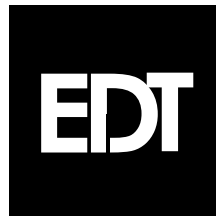


# S1V

Video Capture and Frame Buffer for the Sun SBus

## USER'S GUIDE

008-00226-02



The information in this document is subject to change without notice and does not represent a commitment on the part of Engineering Design Team, Inc. The software described in this document is furnished under a license agreement or nondisclosure agreement. The software may be used or copied only in accordance with the terms of the agreement.

No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, without the express written agreement of Engineering Design Team, Inc.

© Copyright Engineering Design Team, Inc. 1992–1997. All rights reserved.

Refer questions or problems with this manual or the hardware or software described herein to:

Engineering Design Team, Inc.  
1100 NW Compton Drive, Suite 306  
Beaverton, Oregon 97006

Phone: (503) 690-1234  
Fax: (503) 690-1243  
E-mail: [info@edt.com](mailto:info@edt.com)  
Web: [www.edt.com](http://www.edt.com)

Sun, SunOS, SBus, SPARC, and SPARCstation are trademarks of Sun Microsystems, Incorporated.

UNIX and OPEN LOOK are registered trademarks of UNIX System Laboratories, Inc.

X Window System is a product of the Massachusetts Institute of Technology.

EDT and Engineering Design Team are trademarks of Engineering Design Team, Inc.

# Contents

Overview .....	1
Installation.....	2
Installing the Hardware.....	2
Installing the Software .....	3
Using SunOS Version 4.1.x (Solaris 1.x) .....	3
Using SunOS 5.x (Solaris 2.x) .....	3
Included Files.....	4
The S1V Tool.....	6
Invoking the S1V Tool.....	6
Options.....	6
S1V Tool User Interface Reference .....	7
Writing Applications .....	9
Opening the Device.....	9
Configuring the Device.....	9
Acquiring an Image.....	10
Single-Frame Acquisition .....	10
Continuous Acquisition .....	10
Accessing the Data .....	11
Displaying the Data.....	12
Example Applications.....	12
s1vread.....	13
mread .....	13
Usage.....	13
s1vwrite .....	14
framecount .....	14
grayscale.....	14
Usage.....	14
s1vconfig.....	15
S1V Video Library.....	16
sv_open(device_name, lock) .....	17
sv_close(sv_dev) .....	17
sv_set_defaults(sv_dev).....	18
sv_video_format(sv_dev) .....	18
sv_format_height(v_format) .....	19
sv_format_width(v_format).....	19
sv_set_port(sv_dev, port) .....	20
sv_get_port(sv_dev) .....	20
sv_set_gain(sv_dev, value).....	21
sv_get_gain(sv_dev).....	21
sv_set_read_mask(sv_dev, mask) .....	22
sv_get_read_mask(sv_dev) .....	22
sv_set_trigger(sv_dev, edge) .....	23
sv_get_trigger(sv_dev, edge).....	23
sv_set_mon_out(sv_dev, output) .....	24
sv_get_mon_out(sv_dev).....	24
sv_set_bank(sv_dev, bank) .....	25
sv_get_bank(sv_dev).....	25

