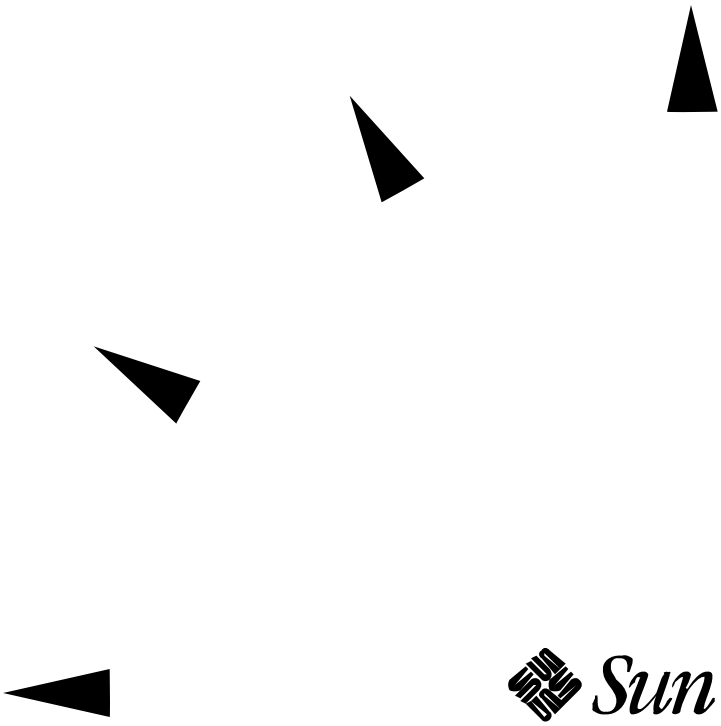
Platform Notes: SMCC™ Frame Buffers



Sun Microsystems Computer Company
A Sun Microsystems, Inc. Business
2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

Part No: 802-5326-10
Revision A, May 1996

Please
Recycle

Contents

Tables

Preface

Platform Notes: SMCC Frame Buffers contains important information on configuration options for the various frame buffers. The frame buffer devices described in this document are:

- TurboGXplus™ Frame Buffer
- S24™ Frame Buffer
- ZX and TurboZX™ Graphics Accelerator
- SX Frame Buffer
- Creator and Creator 3D Graphics Accelerators

Who Should Use This Book

This book is intended for use by anyone who needs to configure one of the described frame buffers or graphics accelerators to meet specific video or graphics requirements.

How This Book Is Organized

Section 1, "TurboGXplus Frame Buffer," describes how to configure a system using a TurboGXplus card to suit different screen resolutions and to support multiple monitors.

Section 2, "S24 Frame Buffer," describes the S24 Frame Buffer hardware options.

Section 3, “ZX and TurboZX Graphics Accelerator,” describes how to change ZX and TurboZX Graphics Accelerator screen resolution to work properly with different monitors.

Section 4, “SX Frame Buffer,” describes how to change the display resolution on the SX Frame Buffer (cgfourteen).

Section 5, “Creator Graphics Accelerator,” describes a UNIX® shell-based utility that can be used to select configuration options for the Creator or Creator 3D Graphics Accelerator, the attached monitor, and the associated X11 screen.

Chapter 6, “Creator Window System,” describes the behavior of the Solaris X11 window system on the Creator and Creator 3D Graphics Accelerators.

Chapter 7, “XIL Acceleration on Ultra/Creator,” describes the XIL functions that are specific to the Creator and Creator 3D Graphics Accelerators.

Section 8, “Multiple Monitors on a System,” describes how to use multiple monitors on a SPARCstation system.

What Typographic Changes Mean

The following table describes the typographic changes used in this book.

Table P-1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% You have mail.`
`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name% su` `Password:`
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type `rm` *filename.*
AaBbCc123	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in *User’s Guide.* These are called *class* options. You *must* be root to do this.

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

Table P-2 Shell Prompts

Shell	Prompt
C shell prompt	`machine_name%`
C shell superuser prompt	`machine_name#`
Bourne shell and Korn shell prompt	`$`
Bourne shell and Korn shell superuser prompt	`#`

Ordering Sun Documents

The SunDocs Order Desk is a distribution center for Sun Microsystems technical documents. You can use major credit cards and company purchase orders. You can order documents in the following ways:

Country	Telephone	Fax
United States	1-800-873-7869	1-800-944-0661
United Kingdom	0-800-89-88-88	0-800-89-88-87
France	05-90-61-57	05-90-61-58
Belgium	02-720-09-09	02-725-88-50
Luxembourg	32-2-720-09-09	32-2-725-88-50
Germany	01-30-81-61-91	01-30-81-61-92
The Netherlands	06-022-34-45	06-022-34-46
Sweden	020-79-57-26	020-79-57-27
Switzerland	155-19-26	155-19-27
Japan	0120-33-9096	0120-33-9097
World Wide Web: `http://www.sun.com/sunexpress/`		

Sun Welcomes Your Comments

Please use the *Reader Comment Card* that accompanies this document. We are interested in improving our documentation and welcome your comments and suggestions.

If this card is not attached, you can also email or fax your comments to us. Please include the part number of your document in the subject line of your email or fax message.

- Email: `smcc-docs@sun.com`
- Fax: SMCC Document Feedback
  1-415-786-6443

TurboGXplus Frame Buffer 1

This chapter describes how you can configure your system using a TurboGXplus card to suit your specific video / graphics requirements. The information describes how to set up your TurboGXplus to support different screen resolutions and how to set up the system to support multiple monitors.

TurboGXplus Supported Monitors

Table 1-1 shows the list of monitors supported by the TurboGXplus card.

1

Table 1-1 Monitors Supported by TurboGXplus

Model	Sun Part Number	Type/Size/FCC	Monitor ID Sense Code	Standard Resolution and Refresh Rate
X248A	365-1068-01	Color 21"	2	**1280 × 1024 at 76 Hz**
GDM-20D10	365-1167-01	Color 20"	4	**1152 × 900 at 76 Hz** 1152 × 900 at 66 Hz 1280 × 1024 at 67 Hz 1280 × 1024 at 76 Hz
GDM-1955A15	365-1081-01	Color 19"	3	**1152 × 900 at 66 Hz**
GDM-1962	365-1095-01	Color 19"	4	**1152 × 900 at 76 Hz** 1152 × 900 at 66 Hz 1280 × 1024 at 67 Hz
GDM-1962B	365-1160-01	Color 19"	4	**1152 × 900 at 76 Hz** 1152 × 900 at 66 Hz 1280 × 1024 at 67 Hz
GDM-1604A15	365-1079-01	Color 16"	3	**1152 × 900 at 66 Hz**
GDM-1662B	365-11593-01	Color 16"	6	**1152 × 900 at 76 Hz** 1152 × 900 at 66 Hz
CPD-1790	365-1151-01	Color 16"	3	**1152 × 900 at 66 Hz** 1024 × 768 at 77 Hz
X449	365-1286-01	Color 15"	0	**1024 × 768 at 77 Hz**
GDM-20S5	365-1168-01	Greyscale 20"	2 or 4*	**1280 × 1024 at 76 Hz** or **1152 × 900 at 76 Hz** 1280 × 1024 at 67 Hz
17SMM4 A	365-1100-01	Grayscale 17"	6	**1152 × 900 at 76 Hz**
M20P110	365-1099-01	Grayscale 19"	4	**1152 × 900 at 76 Hz**
Non-Sun	--	Unknown	7	**1152 × 900 at 66 Hz**

Resolutions in **bold type** are the default resolution at power-on initialization.
* Monitor ID sense code is user-selectable by switch on rear.

Note – The monitors listed in Table 1-1 are subject to change as Sun Microsystems® announces new monitors. Contact your local Sun representative for a listing of supported monitors.

Default Screen Resolutions

Table 1-2 lists the default screen resolutions by monitor ID sense code.

Table 1-2 TurboGXplus Monitor Sense Codes

Code	Screen Resolution
7	1152 × 900 at 66 Hz
6	1152 × 900 at 76 Hz
5	1024 ×x 768 at 60 Hz
4	1152 × 900 at 76 Hz
3	1152 × 900 at 66 Hz
2	1280 × 1024 at 76 Hz
1	1600 × 1280 at 76 Hz
0	1024 × 768 at 77 Hz

Programming the Screen Resolution

Nvramrc is a nonvolatile PROM script memory. When the PROM reaches the device probing stage, it checks the variable use-nvramrc?, if it is true, executes the Forth code that resides in `nvramrc`, otherwise it calls `probe-sbus`, `install-console` and `banner`.

The following code places resolution initialization between the probe-sbus stage and the install-console and banner stages.

```
#!/bin/sh
eeprom fcode-debug\?=true
eeprom use-nvramrc\?=true
eeprom nvramrc=’probe-sbus
: vsetup “ 117000000,71691,67,16,112,224,1280,2,8,33,1024,COLOR,0OFFSET” ;
vsetup 4
“ /sbus/cgsix@1” “ override” execute-device-method drop
install-console
banner
‘
```

First `probe-sbus` is called to probe all devices, so that the device tree is created, and the devices are initialized.

The next line defines a Forth word called `vsetup` which contains the monitor video setup values. The following string of values (defined in Table 1-3) are the specifications for a video setup:

" 117000000,71691,67,16,112,224,1280,2,8,33,1024,COLOR,0OFFSET"

Table 1-3 Video Setup Specifications

Value	Description
117000000	Pixel frequency or dot clock in Hz
71691	Horizontal frequency in Hz
67	Vertical frequency in Hz
16	Horizontal Front Porch (in pixels)
112	Horizontal Sync Width (in pixels)
224	Horizontal Back Porch (in Pixels)
1280	Horizontal Displayed pixels (in pixels)
2	Vertical Front Porch (in lines)
8	Vertical Sync Width (in lines)
33	Vertical Back Porch (in lines)
1024	Vertical Displayed lines (in lines)
COLOR	Color monitor flag
0OFFSET	No sync pedestal flag

The line, `vsetup 4`, basically pushes the video string on the stack, the number 4 defines the sense code of the monitor we want to change the resolution on. See Table 1-4 on page 6 for supported monitor codes.

