Sirius Video Programming and Configuration Guide

Document Number 007-2238-003

CONTRIBUTORS

Written by Carolyn Curtis
Illustrated by Dan Young, Cheri Brown, and Carolyn Curtis
Edited by Christina Cary
Production by Julia Lin
Engineering contributions by Kirk Law, Scott Pritchett, Vince Uttley, Mohsen Hosseini, Ed Miszkiewicz, Paul Spencer, John Hallesy, Sam Gupta, Bharat Patel, and Gary Sanders
Cover design and illustration by Rob Aguilar, Rikk Carey, Dean Hodgkinson, Erik Lindholm, and Kay Maitz

Table 3-4, in Chapter 3, is derived from Thomas Porter and Tom Duff, “Compositing Digital Images,” published by the Association for Computing Machinery, 1984.

Sirius Video Programming and Configuration Guide
Document Number 007-2238-003

Contents

Figures

Tables

About This Guide

Sirius Video™ is a video option that provides IRIS® and POWER Onyx™ workstations equipped with RealityEngine™, RealityEngine²™, or VTX™ graphics with broadcast-quality video. Sirius Video can also be installed in CHALLENGE™, POWER Series™, and POWER CHALLENGE™ server configurations, making its broadcast-quality video a network resource.

Note: See the release notes for compatible versions of IRIX on various platforms.

For controlling videotape recorders, Sirius Video includes an on-board V-LAN™ transmitter. The transmitter sends high-level commands to videotape recorders that are equipped with receivers purchased separately from third parties.

The option consists of a board that fits into a VME slot in the chassis, a paddleboard, a breakout box, cables, software, and documentation.

Note: Sirius Video uses the Video Library™ (VL). VL device-independent calls and controls are explained in the *Digital Media Programming Guide*. The *Sirius Video Programming and Configuration Guide* is written on the presumption that you are familiar with the VL information in the *Digital Media Programming Guide*.

Structure of This Guide

This guide includes the following chapters and appendices:

Chapter 1 "Sirius Video Features and Capabilities" outlines the main components of Sirius Video, explains Sirius Video video formats and video handling, and introduces the V-LAN interface.

Chapter 2	"Programming Sirius Video" describes VL and Sirius Video specific controls for using Sirius Video to accomplish common specific tasks.
Chapter 3	"Sirius Video Blending and Keying" explains how to use VL and Sirius Video controls to blend images.
Chapter 4	"Controlling the General-Purpose Interface (GPI)" explains the Sirius Video device-dependent controls for the GPI port on the Sirius Video breakout box.
Chapter 5	"Controlling the V-LAN Interface" explains how to control other video devices using V-LAN commands with Sirius Video device-dependent controls.
Chapter 6	"Using Sirius Video Utility and Demonstration Programs" explains how to perform certain tasks with little or no programming using utility and demonstration programs included in the Sirius Video software.
Appendix A	"Hardware Specifications" summarizes technical specifications for Sirius Video.
Appendix B	"Configuring Sirius Video" gives sample configurations for various combinations of video equipment.
Appendix C	"Setting Up *vcp* for Your Video Hardware" describes connecting video equipment to the Sirius Video breakout box and using the control panel *vcp* to configure Sirius Video for the equipment.
Appendix D	"Sirius Video Color-Space Conversions" explains Sirius Video color spaces, the mathematical operations performed during conversions, and the implications of color space conversions.
Appendix E	"Example Programs" gives the parameters for some of the example programs included with the Video Library for Sirius Video.

An index completes this guide.

Conventions

In command syntax descriptions and examples, square brackets ([ ]) surrounding an argument indicate an optional argument. Variable parameters are in italics. Replace these variables with the appropriate string or value.

In text descriptions, IRIX™ filenames are in italics. The names of IRIS® keyboard keys are printed in boldface typewriter font and enclosed in angle brackets, such as <Enter> or <Esc>.

Messages and prompts that appear on-screen are shown in typewriter font. Entries that are to be typed exactly as shown are in boldface typewriter font.

Chapter 1

Sirius Video Features and Capabilities

Sirius Video is a video option for IRIS workstations equipped with RealityEngine, RealityEngine^2, or VTX graphics. It fully integrates broadcast-quality video with Silicon Graphics supercomputer graphics capabilities. Sirius Video can also be installed in CHALLENGE servers and POWER Series server configurations, making its broadcast-quality video a network resource.

The option utilizes one VME slot; input and output connectors are on a breakout box that can be mounted in a rack. Multiple Sirius Video boards can be operated in serial or parallel for multilayered video effects or video server capability.

Sirius Video supports real-time input and output of video in the full range of broadcast video formats. You can view graphics images from the workstation in low resolution, or capture video images in real time and view them on the workstation monitor. Sirius Video enables you to apply the full power of RealityEngine graphics to manipulate live video images.

Sirius Video enables you to blend graphics and frames from video in many ways, including alpha blending and chroma and luma keying. You can also generate pixel fades and wipes in real time from external alpha. You can blend video input with workstation-derived images for video output.

For controlling videotape recorders, Sirius Video includes an on-board V-LAN controller (transmitter) to send high-level VTR commands over a coaxial network to V-LAN receivers purchased from third parties.

Sirius Video is designed for the sophisticated video user in a professional or research environment. It fully utilizes all calls and controls in the Silicon Graphics Digital Media library, such as the Video Library, as well as controls that are native to Sirius Video only.

This chapter explains

- major components of Sirius Video
- video formats and video handling
- the V-LAN interface

Major Components of Sirius Video

Figure 1-1 diagrams how the Sirius Video expansion board (VO2) interacts with other workstation components.

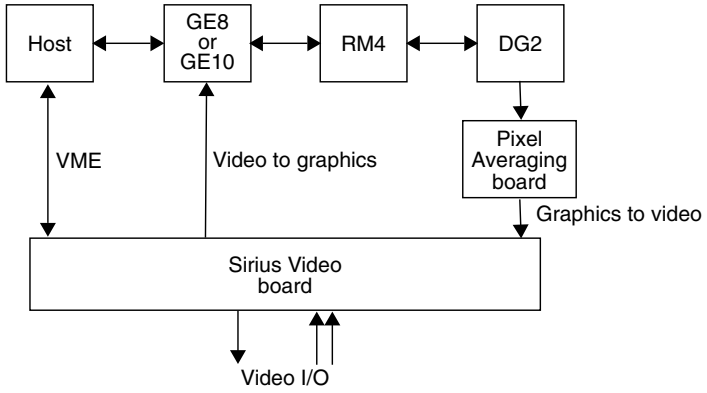

Figure 1-1 Sirius Video Top-Level Block Diagram

Figure 1-2 diagrams Sirius Video board functionality.

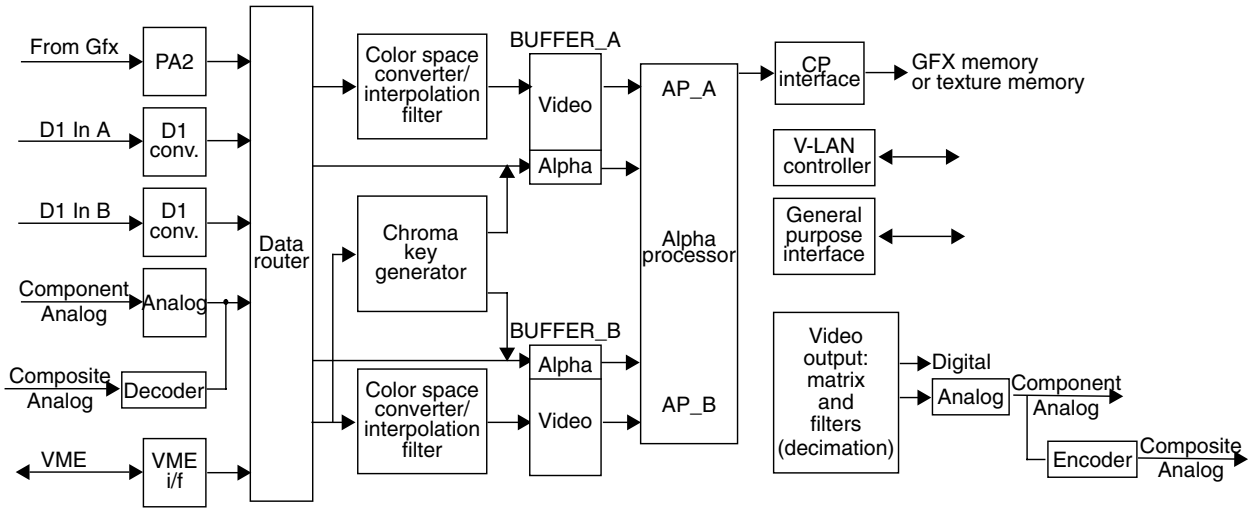

Figure 1-2 Sirius Video Board Architecture

The Sirius Video expansion board can be divided into four quadrants, as shown in Figure 1-3. Frame buffers serve as the quadrant boundaries. The frame rates and clock rates of each quadrant are fully independent, allowing the quadrants to have different inputs and outputs in different timings.

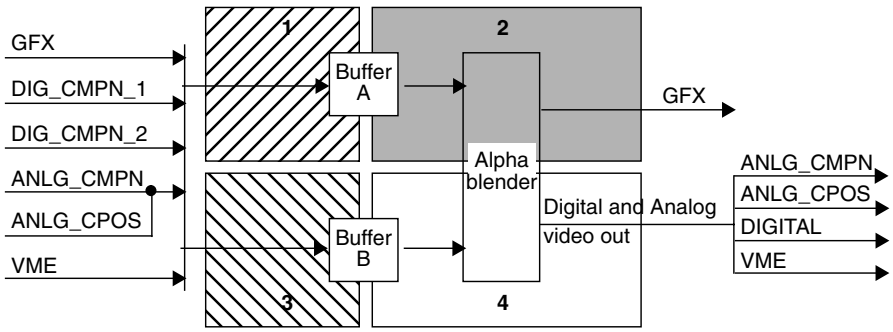

Figure 1-3 Sirius Video Board Quadrants

The rest of this section details the functionality of these system components:

- inputs and outputs
- digital video ports
- VME interface
- data router
- color-space converters
- frame buffers
- alpha processor and chroma key generator
- interpolation and decimation filters
- video-to-graphics (CP) interface

Inputs and Outputs

Sirius Video supports real-time input and output of video at field rates of 50 or 59.94 frames per second in the full range of video and Silicon Graphics graphics formats. The Sirius Video breakout box provides input to and output from the on-board field buffers, as summarized in Table 1-1.

Table 1-1 Sirius Video Inputs and Outputs

Input to Sirius Video Frame Buffers	Sirius Video Destination Frame Buffers
10-bit digital (4:4:4:4, 4:2:2:4, or 4:2:2) (two inputs)	10-bit digital (4:4:4:4, 4:2:2:4, or 4:2:2) (one output)
10-bit analog RGBa/YUVa	10-bit digital RGBa to GFX
10-bit digital RGBa from the graphics subsystem (GFX)	VME 8-bit or 10-bit/component
8-bit composite (NTSC, PAL, or S-VHS)	
VME 8-bit or 10-bit/component	

Video can be directed to more than one output at a time, making possible a broad range of processing and monitoring arrangements.

Digital Video Ports

The Sirius Video breakout box has two 10-bit digital video ports for equipment that complies with the CCIR 601 standard. Each port supports a parallel interface at 27 MHz and an optional serial interface (marketing code D4-SD1) at 270 MHz.

The ports can be configured for 4:4:4:4 or 4:2:2:4 dual-link mode; for 4:2:2 single-link mode, alpha is ignored.

Each port consists of two unidirectional interconnections, Link A and Link B:

- In 4:4:4:4 mode, Link A carries Y plus the U and V from even-numbered sample points; Link B carries alpha plus the U and V from odd-numbered sample points.
- In 4:2:2:4 mode, Link A carries Y plus the U and V from even-numbered sample points; Link B carries alpha only.

The video format selected determines Link A and Link B usage. For more information on the Link A and B connectors on the breakout box, see Table A-9 in Appendix A, “Technical Specifications.”

For more information, see the following standards, which contain provisions for video signals:

- CCIR 601-2: Encoding Parameters of Digital Television for Studios (4:2:2 component video signals, single link)
- ANSI/SMPTE 125M-1992: Television—Component Video Signal 4:2:2—Bit-Parallel Digital Interface
- SMPTE Recommended Practice (RP) 175-1993: Digital Interface for 4:4:4:4 Component Video Signals (Dual Link)
- SMPTE 259M, Television—10-Bit 4:2:2 Component and $4f_{sc}$ NTSC composite Digital Signals—Serial Digital Interface
- SMPTE RP 157-1990: Key Signals

VME Interface

The VME interface on the Sirius Video board initializes and configures the hardware and, in master mode, transfers video to and from system memory. A Sirius Video application can send live video to raster or texture memory in the graphics subsystem, or it can send the video to host memory over the VME bus. Regardless of input format, the video stream within Sirius Video is 4:4:4:4 RGBa/YUVa, 10 bits per component. See Appendix A, "Technical Specifications," for more details on this interface.

The format of the data sent over the VME bus is user-controlled; Table 2-4 in Chapter 2, "Programming Sirius Video," has details on packing formats.

Data Router

The data router subsystem performs crosspoint switching. The data router subsystem is comprised of four routers, one for each component (YUVa or RGBa). Any input source and its alpha can be sent to either frame buffer.

In data router mode, five input channels are multiplexed onto two separate output channels. Three of the five inputs can also go through a lookup table.

Color-Space Converters

Three color spaces are native to Sirius Video: RGBa, CCIR601, and full-range YUV (not compressed into the CCIR range). Any color space can be converted to any other color space on input or output. Specifying a format includes a color space; for example, to capture D1 data as RGB, use component digital format at the appropriate video source node and RGB at the memory drain node.

Color-space conversion is accomplished in hardware. Each of the two inputs to the frame buffers has a digital color space converter for YUV/RGB color-space conversion, which is a 3 x 3 matrix multiplier with programmable coefficients. The matrix processes 12 data bits and uses 10-bit coefficients and 23-bit precision internally. Lookup tables control overflow and underflow, rounding two's complement numbers.

Sirius Video can also maintain the color space of the input source throughout the video stream, rendering the digital processing of the video signal transparent.

Note: For more information on color-space conversion, see Appendix D, "Sirius Video Color-Space Conversions," later in this guide.

Frame Buffers

The two Sirius Video frame buffers are symmetrical, except that the alpha buffer of frame store A can be accessed independently of its corresponding video buffer, whereas the same is not true of the alpha buffer of frame store B. Each frame buffer is ten bits wide.

Each frame buffer stores four fields of R/R-Y, G/Y, B/B-Y, and alpha pixels. The four component field buffer memory is comprised of field memory chips, where each chip is organized as a 256 K deep FIFO with two data ports. Each port has its own dedicated control and clock signals. Each buffer has its own controller, which contains the logic that generates the signals controlling reads and writes to the buffers.

The four field buffers are configured as a ring and are accessed sequentially. A field buffer cannot be written to and read from in the same operation; a buffer's contents cannot be displayed while it is being written to. If the write and read rates differ substantially, then paired fields, or frames, are dropped or repeated as needed.

Alpha Processor and Chroma Key Generator

The alpha processor produces alpha blends from the inputs to the two alpha buffers (A and B). The alpha processor blends the video and rate/scan-converted graphics based on the values of the alpha specified for each channel. The alpha can be from an external source or sources or graphics alpha, or can be generated from the chroma key generator. Color or alphas can also be passed through to output without going through the blend function.

The chroma key generator extracts an alpha from any input channel; it can direct its output to the alpha frame buffers.

Interpolation and Decimation Filters

Digital filters aid in the conversion of video between 4:2:2 and 4:4:4 formats. The filters are 55-tap, fixed-coefficient, linear-phase half-band (low-pass) filters that can halve or double the sample rate.

When a 4:2:2 input signal is selected, interpolating filters on each chroma difference channel can be used to double the sampling rate of U and V. A line FIFO on the Y channel acts as a programmable delay element to compensate for the delays produced by the filtering. If the filters are bypassed, U and V are replicated, not interpolated.

When 4:2:2 output mode is selected, decimating filters can be used to halve the sampling rate of U and V. The same types of filters are used to provide two times over-sampled digital data to the digital-to-analog converters for component output.

CP Interface

The CP interface moves data between Sirius Video and the Geometry Engine® (GE) board in the graphics subsystem. The CP interface consists of a controller, a matrix multiplier, dedicated lookup tables, a frame buffer, and data drivers. Bandwidth is 40 MHz.

The Sirius Video CP interface can convert data from YUVa to RGBa and format the data for the raster or texture memory. The formatted video data can then be sent over the 48-bit 33 MHz CP bus to graphics, packed as follows:

- RGBa_12 raster data
- RGBa_5 texture data (R = 5 bits, G = 6 bits, B = 5 bits)
- RGBa_4 texture data (R = 4 bits, G = 4 bits, B = 4 bits)
- RGBa_8 texture data (R = 8 bits, G = 8 bits, B = 8 bits)

Video Formats and Video Handling

Table 1-2 video input and output formats that Sirius Video supports.

Table 1-2 Sirius Video Video Formats

Port	Formats Supported
Input	
Serial/parallel digital (2 pairs)	CCIR 601, SMPTE dual link; optional serial
Analog component	I/O
Analog composite	RGBa, YUVa, PrYPbA
	NTSC, PAL, S-Video
Output	
Serial/parallel digital (1 pair)	CCIR 601, SMPTE dual link; optional serial
Analog component	I/O
Analog composite	RGBa, YUVa, PrYPbA
	NTSC, PAL, S-Video

Sirius Video displays full-size video windows in NTSC (646 x 486 or 720 x 486) and PAL (768 x 576 or 720 x 576) formats.

This section explains aspects of Sirius Video workstation graphics and video manipulation:

- scan conversion/pixel averaging
- video-to-graphics conversion
- live video
- blending and keying

Scan Conversion/Pixel Averaging

On graphics systems, graphics and its alpha can be captured and converted to several sizes of 525 and 625 video. Output is always to a standard video size; the grab area can be changed. The captured graphics can also be passed through the system without scan conversion.

In pixel averaging, Sirius Video performs a resolution reduction of one image into another using filtering and decimation. A 1280 x 972-line portion

of a workstation image can be converted to a 525-timing image (646 x 486) in real time by a vertical decimation of two and a combination of horizontal decimation and zooming. If 1280x1024 is converted to 646x486, the result is filtered down to 646x512, and then further decimated to 646x486.Likewise, a workstation image can be converted to a 625-timing image (768 x 576) in real time by a three-fifths decimation horizontally and vertically.

Using filtering, Sirius Video can convert an arbitrarily sized region of the workstation display to a video-sized raster. Some loss of sharpness in image quality might occur.

On Sirius Video installed in servers, scan conversion is not available.

If Sirius Video and Multi-Channel Option are installed on the same graphics pipeline of your system, Sirius Video is not capable of live graphics to video, either directly or pixel-averaged. (You can, however, snap the workstation display, save it as a file, and send the file to video.) Live video input is possible if *hinv* returns this report:

```
Sirius Video: unit 0 revision 4 on bus 0 with CPI BOB options
```

Live video output and input are possible if *hinv* returns this report:

```
Sirius Video: unit 0 revision 4 on bus 0 with DGI CPI BOB options
```

Video-to-Graphics Conversion

Sirius Video processes video pixels by software calls to the Silicon Graphics Graphics Library™ and other compatible libraries allowing, for example, the use of video for texturing.

A Sirius Video application uses calls to the IRIS Video Library (VL), the IRIS Graphics Library™, or other compatible libraries of custom routines to generate graphics effects with the video, including windows, zooms, and pans.

The interface to the graphics subsystem operates in field or frame mode. An application can select the number of bits (48, 32, or 16) formatted per pixel.

Data can be sent to memory in two modes:

- raster emory: field (50 or 59.94 fields per second)
- texture memory: field (50 or 59.94 fields per second) or frame (25 or 29.97 frames per second)

Blending and Keying

Sirius Video supports alpha blending of any two inputs in real time. Blending can be based on graphics-buffer alpha planes, external alpha values, or output of the on-board alpha/key generator.

Path A alpha values can come from

- graphics buffer alpha planes
- VME
- any input alpha source

Path B alpha values can come from

- alpha channel of the source driving path B
- alpha/key generator output derived from the source driving path B

An external 10-bit alpha key can be used for blending; the alpha processor generates an output or destination alpha from any pair of frame buffer alpha inputs.

Sirius Video fully supports chroma and luma keys. Source input is processed in the 3 x 3 matrix multiplier; editable lookup tables can then be used to manipulate the keys.

Component outputs from lookup tables are used to produce an alpha/key in the alpha/key generator. Edges on chroma or alpha keys can be softened.

Note: Chapter 3, “Sirius Video Blending and Keying,” is a complete guide to using Sirius Video blending and keying capabilities.

The V-LAN Interface

V-LAN is a video device network protocol designed and implemented by Videomedia® Inc., to provide reliable, frame-accurate control of professional and industrial-quality video equipment. The Sirius Video on-board V-LAN controller (transmitter) handles high-level VTR commands and distributes them over a coaxial network.

To use the V-LAN interface, purchase a receiver from Videomedia that is appropriate for the VTR equipment you have, and attach it to the V-LAN network connector on the Sirius Video breakout box. You can attach as many as 31 V-LAN receivers, each controlling a particular class of VTR.

The Sirius Video on-board V-LAN controller reduces the cost of tape control. Since you do not need to purchase a V-LAN controller unit, the single coaxial connection also has the advantage of replacing the tangle of RS-422 cables typically found in postproduction environments.

The Sirius Video software implements V-LAN commands as controls. Chapter 5, “Controlling the V-LAN Interface,” later in this guide contains complete information for using V-LAN with Sirius Video.

Note: Other tape control devices can work with Sirius Video by interfacing triggered events via the GPI connectors on the Sirius Video breakout box.

Chapter 2

Programming Sirius Video

The Video Library (VL) contains generic video tools, including simple tools for importing and exporting digital data to and from current and future Silicon Graphics video products, as well as to and from third-party video devices that adhere to the Silicon Graphics architectural model for video equipment.

For example, VL calls enable you to blend computer-generated graphics with frames from videotape or any video source, to present video in a window on the workstation screen, and to digitize and output video data.

This chapter explains

- VL basics for Sirius Video
- Sirius Video controls
- source node controls
- drain node controls

  Note: Blender node controls are discussed in Chapter 3, "Sirius Video Blending and Keying."
- using filters
- Sirius Video events and triggering
- passing video data through the graphics subsystem

Note: If Sirius Video and Multi-Channel Option are installed on the same graphics pipeline of your system, Sirius Video is not capable of live graphics to video, either directly or pixel-averaged. (You can, however, snap the workstation display, save it as a file, and send the file to video.) Live video input is possible if *hinv* returns this report:

```
Sirius Video: unit 0 revision 4 on bus 0 with CPI BOB options
```

Live video output and input are possible if *hinv* returns this report:

```
Sirius Video: unit 0 revision 4 on bus 0 with DGI CPI BOB options
```

VL Basics for Sirius Video

To run VL, you must

- install *dmedia_dev* option
- link with *libvl*
- include *vl/vl.h* and *vl/dev_sirius.h*

The client library for VL is */usr/lib/libvl.so*. The header files for the VL are in */usr/include/dmedia/vl*; the main file is *vl.h*. This file contains the main definition of the VL API and controls that are common across all hardware.

Note: When building a VL-based program, you must add *-lvl* to the linking command.

For more information, see Chapter 12, "Getting Started With the Video Library," in the *Digital Media Programming Guide* (007-1799-040).

This section explains

- central VL concepts
- VL object classes
- VL nodes for Sirius Video
- VL data transfer functions

Central VL Concepts

The two central concepts for VL are

- *path*: an abstraction for a way of moving data around
- *node*: an endpoint of the path

The basic nodes are a *source* (such as a VTR) and a *drain* (such as graphics). Figure 2-1 diagrams the simplest VL path, with one of each of these two nodes.

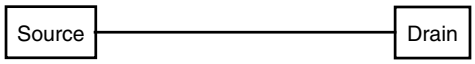

Figure 2-1 Simple VL Path

The two other types of node besides source and drain are *device* node and *internal* node. Figure 2-2 diagrams a more complex path, a path for a simple blend, which includes two source nodes, a drain node, and an internal (blender) node.

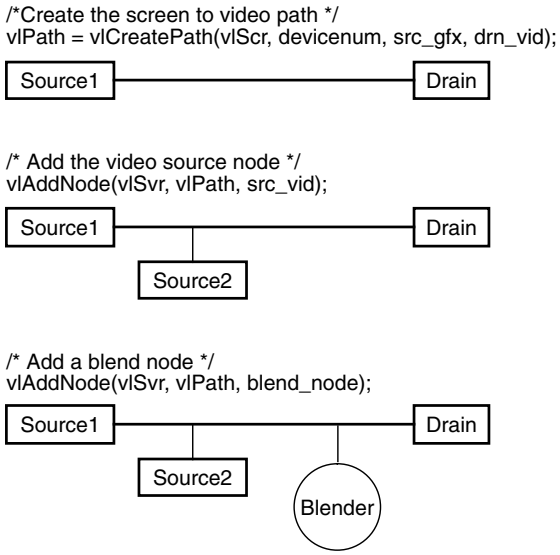

Figure 2-2 VL Blending

VL Object Classes

The VL recognizes five classes of objects:

- *devices*, each including sets of nodes
- *nodes*: sources, drains, and internal nodes (as discussed in the preceding section)
- *paths*, connecting sources and drains (as discussed in the preceding section)
- *buffers*, for sending and receiving frame data to and from host memory

  The VL buffers are implemented as ring buffers; each maintains a pointer, a size, and pointers to the head (oldest) and tail (newest) valid data.
- *controls*, or parameters that modify how data flows through nodes; for example:
  - video device parameters, such as blanking width, gamma value, horizontal phase, sync source
  - video data capture parameters
  - blending parameters

VL controls fall into two categories:

- *device-independent* (prefix VL_), which can be used by several Silicon Graphics video products

  For details of the device-independent controls, see Chapter 12, "Getting Started With the Video Library," of the *Digital Media Programming Guide*.
- *device-dependent* (prefix VL_SIR_ for Sirius Video), specific to a particular video device

  Both types of VL controls are explained in this chapter with respect to their usage with Sirius Video.

Note: For more detail on VL classes, see Chapter 12, "Getting Started With the Video Library," of the *Digital Media Programming Guide*.

VL Nodes for Sirius Video

Use **vlGetNode()** to specify nodes. This call returns the node's handle. Its function prototype is:

```
VLNode vlGetNode(VLServer vlServer, int type, int kind, int number)
```

In this prototype, variables are as follows:

VLNode Handle for the node, used when setting controls or setting up paths.

vlServer Names the server (as returned by *vlOpenVideo*()).

type Specifies the type of node:

- VL_SRC: source, such as a tapdeck connected to the Sirius Video breakout box
- VL_DRN: drain, such as a window on the workstation graphics monitor
- VL_DEVICE: Sirius Video global control, such as trigger, GPI, sync, or genlock voltage level, or default source;Table 2-1 summarizes the values for this type

  Note: If you are using VL_DEVICE, the *kind* (see below) should be set to 0.
- VL_INTERNAL: internal node, such as the blend node and chroma key generator node

kind Specifies the kind of node.

If *type* is VL_SRC, *kind* values can be

- VL_VIDEO: connection to a video device equipment; for example, a video tape deck or camera

  If the *type* is VL_SRC, values can be

  - SIR_SRC_DIGITAL_VIDEO_1 (source node)

  - SIR_SRC_DIGITAL_VIDEO_2 (source node)

  - SIR_SRC_ANALOG_VIDEO (source node)

  These values correspond to input sockets on the Sirius Video breakout box.

- VL_GFX: system graphics

  Note: Sirius Video uses VL_GFX instead of VL_SCREEN, which is referred to in the section on the Video Library in the *Digital Media Programming Guide*.

- VL_MEM: region of workstation memory
- VL_ANY: use any available node (factory setting)

  This setting accommodates settings in the control panel *vcp*, in which input and output parameters can easily changed.

If *type* is VL_DRN, *kind* values can be

- VL_VIDEO: connection to a video device equipment; for example, a video tape deck or camera
- VL_GFX: system graphics
- VL_MEM: region of workstation memory
- VL_TEXTURE: texture RAM

If *type* is VL_INTERNAL, *kind* values can be

- VL_BLENDER: a blender node (internal node)
- VL_KEYGEN: key generator node (internal node)

number Number of the node in cases of two or more identical nodes, such as two video source nodes. The default value for all *kind*s is 0, except VL_VIDEO, which must be SIR_SRC_DIGITAL_VIDEO_1, SIR_SRC_DIGITAL_VIDEO_1, or SIR_SRC_ANALOG_VIDEO.

Besides being a *kind* value, VL_ANY is also used as a value for *number*.

The drain nodes VL_TEXTURE and VL_GFX use the same port and cannot both be active at the same time; likewise, the drain nodes VL_VIDEO and VL_MEM use the same port and cannot be active at the same time. Source nodes, however, are independent of each other. Figure 2-3 diagrams this configuration.

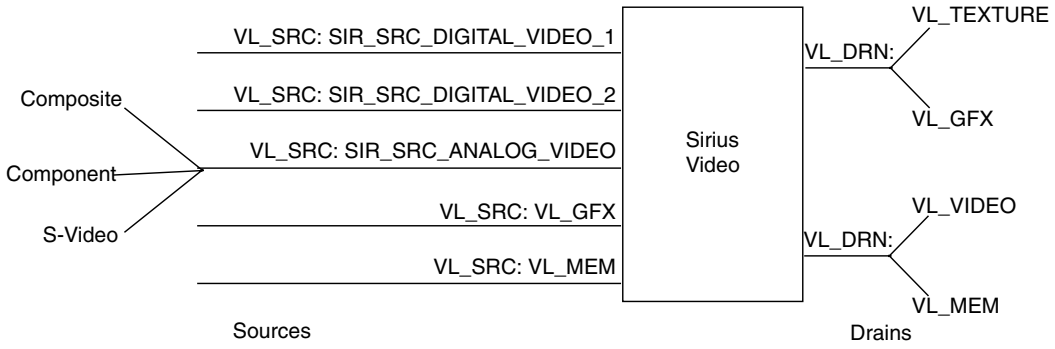

Figure 2-3 Sirius Video Source and Drain Nodes

The following fragment creates a digital video input 1 source node and a graphics drain node, and creates the path.

```
src = vlGetNode(svr, VL_SRC, VL_VIDEO SIR_SRC_DIGITAL_VIDEO_1, VL_ANY);
drn = vlGetNode(svr, VL_DRN, VL_GFX, VL_ANY);
if((path = vlCreatePath(svr, VL_ANY, src, drn)) < 0)
     exit(1);
vlSetupPaths(svr, (VLPathList)&path, 1, VL_SHARE, VL_SHARE);
```

Note: For details on *vlSetupPaths*(), see "Displaying Video Data Onscreen," in Chapter 12, "Getting Started With the Video Library," of the *Digital Media Programming Guide*.

The following fragment illustrates the use of VL_ANY as the default node *kind*; it allows a program to accept input from whatever video equipment is specified in the video control panel *vcp*.

```
src = vlGetNode(svr, VL_SRC, VL_VIDEO, VL_ANY);
```

It is possible that the actual source or drain node can vary from the source or drain node specified. To discover what the source node is, use the control VL_DEFAULT_SOURCE with *vlGetControl*() after getting the node handle the normal way. For example:

```
vlGetControl(svr, path, VL_ANY, VL_DEFAULT_SOURCE, &ctrlval);
nodehandle = vlGetNode(svr, VL_SRC, VL_VIDEO, ctrlval.intVal);
```

In the second line above, the last argument is a struct that gets the value.

The following fragment adds a video drain node:

```
VLServer svr;
VLPath path;
VLNode src;
VLNode drn;
VLControlValue timing,format;

drn = vlGetNode(svr, VL_DRN, VL_VIDEO, VL_ANY); /*Get a video drain node */
vlAddNode(svr, path, drn); /* Add node to the exisiting path */
```

After nodes are specified, the video timing (for example, 525 or CCIR 525) and format (for example S-Video or digital component) must be specified via controls (*vlSetControl*()). Controls for each node are given in "Source Node Controls" and "Drain Node Controls" later in this chapter and are summarized in Table 2-3.

VL Data Transfer Functions

This section presents a brief summary of VL syntax elements, data transfer categories, and the basic steps of creating an application. For details, please see Chapter 12, "Getting Started With the Video Library," in the *Digital Media Programming Guide*.

VL syntax elements are as follows:

- VL types and constants begin with uppercase VL; for example, VLServer
- VL functions begin with lowercase vl; for example, *vlOpenVideo*()

VL data transfers fall into two categories:

- transfers involving memory (video to memory, memory to video), which require setting up a ring buffer
- transfers not involving memory (such as video to screen and graphics to video), which do not require a ring buffer

For the two categories of data transfer, based on the VL programming model, the process of creating a VL application consists of these steps:

1. opening a connection to the video daemon (*vlOpenVideo*())
2. specifying nodes on the data path (*vlGetNode*())
3. creating the path (*vlCreatePath*())
4. optional step: adding more connections to a path (*vlAddNode*())
5. setting up the hardware for the path (*vlSetupPaths*())
6. specifying path-related events to be captured (*vlSelectEvents*())
7. setting input and output parameters (controls) for the nodes on the path (*vlSetControl*()); video format and timing must be specified
8. transfers involving memory: creating a ring buffer to hold data for memory transfers (*vlGetTransferSize*(), *vlCreateBuffer*())
9. transfers involving memory: registering the buffer (*vlRegisterBuffer*())
    - starting the data transfer (*vlBeginTransfer*())
10. transfers involving memory: getting the data (*vlGetNextValid*() or *vlGetLatestValid*(), *vlGetActiveRegion*(), *vlPutFree*()) to manipulate frame data
11. cleanup (*vlEndTransfer*(), *vlDeregisterBuffer*(), *vlDestroyPath*(), *vlDestroyBuffer*(), *vlCloseVideo*())

For details on these functions, see Chapter 12, “Getting Started With the Video Library,” in the *Digital Media Programming Guide*.

Sirius Video Controls

To determine the available devices (that is, video options in the workstation, such as Sirius Video) and the nodes available on them, run *vlinfo*. To determine possible controls for each device, run

```
vlinfo -l
```

To set controls for Sirius Video nodes, use *vlSetControl*(). The following example sets video format and timing on a node:

```
timing.intVal = VL_TIMING_525_CCIR601;
format.intVal = VL_FORMAT_RGB;

if (vlSetControl(svr, path, drn, VL_TIMING, &timing) <0)
{
     vlPerror("VlSetControl:TIMING");
     exit(1);
}
if (vlSetControl(svr, path, drn, VL_FORMAT, &format) <0)
{
     vlPerror("VlSetControl:FORMAT");
     exit(1);
 }
```

It is possible that the actual source or drain node can vary from the source or drain node specified. To discover what the source node is, use the control VL_DEFAULT_SOURCE with *vlGetControl*() after getting the node handle the normal way. For example:

```
vlGetControl(svr, path, VL_ANY, VL_DEFAULT_SOURCE, &ctrlval);
nodehandle = vlGetNode(svr, VL_SRC, VL_VIDEO, ctrlval.intVal);
```

For details on *vlSetControl*() and *vlGetControl*(), see Chapter 12, "Getting Started With the Video Library," in the *Digital Media Programming Guide*.

Tables in this section summarize

- device-global controls for Sirius Video
- controls for Sirius Video nodes
- control values and uses

Table 2-1 summarizes the device-global controls for Sirius Video.