sv_set_merge(sv_dev, merge).....	26
sv_get_merge(sv_dev) .....	26
sv_set_interlace(sv_dev, intlc) .....	27
sv_get_interlace(sv_dev) .....	27
sv_get_genlock(sv_dev).....	28
sv_set_genlock(sv_dev, genlock) .....	28
sv_set_output_cmap(sv_dev, map) .....	29
sv_get_output_cmap(sv_dev, map).....	30
sv_enable_continuous(sv_dev, flag).....	31
sv_grab(sv_dev).....	32
sv_grab_mem(sv_dev, addr) .....	32
sv_appdone(sv_dev) .....	33
sv_framecnt(sv_dev) .....	33
sv_grabcnt(sv_dev) .....	34
sv_videofb_checksum(sv_dev, bank) .....	34
sv_merge(buf, width, height, outbuf).....	35
sv_write_rasfile(fname, addr, width, height) .....	35
sv_read_rasfile(fname, width, height, red, green, blue) .....	36
sv_b0_odd(sv_dev).....	37
sv_b0_even(sv_dev) .....	37
sv_b1_odd(sv_dev).....	38
sv_b1_even(sv_dev) .....	38
sv_videofb_to_memory(sv_dev, bank, memp).....	39
sv_videofb_to_hostfb(sv_dev, bank, memp, x, y, rowinc, skip).....	40
sv_memory_to_videofb(sv_dev, bank, memp).....	41
sv_hostfb_to_videofb(sv_dev, bank, memp, x, y, rowinc) .....	42
sv_partial_videofb_to_memory(sv_dev, bank, memp, sx, sy, dx, dy) .....	43
sv_partial_videofb_to_hostfb(sv_dev, bank, memp, x, y, rowinc, sx, sy, dx, dy).....	43
sv_partial_memory_to_videofb(sv_dev, bank, memp, sx, sy, dx, dy) .....	45
sv_partial_hostfb_to_videofb(sv_dev, bank, memp, x, y, rowinc, sx, sy, dx, dy).....	45
Registers.....	47
Brooktree Color Map Registers .....	48
Color Map RAM Data .....	50
Color Map Overlays .....	50
Read Mask.....	51
Command Register.....	51
Status Register.....	51
Reset Interrupt Register .....	52
Configuration Registers.....	53
Video Control Register .....	54
Interfaces .....	54
SBus Interface .....	54
Monitor Interface .....	54
Glossary.....	55
Specifications .....	57
Video Interface .....	57
Software.....	57
SBus Compliance.....	57
Power .....	57
Environmental.....	57

# S1V User's Guide

Physical.....	57
References.....	58
Free Imaging Software.....	58

## Figures

S1V Tool .....	7
S1V Frame Buffer Memory (RS-170 Version).....	11
S1V Frame Buffer Memory (CCIR Version) .....	11
S1V Memory .....	47

## Tables

Input Gain Required to Saturate the A–D Converter .....	49
Brooktree Color Map Register Definitions .....	49
Command Register Bit Definitions .....	51
Status Register Bit Definitions .....	52
Reset Interrupt Register Bit Definitions .....	52
Configuration Register Bit Definitions .....	53
Video Control Register Bit Definitions .....	54
Monitor Interface Timing Characteristics .....	54



## Overview

The S1V Video Capture and Frame Buffer Module provides grayscale video capture and output capability for the Sun SPARCstation. A version of the S1V is available that supports either 60 Hz RS-170 or 50 Hz CCIR signals. Unfiltered NTSC or PAL signals can also be digitized. Each pixel is defined by 8 bits of data, providing 256 levels of gray.

Using the S1V, you can connect an RS-170 or CCIR video camera and monitor to your Sun workstation and display the video input on your Sun display and on the monitor. You can also display Sun raster files on the video monitor and store your video input images as Sun raster files.

The S1V has 1 MB of RAM divided into two banks. Each bank stores one full frame of video data. You can therefore store two images, or use the banks to provide double-buffering. The S1V can capture an image into either bank; it can also display either bank.

The S1V has two inputs, A and B. The B input can be used as either a video input or as a frame capture trigger, allowing you to grab a frame in response to an external stimulus. If you are not using the external trigger, the board can continuously acquire video data, or it can operate in single-frame mode, acquiring one frame of video data at a time.

The S1V has two outputs. The MON output can show you either the contents of the S1V frame buffer or the current video input. The OUT output shows you the image in the S1V frame buffer.

The S1V allows you to set synchronization in several ways, depending on your application. (See **s1vconfig** on page 15 for details.)

The S1V also includes a standard SunOS or Solaris loadable device driver.

This document describes how to install the S1V video capture and frame buffer module and write applications for it. It is divided into the following sections:

<b>Installation</b>	describes how to install the S1V module and its related software.
<b>The S1V Tool</b>	explains how to use the graphical interface provided to control the S1V.
<b>Writing Applications</b>	details how to write applications for the S1V. It also describes the two example applications provided.
<b>S1V Video Library</b>	lists and explains the routines available to access the S1V hardware with your program.
<b>Registers</b>	describes the S1V registers, as well as the interface between the S1V and the SBus, and between the S1V and the video monitor.
<b>Glossary</b>	defines the terms and acronyms used in this document, in case any are unfamiliar to you.
<b>Specifications</b>	lists the specifications.
<b>References</b>	refers you to other documents and software that may prove useful to engineers writing applications for the S1V.