The next line, pushes the string `/sbus/cgsix@1` onto the Forth stack, the path for the device where the resolution is to be changed.

The following example changes the cgsix frame buffer on SBus slot 1.

```
ok nvedit
0: probe-sbus
1: : vsetup “ 117000000,71691,67,16,112,224,1280,2,8,33,1024,COLOR,0OFFSET” ;
2: vsetup 4
3: “ /sbus/cgsix@1” “ override” execute-device-method drop
4: install-console
5: banner
6: ^C
ok nvstore
ok setenv use-nvramrc? true
ok setenv fcode-debug? true
```

The string “override” is the actual entry point in the cgsix fcode PROM which reconfigures the resolution from the data present on the forth stack. execute-device-method actually calls override and returns a pass/fail flag which is ignored by the drop command that follows.

The remaining two lines install-console and banner, installs a terminal driver on the display device, then prints the banner at reset time or reboot time.

Configuring Monitors Using a UNIX Script

The following is an example of a UNIX script used to configure the TurboGXplus for a resolution of 1280 × 1024 at 67 Hz:

```
#!/bin/sh
eeprom fcode-debug\?=true
eeprom use-nvramrc\?=true
eeprom nvramrc=’probe-sbus
: vsetup “ 117000000,71691,67,16,112,224,1280,2,8,33,1024,COLOR,0OFFSET” ;
vsetup 4
“ /sbus/cgsix@1” “ override” execute-device-method drop
install-console
banner
‘
```

Configuring Monitors Using the PROM Method

The following example uses the PROM method to configure the TurboGXplus for a resolution of 1280 × 1024 at 67 Hz:

```
ok nvedit
0: probe-sbus
1: : vsetup “ 117000000,71691,67,16,112,224,1280,2,8,33,1024,COLOR,0OFFSET” ;
2: vsetup 4
3: “ /sbus/cgsix@1” “ override” execute-device-method drop
4: install-console
5: banner
6: ^C
ok nvstore
ok setenv use-nvramrc? true
ok setenv fcode-debug? true
```

Table 1-4 contains codes for TurboGXplus supported resolutions:

Table 1-4 TurboGXplus Resolution Codes

Resolution	Code
1024 × 768 at 60 Hz	“ 64125000,48286,60,16,128,160,1024,2,6,29,768,COLOR”
1024 × 768 at 70 Hz	“ 74250000,56593,70,16,136,136,1024,2,6,32,768,COLOR”
1024 × 768 at 77 Hz	“ 84375000,62040,77,32,128,176,1024,2,4,31,768,COLOR”
1152 × 900 at 66 Hz	“ 94500000,61845,66,40,128,208,1152,2,4,31,900,COLOR”
1152 × 900 at 76 Hz	“ 108000000,71808,76,32,128,192,1152,2,4,31,900,COLOR,0OFFSET”
1280 × 1024 at 67 Hz	“ 117000000,71691,67,16,112,224,1280,2,8,33,1024,COLOR,0OFFSET”
1280 × 1024 at 76 Hz	“ 135000000,81128,76,32,64,288,1280,2,8,32,1024,COLOR,0OFFSET”
1600 × 1280 at 76 Hz	“ 216000000,101890,76,24,216,280,1600,2,8,50,1280,COLOR,0OFFSET”

Setting up a Single Monitor Using the PROM Method

The following is an example of how to set up a TurboGXplus card in slot 2 to 1024 × 768 at 60 Hz using a 16-inch monitor:

```
ok nvedit
   0: probe-sbus
   1: : vsetup “ 64125000,48286,60,16,128,160,1024,2,6,29,768,COLOR” ;
   2: vsetup 6
   3: “ /sbus/cgsix@2” “ override” execute-device-method drop
   4: install-console
   5: banner
   6: ^C
ok nvstore
ok setenv use-nvramrc? true
ok setenv fcode-debug? true
```

Setting up a Single Monitor Using a UNIX Script

The following example is a UNIX script that sets a 1024 × 768 at 60 Hz for the TurboGXplus card in slot 2.

```
#!/bin/sh
eeprom fcode-debug\?=true
eeprom nvramrc=’probe-sbus
: vsetup “ 64125000,48286,60,16,128,160,1024,2,6,29,768,COLOR” ;
vsetup 6
“/sbus/cgsix@2” “ override” execute-device-method drop
install-console
banner
‘
eeprom use-nvramrc\?=true
```

Setting up Multiple Monitors using a UNIX Script

The following screen shows a UNIX script, which sets up the TurboGXplus cards in slot 1 to 1152 × 900 at 76 Hz, and another TurboGXplus card in slot 3 to 1280 × 1024 at 67 Hz using two 19-inch monitors:

```
#!/bin/sh
eeprom fcode-debug\?=true
eeprom nvramrc='probe-sbus
: vsetup1 " 108000000,71808,76,32,128,192,1152,2,4,31,900,COLOR,0OFFSET" ;
vsetup1 4
" /sbus/cgsix@1" " override" execute-device-method drop
: vsetup2 " 117000000,71691,67,16,112,224,1280,2,8,33,1024,COLOR,0OFFSET" ;
vsetup2 4
" /sbus/cgsix@3" " override" execute-device-method drop
install-console
banner
'
eeprom use-nvramrc\?=true
```

For more information on running multiple monitors, see Chapter 8, "Multiple Monitors on a System."

S24 Frame Buffer 2

This chapter describes the S24™ Frame Buffer hardware options.

S24 Application Compatibility

The default visual of OpenWindows™ running on a system with an S24 Frame Buffer is 24-bit TrueColor Visual. Some older 8-bit windows applications that have been written without consideration for portability may not work with the default 24-bit visual of the S24 Frame Buffer.

There is a simple workaround for these applications when starting OpenWindows. As root, enter:

```
# openwin -dev /dev/fbs/tcx0 defdepth 8
```

This workaround sets the default visual to 8 bits. All applications can now be executed in 8-bit mode unless they explicitly request 24 bits or the best available visual.

The following 24-bit visuals are exported in the Solaris™ 2.4 software environment:

- 24-bit TrueColor visual
- 24-bit TrueColor Linear visual
- 24-bit DirectColor visual

The nonlinear visual is displayed before the linear visual on the screen visual list. Nonlinear visual is the default 24-bit TrueColor visual. If you prefer gamma-corrected 24-bit TrueColor to become your default value, you can modify the order of the visual list by using the `tcxconfig` command, which is provided in the `SUNWtcxow` package. Refer to the `tcxconfig` man page for more information.

OpenWindows should not be running when you run the `tcxconfig` script. Start OpenWindows after `tcxconfig` has set the linearity you desire.

Entering `tcxconfig` without options displays the current default setting.

```
# /usr/sbin/tcxconfig
linear
```

`linear` means that the linear visual is the default 24-bit TrueColor visual. This means that color is gamma corrected.

`nonlinear` means that the nonlinear visual is the default 24-bit TrueColor visual.

To change the setting, enter the `tcxconfig` command with one of the above options. For example:

```
# /usr/sbin/tcxconfig nonlinear
```

S24 Frame Buffer Screen Resolutions

The S24 frame buffer in the SPARCstation™ 5 supports three different screen resolutions. You may select a screen resolution other than the default value. This is performed at the `ok` prompt.

Default Screen Resolutions

Table 2-1 lists the default screen resolutions by monitor ID sense code.

Table 2-1 S24 Frame Buffer Monitor Sense Codes

Code	Screen Resolution
7	1152 × 900 at 66 Hz
6	1152 × 900 at 76 Hz
5	1024 × 768 at 70 Hz
4	1152 × 900 at 76 Hz
3	1152 × 900 at 66 Hz
2	1152 × 900 at 76 Hz
1	1152 × 900 at 66 Hz
0	1152 × 900 at 66 Hz

Changing the Screen Resolution

There are two methods of changing the screen resolution, depending on the frame buffer you select to be console. One method sets the default console device to the desired resolution. The other method forces the console to be the S24 monitor at a specified resolution.

To change the screen resolution:

1. **With the SPARCstation 5 powered on, bring the system down to the** `ok` **prompt.**

2. **At the** ok **prompt for the console-device selected by the CPU boot PROM, enter:**

```
ok setenv output-device screen:resolution
```

where *resolution* is one of the following:

resolution	Screen Resolution
r1152x900x66	1152 × 900 at 66 Hz
r1152x900x76	1152 × 900 at 76 Hz
r1024x768x70	1024 × 768 at 70 Hz

3. **To force the console to be the S24 monitor at the specified resolution, enter:**

```
ok setenv output-device /iommu/sbus/tcx:resolution
```

4. **To reset the system so that the NVRAM entry overwrites the monitor ID resolution selection, enter:**

```
ok reset
```

5. **To boot your SPARCstation 5 system, enter:**

```
ok boot -r
```

The monitor resolution is now reset to the specified resolution.

ZX and TurboZX Graphics Accelerator

3

This chapter describes how to change the ZX and TurboZX Graphics Accelerator screen resolution to work properly with different monitors. Since the following discussions are identical for the ZX and the TurboZX, the product name in this chapter is shortened to ZX to simplify the discussion.

You can change the ZX screen resolution through the use of the `leoconfig` program. See the `leoconfig`(8) man page for more information on this program than is described here.

There are two elements to `leoconfig`: the `leoconfig` program and the `leoconfig` script. The `leoconfig` program initializes the ZX Graphics Accelerator and downloads microcode from the host CPU. The `leoconfig` program is normally run as a part of the `/etc/init.d/leoconfig` script to download the ZX microcode file and to complete ZX installation.

The `leoconfig` program is also useful to change the default screen configuration to some other resolution, including stereo. The default screen resolution for the ZX Graphics Accelerator is defined by the monitor ID code, read from the monitor. If the monitor returns an unknown ID code, the ZX Graphics Accelerator defaults to a screen resolution of 1152 × 900 at 66 Hz. While this resolution works with all of the monitors available for the workstations supplied by Sun, some monitors are able to take advantage of other resolutions available on the ZX Graphics Accelerator.

