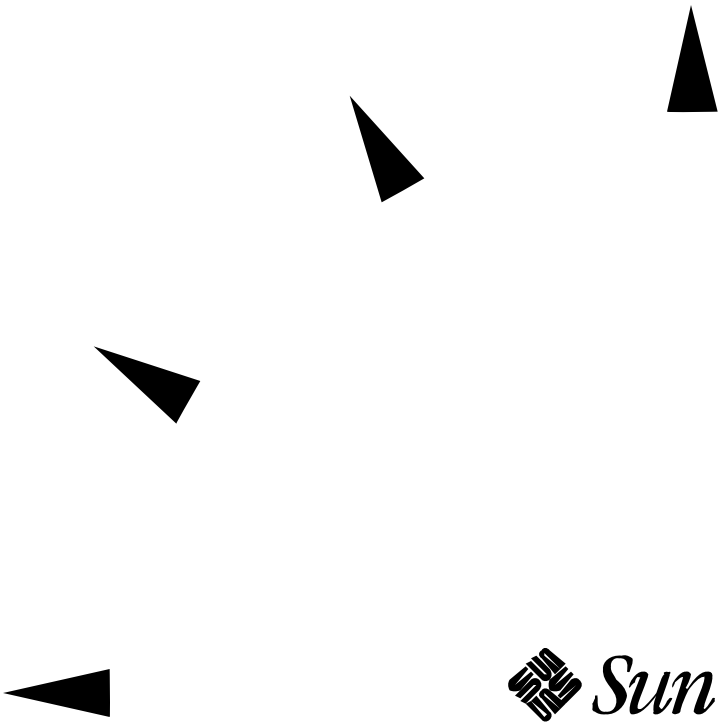
Platform Notes: SMCC Frame Buffers



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

Part No: 802-7677-10
Revision A, April 1997

Sun

Contents

Tables

Preface

Platform Notes: SMCC Frame Buffers contains important information on configuration options for 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
- M64 Graphics Accelerator

This book is intended for 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

This book consists of the following chapters:

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

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

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

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

Chapter 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, “The Creator Window System,” describes how the Solaris X11 window system works on the Creator and Creator 3D Graphics Accelerators.

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

Chapter 8, “M64 Graphics Accelerator,” describes the use of the `m64config` utility to change the display resolution on the M64 Graphics Accelerator.

Section 9, “Multiple Monitors on a System,” describes how to use multiple monitors on a SPARCstation system.

Typographic Conventions

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 the *User's Guide.* These are called *class* options. You *must* be root to do this.

Shell Prompts

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

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

Ordering Sun Documents

SunDocs℠ is a distribution program for Sun Microsystems technical documentation. Easy, convenient ordering and quick delivery is available from SunExpress™. You can find a full listing of available documentation on the World Wide Web: `http://www.sun.com/sunexpress/`

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	0800-90-61-57	0800-90-61-58
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	0800-55-19-26	0800-55-19-27
Japan	0120-33-9096	0120-33-9097

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 and 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 the rear switch.

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 `use-nvramrc?` variable, and if it is true, executes the Forth code that resides in `nvramrc`. Otherwise, it calls `probe-sbus`, `install-console` and `banner`.

Code Example 1-1 places resolution initialization between the `probe-sbus` stage and the `install-console` and banner stages.

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

Code Example 1-1 Resolution Initialization Between probe-sbus and install-console Stage

```
#!/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
'
```

The next line defines a Forth word called `vsetup` that 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`, pushes the video string on the stack, the number 4 defines the sense code of the monitor to change the resolution on. See Table 1-4 on page 7 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.

Code Example 1-2 changes the `cgsix` frame buffer on SBus slot 1.

Code Example 1-2 Changes to cgsix Frame Buffer in 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 `“override”` string is the actual entry point in the `cgsix` fcode PROM that reconfigures the resolution from the data on the forth stack. `execute-device-method` actually calls override and returns a pass or 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

Code Example 1-3 is a UNIX script that is used to configure the TurboGXplus for a resolution of 1280 × 1024 at 67 Hz:

Code Example 1-3 UNIX Script Method

```
#!/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

Code Example 1-4 uses the PROM method to configure the TurboGXplus for a resolution of 1280 × 1024 at 67 Hz:

Code Example 1-4 PROM Method

```
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

Code Example 1-5 shows how to set up a TurboGXplus card in slot 2 to 1024 × 768 at 60 Hz using a 16-inch monitor:

Code Example 1-5 PROM Method for Single Monitor Setup

```
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
```

1

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

Code Example 1-6 shows a UNIX script that 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:

Code Example 1-6 UNIX Script Method for Single Monitor Setup

```
#!/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
```

For more information on running multiple monitors, see Chapter 9, "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 were written without consideration for portability may not work with the default 24-bit visual of the S24 Frame Buffer.

Use the following workaround for these applications when starting OpenWindows. As superuser, enter:

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

This workaround sets the default visual to 8 bits, so all applications can 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 as 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.

Do not run OpenWindows when you run the `tcxconfig` script. Start OpenWindows after `tcxconfig` has set the linearity you want.