# Installation

Installing the S1V video capture and frame buffer module is a two-step process. First you must physically install the board inside the host computer. Then you must install the software driver, so that applications can access the S1V. Hardware installation is described in the following section. Software installation is described in the section after.

## Installing the Hardware

The S1V is a single-slot SBus board. Refer to your SBus host computer documentation for information on installing an SBus card. For example, for SPARCstation models 4/60 and 4/65, see the *SPARCstation Installation Guide*, Sun part number 800-4036-10, Appendix A: *Installing SBus Boards and Cards*.

Use the following procedure to install the S1V:

---

---

### CAUTION

Both the S1V and your SBus host computer contain static-sensitive components. Install the S1V at a static-free work area. If a static-free work area is not available, take the following precautions to reduce the risk of component damage:

1. Remove from the immediate area all materials that can generate or hold a static charge.
  2. Discharge yourself by touching both hands to a metal portion of the host computer's chassis before you open the host computer or open the S1V static-shielded bag.
- 
- 

1. Unpack the S1V from the shipping packaging. Do not remove the S1V from the static shielding bag until all you remove all other packaging materials from the area and established a static-free work area
2. Install the S1V in the SBus host, following the directions provided with the SBus host. The S1V can be installed in any slot.
3. Connect either the A or the B input (or both) to an RS-170 video camera using a standard 75-Ω coaxial cable (type RG-59) with BNC connectors on both ends. (The software library default is A.)
4. Connect either the MON or the OUT output (or both) to the VIDEO IN connector of an RS-170 video monitor, using the same kind of cable. (The software library default is that OUT displays bank 0 of the S1V frame buffer, and MON displays the input from the video camera.)

To remove the S1V, reverse the installation procedure.

## Installing the Software

The S1V can run on a Sun workstation using either SunOS version 4.1.x or Solaris version 2.x. The installation procedures differ. Both are given below.

### Using SunOS Version 4.1.x (Solaris 1.x)

If you are using SunOS Version 4.1, use the following procedure to install the S1V driver:

1. Become root or superuser.
2. Change to the directory in which you wish to install the S1V driver.
3. Place the diskette that came with the S1V into the diskette drive.
4. The driver and related files can be found on the diskette in tar format. To copy them to your hard disk, enter:

```
tar xvf /dev/rfd0
```

5. The tar program extracts a number of files. (The list of files distributed is provided in **Included Files** on page 4.) To install the driver, enter:

```
make install
```

6. The makefile provided installs and loads the S1V driver.
7. During the installation, the following question appears on the display:

```
Automatically load the slv driver during each reboot? [y|n] (y):
```

8. Entering *y* (or simply typing <Return>) causes the S1V driver to be loaded whenever you reboot your host computer. If you respond with *n*, you must manually reload the driver after rebooting. To do so, enter:

```
make load
```

9. During the installation, the following question appears on the display:

```
How many slv devices do you want? (1):
```

10. Enter the number corresponding to the number of S1V boards you have installed in your system. If you simply type <Return>, one driver is installed.

**NOTE: If you anticipate installing more than one S1V board into your system, install as many S1V drivers as you will ultimately require. The extra drivers will do no harm and will be there when you need them, saving you a step.**

11. If the S1V module has not been installed inside the host computer, or has been installed incorrectly, the following message appears on the display:

```
Can't load this module
```

12. If you see this message, go back to **Installing the Hardware** on page 2 and reinstall the board.

### Using SunOS 5.x (Solaris 2.x)

If you are using Sun System V Release 4 (Solaris 2.x), use the following procedure to install the S1V driver:

1. Become root or superuser.
2. Change to the directory in which you wish to install the S1V driver.

3. Place the diskette that came with the S1V into the diskette drive.
4. Tell the Solaris Volume Management to check for a new floppy in the drive, by typing
 

```
volcheck
```

 (Note: if you are running File Manager, it may bring up a "Format" pop-up at this point, alerting you to the presence of an unformatted floppy. Dismiss this window by selecting Cancel)
5. At the shell prompt, enter:
 

```
pkgadd -d /vol/dev/aliases/floppy0 EDTslv
```
6. Move or copy the files `xilioslv.so*` to the directory `$XILHOME/lib/pipelines`. (Set the environment variable `$XILHOME` if it is not already set.)