There are two ways to implement the change in screen resolution:

- Temporarily, by running the `leoconfig` program

- So that it will always boot up in the new resolution, by modifying the `leoconfig` script file

ZX Supported Monitors

Table 3-1 lists the monitors supported by the ZX Graphics Accelerator.

Table 3-1 Monitors Supported by ZX

Model	Sun Part Number	Type and Size	Monitor ID Sense Code	Supported Resolution and Refresh Rate
X248A	365-1068-01	Color 21"	2	**1280 × 1024 at 76 Hz**
GDM-20D10	365-1167-01	Color 20"	4	**1280 × 1024 at 67 Hz** 1280 × 1024 at 76 Hz 1152 × 900 at 76 Hz 1152 × 900 at 66 Hz 960 × 680 at 112 Hz (stereo)
GDM-1955A15	365-1081-01	Color 19"	3	**1152 × 900 at 66 Hz**
GDM-1962	365-1095-01	Color 19"	4	**1280 × 1024 at 67 Hz** 1152 × 900 at 76 Hz 1152 × 900 at 66 Hz
GDM-1962B	365-1160-01	Color 19"	4	**1280 × 1024 at 67 Hz** 1152 × 900 at 76 Hz 1152 × 900 at 66 Hz
GDM-1604A15	365-1079-01	Color 16"	3	**1152 × 900 at 66 Hz**
GDM-1662B	365-1159-01	Color 16"	6	**1152 × 900 at 76 Hz** 1152 × 900 at 66 Hz 1280 × 1024 at 67 Hz
CPD-1790	365-1151-01	Color 16"	3	**1152 × 900 at 66 Hz** 1024 × 768 at 76 Hz
X449A	365-1286-01	Color 15"	0	**1024 × 768 at 76 Hz**
GDM-20S5	365-1168-01	Grayscale 20"	2 or 4*	**1280 × 1024 at 76 Hz** or **1280 × 1024 at 67 Hz** 1152 × 900 at 76 Hz
17SMM4 A	365-1100-01	Grayscale 17"	6	**1152 × 900 at 76 Hz**
Non-Sun	--	Unknown	7	**1152 × 900 at 66 Hz**

Table 3-1 Monitors Supported by ZX *(Continued)*

Model	Sun Part Number	Type and Size	Monitor ID Sense Code	Supported Resolution and Refresh Rate
Resolutions in **bold type** are the default resolution at power-on initialization. * Monitor ID sense code is user-selectable by switch on rear.				
Monitors not supported:				
M20P110	365-1099-01	Grayscale 19"	4	N / A

Default Screen Resolutions

Table 3-2 lists the default screen resolutions by monitor ID sense code.

Table 3-2 ZX Frame Buffer Monitor Sense Codes

Code	Screen Resolution
7	1152 × 900 at 66 Hz
6	1152 × 900 at 76 Hz
5	1152 × 900 at 66 Hz
4	1280 × 1024 at 67 Hz
3	1152 × 900 at 66 Hz
2	1280 × 1024 at 76 Hz
1	1152 × 900 at 66 Hz
0	1024 × 768 at 76 Hz

3

Supported Screen Resolutions

Table 3-3 lists the screen resolutions the ZX Graphics Accelerator supports.

Table 3-3 ZX Supported Screen Resolutions

Screen Resolution	Vertical Refresh Rate	Description
1280 × 1024	67 Hz	Non-interlaced
1280 × 1024	76 Hz	Non-interlaced
1152 × 900	76 Hz	Non-interlaced
1152 × 900	66 Hz	Non-interlaced
1024 × 768	76 Hz	Non-interlaced
1024 × 768	60 Hz	Non-interlaced
960 × 680	108 Hz	Stereo, non-interlaced, 54 Hz field rate per eye
960 × 680	112 Hz	Stereo, non-interlaced, 56 Hz field rate per eye
770 × 575	50 Hz	Interlaced – PAL
640 × 480	59.94 Hz	Interlaced – NTSC

Changing the Screen Resolution Temporarily

To change the screen resolution temporarily:

1. **Exit from the window system.**
   If you are in a windowing environment, exit from it and wait for the system prompt to appear.

2. **As superuser (root), type the following command:**

```
example# /etc/opt/SUNWleo/bin/leoconfig -M monitor_type
```

where *monitor_type* is one of the values listed in Table 3-4. See also Table 3-3 on page 16.

Table 3-4 Monitor Types

monitor_type	Screen Resolution
1280_76	1280 × 1024 at 76 Hz, non-interlaced
1280_67	1280 × 1024 at 67 Hz, non-interlaced
1152_76	1152 × 900 at 76 Hz, non-interlaced
1152_66	1152 × 900 at 66 Hz, non-interlaced
1024_76	1024 × 768 at 76 Hz, non-interlaced
1024_60	1024 × 768 at 60 Hz, non-interlaced
stereo_108	960 × 680 at 108 Hz non-interlaced stereo, 54 Hz field rate per eye
stereo_114	960 × 680 at 112 Hz, non-interlaced stereo, 56 Hz field rate per eye
pal	770 × 575 at 50 Hz, interlaced (PAL)
ntsc	640 × 480 at 60 Hz, interlaced (NTSC)
default	The default resolution, defined by the monitor sense pins

For example, to change screen resolution to stereo at a 108 Hz vertical refresh rate, enter:

```
example# /etc/opt/SUNWleo/bin/leoconfig -M stereo_108
```

3. Restart the window system.

Modifying the `leoconfig` *Initialization File*

Before performing the following steps, read “Restrictions to Changing the Default Screen Resolution” on page 20. To change the `leoconfig` script so that the system boots up in the new screen resolution, edit the `leoconfig` script in the `/etc/init.d/leoconfig` file, as follows:

1. **As superuser (root), open the `leoconfig` file with the editor.**
   For example, to use the vi editor:

```
example# vi /etc/init.d/leoconfig
```

2. **Search for the "`MONTYPE=`" string in the file.**
   This string is usually one of the first lines in the file. You should see the lines shown below.

   There is one `MONTYPE=` line for each available screen configuration. By default, all but one of the lines are commented out (with the # character).

```
MONTYPE="-m default"
# MONTYPE="-m 1280_76"
# MONTYPE="-m 1280_67"
# MONTYPE="-m 1152_76"
# MONTYPE="-m 1152_66"
# MONTYPE="-m 1024_76"
# MONTYPE="-m 1024_60"
# MONTYPE="-m stereo_108"
# MONTYPE="-m stereo_114"
# MONTYPE="-m pal"
# MONTYPE="-m ntsc"
```

3. **Comment out the line that specifies the current screen configuration.**
   In the above example, you would comment out the "-m default" line, as follows:

```
# MONTYPE="-m default"
```

4. **Delete the comment out character (#) from the line that supports your monitor.**
   The supported monitor types are listed in Table 3-3 on page 16. See also Table 3-1 on page 14.

For example, to change the screen resolution from the default to the higher resolution of 1280 × 1024 at 76 Hz, delete the comment (the # character) from the `MONTYPE="-m 1280_76"` line. The file should now look like this:

```
# MONTYPE="-m default"
MONTYPE="-m 1280_76"
# MONTYPE="-m 1280_67"
# MONTYPE="-m 1152_76"
# MONTYPE="-m 1152_66"
# MONTYPE="-m 1024_76"
# MONTYPE="-m 1024_60"
# MONTYPE="-m stereo_108"
# MONTYPE="-m stereo_114"
# MONTYPE="-m pal"
# MONTYPE="-m ntsc"
```

5. **Save the file and exit the editor.**
   In vi, press Esc and type:

   ```
   :wq
   ```

6. **Save all your work.**
   Refer to Chapter 6, "Working with Documents," in the *Sun System User's Guide* for more information about ending a work session and saving your files. If you do not save your work, you can lose it when you reboot the system.

7. **Exit from the window system.**
   If you are in a windowing environment, exit from it and wait for the system prompt to appear.

8. **As superuser (root) again, execute the `leoconfig` program:**

   ```
   example# /etc/init.d/leoconfig
   ```

9. **Exit superuser and restart the window system.**
   The system should now be in the new screen resolution.

3

ZX Graphics Accelerator Restrictions

The ZX Graphics Accelerator has some limitations on its ability to support alternate screen resolutions. If you are using a Sun monitor and not changing the default screen resolution by way of the `leoconfig` program, you can disregard the following restrictions.

Using a Non-Sun Monitor as Console

If you use a non-Sun monitor as the workstation console, the monitor you use must meet both of the following requirements:

1. The monitor must support a screen resolution of 1152 × 900 at 66 Hz (the default screen resolution for a non-Sun monitor, as shown in Table 3-1 on page 14).
2. The monitor must not drive the monitor ID sense lines, or must conform to the sense codes and resolutions listed in Table 3-1 on page 14.

Restrictions to Changing the Default Screen Resolution

There are restrictions to changing the default screen resolution with the `leoconfig` program.

When you modify the `leoconfig` initialization program to change from the default screen resolution to a resolution of 1024 × 900 or less, excluding stereo, you will be unable to see the bottom portion of the display area during boot up before the window system starts. This means that you may not be able to see all of the start-up messages or to see what you are typing when you log in. Specifically, to avoid this problem you must not set the monitor type to any one of the following:

```
1024_76
1024_60
pal
ntsc
```

For those applications that require the lower resolutions, such as `pal` and `ntsc`, use a two-headed system. With a two-headed system where the ZX monitor is not used as the boot console, you may operate the ZX monitor in any of the supported screen resolutions. For more information, see Chapter 8, “Multiple Monitors on a System.”

3

SX Frame Buffer 4

This chapter describes how to change the display resolution on the SX Frame Buffer (cgfourteen). If you want to run OpenWindows on a SPARCstation 10SX system or a SPARCstation 20 system, you may need to perform additional configuration tasks after the initial installation.

You can use the `/usr/kvm/cg14config` utility to:

- Specify a different screen resolution.
- Change the values in the gamma lookup table.

For more information, see the `cg14config` man page.