♦ **To display the current default setting, enter `tcxconfig` without options.**

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

- `linear` means that the linear visual is the default 24-bit TrueColor visual, which 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:**

```
# /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

You can change the screen resolution using two methods, depending on the frame buffer you select for the 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 become 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 `leoconfig` program. See the `leoconfig`(8) man page for more information.

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, which is 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 available monitors for the workstations supplied by Sun, some monitors can take advantage of other resolutions available on the ZX Graphics Accelerator.

You can change the screen resolution in two ways:

- Temporarily, by running the `leoconfig` program
- So that it always boots up in the new resolution by modifying the `leoconfig` script file

3

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**

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

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

Model	Sun Part Number	Type and Size	Monitor ID Sense Code	Supported Resolution and Refresh Rate*
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

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

Table 3-3 ZX Supported Screen Resolutions *(Continued)*

Screen Resolution	Vertical Refresh Rate	Description
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.**
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.)

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

Table 3-4 Monitor Types *(Continued)*

monitor_type	Screen Resolution
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. Follow this procedure to change the `leoconfig` script so that the system boots up in the new screen resolution.

▼ To Modify the `leoconfig` File:

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

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

2. **Search for the "`MONTYPE=`" string.**
   This string is usually one of the first lines in the file. The lines below are displayed. 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, 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 15. (See also Table 3-1.)

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.**
   If you do not save your work, it will be lost when you reboot the system.

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

8. **As superuser (root), 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.

ZX Graphics Accelerator Restrictions

The ZX Graphics Accelerator has some alternate screen resolution support limitations. 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:

- 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).
- The monitor must not drive the monitor ID sense lines, or must conform to the sense codes and resolutions listed in Table 3-1.

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 not be able 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. To avoid this problem you must *not* set the monitor type to any one of the following:

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

For applications that require 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 9, “Multiple Monitors on a System.”

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 as follows:**

```
# /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 following line to the cg14 frame buffer entry in the `/usr/openwin/server/etc/OWconfig` file:

```
pixelmode="8"
```

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		

Table 4-4 SX Frame Buffer Visuals and Double-buffering *(Continued)*

Underlay Visuals	**32-bit**	**16-bit**
24-bit pixmaps	Software	No
8-bit pixmaps	Hardware	Software

Note: Overlay visuals are limited to 230 colors.

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 the Solaris 2.4 software environment, some performance degradation may occur in some of the OpenWindows Xlib functions.

Creator Graphics Accelerator 5

This chapter describes how to change the Creator and Creator 3D Graphics Accelerator screen resolution to work 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.

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
- How to set the default screen visual
- How OpenGL visuals will be supported
- Whether Server Overlay Visuals (SOV) will be available
- The maximum number of Creator X channel pixels reserved for use as WIDs
- Whether the pseudocolor overlay visual will come before the pseudocolor underlay visual

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

Supported Screen Resolutions

Table 5-2 lists the Creator-supported screen resolutions. Some resolutions are supported only on Creator3D Revision 2 and later.

Table 5-2 Creator-supported Screen Resolutions

Screen Resolution	Vertical Refresh Rate	Description
1920 × 1200	70 Hz	Non-interlaced, Hi-res, only Creator 3D, rev 2
1920 × 1080	72 Hz	Non-interlaced, Hi-res, only Creator 3D, rev 2
1600 × 1280	76 Hz	Non-interlaced, Hi-res, only Creator 3D, rev 2
1600 × 1000	76 Hz	Non-interlaced, Hi-res, only Creator 3D, rev 2
1600 × 1000	66 Hz	Non-interlaced, Hi-res, only Creator 3D, rev 2
1440 × 900	76 Hz	Non-interlaced, Hi-res, only Creator 3D, rev 2
1280 × 1024	76 Hz	Non-interlaced

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

Screen Resolution	Vertical Refresh Rate	Description
1280 × 1024	67 Hz	Non-interlaced
1280 × 1024	60 Hz	Non-interlaced, only Creator, rev 2
1280 × 800	76 Hz	Non-interlaced, only Creator, rev 2
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	75 Hz	Non-interlaced, only Creator 3D, rev 2
1024 × 768	70 Hz	Non-interlaced
1024 × 768	60 Hz	SVGA
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
640 × 480	60 Hz	Non-interlaced, only Creator 3D, rev 2

Some monitors may not support some the resolutions supported by the Creator. You can get the list of resolutions supported by the Creator and the connected monitor.

▼ To find resolutions supported by Creator and the connected monitor

♦ **Use the** `ffbconfig` **command as follows:**

```
ffbconfig -res \?
```

5

Changing Screen Resolutions

You can change the screen resolution temporarily as a test to determine if the monitor supports the specified resolution.

Caution – Do not change screen resolution while the window system is running. Changing screen resolution while the window system is running may put the screen display in an unusable state.

▼ To Change Screen Resolutions Temporarily

♦ **Use the `ffbconfig` command as follows:**

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

Table 5-3 lists the *video-mode* options. You will have five seconds to confirm the video mode by typing **y**.

Table 5-3 Creator Screen Resolution Formats

Video Mode		
Built-in	**Symbolic Name**	**Resolution**
`1920x1200x70`		1920 × 1200 at 70 Hz
`1920x1080x72`		1920 × 1080 at 72 Hz
`1600x1280x76`		1600 × 1280 at 76 Hz
`1600x1000x76`		1600 × 1000 at 76 Hz
`1600x1000x66`		1600 × 1000 at 66 Hz
`1440x900x76`		1440 × 900 at 76 Hz
`1280x1024x76`	`1280`	1280 × 1024 at 76 Hz
`1280x1024x67`		1280 × 1024 at 67 Hz
`1280x1024x60`		1280 × 1024 at 60 Hz
`1280x800x76`		1280 × 800 at 76 Hz

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

Video Mode		
Built-in	**Symbolic Name**	**Resolution**
`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
`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
`640x480x60`		640 × 480 at 60 Hz

Changing Screen Resolution to Stereo

▼ **To change the screen resolution to stereo, enter the following**

```
ffbconfig -res stereo
```

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

5

Changing Screen Visuals List

The X server screen visuals list can be altered through `ffbconfig`. The `ffbconfig` options in Table 5-4 can be used to configure the list of the exported visuals for the specified device.