Table 2-1 Device Controls for Sirius Video

Control	Values	Use
VL_DEFAULT_SOURCE	SIR_SRC_DIGITAL_VIDEO_1 SIR_SRC_DIGITAL_VIDEO_2 SIR_SRC_ANALOG_VIDEO SIR_SRC_GRAPHICS SIR_SRC_MEMORY	Determine the source node selected by VL when VL_ANY has been specified as the source node
VL_SIR_SYNC_LEVEL	SIR_SYNC_LEVEL_VIDEO SIR_SYNC_LEVEL_TTL	Sets voltage level at the **SYNC** connector (under **GBRA/YUVA COMPONENT**): 1 V peak-to-peak 4 V peak-to-peak
VL_SIR_GENLOCK_LEVEL	SIR_SYNC_LEVEL_VIDEO SIR_SYNC_LEVEL_TTL	Sets voltage level at the **GENLOCK IN** connector (under **SYNC**): 1 V peak-to-peak 4 V peak-to-peak
VL_SIR_TRIGGER	VL_SIR_TRIGGER_VLAN VL_SIR_TRIGGER_GPI_1 VL_SIR_TRIGGER_GPI_2 VL_SIR_TRIGGER_NONE	Determines whether triggered transfers are triggered on V-LAN, one of two GPI signals, or neither
VL_SIR_VLAN_NODE	Node ID	Associates V-LAN with the node it is genlocked to: a video source or a video drain
VL_SIR_VLAN_CMD	ASCII string containing V-LAN command	Sends V-LAN command string to V-LAN transmitter
VL_SIR_GPI_CHAN1_MODE VL_SIR_GPI_CHAN2_MODE		Determines the voltage expected on the GPI (General-Purpose Interface) port on the breakout box:
	VL_SIR_GPI_ACTIVE_LOW	The negative pin is disconnected and internally pulled up. Input is triggered by grounding the positive pin.
	VL_SIR_GPI_ACTIVE_HIGH	The negative pin is disconnected and internally grounded. Input is triggered by applying power to the positive pin.
VL_SIR_GPI_OUT_CHAN1 VL_SIR_GPI_OUT_CHAN2	(0,1): 0 = off, 1 = on	Provides trigger information to external devices

Table 2-2 summarizes Sirius Video specific controls and device-independent controls as they apply to Sirius Video.

Table 2-2 Controls for Sirius Video Nodes

Control	Source Nodes				Drain Node			
	Digital Video 1, 2	Analog Video	Graphics	Memory	Video	Graphics	Memory	Texture
VL_BRIGHTNESS		X						
VL_CAP_TYPE				X			X	X
VL_CONTRAST		X						
VL_FORMAT	X	X	X	X	X	X	X	X
VL_H_PHASE	X	X			X			
VL_HUE		X						
VL_OFFSET (read-only)	X	X	X	X	X	X	X	X
VL_ORIGIN			X					
VL_PACKING				X			X	X
VL_RATE (read-only)				X			X	
VL_SATURATION		X						
VL_SIR_ALPHA_GAIN		X			X			
VL_SIR_ALPHA_OFFSET		X						
VL_SIR_AUTO_GAIN_CONTROL		X						
VL_SIR_BLUE_GAIN		X			X			
VL_SIR_BLUE_OFFSET		X						
VL_SIR_FIELD_DOMINANCE	X	X			X			
VL_SIR_FILTER	X			X	X		X	
VL_SIR_GFX_FLICKER			X					
VL_SIR_GFX_SIZE			X					

Table 2-2 (continued) Controls for Sirius Video Nodes

Control	Source Nodes				Drain Node			
	Digital Video 1, 2	Analog Video	Graphics	Memory	Video	Graphics	Memory	Texture
VL_SIR_GREEN_GAIN		X			X			
VL_SIR_GREEN_OFFSET		X						
VL_SIR_H_PHASE_ALPHA	X	X						
VL_SIR_LINK_DELAY_A VL_SIR_LINK_DELAY_B	X X							
VL_SIR_RED_GAIN		X			X			
VL_SIR_RED_OFFSET		X						
VL_SIZE (read-only)	X	X	X	X	X	X	X	X
VL_SYNC					X			
VL_SYNC_SOURCE		X			X			
VL_TIMING	X	X	X	X	X	X	X	X
VL_ZOOM (read-only)	X	X		X	X	X	X	X

Table 2-3 summarizes the values and uses of controls for Sirius Video.

Table 2-3 Sirius Video Control Values and Uses

Control	Values or Range	Use
VL_BRIGHTNESS	(.5, 1,5)	Sets brightness level for composite or YC (S-Video)
VL_CAP_TYPE	Memory or texture drain node: VL_CAPTURE_NONINTERLEAVED VL_CAPTURE_INTERLEAVED Memory source is read-only: VL_CAPTURE_NONINTERLEAVED	Selects type of frame(s) or field(s) to capture
VL_CONTRAST	(0.0, 2.0)	Sets contrast level for composite or YC (S-Video)

Table 2-3 (continued) Sirius Video Control Values and Uses

Control	Values or Range	Use
VL_FORMAT		Sets video format in or out:
	VL_FORMAT_RGB (graphics source or drain read-only)	RGB
	VL_FORMAT_BETACAM	Betacam
	VL_FORMAT_MII	MII
	VL_FORMAT_SMPTE_YUV	SMPTE YUV
	VL_FORMAT_COMPOSITE	Composite
	VL_FORMAT_SVIDEO	S-Video
	VL_FORMAT_DIGITAL_COMPONENT	Parallel 4:2:2:4
	VL_FORMAT_DIGITAL_COMPONENT_SERIAL	Serial 4:2:2:4
	VL_FORMAT_DIGITAL_COMPONENT_DUAL	Parallel 4:4:4:4
	VL_FORMAT_DIGITAL_COMPONENT_DUAL_SERIAL	Serial 4:4:4:4
VL_H_PHASE	-32 to +32 pixels	Sets horizontal phase
VL_HUE	-45 to +44 degrees	Sets hue
VL_OFFSET	Read-only; currently set to 0	Sets offset: on video nodes, the offset to the active region of the video; on all other nodes, the offset within the video
VL_ORIGIN	Coordinates; for graphics drain, read-only: 0,0	Sets upper left corner of image in graphics source (usually a window); the offset within the node
VL_PACKING	See Table 2-4 for packing format contents	Sets packing format for memory source or drain node
	VL_PACKING_RGBA_8 (default, memory source, or memory drain)	
	VL_PACKING_RGB_8	
	VL_PACKING_YVYU_422_8	
	VL_PACKING_YUV_444_8	
	VL_PACKING_YUVA_4444_8	
	VL_PACKING_ABGR_8	
	VL_PACKING_AUYV_8	
	VL_PACKING_A_2_BGR_10	
	VL_PACKING_A_2_UYV_10	
	VL_PACKING_AYU_AYV_10	
	VL_SIR_TEX_PACK_RGB_5 (texture node factory setting)	Sets packing format for texture node
	VL_SIR_TEX_PACK_RGBA_4	
	VL_SIR_TEX_PACK_RGBA_8	

Table 2-3 (continued) Sirius Video Control Values and Uses

Control	Values or Range	Use
VL_RATE	Read-only; 30 or 60, 25 or 50	Sets transfer rate: fields or frames, depending on the capture type as specified with VL_CAP_TYPE
VL_SATURATION	(0.0, 2.0)	Sets saturation
VL_SIGNAL	VL_SIGNAL_BLACK VL_SIGNAL_REAL_IMAGE Note: Do not use VL_SIGNAL_NOTHING; it returns VLValueOutOfRange.	Sets the video output to black or real image for analog and digital outputs See also VL_SIR_ANALOG_DRAIN_BLANK_ENABLE and VL_SIR_DIGITAL_DRAIN_BLANK_ENABLE
VL_SIR_ANALOG_DRAIN_BLANK_ENABLE	VL_SIGNAL_BLACK VL_SIGNAL_REAL_IMAGE	Sets the video output to black or real image for analog outputs only, such as to disable analog output when running digital out; see "Setting Up the Output (Drain)," in Appendix C, "Setting Up Sirius Video for Your Video Hardware," for more information
VL_SIR_AUTO_GAIN_CONTROL	(0,1): 0 = off, 1 = on	For composite or YC (S-Video) source, sets automatic gain adjustment
VL_SIR_DIGITAL_DRAIN_BLANK_ENABLE	VL_SIGNAL_BLACK VL_SIGNAL_REAL_IMAGE	Sets the video output to black or real image for digital outputs only

Table 2-3 (continued) Sirius Video Control Values and Uses

Control	Values or Range	Use
VL_SIR_FIELD_DOMINANCE	VL_SIR_F1_IS_DOMINANT VL_SIR_F2_IS_DOMINANT Note: Frames that are output are deinterlaced differently depending on the choice of output field dominance. Deinterlacing is specified in the application; see "Setting Up Analog Source Video," in Appendix C, "Setting Up Sirius Video for Your Video Hardware" for details.	Specifies whether edit occurs on nominal video field boundary (field 1) or on intervening field boundary (field 2); for more information, see "Setting Up Analog Source Video," in Appendix C, "Setting Up Sirius Video for Your Video Hardware"
VL_SIR_FILTER	(0,1): 0 = off, 1 = on	Activates chroma-interpolating filter, converts 4:2:2:4 to 4:4:4:4 on input; converts 4:4:4:4 to 4:2:2:4 on output
VL_SIR_GFX_SIZE	Pixel units	Sets size of graphics on a pixel-by-pixel basis
VL_SIR_GFX_FLICKER	(0,1): 0 = off, 1 = on; default = 0	Reduces flicker in graphics
VL_SIR_H_PHASE_ALPHA	-32 to +32 pixels	Sets horizontal phase on the separate alpha link
VL_SIR_LINK_DELAY_A VL_SIR_LINK_DELAY_B	0 to 3 pixels	When using dual link (two cables) for 4:4:4:4, sets delay to compensate for differing delay on the other link
VL_SIR_RED_GAIN VL_SIR_GREEN_GAIN VL_SIR_BLUE_GAIN VL_SIR_ALPHA_GAIN	(0.0,2.0)	Sets gain for separate inputs or outputs
VL_SIR_RED_OFFSET VL_SIR_GREEN_OFFSET VL_SIR_BLUE_OFFSET VL_SIR_ALPHA_OFFSET	(-1,+1)	Sets offset for separate inputs or outputs

Table 2-3 (continued) Sirius Video Control Values and Uses

Control	Values or Range	Use
VL_SIR_SYNC_LEVEL	VL_SIR_SYNC_LEVEL_VIDEO VL_SIR_SYNC_LEVEL_TTL	Sets voltage level at the **SYNC** connector (under **GBRA/YUVA COMPONENT**) for video out only: 1 V peak-to-peak 4 V peak-to-peak
VL_SIZE	Coordinates (read-only)	Reads coordinates from input
VL_SYNC	VL_SYNC_INTERNAL VL_SYNC_GENLOCK	Sets sync mode for analog video source or drain; on source, this is set to VL_SYNC_GENLOCK
VL_SYNC_SOURCE	VL_SIR_SYNC_HOUSE VL_SIR_SYNC_GREEN VL_SIR_SYNC_DIGITAL_1 VL_SIR_SYNC_DIGITAL_2 VL_SIR_SYNC_COMPOSITE VL_SIR_SYNC_COMPONENT VL_SIR_SYNC_YC	Sets sync source for analog video source or drain
VL_TIMING		Sets or gets video timing
		Analog source or drain:
	VL_TIMING_525_SQ_PIX	12.27 MHz, 646 x 486
	VL_TIMING_625_SQ_PIX	14.75 MHz, 768 x 576
		Analog or digital source or drain:
	VL_TIMING_525_CCIR601	13.50 MHz, 720 x 486
	VL_TIMING_625_CCIR601	13.50 MHz, 720 x 576
VL_ZOOM	Zoom factor (read-only)	Reads zoom factor of video stream

Note: Blending and keying controls for Sirius Video are explained in Chapter 3, “Sirius Video Blending and Keying,” later in this guide. Device-independent controls are discussed in the chapter on VL Blending in Part III of the *Digital Media Programming Guide.*

Table 2-4 summarizes packing types and their sizes and formats for Sirius Video.

Table 2-4 Sirius Video Packing Type Sizes and Formats

Type	Bits per Pixel	Format MSB----------LSB
VL_PACKING_RGBA_8	32	$AAAAAAAA_0$ $BBBBBBBB_0$ $GGGGGGGG_0$ $RRRRRRRR_0$
VL_PACKING_RGB_8	32	$XXXXXXXX_0$ $BBBBBBBB_0$ $GGGGGGGG_0$ $RRRRRRRR_0$
VL_PACKING_YVYU_422_8	16	$UUUUUUUU_0$ $YYYYYYYY_0$ $VVVVVVVV_0$ $YYYYYYYY_1$
VL_PACKING_YUV_444_8	32	$XXXXXXXX_0$ $UUUUUUUU_0$ $YYYYYYYY_0$ $VVVVVVVV_0$
VL_PACKING_YUVA_4444_8	32	$AAAAAAAA_0$ $UUUUUUUU_0$ $YYYYYYYY_0$ $VVVVVVVV_0$
VL_PACKING_ABGR_8	32	$RRRRRRRR_0$ $GGGGGGGG_0$ $BBBBBBBB_0$ $AAAAAAAA_0$
VL_PACKING_AUYV_8	32	$VVVVVVVV_0$ $YYYYYYYY_0$ $UUUUUUUU_0$ $AAAAAAAA_0$
VL_PACKING_A_2_BGR_10	32	$RRRRRRRR_0$ $RRGGGGGG_0$ $GGGGBBBB_0$ $BBBBBBAA_0$
VL_PACKING_A_2_UYV_10	32	$VVVVVVVV_0$ $VVYYYYYY_0$ $YYYYUUUU_0$ $UUUUUUAA_0$
VL_PACKING_AYU_AYV_10	32	Pixels 0, 2, 4, 6: $UUUUUUUUUU_0$ $YYYYYYYYYY_0$ $AAAAAAAAAA_0$ XX Pixels 1, 3, 5, 7: $VVVVVVVVVV_0$ $YYYYYYYYYY_1$ $AAAAAAAAAA_1$ XX

Note: The subscript in each set indicates the pixel number in the source image.

Table 2-5 summarizes texture node packing types for Sirius Video texture drain nodes.

Table 2-5 Sirius Video Texture Packing Types

Type	Bits per Component
VL_SIR_TEX_PACK_RGB_5	5 (factory setting)
VL_SIR_TEX_PACK_RGBA_4	4
VL_SIR_TEX_PACK_RGBA_8	8

Source Node Controls

This section explains the use of source node controls in separate sections:

- digital video
- analog video
- graphics
- memory

Note: For maximum ease of use, set controls in the panel */usr/sbin/vcp*. Save the settings as explained in Appendix C, "Setting Up Sirius Video for Your Video Hardware," later in this guide. You do not need to open the panel to put its settings into effect.

Note: Blender node controls are discussed in Chapter 3, "Sirius Video Blending and Keying."

Digital Video Source Node Controls

Select the Digital 1 input node with a statement such as:

```
src = vlGetNode(svr, VL_SRC, VL_VIDEO, SIR_SRC_DIGITAL_VIDEO_1);
```

Select the Digital 2 input node with a statement like the following:

```
src = vlGetNode(svr, VL_SRC, VL_VIDEO, SIR_SRC_DIGITAL_VIDEO_2);
```

The controls for the VL_SRC nodes SIR_SRC_DIGITAL_VIDEO_1 and SIR_SRC_DIGITAL_VIDEO_2 are as follows:

- VL_FORMAT: values are
  - parallel 4:2:2:4: VL_FORMAT_DIGITAL_COMPONENT
  - serial 4:2:2:4: VL_FORMAT_DIGITAL_COMPONENT_SERIAL
  - parallel 4:4:4:4: VL_FORMAT_DIGITAL_COMPONENT_DUAL
  - serial 4:4:4:4: VL_FORMAT_DIGITAL_COMPONENT_DUAL_SERIAL

- VL_TIMING: either VL_TIMING_525_CCIR601 or VL_TIMING_625_CCIR601

  Note: Specify format first, then timing, and then other controls.

- VL_H_PHASE, VL_SIR_H_PHASE_ALPHA: the value for each is usually 0
- VL_SIR_FILTER
- VL_SIR_LINK_DELAY_A, VL_SIR_LINK_DELAY_B
- VL_SIR_FIELD_DOMINANCE
- VL_SIZE, VL_OFFSET, VL_ZOOM (all three read-only)

For digital video source nodes, set Pixel Format to parallel or serial 4:2:2:4, or to parallel or serial 4:4:4:4, depending on the equipment you have cabled to the Sirius Video breakout box.

- In 4:4:4:4 mode, Link A carries Y plus the U and V from even-numbered sample points; Link B carries alpha plus the U and V from odd-numbered sample points.
- In 4:2:2:4 mode, Link A carries Y plus the U and V from even-numbered sample points; Link B carries alpha only. If Link B is not connected to external video equipment, the system is 4:2:2 only.

Analog Video Source Node Controls

Select the video input node with a statement such as:

```
src = vlGetNode(svr, VL_SRC, VL_VIDEO, SIR_SRC_ANALOG_VIDEO);
```

The controls for VL_SRC node SIR_SRC_ANALOG_VIDEO are as follows:

- VL_FORMAT: values are
  - VL_FORMAT_RGB
  - VL_FORMAT_BETACAM
  - VL_FORMAT_MII
  - VL_FORMAT_SMPTE_YUV

  - VL_FORMAT_COMPOSITE
  - VL_FORMAT_SVIDEO

  VL_FORMAT determines the color space at which the data goes out; for example, for RGB data, set VL_FORMAT to VL_FORMAT_RGB; for Betacam, set VL_FORMAT to VL_FORMAT_BETACAM.

- VL_TIMING: values are
  - 525: VL_TIMING_525_SQ_PIX
  - 625: VL_TIMING_625_SQ_PIX
  - CCIR601 525: VL_TIMING_525_CCIR601
  - CCIR601 625: VL_TIMING_625_CCIR601
- VL_H_PHASE, VL_SIR_H_PHASE_ALPHA
- VL_SIR_RED_GAIN, VL_SIR_GREEN_GAIN, VL_SIR_BLUE_GAIN,VL_SIR_ALPHA_GAIN
- VL_SIR_RED_OFFSET, VL_SIR_GREEN_OFFSET, VL_ SIR_BLUE_OFFSET, VL_SIR_ALPHA_OFFSET
- VL_BRIGHTNESS, VL_CONTRAST, VL_SATURATION, VL_HUE
- VL_AUTO_GAIN_CONTROL
- VL_SIR_FIELD_DOMINANCE
- VL_SYNC (read-only), VL_SYNC_SOURCE
- VL_SIZE, VL_OFFSET, VL_ZOOM (all three read-only)

The following fragment sets the input sync source:

```
src = vlGetNode(svr, VL_SRC, VL_VIDEO, SIR_SRC_ANALOG_VIDEO);
vlAddNode(svr, path, src); /* Add node to existing path */

/* Set the input sync src */
syncsrc.intVal = SIR_SYNC_HOUSE;

if (vlSetControl(svr, path, drn, VL_SYNC_SRC, &syncsrc) <0)
{
     vlPerror("VlSetControl:SYNC SRC");
     exit(1);
}
```

Graphics Source Node Controls

Select the graphics input node with a statement such as:

```
src = vlGetNode(svr, VL_SRC, VL_GFX, VL_ANY);
```

Note: If Sirius Video and Multi-Channel Option are installed on the same graphics pipeline of your system, Sirius Video is not capable of live graphics to video. (You can, however, snap the high-resolution display, save it as a file, and send the file to video.)

The controls for VL_SRC node VL_GFX are as follows:

- VL_FORMAT (read-only: VL_FORMAT_RGB)
- VL_TIMING: values are
  - 525: VL_TIMING_525_SQ_PIX
  - 625: VL_TIMING_625_SQ_PIX
  - CCIR601 525: VL_TIMING_525_CCIR601
  - CCIR601 625: VL_TIMING_625_CCIR601

  Note: In many situations, CCIR timing gives better results.
- VL_ORIGIN, VL_SIR_GFX_SIZE

  Note: After setting VL_ORIGIN and VL_SIR_GFX_SIZE with *vlSetControl*(), immediately use *vlGetControl*() to determine the actual value set.
- VL_SIR_GFX_FLICKER
- VL_SIZE, VL_OFFSET (both read-only)

The following fragment sets typical graphics source node controls:

```
/* Get the first Gfx input */
src = vlGetNode(svr, VL_SRC, VL_GFX, VL_ANY);

/* See what output timing is */
if (vlGetControl(svr,path,drn,VL_TIMING,&timing) < 0 )
{
     vlPerror("vlSetControl");
     exit(1);
}
/* Set the src timing to drn timing */
if (vlSetControl(svr,path,src,VL_TIMING,&timing) < 0 )
{
     vlPerror("vlSetControl");
     exit(1);
}

/* Set the gfx grab origin */
origin.xyVal.x = xcoord;
origin.xyVal.y = ycoord;

/* Set the gfx grab area */
size.xyVal.x = xsize;
size.xyVal.y = ysize;

if (vlSetControl(svr,path,src,VL_SIR_GFX_SIZE,&size) < 0 )
{
     vlPerror("vlSetControl");
     exit(1);
}

if (vlSetControl(svr,path,src,VL_ORIGIN,&origin) < 0 )
{
     vlPerror("vlSetControl");
     exit(1);
}
```

Memory Source Node Controls

Set the memory input node with a statement such as:

```
src = vlGetNode(svr, VL_SRC, VL_MEM, VL_ANY);
```

The controls for VL_SRC node VL_MEM are as follows:

- VL_FORMAT: this control is always required for memory nodes; default is VL_FORMAT_DIGITAL_COMPONENT

  VL_FORMAT determines the color space at which the data comes in:

  - for RGB data, set VL_FORMAT to VL_FORMAT_RGB
  - for uncompressed YUV, set VL_FORMAT to VL_FORMAT_BETACAM, VL_FORMAT_MII, or VL_FORMAT_SMPTE_YUV
  - for CCIR-601-style compressed YUV, set VL_FORMAT to VL_FORMAT_DIGITAL_COMPONENT (the default)

  Note: For memory modes, format, timing, and packing are required. You must specify format first, then timing, then packing, and then other controls.

- VL_TIMING: this control is always required for memory nodes; values are

  – 525: VL_TIMING_525_SQ_PIX

  – 625: VL_TIMING_625_SQ_PIX

  – CCIR601 525: VL_TIMING_525_CCIR601(the default)

  – CCIR601 625: VL_TIMING_625_CCIR601

  Note: In many situations, CCIR timing gives better results.

- VL_PACKING: factory setting is VL_PACKING_YUVA_RGBA_8

  Note: Always set packing to one of the values in Table 2-4.

- VL_SIR_FILTER: controls whether the chroma-interpolating filters are used
- VL_CAP_TYPE (read-only: VL_CAPTURE_NONINTERLEAVED)

- VL_RATE (read-only: either 50 or 59.94 [field only])
- VL_SIZE, VL_OFFSET, VL_ZOOM (all three read-only)

Drain Node Controls

This section explains the use of drain node controls in separate sections:

Note: For maximum ease of use, set controls in the panel */usr/sbin/vcp*. Save the settings as explained in Appendix C, "Setting Up Sirius Video for Your Video Hardware," later in this guide. You do not need to open the panel to put its settings into effect.

- video
- graphics
- memory
- texture

Note: Blender node controls are discussed in Chapter 3, "Sirius Video Blending and Keying."

Video Drain Node Controls

Set the video output node with a statement such as:

```
drn = vlGetNode(svr, VL_DRN, VL_VIDEO, VL_ANY);
```

The controls for VL_DRN node VL_VIDEO are as follows:

- VL_FORMAT: values are
  - VL_FORMAT_RGB
  - VL_FORMAT_BETACAM
  - VL_FORMAT_MII
  - VL_FORMAT_SMPTE_YUV
  - VL_FORMAT_COMPOSITE
  - VL_FORMAT_SVIDEO
  - VL_FORMAT_DIGITAL_COMPONENT (digital 4:2:2:4)
  - VL_FORMAT_DIGITAL_COMPONENT_DUAL (digital 4:4:4:4)

  Note: As you do for video source node control, always specify format first, then timing, and then other controls.
- VL_TIMING: values are
  - 525: VL_TIMING_525_SQ_PIX
  - 625: VL_TIMING_625_SQ_PIX
  - CCIR601 525: VL_TIMING_525_CCIR601
  - CCIR601 625: VL_TIMING_625_CCIR601

  Note: In many situations, CCIR timing gives better results.
- VL_SIGNAL
- VL_SIR_ANALOG_DRAIN_BLANK_ENABLE, VL_SIR_DIGITAL_DRAIN_BLANK_ENABLE
- VL_SIR_RED_GAIN, VL_SIR_GREEN_GAIN, VL_SIR_BLUE_GAIN,VL_SIR_ALPHA_GAIN
- VL_SIR_FILTER
- VL_H_PHASE
- VL_SYNC, VL_SYNC_SOURCE, VL_SIR_SYNC_LEVEL
- VL_SIR_FIELD_DOMINANCE
- VL_SIZE, VL_OFFSET, VL_ZOOM (all three read-only)

Graphics Drain Node Controls

Set the video input node with a statement such as:

```
drn = vlGetNode(svr, VL_DRN, VL_GFX, 0);
```

The controls for VL_DRN node VL_GFX are as follows:

- VL_FORMAT (read-only: VL_FORMAT_RGB)
- VL_TIMING (available values are the same as for video drain nodes)
- VL_ORIGIN (read-only: 0,0)
- VL_OFFSET, VL_SIZE, VL_ZOOM (all three read-only)

Memory Drain Node Controls

Set the video input node with a statement such as:

```
drn = vlGetNode(svr, VL_DRN, VL_MEM, VL_ANY);
```

The controls for VL_DRN node VL_MEM are as follows:

- VL_FORMAT: this control is always required for memory nodes; default is VL_FORMAT_DIGITAL_COMPONENT

  VL_FORMAT determines the color space at which the data goes out; for example, for RGB data, set VL_FORMAT to VL_FORMAT_RGB; for Betacam, set VL_FORMAT to VL_FORMAT_BETACAM.

  Note: For memory modes, format, timing, and packing are required. You must specify format first, then timing, then packing, and then other controls.

- VL_TIMING: this control is always required for memory nodes; values are the same as for video drain nodes
- VL_PACKING: required; factory setting is VL_PACKING_YUVA_4444_8

  Note: Always set packing to one of the values in Table 2-4.

- VL_SIR_FILTER
- VL_CAP_TYPE: values are VL_CAPTURE_NONINTERLEAVED and VL_CAPTURE_INTERLEAVED

- VL_OFFSET, VL_SIZE. VL_ZOOM (all three read-only)
- VL_RATE (read-only: either 50 or 59.94 [field only])

Texture Drain Node Controls

Set the video input node with a statement such as:

```
drn = vlGetNode(svr, VL_DRN, VL_TEXTURE, VL_ANY);
```

The controls for VL_DRN node VL_TEXTURE are as follows:

- VL_FORMAT (read-only: VL_FORMAT_RGB)

  Note: Always specify format first, then timing, then other controls.
- VL_TIMING: values are the same as for video drain nodes
- VL_PACKING: required for correct functioning of the texture node; its values are
  - VL_SIR_TEX_PACK_RGB_5 (factory setting)
  - VL_SIR_TEX_PACK_RGBA_4
  - VL_SIR_TEX_PACK_RGBA_8
- VL_CAP_TYPE: values are
  - VL_CAPTURE_NONINTERLEAVED
  - VL_CAPTURE_INTERLEAVED
- VL_OFFSET, VL_SIZE, VL_ZOOM (all three read-only)

Using Filters

Digital filters convert video between 4:2:2 and 4:4:4 formats. Digital video and VME sources use interpolating filters; digital video and VME drains use decimating filters.

If you are converting from 4:2:2 to 4:4:4, set the interpolating filter on for the source and off for the drain. If you are converting 4:4:4 to 4:2:2, set the filter off for the source and set the decimating filter on for the drain. For example, if data from a 4:2:2 digital source is to be output as RGB (which is always 4:4:4), the interpolating filter must be turned on for the input. If data from a graphics source (always 4:4:4) is to be output as 4:2:2 digital, the decimating filter must be turned on for the output.

Table 2-6 summarizes these filter settings.

Table 2-6 Setting Filters

Input	On/Off	Output	On/Off
4:2:2	On	4:4:4	Off
4:4:4	Off	4:2:2	On
4:2:2	Off	4:2:2	Off
4:4:4	Off	4:4:4	Off

If you are passing data through in the same format (4:2:2 to 4:2:2, or 4:4:4 to 4:4:4), filters are not needed. If the filters are set on when the source or drain are 4:4:4, the results are not fatal, but image quality is reduced.

Note: For maximum ease of use, set filters and other settings in the panel */usr/sbin/vcp*. Save the settings as explained in Appendix C, "Setting Up Sirius Video for Your Video Hardware," later in this guide. You do not need to open the panel to put its settings into effect.

Sirius Video Events and Triggering

The VL provides several ways of handling data stream events, such as completion or failure of data transfer, vertical retrace event, loss of the path to another client, lack of detectable sync, or dropped fields or frames. The method you use depends on the kind of application you're writing:

- For a strictly VL application, use:
  - *vlSelectEvents*() to choose the events to which you want the application to respond
  - *vlCallback*() to specify the function called when the event occurs
  - your own event loop or a main loop (*vlMainLoop*()) to dispatch the events
- For an application that also accesses another program or device driver, or if you're adding video capability to an existing X or OpenGL application, set up an event loop in the main part of the application and use the IRIX file descriptor (FD) of the event(s) you want to add.

For more information on these functions, see Chapter 14, "VL Event Handling," in the *Digital Media Programming G*uide.

Table 2-7 summarizes events for Sirius Video. For Sirius Video, this table supersedes the table of events in Chapter 14, "VL Event Handling," in the *Digital Media Programming G*uide; Sirius Video supports only the events listed in Table 2-7.

Table 2-7 Events Supported by Sirius Video

Event	Use
VL_TRANSFER_COMPLETE	Sent for each field buffer transfer completed
VL_TRANSFER_FAILED	Sent if transfer failed for any reason
VL_STREAM_STARTED	Issued at the beginning of a DMA sequence
VL_STREAM_STOPPED	Issued when the DMA sequence has completed
VL_SEQUENCE_LOST	Issued if the kernel interrupts the DMA for any reason (such as timeout)
VL_SYNC_LOST	Issued if the hardware issues a sync lost interrupt

Note: No VL event is associated with the GPI inputs.

For GPI triggers, the hardware plugged into the GPI port on the Sirius Video breakout box must provide the GPI event, which should match the input level expected by the GPI port as selected by the device control panel or the VL_SIR_GPI_CHAN1_IN_MODE or VL_SIR_GPI_CHAN2_IN_MODE control. See Chapter 4, "Controlling the General-Purpose Interface (GPI)," for full details on GPI triggering.

For V-LAN triggers, issue V-LAN commands to the V-LAN controller so that it generates a coincidence pulse one frame before the point from which you want to grab or lay down an edit clip. See Chapter 5, "Controlling the V-LAN Interface," for full details on V-LAN triggering.

Passing Video Data Through the Graphics Subsystem

Included with Sirius Video system software are four special video output formats (VOFs) for passing video data through the graphics subsystem:

- 1280x1024_25f
- 1280x1024_30f
- 1280x1024_50f
- 1280x1024_60f

Invoke them with *setmon*(1G); for example:

```
setmon -n 1280x1024_25f
```

Note: You cannot set formats with *setmonitor***()**.

These formats configure the workstation display to lock to a video sync source connected to the RealityEngine's genlock input; they automatically select external genlock. If no video sync signal is present, these formats free-run at an inaccurate timebase (for example, 58 Hz..62 Hz for 1280x1024_60f, or 48 Hz..52 Hz for 1280x1024_50f).

In all of these formats, the graphics subsystem runs at twice the input video rate. The difference between the 30/60 or 25/50 versions of the formats is in the rate at which the graphics subsystem executes the *swapbuffers*() graphics function. The 50/60 versions allow swap buffers to run at twice the video frame rate. The 25/30 versions synchronize the workstation update with the video frame rate.

Use image tools in */usr/sbin izoom* (for example, *izoom*, *fromyuv*, or *snoop*) to perform operations on the RGB image files.

Note: To adjust values in the video source, see Appendix C, "Setting Up Sirius Video for Your Video Hardware," later in this guide.

Chapter 3

Sirius Video Blending and Keying

Sirius Video blending can implement the full compositing algebra, computing an output value for three input sources, as well as input from the chroma key generator. On each input, a normalization control can scale the input RGB or YUV values by the selected alpha before blending.

Note: See Chapter 15, "VL Blending," of the *Digital Media Programming Guide* for background information on blending and keying.

This chapter explains

- using *vcp* to set chroma key generator values
- using *vcp* to set blend function values
- using VL chroma key generator node controls
- using VL blend node controls

This chapter concludes with an example fragment.

Note: The *gfxvidkeytovid* program for the chroma key generator blends live video and graphics: it sets input, video output, and grab area, sets blender functions, sets values, ranges, and softness parameters for each component (A, B, and C), and creates special keys in a proprietary color space. This program is described in Appendix E, "Example Programs," later in this guide. If Sirius Video and Multi-Channel Option are installed on the same VME section of your system, Sirius Video is unable to use this program.

Using *vcp* to Set Chroma Key Generator Values

Use the “Sirius Chroma Key Generator” menu item in the *vcp* Pro menu to set levels—value, range, and softness (transition)—for the three input color components A, B, and C. By default, the color components A, B, and C correspond to V, Y, and U, respectively, but they can be set to RGB or any other color space with application software.

In the Pro menu, select “Chroma Key Generator Controls”, and then select “Keying Controls”. The Sirius Chroma Key Generator-Keying window appears, as shown in Figure 3-1.

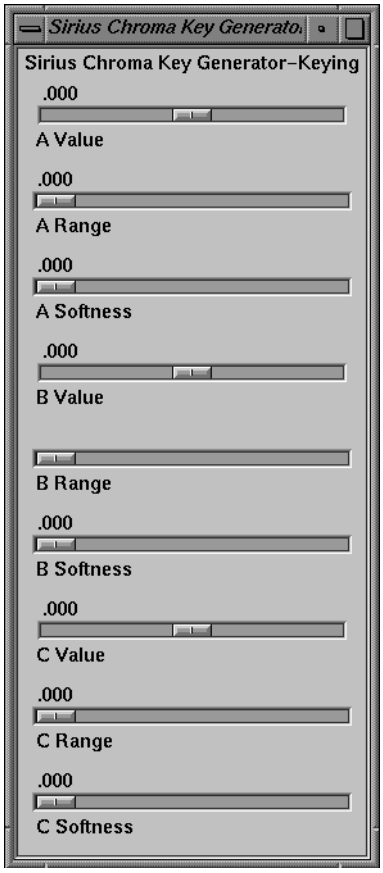

Figure 3-1 Chroma Key Generator Controls

In this window:

- *Value* is
  - for the B (Y) channel: brightness
  - for the A (V) and C (U) channels: hue and saturation
  - all three
- *Range* extends from each side of the value to the beginning of the transition.
- If the input signal varies smoothly in color, the optional *softness* value, or transition, allows you to control the range of input colors, which translate to a partially opaque key signal.

Figure 3-2 shows the relationships between value, range, and softness (transition) for a single channel (for example, A).