4

SX Supported Monitors

Table 4-1 lists the monitors supported by the SX Frame Buffer and the alternate screen resolutions, if any, that each monitor supports.

Table 4-1 Monitors Supported by SX

Model	Sun Part Number	Type and Size	Monitor ID Sense Code	Supported Resolution and Refresh Rate
GDM-20D10	365-1167-01	Color 20"	4	**1152 × 900 at 76 Hz** 1280 × 1024 at 67 Hz 1280 × 1024 at 76 Hz 1152 × 900 at 66 Hz
GDM-1955A15	365-1081-01	Color 19"	3	**1152 × 900 at 66 Hz**
GDM-1962	365-1095-01	Color 19"	4	**1152 × 900 at 76 Hz** 1280 × 1024 at 67 Hz 1152 × 900 at 66 Hz
GDM-1962B	365-1160-01	Color 19"	4	**1152 × 900 at 76 Hz** 1280 × 1024 at 67 H**z** 1152 × 900 at 66 Hz
GDM-1604A15	365-1079-01	Color 16"	3	**1152 × 900 at 66 Hz**
GDM-1662B	365-1159-01	Color 16"	6	**1152 × 900 at 76 Hz** 1152 × 900 at 66 Hz 1280 × 1024 at 67 Hz
CPD-1790	365-1151-01	Color 16"	3	**1152 × 900 at 66 Hz** 1024 × 768 at 76 Hz
GDM-20S5	365-1168-01	Grayscale 20"	2 or 4*	**1280 × 1024 at 67 Hz** **1152 × 900 at 76 Hz**
17SMM4 A	365-1100-01	Grayscale 17"	6	**1152 × 900 at 76 Hz**
Non-Sun	--	Unknown	7	**1152 × 900 at 66 Hz**

Resolutions in **bold type** are the default resolution at power-on initialization.
* Monitor ID sense code is user-selectable by switch on rear.

Default Screen Resolutions

Table 4-2 lists the default screen resolutions by monitor ID sense code.

Table 4-2 SX Frame Buffer Monitor Sense Codes

Code	Screen Resolution
7	1152 × 900 at 66 Hz
6	1152 × 900 at 76 Hz
5	1024 × 768 at 60 Hz
4	1152 × 900 at 76 Hz
3	1152 × 900 at 66 Hz
2	1280 × 1024 at 76 Hz*
1	1600 × 1280 at 76 Hz*
0	1024 × 768 at 60 Hz

* The 4-Mbyte VSIMM drops to 8 bits per pixel at these resolutions.

4

Changing the Screen Resolution

To change the screen resolution, use the `cg14config` command with the following format:

```
# /usr/kvm/cg14config -d device -r resolution
```

where

device is the cgfourteen device to configure. The default is `/dev/fb`.
resolution is one of the values listed in Table 4-3.

Table 4-3 SX Supported Screen Resolutions

resolution	Screen Resolution
1600x1280@66	1600 × 1280 at 66 Hz
1280x1024@66	1280 × 1024 at 66 Hz
1152x900@66	1152 × 900 at 66 Hz
1152x900@76	1152 × 900 at 76 Hz
1024x800@84	1024 × 800 at 84 Hz
1024x768@70	1024 × 768 at 70 Hz
1024x768@66	1024 × 768 at 66 Hz
1024x768@60	1024 × 768 at 60 Hz

For example, to change the screen resolution to 1280 × 1024 at 66 Hz, enter:

```
# /usr/kvm/cg14config -d /dev/fb -r 1280x1024@66
```

Changing the Pixel Depth

After starting OpenWindows, the window server configures the SX Frame Buffer to support the maximum pixel depth for the screen resolution and frame buffer memory size that you selected. Typically, 32 bits are allocated to each display pixel. But, on the 4-megabyte SX frame buffer, you can increase the screen resolution by choosing a depth of 16 bits per pixel, with some loss of features.

Any frame buffer memory not visible on the monitor display is available to the window server for storing pixmaps. If you want to maximize the amount of off-screen pixmap storage available for your applications, you may need to add the line

```
pixelmode="8"
```

to the cg14 frame buffer entry in the `/usr/openwin/server/etc/OWconfig` file. This forces the window server to initialize the SX Frame Buffer at 16 bits per pixel, regardless of the frame buffer memory size. Table 4-4 summarizes the available features at the 16-bit and 32-bit pixel depths.

Table 4-4 SX Frame Buffer Visuals and Double-buffering

Underlay Visuals	**32-bit**	**16-bit**
24-bit TrueColor	Yes	No
8-bit PseudoColor	Yes	Yes
8-bit StaticColor	Yes	Yes
8-bit Greyscale	Yes	Yes
8-bit StaticGrey	Yes	Yes
8-bit TrueColor	Yes	Yes
8-bit DirectColor	Yes	Yes
Overlay Visuals		
8-bit PseudoColor	Yes	Yes
Double Buffering		
24-bit pixmaps	Software	No
8-bit pixmaps	Hardware	Software
Note: Overlay visuals are limited to 230 colors.		

4

To support the addition of overlay visuals in Solaris 2.4, the minimum SX Frame Buffer depth has been increased from 8 bits to 16 bits per pixel. If you are using `pixelmode="8"` and also upgrading from the Solaris 2.3 to Solaris 2.4 software environment, you may notice some performance degradation of some of the Xlib functions in OpenWindows.

Creator Graphics Accelerator 5

This chapter describes how to change the Creator™ and Creator 3D™ Graphics Accelerator screen resolution to work properly with different monitors.

You can change the Creator X11 screen and associated graphics hardware through the `ffbconfig` utility. Options are specified on the command line. The specified options are stored in the `OWconfig` file. You use these options to initialize the Creator device the next time Xsun is run on that device. Updating options in the `OWconfig` file provides persistence of these options across Xsun sessions and system reboots.

You use the `ffbconfig` utility to specify the following:

- Video mode (screen resolution and refresh rate)
- Type of visuals (linear or nonlinear)
- Whether to use 8-bit pseudocolor visual (overlay visual)
- Whether linear visuals will come before their nonlinear counterparts
- The maximum number of Creator X channel pixels reserved for use as WIDs

5

Default Screen Resolutions

Table 5-1 lists the default screen resolutions by monitor ID sense code.

Table 5-1 Creator Frame Buffer Monitor Sense Codes

Code	Screen Resolution
7	1152 × 900 at 66 Hz
6	1152 × 900 at 76 Hz
5	1152 × 900 at 66 Hz
4	1280 × 1024 at 67 Hz
3	1152 × 900 at 66 Hz
2	1280 × 1024 at 76 Hz
1	1152 × 900 at 66 Hz
0	1024 × 768 at 76 Hz

If the Creator is unable to determine the monitor type, such as for non-Sun monitors, it defaults to a resolution of 1152 × 900 at 66 Hz.

Supported Screen Resolutions

Table 5-2 lists the screen resolutions the Creator Graphics Accelerator supports.

Table 5-2 Creator Supported Screen Resolutions

Screen Resolution	Vertical Refresh Rate	Description
1280 × 1024	76 Hz	Non-interlaced
1280 × 1024	67 Hz	Non-interlaced
1152 × 900	76 Hz	Non-interlaced
1152 × 900	66 Hz	Non-interlaced
1024 × 800	84 Hz	Non-interlaced
1024 × 768	77 Hz	Non-interlaced
1024 × 768	70 Hz	Non-interlaced
1024 × 768	60 Hz	SVGA

Table 5-2 Creator Supported Screen Resolutions *(Continued)*

Screen Resolution	Vertical Refresh Rate	Description
960 × 680	112 Hz	Stereo, non-interlaced, 56 Hz field rate per eye
960 × 680	108 Hz	Stereo, non-interlaced, 54 Hz field rate per eye
768 × 575	50	Interlaced – PAL
640 × 480	60 Hz	Interlaced – NTSC

Changing the Screen Resolution Temporarily

This example changes the screen resolution temporarily, for example as a test to determine if the monitor supports the specified resolution.

```
ffbconfig -res video-mode try
```

Table 5-3 lists the *video-mode* options.

You will have 5 seconds to confirm the video mode by typing **y**.

Table 5-3 Creator Screen Resolution Formats

Video Mode		
Built-in	**Symbolic Name**	**Resolution**
`1280x1024x76`	`1280`	1280 × 1024 at 76 Hz
`1280x1024x67`		1280 × 1024 at 67 Hz
`1152x900x76`	`1152`	1152 × 900 at 76 Hz
`1152x900x66`		1152 × 900 at 66 Hz
`1024x800x84`		1024 × 800 at 84 Hz
`1024x768x77`		1024 × 768 at 77 Hz
`1024x768x70`		1024 × 768 at 70 Hz
`1024x768x60`	`svga`	1024 × 768 at 60 Hz
`960x680x112s`	`stereo`	960 × 680 stereo at 112 Hz per eye

Table 5-3 Creator Screen Resolution Formats *(Continued)*

Video Mode		
Built-in	**Symbolic Name**	**Resolution**
`960x680x108s`		960 × 680 stereo at 108 Hz per eye
`768x575x50i`	`pal`	768 × 575 at 50 Hz, interlaced
`640x480x60i`	`ntsc`	640 × 480 at 60 Hz, interlaced

Changing Screen Resolution to Stereo

This example changes the screen resolution to 960 × 680 at 112 Hz, stereo the next time Xsun is run.

```
ffbconfig -res stereo
```

Changing the Visual List Order

By default, the nonlinear visual comes before the linear visual on the screen visual list. You can modify the order of the visual list by using the `ffbconfig` command.

Most 3D applications want to use a linear visual. Some 3D applications do not search for a linear visual using XSolarisGetVisualGamma(3). Instead, most of these applications simply search the screen visual list for the first 24-bit TrueColor visual they find. To allow these applications to run with the correct visual, the `-linearorder` option allows you to change the visual list order so that the linear 24-bit TrueColor visual is the first one the application finds.

OpenWindows should not be running when you run the `ffbconfig` script. Start OpenWindows after `ffbconfig` has set the linear order you desire.