To remove the S1V driver:

1. Become root or superuser.
2. Enter:

```
pkgrm EDTslv
```

For further details, consult your Solaris 2.0 documentation, or call the Engineering Design Team.

## Included Files

The S1V video capture and frame buffer software contains the following files (see the *readme* file for a complete, up-to-date listing).

<code>s1v.o.sun4c</code>	The executable S1V driver for SunOS 4.1.3 on a Sun 4C architecture such as a SPARCStation 1, 1+, 2, or IPC.
<code>s1v.o.sun4m</code>	The executable S1V driver for SunOS 4.1.3 on a Sun 4M architecture such as a SPARCStation 5, 10, 20, LX, Classic, or an Ultra 1 or 2.
<code>s1v</code>	The executable S1V driver for Solaris Version 2.x.
<code>makefile</code>	The makefile for installing, loading, and unloading the S1V driver in SunOS Version 4.1.x.
<code>README</code>	An ASCII file containing last-minute information about the S1V software.
<code>s1vtool</code>	A graphical user interface for controlling the S1V board inputs, outputs, gain, and external trigger.
<code>s1vlib.c</code>	Source for the driver interface routines to support the example programs and user application programs.
<code>s1vlib.h</code>	Header file for the driver interface routines.
<code>s1vlib.o</code>	Executable driver interface routines.
<code>s1vxfer.c</code>	Utility file for the driver interface routines.
<code>s1vxfer.o</code>	Utility file for the driver interface routines.
<code>s1v.h</code>	A header file for the S1V driver input and output controls.
<code>s1vreg.h</code>	A header file for the S1V driver registers.
<code>s1vtest.c</code>	A general-purpose diagnostic program, including tests of the frame buffer memory, color maps, read mask, and registers.

- `slvread.c` An example program that shows different ways to read a frame from the S1V and display the image on the host display, including grabbing multiple frames and partial frames.
- `slvwrite.c` An example program that reads from a Sun raster file, places the resulting image in the S1V memory, and displays the results on the S1V monitor.
- `framecount.c` An example program that outputs frame numbers to the monitor, as well as 'E' when an even field is being displayed and 'O' when an odd field is being displayed. This can be useful for diagnostic purposes.
- `grayscale.c` An example program that displays a gray scale on the S1V monitor.
- `mread.c` An example program that places the S1V in continuous mode and acquires the number of frames provided as an argument to the executable, showing the use of the `read()` system call to do so.
- `slvconfig.c` An example program that configures the S1V driver.
- `xwin.c` An example program that acquires a frame and displays it in an X window.
- `setdebug.c` Sets the debug levels for the S1V.
- Debug level number:
1. displays driver status in console in response to `modstat(8)` (`modinfo` for Solaris)
  2. shows history of events with time stamps in response to `modstat(8)` (`modinfo` for Solaris)
- Interrupt debug level number 1 shows events as they occur, with time stamps.
- `xiltake.c` An example program that shows the use of the X Imaging Library interface for the S1V under Solaris. See the Sun AnswerBook documentation for the X Imaging Library.
- `xilios1v.so` The S1V library for the X Imaging Library interface (Solaris only). See the Sun AnswerBook documentation for the X Imaging Library. This file must reside in the directory `$XILHOME/lib/pipelines`. (Set the environment variable `$XILHOME` if it is not already set.)

In addition, Solaris users receive executable files for all the example programs.

## The S1V Tool

The S1V Tool (`s1vtool`) is an X-Windows based application program which provides a graphical user interface to the S1V board on your host display. It allows you to acquire and display images on the screen and control nearly all aspects of the S1V operation.

To run the `s1vtool`, you must be running an Xwindows display manager. If you are running under the Solaris 2.x operating system, you will need to make sure that the dynamically linked X Imaging Library (XIL) path has been added to your `$LD_LIBRARY_PATH` environment variable in order to run the `s1vtool` or other s1v applications that use XIL. Assuming that the XIL files are in their default location, a the most common way to do this is to add the following lines in your `.login` file *after* any existing `LD_LIBRARY_PATH`:

```
setenv XILHOME /opt/SUNWits/Graphics-sw/xil
setenv LD_LIBRARY_PATH "$XILHOME/lib:$LD_LIBRARY_PATH"
```