Table 5-4 The Default Settings of the `ffbconfig` Visual Flags

Name	Possible Values	Defaults in Solaris2.5.1/ 2.5.1 SHWP
`linearorder`	`first/last`	`last`
`deflinear`	`true/false`	`false`
`overlayorder`	`first/last`	`last`
`defoverlay`	`true/false`	`false`
`expvis`	`enable/disable`	`disable`
`sov`	`enable/disable`	`disable`

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

The desired visual ordering in the screen visuals list will be available whenever the window system is restarted.

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

```
ffbconfig -linearorder first
```

By default, the 8-bit PseudoColor visual comes before the 8-bit PseudoColor Overlay visual on the screen visual list. You can modify the order of the visual list by using the `ffbconfig` command.

Some applications that use the 8-bit PseudoColor Overlay visual, search the visual list for the first 8-bit PseudoColor visual they find. To enable these applications to run with the correct visual, use the `-overlayorder` option to change the visual list order so that the 8-bit PseudoColor Overlay visual is the first 8-bit PseudoColor visual the application finds.

The desired visual ordering in the screen visuals list will be available whenever the window system is restarted.

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

```
ffbconfig -overlayorder first
```

Changing the Default Visual

By default, the 8-bit PseudoColor underlay visual is the default visual of the screen. The default visual can be changed to either a linear underlay visual or an overlay visual through `ffbconfig`.

- **To set the default visual to be a linear visual, enter the `ffbconfig` command as follows:**

```
ffbconfig -deflinear true
```

- **To set the default visual to be an overlay visual, enter the `ffbconfig` command as follows:**

```
ffbconfig -defoverlay true
```

Caution – Since there is no linear overlay visual, the user should not specify both "`-deflinear true`" and "`-defoverlay true`" simultaneously, or the result will be undefined.

Caution – Note that the visual ordering options (`overlayorder` and `linearorder`) are independent of the default visual options (`defoverlay` and `deflinear`). Moving the overlay visual groups, for example, to the front does not automatically make it a default visual. Some applications make this assumption and hence receive a "BADMATCH" X error when they try to match the colormap created by the default visual and the first 8-bit PseudoColor visual they can find.

Changing OpenGL Visual Support

Solaris 2.5.1 SHWP supports the OpenGL visual expansion. With visual expansion, five visual groups: the 8-bit PseudoColor, 24-bit TrueColor (Linear and Non-Linear), 24-bit DirectColor and 8-bit PseudoColor Overlay, will be expanded from a single visual to multiple visual instances of the same visual type. Different instances of in the same visual groups represent different GLX capabilities (e.g. Single-buffer or double-buffer capable, monoscopic or stereoscopic capable, or a combination of both). The number of visual instances depends on whether the X server is started in monoscopic or stereoscopic mode, and also whether the device is a Creator or Creator 3D.

▼ **To activate OpenGL visual support (visual expansion), enter the following:**

```
ffbconfig -expvis enable
```

Changing SERVER_OVERLAY_VISUALS Support

SERVER_OVERLAY_VISUALS is one of the root window's properties that contains the visual ID, transparent type, transparent value, and layer of the server overlay visuals (SOV) of the screen. You can toggle the advertisement of this property and the export of the transparent server overlay visuals using `ffbconfig`.

▼ **To advertise SERVER_OVERLAY_PROPERTY and export SOV, enter the following:**

```
ffbconfig -sov enable
```

Impact on Screen Visual List by Various `ffbconfig` *Visual Flags*

Effect on the Default Visual and the Visual Group Ordering

In summary, the appearance of the X server screen visual list can be changed to fit the user's needs using any of the `ffbconfig` visual flags (e.g. `linearorder`, `overlayorder`, `expvis`, `sov`, etc.). This section briefly shows the effect of these visual options on the visual list.

Without any alteration using `ffbconfig`, the X server screen visual list can roughly be categorized in the following visual groups and order:

- 8-bit PseudoColor
- 8-bit Miscellaneous (StaticColor, Non-linear StaticGray, etc)
- 8-bit Linear StaticGray

- 8-bit PseudoColor Overlay
- 24-bit Non-linear TrueColor
- 24-bit DirectColor
- 24-bit Linear TrueColor

The default screen visual will be the 8-bit PseudoColor visual. The `deflinear` and `defoverlay` options will alter this pointer to point to the first 8- or 24-bit linear visual, depending on the X server depth when it starts up, and the first 8-bit overlay visual, respectively. You can rearrange the entire “8-bit PseudoColor Overlay” group, the “8-bit Linear StaticGray” group and the “24-bit Linear TrueColor” group to be ahead of all other visual groups of the same depth by using the `linearorder` and `overlayorder` flags. For example, if you specify the following:

```
ffbconfig -overlayorder first
```

The screen visuals list will be rearranged in the following order:

- 8-bit PseudoColor Overlay
- 8-bit PseudoColor
- 8-bit Miscellaneous (StaticColor, Non-linear StaticGray... etc)
- 8-bit Linear StaticGray
- 24-bit Non-linear TrueColor
- 24-bit DirectColor
- 24-bit Linear TrueColor