To change the setting, enter the `ffbconfig` command with one of the `-linearorder` options. For example:

```
ffbconfig -linearorder first
```

Creator Window System 6

This chapter describes the behavior of the Solaris X11 window system on the Creator and Creator 3D Graphics Accelerators.

Introduction

The Creator accelerators are in many ways a high performance combination of the GX, SX, ZX, and S24 display adaptors.

Like the ZX graphics accelerator, the Creator accelerator provides advanced frame buffer capabilities such as:

- Multiple plane groups
- Hardware double buffering
- Non-interfering transparent overlays
- 3D acceleration
- Stereoscopic support

Like the SX accelerator, the Creator accelerator is an X-channel architecture display adaptor. See “Creator Visuals.”

Like the GX accelerator, the Creator accelerator provides accelerated X11 rendering operations and hardware cursor support. See “Cursor Management.”

Like the S24 accelerator, the Creator accelerator provides both gamma corrected and uncorrected visuals. See the section “Creator Visuals.”

6

Like GX, SX, and ZX, the Creator accelerator supports multiple monitor video modes. See the section “Device Configuration.”

The Creator accelerator comes in two configurations: SB (Creator) and DBZ (Creator 3D). SB stands for “Single Buffer” and DBZ stands for “Double Buffer plus Z.” Z values are per-pixel depth information. These configurations differ in the amount of video memory on the board. The Creator 3D accelerator provides additional memory for hardware double buffering and 3D rendering.

Creator Visuals

Default Visual

The built-in factory default visual for the Creator accelerator is eight-bit PseudoColor. The user can specify a different default visual using the defdepth and defclass options of Xsun(1) and the -defoverlay and -deflinear options of ffbconfig(1m).

Note – Another name for the Creator accelerator is FFB, the Fast Frame Buffer. This name is used in Creator software package names, loadable device pipeline module names, the name of the configuration program, and device man pages. Hence the name ffbconfig.

When starting OpenWindows, the user can specify an alternate default visual using the normal OpenWindows command line options. Any of the exported visuals can be selected as the default. For example, the user can select the 24-bit TrueColor visual to be the default by using the openwin defdepth 24 option (see Xsun(1)). A 24-bit default is recommended to reduce colormap flashing.

List of Visuals

The Creator accelerator exports eleven visuals on the X11 screen visual list. These visuals can be queried using XGetVisualInfo(3) or XMatchVisualInfo(3). The linearity of a visual can be queried using XSolarisGetVisualGamma(3).

8-bit PseudoColor
8-bit StaticColor
8-bit GrayScale
8-bit StaticGray

8-bit TrueColor
8-bit DirectColor
8-bit StaticGray Linear

24-bit TrueColor
24-bit DirectColor
24-bit TrueColor Linear

8-bit PseudoColor Overlay

Note – In the above list, a visual is non-linear (not gamma corrected) unless it is explicitly specified to be linear. Also, an 8-bit visual resides in the Creator accelerator underlay plane group unless it is explicitly specified to reside in the overlay. 24-bit visuals are always underlay.

Figure 6-1 shows how the Creator accelerator visuals relate to the pixel storage (plane groups) in the frame buffer. X, B, G, and R denote the four 8-bit channels in which pixel data can be stored. The B, G, and R channels can either store 8-bit pixel data (in the red channel only) or it can store 24-bit pixel data (using all three channels). The R channel provides storage for windows of seven different visual types (the 8R visuals). The BGR channels provide storage for windows of three different visual types (the 24-bit visuals). There is only one visual provided by the X channel: 8X PseudoColor.

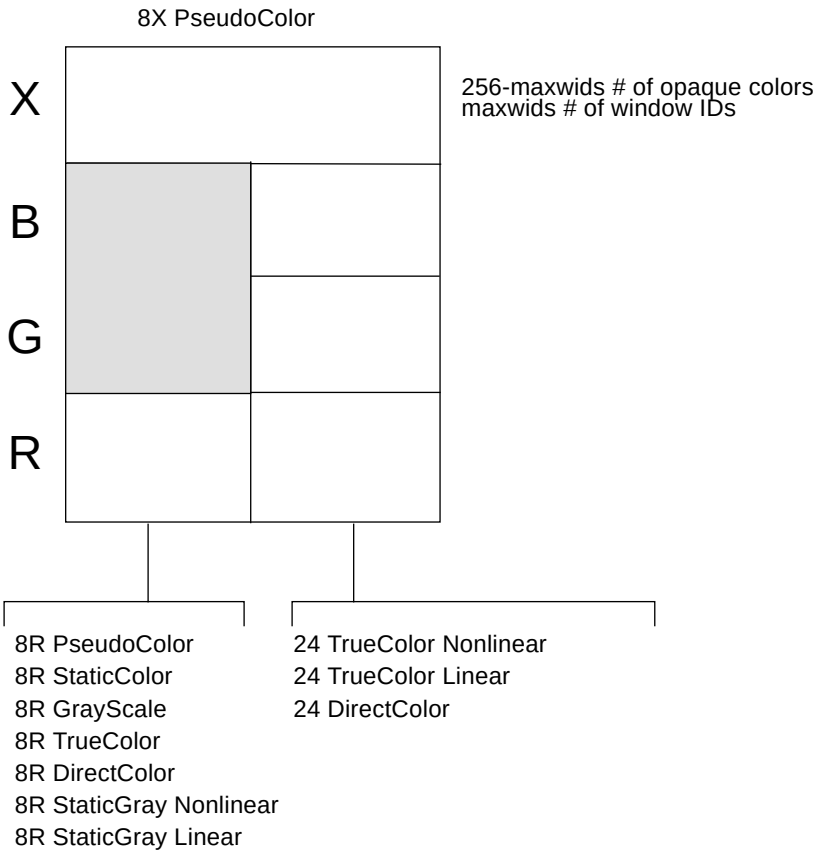


Figure 6-1 The 11 Creator Accelerator Visuals

The colormap_size of the underlay visuals is 256. The colormap_size of the overlay visual is 256 - `maxwids`.

Overlay/Underlay Structure

The underlay 8-bit PseudoColor visual is sometimes referred to as the 8R visual, because the pixel is stored in the red channel of the frame buffer. The overlay 8-bit PseudoColor visual is referred to as the 8X visual, because it gets stored in the X channel.

The pixels of windows in the overlay visual do not interfere with the pixels of windows in the underlay visuals. However the pixels of windows in the underlay visuals do interfere with pixels of windows in the overlay visual. This is different from the ZX accelerator, which has mutually non-interfering underlays and overlays. Like the ZX accelerator, the pixels of Creator 8-bit underlay windows interfere with the pixels of 24-bit underlay windows.

The Creator accelerator follows an *X-channel architecture*. In this architecture, some of the pixel values in the 8X plane group display opaque colors and some of the codes are used as window IDs which control the display of pixels in the underlay visuals.

`maxwids` is an ffbconfig(1m) configuration option which specifies how many of the overlay pixel values are to be used as hardware window IDs. See "Hardware Window IDs." The minimum legal value for `maxwids` is 1. The default value is 32. Thus, the overlay visual is a *partial* visual—it has less than the usual 256 colormap entries. For colormaps of this visual, if the client renders with a pixel value greater or equal to the specified number of colormap entries no error will be generated and the colors displayed will be undefined.

Comparison with the SX Accelerator

The visual architecture of the Creator accelerator is most similar to the CG14, the display adaptor of the SX accelerator. CG14 is also an X-channel architecture display adaptor which has 8-bit and 24-bit underlay visuals and a single 8-bit PseudoColor overlay visual. However, there are two primary differences.

Gamma Correction

The Creator accelerator has some visuals that are gamma corrected and some that are not. A gamma corrected visual is called a *linear* visual. The linear visuals are:

- 24-bit TrueColor Linear
- 8-bit StaticGray Linear

Both linear and non-linear visuals are present on the X11 screen visual list and these can be queried with XGetVisualInfo. Because linearity is not a visual property recognized by the X11 core protocol, an extended routine must be

called to distinguish a linear visual from its nonlinear counterpart. This routine is XSolarisGetVisualGamma. Refer to the XSolarisGetVisualGamma(1) man page for further details.

CG14 does not have linear visuals like the Creator accelerator. It performs gamma correction using a special Gamma LUT that affects the whole screen. Thus, it is not possible on the CG14 to have both gamma corrected and uncorrected 24-bit windows on the screen at the same time. But on the Creator accelerator, this is possible.

Single Color LUT

CG14 has two hardware color LUTs. One is used by the 8-bit underlay visuals and one is used by the 8-bit overlay visual. In contrast, the Creator accelerator has only one hardware color LUT. This means that an overlay window on the Creator accelerator will colormap flash against an 8-bit underlay window unless care is taken to make sure that the colormaps of the two windows have the same colors in the same pixel locations.

Programmers should take care to share overlay and underlay colors when writing transparent overlay applications which are intended to run on the Creator accelerator. Since the overlay visual is always a different visual from the underlay visual, a transparent overlay application always requires at least two separate colormaps: one for the overlay and one for the underlay. The overlay window is usually a child of the underlay window and the pixels are kept correlated (i.e. spatially congruent) by the application. In this situation, when the mouse pointer is inside the boundary of the underlay/overlay window pair, the overlay colormap will be installed in the hardware CLUT and the underlay colormap will not be installed. Thus, care should be taken to ensure that underlay pixels display the correct colors when viewed through the overlay colormap. This can be done by allocating colors in the same position in both the underlay and overlay colormaps.

Comparison with the ZX Accelerator

Default Visual is not Overlay

Unlike the ZX accelerator, the built-in factory default visual on the Creator accelerator is not an overlay. On the ZX accelerator, the default visual is the overlay 8-bit PseudoColor visual. But on the Creator accelerator, like the SX accelerator, it is the 8-bit *underlay* PseudoColor visual. This means that

applications that wish to create popup windows that are non-damaging with underlay windows cannot simply use the default visual. Instead, applications should call XSolarisOvlSelectBestOverlay to find a non-damaging overlay visual. Refer to the Solaris documentation on the OVL extension to X.