## Invoking the S1V Tool

Invoke the S1V tool with the command:

```
s1vtool
```

## Options

- `-u n` specifies which s1v unit (default 0)
- `-a` use all 256 colormap entries (default 248 -- reserves the bottom 8 for the window manager)
- `-x` enables standard X updates instead of XIL. (Solaris 2.X only - use if `$XILHOME/lib/pipelines` file is not installed)
- `-r` enables raw display using `pixrect` - faster but no clip (Solaris 1.x only)

By default, the `s1vtool` uses 248 shades of gray. If you are running on an 8 bit display, the `s1vtool` will install its own grayscale color map in the high 248 colormap entries any time the mouse cursor is in its display window. If you are running OPEN Windows on an 8 bit display with anything other than a grayscale colormap, you may lack some gray shades and your display may change its appearance whenever you move the cursor in to or out of the S1V tool window. In addition, using 248 shades of gray will result in some loss of grayscale resolution in your image. To avoid either of these conditions you will need to bring up OPEN Windows with a grayscale colormap. To do this, invoke OPEN Windows with the following command:

```
openwin -dev /dev/fb grayvis
```

And run S1Vtool with the `-a` option.

Normally, the S1V tool uses the X window server to update the display. This may be necessary under certain circumstances, for example, if you are running over a network. However, if it is not necessary, you can update the display much more quickly by invoking the S1V tool with the argument `-x` (standard X mode):

```
s1vtool -x
```

Under SunOS 4.1.x, the default mode uses `pixrect`, which bypass the X window server. Therefore, windows do not update as expected when running the S1V Tool unless you use standard X mode.

Under Solaris, the default mode uses XIL. It does not bypass the X Window server, but the files `xilios1v.so*` must be found in the directory `$XILHOME/lib/pipelines`.

The S1V tool appears as shown below.



Figure 1. S1V Tool

## S1V Tool User Interface Reference

From the top left to the bottom right, the S1V tool buttons work as follows:

- Update** This controls the acquisition mode of the S1V. *Continuous* means that the board acquires one frame of video data and then immediately begins to acquire the next. In continuous acquisition mode, first bank 0 and then bank 1 is used. Then bank 0 is reused, overwriting the image it contained previously.
- Single* means that the board acquires one video image and then waits for this button to be pressed again.
- Load** loads the Sun raster file named in the **filename** box into the S1V frame buffer. (A full pathname can be specified.)

- Save** saves the image in the S1V frame buffer to a Sun raster file named as specified in the **filename** box. (A full pathname can be specified.) After typing the filename, press <Return> and then press the **Save** button.
- Quit** quits the S1V tool application.
- Gain** allows you to specify the amount of gain by moving the slider, or clicking to either side. The input gain is the amount by which to exaggerate or minimize differences in the input signal. This control allows you to adjust for differences in particular video cameras.
- The minimum input gain is 0, the default is 8, and the maximum value is 15. Values of 0–7 decrease gain by minimizing differences. Values of 9–15 increase gain by exaggerating them. The default is set to be compatible with a typical 700 mV RS-170 or CCIR signal.
- Monitor** allows you to specify whether the video monitor is to show the contents of the S1V frame buffer, or the input coming in directly from the video camera. If you choose the frame buffer, it shows you the contents of bank 0 and bank 1 alternately each time you press the Frame Buffer button.
- Input** allows you to specify whether the S1V is to receive its video input from either A or B. Choose the input to which you have connected the video camera.

**NOTE:** The S1V tool does not allow you to use the external trigger function.

## Writing Applications

A typical application using the S1V to display or process video data includes the following steps:

1. Open the device.
2. Set the device to continuous mode.
3. Use the `read()` system call to acquire a specified number of frames.
4. Close the device.

Another typical application approach is:

1. Open the device.
2. Set the device to continuous mode.
3. Loop over the following three steps as many times as required:
  - a. Acquire a frame using `sv_grab`.
  - b. Process the frame.
  - c. Tell the driver that the application is ready for the next frame.
4. Close the device.

For your convenience, EDT has provided a library of routines to access the S1V and the video data it has acquired. We encourage you to use this library if you wish your application to be portable between different versions of the S1V, including future versions. The library routines are described in detail in **S1V Video Library** on page 16.

## Opening the Device

In order to open the device, the S1V driver must be loaded into your system. If you have not already done so, instructions are provided in **Installing the Software** on page 3.