Effect on the Number of Visual Instances Within Selected Groups

The `expvis` flag will change the number of visual instances in the following visual groups:

- 8-bit PseudoColor
- 8-bit PseudoColor Overlay
- 24-bit Non-linear TrueColor
- 24-bit DirectColor
- 24-bit Linear TrueColor

The number of instances that `expvis` will alter depends on whether the monitor is in monoscopic or stereoscopic resolution. In monoscopic resolution, if `expvis` is enabled, the order of the screen visual groups will be preserved, but within each group mentioned above, a “double-buffer capable” visual instance will be added. In stereoscopic resolution, two additional visual instances: “double-buffer, stereo capable” and “single-buffer, stereo capable”, will be added. The “double-buffer capable” visual instances, if present, will always come ahead of the “single-buffer capable” visual instances, and the monoscopic visual instances will always come ahead of the stereoscopic ones.

For example, if you specify the following in stereoscopic resolution:

```
ffbconfig -overlayorder first -expvis enable
```

the screen visual list will be:

- 8-bit PseudoColor Overlay (Mono, Stereo)
- 8-bit PseudoColor (DB Mono, SB Mono, DB Stereo, SB Stereo)
- 8-bit Miscellaneous (StaticColor, Non-linear StaticGray... etc)
- 8-bit Linear StaticGray
- 24-bit Linear TrueColor (DB Mono, SB Mono, DB Stereo, SB Stereo)
- 24-bit Non-linear TrueColor (DB Mono, SB Mono, DB Stereo, SB Stereo)
- 24-bit DirectColor (DB Mono, SB Mono, DB Stereo, SB Stereo)

Note – There is no double-buffer capable overlay visual instance.

Addition of the SERVER_OVERLAY_VISUALS Property and the Transparent SOV Visuals in the 8-bit Overlay Group

Without the `sov` option being enabled, the only overlay visuals available are the ones without transparency. Enabling the `sov` option will add the transparent SOV visual instances into the screen visual list and also add the SERVER_OVERLAY_VISUALS property to the root window property. The transparent SOV visual instances belong to the “8-bit PseudoColor Overlay” visual group. The SERVER_OVERLAY_VISUALS property will contain the visuals’ ID, transparent type, value, and layer of all available overlay visuals of the screen.

For example, if you specify the following in stereoscopic resolution:

```
ffbconfig -overlayorder first -expvis enable -sov enable
```

the screen visuals list will be:

- 8-bit PseudoColor Overlay (Mono, Stereo, Mono SOV, Stereo SOV)
- 8-bit PseudoColor (DB Mono, SB Mono, DB Stereo, SB Stereo)
- 8-bit Miscellaneous (StaticColor, Non-linear StaticGray... etc)
- 8-bit Linear StaticGray
- 24-bit Linear TrueColor (DB Mono, SB Mono, DB Stereo, SB Stereo)
- 24-bit Non-linear TrueColor (DB Mono, SB Mono, DB Stereo, SB Stereo)
- 24-bit DirectColor (DB Mono, SB Mono, DB Stereo, SB Stereo)

and the SERVER_OVERLAY_VISUALS property will contain the following information:

```
============================================================
              SERVER OVERLAY VISUALS (SOV) Info
============================================================
No. of SOV visuals = 4
SOV #0,ID 0x36, TRANSPARENT_TYPE 0, VALUE   0, LAYER 1
SOV #1,ID 0x37, TRANSPARENT_TYPE 0, VALUE   0, LAYER 1
SOV #2,ID 0x38, TRANSPARENT_TYPE 1, VALUE 255, LAYER 1
SOV #3,ID 0x39, TRANSPARENT_TYPE 1, VALUE 255, LAYER 1

============================================================
```

The Creator Window System 6

This chapter describes how the Solaris™ X11 window system is used with the Creator and Creator 3D Graphics Accelerators.

The Creator accelerators are 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 “Creator Visuals.”

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

6

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 – The Creator accelerator is also called the Fast Frame Buffer (FFB). The FFB name is used in Creator software package names, loadable device pipeline module names, the configuration program, and device man pages.

When starting OpenWindows, you can specify an alternate default visual using standard OpenWindows command line options. Any of the exported visuals can be selected as the default. For example, you 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 helps to reduce colormap flashing.

List of Visuals

The Creator accelerator exports eleven visuals on the X11 screen visual list. You can query these visuals using `XGetVisualInfo`(3) or `XMatchVisualInfo`(3). The linearity of a visual can be queried using `XSolarisGetVisualGamma`(3). The visuals are:

- 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 previous 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 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 for three different visual types (the 24-bit visuals). Only one visual is provided by the X channel: 8X PseudoColor.