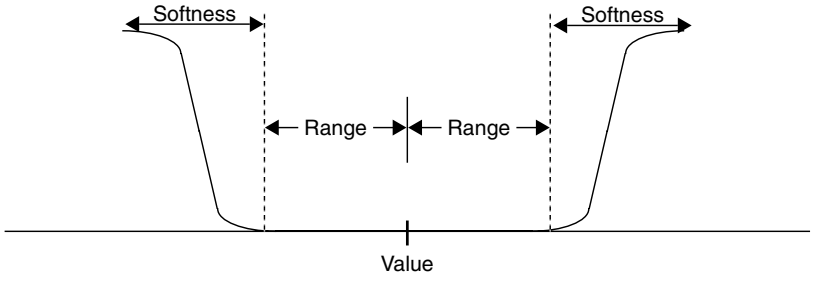

Figure 3-2 Value, Range, and Softness for a Channel

See "Using VL Chroma Key Generator Node Controls," later in this chapter, for more information on chroma key generation.

Using *vcp* to Set Blend Function Values

The blender has two inputs, path A and path B. Each has an associated alpha that is separately controllable. Sirius Video specific VL controls set these inputs; they are discussed in "Using VL Blend Node Controls," later in this chapter. Settings in the panel control blending of these paths.

To use the panel to set the blend functions that control mixing of frames from paths A and B, select "Blender Controls" in the Pro menu, and then select "Blending Controls". The Sirius Blender-Blending window appears, as shown in Figure 3-3.

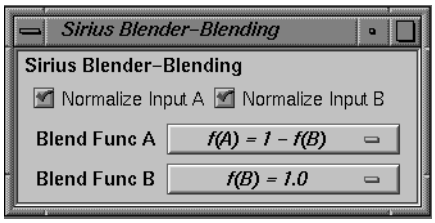

Figure 3-3 Setting Blender Function Values

Either source (Blend Function A or Blend Function B) can be background or foreground.

The choices for the two blend functions A and B correspond exactly, as shown in Table 3-1, although the order varies slightly.

Table 3-1 Choices for Blend Functions A and B

Blend Func A	Blend Func B
f(A) = 0.0	f(B) = 0.0
f(A) = 1.0	f(B) = 1.0
f(A)	f(B) = f(A)
f(A) = 1 - f(A)	f(B) = 1 - f(A)
f(A) = f(B)	f(B)
f(A) = 1 - f(B)	f(B) = 1 - f(B)

Note: For more information on binary compositing, see Table 3-5 later in this chapter.

The check boxes at the top of the Blender window set normalization on for each blending source. Normalized background pixels for a frame are premultiplied by their corresponding alphas before they are blended (Porter-Duff model).

Using VL Chroma Key Generator Node Controls

Blending for Sirius Video can be based on the *alpha value* (opacity) that the *chroma key generator* determines for each pixel for one of the sources to be blended. Sirius Video generates this value by chroma extraction, luma extraction, or both.

The Sirius Video chroma key generation controls set parameters for three input color components, A, B, and C. By default, these components are V, Y, and U, respectively, but they can be changed by application software to R, G, and B, or any other color space via a Sirius Video control. The resulting alpha value can be used by the blender or passed on to the VME, video, or graphics outputs.

Note: Because keying parameters and some blending parameters are implemented as device-dependent VL controls, see Chapter 15, "VL Blending," of the *Digital Media Programming Guide* for background information on blending and keying.

This section explains

- the color key volume
- color-space conversion

The Color Key Volume

The chroma key generator creates a soft-edged rectangular color volume, considered to be transparent (its alpha is 0), which is the excluded region for keying. This region specifies both color and luminance; setting either the chroma or luminance range to 1 yields pure luma or pure chroma keying.

The remainder of the YUV color space is considered opaque for keying. Luma keying is typically used to overlay a fixed image on video, such as the name and title of an individual being interviewed, a cable channel's logo, or a symbol that denotes an ongoing news story during a newscast. Chroma keying overlays one image on another based on the color value. Figure 3-4 diagrams a common chroma key application.

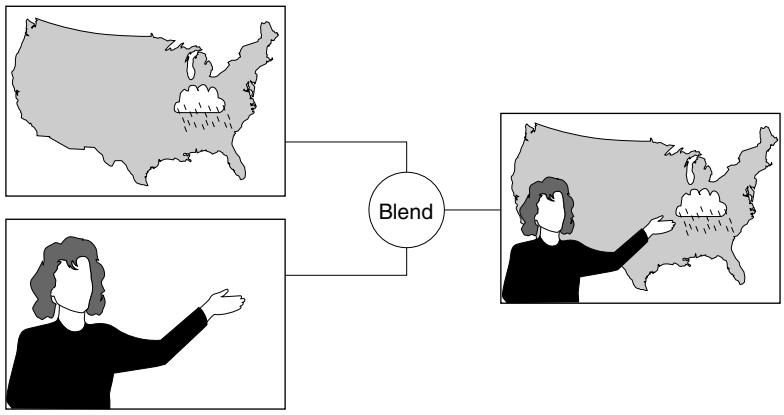

Figure 3-4 Chroma Keying Application

Each input color component (A, B, and C) has a value control, a range control and a softness control:

- *Value* is:
  - for the B (Y) channel: brightness
  - for the A (V) and C (U) channels: hue and saturation
  - all three
- *Range* extends from each side of the value to the beginning of the transition.
- If the input signal varies smoothly in color, the optional *softness* value, or transition (computed with the formula $3x^2 - 2x^3$), allows you to control the range of input colors, which translate to a partially opaque key signal.

Note: The *gfxvidkeytovid* example program, printed in Appendix C, enables the chroma key generator.

Table 3-2 summarizes the controls for Sirius Video chroma key generation.

Table 3-2 Sirius Video Chroma Keying Controls

Control	Range	Sets...
VL_SIR_KEYGEN_A_VALUE	-1..1	Central value for channel A (default: V)
VL_SIR_KEYGEN_A_RANGE	-1..1	Range for channel A (default: V)
VL_SIR_KEYGEN_A_SOFTNESS	-1..1	Sharpness of transition for channel A (default: V)
VL_SIR_KEYGEN_B_VALUE	-1..1	Central value for channel B (default: Y)
VL_SIR_KEYGEN_B_RANGE	-1..1	Range for channel B (default: Y)
VL_SIR_KEYGEN_B_SOFTNESS	-1..1	Sharpness of transition for channel B (default: Y)
VL_SIR_KEYGEN_C_VALUE	-1..1	Central value for channel C (default: U)
VL_SIR_KEYGEN_C_RANGE	-1..1	Range for channel C (default: U)
VL_SIR_KEYGEN_C_SOFTNESS	-1..1	Sharpness of transition for channel C (default: U)
VL_SIR_KEYGEN_MATRIX	Float matrix [3][3]	Key generator matrix, for example, changing from the default (YUV) to another color space, such as RGB

Color-Space Conversion

The Sirius Video chroma key generator automatically converts from the input color space to YUV (C_rYC_b). For example, if the input is RGB, it converts using the CCIR values in Table 3-3.

Table 3-3 RGB to YUV Color-Space Conversion

	R	G	B
C_r (V)	0.50000	-0.4900	-0.08100
Y	0.29900	0.58700	0.11400
C_b (U)	-0.16900	-0.33100	0.50000

You can change the default YUV color space to RGB or another color space with the matrix control VL_SIR_KEYGEN_MATRIX. The key generator matrix is concatenated to the precomputed matrix. For example, to perform key generation in RGB space, specify the values shown in Table 3-4.

Table 3-4 YUV to RGB Color-Space Conversion

	C_r	Y	C_b
R	1.401687	1.000000	-0.000927
G	-0.714169	1.000000	-0.343695
B	0.000990	1.000000	1.772160

Note: For more information on color-space conversion, see Appendix D, “Sirius Video Color-Space Conversions,” in this guide.

The following example (a fragment from *gfxvidkeytovid.c*) illustrates setting the value of components:

```
/* Get denominator for the Key Generator fractions */
if (vlGetControl(svr, path, key, VL_SIR_KEYGEN_A_VALUE ,&val) < 0 ) {
     vlPerror("vlSetControl:KEYGEN:A_VALUE");
     exit(1);
}

/* Set A value */
val.fractVal.numerator = aval*val.fractVal.denominator;
if (vlSetControl(svr, path, key, VL_SIR_KEYGEN_A_VALUE ,&val) < 0 ) {
     vlPerror("vlSetControl:KEYGEN:A_VALUE");
     exit(1);
}
/* Set B value */
val.fractVal.numerator = bval*val.fractVal.denominator;
if (vlSetControl(svr, path, key, VL_SIR_KEYGEN_B_VALUE ,&val) < 0 ) {
     vlPerror("vlSetControl:KEYGEN:A_range");
     exit(1);
}
/* Set C Value */
val.fractVal.numerator = cval*val.fractVal.denominator;
if (vlSetControl(svr, path, key, VL_SIR_KEYGEN_C_VALUE ,&val) < 0 ) {
     vlPerror("vlSetControl:KEYGEN:A_softness");
     exit(1);
}
```

The following example (a fragment from *gfxvidkeytovid.c*) illustrates the use of VL_SIR_KEYGEN_MATRIX.

```
/*
* The following is an example of using custom matrices to create
* special keys in a private color space.
*/

val.matrixVal[0][0] = 1.401687;
val.matrixVal[0][1] = 1.0;
val.matrix[Val0][2] = -0.000927;

val.matrixVal[1][0] = -0.71469;
val.matrixVal[1][1] = 1.0;
val.matrixVal[1][2] = -0.343695;

val.matrixVal[2][0] = 0.00099;
val.matrixVal[2][1] = 1.0;
val.matrixVal[2][2] = 1.772160;

if (vlSetControl(svr, path, key, VL_SIR_KEYGEN_MATRIX ,&val) < 0 ) {
     vlPerror("vlSetControl:KEYGEN");
     exit(1);
}
```

The output of the Sirius Video chroma key generator can be passed to the blend node or any output node.

Using VL Blend Node Controls

The Sirius Video blend node is supplied by up to four independent inputs:

- a source (A), which can be either background or foreground
- the alpha value for source A
- a second source (B), which can be either foreground or background
- one of the following:
  - input from the chroma key generator, including soft transition
  - the alpha value for source B

Figure 3-5 diagrams the Sirius Video blend node.

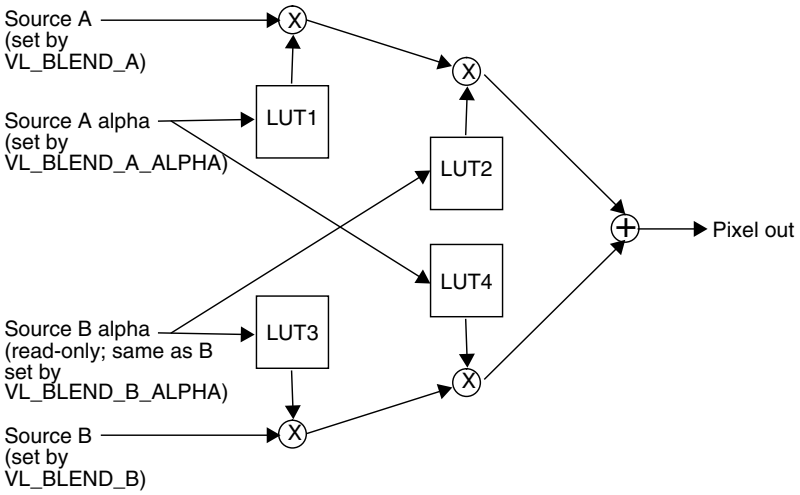

Figure 3-5 Sirius Video Blend Node

The Sirius Video blend node has two multiplier stages, indicated by ⊗ in Figure 3-5, and one adder stage, indicated by ⊕. The values in the four lookup tables are based on the blending functions selected and on the input normalization controls. Table 3-5 gives some examples of compositing, assuming normalized inputs.

Table 3-5 Binary Compositing *

Operation	Diagram	f(A) =	f(B) =
Clear		0	0
A		1	
B			1
A over B		1	1-f(A)

Table 3-5 (continued) Binary Compositing *

Operation	Diagram	f(A) =	f(B) =
B over A		1-f(B)	1
A in B		f(B)	
B in A			f(A)
A held out by B		1-f(B)	
B held out by A			1-f(A)
A atop B		f(B)	1-f(A)
B atop A		1-f(B)	f(A)
A xor B (union of A out B and B out A)		1-f(B)	1-f(A)

* Table derived from Thomas Porter and Tom Duff, "Compositing Digital Images," published by the Association for Computing Machinery, 1984.

Normally, chroma is multiplied (scaled) by the selected alpha. For example, the value on source A can be multiplied by its own alpha value or that from source B. In a normal blend, f(A), the incoming alpha of source A is applied to the value for A. In the inverse of this blend, f(A)=1-f(A), the region that was considered opaque (turned off), that is, outside the volume defined for keying, is applied to source A.

In another way of blending, the alpha from source B can be applied to the component represented by source A. In the inverse of this blend, f(A)=1-f(B), the region that was turned off for source B is applied to source A.

The alpha for source B can be set to the chroma key generator; that is, values from the CKG are passed to the blend node as Bα. Thus Bα is either the alpha from source B or the value from the CKG. On the other hand, the values for source A and for source Aα can be specified independently of each other.

Note: The alpha for source B is read-only.

Table 3-6 summarizes the VL controls that apply to the Sirius Video blend node.

Table 3-6 Blend Controls

Control	Values	Selects...
VL_BLEND_A_FCN type *intVal*	VL_BLDFCN_ZERO VL_BLDFCN_ONE VL_BLDFCN_B_ALPHA (B source alpha)/255 VL_BLDFCN_MINUS_B_ALPHA: 1 ((B source alpha) / 255)	Blend function that controls mixing of A source signals
VL_BLEND_B_FCN type *intVal*	VL_BLDFCN_ZERO VL_BLDFCN_ONE VL_BLDFCN_A_ALPHA (A source alpha)/255 VL_BLDFCN_MINUS_A_ALPHA 1 - ((A source alpha) / 255)	Blend function that controls mixing of B source signals
VL_BLEND_A type *intVal*	VLNode type, derived from **vlGetNode()**; must be one of the two source nodes	A source image input
VL_BLEND_B type *intVal*	VLNode type, derived from **vlGetNode()**; must be one of the two source nodes	B source image input
VL_BLEND_A_ALPHA type *intVal*	VLNode type, derived from **vlGetNode()**; must be one of the two source nodes	A source alpha input
VL_BLEND_B_ALPHA type *intVal*	VLNode type, derived from **vlGetNode()**; must be one of the two source nodes	B source alpha input

Table 3-6 (continued) Blend Controls

Control	Values	Selects...
VL_BLEND_A_NORMALIZE type *boolVal*	(0,1) 0 = off, 1 = on	Follows Porter-Duff model (B source pixels premultiplied by their corresponding alphas before blending)
VL_BLEND_B_NORMALIZE type *boolVal*	(0,1) 0 = off, 1 = on	Follows Porter-Duff model
VL_BLEND_OUT_NORMALIZE type *boolVal*	Read-only; always on	Follows Porter-Duff model

Example

The following fragment (from *gfxvidtovid.c*) blends graphics input and video input.

```
/* Get the first Gfx input */
gfx_src = vlGetNode(svr, VL_SRC, VL_GFX, 0);
vid_src = vlGetNode(svr, VL_SRC, VL_VIDEO, 0);
/* Get the first Video output */
drn = vlGetNode(svr, VL_DRN, VL_VIDEO, 0);

blend = vlGetNode(svr, VL_INTERNAL, VL_BLENDER, VL_ANY);

/* Create paths */
path = vlCreatePath(svr, VL_ANY, gfx_src, drn);
if (path < 0)
{
     vlPerror("vlCreatePath");
     exit(1);
}

vlAddNode(svr, path, blnd);
vlAddNode(svr, path, vid_src);

/* setup path */
if (vlSetupPaths(svr, (VLPathList)&path,1, VL_SHARE, VL_SHARE) < 0)
{
     vlPerror("vlSetupPaths");
     exit(1);
}
```

```
/* gfx src controls */
origin.xyVal.x = xcoord;
origin.xyVal.y = ycoord;

size.xyVal.x = xsize;
size.xyVal.y = ysize;

if (vlSetControl(svr,path,gfx_src,VL_SIR_GFX_SIZE,&size) < 0 ) {
     vlPerror("vlSetControl:GFX SIZE");
     exit(1);
}

if (vlSetControl(svr,path,gfx_src,VL_ORIGIN,&origin) < 0 )
{
     vlPerror("vlSetControl:ORIGIN");
     exit(1);
}
/* Set the Blender A source to be the gfx source */
val.intVal = gfx_src;
vlSetControl(svr, path, blnd, VL_BLEND_A, &val);
/* Set the Blender A_ALPHA source to be the gfx source */
val.intVal = gfx_src;
vlSetControl(svr, path, blnd, VL_BLEND_A_ALPHA, &val);
/* Set the Blender B source to be the video source */
val.intVal = vid_src;
vlSetControl(svr, path, blnd, VL_BLEND_B, &val);
```

Chapter 4

Controlling the General-Purpose Interface (GPI)

Use the two-channel GPI (General-Purpose Interface) port on the Sirius Video breakout box to communicate with various video devices. Figure 4-1 shows the location of the nine-pin GPI port on the Sirius Video breakout box.

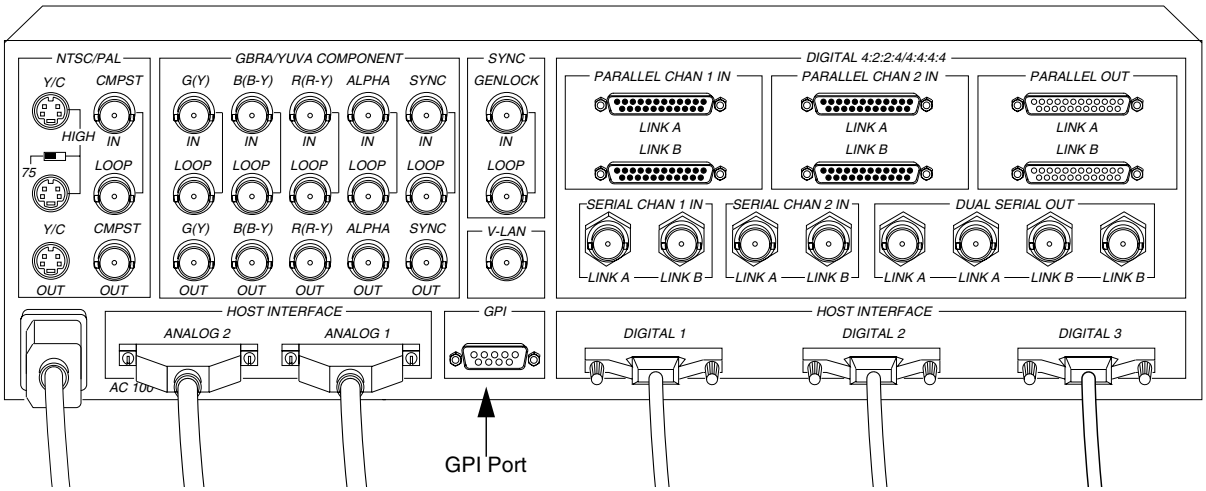

Figure 4-1 GPI Port on Sirius Video Breakout Box

This chapter explains how to control the GPI:

- using Sirius Video utilities for the GPI
- using VL controls for the GPI
- GPI pinouts

Using Sirius Video Utilities for the GPI

This section explains how to use the following utilities to control the GPI:

- *vcp*
- *sir_vidtomem* and *sir_memtovid*

Using *sir_vidtomem* and *sir_memtovid* to Control the GPI

The command-line option for *sir_vidtomem* and *sir_memtovid* that generates a pair of GPI notify events is `-T`. This option generates the notify events at the start (GPI_1 output) and end (GPI_2 output) of the transfer.

The command-line option for *sir_vidtomem* and *sir_memtovid* that specifies an external trigger source that initiates the output DMA sequence is `-t#`.

For more information on *sir_vidtomem* and *sir_memtovid*, see Appendix E, "Example Programs," later in this guide.

Using *vcp* to Control the GPI

To use the video control panel to set modes for GPI channels 1 and 2, follow these steps:

1. Call up *vcp*:

   ```
   /usr/sbin/vcp
   ```

2. In the Pro menu (in the menu bar), select Device Controls; select Synchronization Controls. The Sirius Device-Sync Input Controls window appears, as shown in Figure 4-2.

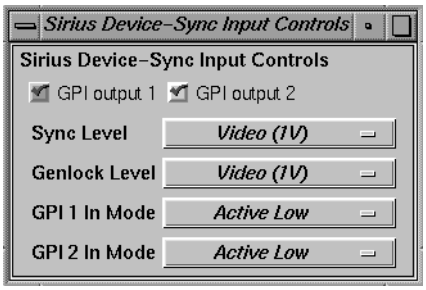

Figure 4-2 Device Synchronization Input Controls

3. Click GPI output 1, GPI output 2, or both, depending on the equipment you are using. Red checks in the boxes mean that these outputs are enabled for *vcp*.
4. To set input to active low, select "Active Low" in the GPI 1 In Mode or GPI2 In Mode menu item of the Sync Input Controls menu, as shown in Figure 4-2.

   In active low mode (Abekas switch closure mode), the triggering device must drive either GPI_IN + line high with its GPI_OUT + line. The GPI_IN + line from the triggering device should be connected to the Sirius Video breakout box's ground pin (pin 5).
5. To set input to active high, select "Active High" in the GPI 1 In Mode or GPI2 In Mode menu item of the Sync Input Controls menu, as shown in Figure 4-2.

   In active high mode, the triggering device must drive either GPI_IN + line high with its GPI_OUT + line. The GPI_IN + line from the triggering device should be connected to the Sirius Video breakout box's ground pin (pin 5).

Using VL Controls for the GPI

This section explains the device-dependent VL controls for controlling the GPI. Table 4-1 summarizes these controls.

Table 4-1 VL Controls for GPI Triggering

Purpose	Control
Trigger a GPI event	VL_SIR_TRIGGER: SIR_TRIGGER_GPI_1, SIR_TRIGGER_GPI_2
Issue GPI events relative to the start and end of the transfer	VL_SIR_XFER_[START,STOP}_NOTIFY: SIR_TRIGGER_NONE, SIR_TRIGGER, GPI_1, or SIR_TRIGGER_GPI_2
Configure GPI inputs	VL_SIR_GPI_CHAN1_IN_MODE VL_SIR_GPI_CHAN2_IN_MODE
Write to GPI outputs	VL_SIR_GPI_OUT_CHAN0 VL_SIR_GPI_OUT_CHAN1

Sending and Receiving GPI Events

Sirius Video responds to GPI input if the VL_SIR_TRIGGER control is applied, that is, if a GPI trigger event is requested; thus, memory-based transfers constitute the communication from GPI input to Sirius Video. You can send a GPI event at the start or at the end of a transfer. You can also write directly to the GPI, as explained later in this chapter.

To turn off GPI triggering, set the control(s) to SIR_TRIGGER_NONE.

Note: No VL event is associated with the GPI inputs.

Input Events

If a GPI trigger event is requested, the transfer pauses and waits for a GPI event from the external device connected to the GPI port on the breakout box. When such an event is received, the read/write DMA transfer proceeds. When the last DMA request has been serviced, the software checks to see if an end transfer notify trigger was requested. If it was, the requested GPI line is pulled low for 20 msec.

For GPI triggers, the hardware plugged into the GPI port on the Sirius video breakout box must provide the GPI event. That event should match the input level expected by the GPI port as selected by the device control panel, or by the VL_SIR_GPI_CHAN1_IN_MODE or VL_SIR_GPI_CHAN2_IN_MODE control.

Output Events

You can also generate GPI output events at the beginning and at the end of each transfer by setting the VL_SIR_TRANSFER_START_NOTIFY and VL_SIR_TRANSFER_STOP_NOTIFY controls, respectively, to either VL_SIR_TRIGGER_GPI_1 or VL_SIR_TRIGGER_GPI_2.

At the start of a DMA transfer, the transfer start notify trigger (VL_SIR_GPI_XFER_START_NOTIFY) drives the GPI output line low for 20 msec. The start and stop GPIs should usually go to different pins.

Configuring GPI Input

Specify mode for each GPI channel separately:

```
vlSetControl(svr, path, dev, VL_SIR_GPI_CHAN1_IN_MODE [value])
vlSetControl(svr, path, dev, VL_SIR_GPI_CHAN2_IN_MODE [value])
```

GPI input can be configured as active low or active high only; specify the input according to the hardware you are using. Each possibility is explained separately.

Active Low Mode (Abekas Switch Closure Mode)

In active low mode, the triggering device must drive either GPI_IN + line high with its GPI_OUT + line. The GPI_IN + line from the triggering device should be connected to the Sirius Video breakout box's ground pin (pin 5).

To set input to active low, specify the channel and the mode; for example:

```
vlSetControl(svr, path, dev,
VL_SIR_GPI_CHAN1_IN_MODE, SIR_GPI_IN_MODE_ABEKAS_SWITCH_CLOSURE)
```

or

```
vlSetControl(svr, path, dev,
VL_SIR_GPI_CHAN1_IN_MODE SIR_GPI_IN_MODE_ACTIVE_LOW)
```

Active High

In active high mode, the triggering device must drive either GPI_IN + line high with its GPI_OUT + line. The GPI_IN + line from the triggering device should be connected to the Sirius Video breakout box's ground pin (pin 5).

To set input to active high, specify the channel and the mode; for example:

```
vlSetControl(svr, path, dev,
VL_SIR_GPI_CHAN1_IN_MODE, SIR_GPI_IN_MODE_ACTIVE_HIGH)
```

Configuring GPI Output

Specify output mode for each GPI channel separately:

```
vlSetControl(svr, path, dev, VL_SIR_GPI_OUT_CHAN1 [value])
```

```
vlSetControl(svr, path, dev, VL_SIR_GPI_OUT_CHAN2 [value])
```

Like input, GPI output can be driven either high (off) or low (on). When GPI output is driven high, the photovoltaic relay is in an active state; when GPI output is driven low, the photovoltaic relay is passive.

Note: For GPI output, 0 is on and 1 is off.

To set output to high (off), specify the channel and the mode; for example:

```
vlSetControl(svr, path, dev, VL_SIR_GPI_OUT_CHAN1 SIR_GPI_OUT_OFF)
```

To set output to low (on), specify the channel and the mode; for example:

```
vlSetControl(svr, path, dev, VL_SIR_GPI_OUT_CHAN1 SIR_GPI_OUT_ON)
```

GPI Pinouts

For each of two channels, the GPI provides positive and negative (actually, bidirectional) input and output wires. Figure 4-3 shows pinouts for the GPI.

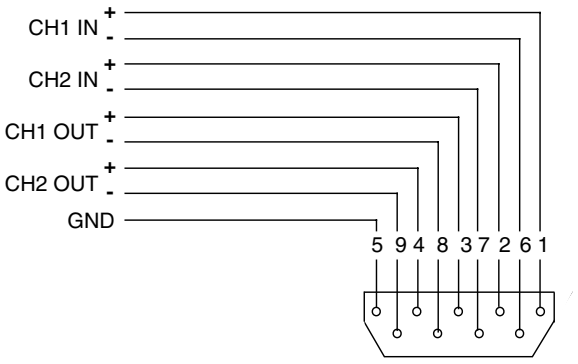

Figure 4-3 GPI Interface Pinouts

Table 4-2 summarizes these pinouts in numerical order.

Table 4-2 GPI Pin Usage

Pin	Use
1	Channel 1 input positive
2	Channel 2 input positive
3	Channel 1 output positive
4	Channel 2 output positive
5	GND
6	Channel 1 input negative
7	Channel 2 input negative
8	Channel 1 output negative
9	Channel 2 output negative

Note: Input and output pins in this table are actually bidirectional; positive and negative nomenclature is for reference only.

Controlling the V-LAN Interface

V-LAN is a video device network protocol designed and implemented by Videomedia Inc. It provides reliable, frame-accurate control of professional and industrial-quality video equipment. This chapter explains how to use VL and Sirius Video device-dependent Video Library controls to communicate with the Sirius Video on-board V-LAN controller (transmitter). Topics include:

- sending V-LAN commands
- using V-LAN triggers
- using V-LAN commands
- sample V-LAN sequences with *sir_vlan.c*

Note: To use the V-LAN interface, purchase the appropriate receiver from Videomedia, Inc.:

- Videomedia, Inc.
  211 East Weddell Drive
  Sunnyvale, CA 94089 USA
  Phone: 1-408-745-1700
  Fax: 1-408-745-6721)
- VME, Videomedia Europe
  Unit 7, Portland Business Centre
  Manor House Lane, Datchet, Berkshire SL3 9EG
  Great Britain
  Phone: (0753) 581596
  Fax: (0753) 540612

Attach the device to the V-LAN network connector on the Sirius Video breakout box. You can attach as many as 31 V-LAN receivers, each controlling a particular class of VTR.

sir_vlan, a Sirius Video configuration and command tool, performs the basic V-LAN operations of initializing the on-board V-LAN controller, querying controller status, and sending V-LAN commands to it. For information on this program, see Appendix E, “Example Programs,” later in this guide.

Sending V-LAN Commands

Use VL_SIR_VLAN_CMD to send a V-LAN command to the controller and read the response from the V-LAN controller. Each V-LAN command consists of two ASCII characters and, where appropriate, a parameter. The extensive V-LAN command set controls tape deck recording, transport movement, editing, edit point setup, frame grab, and more; it is fully explained in this chapter.

Note: Sirius Video utilizes version 3.11 of the V-LAN command set. This command set includes commands not described in this chapter. Documentation is included with Videomedia V-LAN receivers; contact Videomedia for the latest revision.

Send a V_LAN control with

```
vlSetControl(svr, path, dev, VL_SIR_VLAN_CMD, &vlanCommand)
```

If the command string exceeds the length of the *vlSetControl*() buffer, the string is truncated and a warning message is printed, but the command is executed.

Using V-LAN Triggers

For V-LAN triggers, issue V-LAN commands to the V-LAN controller so that it generates a coincidence pulse one frame before the point from which you want to grab or lay down an edit clip. A typical V-LAN command sequence for this purpose is in Table 5-1.

Table 5-1 Typical V-LAN Sequence for Triggering

V-LAN Command	Purpose
SI#	Sets inpoint in SMPTE time code
SD#	Sets duration
PR#	Sets five-second preroll
TSV	Sets track select video
CO	Turns on coincidence pulse
SC#	Sets inpoint: one frame in SMPTE time code
RV	Reviews the edit for grabs
PF	Performs the edit, for an edit laydown

Using V-LAN Commands

V-LAN commands fall into several groups:

- movement commands
- GOTO commands
- edit commands
- wait mode commands
- edit point setup commands
- status commands
- frame grab commands
- validity checks

- sync play commands
- relays for a GPSI (General-Purpose System Interface) box
- slow motion commands
- miscellaneous commands

Table 5-2 summarizes the commands in each group.

Table 5-2 V-LAN Commands

Group	Command	Code	Notes
Movement	PLAY	PY	
	PAUSE	PS	
	STOP	ST	
	EJECT	EJ	Does not work on parallel systems
	FAST FORWARD	FF	
	FAST REWIND	RW	
	SHUTTLE	SH#	# = 0... 9
	JOG FORWARD	JF	
	JOG REVERSE	JR	
	RECORD	RC	
GOTO	GOTO IN	GI	Searches to inpoint
	GOTO OUT	GO	Searches to outpoint
	GOTO PREROLL	GP	Searches to inpoint less preroll
	GOTO LOCATION	GT#	# = location
Edit	PERFORM	PF#	# = nodes to sync with recorder (1-9)
	REHEARSE	RH#	# = nodes to sync with recorder (1-9)
	REVIEW	RV#	# = nodes to sync with recorder (1-9)

Table 5-2 (continued) V-LAN Commands

Group	Command	Code	Notes
	RECORD NODE SELECT	NR#	# = 1-31, separated by commas
	SOURCE NODE SELECT	NS#	# = 1-31, separated by commas
	ROLL BEGIN	RB#	# = start time after preroll
	ROLL END	RE#	# = stop time after preroll
	EDIT ERROR	EE	Returns edit error
	EDIT SYNC	SY#	Selects edit sync (0, 1, 2)
	COLOR BUMP	CB	Bumps to next color frame
	OUT ON THE FLY	OF	Selects outpoint on the fly
	END CONDITION	EC c	c = command to execute after edit
	ABORT SEQUENCE	AB	
Wait Mode	WAIT ON	WN	Turns wait mode on
	WAIT OFF	WF	Turns wait mode off
	WAIT GO	WG	Go now from wait position
Edit Point Setup	NODE ADDRESS	ND#	# = node to talk with
	TRACK SELECT	TS m	m = mode: V = video; 1,2 = audio; A = assemble
	SET IN	SI#	# = location
	SET OUT	SO#	# = location
	SET DURATION	SD#	# = location
	TRIM INPOINT	TI#	# = amount to trim inpoint
	TRIM OUTPOINT	TO#	# = amount to trim outpoint
	TRIM DURATION	TD#	# = amount to trim duration
	PREROLL SET	PR#	# = length of preroll (default 5 sec)

Table 5-2 (continued) V-LAN Commands

Group	Command	Code	Notes
	POSTROLL SET	PT#	# = length of postroll (default 1 sec)
	AUTO INCREMENT	AI#	# = amount in/outpoints automatically incremented
	ENTER IN	EI	Present location entered for IN
	ENTER OUT	EO	Present location entered for OUT
	READ IN	RI	Inpoint returned
	READ OUT	RO	Outpoint returned
	READ DURATION	RD	Duration returned
	CLEAR EVENT	CL	In, out, and duration cleared
	ADD TRACKS	AT t	t = tracks: V = video, audio = 1-9
	SET SPLIT	SS#	# = record tape location where split starts
	NEW TRACK SELECT	NT t#	t = tracks # = time after preroll
Status	STATUS REQUEST	SR#	# = optional node, ASCII status returned
	LOCATION REQUEST	LR#	# = optional node, location returned
	LOCATION CONT	LC	Location returned until <Enter>
	LOCATION + STATUS	LS#	# = optional mode, LOC and STATUS returned
	STATUS ENCODED	SE#	# = optional node
	EDIT STATUS	ES	12-byte edit status returned
	APPLICATION TYPE	AP	# optional node; returns # (0...9) indicating type of transport
	USER BIT RETURN	UB	Returns 8 bytes of user bit data

Table 5-2 (continued) V-LAN Commands

Group	Command	Code	Notes
Frame Grab	COINCIDENCE ON	CO	Enables coincidence character return
	COINCIDENCE OFF	CF	Coincidence checking disabled
	SET COINCIDENCE	SC#	Coincidence time register set
Validity Checks	CODE VALID	CV	Returns Y if time code valid, else N
	DEVICE TYPE	DT	Device type code and version
	SYNC CHECK	SN	Checks if valid sync applied to the controller
Sync Play	ROLL	RL	Rolls and syncs to master running frame count
	END ROLL	ER	Ends sync roll
	SET FRAME COUNT	SF#	Sets running frame count
	READ FRAME COUNT	RF	Current value of V-LAN frame count returned
Relays for GPSI Box	RELAY SET	RS#,#,..	Sets specified relays, # separated by commas Valid relays are 1-16; #0 = all relays
	RELAY RESET	RR#,#,..	Resets specified relays
	RELAY PULSE	RP#,#,..	Pulses specified relays
	SET PULSE TIME	SP#	# = time in frames; sets pulse length
	CLEAR RELAYS	CR	Clears relay events
	RELAY TRIGGER	RT#, t	t = time after preroll; programs relays
Slow Motion	VARIABLE SPEED PLAY	VP #	# = speed

Table 5-2 (continued) V-LAN Commands

Group	Command	Code	Notes
	VARIABLE SHUTTLE	VS #	# = shuttle speed
	RETURN CURRENT VARIABLE SPEED	VR	Returns 3 bytes of speed data
	TEST IF VARIABLE SPEED AVAILABLE	TV	Returns Y or N
	CHECK AVAILABILITY OF SPEED	CA #	# = speed to check; returns 3 bytes of data
	SET VARIABLE SPEED	SV #	# = speed for edit
	VARIABLE SPEED	VC	Changes edit speed during edit
Miscellaneous	ECHO ON	EH	Turns echo on
	ECHO OFF	EF	Turns echo off
	EE ON	YE	EE mode on
	EE OFF	NE	EE mode off
	CODE MODE SELECT	CS#	# = 0, 1: 1 = time code, 0 = control track
	CODE TOGGLE	TC	Toggles between modes
	SET LOCATION	SL#	Sets VTR location; # = control track
	GET CODE MODE	GC	Transport's location code mode
	SET CODE	CD d	d = data; sets V-LAN timecode generator's timecode type (not included in Sirius Video's V-LAN controller)
	RESET	ZZ	Resets controller

The rest of this section details the V-LAN commands; the commands are grouped as in Table 5-2.