When you open the device (using the `sv_open` routine), it returns a device handle for the other library routines to use, and can lock the device against any other attempts to access it. After you have opened the device, you are accessing the hardware directly, bypassing the SunOS kernel.

## Configuring the Device

Before you can acquire an image, you must configure the device appropriately:

- You can acquire frames after each field instead of each frame—that is, sixty times a second instead of thirty—using the `sv_set_interlace` routine.
- If you wish to acquire image data at a faster rate, it is also possible to grab just a portion of a frame instead of the entire frame. You can supply a starting `x, y` and a delta `x, y` to any data transfer routine whose name includes the word `partial` to specify the top left and bottom right corners of the rectangle of interest.
- When you read a frame from a bank on the S1V to host memory, you can choose whether you wish to merge even and odd lines or leave them separate. Merging even and odd lines is the default; it provides a complete picture but takes longer.

- Finally, you must set the device in the appropriate acquisition mode: continuous or single-frame.

## Acquiring an Image

Single-frame acquisition acquires one frame at a time. Continuous acquisition acquires frames continuously, which can be done in several ways.

### Single-Frame Acquisition

In single-frame acquisition mode, you can use the `sv_set_bank` routine to set the bank into which the S1V puts the frame. Acquire the frame with `sv_grab`; this routine always places the image in the selected bank and returns the bank into which the image was placed.

### Continuous Acquisition

In continuous acquisition mode, the driver fills first one bank and then the next, cycling between the two banks at each vertical synchronization—that is, 30 times per second. When `sv_grab` is called, it returns the last bank into which it acquired data. The S1V driver then continuously acquires frames into only the other bank, while the application processes the image in the bank returned by `sv_grab`. The driver does not cycle between both banks again until the application calls the routine `sv_appdone`. This prevents data loss in cases where the application and platform, working together, cannot copy data as fast as the S1V can acquire it.

On some platforms, a continuous series of whole frames can be acquired using the `read()` system call by supplying an argument specifying the amount of data to acquire: that is, the number of frames times the width of the frame times the height of the frame. Compare the results returned by the routines `sv_framecnt` and `sv_grabcnt` to ascertain whether the driver missed any contiguous frames.

**NOTE:** If you are acquiring data by means of sequential reads (bypassing `sv_grab`) and a read call fails, returning `-1`, it may be because you have acquired over 2 GB of information, causing the file offset to become negative and returning an `EINVAL` error. To work around the problem, reset the file offset to 0, 0 after a certain number of reads. For example, if you are acquiring 4 MB of data per read, include the following line in your application after every 500 reads:

```
(void) lseek (fd, 0, 0)
```

where `fd` is the file descriptor for the device.

## Accessing the Data

The S1V frame buffer memory is arranged as shown in the figures below.