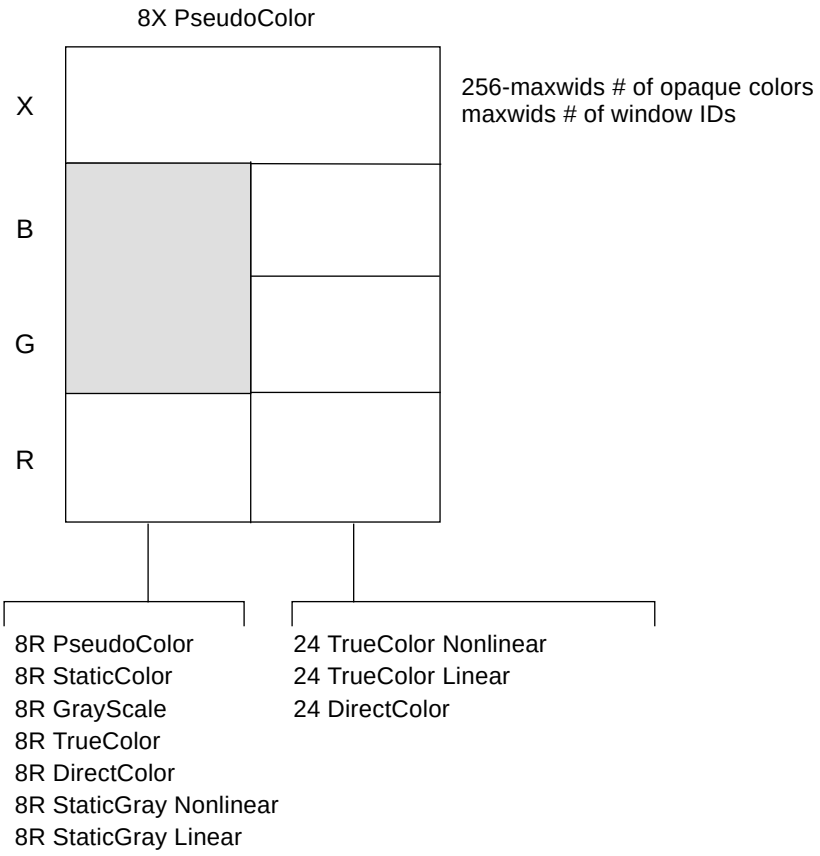


Figure 6-1 The 11 Creator Accelerator Visuals

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

Overlay and 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 is stored in the X channel.

The window pixels in the overlay visual do not interfere with the window pixels in the underlay visuals. However, window pixels in the underlay visuals do interfere with window pixels 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 pixel values in the 8X plane group display opaque colors and some codes are used as window IDs that control the display of pixels in the underlay visuals.

The `maxwids` configuration option to `ffbconfig`(1m) specifies how many of the overlay pixel values are used as hardware window IDs. See “Hardware Window IDs” for details. The minimum legal value for `maxwids` is 1. The default value is 32. Thus, the overlay visual is a *partial* visual because it has less than the usual 256 colormap entries. If the client renders colormaps of this visual with a pixel value greater or equal to the specified number of colormap entries, no error is generated and the colors displayed are 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 that has 8-bit and 24-bit underlay visuals and a single 8-bit PseudoColor overlay visual. However, there are two primary differences, as described in the sections that follow.

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, which 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 entire 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. This is possible on the Creator accelerator.

Single Color LUT

CG14 has two hardware color LUTs. One is used by the 8-bit underlay visuals and the other 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 precautions are taken to make sure that the colormaps of the two windows use the same colors in the same pixel locations.

When programming, take precautions to share overlay and underlay colors when writing transparent overlay applications that 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 correlated (i.e. spatially congruent) by the application. In this situation, when the mouse pointer is inside the boundary of the underlay and overlay window pair, the overlay colormap will be installed in the hardware CLUT and the underlay colormap will not be installed. Thus, take care 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.

Comparing Creator with the ZX Accelerator

The following sections describe differences and similatities between visuals on the Creator Graphics Accelerator and visuals on the ZX Graphics 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. On the Creator accelerator, like the SX accelerator, it is the 8-bit *underlay* PseudoColor visual. Thus, if you want to create applications with pop-up windows that are non-damaging with underlay windows, you cannot 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 on Solaris 2.4, define the external reference to this function 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 is not portable to other devices.

Window Manager Implications

Because the Creator accelerator default visual is not an overlay, problems occur when overlay windows are not override-redirect (i.e. 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. This occurs even if an application specifies a non-default visual for the pop-up window.

For example, when the default visual is 8-bit underlay PseudoColor, the window manager will wrap it with an 8-bit *underlay* decoration window even if an application specifies to a toolkit that a pop-up window is to be placed in the 8-bit *overlay* PseudoColor visual. Thus, the pop-up continues to damage other underlay windows, not the intended effect.

Use the following workarounds to this limitation:

- Require that the end user configure the default visual as the overlay visual by typing the following before starting the window system:

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

Keep in mind that the overlay visual has less colormap entries than the underlay 8-bit visual. Thus, the default colormap may fill up sooner, which may lead to increased colormap flashing.

- 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 colormap flash against each other. Refer to “Reducing Colormap Flashing” for how to avoid 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 is optimized for 24-bit rendering. Consequently, it is very desirable for users and X client programmers to use this visual.

Notes 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 was done to accommodate X applications that don't handle a 24-bit visual properly. It is better to have programs run and colormap flash than to not run at all. Fortunately, the majority of desktop applications do run properly 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** `/usr/dt/config/Xservers` **file and add "**`defdepth 24`**" to the appropriate X start-up command for your server.**

Be aware of the following conditions when using this mode:

- Some X applications cannot handle the 24-bit default. These type of programs usually fail to run and issue 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 probably cannot handle a 24-bit TrueColor default visual. Contact the application supplier and request an upgraded program. In the meantime, use the factory default 8-bit PseudoColor visual mode until the application is fixed.
- When the default visual depth is 24-bit, pixmaps and window backing store will occupy four times the space as in an 8-bit depth. This usually does not increase the working set of the server, but it does increase swap space. If you run programs that use lots of pixmaps or backing store windows, stay in 8-bit mode.