Note: In this chapter, a *transport* is defined as a motion control device, namely, those that record or play back audio and/or video signals.

Movement Commands

Table 5-3 summarizes V-LAN movement commands.

Table 5-3 Movement Commands

Command	Code	Use
PLAY	PY	Puts current transport into play.
PAUSE	PS	Toggles current transport between play and pause. To guarantee a pause, use SH0 (SHUTTLE command, speed 0).
STOP	ST	Stops current transport. Tape is unthreaded; standby is set to OFF.
EJECT	EJ	Ejects tape from current transport. This command does not work on parallel devices.
FAST FORWARD	FF	Puts current transport into fast forward.
FAST REWIND	RW	Puts current transport into fast rewind.
SHUTTLE	SH#	Shuttles current transport at the specified speed. Positive numbers shuttle forward, negative numbers shuttle reverse. The range is -9 to +9; 0 = paused.
JOG FORWARD	JF	Reverses tape in current transport one frame.
JOG REVERSE	JR	Advances tape in current transport one frame.
RECORD	RC	Puts current transport into record. To end a recording, use STOP (ST).

GOTO Commands

Table 5-4 summarizes V-LAN GOTO commands.

Table 5-4 GOTO Commands

Command	Code	Use
GOTO IN	GI	Transport searches to the previously stored inpoint.
GOTO OUT	GO	Transport searches to the previously stored outpoint.
GOTO PREROLL	GP	Transport is sent to the inpoint minus the specified preroll; the default preroll is 5 seconds.
GOTO LOCATION	GT#	Transport searches to the location specified by #.

Edit Commands

When you set up an edit, you can specify the devices to be rolled in various ways. You can specify the node numbers after the PERFORM, REHEARSE, or REVIEW commands, or use RECORD NODE or SOURCE NODE SELECT commands. The node to which the edit command is directed becomes the master node in the sequence.

Table 5-5 summarizes edit commands.

Table 5-5 Edit Commands

Command	Code	Use
PERFORM	PF#	# = nodes to sync with recorder (1-9) Performs the edit on the currently selected node. Tracks should be set up ahead of time. Edit length is the duration held in the recorder's duration register; this duration is rippled to all selected sources. To synchronize other nodes to the recorder, enter their node numbers after PF. When the recorder reaches its inpoint, all other transports will also be at their inpoints.
REHEARSE	RH#	# = nodes to sync with recorder (1-9) This command works the same way as the PERFORM command, except that no actual editing is done. The record transport enters EE mode on the selected tracks. This command or the REVIEW command can be used to create multiple deck sync rolls.
REVIEW	RV#	# = nodes to sync with recorder (1-9) This command works the same way as the REHEARSE command, except that the record transport does not go into selective EE at the edit point.
RECORD NODE SELECT	NR#	# = 1-31, separated by commas Allows the user to specify multiple record transports.
SOURCE NODE SELECT	NS#	# = 1-31, separated by commas Allows the user to specify multiple source transports.

Table 5-5 (continued) Edit Commands

Command	Code	Use
ROLL BEGIN	RB#	# = start time after preroll Allows the user to program the start time for rolling a particular transport. It is useful to use # to implement a delayed roll of some machines in a sequence. This command cannot be used on the master record node in a sequence.
ROLL END	RE#	# = start time after preroll Allows the user to program the end time for rolling a particular transport—for example, to allow some machines in a sequence to stop before the rest. This command cannot be used on the master record mode in a sequence.
EDIT ERROR	EE	Edit error is returned by the node selected with ND#. The returned range is +/-0 to 99 frames. A number other than zero before the edit point indicates bad control track, bad time code, bad sync reference, or an error in capstan bumping. Three data bytes are returned: -01, 01, and so forth, indicating how far off the transport is from the desired spot.
EDIT SYNC	SY#	Selects the sync mode (0, 1, or 2) used for tape speed override. Record lock mode (0) locks the sources to the recorder (default mode). Source lock (2) uses the source as the edit sync reference, which is useful on some devices when time-base correctors are not used or when unstable video is likely to feed the recorder during preroll. System mode (1) is used only for Ampex direct color machines. This mode uses the V-LAN's internal clock as the sync reference.
COLOR BUMP	CB	Bumps the internal color frame clock to the next color frame. If the color frame is off, correct the situation by issuing SY1 and then this command.

Table 5-5 (continued) Edit Commands

Command	Code	Use
OUT ON THE FLY	OF	Allows user to select the outpoint at any time during the edit. When OF is sent, the current tape location is put in the outpoint register, and the edit is stopped after the postroll.
END CONDITION	EC c	c = command to execute after edit Instructs the transport on what to do at the end of an edit. For example, the command EC PF causes the transport to perform the edit again and again until it is stopped. If the auto increment register contains a value, a continuous animator is created. A simple command such as STOP causes the current node transport to be stopped. If a continuous loop is created, you can easily stop it by sending another EC command that has no data. If nothing is set, all transports pause.
ABORT SEQUENCE	AB	Allows the user to abort an edit sequence. This command is the preferred method for aborting edits.

Wait Mode Commands

Wait mode is useful when you want to cue up a transport but not roll it until another command is received. When wait mode is on, transports cue up and display the ready status. Sending WG (WAIT GO) starts the edit sequence. If you want consistent frame timing, send the WG command near the middle of the video frame.

If the edit sync is set to the system mode with the SY command, the transport is always at a specific location after the WG command is received, giving the transport time to synchronize.

Table 5-6 summarizes wait mode commands.

Table 5-6 Wait Mode Commands

Command	Code	Use
WAIT ON	WN	Enables wait mode
WAIT OFF	WF	Disables wait mode (the default)
WAIT GO	WG	Starts roll from the wait position

Edit Point Setup Commands

Table 5-7 summarizes V-LAN edit point setup commands.

Table 5-7 Edit Point Setup Commands

Command	Code	Use
NODE ADDRESS	ND#	# = node to talk with Sets the current node address. This node number specifies the device to which subsequent commands are sent. During editing, this command selects the record transport.
TRACK SELECT	TS m	m = mode: V = video; 1,2 = audio; A = assemble Selects the track to use on the next edit. The codes need not be in any specific order. The track selection is set for each node individually; if only one node is recording, be sure to set tracks for it. Default: no tracks selected.
SET IN	SI#	# = location Sets the inpoint register.
SET OUT	SO#	# = location Sets the outpoint register.
SET DURATION	SD#	If the duration is changed, the output changes. If the output is changed, the duration changes. Setting two of the registers automatically sets the third.

Table 5-7 (continued) Edit Point Setup Commands

Command	Code	Use
TRIM INPOINT	TI#	# = amount to trim inpoint Trims the inpoint by the specified amount.
TRIM OUTPOINT	TO#	# = amount to trim outpoint Trims the outpoint by the specified amount.
TRIM DURATION	TD#	# = amount to trim duration Trims the duration by the specified amount.
PREROLL SET	PR#	# = length of preroll (default 5 seconds) Do not set preroll to 0. A recommended setting for preroll is at least 1 second.
POSTROLL SET	PT#	# = length of postroll (default 1 second) Do not set postroll to 0. A recommended setting for postroll is at least 20 frames.
AUTO INCREMENT	AI#	Sets the amount that inpoints and outpoints are automatically incremented.
ENTER IN	EI	Sets the current tape location as the inpoint; default is 0.
ENTER OUT	EO	Sets the current tape location as the outpoint.
READ IN	RI	Returns the inpoint (12 bytes).
READ OUT	RO	Returns the outpoint (12 bytes).
READ DURATION	RD	Returns the duration (12 bytes).
CLEAR EVENT	CL	Clears all registers for the selected node (IN, OUT, DURATION, ROLL START TIME, ROLL STOP TIME).
ADD TRACKS	AT t	t = tracks: V = video, 1-9 = audio; more than one can be specified. This command is used with the SS command (see below).

Table 5-7 (continued) Edit Point Setup Commands

Command	Code	Use
SET SPLIT	SS#	# = record tape location where split starts The tracks selected by the AT (ADD TRACKS) command are added in at this point. The number should be between the recorder's inpoint and outpoint.
NEW TRACK SELECT	NT t#	t = tracks; # = time after preroll Selects a new track during an edit. Enter the track to change to and the time after the preroll point. You can set up to 5 track changes. Send commands in order of execution, with the lowest time value first, highest last. To reset the table of track changes, send NT with no data.

Status Commands

Table 5-8 summarizes V-LAN status commands.

Table 5-8 Status Commands

Control	Code	Use
STATUS REQUEST	SR#	# = optional node, ASCII status returned Returns a 12-byte ASCII status string indicating machine status. Possible status messages are `Power off`, `Stop`, `Threading`, `Unthreading`, `Wait`, `Play`, `Pause`, `Shuttling`, `Braking`, `Searching`, `Fast Forward`, `Rewind`, `Ready`, `Editing`, `Local`, `Ejected`, `Record`, `Var Play`, `Calibrating`, `Rehearse`, `Review`, and `Off Line`. If the transport had an error, the word `error` is returned, followed by a code number: 1 Execution error 2 Control track error 3 Search abort 4 Illegal command 5 Illegal search command 6 Synchronization error 7 Servo lock error 8 Not ready to search
LOCATION REQUEST	LR#	# = optional node, location returned Returns a 12-byte ASCII string representing the location in standard SMPTE format. If the number is positive, the first byte is a space. If the number is negative, the first byte is a minus sign (-). If the time code on the tape is drop frame code, the colon before the frames value becomes a period. When a tape is paused, its longitudinal time code (LTC) is unreadable. Thus, the still frame tape location depends on the type of VTR used. The only way to guarantee accuracy in a still frame is by using VITC time code.
LOCATION CONT	LC	Returns the location continuously until the next <Enter>. This command is recommended for terminal mode only, with echo on.

Table 5-8 (continued) Status Commands

Control	Code	Use
LOCATION STATUS	LS#	# = optional node, LOCATION and STATUS returned Returns a 12-byte location string and a 12-byte status string.
STATUS ENCODED	SE#	# = optional node First, this command returns a 12-byte ASCII location string. Next, it returns a byte indicating whether or not the transport is currently reading a valid time code. An ASCII space character means that a valid time code is not being read; an asterisk (*) means that a valid code is being read. The last byte of the string is the hex code for the transport status, as follows: 80 Blank 81 Power Off 82 Stopped 83 Threading 84 Unthreading 85 Wait 86 Play 87 Pause 88 Stick 89 Shuttling 8a Cruise 8B Braking 8C Searching 8D Fast Forward 8E Rewind 8F Ready 90 Editing 91 Local 92 Ejected 93 Record 94 Var play 95 Calibrate 97 Rehearse 98 Review 99 Tone FF Offline 01 Execution error 02 Control track error 03 Search abort 04 Illegal command 05 Illegal search command 06 Synchronization error 07 Servo error 08 Not ready to search

Table 5-8 (continued) Status Commands

Control	Code	Use
EDIT STATUS	ES	Returns a 12-byte edit status string indicating the stage of the edit. The possible codes are CUEING, EDITING, ABORTED, and DONE.
APPLICATION TYPE	AP	Returns a number between 0 and 6 indicating the type of device, as follows: 0 Offline 1 Transport 2 Video switcher 3 Keyboard 4 Audio switcher 5 Relay box 6 Universal serial control box
USER BIT RETURN	UB	Returns 8 bytes of user bit data.

Frame Grab Commands

Table 5-9 summarizes V-LAN frame grab commands.

Table 5-9 Frame Grab Commands

Command	Code	Use
COINCIDENCE ON	CO	Enables the coincidence character ($) return on the serial line.
COINCIDENCE OFF	CF	Disables coincidence checking over the RS-232-C line. This is the default setting on power-on.
SET COINCIDENCE	SC#	Selects the time for the coincidence pulse ($) to be sent to Sirius Video. The number entered is the delay time from the cue point.

In setting the coincidence pulse, if the preroll is 5:00, for example, and you want the coincidence pulse to occur one frame before the inpoint, set the coincidence time register to 4:00. The controller sends the coincidence character about 3 msec after vertical. If the frame-grab pulse is required one

frame before the inpoint, select 4:29 as the coincidence time for 525 timing or 4:24 for 625 timing (preroll 5 seconds in each case).

Validity Check Commands

Table 5-10 summarizes V-LAN validity check commands.

Table 5-10 Validity Check Commands

Command	Code	Use
CODE VALID	CV	Returns Y if the time code is valid; returns N if it is invalid.
DEVICE TYPE	DT	Returns five bytes, of which the first three are the device type code and the last two are the version number.
SYNC CHECK	SN	Returns Y if a valid sync is connected to the controller; returns N if it is not connected.

Sync Play Commands

Table 5-11 summarizes V-LAN sync play commands.

Table 5-11 Sync Play Commands

Command	Code	Use
ROLL	RL	Rolls and syncs the current node to the master running frame count. The inpoint for the current node should be set to where you want the synchronization. When the transport receives this command, it is cued to its preroll point, where it remains until its time matches the running frame count time. When the times match, the device is rolled and synced to the controller running frame count. The machine plays until stopped with the ER (END ROLL) command.
END ROLL	ER	Ends the sync roll for the current device and pauses it. Although you can stop the device with direct motion commands, this command is preferred for this purpose.
SET FRAME COUNT	SF#	Sets the running frame count. The number is loaded into the running frame count. This clock is updated by the sync signal supplied to the V-LAN controller so that it stays locked to the system clock.
READ FRAME COUNT	RF	Returns the current value of the V-LAN's frame count sync play commands.

Relays for GPSI (General-Purpose System Interface) Box

The GPSI box controls external video equipment, such as videotape controllers and switchers. Table 5-12 summarizes GPSI commands.

Note: Valid relays are 1-16; 0 = all relays.

Table 5-12 GPSI Commands

Control	Code	Use
RELAY SET	RS#,#,..	Sets the specified relays; the numbers must be separated by commas.
RELAY RESET	RR#,#,..	Resets the specified relays; the numbers must be separated by commas.
RELAY PULSE	RP#,#,..	Pulses the pin corresponding to the specified relays. If the relay was set originally, the pulse is reset, set. If the relay was reset originally, the pulse is set, reset.
SET PULSE TIME	SP#	Sets the length of the pulse to the specified time (number of frames).
CLEAR RELAYS	CR	Clears relay events; that is, all triggers set by the RT (RELAY TRIGGER) command (see below).
RELAY TRIGGER	RT#, t	Allows you to program relays within an event for a specific time. When this command is entered, a log is made of the relay number, the time after preroll point, and the relay type (set, reset, or pulse). The LAN determines the relay type according to the last RS, RR, or RP sent. For example: `RT5, 00:00:00:10` This command sets relay 5 to go 10 frames from time 0. If RS# was sent before this command, the next RT command "sets" the corresponding pin for the specified number at the specified time (set mode).

Note: To set the mode, send an RS, RR, or RP command with no number. Sending one of these commands is advisable (though not required) before you send RT to ensure that the correct relay type is logged to the appropriate trigger.

Slow Motion Commands

Table 5-13 summarizes V-LAN slow motion commands.

Table 5-13 Slow Motion Commands

Command	Code	Use
VARIABLE SPEED PLAY	VP #	# = speed between -99 and +99 frames/second Puts device into variable speed mode at indicated speed. If the device does not support that speed, it uses the closest available speed.
VARIABLE SHUTTLE	VS #	# = shuttle speed between -31 and +31 Shuttles a device over its speed range. Speeds are device-dependent and give the range of the device.
RETURN CURRENT VARIABLE SPEED	VR	Returns 3 bytes of speed data (-01, 01). Returns current slow motion speed. Useful in conjunction with the VARIABLE SHUTTLE command to determine selected speed.
TEST IF VARIABLE SPEED AVAILABLE	TV	Returns Y if the device supports variable speed control, N if it does not.
CHECK AVAILABILITY OF SPEED	CA #	# = speed to check Returns the selected speed or the closest speed to the speed selected that the device supports (3 bytes of data).

Table 5-13 (continued) Slow Motion Commands

Command	Code	Use
SET VARIABLE SPEED	SV #	# = speed for edit Sets the speed used in the edit. Setting to the play speed value (NTSC: 30; PAL: 25) resets variable speed edit mode. The CLEAR ALL command also resets speed to play speed.
VARIABLE SPEED	VC st	s = speed, t = time Changes edit speed during an edit. You can set up to 7 speed changes. To set speeds, send commands in order of execution, with the lowest time value first, highest last. To reset the table of track changes, send VS with no data.

Miscellaneous Commands

Table 5-14 summarizes miscellaneous V-LAN commands.

Table 5-14 Miscellaneous Commands

Command	Code	Use
ECHO ON	EH	Turns echo on so that all characters and linefeeds sent to the controller are also sent back.
ECHO OFF	EF	Turns echo off, which is the case on device power-on. Leave echo off when you use your computer to control the system.
EE ON	YE	Turns EE mode on for the current transport. Selects video input to a transport as the video out.
EE OFF	NE	Turns EE mode off for the current transport.
CODE MODE SELECT	CS#	# = 1 (time code) or 0 (control track) On power-on, the transport is in time code mode. CS0 disables time code reading; CS1 resets the device to time code.
CODE TOGGLE	TC	Toggles between time code and control track modes.

Table 5-14 (continued) Miscellaneous Commands

Command	Code	Use
SET LOCATION	SL#	Sets the transport to the specified location; if the time code is read, this number is overwritten.
GET CODE MODE	GC	Returns the location code mode of the selected transport: 0 = control track 1 = time code 3 = VITC time code
SET CODE	CD d	d = data For NTSC video mode, sets the time code type: 0 = nondrop 1 = drop frame
RESET	ZZ	Resets the on-board V-LAN controller; same as toggling power off and on. This command does not return <Enter>.

Sample V-LAN Sequences With *sir_vlan.c*

This section consists of two sequences, one for performing a single-frame edit and one for digitizing an image from tape. Each uses the Sirius Video V-LAN tool *sir_vlan.c*, whose contents are printed in Appendix E, "Example Programs." See also the reference page for *sir_vlan.c* for an additional example.

Performing a Single-Event Edit

To perform the edit, follow these steps. Terminate each command with <Enter>.

1. Select the node, for example:

   ```
   sir_vlan -c "ND 3"
   ```

2. Clear edit information:

   ```
   sir_vlan -c "CL"
   ```

3. Select the tracks:

   ```
   sir_vlan -c "TSV "
   ```

4. Select the preroll, for example:

   ```
   sir_vlan -c "PR 5:00"
   ```

5. Set the inpoint, for example:

   ```
   sir_vlan -c "SI 00:00:16:30"
   ```

   Leading zeroes are not necessary.

6. Set the duration to one frame:

   ```
   sir_vlan -c "SD 1"
   ```

7. Perform the edit:

   ```
   sir_vlan -c "PF"
   ```

Digitizing an Image From Tape

To digitize an image from tape, follow these steps. Terminate each command with <Enter>.

1. Select the node, for example:

   ```
   sir_vlan -c “ND 3”
   ```

2. Select the tracks:

   ```
   sir_vlan -c “TSV “
   ```

3. Turn coincidence on:

   ```
   sir_vlan -c “CO”
   ```

4. Set preroll to the default:

   ```
   sir_vlan -c “PR 5:00”
   ```

5. Set the coincidence time:

   ```
   sir_vlan -c “SC 4:29”
   ```

6. Set the inpoint, for example:

   ```
   sir_vlan -c “SI 00:00:16:30”
   ```

   Leading zeroes are not necessary.

7. Set the duration to one frame:

   ```
   sir_vlan -c “SD 1”
   ```

8. Review the edit:

   ```
   sir_vlan -c “RV”
   ```

When the frame corresponding to the coincidence time is reached, the Sirius Video integral V-LAN controller issues a frame grab pulse to the Sirius Video hardware; the hardware must be preprogrammed to receive a frame grab trigger event.

Chapter 6

Using Sirius Video Utility and Demonstration Programs

Sirius Video utility and demonstration programs make it possible to use Sirius Video for performing certain tasks with little or no programming. This chapter explains

- displaying video on the workstation monitor using *vidtogfx*
- saving video to disk using *sir_vidtomem*
- displaying saved video on the workstation using *movie*
- displaying captured frames or live video on a television monitor
- loading video from disk using *sir_memtovid*
- manipulating video using *sirius_distort*
- manipulating video using *shatter*

Displaying Video on the Workstation Monitor Using *vidtogfx*

To display video on the workstation monitor, use *vidtogfx*, a Sirius Video utility program included in *usr/dmedia/bin/SIRIUS*. *vidtogfx* continuously captures a stream of live video from a video source and displays it via the Iris GL in a window on the workstation monitor.

You can set the video source on the panel (*vcp*) or with a *vidtogfx* option. To use *vidtogfx* parameters, use *vlinfo* to get the device number. For example, *vlinfo* might display:

```
name of server:
number of devices on server: 1

device: Sirius 0
       nodes = 12
       Sirius Device, type = Device, kind = 0, number = 0
       Sirius Digital Video Source 1, type = Source, kind = Video, number = 0
       Sirius Digital Video Source 2, type = Source, kind = Video, number = 1
       Sirius Analog Video Source, type = Source, kind = Video, number = 2
       Sirius Memory Source, type = Source, kind = Memory, number = 0
       Sirius Graphics Source, type = Source, kind = Graphics, number = 0
       Sirius Chroma Key Generator, type = Internal, kind = No ui for this,
number = 0
       Sirius Blender, type = Internal, kind = Blender, number = 0
       Sirius Video Drain, type = Drain, kind = Video, number = 0
       Sirius Memory Drain, type = Drain, kind = Memory, number = 0
       Sirius Graphics Drain, type = Drain, kind = Graphics, number = 0
       Sirius Texture Drain, type = Drain, kind = Texture, number = 0
```

For example, to display video from a D1 source, type

```
vidtogfx -n 0
```

Note: To adjust values in the video source, see Appendix C, “Setting Up Sirius Video for Your Video Hardware,” in this guide. Appendix E, “Example Programs,” gives the complete syntax and code for *vidtogfx.c*.

Saving Video to Disk Using *sir_vidtomem*

You can use *sir_vidtomem*, a Sirius Video utility program included in *usr/dmedia/bin/SIRIUS*, to save a number of frames from a video source, such as a videotape or live video input, as RGB files on disk. To do so, follow these steps:

1. Make sure the input device—video camera or videotape—is functioning and connected to the Sirius Video breakout box.
2. In the *vcp* Utilities pull-down menu, select "Live Video Input." The live video window appears.

   Note: If there is a problem with the display in the live video window, the cause might be lack of sync. Select "Live Video Input" a second time.
3. Use the panel to set parameters for source and drain as needed.
4. To display the frames in a window on the workstation as a test before you save them, type

   ```
   sir_vidtomem -c# -F1 -v# -w -d
   ```

   where

   - `-c#` is the number of frames you want to save
   - `-F1` specifies frames (not fields)
   - `-v#` specifies the input node corresponding to the port on the breakout box, such as Digital 1 (`0`), Digital 2 (`1`), or Analog (`2`); the default is Digital 1
   - `-w` disables writing to files
   - `-d` displays the video in a window

   Note: For complete command-line options for *sir_vidtomem*, see Appendix E, "Example Programs," later in this guide.

   For example, to display 60 frames from a source on parallel digital channel 1, type

   ```
   sir_vidtomem -c60 -F1 v 0
   ```
5. To save a number of frames to disk (in the */usr/dmedia/bin/SIRIUS* directory), type

   ```
   sir_vidtomem -c# -F1 -v#
   ```

For example, to save 60 frames from an analog source, type

```
sir_vidtomem -c60 -F1 -v2
```

Note: While *sir_vidtomem* is grabbing fields, the displayed video does not move smoothly. After the frames are grabbed, the video moves smoothly while the frames are being written to disk.

Loading Video From Disk Using *sir_memtovid*

You can take advantage of the Sirius Video capability to bring in video in various formats and output it in a different format by using *sir_memtovid*, a Sirius Video utility program included in *usr/dmedia/bin/SIRIUS*. For example, D1 input that was saved as *.rgb* files can be output to tape as NTSC.

The arguments for *sir_memtovid* are similar to those for *sir_vidtomem*. Specify files with *-n filenameprefix*. Use *-l* to run the frames continuously in a loop.

For example, to play all the frames starting with *test* in a continuous loop, type

```
sir_memtovid -F1 -l -N test
```

You can use *vcp* to specify source and drain parameters, as for *sir_vidtomem*.

Displaying Saved Video Using *movie*

You can use the program */usr/people/4Dgifts/iristools/imgtools/movie* to display saved video frames or fields in a window on the workstation monitor.

To play all files starting with a certain prefix, type

```
movie filename*
```

For example, `movie test*` plays all files starting with the prefix `test`.

After the program has loaded the files into memory, you can use its menu to select “Loop” for repeat continuous play.

Displaying Live Video on a Low-Resolution Monitor

To send live video through the workstation out to a television monitor, follow these steps:

1. If necessary, connect a live video source and use *vcp* to set up Sirius Video as explained in Appendix C, "Setting Up Sirius Video for Your Video Hardware."
2. Call up the panel:

   ```
   /usr/sbin/vcp
   ```

3. In the Video Drain section of the *vcp*, set format and timing for the destination television monitor as necessary. Make sure the controls on the television monitor are congruent with the format (timing) you have set for the video drain.
4. Select "Live Video Input" in the Utilities menu. The Live Video Input window appears, displaying video from an input source that is connected.
5. Select "Live Video Output" in the Utilities menu. The Live Video Output window title and white outline appear.
6. Move the Live Video Output window over the Live Video Input window.
7. Select "Send Screen" in the Live Video Output window menu to begin sending frames.

   Note: To send almost the entire screen (1280 x 960), select "Send Full Screen."

Manipulating Video using *sirius_distort*

You can use the *sirius_distort* program, which is included in */usr/dmedia/bin/SIRIUS*, to manipulate live video. Follow these steps:

1. Use *sir_memtovid* to display the video frames you want to manipulate.
2. In the IRIX shell, call up *sirius_distort*. The window for *sirius_distort* appears.
3. Move the *sirius_distort* window so that it covers the live video output screen.
4. Manipulate the image:
   - Use the right mouse button to select Ripple or Rubber effect.
   - Hold down the left mouse button and move the mouse to distort the image.

If desired, output the results to tape or disk following instructions in "Saving Video to Disk Using sir_vidtomem" or "Displaying Live Video on a Low-Resolution Monitor," respectively.

Manipulating Video using *shatter*

You can use the *shatter* program, included in */usr/dmedia/bin/SIRIUS*, to manipulate live video. Follow these steps:

1. Use *sir_memtovid* to display the video frames you want to manipulate.
2. In the IRIX shell, call up *shatter*. The window for *shatter* appears.
3. Move the *shatter* window so that it covers the live video output screen.
4. Manipulate the image:
   - Hold down the left mouse button and move the mouse to move the image left, right, up, or down, or to rotate it.
   - Hold down the middle mouse button and move the mouse to zoom the image in or out.
   - Use the space bar to release the ball that shatters the image.

Appendix A

Technical Specifications

This appendix details hardware specifications for Sirius Video:

- compatibility
- CP interface
- VME interface
- analog input and output channel specifications
- video formats
- host connector specifications
- Sirius Video breakout box connectors
- Sirius Video connectors and controls

Compatibility

Sirius Video is compatible with the NTSC and PAL television standards. It is also compatible with CCIR 601 and SMPTE 125M digital video standards.

CP Interface

CP interface bandwidth is 40 MHz, at up to 48 bits wide. You can choose between 8 and 10 bits per component.

VME Interface

The VME interface on the Sirius Video board initializes and configures the hardware and, in master mode, transfers video to and from system memory. A Sirius Video application can send live video to raster or texture memory in the graphics subsystem, or it can send the video to host memory over the VME bus. Regardless of input format, the video stream within Sirius Video is 4:4:4:4 RGBa/YUVa, 10 bits per component.

With the exception of video frame buffers, all other Sirius Video board devices can be accessed by programmed I/O (PIO) cycles. Supported VME PIOs are A16, D32, read and write, nonprivileged access; address modifier is 0x29. Except for the VME controller registers, any Sirius Video board device can be accessed using the DMA transfer (read or write) cycle.

Four different interrupt levels are supported; the four upper bits of the status and ID registers are software-programmable.

In its default slave mode, the VME interface initializes and configures the Sirius Video hardware.

VME Master Mode

The interface can be programmed to select VME master mode, perform 32-bit (D32) or 64-bit (D64) DMA transfers to or from host memory over the VME bus, and return to VME slave mode. The VME master can be programmed to give up the bus when other VME masters request it (Release On Request) and finish its assigned tasks later when the bus is available again, or to hold onto the bus until its tasks are finished (Release When Done).

For multiple-pixel transfers, Sirius Video supports pixel packing and unpacking. Transfer rate is ~28 MB/second for 32-bit transfers and ~55 MB/second for 64-bit transfers. D64 mode is available only on Sirius Video installed in Onyx and CHALLENGE systems.

VLISTs

The VME interface includes VLISTs, which are FIFOs implemented in local memory on the Sirius Video board. The host can put commands for managing the state of the Sirius Video board into the VLISTs, thus freeing the host from having to manage the Sirius Video board state directly during vertical blanking.

Because a dedicated hardware controller performs the writes during vertical blanking, greater data rates are possible than if the host performed this task directly over the VME bus. VLIST commands are in the form of writes to Sirius Video board resources, such as command registers and lookup tables.

Analog Input and Output Channel Specifications

Table A-1 lists some general technical specifications for analog input and output channels. and genlock jitter.

Table A-1 Analog Input and Output Channel Specs

Specification	Channel	Value
Input impedance	Input	50 KOhm; 5 pF capacitance
Frequency response	Input	+/- 0.25 dB to 1 MHz; +/- 0.5 dB to 5.4 MHz
	Output	+/- 0.25 dB to 1 MHz; +/- 0.5 dB to 5.2 MHz with sin X/X compensation
Group delay	Both types	+/- 20 nsec to 5.5 MHz
Return loss	Input	>40 dB to 5.5 MHz
Analog S/N ratio	Input	>55 dB, unweighted 5.0 MHz (GBR or YUV)
Analog S/N ratio	Output	>63 dB, typical unweighted 5.0 MHz (GBR or YUV)
Line-locked genlock sync jitter	Output genlock	<10 nsec

Frequency response, transient response, signal-to-noise (S/N) ratio, and crosstalk are all included for both input and output sections, including the breakout box. The measurements were made with a Tektronix model 2465A (with TV trigger) oscilloscope and a Tektronix VM700A video measurement

set. Analog test signals were generated by Tektronix generator models TSG-130A, TSG-131A, and TSG-100; and digital test signals were generated by Tektronix generator model TSG-422.

This section discusses analog input and output measurements separately.

Analog Input Measurements

This section discusses

- frequency response
- transient (K-factor) response
- signal-to-noise ratio
- input genlock jitter
- input genlock signal-to-noise ratio and crosstalk
- composite input

Frequency Response

The input signal is an analog 700mVp-p frequency sweep, from 0.5 MHz to 5.75 MHz. The input operating mode is analog RGB, with 625 timing.

The passband ripple measures 0.50 dB p-p to 5.4 MHz. See Figure A-1 for a typical response. The sweep signal in the figure is measured at the input to the A/D converter and is more than 2.0 V p-p in amplitude.

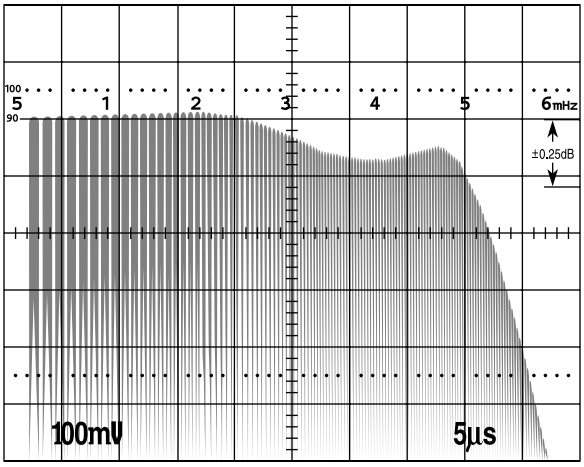

Figure A-1 Input Filter Frequency Response

Transient (K-Factor) Response

The input signal is an analog 700mVp-p (PAL) 2T Pulse with a HAD of 200 nsec. The input operating mode is analog RGB, with 625 timing.

The K-factor measurements ranged from 1.1-1.3% and averaged1.2% for a typical response; see Figure A-2. For an HAD of 250 nsec(NTSC), all channels would be <1.0%.

The K-factor measurements meet a <1% figure for NTSC 2T pulses.

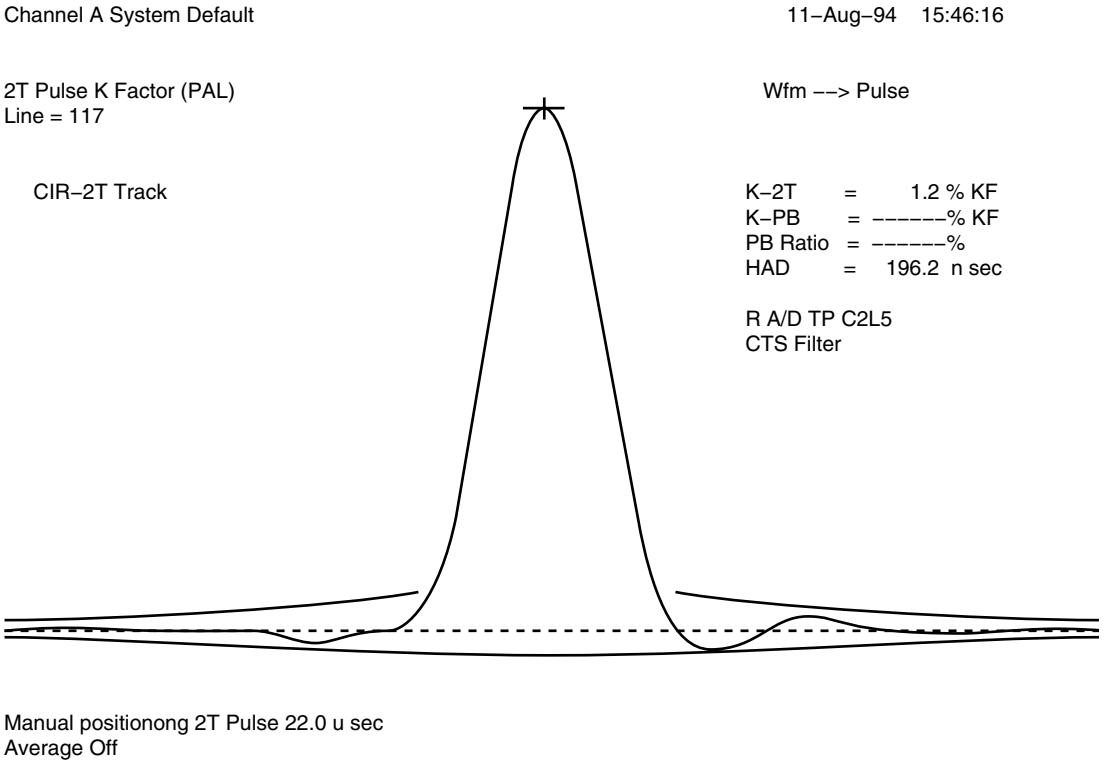

Figure A-2 K-Factor Measurements

Signal-to-Noise Ratio (S/N Ratio)

The S/N ratio is measured at the input to the A/D converters. This includes the effects of the breakout box on the signals. Noise components can result from analog or digital crosstalk in the breakout box and on the Sirius VME board, and from thermal, excess, or popcorn noise sources within the circuitry.

