ERRORS, 124 thru 125
cURRENT_POSITION, 103
data type definitions, 111 thru 120
dataspace, 16
deactivate_vws, 16
define_bundle_index, 56
device coordinates (see screen space), 17
device_viewport, 19
disable_events, 98
disjoint_polyline, 34
Dissociate, 86
documentation conventions, xv
drawing mode, 6, 42
drawing modes, 48 thru 50
Ellipse, 41
Elliptical Arc, 41
Elliptical Arc Close, 42
Enable Events, 95
Enable Events, 95
errors
color, 124
color_table, 74 thru 75
color_table, 59, 74, 157
color_table, 74
cOLOR_ATTRIBUTES, 74 thru 75
cOLOR_TABLE, 74
cOLOR_TABLE, 59, 74, 157
cOLOR_TABLE, 74
cOORDINATE_DEFINITION_ERRORS, 124 thru 125
cURRENT_POSITION, 103
data type definitions, 111 thru 120
dataspace, 16
deactivate_vws, 16
define_bundle_index, 56
device coordinates (see screen space), 17
device_viewport, 19
disable_events, 98
disjoint_polyline, 34
Dissociate, 86
documentation conventions, xv
drawing mode, 6, 42
drawing modes, 48 thru 50
polygon
   with undrawn edge(s), 36
polygonal primitives, 33, 33 thru 38
Polylne, 34, 166
Polyline Bundle Index, 57, 166
polyline_bundle_index, 57
Polymarker, 35, 166
polymarker attributes, 60 thru 61
   marker_color, 61
   marker_size, 61
   marker_size_specification_mode, 60
   marker_type, 60
polymarker_bundle_index, 60
Polymarker Bundle Index, 60, 166
polymarker_bundle_index, 60

R
raster primitives, 33, 42 thru 48
Rectangle, 38, 166
Release Input Device, 85, 166
release_input_device, 85
Request Input, 91, 168
request_register, 92, 97
request_input, 91
Reset to Defaults, 20, 168
reset_to_defaults, 20
retained windows, 14

S
Sample Input, 97, 168
sample_input, 97
screen space, 5, 17
   definition, 19
Selective Flush of Event Queue, 96, 168
selective_flush_of_event_queue, 96
Set Aspect Source Flags, 56, 168
Set Default Trigger Associations, 86, 168
Set Drawing Mode, 49, 168
Set Error Warning Mask, 22, 168
Set Global Drawing Mode (SunCGI Extension), 50, 168
Set Initial Value, 87, 168
Set Up SIGWINCH (SunCGI Extension), 23, 168
Set VALUATOR Range, 87, 168
set_aspect_source_flags, 56
set_default_trigger_associations, 86
set_drawing_mode, 49
set_error_warning_mask, 22
set_global_drawing_mode, 50
set_initial_value, 87
set_up_sigwinch, 23
set_valuator_range, 87
Short C Binding, 4, 173
SIGWINCH, 6, 22
solid object attributes, 61 thru 68
   fill_area_bundle_index, 62
   fill_color, 63
   interior_style, 62
specified device, 28
state errors, 123 thru 124
status inquiries, 98 thru 100
Sun Workstation, 25
SunCGI, 3
   with SunCGI, 22 thru 24
SunView
   set_up_sigwinch, 23
   using SunCGI with, 22, 24
synchronous input functions, 90 thru 92

T
Text, 42, 168
Text Alignment, 72, 168
text attributes, 68 thru 74
   character_expansion_factor, 70
   character_height, 70
   character_orientation, 71
   character_path, 72
   character_set_index, 69
   character_spacing, 70
   fixed_font, 71
   text_alignment, 72
   text_bundle_index, 68
   text_color, 71
   text_font_index, 69
   text_precision, 68
Text Bundle Index, 68, 168
Text Color, 71, 168
Text Font Index, 69, 168
Text Precision, 68, 168
text precision
   detailed definition, 68
text, 42
   appended, 43
text_alignment, 72
text, 42
text_bundle_index, 68
text_color, 71
text_font_index, 69
text_precision, 68
textured line, 58
timeout, 83
track, 88
Track Off, 89, 168
Track On, 88, 168
track_off, 89
track_on, 88
tracking, 88 thru 90
trigger, 6, 27, 86
Trigger
   Capabilities, 29
trigger
   interaction with STROKE device, 86
status, 98
type definitions, 111 thru 120

U
unsupported CGI functions, 107 thru 108
using SunCGI, 3
Index Continued

V
- Extent, 17, 168
- C space, 5, 17
- _extent, 17
- I, xv
- W Text, 43, 168
- n_text, 43
- w_surface, 11
- clear control, 21
- clearing, 20
- default states, 15
- w_surface control, 17 thru 22
- clear_control, 21
- clear_view_surface, 21
- clip_indicator, 19
- clip_rectangle, 20
- device_viewport, 19
- hard_reset, 20
- reset_to_defaults, 20
- set_error_warning_mask, 22
- vdc_extent, 17
- w_surfaces, 15
- active, 5
- initializing, 13
- multiple, 5, 13
- valid errors
  - possible causes, 131 thru 134

W
- windows
  - nonretained, 14
  - retained, 14
- rld coordinates (see vdc space), 17
## Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5/15/85</td>
<td>2.0 Production Release.</td>
</tr>
<tr>
<td>B</td>
<td>2/17/86</td>
<td>3.0 Production Release.</td>
</tr>
</tbody>
</table>
Notes