Note – XSolarisOvlSelectBestOverlay was first introduced in Solaris 2.4. If an application needs to run on Solaris 2.3 or earlier as well as 2.4, it is recommended that the external reference to this function be defined as #pragma weak. The program can then check the value of this symbol. If this symbol has the value of 0, then the program is running on Solaris 2.3 or earlier. In this case, XSolarisOvlSelectBestOverlay cannot be called to find the overlay visual. Instead, the application can use XGetVisualInfo to find the first 8-bit visual with less than 256 colormap entries. However, this technique is specific to the Creator accelerator and code using it will not necessarily be portable to other devices.

Window Manager Implications

The fact that the Creator accelerator default visual is not an overlay causes problems when overlay windows are not override-redirect (i.e. are wrapped with a window manager decoration window). The Solaris-supported window managers olwm, mwm, and dtwm always wrap toolkit subwindows with decoration windows in the default visual. It does this even if an application specifies a non-default visual for the popup window.

For example, when the default visual is 8-bit underlay PseudoColor, even if an application specifies to a toolkit that a popup window is to be placed in the 8-bit *overlay* PseudoColor visual, the window manager will wrap it with an 8-bit *underlay* decoration window. Thus, the popup will continue to damage other underlay windows, which is contrary to the user's intention.

There are two ways to get around this limitation:

1. One way is to require the end user to configure the default visual to be the overlay visual. This may be done by typing the following before starting the window system:

```
/usr/sbin/ffbconfig -defoverlay true
```

But keep in mind that the overlay visual has less colormap entries than the underlay 8-bit visual. Thus, the default colormap may fill up quicker and this may lead to increased colormap flashing.

2. The other way is to rewrite the application to create override redirect pop-ups and manage them directly through Xlib, bypassing the toolkit.

Hardware Color LUT Usage

The Creator accelerator has a single hardware color LUT. The following visuals use this color LUT:

8-bit PseudoColor
8-bit StaticColor
8-bit GrayScale
8-bit TrueColor
8-bit DirectColor

24-bit DirectColor

8-bit PseudoColor Overlay

The colormaps of these visuals will colormap flash against each other. Refer to “Reducing Colormap Flashing” for a method of avoiding colormap flashing.

The other Creator accelerator visuals don’t use color LUT resources. Colormaps of these visuals never flash.

Reducing Colormap Flashing

The 24-bit TrueColor visual of the Creator accelerator can display over 16 million colors simultaneously without colormap flashing. Furthermore, the Creator rendering engine has been optimized for 24-bit rendering. Consequently, it is very desirable for users and X client programmers to put their applications into this visual.

Advice to End Users

Even though 24-bit TrueColor offers fast rendering and no colormap flashing, the built-in factory default visual on the Creator accelerator is 8-bit PseudoColor. This choice was made to accommodate X applications that don't properly handle a 24-bit visual. It is better to have programs run and colormap flash than to not run at all. Fortunately, the majority of desktop applications do properly run with this visual.

Users who desire less colormap flashing on their desktop can run the window system with the default visual configured to 24-bit TrueColor. This is the recommended mode of running the window system on the Creator accelerator.

To run OpenWindows in this mode, use the following command:

```
openwin -dev /dev/fbs/ffb0 defdepth 24
```

To run CDE in this mode, edit the file `/usr/dt/config/Xservers` and add "`defdepth 24`" to the appropriate X start-up command for your server.

When using this mode, you should be aware of the following.

1. As mentioned above, some X applications may not be able to handle the 24-bit default. Most programs of this sort fail to run, issuing a BadMatch error message. Other programs may core dump or draw incorrect colors. If you encounter such an application, you can diagnose the problem by rerunning the application under the 8-bit PseudoColor default visual. If the program works, it is probably not able to handle a 24-bit TrueColor default visual. Contact the application supplier and request that the program be upgraded. In the meantime, you will need to use the factory default 8-bit PseudoColor visual mode until the application is fixed.
2. When the default visual depth is 24-bit, pixmaps and window backing store will occupy four times the space that they did with an 8-bit depth. This typically does not increase the working set of the server, but it does increase its swap space usage. If you run programs that use lots of pixmaps or backing store windows, you may want to consider staying in 8-bit mode.

Advice to Programmers

X client programmers should strive to write programs that are *24-bit clean*. This means that they run properly when the default visual is 24-bit TrueColor.

There are several reasons why a program might fail to be 24-bit clean. Here are some examples of programming practices to avoid:

1. Never automatically assume that the default visual is 8-bit PseudoColor.

   Some programs only run in 8-bit depth because they are ports from 8-bit only systems and there are not enough resources to upgrade them. Some programs might store their lists of pixels in a 256-element array. Other programs may require a modifiable colormap in order to perform colormap double buffering.

   It is strongly recommended that programs should be made 24-bit clean. But if your program really absolutely requires 8-bit PseudoColor, be sure to check the depth and class of the default visual. If it's not the 8-bit PseudoColor visual, search the visual list until you find it.

2. Don't inherit the border pixel from the parent.

   On a multiple plane group device like the Creator accelerator, the depth the window you are creating may not match the depth of the parent window (which is often the root window). Unless you specify an explicit border pixel value, the window's border pixel value will be inherited from the parent. If the depths differ, XCreateWindow will fail with a BadMatch error. Thus, it is good programming practice to always to use XCreatewindow (rather than XCreateSimpleWindow) and to explicitly specify a border pixel value.

Programmers are strongly encouraged to test their applications under both "defdepth 8" and "defdepth 24" modes of the window system.

Hardware Window IDs

Each window requires a hardware Window ID (WID) to display the contents of its pixels.

Note – The term *Window ID* used in this context should not be confused with the X protocol term *window ID*. An X protocol window ID is an XID which uniquely identifies a window. A hardware window ID is a value rendered into the frame buffer which controls window appearance.

Some overlay pixel codes are treated as opaque pixels. These display visible colors. Other overlay pixel codes are used to control the display attributes of underlay windows. These codes are referred to as hardware window IDs (WIDs). Specifically, a certain number of codes at the high end of the colormap (toward 255) are used as WIDs. The actual number of WIDs is configurable via the `ffbconfig -maxwids` option. For each overlay code used as a WID, the number of overlay colormap entries is reduced by one. The default value of `maxwids` is 32, so there are 224 opaque pixel values in the overlay.

One hardware WID is always reserved for windows of the default visual. The rest of the WIDs are assigned on a first-come first-serve basis to windows that:

- Have a non-default visual, or
- Are double buffered, or
- Have a unique WID assigned to them for the purposes of hardware WID clipping. (This clipping technique is used by Sun's 3D rendering libraries).

Non-default visual windows can share a WID with other windows of the same visual. However, like unique-WID windows, double buffered windows always require a unique, dedicated WID.

Creating a window of one of these types reduces the number of windows of other types that can be created. If all available WIDs have been assigned, the call to XCreateWindow will return a BadAlloc failure.

`maxwids` must be a power-of-two. Thus, legal values for `maxwids` are: 1, 2, 4, 8, 16, and 32. The default value of `maxwids` is 32.

Reducing `maxwids` increases the number of opaque color pixels in the overlay visual, but reduces the number of XGL, double buffered, and non-default visual windows that the user can create.

Cursor Management

The Creator accelerator has a hardware cursor. The image of this cursor is directly mixed into the output video signal. A software cursor, on the other hand, must be rendered into the frame buffer and the previous frame buffer contents must be temporarily stored. A software cursor incurs more overhead than a hardware cursor. A hardware cursor provides optimal interactive response when moving the cursor.

The maximum size of the Creator hardware cursor is 64 × 64. Cursors with an image whose width or height is less than or equal to 64 use the hardware cursor. Otherwise, they are rendered as software cursors.

The software cursor on the Creator accelerator is rendered in the overlay plane group. Therefore software cursors interfere with pixels of overlay window but not with pixels of underlay windows.

An X11 cursor has a foreground and background color that the client application requests. The colors of hardware cursors are exactly what was requested. However, the colors of software cursors may be approximations of the colors requested.

Note – To work around a bug in existing Solaris Visual graphics libraries, which don't properly remove software cursors, the cursor will, in the presence of any DGA-grabbed window, be forced to be a hardware cursor, even if this entails truncating the cursor image.

Hardware Double Buffering

Hardware double buffering is supported through MBX and XGL. MBX and XGL double buffering cannot both be used on a window at the same time.

Hardware double buffering is provided for 8R windows on all the Creator accelerator configurations. Hardware double buffering for 24-bit windows is only provided on the Creator/DBZ configuration. 8X overlay windows are never hardware double buffered on any Creator accelerator configuration; 8X windows are always software double buffered.

Activating hardware double buffering through MBX consumes one WID. This will reduce the number of other WID-consuming windows that can be created. See section "Hardware Window IDs." If no WID is available, MBX will fall back to software buffering mode in which a copy from a back buffer pixmap to the window is used to flip the buffers.

In MBX, the buffer flips occur synchronously. That is, the server request does not return until the vertical retrace period of the monitor occurs and the buffer is flipped. X11 clients may continue to run and send requests to the server, but they will not be processed until any pending buffer flip is complete.

Device Configuration

The program `/usr/sbin/ffbconfig` allows the user to alter the Creator accelerator's monitor video mode, the default visual, the default linear order, and the number of WIDs. Refer to the `ffbconfig`(1m) man page for details.

Attempts to grab stereo through DGA are rejected unless the monitor has been configured in a stereo video mode.

When the "shared" OWconfig file is being updated via `ffbconfig` an error will occur if the `/usr/openwin` directory is remotely mounted. This occurs even though `ffbconfig` is a setuid root program. This is because in Unix the root user of the local machine is different from the root users on a remote machine. The work-around is to log in to the remote machine in order to execute `ffbconfig`.

Performance Notes

Direct Xlib

Direct Xlib is not supported on the Creator accelerator. It is recommended that the X shared memory transport feature (new in Solaris 2.5) be used instead. To enable shared memory transport, set the following environment variables in the client environment:

```
setenv DISPLAY :0
setenv XSUNTRANSPORT shmem
setenv XSUNSMESIZE 512
```

Note – The last line specifies a client request buffer size of 512 kilobytes. Different request buffer sizes can be specified. However, 512 is a good compromise between the transport speed and the system memory resources consumed.