6

Notes to Programmers

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

A program might fail to be 24-bit clean for several reasons. Following are some programming practices to avoid:

- Do not assume that the default visual is 8-bit PseudoColor.

  Some programs only run in 8-bit depth because they have been ported from 8-bit-only systems and they have not been upgraded. Some programs might store their lists of pixels in a 256-element array. Other programs may require a modifiable colormap to perform colormap double buffering.

  If your program requires 8-bit PseudoColor, 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.

- Do not inherit the border pixel from the parent.

  On a multiple plane group device like the Creator accelerator, the depth of 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 border pixel value of the window is inherited from the parent. If the depths differ, `XCreateWindow` will fail with a BadMatch error. Always use `XCreatewindow` rather than `XCreateSimpleWindow` and explicitly specify a border pixel value.

Test your 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 – Do not confuse the term *Window ID* used in this context 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 that controls window appearance.

Some overlay pixel codes are treated as opaque pixels, which 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 through 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 remaining WIDs are assigned on a priority basis to windows that have the following characteristics:

- Non-default visual
- Double buffered
- Assigned a unique WID for the purposes of hardware WID clipping. (This clipping technique is used by the Sun 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 these types of windows reduces the number of other types of windows that can be created. If all available WIDs are 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 contents of the previous frame buffer 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 windows 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 rendered exactly as they are requested. However, the colors of software cursors may be approximations of the requested colors.

Note – A bug in existing Solaris Visual graphics libraries means that software cursors are not properly removed. To work around this bug, 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 (Creator3D) 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 reduces the number of other WID-consuming windows that can be created. See “Hardware Window IDs.” If no WID is available, MBX falls 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 will not be processed until pending buffer flips are complete.

Device Configuration

Use the `/usr/sbin/ffbconfig` program to alter the Creator accelerator's monitor video mode, default visual, 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 is configured in a stereo video mode.

When the "shared" `OWconfig` file is being updated via `ffbconfig`, an error occurs 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 workaround is to log in to the remote machine to execute `ffbconfig`.

Performance Notes

The following sections contain performance notes for the Creator Accelerator on various applications.

Direct Xlib

Direct Xlib is not supported on the Creator accelerator. The X shared memory transport feature (new in Solaris 2.5) should 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 X11perf Benchmark" for details.

The X11perf Benchmark

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 that provided by the default `shmsys` configuration. The X11R5 version of x11perf version will core dump, while the X11R6 version will report an error and make no measurement.

To run this test, you will need to increase the amount of shared memory.

- **To increase the amount of shared memory, 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. For example, the ShowMe Whiteboard "Snap Region" image drag operation may slow down while an active SunVideo stream is being rendered. This effect does not occur on a TurboGX accelerator, which has pixel copy hardware.

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 to

be noticed. 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 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 moment, 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.

This effect is not as noticeable on the SX frame buffer, because it is more difficult to detect a contrast difference between yellow and white. On the Creator accelerator however, the difference between cyan and white is more pronounced. The effect also takes longer to go away when the dragged window is partially outside the imagetool window. This occurs because the Creator hardware has more rendering state to maintain than the SX frame buffer; 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. Applications that use this type of window should use some other background mode to avoid these effects. The X server can then repair the damage soon after it occurs.

XIL Acceleration on the Creator Graphics Accelerator

This chapter describes the XIL functions that are specific to the Creator and Creator3D Graphics Accelerators.

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
- XIL_PIXEL_SEQUENTIAL type images

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		See Note 1
`xil_decompress`					See Note 2, 5
`xil_get_pixel`			x		See Note 4
`xil_lookup`	1 – 4 bands	1 – 4 bands	x	x	See Note 5
`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)*

	VIS Acceleration		Molecules		
Function	**8-bit**	**16-bit**	**Display**	**Other**	**Comments**
`xil_rescale`	1 – 4 bands		x	x	See Note 3, 5
`xil_rotate`	1 – 4 bands		x		See Note 6
`xil_scale`	1 – 4 bands	1 – 4 bands	x	x	See Note 5, 6
`xil_set_pixel`			x		See Note 4
`xil_set_value`	1 – 4 bands	1 – 4 bands	x		
`xil_subtract`	1 – 4 bands	1 – 4 bands	x		
`xil_subtract_const`	1 – 4 bands	1 – 4 bands	x		
`xil_subtract_from_const`	1 – 4 bands	1 – 4 bands	x		
`xil_tablewarp`			x		See Note 4
`xil_threshold`	1 – 4 bands	1 – 4 bands	x	x	See Note 3, 5
`xil_transpose`	1 – 4 bands	1 – 4 bands	x		
`xil_xor`	1 – 4 bands	1 – 4 bands	x		
`xil_xor_const`	1 – 4 bands	1 – 4 bands	x		

Notes to Table 7-1

1. `xil_convolve` is accelerated for 3×3, 5×5, and 7×7 kernels.
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` (JPEG compressed image or sequence and MPEG1 compressed sequence)
   - `xil_decompress` + `color_convert` + `scale` + `display` (MPEG1 compressed sequence)

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

   There are a few instances when decompression of streams will not be accelerated:

- Decompression of JPEG or MPEG1 streams that are not one contigous buffer in memory is not accelerated. For example, streams split up using xil_cis_put_bits_ptr might not be accelerated.
- If one frame in a stream is not accelerated for some reason (e.g., the decompressor encountered invalid data in the frame), subsequent frames in the stream will not be accelerated.
- Acceleration is also disabled if one tries to seek backwards or forwards in a stream.

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. See Figure 7-1.
   - The images must be 1-banded `XIL_BYTE` images.