First, measurements were taken with crosstalk effects eliminated as much as possible. A 50% gray flat field was applied to one channel at a time with no signals applied to the D1 inputs or to the inputs of the channels not being tested. Finally, 5.75 MHz full amplitude sweeps were applied to all analog input except for the input under test; simultaneously, D1 4:2:2:4 5.75 MHz

full amplitude digital sweeps were applied to the D1 inputs. This second case is a worst-case scenario that does not represent normal operating conditions.

The S/N results summarized in Table A-2 were obtained with the VM700A and active probe modified for variable gain. A bandwidth of 100kHz-5.0 MHz was used with no weighting. The A/D input test points were the point of measurement. The specification is >=65 dB, weighted.

Table A-2 S/N Ratio, A/D Inputs, No Crosstalk, 100kHz-5.0 MHz, No Weighting

Channel	Green	Blue	Red	Alpha
S/N	66.8 dB	66.6 dB	66.4 dB	66.5 dB

Typical S/N ratio with crosstalk included is 55 dB.

Input Genlock Jitter

A horizontal rate signal from the input genlock phase-locked loop was measured for jitter using a Hewlett-Packard time domain analyzer. This instrument measures peak-to-peak jitter and standard deviation jitter; it was set to collect data on 1000 samples.

The maximum peak-to-peak jitter was 9.8 nsec, with a standard deviation of 0.65 nsec. It was estimated that 95% of the samples had a peak-to-peak jitter of less than 4.0 nsec. These are good numbers for an H-rate genlock system.

Input Genlock Signal-to-Noise Ratio and Crosstalk

The input genlock circuit contains a mux that selects from the various possible genlock sources. Both the S/N ratio and crosstalk were measured by applying multiburst signals to the composite, R, G, B, alpha, and sync inputs. The house (genlock) input was selected as the output genlock source, but no signal was applied to this input. The input to the genlock A/D converter was then measured for S/N ratio, which includes crosstalk.

The resulting S/N ratio was 76 dB over a bandwidth of 100 kHz to 5.0 MHz, no weighting, and with averaging on. This indicates that there is a very high S/N ratio with negligible crosstalk.

Figure A-8 shows input genlock S/N ratio; crosstalk is worst-case and not comparable to normal operation. Multiburst was applied to the applied to R, G, B, A, composite, and sync inputs. On the mux, "house" was selected, with nothing connected to that input.

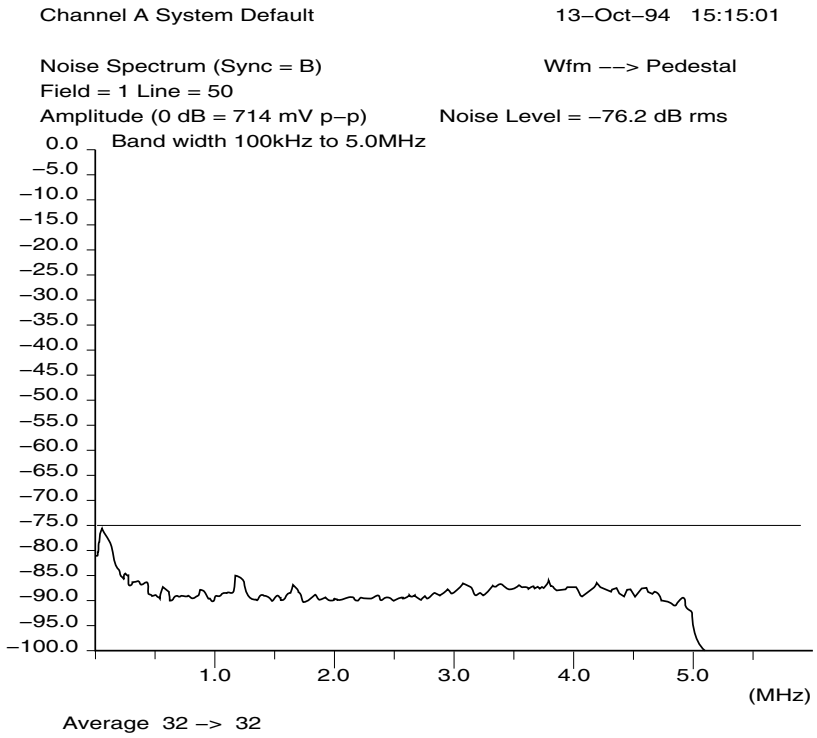

Figure A-3 Input Genlock S/N Ratio and Crosstalk

Composite Input

The quality of the composite and Y/C inputs was measured by inputting composite analog test signals and measuring the composite output signal. This technique measures the combined performance of signal decoding and encoding. The output signal quality was measured separately and is summarized later. The frequency response and S/N ratio using the Y/C inputs would be even better, due to the absence of chroma/luma separation. The encoder and decoder only process 8-bit video.

An NTSC multiburst signal was applied to the composite input (and to all other inputs) and measured on the composite output; Figure A-4. The frequency response is quite flat to beyond 2.0 MHz. Crosscolor, which is a normal effect, is visible on the last three bursts. The level of the signal, as well as the shape and amplitude of sync and burst, are correct.

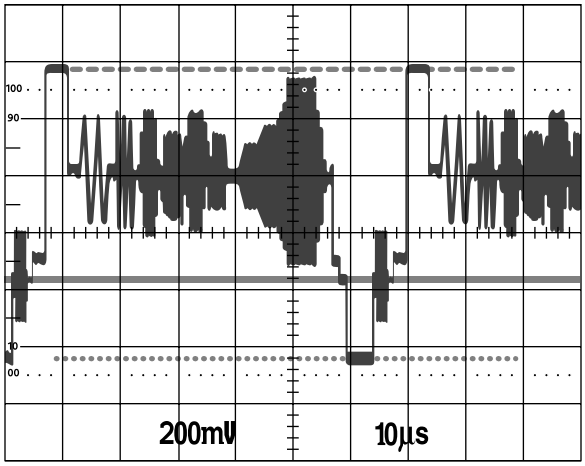

Figure A-4 Composite In—Composite Out: TSG-100 Sweep Input, Color Mode on Decoder

Figure A-5 shows the decoded and encoded 75% color-bar signal (TSG-100). Figure A-6 shows the vector display of this color-bar signal. Except for the slightly hot subcarrier level, the vector accuracy is within all of the small boxes, with negligible offset.

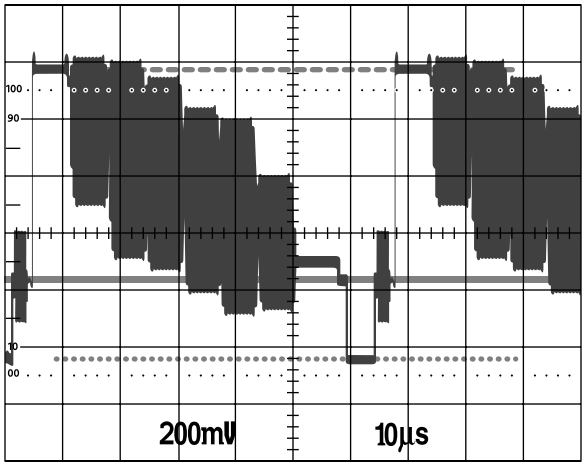

Figure A-5 Composite In—Composite Out: Decoded and Encoded 75% Color Bars

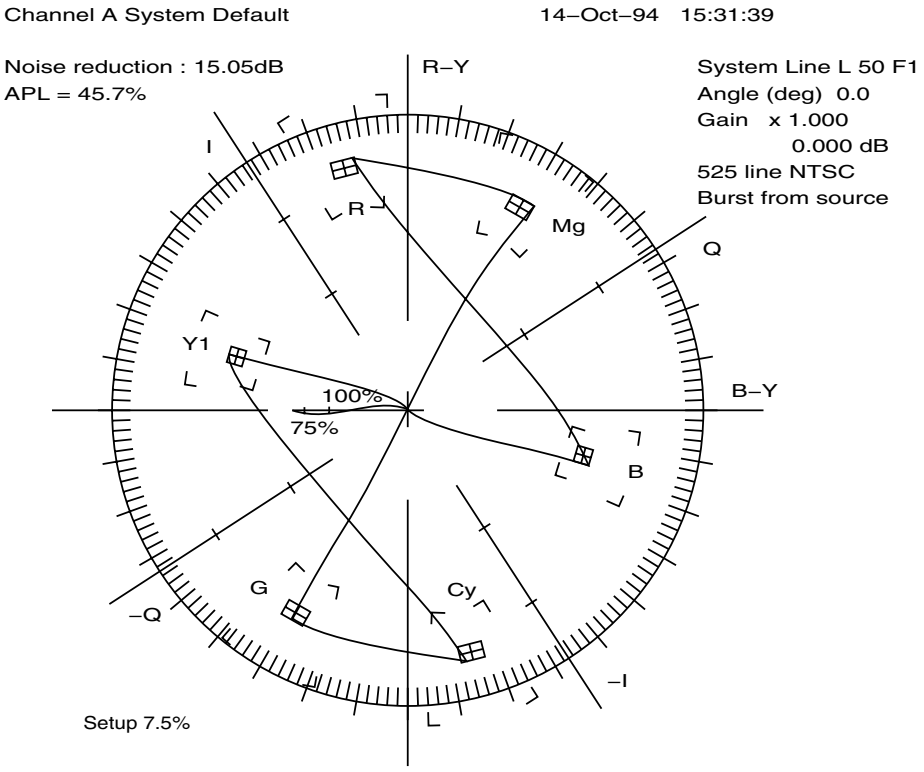

Figure A-6 Composite In—Composite Out: Decoded and Encoded 75% Color Bars

The S/N ratio of the composite in-out path was measured using a 50% gray field from a TSG-100 generator, and making the measurement with a VM-700A. Figure A-7 shows the results. A S/N ratio of -49 dB was obtained with a bandwidth of 10 0 kHz-5.0 MHz and no weighting.

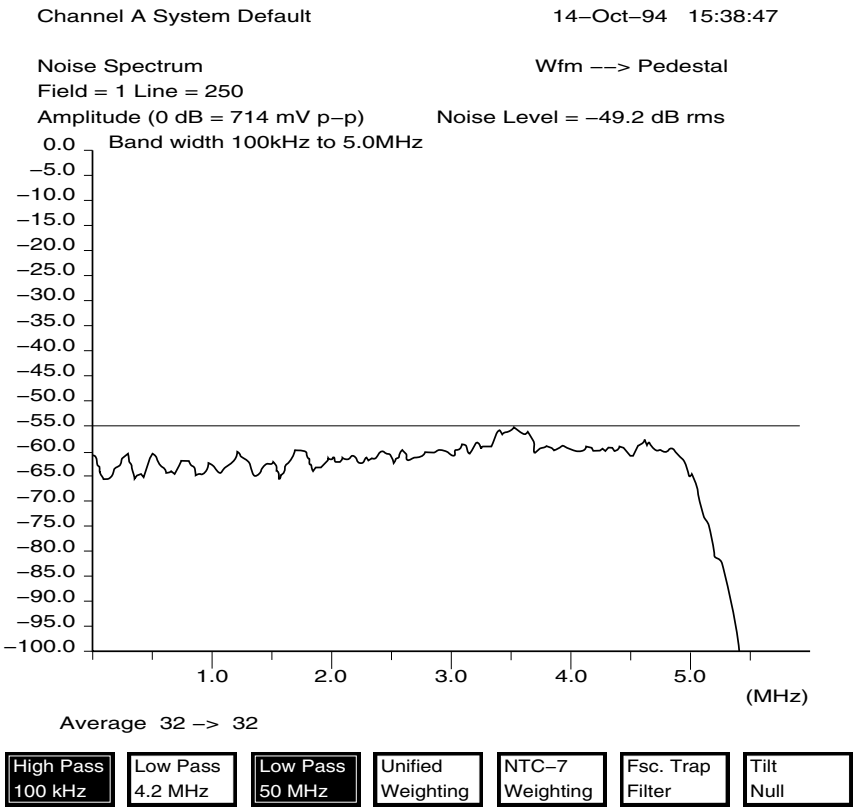

Figure A-7 Composite In—Composite Out: S/N Ratio

Analog Output Measurements

This section discusses

- frequency response
- transient (K-factor) response
- signal-to-noise ratio
- output sync waveforms
- output jitter

- output genlock crosstalk
- composite output
- output timing

Frequency Response

D1 4:2:2:4 video in, with measurements made on the G, B, R, and alpha output channels of the breakout box. The input operating mode is parallel 4:2:2:4 input with CCIR601 525 timing. The output operating mode is RGB output, also with CCIR601 525 timing. The D1 test signal is a full amplitude 5.75 MHz sweep signal. The RGB output sweep was measured on an oscilloscope with signal-to-noise ratio amplitude markers set for a 0.5 dB p-p spread.

The measured response under +-0.5 dB ripple conditions extends to 5.2 MHz with sinX/X correction, relative to 100kHz; see Figure A-8.

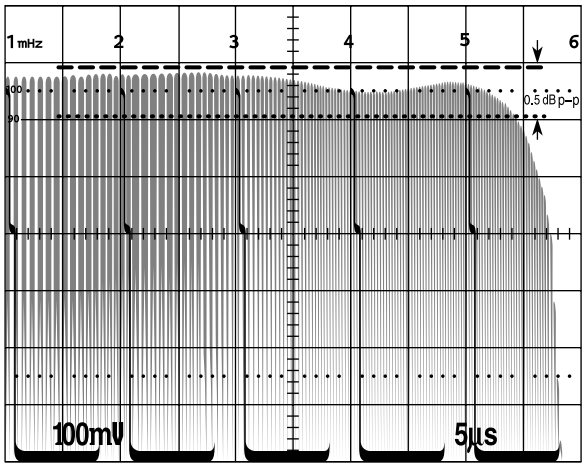

Figure A-8 Output Filter Frequency Response with SinX/X Compensation

The measured response under 1.0 dB p-p ripple conditions extends to 5.5 MHz, relative to 100 kHz. The specification is +-0.5 dB to 5.5 MHz. This response includes SinX/X correction for the 601 sample clocks. For the square pixel 525 mode, the response will be down by an additional 0.1 dB at

5.0 MHz; for the square pixel 625 mode, the response will rise by an additional 0.08 dB at 5.0 MHz.

The output frequency response typically meets the specifications.

Transient (K-Factor) Response

The input signal is a D1 4:2:2:4 PAL 2T Pulse with a HAD of 200 nsec. The output operating mode is analog RGB, with 625 timing.

The PAL K-factor measurements uniformly measured 0.8% for a typical response; see Figure A-9. For an HAD of 250 nsec (NTSC), all channels would be <0.8%.

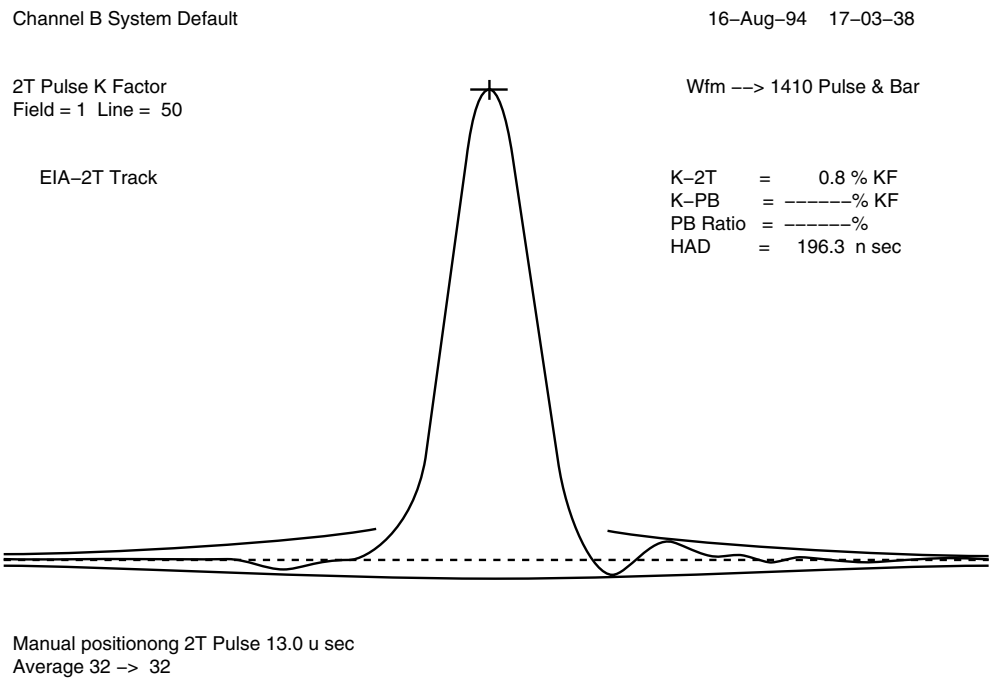

Figure A-9 Response under +-0.25 dB Ripple Conditions

The output K-factor measurements meet a <1% figure for both NTSC and PAL 2T Pulses.

Signal-to-Noise Ratio

Disregarding crosstalk, the inherent S/N ratio of any of the R, G, B, or alpha channels is 78dB, measured 100kHz to 5.0MHz, with no weighting. The effect of output crosstalk on S/N ratio was measured by outputting a 5.75MHz sweep signal on the RGB and Alpha outputs, but disabling the output lookup table on one channel at a time, and measuring the S/N ratio of that channel. A bandwidth of 100 kHz to 5.0 MHz and no weighting was used. The results are summarized in Table A-3.

Table A-3 Inherent S/N Ratio

GB Alpha into R	GR Alpha into B	BR Alpha into G	GBR into Alpha
65.2 dB	63.8 dB	68.4 dB	56.7 dB

Note: These crosstalk conditions are worst-case and not comparable to normal operation, particularly because most alpha channel signals will not have the high-frequency content of the sweep signal.

Output Sync Waveforms

The output sync signal is available in either 300mVolt or TTL (4V) level, as controlled by software.

The 300mV version has a 75 Ohm output impedance and is measured into a 75 Ohm load. The level is very precise, with an accuracy of <3%, is Gaussian-shaped, and has a rise-fall time of 140 nsec (+-10%). Figure A-10, Figure A-11, and Figure A-12 show typical waveforms.

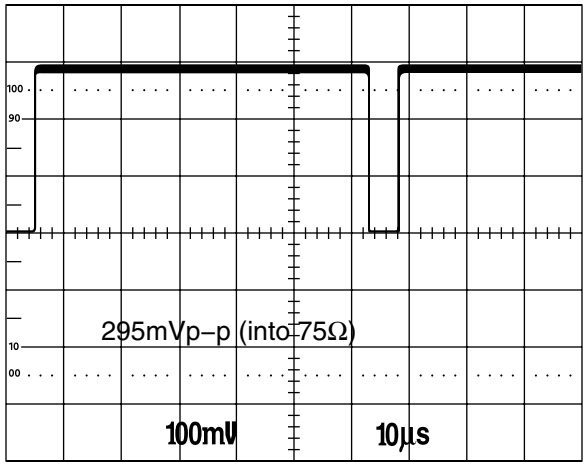

Figure A-10 Typical Waveform, Example 1

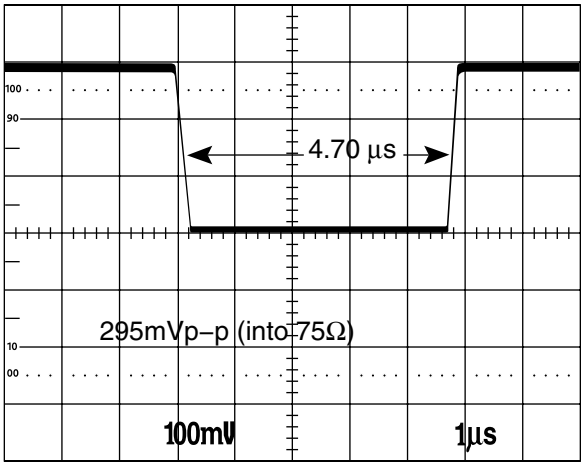

Figure A-11 Typical Waveform, Example 2

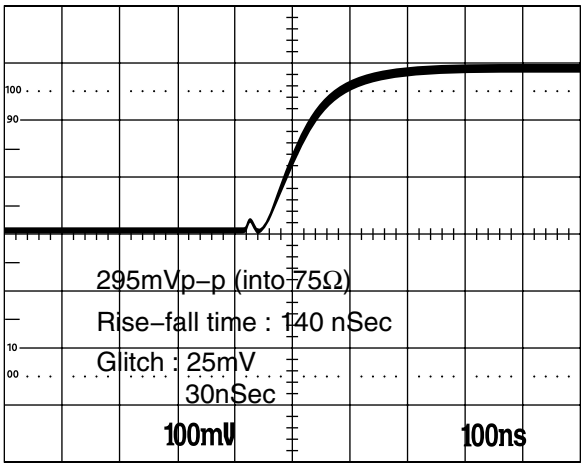

Figure A-12 Typical Waveform, Example 3

There is a small 25 mV 30 nsec glitch at the beginning of each transition; but this is unlikely to cause a problem with any external sync detector. The glitch is due to crosstalk around the TTL buffer that produces the 4V sync.

The 4 Volt sync output was designed to be terminated in a 75-Ohm load; with a typical amplitude of 4.5 V p-p +-.25V under these conditions. It has a rise-fall time of 18 nsec, and a very low output impedance. The DC level of sync tip is ground, the polarity is negative-going.

Caution: When operating in the 4V mode, it is important to terminate the sync output with a 75-Ohm load, or excessive ringing will appear due to the EMI filter on the output of the breakout box. Figure A-13, Figure A-14, and Figure A-15 show typical waveforms.

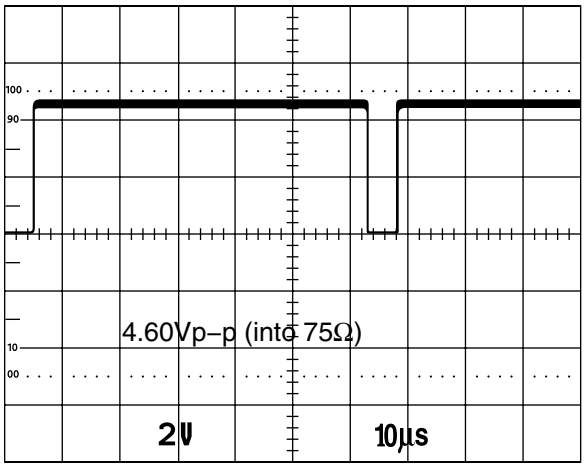

Figure A-13 Typical Waveform, Example 4

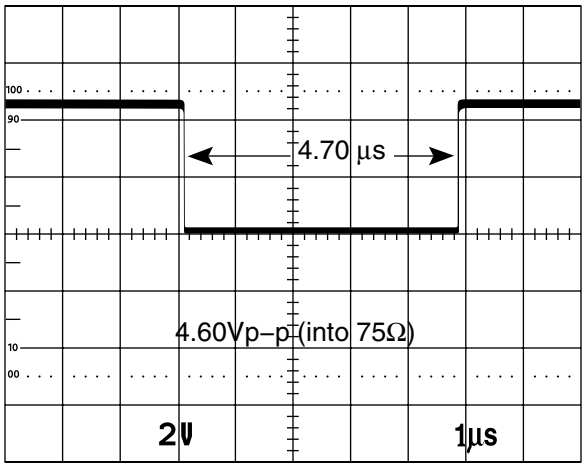

Figure A-14 Typical Waveform, Example 5

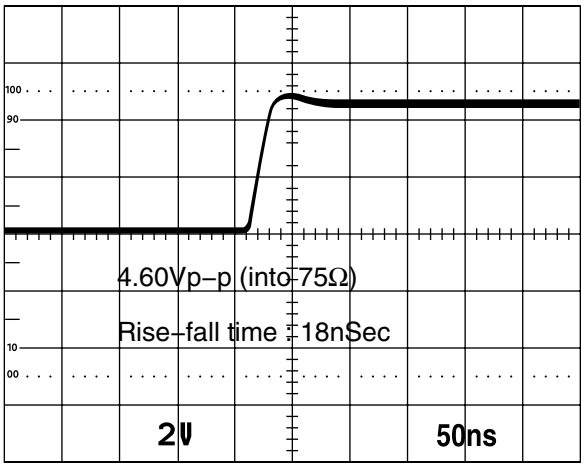

Figure A-15 Typical Waveform, Example 6

Output Jitter

The output jitter was measured with a Tektronix VM700A video measurement set by measuring the video and sync waveforms on the output of the breakout box, and by a time domain analyzer (TDA) that measures jitter on TTL logic waveforms. The VM700A has a minimum jitter of 3 to 4 nsec, when measuring a high-quality generator output signal. Jitter was measured with no additional filtering on the VM700A, and with a 1.5 MHz Gaussian filter inserted in the path.

The output genlock jitter is very low, at approximately 1nsec measured with the TDA. The results shown in Table A-4 were obtained with the VM700A, and are for the CCIR 601 525/625 modes, with essentially the same results obtained for the square pixel 525/625 modes.

Table A-4 Output Jitter

	Sync Out	G/Y Out	Composite Out
NTSC: No filtering	4 nsec	12 nsec	5 nsec
NTSC: Gaussian filter	-	5 nsec	4 nsec
PAL: No filtering	7 nsec	13 nsec	15 nsec
PAL: Gaussian filter	4-5 nsec	5 nsec	10 nsec

The **SYNC OUT** and **GBRA/YUVA COMPONENT SYNC** outputs on the breakout box have very low jitter when high frequency noise is filtered out. The jitter approaches the residual of 3-4 nsec. When correlated with the results of the TDA, the conclusion is that the non-filtered jitter is the result of high-frequency noise contaminating the video appearing at the breakout box outputs.

These conclusions also apply to NTSC composite output, which exhibits similar measurements as the Sync and component outputs. However, the PAL composite output has much higher jitter. This increased jitter is due to noise on the chroma output that occurs during the sync edge. The Y/C Luminance output signal does not have the noise, and does not exhibit the increased jitter. The noise on the chroma channel may be produced within the encoder chip.

By using the “square root of the sum of squares” formula, with a residual jitter measurement of 3.5 nsec and a measured jitter of 5 nsec, the output genlock jitter is calculated at 3.5 nsec. This is an acceptable figure for a line-locked system

Output Genlock Crosstalk

Output genlock crosstalk and S/N ratio were measured by applying full amplitude 5.75 MHz sweep signals to all inputs except for either sync or house (genlock) input. The S/N ratio was measured at the analog input to the genlock circuit, with a bandwidth of 100 kHz to 5.0 MHz, no weighting, and averaging on. The S/N ratio was measured both on line 17, where there could be no crosstalk because it is not an active line, and on line 50, where

there would be crosstalk. The output genlock mux was used to select either the sync or house input. Table A-5 summarizes these results.

Table A-5 Output Genlock Crosstalk

Genlock (House) Input		Sync Input	
Line 17	Line 50	Line 17	Line 50
67.3dB	64.7dB	67.0dB	59.3dB

These results are worst-case conditions not comparable to normal operation. The 59 dB figure, when the sync input is selected, is due primarily to equal crosstalk from the green and composite inputs. This figure indicates that the house input is the preferable one to use to minimize crosstalk.

Composite Output

The quality of the composite and Y/C outputs was measured by inputting digital 601 75% color bars and outputting composite color bars.

Frequency response was measured by inputting a digital 601 5.75 MHz sweep signal and viewing the composite output. The response is flat to beyond 5.0 MHz, as seen in Figure A-16.

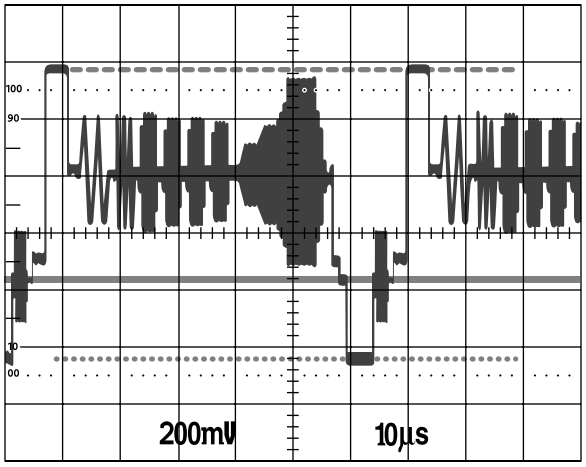

Figure A-16 Composite In—Composite Out: S/N Ratio

Digital 75% color bars were input to the system. In the composite output, the levels and appearance of the bars is correct, and the levels, shape, and timing of sync and burst are correct.

Figure A-19 shows the vector display of color bars, and shows a slightly hot subcarrier amplitude with excellent vector accuracy and negligible offset. A flat 50% gray field was applied to the digital 601 input and output on composite. Figure A-20 shows that the S/N ratio is 50 dB.

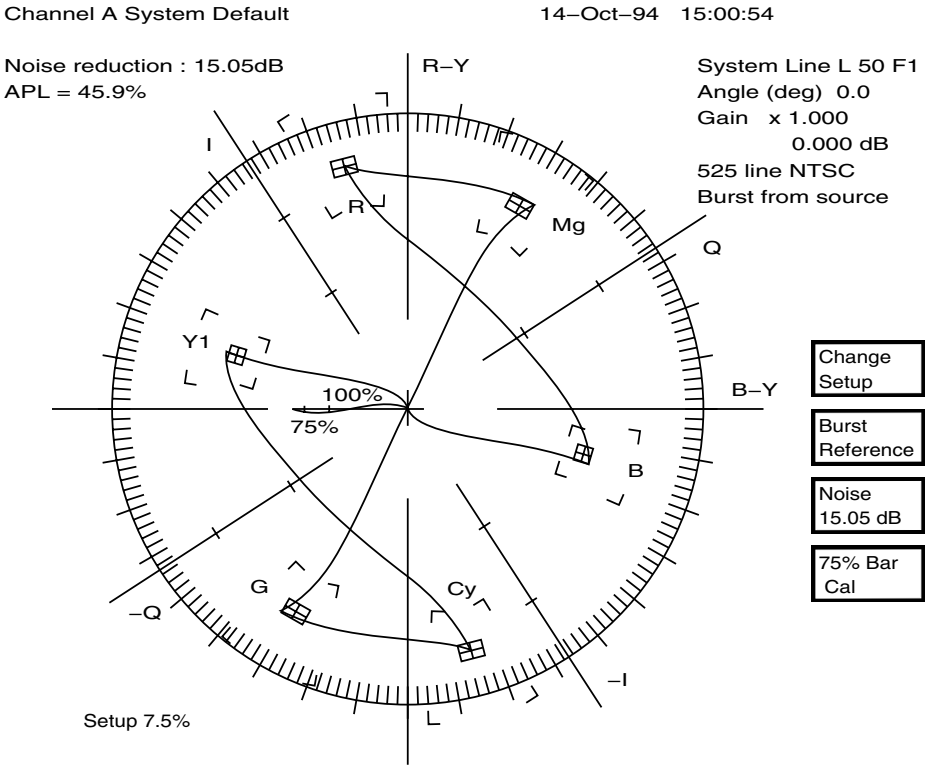

Figure A-17 Vector Display, Digital Color Bars

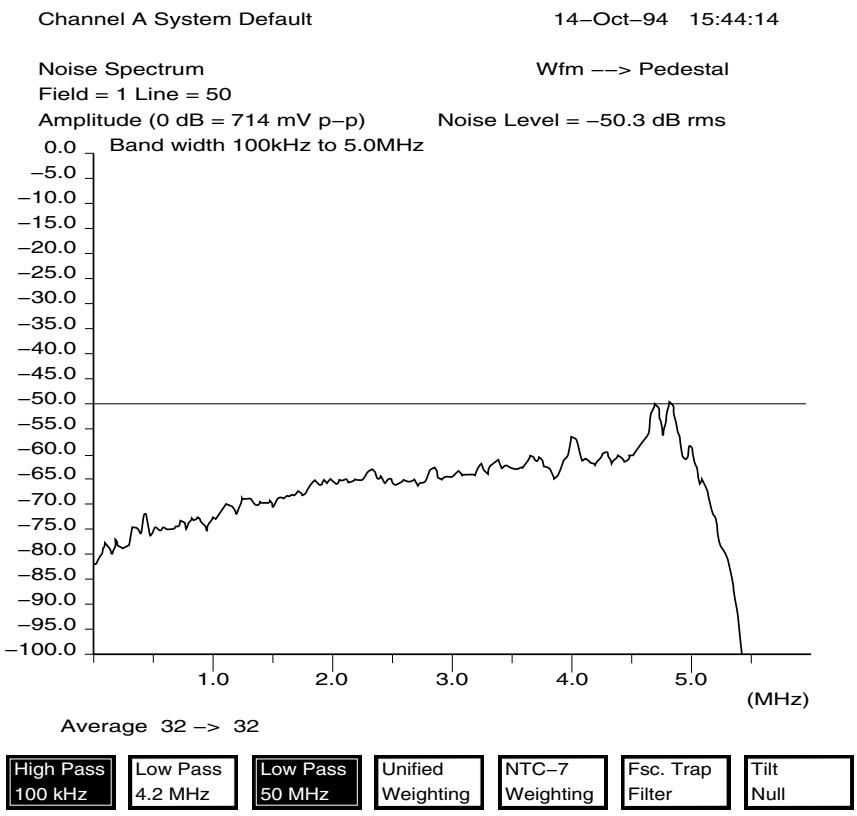

Figure A-18 S/N Ratio, Digital Color Bars

Output Timing

Figure A-19 and Figure A-20 show the output timing of green versus red, and green versus blue; and indicate that the channels are coincident. The input to the breakout box was composite color bars from a TSG-100 generator, and the output was RGB from the breakout box.