Applications that rely on XPutImage and Direct Xlib for fast pixel transfer into the frame buffer should instead use the MITSHM extension function XShmPutImage on the Creator accelerator. This function provides the fastest transfer of pixels into the frame buffer when the client is on the same machine as the X server.

If you are using XShmPutImage to a 24-bit visual, you may need to increase the amount of allocated shared memory beyond the default amount. See the next section for details.

X11perf -shmput<nn>

If you are running the x11perf benchmark when the default server visual is 24-bit TrueColor, the -shmput<nn> tests will fail. This is because this test requires more shared memory than provided by the default shmsys configuration. The X11R5 version of x11perf version will actually core dump, while the X11R6 version will report an error and make no measurement.

You will need to increase the amount of shared memory to allow this test to run. To do this, add the following line to `/etc/system`:

```
set shmsys:shminfo_shmmax=8192000
```

No Creator Pixel Copy Hardware

The Creator accelerator does not have hardware for copying pixels within the frame buffer. Pixels are copied by reading the pixels into CPU registers and then writing them back to the frame buffer. Thus, pixel copies within the frame buffer are CPU intensive. When the system is under heavy load, the performance of window dragging operations may suffer. One example of this is the slow-down of the ShowMe Whiteboard “Snap Region” image drag operation while an active SunVideo stream is being rendered. This slow down effect does not occur on a TGX accelerator, which has pixel copy hardware.

Miscellaneous

Background None Window Transient Color Effects

Dragging the imagetool image palette window with mouse shift-left over the main imagetool window can leave areas of the main window temporarily cyan-colored. These areas are quickly repaired, but they persist long enough for the user to notice them. The effect is particularly pronounced when the window being dragged is partially inside and partially outside the imagetool main window. These effects occur because the imagetool main window is an Xview WIN_TRANSPARENT window. This means that the background of the

corresponding X window is set to None. For this type of window, the server relies on the application to repair damaged areas after an occluding window is moved away. This method of damage repair is inherently slow, because the client must process damage events and send rendering requests.

The imagetool image palette window is an 8-bit window. The pixel data resides in the Creator accelerator 8-bit red channel. The background pixel value of this 8-bit window is a small number, typically 0x02. The data in the red channel doesn't damage data in the green and blue channels. The background of the main window was white (0xffffff) but when the 8-bit window occludes the pixel values are 0xffff02. When the 8-bit window occludes the area, it is viewed as an 8-bit window. But when the 8-bit window moves away, the server immediately changes the window ID values in this region, indicating that the region is now to be viewed as a 24-bit window. This happens before the background is repaired. So, for a brief instant, the old pixel values in this region (0xffff02) are viewed as a 24-bit pixel. This pixel value is viewed as pure blue + pure green, which forms cyan.

Note – This effect can also be seen on the SX frame buffer, although the SX frame buffer stores its 8-bit windows in the blue channel, so the transient color is green + red, which appears as yellow.

On the SX frame buffer, the effect is hardly noticeable because it is harder for the human eye to notice a contrast difference between yellow and white. But on the Creator accelerator, the difference between cyan and white is more noticeable. Also the effect takes longer to go away when the dragged window is partially outside the imagetool window. This is because the Creator hardware has more rendering state to maintain than the SX frame buffer and dragging outside the imagetool window causes the window manager to continuously update the imagetool window header. This causes text rendering operations to be interleaved with copy operations. And since both operations use different rendering state, the state must be continually loaded and reloaded into the Creator hardware context. Thus the repair of the transient color regions in on Creator in this situation is slower than the SX.

This example has dealt specifically with imagetool, but this will happen with any X window whose background is set to None. A bug for this has been filed for imagetool (Bug id 1215303).

To summarize, transient color effects like the one described above will occur on any window that has Background set to None. To avoid these effects, applications using this kind of window should use some other background mode instead. This allows the X server to repair damage as soon as possible after it occurs.

XIL Acceleration on Ultra/Creator 7

This chapter describes the XIL functions that are specific to the Creator and Creator 3D Graphics Accelerators.

Introduction

XIL is Sun's foundation imaging and video library. Several important XIL functions have been accelerated for the Ultra platforms using the UltraSPARC Visual Instruction Set (VIS) and the Creator Graphics Accelerator.

XIL Data Types

XIL supports a very general image structure, and not all types of XIL images are accelerated using VIS and Creator. The XIL data types supported in the VIS port are:

- 1, 2, 3, and 4-banded unsigned byte images.
- 1, 2, 3, and 4-banded signed short images.
- Band-aligned child images.
- All regions of interest.

7

Accelerated Functions

Table 7-1 shows the XIL functions accelerated using VIS. Note that display molecules are defined for one- and three-banded XIL_BYTE images.

Table 7-1 Accelerated XIL Functions

	VIS Acceleration		Molecules		
Function	**8-bit**	**16-bit**	**Display**	**Other**	**Comments**
xil_absolute		1 – 4 bands			
xil_affine	1 – 4 bands		x		
xil_add	1 – 4 bands	1 – 4 bands	x		
xil_add_constant	1 – 4 bands	1 – 4 bands	x		
xil_and	1 – 4 bands	1 – 4 bands	x		
xil_and_constant	1 – 4 bands	1 – 4 bands	x		
xil_band_combine	1 – 4 bands	1 – 4 bands			
xil_blend	1 – 4 bands	1 – 4 bands			
xil_cast					
8 → 16	1 – 4 bands	1 – 4 bands			
16 → 8	1 – 4 bands	1 – 4 bands			
xil_copy	1 – 4 bands	1 – 4 bands	x		
xil_convolve	1 – 4 bands	1 – 4 bands	x	x	See Note 1
xil_decompress					See Note 2
xil_get_pixel	1 – 4 bands	1 – 4 bands			
xil_lookup	1 – 4 bands	1 – 4 bands	x		
xil_min	1 – 4 bands	1 – 4 bands	x		
xil_max	1 – 4 bands	1 – 4 bands	x		
xil_multiply	1 – 4 bands	1 – 4 bands	x		
xil_multiply_constant	1 – 4 bands	1 – 4 bands	x		
xil_not	1 – 4 bands	1 – 4 bands	x		
xil_or	1 – 4 bands	1 – 4 bands	x		
xil_or_constant	1 – 4 bands	1 – 4 bands	x		

Table 7-1 Accelerated XIL Functions *(Continued)*

Function	VIS Acceleration 8-bit	VIS Acceleration 16-bit	Molecules Display	Molecules Other	Comments
xil_rescale	1 – 4 bands		x	x	See Note 3
xil_rotate	1 – 4 bands		x		
xil_scale	1 – 4 bands	1 – 4 bands	x		
xil_set_pixel	1 – 4 bands	1 – 4 bands	x		
xil_set_value	1 – 4 bands	1 – 4 bands	x		
xil_subtract	1 – 4 bands	1 – 4 bands	x		
xil_subtract_from_constant	1 – 4 bands	1 – 4 bands	x		
xil_threshold	1 – 4 bands	1 – 4 bands	x		

Note 1: xil_convolve is accelerated for 3 × 3, 5 × 5, and 7 × 7 kernels. The XIL_EDGE_EXTEND option is not accelerated. Separable kernels are accelerated as molecules for 3 × 3, 5 × 5, and 7 × 7 kernels.

Note 2: xil_decompress is accelerated for the following cases:

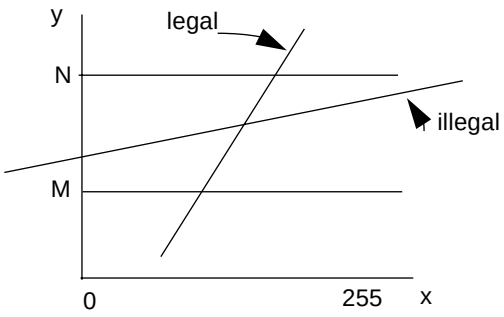
- xil_decompress + color_convert (JPEG compressed image or sequence and MPEG1 compressed sequence)
- xil_decompress + color_convert + display (MPEG1 compressed sequence)
- xil_decompress + color_convert + scale + display (MPEG1 compressed sequence)

This decoder adheres to full-precision 24-bit CCIR 601 YCC → RGB709 color space conversion.

Note 3: xil_rescale + xil_threshold + xil_threshold + display molecule is accelerated using the Creator depth-cueing hardware. However, some restrictions apply as follows:

- The first threshold must have a high value of 255. The low and map values must be equal (any number N between 0 and 255.)
- The second threshold must have a low value of 0. The high and map values must be equal (any number M between 0 and N).

- The parameters for xil_rescale must be chosen such that y values of the line represented must span the range of values between N and M while the x values are within 0 and 255. This is shown below.
- The images must be 1-banded XIL_BYTE images.



The following code is an example of a correct call to invoke this molecule:

```
float scale[1] = 1.5;
float offset[1] = -10;
float t1_lo[1] = 255;
float t1_hi[1] = 240;
float t1_map[1] = 240;
float t2_lo[1] = 0;
float t2_hi[1] = 15;
float t2_map[1] = 15;
xil_rescale(src,tmp1,scale,offset);
xil_threshold(tmp1,tmp2,t1_lo,t1_hi,t1_map);
xil_threshold(tmp2,tmp3,t2_lo,t2_hi,t2_map);
xil_display(tmp3,display);
```

However, if the same calling sequence is done with scale[1] = .40, the molecule will not be invoked because the line y = 0.40*x –10 only spans the range y = –10 to y = (255*0.40) – 10 = 92 between x = 0 and x = 255. For the molecule to execute, y must span at least the range y = 15 to y = 240 for x between 0 and 255.

Multiple Monitors on a System 8

This chapter describes how to use multiple monitors on a SPARCstation system. Skip this chapter if you are not interested in running multiple monitors on your system.

Most SPARCstations and UltraSPARC systems support multiple monitor configurations, providing that additional SBus slots (or PCI slots for some systems) are available.

This procedures described in this chapter require some knowledge of UNIX®. You should be familiar with UNIX and basic editing tools such as vi or emacs.