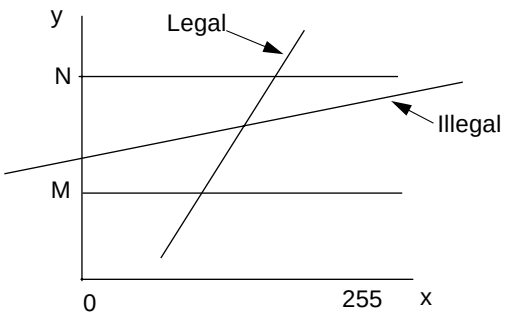


Figure 7-1 Creator Depth Cueing

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.

4. `xil_get_pixel`, `xil_set_pixel`, and `xil_tablewarp` are accelerated only as display molecules. For `xil_tablewarp`, only `interpolation = "nearest"` is accelerated.

5. All of the following molecules (other than a single xil function + display) are accelerated:
   - `xil_decompress + xil_colorconvert [+ display]` (for JPEG)
   - `xil_decompress + xil_colorconvert [+ display]` (for MPEG1)
   - `xil_decompress + xil_colorconvert + xil_scale + display` (for MPEG1)
   - `xil_threshold + xil_threshold [+ display] + [xil_cast] + display`
   - `xil_scale + xil_lookup + display` (the accelerated molecules are for 8- or 16-bit bilinear scale followed by 8-to-8 bit or 16-to-8 bit lookup)
   - xil_scale + xil_lookup + display (the accelerated molecules are 8 or 16-bit bilenear scale followed by 8-to-8 bit or 16-to-8 bit lookup).

6. There are some restrictions on the acceleration of `xil_affine`, `xil_scale`, and `xil_rotate`: `xil_affine` and `xil_rotate` are accelerated only for 1-, 3-, and 4-banded images with "nearest" interpolation, and only for 1- and

3-banded images with "bilinear" and "bicubic" interpolation. The restriction for `xil_scale` is that in the case of `interpolation = "general"` the size of the resampling kernels are limited to 8×8 for 8-bit images, and 8×4 for 16-bit images.

M64 Graphics Accelerator 8

This chapter describes how to change the display resolution on the M64 Graphics Accelerator.

You can change the M64 screen and associated graphics hardware through the `m64config` utility. Options are specified on the command line. The specified options can be stored in the `OWconfig` file to provide persistence of the options across window system sessions and system reboots.

Use the `m64config` utility to:

- Specify the video mode (screen resolution and refresh rate).
- Specify the `OWconfig` file to update.
- Reset all option values to their default values.
- Print the current values of all M64 options in the `OWconfig` file.
- Print the M64 hardware configuration.

8

Supported Screen Resolutions

Table 8-1 lists the screen resolutions that are supported by the M64 Graphics Accelerator.

Table 8-1 M64-supported Screen Resolutions

Screen Resolution	Vertical Refresh Rate	Description
1600 × 1000	76 Hz	Non-interlaced
1600 × 1000	66 Hz	Non-interlaced
1440 × 900	76 Hz	Non-interlaced
1280 × 1024	76 Hz	Non-interlaced
1280 × 1024	75 Hz	Non-interlaced
1280 × 1024	67 Hz	Non-interlaced
1280 × 1024	60 Hz	Non-interlaced
1280 × 800	76 Hz	Non-interlaced
1152 × 900	76 Hz	Non-interlaced
1152 × 900	66 Hz	Non-interlaced
1024 × 768	75 Hz	Non-interlaced
1024 × 768	70 Hz	Non-interlaced
1024 × 768	60 Hz	SVGA (non-interlaced)
800 × 600	75 Hz	Non-interlaced
768 × 575	50 Hz	PAL (interlaced)
640 × 480	60 Hz	NTSC (interlaced)

Changing the Screen Resolution Temporarily

This example changes the screen resolution temporarily. It can be used to verify that the monitor supports the specified resolution.

```
/usr/sbin/m64config -res video-mode try
```

Table 8-2 lists the *video-mode* options. You are given ten seconds to confirm the video mode by typing **y**.

Table 8-2 M64-screen Resolution Formats

Video Mode		
Width×Height×Rate	**Symbolic Name**	**Resolution**
1600x1000x76		1600 × 1000 at 76 Hz
1600x1000x66		1600 × 1000 at 66 Hz
1440x900x76		1440 × 900 at 76 Hz
1280x1024x76	1280	1280 × 1024 at 76 Hz
1280x1024x75		1280 × 1024 at 75 Hz
1280x1024x67		1280 × 1024 at 67 Hz
1280x1024x60		1280 × 1024 at 60 Hz
1280x800x76		1280 × 800 at 76 Hz
1152x900x76	1152	1152 × 900 at 76 Hz
1152x900x66		1152 × 900 at 66 Hz
1024x768x75		1024 × 768 at 75 Hz
1024x768x70		1024 × 768 at 70 Hz
1024x768x60	svga	1024 × 768 at 60 Hz
800x600x75		1024 × 600 at 75 Hz

Table 8-2 M64-screen Resolution Formats

Video Mode		
Width×Height×Rate	**Symbolic Name**	**Resolution**
`768x575x50i`	`pal`	768 × 575 at 50 Hz, interlaced
`640x480x60i`	`ntsc`	640 × 480 at 60 Hz, interlaced
	`none`	The video mode currently programmed for the device

For example, to try the 1152 × 900 resolution at 76 Hz, enter one of the following:

Example 1: This example uses the Width×Height×Rate format:

```
/usr/sbin/m64config -res 1152x900x76 try
```

Example 2: This example uses the symbolic name format:

```
/usr/sbin/m64config -res 1152 try
```

Printing the M64 Hardware Configuration

♦ **To print the M64 hardware configuration information, type:**

```
/usr/sbin/m64config -prconf
```

The following is a typical display of the hardware configuration information:

```
--- Hardware Configuration for /dev/fbs/m640 ---
ASIC: version 0x41004754
DAC: version 0x0
PROM: version 0x0
Card possible resolutions: 640x480x60, 800x600x75, 1024x768x60
    1024x768x70, 1024x768x75, 1280x1024x75, 1280x1024x76
    1280x1024x60, 1152x900x66, 1152x900x76, 1280x1024x67
    960x680x112S, 960x680x108S, 640x480x60i, 768x575x50i, 1280x800x76
    1440x900x76, 1600x1000x66, 1600x1000x76, vga, svga, 1152, 1280
    stereo, ntsc, pal
Monitor possible resolutions:  720x400x70, 720x400x88, 640x480x60
    640x480x67, 640x480x72, 640x480x75, 800x600x56, 800x600x60
    800x600x72, 800x600x75, 832x624x75, 1024x768x87, 1024x768x60
    1024x768x70, 1024x768x75, 1280x1024x75, 1280x1024x76, 1152x900x66
    1152x900x76, 1280x1024x67, 960x680x112S, vga, svga, 1152, 1280
    stereo
Current resolution setting: 1280x1024x76
Current depth: 8
```

For More Information

The examples in this chapter are only two of the simpler uses of the `m64config` utility. For more information on `m64config`, see the `m64config` man page.

8

Multiple Monitors on a System 9

This chapter describes how to use multiple monitors on a SPARCstation system. Skip this chapter if you do not run 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.

The procedures described in this chapter require some knowledge of 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 determine the system information and `sbus-probe-list`, type:**

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

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

```
% 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 you supply UNIX device file names for frame buffers 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 the `sbus-probe-list` EEPROM variable. 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 with an existing GX frame buffer, the `sbus-probe-list` also determines which is `/dev/cgsix0` and which is `/dev/cgsix1`.

For example, assume that the `sbus-probe-list` on a SPARCstation 10 system has the default value of `f0123` and that 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 chapter use possible device file names to refer to frame buffers. Substitute the device file name that is appropriate for your system.

Checking the Available Frame Buffers

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

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

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

```
 .
 .
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 – Executing the `dmesg` command may result in numerous messages. The system configuration messages shown below may not even be displayed. In this case, reboot your system. After rebooting, repeat the command shown previously.

9

Starting OpenWindows from the Console

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

endif
```

Running OpenWindows on Multiple Monitors

▼ To run multiple monitors with OpenWindows Version 3

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

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

2. **Verify that a device file already exists for the desired frame buffer by typing:**

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

If the device file already exists, a message similar to the one shown below is displayed. If so, skip Steps 3, 4, and 5 and go to Step 6. If the "not found" message is displayed, continue to Step 3.

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

3. **Become superuser and create a device file for your second frame buffer as follows (the second frame buffer is assumed to be** `cgsix1`**):**

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

This creates a device file for the second frame buffer.

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

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

A message similar to the following is displayed, indicating that the device file was 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 want to run by typing:**

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

Note – The order in which the devices are listed 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. 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 section if you know the order in which the SBus devices are addressed, or if you do not want to change the current order.

SBus Addresses

Two-slot SPARCstation systems, such as the 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 also no on-board frame buffer on the SPARCstation 10 and SPARCstation 20 systems. SBus address `f` is reserved for the CPU and should always be the first address in the `sbus-probe-list`. In 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 OpenBoot™ PROM. For the SPARCstation 10 and 20 systems, 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, if you install a frame buffer card into an SBus slot in SPARCclassic™, SPARCstation IPX, and SPARCstation LX systems, the system 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. To have the system 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 that a monitor is attached to the on-board frame buffer. Otherwise, the system will not recognize any other monitor regardless of polling order.

Changing the `sbus-probe-list`

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

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

▼ To change the `sbus-probe-list`

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.

The following is displayed:

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

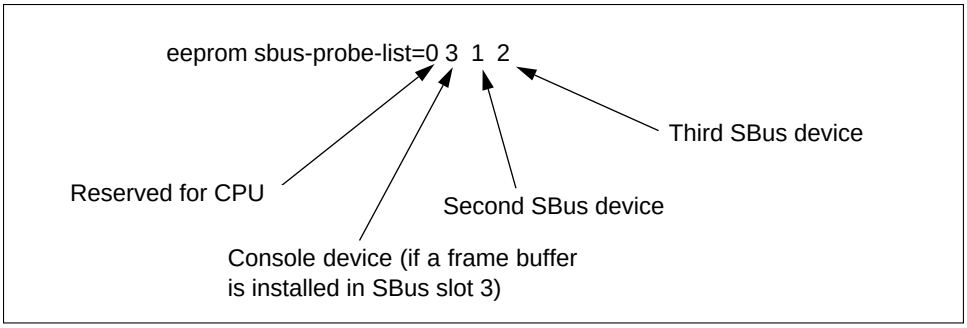


Figure 9-1 SBus Probe List Explanation

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. **Be sure a monitor is connected to the frame buffer identified as the console.**

3. **Reboot the system.**

Index

F

G

H

L

M

N

O

P

Z