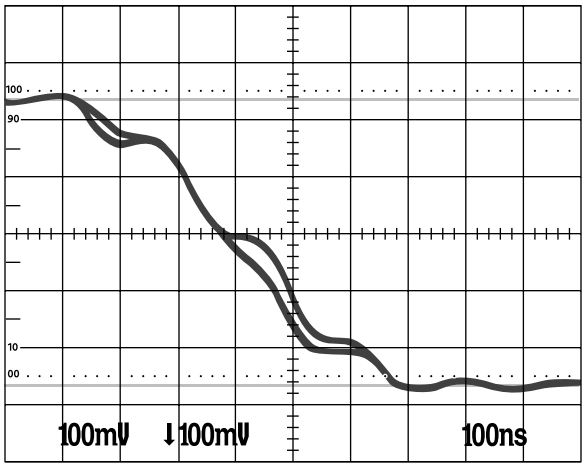

Figure A-19 Output Timing, Green Versus Red

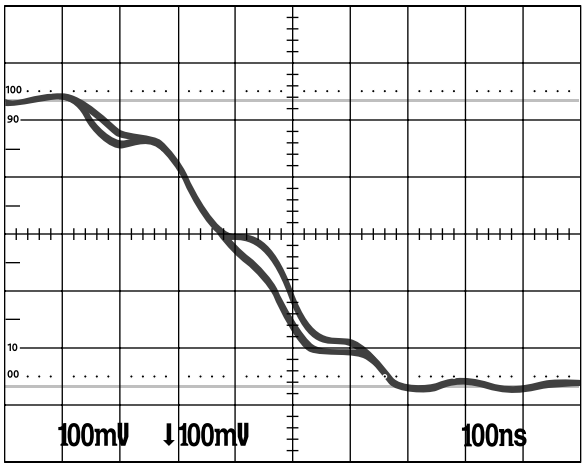

Figure A-20 Output Timing, Green Versus Blue

Video Formats

Table A-6 summarizes Sirius Video video formats.

Table A-6 Sirius Video Formats

Format	Video Stream	Tape Format
Digital component	4:4:4:4, 4:2:2:4, 4:2:2 in 525 and 625 timings	CCIR 601 (8/10 bits) SMPTE dual link (8/10 bits)
Analog component	RGBA, YUVA, PrYPbA in 525 and 625 timings	Betacam, M-II, SMPTE; 10-bit ADC/DAC; 2x oversampling output
Analog composite	525 and 625 timings	NTSC, PAL, S-Video

For more information, see the following standards, which contain provisions for video signals:

- CCIR 601-2: Encoding Parameters of Digital Television for Studios (4:2:2 component video signals, single link)
- ANSI/SMPTE 125M-1992: Television—Component Video Signal 4:2:2—Bit-Parallel Digital Interface
- SMPTE Recommended Practice (RP) 175-1993: Digital Interface for 4:4:4:4 Component Video Signals (Dual Link)
- SMPTE 259M, Television—10-Bit 4:2:2 Component and $4f_{sc}$ NTSC composite Digital Signals—Serial Digital Interface
- SMPTE RP 157-1990: Key Signals

Host Connector Specifications

Table A-7 summarizes the specifications for the host connections on the breakout box.

Table A-7 Breakout Box Host Connections

Connector Label	Definition/Specification
ANALOG 2 **ANALOG 1**	Analog cables to Sirius Video board via I/O door on workstation
DIGITAL 1 **DIGITAL 2** **DIGITAL 3**	Digital cables to Sirius Video board via I/O door on workstation
Power **AC 100-240 1A, 47-63Hz**	Autoranging to accept voltage range shown

Sirius Video Breakout Box Connectors

Figure A-21 shows the Sirius Video breakout box.

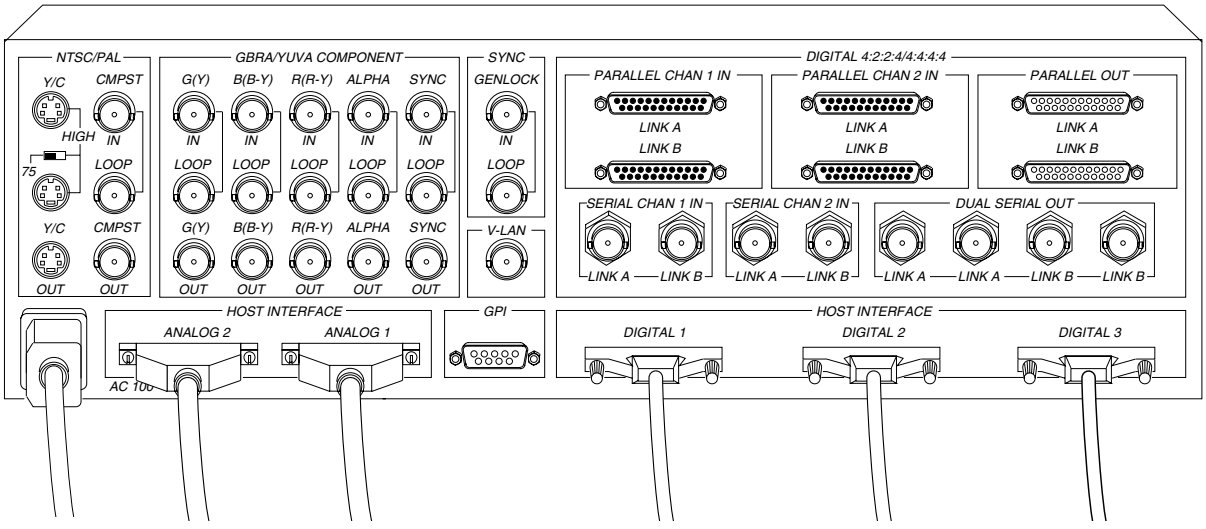

Figure A-21 Sirius Video Breakout Box

Table A-8 summarizes the breakout box connectors and switches that interface with video equipment.

Table A-8 Interface for Video Equipment

Section	Connector or Switch	Description
NTSC/PAL	**Y/C (loop)**	S-Video source (tape deck or camera)
	75/HIGH	Self-termination switch for S-Video input
	CMPST IN	Composite video input from tape deck or camera
	LOOP	Loopthrough for each composite video input, with buffered signal to workstation
	Y/C OUT	S-Video output to tape deck or monitor
	CMPST OUT	Composite video destination (tape deck or monitor)
GBRA/YUVA COMPONENT	**G(Y) IN** **B(B-Y) IN** **R(R-Y) IN** **ALPHA IN** **SYNC IN**	Analog component (GBR or YUV plus alpha and sync) video source (tape deck or camera)
	LOOP	Loopthrough for each component video input, with buffered signals to workstation
	G(Y) OUT **B(B-Y) OUT** **R(R-Y) OUT** **ALPHA OUT** **SYNC OUT**	Analog component (GBR or YUV plus alpha and sync) video destination (tape deck or monitor)
SYNC	**GENLOCK IN**	External sync source (house sync)
	LOOP	Loopthrough for genlock input with buffered signal to workstation
V-LAN		V-LAN transmitter
GPI		Graphics Peripheral Interface source/destination (tape deck or digital recorder); see Chapter 4, “Controlling the General-Purpose Interface (GPI),” for details.

Table A-8 (continued) Interface for Video Equipment

Section	Connector or Switch	Description
DIGITAL 4:2:2:4/4:4:4:4	**PARALLEL CHAN [1,2] IN**	Parallel digital video source (digital tape deck or other recording device)
	PARALLEL OUT	Parallel digital video destination (digital tape deck or other recording device)
	SERIAL CHAN [1,2] IN	Serial digital video source, if optional serializer board (SD1) is installed in breakout box
	DUAL SERIAL OUT	Serial digital video destination, if optional serial board (SD1) is installed in breakout box
		Note: The transfer mode (packing format) selected determines Link A and Link B usage, as explained in Table A-9.

Table A-9 explains the use of **Link A** and **Link B** connectors in the **DIGITAL 4:2:2:4/4:4:4:4** section of the breakout box. Each carries 10-bit wide YUVA.

Table A-9 Transfer Mode Usage for Link A and Link B Digital Connectors

4:2:2:4 Link A	4:2:2:4 Link B	4:4:4:4 Link A	4:4:4:4 Link B
Cb_0	x	Cb_0	Cb_1
Y_0	A_0	Y_1	A_0
Cr_0	x	Cr_0	Cr_1
Y_1	A_1	Y_1	A_1

If Link B is not used in 4:2:2:4 format, the resulting format is 4:2:2

Sirius Video Connectors and Controls

Table A-10 summarizes Sirius Video video connectors.

Table A-10 Video Connectors.

Input	Output
2 pairs of serial/parallel digital (CCIR 601, SMPTE dual link, optional serial I/O)	2 serial/parallel digital (CCIR 601, optional serial I/O)
1 analog component (RGBA, YUVA, PrYPbA)	1 analog component (RGBA, YUVA, PrYPbA)
1 analog composite (NTSC, PAL, S-VHS)	1 analog composite (NTSC, PAL, S-VHS)

Sirius Video controls are

- two GPI input lines
- two GPI output lines
- V-LAN interface
- genlock/sync input
- loopthroughs for all analog inputs

Appendix B

Sirius Video Equipment Configurations

This appendix gives two sample configurations for various combinations of video equipment:

- Sirius Video GPI port, switch closure mode
- Sirius Video output to Abekas A65/A66 input

Sirius Video Input Configuration

Figure B-1 shows an example input configuration: the Sirius Video GPI port in switch closure mode.

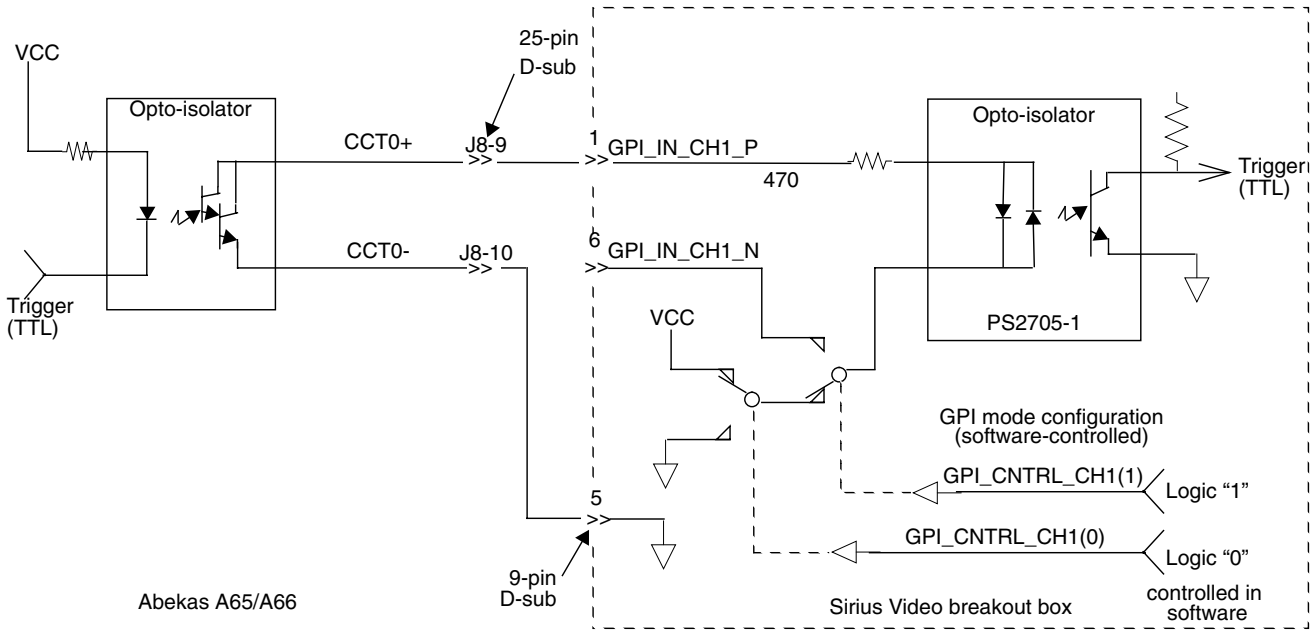

Figure B-1 Example Configuration: Sirius Video GPI Port, Switch Closure Mode

Note: The GPI input port does not support differential mode configuration.

Sirius Video Output Configuration

Figure B-2 shows an example output configuration: Sirius Video GPI output to Abekas™ A66 input.

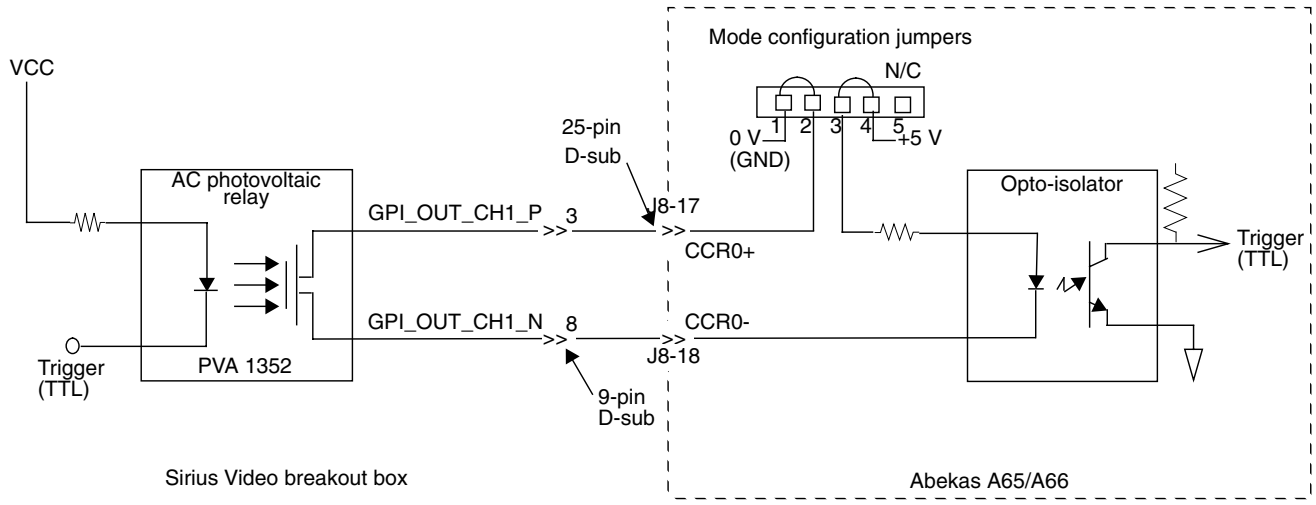

Figure B-2 Example Configuration: Sirius Video GPI Output to Abekas A65/A66 Input

Note: In Figure B-2, CCR refers to the Abekas Contact Closure Receiver.

To interface to other equipment, observe the electrical specifications for the output stage of the international rectifier PVA 1352, as listed in Table B-1.

Table B-1 PVA 1352 Electrical Specifications

Specification	Value
Operating voltage (AC/DC)	+/-100 V peak
Load current (DC)	315 mA (maximum at 40 degrees C)
Off-state resistance	10^5 ohms at 25 degrees C
On-state resistance	5 ohms at 25 degrees C
T_{on}/T_{off} (maximum) (50 mA load/as mV control, 50 VDC)	300µs/50µs
Output capacitance	15 pF (at 50 VDC)

Appendix C

Setting Up Sirius Video for Your Video Hardware

This appendix illustrates how to connect your video equipment to connectors on the Sirius Video breakout box and how to use the video control panel *vcp* to set the Sirius Video board to match your installation.

This appendix explains

- setting up digital source video
- setting up analog source video
- setting up an external sync source
- setting up the output (drain)
- adjusting graphics source or drain timing
- adjusting texture drain timing
- saving settings

Setting Up Digital Source Video

Sirius Video has two 10-bit digital video ports for equipment that complies with the CCIR 601 standard. Each port supports a serial interface; these ports are on the optional serial digital board for the breakout box.

The ports can be configured for 4:4:4:4 or 4:2:2:4 dual-link mode; for 4:2:2 single-link mode, ignore the alpha.

Each port consists of two unidirectional interconnections, Link A and Link B:

- In 4:4:4:4 mode, Link A carries Y plus the U and V from even-numbered sample points; Link B carries alpha plus the U and V from odd-numbered sample points.
- In 4:2:2:4 mode, Link A carries Y plus the U and V from even-numbered sample points; Link B carries alpha only.

To set up for a digital video source, follow these steps:

1. Connect the digital source equipment into the appropriate socket on the breakout box.

   Figure C-1 points out sockets for single-link mode.

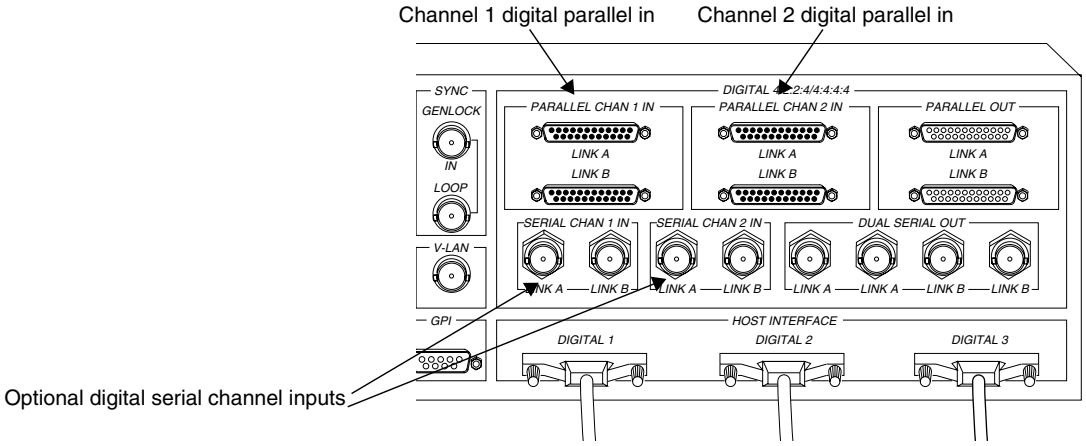

Figure C-1 Digital Video Input Ports for Single-Link Mode

Figure C-2 points out sockets for dual-link mode.

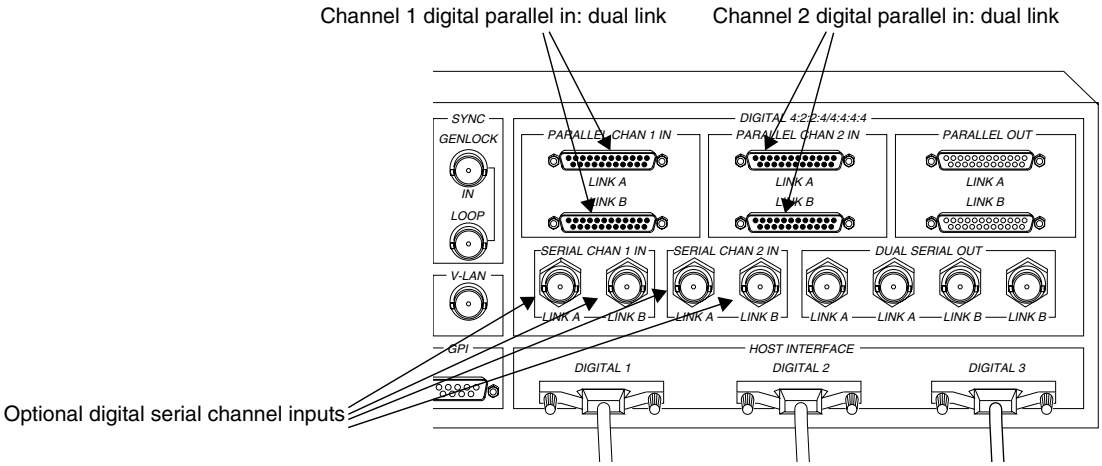

Figure C-2 Digital Video Input Ports for Dual-Link Mode

2. Call up the panel:

   `/usr/sbin/vcp`

3. In the Digital Video Source section of the control panel *vcp* for the channel(s) you are using, select the format that matches your equipment, as shown in Figure C-3.

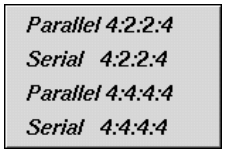

Figure C-3 Selecting Digital Input Video Format in *vcp*

4. In the Digital Video Source portion of the panel for the channel(s) you are using, select the timing that matches your equipment: CCIR 525 or CCIR 625.

5. To adjust horizontal phase for each digital video channel and its separate alpha line, if used, select “Digital Video Source 1” or “Digital Video Source 2” in the Pro menu, and then select “Signal Controls,.” The Sirius Digital Video Source [1, 2] window appears, as shown in Figure C-4.

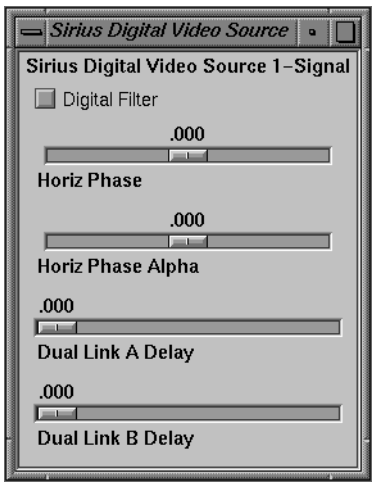

Figure C-4 Digital Video Source Signal Controls

The windows for Digital Video Source 1 and Digital Video Source 2 are identical.

6. To enable conversion of 4:2:2:4 to 4:4:4:4 input, check the Digital Filter check box.

7. If you are using a dual link for 4:4:4:4, use the Dual Link Delay slider to adjust the delay of one link relative to the other to compensate for delay differences between the links.

Setting Up Analog Source Video

To set up for an analog source, such as Betacam, M-II, S-Video, or NTSC/PAL, follow these steps:

1. Connect the analog source equipment into the appropriate socket on the input row of the analog side of the breakout box, as shown in Figure C-5.

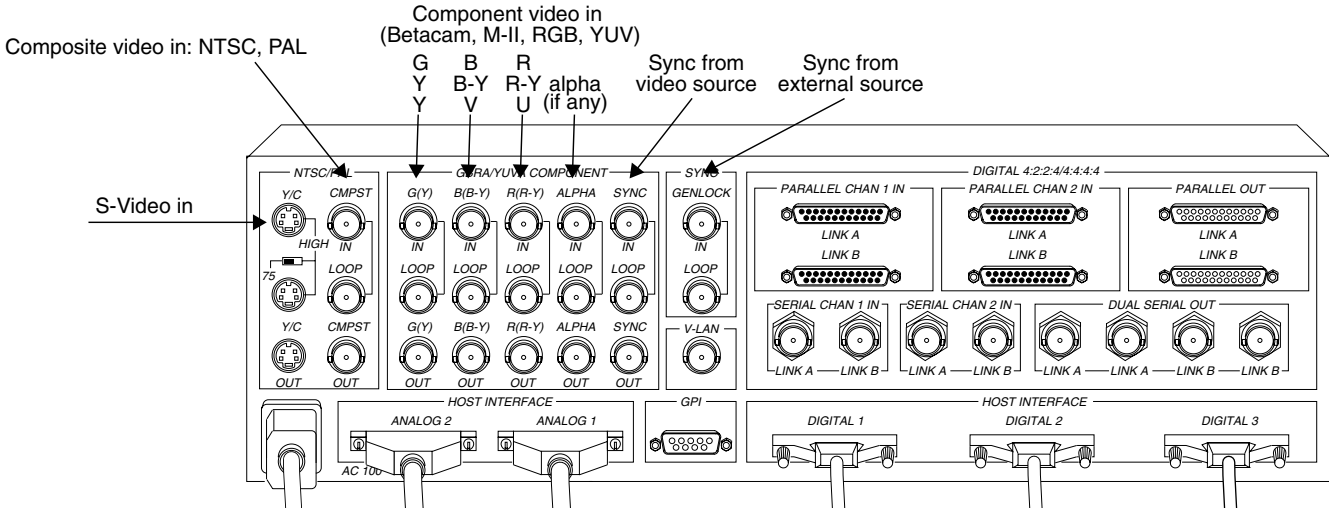

Figure C-5 Analog Input Ports on the Sirius Video Breakout Box

2. If necessary, call up the panel (**`/usr/sbin/vcp`**).
3. In the Analog Video Source portion of the control panel *vcp*, select the format that matches your equipment, as shown in Figure C-6.

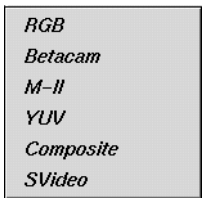

Figure C-6 Selecting Analog Input Video Format in *vcp*

4. In the Analog Video Source portion of the panel, select the timing that matches your equipment: 525, 625, CCIR 525, or CCIR 625.

   Note: In the case of large amounts of jitter, use CCIR 525 instead of 525, and CCIR 625 instead of 625.

5. If you are syncing to the analog input, make sure the sync signal is cabled to the **SYNC IN** port on the breakout box, as shown in Figure C-5.

   In the Analog Video Source portion of the panel, you can select the genlock sync source that matches your equipment, as shown in Figure C-7.

   Note: To genlock to an external sync source, see "Setting Up an External Sync Source," later in this appendix.

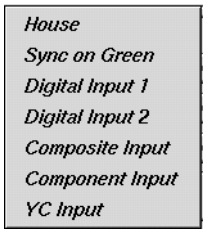

Figure C-7 Selecting Input Genlock Sync

Table C-1 and Figure C-8 summarize the correspondence between these menu choices and breakout box connectors.

Table C-1 Genlock Sync Menu Choices and Breakout Box Sockets

Genlock Sync Menu Choice	Breakout Box Socket
House	**SYNC** **GENLOCK**
Sync on Green	**GBRA/YUVA COMPONENT** **G(Y)**
Digital Input 1	**PARALLEL CHAN 1 IN** or **SERIAL CHAN 1 IN**
Digital Input 2	**PARALLEL CHAN2 IN** or **SERIAL CHAN2 IN**

Table C-1 (continued) Genlock Sync Menu Choices and Breakout Box Sockets

Genlock Sync Menu Choice	Breakout Box Socket
Composite Input	**NTSC/PAL** **CMPST**
Component Input	**GBRA/YUVA COMPONENT** **SYNC**
YC Input	**NTSC/PAL** **Y/C**

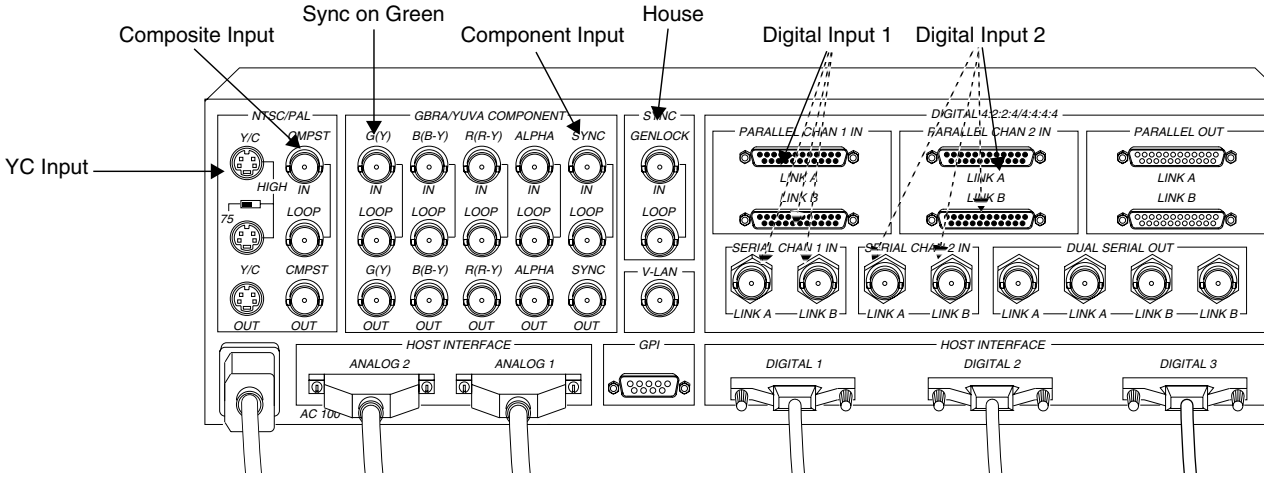

Figure C-8 Genlock Sync Menu Choices and Breakout Box Sockets

Note: If you experience problems genlocking to unstable video sources such as VCRs, VTRs, and laser disc players, try routing the video output from these devices into an external time base corrector, and feed the time base corrector's output into the corresponding connector on the Sirius Video breakout box.

6. To set the voltage level for the genlock sync source, use the *vcp* Pro menu. In the Pro menu, select "Device Controls", and then select "Synchronization Controls." The Sirius Device-Sync Input Controls window appears, as shown in Figure C-9.

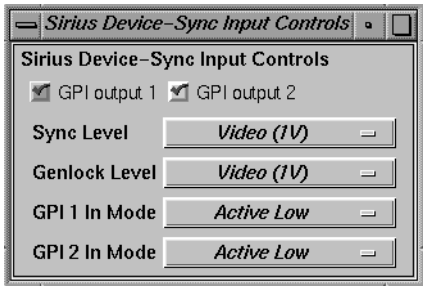

Figure C-9 Setting Genlock Voltage Level

7. To set the voltage level, select Video (1 V peak-to-peak) or TTL (4 V peak-to-peak) in the Sync Level menu item.
8. Use the sliders to adjust the horizontal phase for the picture.
9. To set field dominance or adjust horizontal phase for the video and its separate alpha, if any, select "Analog Video Source" in the Pro menu, and then select "Signal Controls." The Sirius Analog Video Source-Signal window appears, as shown in Figure C-10.

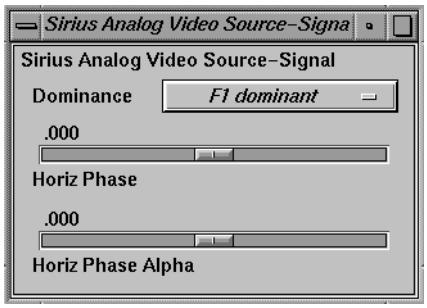

Figure C-10 Setting Analog Video Source Field Dominance or Horizontal Phase

To set field dominance, select at the "Dominance" menu item:

- "F1 dominant": the edit occurs on the nominal video field boundary
- "F2 dominant": the edit occurs on the intervening field boundary

Figure C-11 shows fields as defined for NTSC and PAL.

Note: In digital formats and other input modes, half lines become full lines.

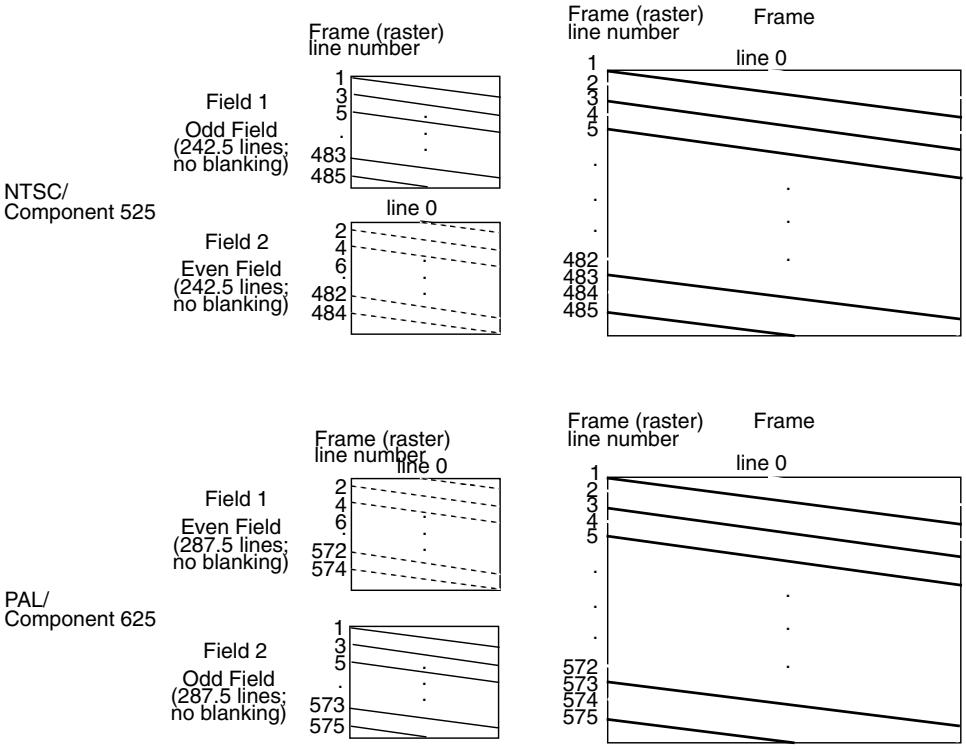

Figure C-11 Fields and Frames for NTSC and PAL

Users typically want to edit on Field 1 boundaries, where Field 1 is defined as the first field in the video standard's two-field output sequence. 525 standards send the second (whole) raster line out to begin the first field, and the first (half) raster line out to begin the second field; 625 standards send the first (half) raster line out to begin the first field, and the second (whole) raster line to begin the second field.

Some users may want to edit on F2 boundaries, which fall on the field in between the video standard's frame boundary. To do so, use this control, then program your deck to select F2 edits.

NTSC users might need to vary their field dominance choice, depending on the origin of the input material they are to edit.

Note: To output a set of frames, they must be deinterlaced into fields differently, depending on the choice of output field dominance. For example, when F1 dominance is selected, the field with the topmost line must be the first field to be transferred; when F2 dominance is selected, the field with the topmost line must be the second field to be transferred. Deinterlacing must be specified in the application; the Sirius Video example program *sir_memtovid.c* code contains an example of how to consult the field dominance control to determine deinterleave order.

```
/*
 * Set the memory node's timing based upon the video drain's timing,
 * which has been set up by the daemon from the defaults file, or by
 * the user via vcp.
 *
 * When we get around to reading image files, we'll check the file
 * size against the size reported by the VL for this node: if the file
 * size does not match the format's, we'll punt.
 */

if (vlGetControl(svr, MEMtoVIDPath, drn, VL_TIMING, &drainTiming) < 0) {
vlPerror("GetControl(VL_TIMING) on video drain failed");
exit(1);
}
if (vlSetControl(svr, MEMtoVIDPath, src, VL_TIMING, &drainTiming) < 0) {
vlPerror("SetControl(VL_TIMING) on memory source failed");
exit(1);
}
/*
 * Read the video drains's field dominance control setting and timing,
 * then set a variable to indicate which field has the first line, so that
 * readimage() will know how to deinterleave frames to fields.
 */
if (vlGetControl(svr, MEMtoVIDPath, drn,
VL_SIR_FIELD_DOMINANCE, &dominance) < 0) {
vlPerror("GetControl(VL_SIR_FIELD_DOMINANCE) on video drain failed");
exit(1);
}
```

```
is_525 = ((drainTiming.intVal == VL_TIMING_525_SQ_PIX)
       || (drainTiming.intVal == VL_TIMING_525_CCIR601));

switch (dominance.intVal) {
case SIR_F1_IS_DOMINANT:
    if (is_525) {
    F1_is_first = 0;
    } else {
    F1_is_first = 1;
    }
    break;
case SIR_F2_IS_DOMINANT:
    if (is_525) {
    F1_is_first = 1;
    } else {
    F1_is_first = 0;
    }
    break;
}

/*
 * Read the video drain's field dominance control setting and set a
 * variable to indicate which field has the first line, in case readimage()
 * needs to deinterleave frames to fields.
 */
if (vlGetControl(svr, MEMtoVIDPath, drn,
VL_SIR_FIELD_DOMINANCE, &val) < 0) {
vlPerror("GetControl(VL_SIR_FIELD_DOMINANCE) on video drain failed");
exit(1);
}

switch (val.intVal) {
case SIR_F1_IS_DOMINANT:
    F1_is_first = 1;
    break;
case SIR_F2_IS_DOMINANT:
    F1_is_first = 0;
    break;
}
```

To assemble fields to frames, the application must consult the field dominance control in order to determine the interleave order. the Sirius Video example program *sir_vidtomem.c* contains an example of how to consult the field dominance control to determine interleave order.

```
/*
 * Set the memory node's timing based upon the video source's timing,
 * which has been set up by the daemon from the defaults file, or by
 * the user via vcp.
 */
if (vlGetControl(svr, path, src, VL_TIMING, &timing) < 0) {
vlPerror("GetControl Failed");
exit(1);
}
if (vlSetControl(svr, path, drn, VL_TIMING, &timing) < 0) {
vlPerror("SetControl Failed");
exit(1);
}
/*
 * Read the video source's field dominance control setting and timing,
 * then set a variable to indicate which field has the first line, so that
 * writeimage() will know how to interleave fields to frames.
 */
if (vlGetControl(svr, path, src,
VL_SIR_FIELD_DOMINANCE, &dominance) < 0) {
vlPerror("GetControl(VL_SIR_FIELD_DOMINANCE) on video source failed");
exit(1);
}

is_525 = ((timing.intVal == VL_TIMING_525_SQ_PIX)
   || (timing.intVal == VL_TIMING_525_CCIR601));

switch (dominance.intVal) {
case SIR_F1_IS_DOMINANT:
    if (is_525) {
    F1_is_first = 0;
    } else {
    F1_is_first = 1;
    }
    break;
case SIR_F2_IS_DOMINANT:
    if (is_525) {
    F1_is_first = 1;
    } else {
    F1_is_first = 0;
    }
    break;
}
```

Adjusting Component Video Input

To adjust gain and offset for component input video, follow these steps:

1. Select “Live Video Input” in the Utilities menu. The Live Video window appears, displaying video from the analog source.
2. Make sure that “RGB”, “Betacam”, “M-II”, or “YUV” is selected at Format in the Analog Video Source portion of the panel.
3. Select “Analog Video Source” in the Pro menu, and then select “Component.” The Sirius Analog Video Source Component window appears, as shown in Figure C-12.