Multiple Monitor Configuration

When the system is booted, it looks for the `sbus-probe-list` which determines the order in which the SBus devices are addressed.

For the SPARCstation 10, SBus address f is reserved for the CPU and should always be the first address in the `sbus-probe-list`.

The addressing numbers are 0, 1, 2, and 3. However, 0 is reserved for the CPU and should always be the first address in the `sbus-probe-list` for all SPARCstation systems except the SPARCstation 10 system.

To find out the system information and `sbus-probe-list` type:

The following system information and `sbus-probe-list` (starting with `f`) will display if you have a SPARCstation 10 system:

```
nevada% eeprom
        .
        .
        .
sbus-probe-list=0123
        .
        .
```

```
nevada% eeprom
        .
        .
        .
sbus-probe-list=f0123
        .
        .
```

Device File Names

If you are using OpenWindows software on multiple monitors, you should be familiar with the way frame buffer devices are assigned to UNIX device file names. Multiple frame buffers used with OpenWindows software require that UNIX device file names for frame buffers be supplied on the command line when either is started.

The UNIX boot messages identify the frame buffer as `/dev/fb` (where fb is the type of frame buffer). The `/dev/fb` usually has another device file name such as `/dev/cgsix0`, `/dev/bwtwo0`, or `/dev/leo0` depending on the type of frame buffer. When a second frame buffer is added, the system decides which is `/dev/fb` based on the SBus slot number of each frame buffer and an EEPROM variable known as `sbus-probe-list`. The `/dev/fb` is the frame buffer in the first SBus slot defined in the `sbus-probe-list`.

If a TurboGXplus card is added to the system that already has a GX frame buffer, the `sbus-probe-list` also determines which one is `/dev/cgsix0` and which one is `/dev/cgsix1`.

Example: Assume that the `sbus-probe-list` on a SPARCstation 10 system has the default value of `f0123` and SBus slots 2 and 3 contain TurboGXplus cards. The TurboGXplus card in slot 2 will be known as `/dev/fb` and `/dev/cgsix0`; the TurboGXplus card in slot 3 will be known as `/dev/cgsix1`.

The command line examples shown in this document use possible device file names to refer to frame buffers. Remember to substitute the device file name appropriate for your system.

Checking the Available Frame Buffers

If you do not know the names of the frame buffer devices present in your system, you can check them by entering:

```
example% /etc/dmesg | more
```

This will display the system configuration including the types of available frame buffers in your system and the slots they occupy. The list of messages might be very long. Look for the lines that start with `cg` or `leo` (for color frame buffers) and `bw` (for black and white frame buffers).

```
Dec 5 11:16
SunOS Release 5.3 Version Generic [UNIX(R) System V Release 4.0]
Copyright (c) 1983-1993, Sun Microsystems, Inc.
mem=49152K (0x3000000)
 .
 .
cgsix0 at SBus0: SBus slot 1 0x0 SBus level 5 sparc ipl 7
cgsix0 is /sbus@1,f8000000/chsix@1,0
cgsix0: screen 1152x900, single buffered, 1M mappable, rev1
 .
 .
```

Note – The execution of the `dmesg` command may show too many other messages; you may not even see the appropriate system configuration messages shown below. If that is the case, you need to reboot your system. After rebooting, repeat the command shown above.

Starting OpenWindows from the Console

The following is an example of a `.login` file configured to start OpenWindows from the console.

```
#
# if possible, start the windows system.  Give user a chance to bail out
#
if ( `tty` == "/dev/console" && $TERM == "sun" ) then
    if ( ${?OPENWINHOME} == 0 ) then
        setenv OPENWINHOME /usr/openwin
    endif
    echo ""
    echo -n "Starting OpenWindows in 5 seconds (type Control-C to
interrupt)"
    sleep 5
    echo ""
    $OPENWINHOME/bin/openwin
    clear   # get rid of annoying cursor rectangle
    logout  # logout after leaving windows system
```

Running OpenWindows on Multiple Monitors

Running multiple monitors with OpenWindows Version 3 is fairly easy:

1. **Set up the OpenWindows Version 3 environment by typing the following command (substitute for `/usr/local` variable tag the actual pathname where OpenWindows Version 3 software is located):**
   Example:

```
example% setenv OPENWINHOME /usr/local/openwin
```

2. **To verify that a device file already exists for the desired frame buffer, enter:**

```
example# ls -l /dev/cgsix1
```

If the device file already exists, a message will display similar to the one shown below. If you see such a message, skip steps 3, 4, and 5 and go to step 6. If the system responds with the "not found" message, continue with step 3.

```
crw-rw-rw- 1 root      67,   0 Jan 10 1991 /dev/cgsix1
```

3. **Become superuser. Create a device file for your second frame buffer as shown in the following example. (The second frame buffer is assumed to be** `cgsix1`**.)**

```
example# cd /dev
example# MAKEDEV cgsix1
```

This creates a device file for the second frame buffer.

4. **Verify the newly created file by typing:**

```
example# ls -l /dev/cgsix1
```

The system will display a message similar to the following indicating that the device file has been successfully created:

```
crw-rw-rw- 1 root      67,   0 Jan 10 1991 /dev/cgsix1
```

5. **Exit the superuser mode.**

6. **Specify the screens that you wish to run by typing:**

```
example% $OPENWINHOME/bin/openwin -dev /dev/fb -dev /dev/cgsix1
```

Note – The order of the devices is important. The first device corresponds to the left screen. The second device corresponds to the right screen. The names of your devices (for example, `/dev/cgsix1`) may differ. Remember to use the device file name that is appropriate for your system.

Changing the Polling Order

This section provides information about the SBus polling order and how to change it. Skip this information if you know the order in which the SBus devices are addressed, or you are not interested in changing the order.

SBus Addresses

The two-slot SPARCstation systems, such as SPARCstation IPX and LX, have four SBus addresses: 0, 1, 2, and 3. SBus address 0 is located on the main logic board and is reserved for system use. SBus slots (SBus addresses) 1 and 2 are for customer-installable SBus cards. SBus slots 1 and 2 are the only physical slots. SBus address 3 is the frame buffer on the main logic board.

The SPARCstation 2 machine has no on-board frame buffer. Slots 1, 2, and 3 in this system are used to install SBus cards.

There is no on-board frame buffer on the SPARCstation 10 and SPARCstation 20 systems either. SBus address `f` is reserved for the CPU and should always be the first address in the `sbus-probe-list`. In the SPARCstation 10 and 20 systems, SBus address lines 0, 1, 2, and 3 are used for customer-installable SBus cards.

Polling Order

The polling order is determined by the `sbus-probe-list` parameter in the system's OpenBoot™ PROM. For the SPARCstation 10 and 20, you must begin with `f`. This parameter is set up to poll the slots in order from 0 to 3. You can change the order of slots 1, 2, and 3, but you must begin with slot 0.

For example, in SPARCclassic™, SPARCstation IPX, and SPARCstation LX configurations, if you install a frame buffer card into an SBus slot, the system automatically looks first for the frame buffer at the SBus slot rather than on the on-board frame buffer. If it finds a frame buffer at an SBus slot, the system establishes the video connection at that slot and looks no further. If you want the system to look first at the on-board frame buffer, you have to change the polling order to 0, 3, 1, 2.

Note – If you change `sbus-probe-list` in a SPARCstation IPX or LX system and select 3 as the console device, make sure a monitor is attached to the on-board frame buffer. Otherwise, the system will not recognize any other monitors regardless of their polling order.

Changing the sbus-probe-list

The following procedure describes how to change the `sbus-probe-list`.

Caution – The procedure described below is for experienced SunOS™ software users only. Ignore this procedure if you have only one monitor or you do not need to change the probe list. Any changes made to the system information displayed after typing the `eeprom` command described below will change the system configuration.

1. **As superuser, type `eeprom sbus-probe-list=0xyz` or `fwxyz` (for a SPARCstation 10 system), where `xyz` or `wxyz` is the order of SBus slots to be probed.**
   For example, in a SPARCclassic, SPARCstation IPX, and SPARCstation LX configuration, 0321 would cause SBus slot 3 (on-board frame buffer) to be probed first. After slot 3 is probed, SBus slots 1 and 2 will be probed, respectively.

```
nevada# eeprom sbus-probe-list=0312
```

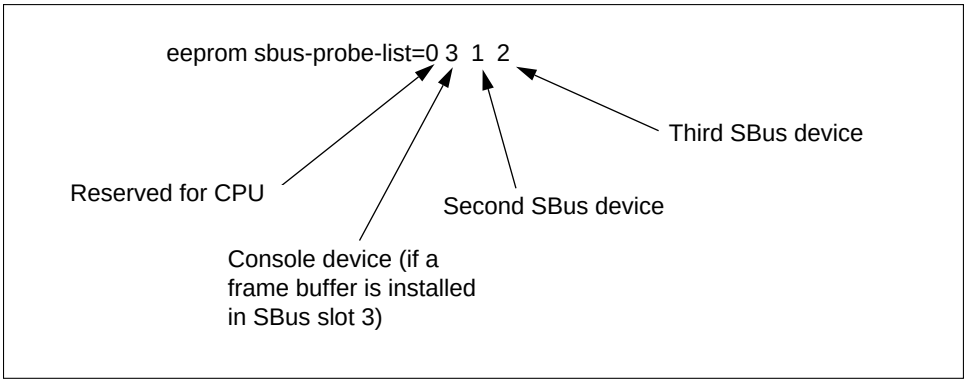


Figure 8-1 SBus Probe List Explanation

The leftmost character, except 0, in `eeprom sbus-probe-list` indicates the device that will be probed first. This will be the console device regardless of its physical location.

Note – If you are using a SPARCstation 10 or 20 system, the leftmost character in `sbus-probe-list` will be `f` (not 0) which is reserved for the CPU.

---

2. **Make sure a monitor is connected to the frame buffer identified as the "new console."**

3. **Reboot the system for the changes to take effect.**

Index

F

G

H

L

M

N

O

P

R

S

T

U

V

W

X

Z