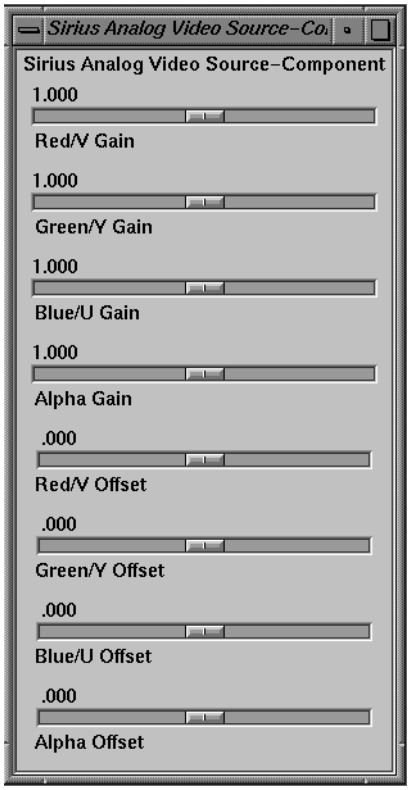

Figure C-12 Adjusting Gain and Offset for Component Video

4. Use the Gain sliders to adjust the voltage level to output the desired amount of red, green, blue, and alpha on each channel.
5. Use the Offset sliders to adjust the active region of the video relative to the black region.

Note: Although the labels for gain and offset are shown as .000 to 2.000 in the window, actual low and high values vary per board. Actual values are displayed when the sliders are at the minimum and maximum positions.

Adjusting Composite and S-Video Video Input

To adjust attributes for composite and Y/C input video, follow these steps:

1. Select "Live Video Input" in the Utilities menu. The Live Video window appears, displaying video from the analog source.
2. Make sure that "Composite" or "SVideo" is selected at Format in the Analog Video Source portion of the panel.
3. Select "Analog Video Source" in the Pro menu, and then select "Composite, Y/C." The Sirius Analog Video Source Composite, Y/C window appears, as shown in Figure C-13.

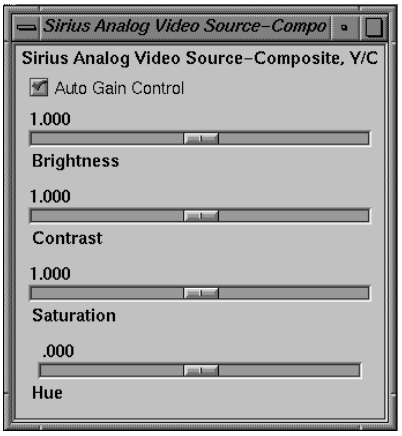

Figure C-13 Adjusting Attributes for Composite Video

4. Use the sliders to adjust the picture.

Setting Up an External Sync Source

To set up Sirius Video for an external sync source, follow these steps:

1. Connect the sync source equipment into the **GENLOCK IN** socket on the **SYNC** section of the breakout box, as shown in Figure C-14.

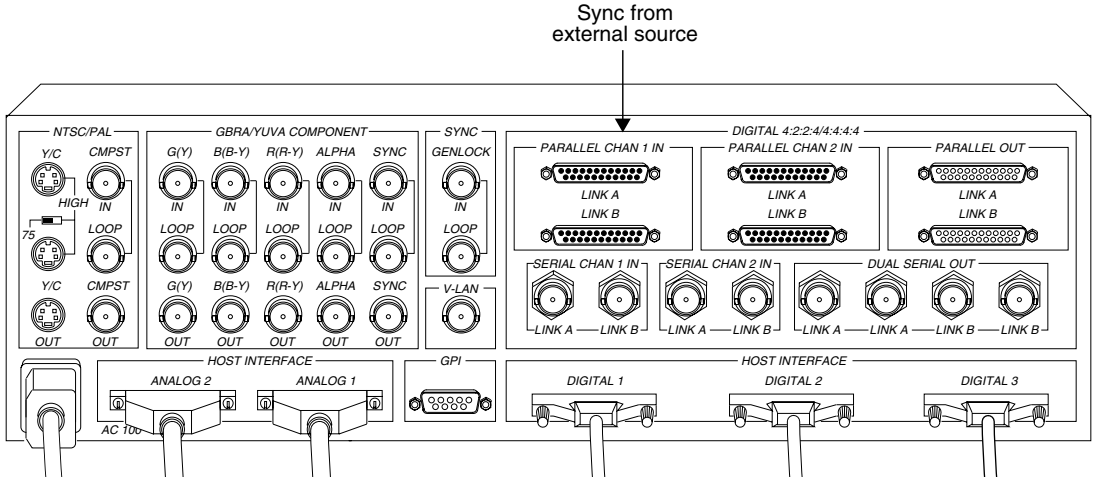

Figure C-14 External Sync Input Port on the Sirius Video Breakout Box

2. If you are using the same signal for other equipment, attach a BNC cable to the **LOOP** socket under **GENLOCK IN** socket to loop the signal through the breakout box. Make sure the final element in the chain is terminated.

   If Sirius Video is the last element in the sync chain, make sure the terminator is attached to the **LOOP** socket under **GENLOCK IN** socket.

3. If necessary, call up the panel (`/usr/sbin/vcp`).

4. In the Genlock Sync menu of the Video Drain section of the control panel, select the format that matches your equipment, as shown in Figure C-15.

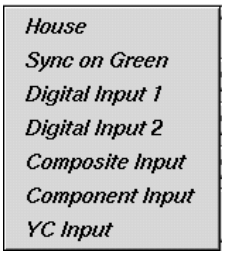

Figure C-15 Selecting Genlock Sync Source

For correspondences between these menu choices and sockets on the breakout box, see Table C-1 and Figure C-8, earlier in this appendix.

5. To set the voltage level for the external sync source, select "Device Controls" in the *vcp* Pro menu, and then select "Synchronization Controls." The Sirius Device-Sync Input Controls window appears, as shown in Figure C-16.

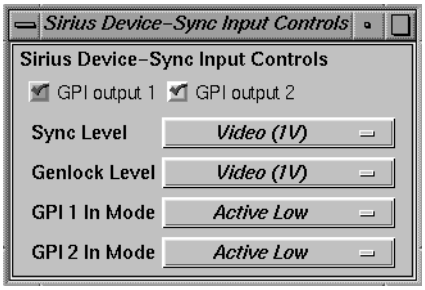

Figure C-16 Setting Genlock Voltage Level

6. To set voltage level, set the voltage level to "Video" (1 V peak-to-peak) or "TTL" (4 V peak-to-peak) in the Genlock Level menu item.

Note: The GPI 1 In Mode and GPI 2 In Mode menu items are discussed in Chapter 4, "Controlling the General-Purpose Interface (GPI)."

Setting Up the Output (Drain)

To set up the drain, follow these steps:

1. Connect the drain equipment into the appropriate socket on the breakout box, as shown in Figure C-17.

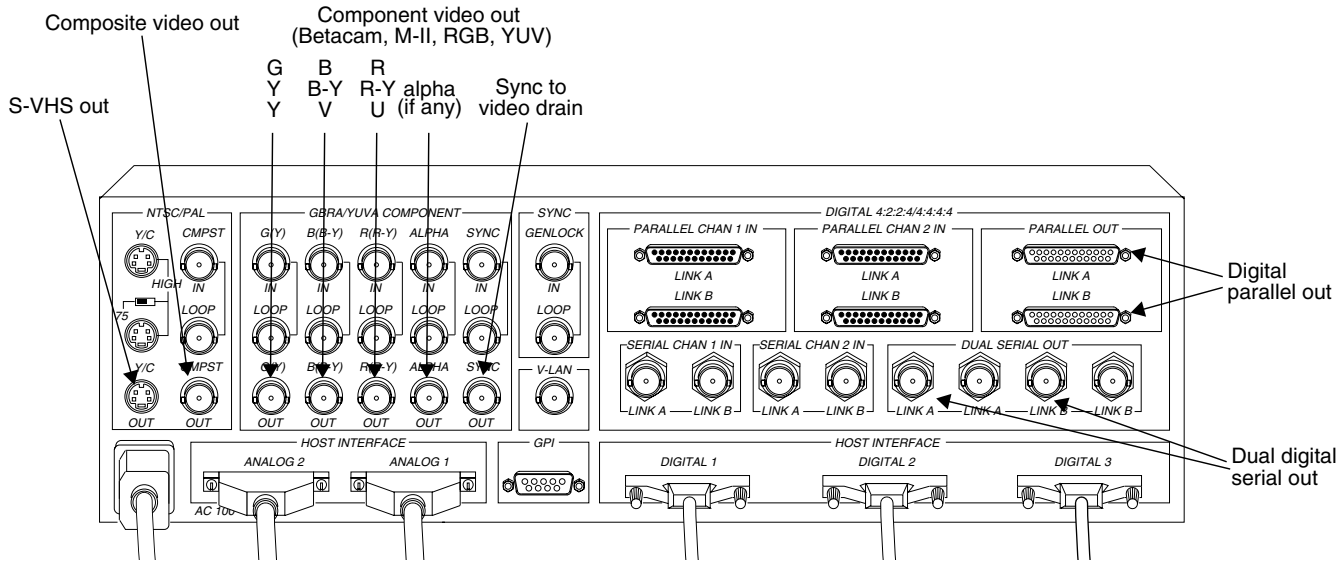

Figure C-17 Output Ports on the Sirius Video Breakout Box

2. If necessary, call up the panel (`/usr/sbin/vcp`).

3. In the Video Drain section of the control panel, select the format that matches your equipment, as shown in Figure C-18.

   Note: The burst lock feature for composite video is not available.

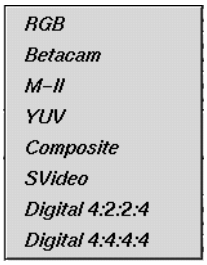

Figure C-18 Selecting Video Drain Format

4. In the Video Drain portion of the panel, select the timing that matches your equipment: 525, 625, CCIR 525, or CCIR 625.

   Hint: In many situations, CCIR timing gives better results.

5. If your drain equipment is synced to the workstation, select "Standalone (internal)" in the Sync Select menu item.

   To sync output to a source other than the workstation, select "Genlock" and select a choice in the Genlock Sync menu item, as shown in Figure C-19.

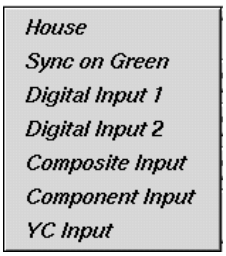

Figure C-19 Selecting Output Genlock Sync

   For correspondences between these menu choices and sockets on the breakout box, see Table C-1 and Figure C-8, earlier in this appendix.

6. You can set signal controls for the drain:

   - convert 4:4:4:4 to 4:2:2:4 output
   - set analog or digital output or both to a rasterful of black (blank) or to a real image
   - select sync voltage level
   - set horizontal phase
   - select field dominance

   To do so, select “Drain” in the Pro menu, and then select “Signal Controls.” The Sirius Video Drain-Signal window appears, as shown in Figure C-20.

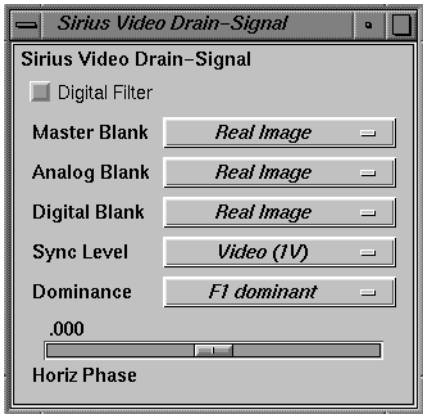

Figure C-20 Adjusting Drain Horizontal Phase

7. Use this window as follows:

   - To convert 4:4:4:4 to 4:2:2:4 output, check the Digital Filter check box.
   - To set analog output to black/blank when you are running digital output formats and vice versa, select “Black/Blank” at the Analog Blank menu item. To set both analog and digital output to black/blank simultaneously, select it at Master Blank.
   - To select sync voltage level, select “Video” (1 V peak-to-peak) or “TTL” (4 V peak-to-peak).

- To set field dominance, at the “Dominance” menu item select “F1 dominant” for the edit to occur on the nominal video field boundary, or “F2 dominant” for the edit to occur on the intervening field boundary.

  See Figure C-11 and “Setting Up Analog Source Video,” earlier in this appendix, for more information.

- To adjust horizontal phase for the output, move the slider.

8. To adjust gain for component output video, select “Video Drain” in the Pro menu, and then select “Component.” The Sirius Video Drain Component window appears, as shown in Figure C-21.

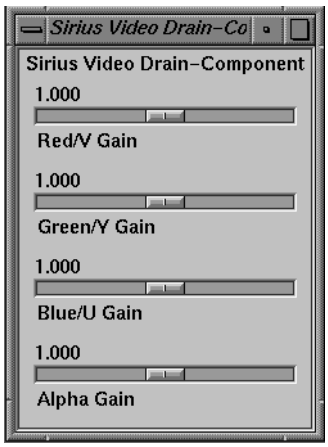

Figure C-21 Adjusting Gain for Component Video Drain

9. Move the sliders to adjust the picture.

   Note: Although the minimum and maximum values for gain are shown in the window as .000 to 2.000, actual ranges vary. These values are displayed when you move the sliders to the minimum and maximum (far left and far right) positions, respectively.

Adjusting Graphics Source or Drain Timing

To set flicker reduction or timing (525, 625, CCIR 525, CCIR 625) for the graphics source, select “Graphics Source” in the *vcp* Pro menu, and then select “Coding Controls.” The Graphics Source-Timing Controls window appears, as shown in Figure C-22.

Figure C-22 Adjusting Graphics Source Timing

Click the check box to activate flicker reduction. Select the appropriate timing at the “Timing” menu item.

Likewise, to set timing (525, 625, CCIR 525, CCIR 625) for the graphics drain, select “Graphics Drain” in the *vcp* Pro menu, and then select “Coding Controls.” The Graphics Source-Timing Controls window appears, as shown in Figure C-23.

Figure C-23 Adjusting Graphics Drain Timing

Select the appropriate timing at the “Timing” menu item.

Adjusting Texture Drain Timing

To set timing (525, 625, CCIR 525, CCIR 625) for the texture drain, select "Texture Drain" in the *vcp* Pro menu, and then select "Coding Controls." The Texture Drain-Timing Controls window appears, as shown in Figure C-24.

Figure C-24 Adjusting Texture Drain Timing

Select the appropriate timing at the "Timing" menu item.

Saving Settings

Once you have set values in *vcp* to match your installation, save them; they are written to */usr/etc/video/videod.defaults*. Select "Restore Settings" on the video control panel File menu to load the values in this file to *vcp*.

The last settings saved are automatically loaded every time the system is reinitialized. If the panel is running, current settings are in effect.

Note: You do not need to open the panel to put its settings into effect.

You can also use File menu choices to restore the factory defaults and close the panel.

Appendix D

Sirius Video Color-Space Conversions

Sirius Video supports three native color spaces—RGB, YUV, and CCIR. The choice of color space is determined by the external equipment for video I/O connections, by the system for connections to the graphics subsystem, and by application software for transfers over the VME bus. Using the VME interface, application software can avoid all color space conversions during video I/O. When color space conversions are done by Sirius, extreme care is taken to assure the correctness and precision of the result.

Understanding the capabilities of Sirius to perform color space conversions and the results of these conversions allows developers and end users to maximize the quality of their output. This appendix explains:

- Sirius Video color spaces
- mathematical operations performed during conversions
- implications of color space conversions

The appendix concludes with examples.

Sirius Video Color Spaces

Sirius Video uses a minimum of ten bits of precision for each color component at all steps of its internal pipeline. Representations for the three native internal color representations are explained separately in this section.

RGB

RGB is the color space used by the graphics subsystem. RGB has the most accurate representation of visible colors since all possible combinations are valid. This color space does not support superblack or other nonvisible color values. Each component is represented by a 10-bit value between 0 and 1023. Black has the value [0,0,0], and white is [1023,1023,1023].

When converting to RGB, each resulting RGB component is clamped to the range [0..1023]. It is possible to overflow the clamping mechanism when dramatically illegal colors are input. Overflows occur only when the resulting red, green or blue value is greater than 2047 or less than -2048.

The Graphics source and drain, the composite/S-Video input and output, and the component RGB input and output use this color space. The VME interface can use this color space if desired.

Note: Do not use 4:2:2 coding with RGB data.

YUV

The YUV color space is obtained from RGB by the matrix transformation in equation 1.

Equation 1

$$\begin{bmatrix} 0.500 & -0.419 & -0.081 \\ 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 512 \\ 0 \\ 512 \end{bmatrix} = \begin{bmatrix} V \\ Y \\ U \end{bmatrix}$$

The V, Y, and U values range from [0..1023]. Black has the VYU value [512,0,512]. White has the value [512,1023,512].

This color space is used by the Betacam, M-II and YUV formats. The VME interface can use this color space. With proper filtering, 4:2:2 coding can be used.

CCIR

The CCIR color space is obtained from RGB by the matrix transformation in equation 2.

Equation 2

$$\begin{bmatrix} 0.500 & -0.419 & -0.081 \\ 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} \times \begin{bmatrix} \frac{896}{1023} \\ \frac{876}{1023} \\ \frac{896}{1023} \end{bmatrix} + \begin{bmatrix} 512 \\ 64 \\ 512 \end{bmatrix} = \begin{bmatrix} Cr \\ Y \\ Cb \end{bmatrix}$$

The Cr, Y, and Cb values are clamped to the range [4..1019]. Black has the CrYCb value [512,64,512]. White has the value [512,940,512].

This color space is used by the component digital formats. The VME interface can use this color space. With proper filtering, 4:2:2 coding can be used.

Mathematical Operations Performed During Conversions

Sirius Video can process and store each color space explained in the previous section. For best precision, the input color space should be maintained through the processing path. For example, an application that implements DDR functionality could choose to store data in the native representation of the input signal: Betacam data could be stored as YUV, input from an RGB camera as RGB, and data from a D1 deck as CCIR. If the application works in this way, no conversions are performed and the data is passed directly through the system. In particular, CCIR601 data coming from a D1 deck is bit-accurate in this case.

However, it might not be desirable for the application to work this way. If that is the case, the application can use all of the conversion, decimation and interpolation capabilities of Sirius Video to perform real-time color space and 4:2:2 ⇔ 4:4:4 conversions.

Conversions are performed only when absolutely required. Each incoming stream can be converted from its current color space to any other color space. Conversions can also be performed when going to graphics and one of either the analog or digital video outputs.

The output color space controls conversions. For example, if you blend a CCIR stream from a digital video input with an RGB stream from graphics and send the result to the digital video output, the RGB signal is converted to CCIR before the blend occurs. The CCIR stream is not converted. If you sent the same blend to a Betacam output, both streams are converted to YUV before the blend.

Implications of Color Space Conversions

The two major concerns when performing conversions from one color space to another are *precision* and *range*.

Precision of Color Conversions Done by Sirius

Sirius Video stores colors with a minimum of 10 bits of precision at all steps in its pipeline. When performing color space conversions, the data is converted to 12-bit signed values before it is passed to the matrix multipliers. The matrix multipliers have 9-bit coefficients and 23-bit accumulators. The most significant 12 bits of the matrix-multiplication result are passed on to additional hardware, which applies any needed offsets and then clamps to the proper range.

Silicon Graphics, Inc., has verified both through simulation and hardware testing that the maximal error for two conversions (RGB to CCIR to RGB) is four units out of 1024. The matrix coefficients have been biased to round slightly high rather than slightly low to avoid the type of problems that can otherwise easily occur in the blue component.

Conversions between RGB and YUV are more accurate (a maximum error of 3 in 1024 after two conversions), since data is not as compressed in the YUV representation.

Range Issues For Color Conversions Done by Any Means

Different color spaces allocate the available bits of precision in different ways. The RGB space is designed to maximize the accuracy of color representations. The YUV and CCIR color spaces are designed to strongly uncouple chrominance and luminance information.

Since RGB represents visible colors, it is contained inside the YUV and CCIR spaces. The CCIR color space also has a slight amount of additional headroom that was intended to prevent aliasing artifacts when Finite Impulse Response filtering operations are performed on the digital data.

Anytime a conversion operation is performed between CCIR and RGB or between CCIR and YUV, the colors that are not representable in the destination color space must be somehow mapped into colors that are representable. The usual way to do this is to clamp each component to the available range in the destination color space. Other methods, such as projecting towards the center of the representable space, might produce results that appear to be better in some cases, but are not feasible to implement in hardware.

When converting from CCIR to YUV, the axes of the two spaces are parallel, so the result of this clamping operation is very predictable. Superblack and superwhite are clipped to black and white, respectively, and oversaturated colors might also be clipped.

When converting from RGB to YUV or CCIR, clamping never occurs, because all RGB colors are representable in those color spaces.

When converting from CCIR or YUV to RGB, the results of clamping are much less intuitive, because these conversions involve rotation and scaling operations, with the result that the component axes in one color space don't align with those in the other.

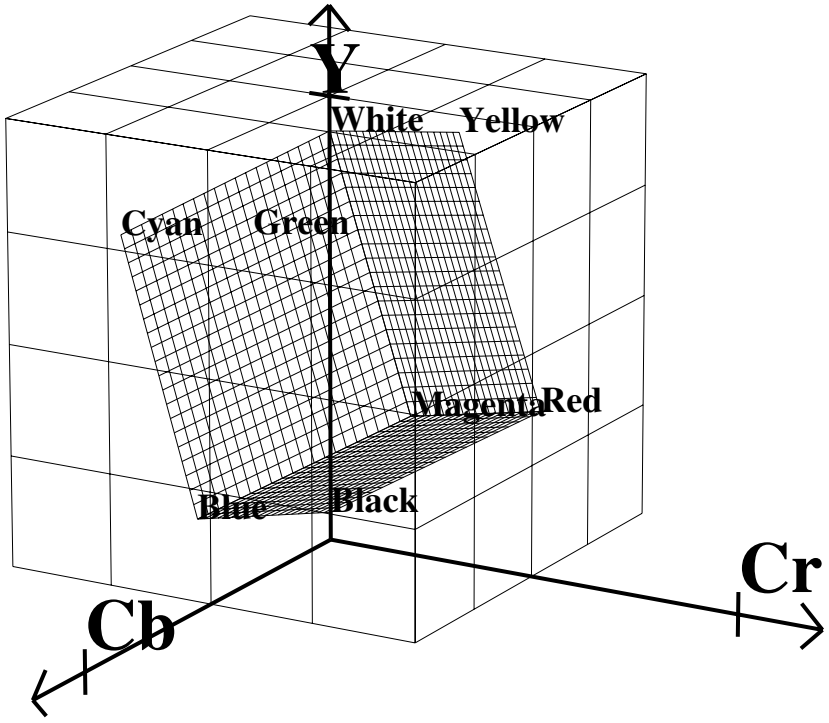

Figure D-1 RGB Cube in CCIR Space

Figure D-1 shows the RGB color cube inside the CCIR color space. The volume contained within the outer (CCIR) cube, but outside the inner (RGB) cube, represents “illegal” colors that cannot be displayed.

As shown in the figure, the CCIR color space allocates almost three quarters of its available bit combinations to illegal colors. When any of these color values are converted to RGB, the result is clamped to the edge of the RGB cube. Since the inner cube contains the displayable colors, this clamping operation has no impact on them.

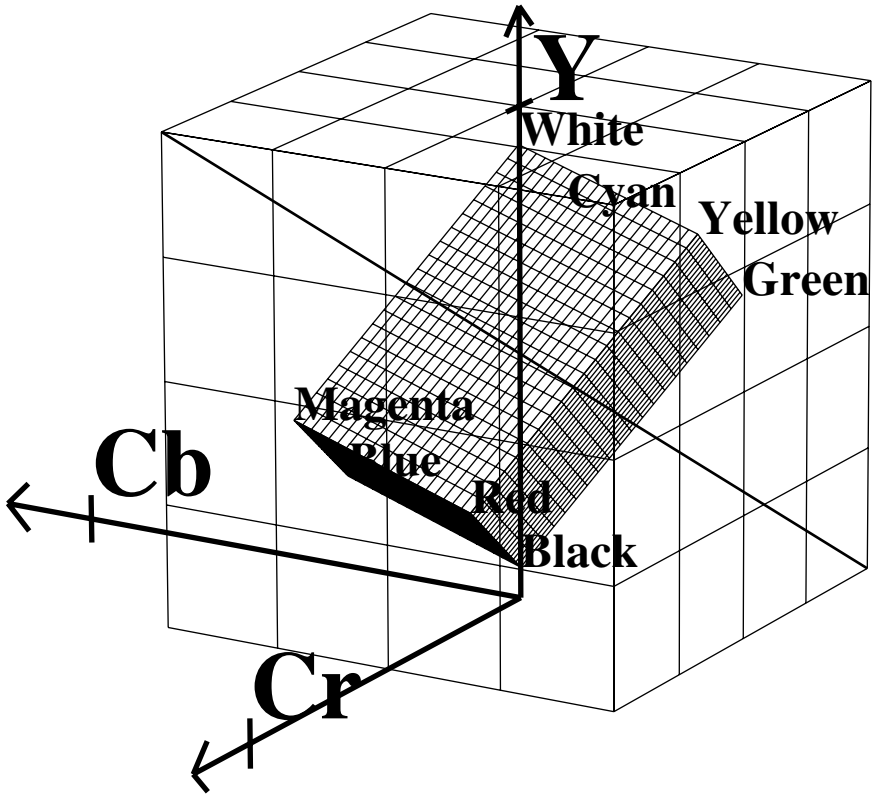

Figure D-2 Color Cube With Luminance/Chrominance Ramp Vector

If CCIR is converted to RGB and back to CCIR using certain types of test signals, the output can appear to be vastly wrong. A common and extreme version of this is the signal that simultaneously ramps Cr, Y, and Cb from the minimum to maximum possible values.

In Figure D-2, the heavy diagonal line passing through the figure is the set of colors in the luma/chroma ramp test signal. As shown in the figure, a large portion of this pattern is outside the RGB cube. In fact, over two thirds of this pattern is outside the displayable range.

Example Color Conversions

This section includes example graphs that display the results of converting from CCIR to RGB and back. They show the same type of result you would see if you brought a digital signal into Sirius, passed it through the graphics subsystem or over VME using RGB format, and sent it back out to the digital output.

These effects do not occur if you simply pass digital data through Sirius or over VME using the CCIR format. In these cases, the output matches the input on a bit-by-bit basis.

Example 1: 100% Color Bars

This example, like the other two in this section, consists of three graphs. Each graph displays the input CCIR pattern, intermediate RGB pattern, and output CCIR pattern for a given color component. Figure D-3 shows the red and Cr components, Figure D-4 the green and Y components, and Figure D-5 the blue and Cb components. In this example and the others, if the input and output CCIR values are identical, only two lines are shown.

In this example, conversion to RGB and back has no effect on the image. The 100% amplitude color bar signal lies within the visible range and therefore is perfectly represented in RGB.

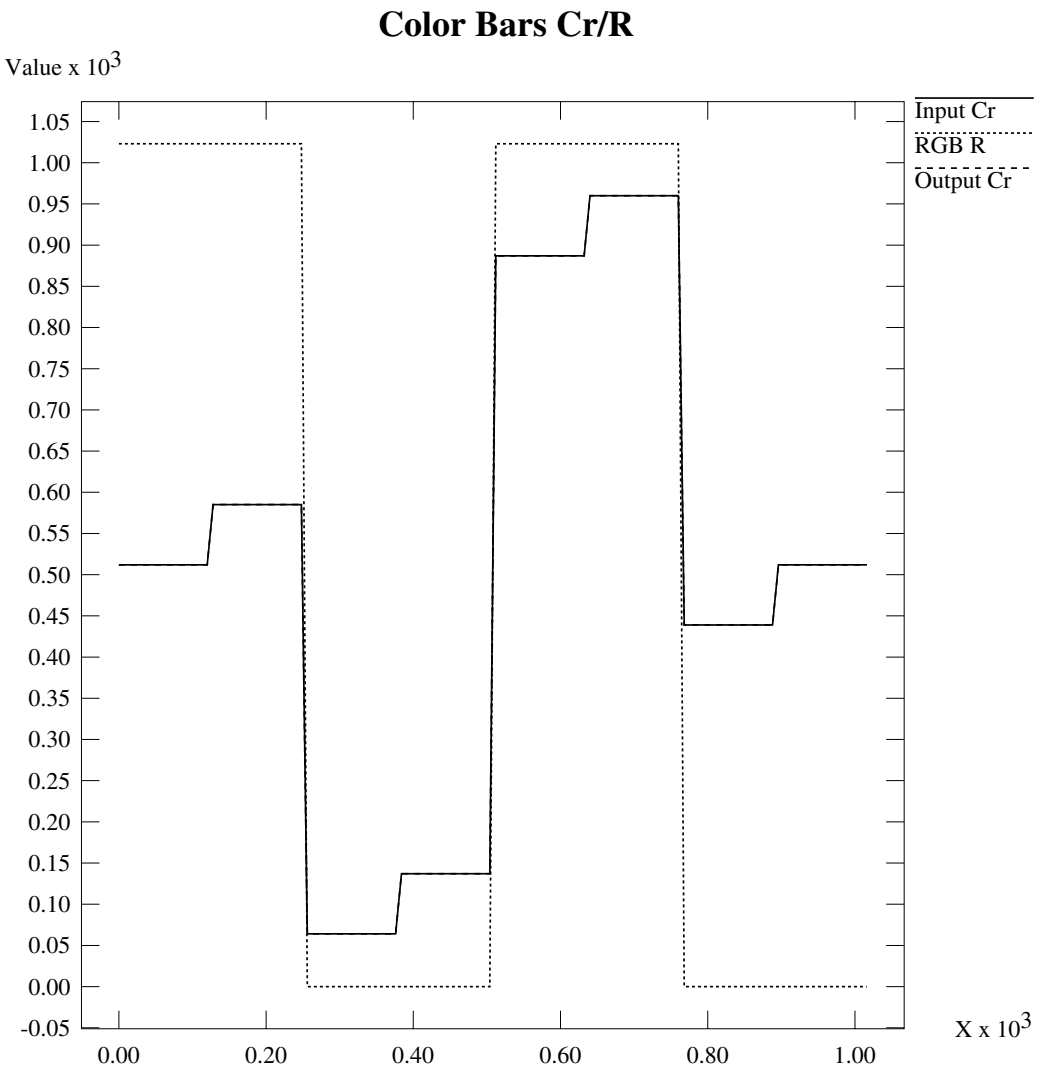

Figure D-3 100% Color Bars: Cr/R

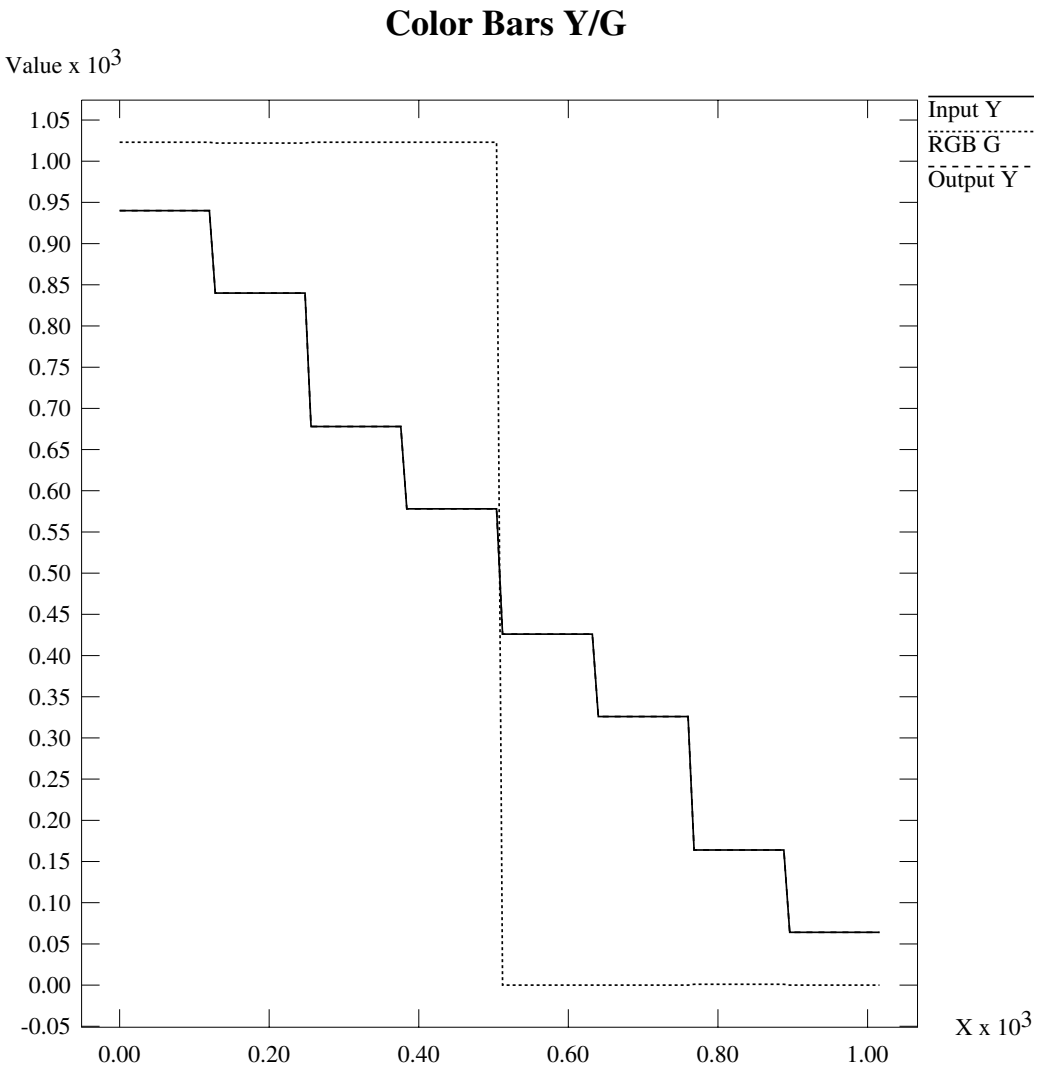

Figure D-4 100% Color Bars: Y/G

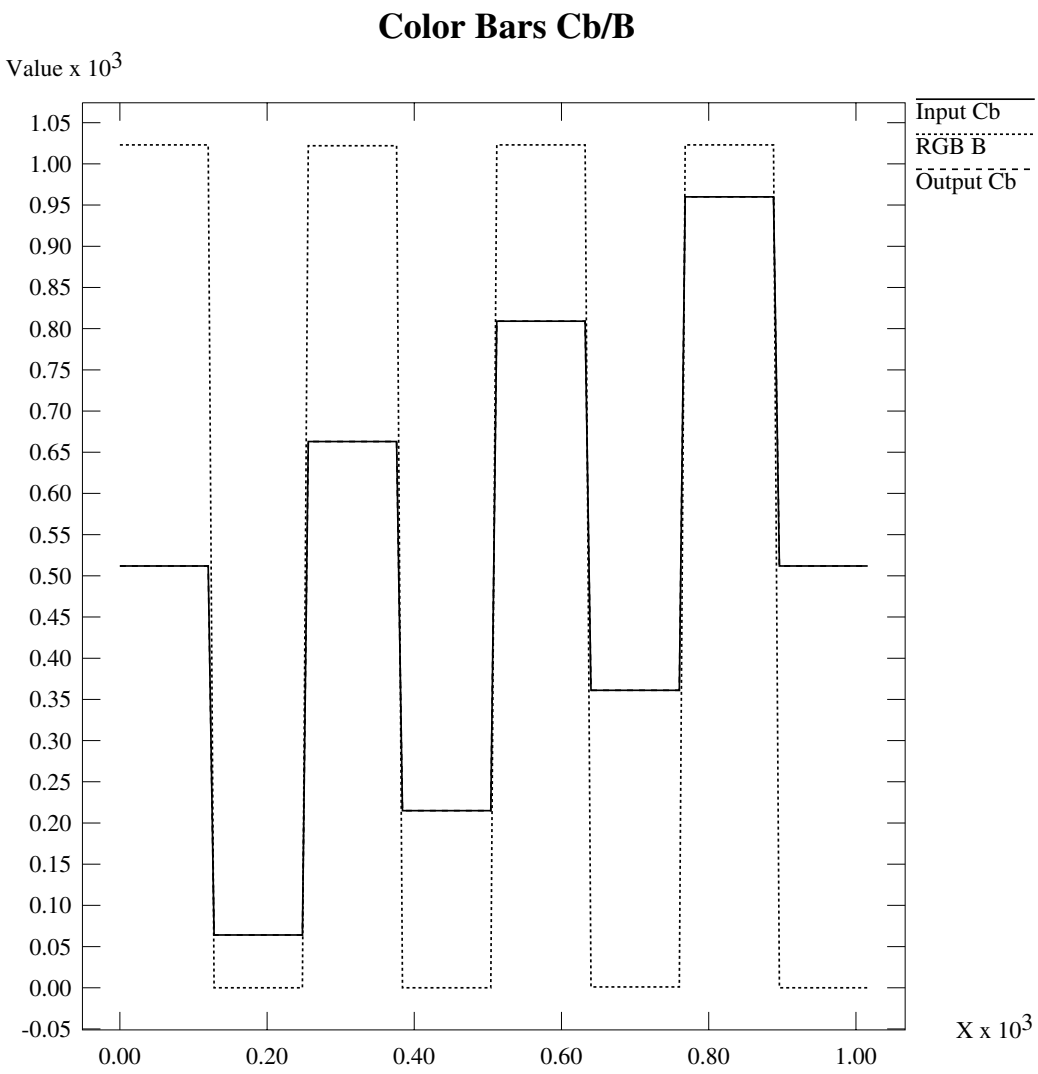

Figure D-5 100% Color Bars: Cb/B

Example 2: Luminance Ramp

In this example, the conversion to RGB and back affects only the superblack and superwhite regions. All luminance values that are blacker than black are clamped to black; all values whiter than white are clamped to white.

In the RGB color space, each component ramps from 0 to 1023 as the input luminance ramps from 64 (black) to 940 (white). This test pattern lies along the Y axis of the color cubes.

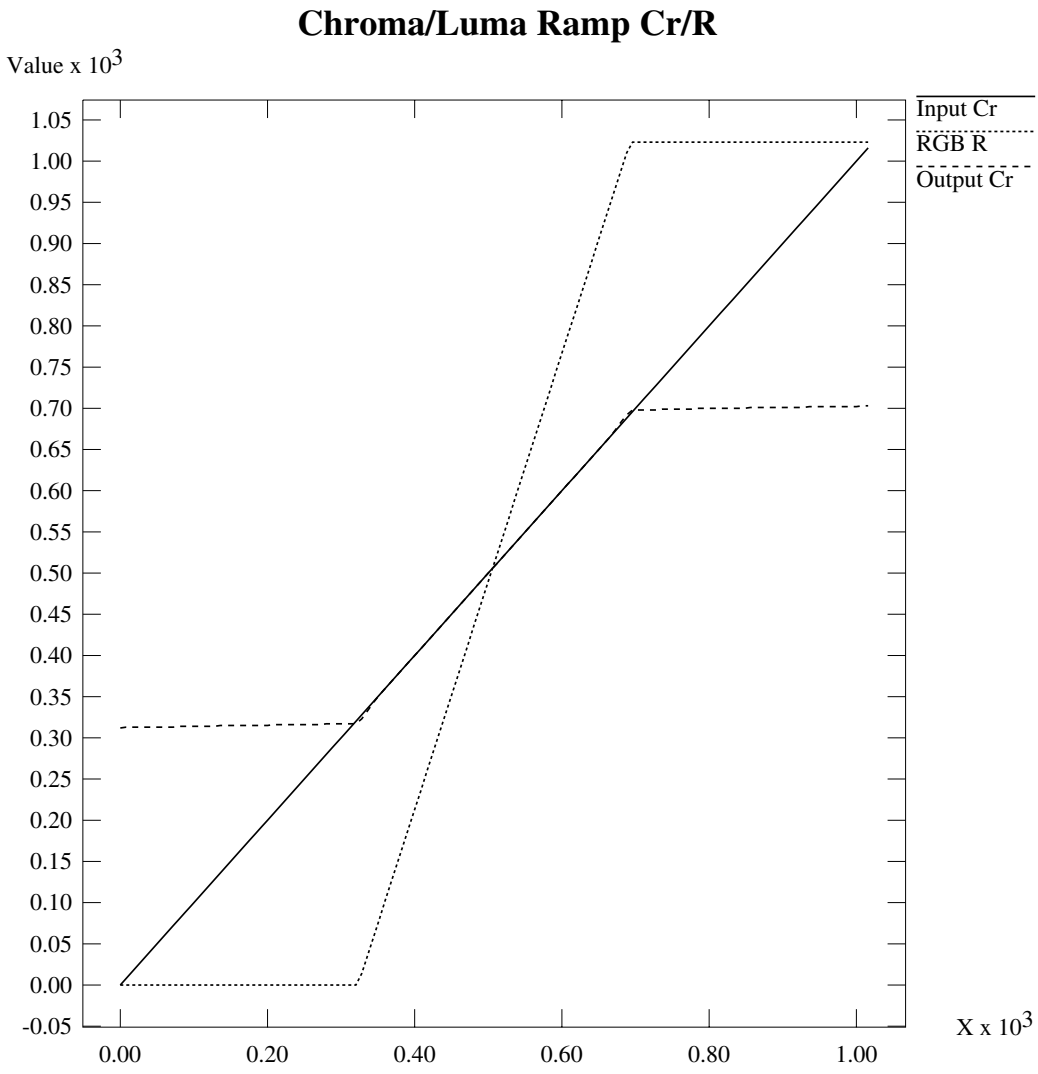

Figure D-6 Luminance Ramp: Cr/R

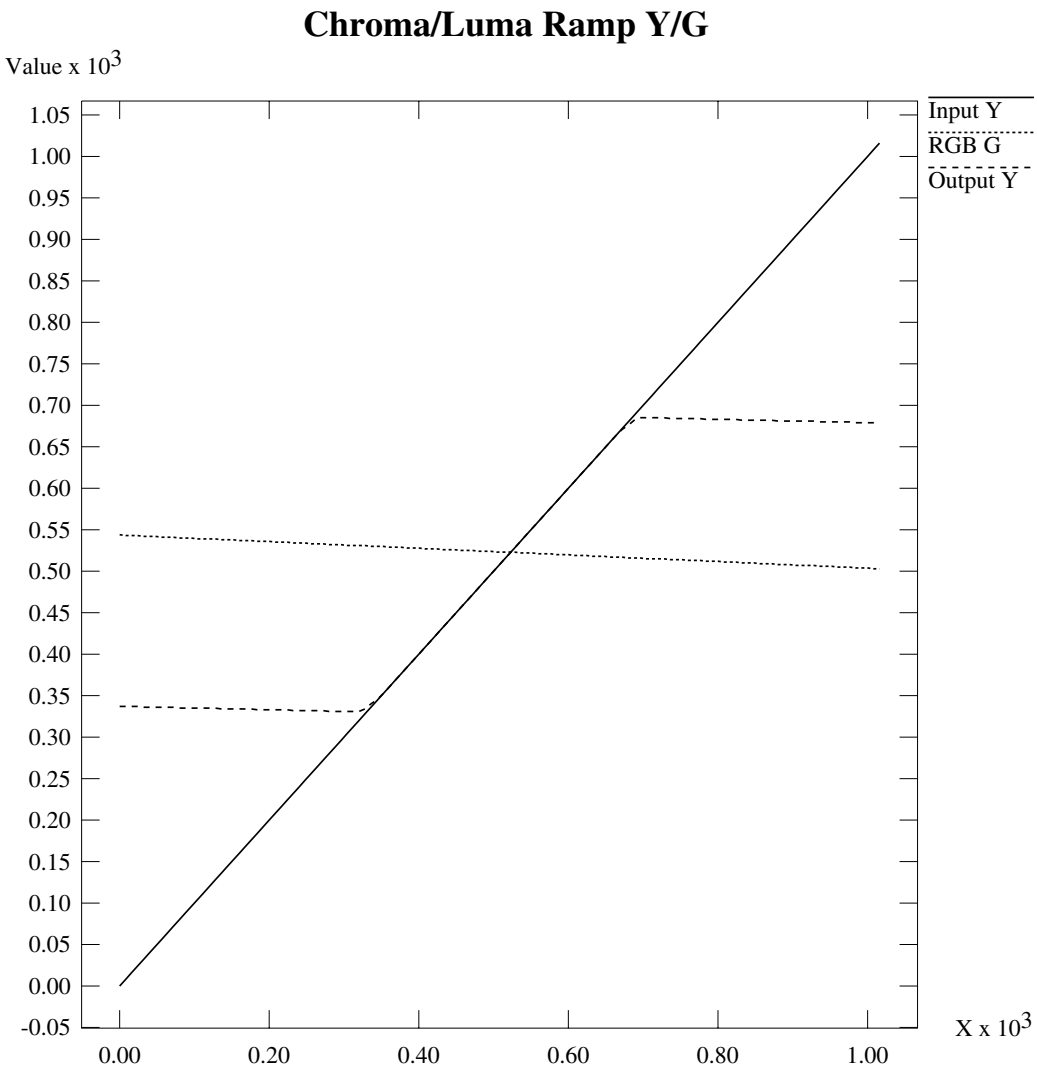

Figure D-7 Luminance Ramp: Y/G

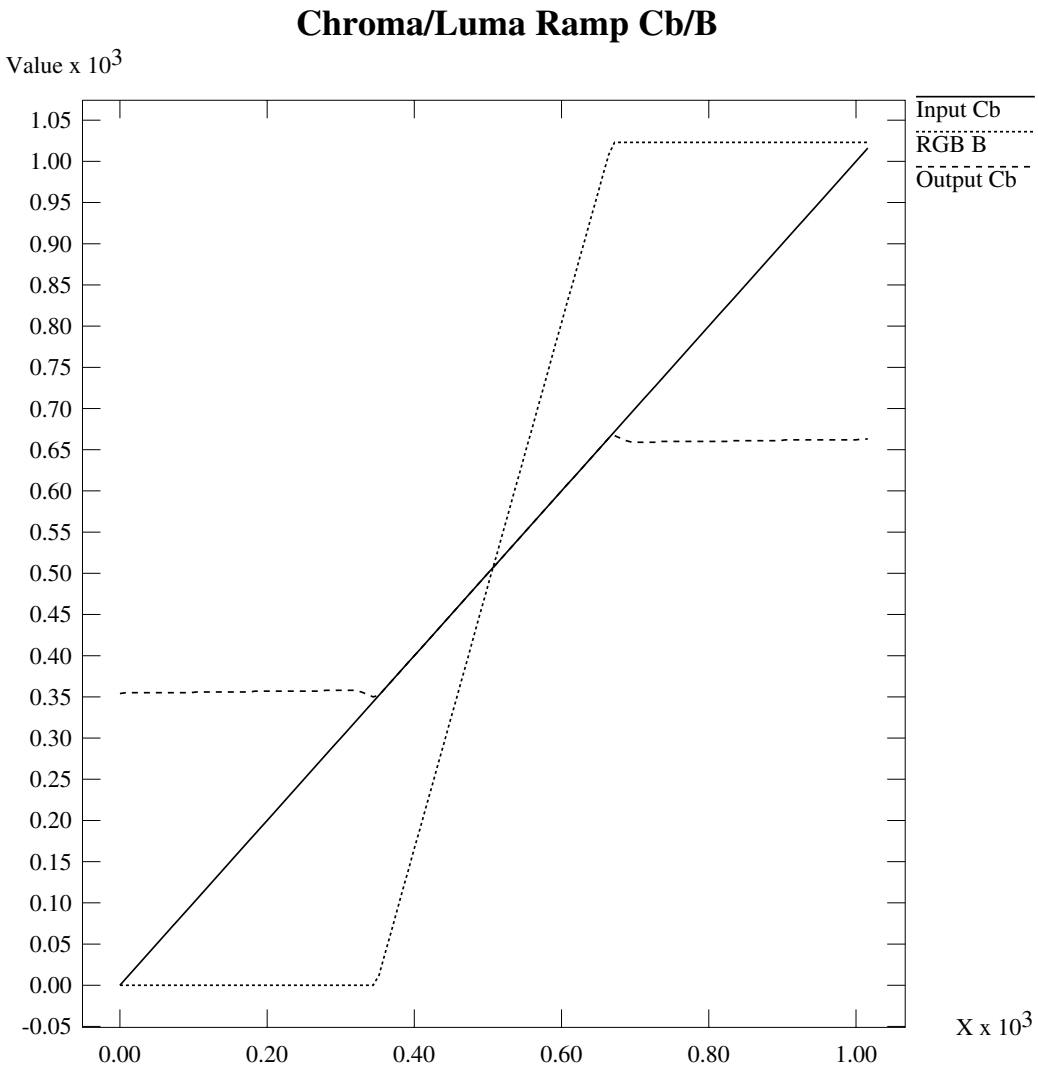

Figure D-8 Luminance Ramp: Cb/B

Example 3: Simultaneous Chroma/Luma Ramp

This example is the most extreme of the three, and shows how surprising the results of color conversions can be when arbitrary synthetic CCIR inputs are used.

Each CCIR input signal ramps from 0 to 1023 simultaneously. As mentioned in the first example, over two thirds of this pattern lies outside the legal range. The portion within the legal range is represented exactly, but the region outside is clamped to the RGB cube surface.

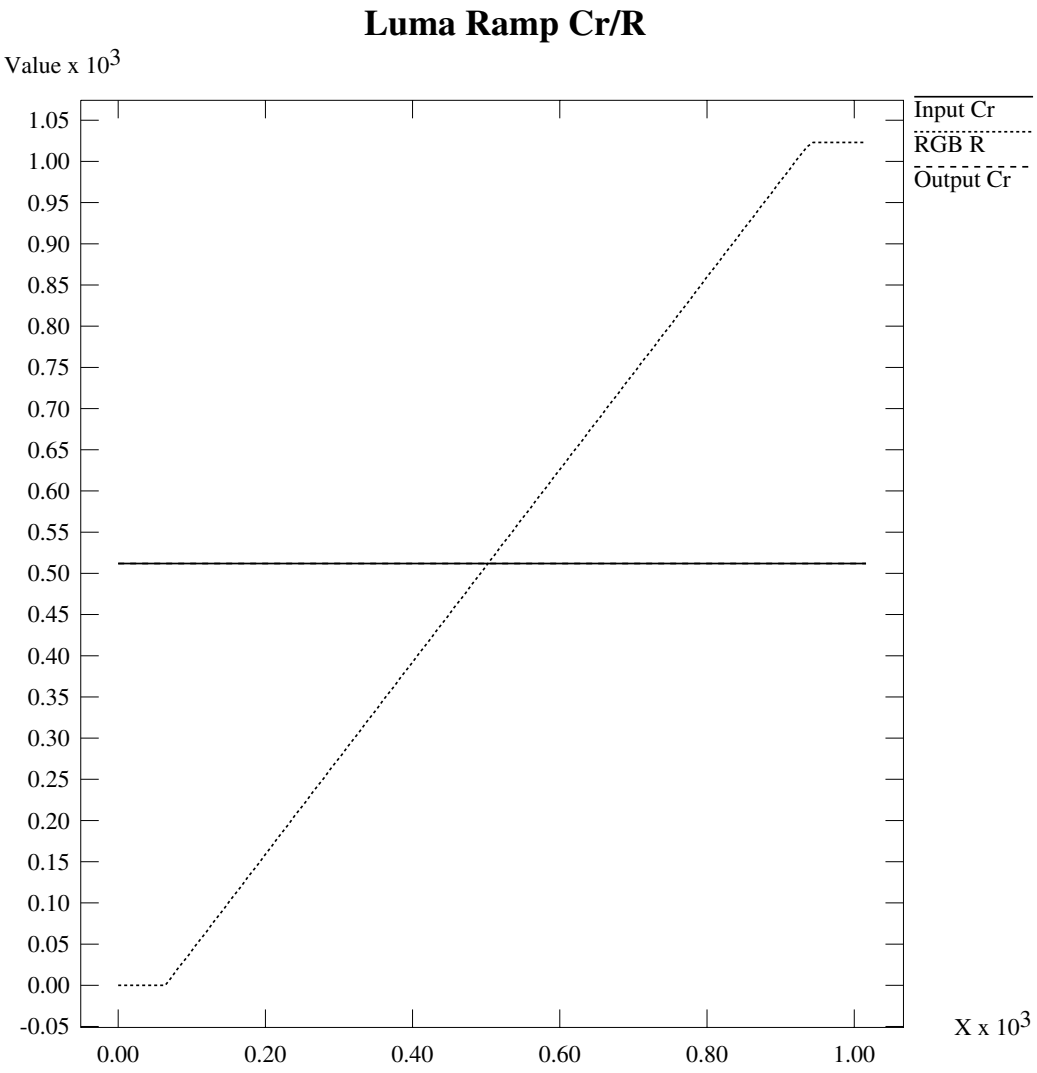

Figure D-9 Chroma/Luma Ramp: Cr/R

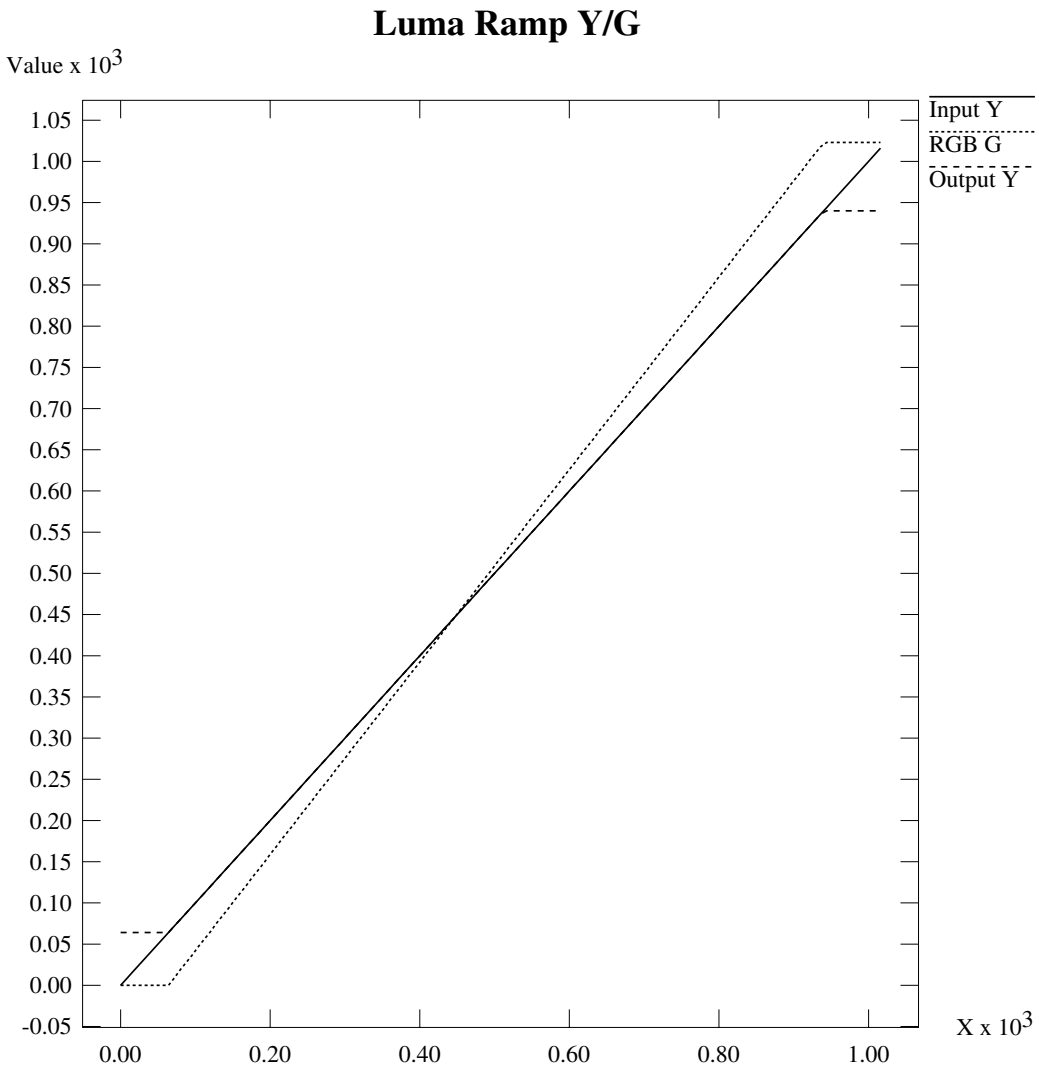

Figure D-10 Chroma/Luma Ramp: Y/G

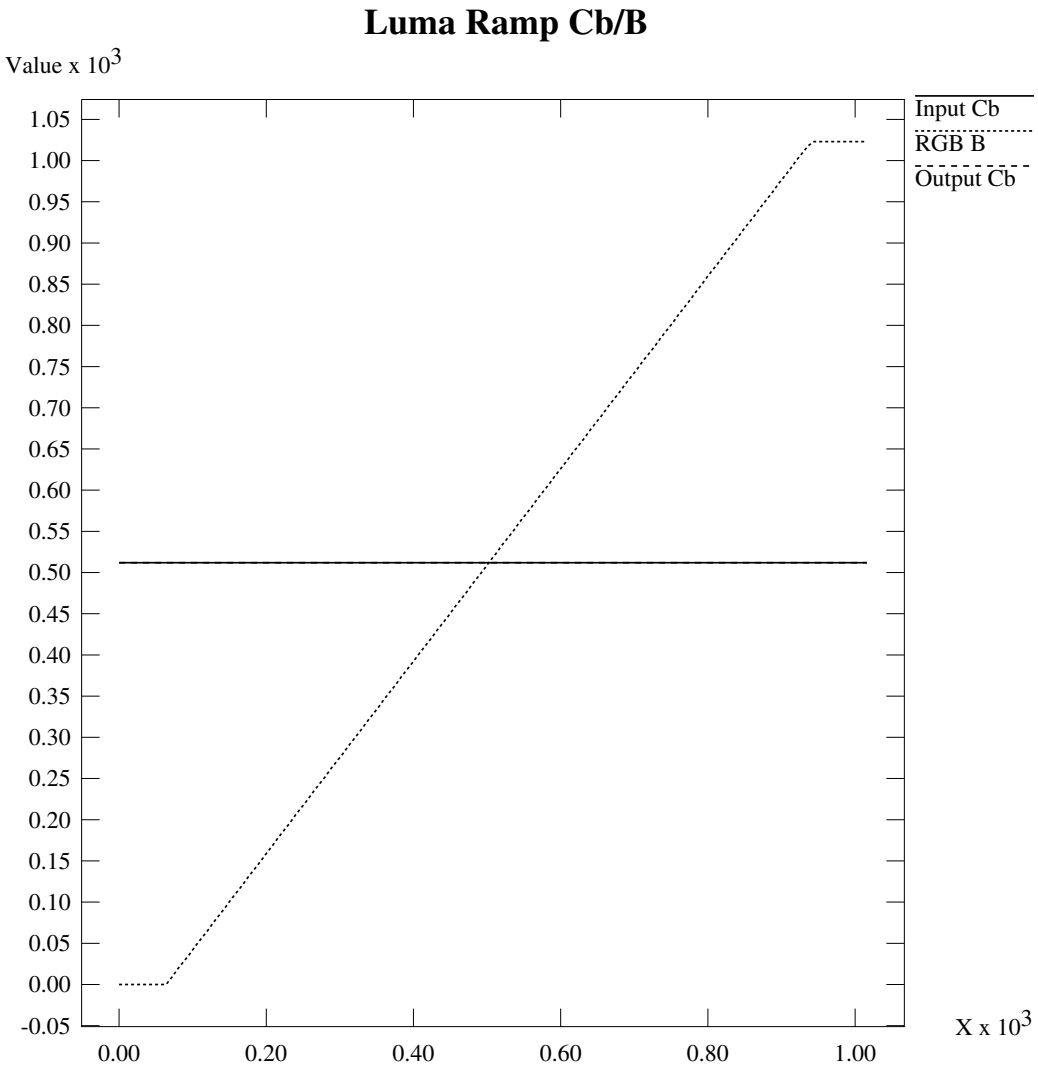

Figure D-11 Chroma/Luma Ramp: Cb/B

Appendix E

Example Programs

This appendix introduces and gives the code for Sirius Video example VL programs:

- *sir_memtovid.c*
- *sir_vidtomem.c*
- *sir_vlan.c*
- *gfxvidkeytovid.c*
- *vidtogfx.c*
- *vidtotex.c*
- *vidtovid.c*

The directory */usr/dmedia/bin/SIRIUS* contains the Sirius Video executables explained in this chapter. To follow instructions in this chapter, switch to this directory.

Note: For general guidelines in using the VL for Sirius Video, see Chapter 2, "Programming Sirius Video," earlier in this guide.

Source for these programs is

- IRIX 5.2 (Sirius Video software version 1.0.2): */usr/people/4Dgifts/examples/video/sirius*
- IRIX 5.3 (Sirius Video software version 1.1): */usr/people/4Dgifts/examples/dmedia/video/sirius*

Note: If Sirius Video and Multi-Channel Option are installed on the same VME section of your system, Sirius Video is unable to use utilities that send graphics to video. These utilities are *gfxvidkeytovid*, *gfxtogfx*, *gfxtovid*, and

gfxvidtovid. The first of these utilities is discussed in this appendix; the others are discussed in the Video Library section of the *Digital Media Programming Guide*.

These example programs construct VL paths between the nodes in their names and demonstrate how to control Sirius Video features. All but the last of these programs require that the Sirius Video board be attached to a system with RealityEngine graphics via the CPI and DG1 options.

Note: To keep these examples as simple as possible, event handling is not implemented.

sir_memtovid.c

sir_memtovid is the primary means of transferring a frame from memory to the video output. It uses the Video Library to route a sequence of RGB images through the Sirius Video board to the video output in a burst.

Video output timing and format must be set beforehand in *vcp*. The files are read in numerically increasing order, using the base filename prefix supplied on the command line. The first filename generated is either *<prefix>_00000.rgb* or *<prefix>_00000.rgba*, depending on whether or not alpha was requested. The default starting file number *00000* can be changed with the *-N* option argument.

Usage:

```
sir_memtovid [-ac# -F# -hln# -N# -pPr# -t# -T] <base filename prefix>
```

where

`-a` reads the files along with the alpha information also

`-c#` specifies the number of frames to read and transfer

`-F#` gives the file format

- `1`: Silicon Graphics RGB frames (the default)
- `2`: Silicon Graphics RGB fields

`-h` prints a usage message

`-l`	runs the frames continuously in a movie loop
`-n#`	specifies the video device to be used, as reported by *vlinfo*
`-N#`	changes the default file naming order to begin with the specified number
`-p`	causes the program to pause at the end of the transfer until a character is typed, at which point *vlEndTransfer*() is called and the path is destroyed
`-P`	prefills the ring buffer before the start of the transfer with as much data as it can hold, which is useful for a triggered transfer
`-r#`	specifies in frames the number of ring buffer elements to allocate; the maximum of 100 fields (50 frames) can be increased if you recompile the source and change the maximum
`-t#`	specifies the external trigger source (GPI or V-LAN input) that initiates the output DMA sequence
`-T`	requests that a pair of GPI notify events be generated at the start (GPI_1 output) and end (GPI_2 output) of the transfer}

sir_vidtomem.c

sir_vidtomem sends frames from the video output to memory. It can write frames to graphics via *lrectwrite* or to disk as disk files. This program uses the Video Library to grab a sequence of still video images through the Sirius Video board into host memory, and optionally to graphics or disk files.

Video input timing and format must be set beforehand in *vcp*. The files are generated in numerically increasing order, using the base filename prefix supplied on the command line. The first filename generated is either *<prefix>_00000.rgb* or *<prefix>_00000.rgba*, depending on whether or not alpha was requested. The default starting file number *00000* can be changed with the *-N* option argument.

Usage:

```
sir_vidtomem [-ac# -dF# -hn# -N# -r# -t# -v# -w] <base filename prefix>
```

where

`-a`	also saves the alpha channel
`-c#`	specifies the number of frames to transfer and write
`-d`	displays the captured video data to graphics (field data is line-replicated)
`-F#`	gives the file format • `1`: Silicon Graphics RGB frames (the default) • `2`: Silicon Graphics RGB fields
`-h`	prints a usage message
`-n#`	specifies the video device to be used, as reported by *vlinfo*
`-N#`	changes the default file naming order to begin with the specified number
`-r#`	specifies in frames the number of ring buffer elements to allocate; the maximum of 100 fields (50 frames) can be increased if you recompile the source and change the maximum
`-t#`	specifies the external trigger source (GPI or V-LAN input) that initiates the input DMA sequence
`-T`	requests that a pair of GPI notify events be generated at the start (GPI_1 output) and end (GPI_2 output) of the transfer
`-v#`	sets the input video node to be used, overriding the default set via *vcp*; set timing and format for the node in *vcp*
`-w`	inhibits the file-writing operation so that you can view the video on the graphics screen with the *-d* option, with no file I/O overhead}

sir_vlan.c

sir_vlan, a Sirius Video configuration and command tool, performs the basic V-LAN operations of initializing the on-board V-LAN controller, querying controller status, and sending V-LAN commands to it.

Usage:

```
ir_vlan [-n <video device number>] [-achiqr]
```

where

`-a`	checks the V-LAN initialization status
`-c`	executes a single V-LAN command
`-h`	prints usage information
`-i`	initializes the V-LAN subsystem, which must be done after you select the timing for video output and before any other V-LAN commands
`-q`	causes operation in quiet mode, so that no status messages are printed
`-r`	executes commands repeatedly; it must be used with the *-c* flag}

gfxvidkeytovid.c

This demo program for the chroma key generator sets input, video output, grab area, blender functions, values, ranges, and softness parameters for each component (A, B, and C). It also creates special keys in a proprietary color space.

Note: If Sirius Video and Multi-Channel Option are installed on the same VME section of your system, Sirius Video is unable to use this program.

Usage:

```
gfxvidkeytovid [-a] [-b] [-c] [-r] [-s] [-v#] [-x#] [-y#] [-w#] [-h#] [-n#]
```

where

`-a`	is the value of the R or V component
`-b`	is the value of the G or Y component
`-c`	is the value of the B or U component
`-r`	is the range for all components
`-s`	is the softness value for all components
`-v#`	is the video input device number, as reported by *vlinfo*
`-x#`	is the left position for the grab area
`-y#`	is the top position for the grab area
`-w#`	is the width of the grab area
`-h#`	is the height of the grab area
`-n#`	is the video output device number, as reported by *vlinfo*}

vidtogfx.c

vidtogfx continuously captures a stream of live video from a video source and displays it via the Iris GL in a window on the attached RealityEngine graphics display.

The video source can be selected via a command-line option, or via the default input device control on *vcp(1)*. For the *-v* option, the valid video source numbers and their mappings can be seen from the output of *vlinfo(1)*.

Usage:

```
vidtogfx [-n <video device number>] [-v <input>]
```

where *-n#* selects the video device to be used, as reported by *vlinfo*, and *-v#* specifies the input stream as reported by *vlinfo*.}

vidtotex.c

vidtotex continuously captures a stream of live video from a video source and repeatedly loads it into an Iris GL texture map. Those texture maps are then used to texture a polygon in an Iris GL window on the RealityEngine graphics display, which gives the appearance of a pane of live video.

The video source can be selected via a command-line option, or via the default input device control on *vcp(1)*. For the *-v* option, the valid video source numbers and their mappings can be seen from the output of *vlinfo(1)*.

Usage:

```
vidtotex [-n <video device number>] [-v <input>]
```

where *-n#* selects the video device to be used, as reported by *vlinfo*, and *-v#* specifies the input stream as reported by *vlinfo*. The default device is 0, the first Sirius board detected by the kernel. The devices are numbered in the order that they are probed and detected by the kernel.

Note: The demo program */usr/dmedia/bin/SIRIUS/shatter* takes this video texturing ability somewhat further and demonstrates a live video textured polygon that you can scale, rotate, and shatter as if it were a pane of glass.

vidtovid.c

vidtovid continuously captures a stream of live video from a video source and displays it on the Sirius Video output. The video source can be selected via a command line option, or via the default input device control on *vcp(1)*. For the *-v* option, the valid video source numbers and their mappings can be seen from the output of *vlinfo(1)*.

Usage:

```
vidtovid [-n <video device number>] [-v <input>]
```

where *-n#* selects the video device to be used, as reported by *vlinfo*, and *-v#* specifies the input stream as reported by *vlinfo*. The default device is 0, the first Sirius board detected by the kernel. The devices are numbered in the order that they are probed and detected by the kernel.}

Index

D

E

F

G

H

I

K

L

M

N

P

R

S

T

V

W

Tell Us About This Manual

As a user of Silicon Graphics products, you can help us to better understand your needs and to improve the quality of our documentation.

Any information that you provide will be useful. Here is a list of suggested topics:

- General impression of the document
- Omission of material that you expected to find
- Technical errors
- Relevance of the material to the job you had to do
- Quality of the printing and binding

Please send the title and part number of the document with your comments. The part number for this document is 007-2238-003.

Thank you!

Three Ways to Reach Us

- To send your comments by **electronic mail**, use either of these addresses:
  - On the Internet: techpubs@sgi.com
  - For UUCP mail (through any backbone site): *[your_site]*!sgi!techpubs
- To **fax** your comments (or annotated copies of manual pages), use this fax number: 650-932-0801
- To send your comments by **traditional mail**, use this address:

  Technical Publications
  Silicon Graphics, Inc.
  2011 North Shoreline Boulevard, M/S 535
  Mountain View, California 94043-1389