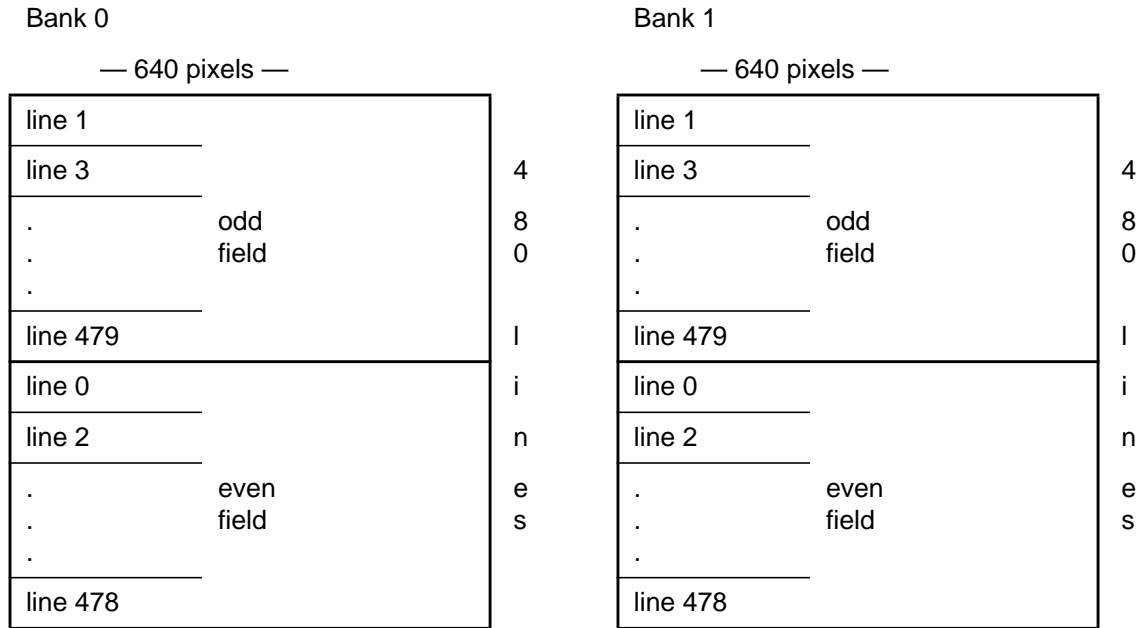


Figure 2. S1V Frame Buffer Memory (RS-170 Version)

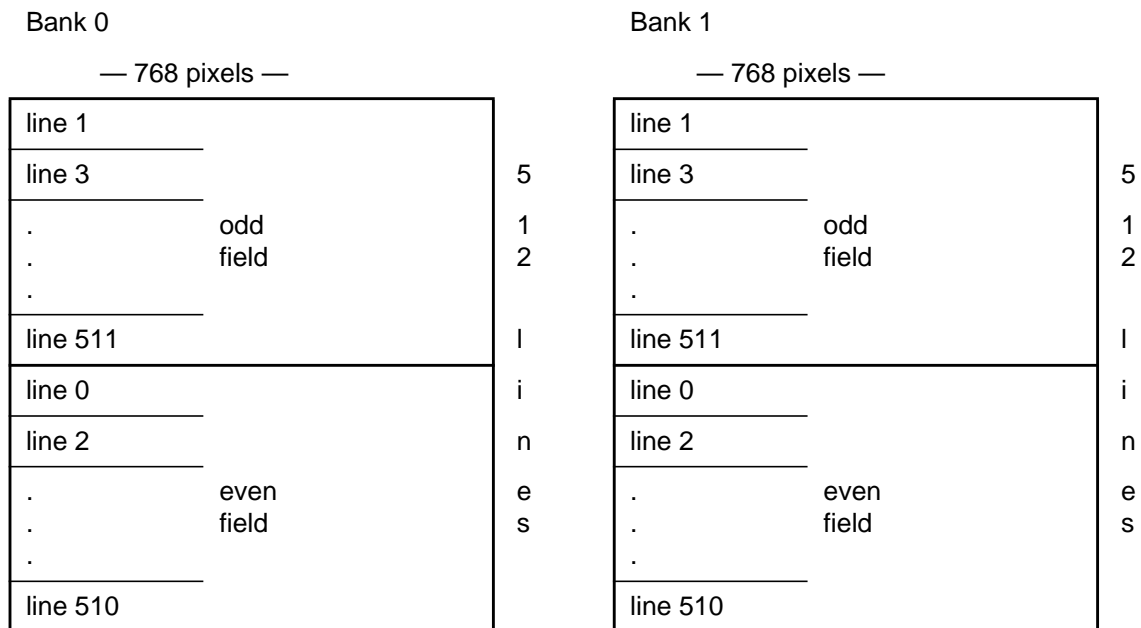


Figure 3. S1V Frame Buffer Memory (CCIR Version)

The video data is divided into fields of even- and odd-numbered lines to support interlacing.

To access the data, copy the image from the S1V frame buffer to the application memory space in your host computer, using the `sv_videofb_to_memory` routine. The S1V transfers the data from the selected bank.

After vertical synchronization, you can change the selected bank (using the `sv_set_bank` routine) and transfer the image in the other bank.

When the video data is in the host memory, it is no longer divided into even and odd fields; instead, the lines are arrayed in consecutive order if it was acquired as merged. If it was acquired unmerged, however, the video data remains divided into even and odd fields when it is in host memory.

**NOTE: The data transfer routines use driver routines and, on some platforms, take advantage of hardware copy support. This is much faster in most cases, but if you are accessing single pixels or small rectangles, use the pointers returned by `sv_b0_odd`, `sv_b0_even`, `sv_b1_odd`, or `sv_b1_even` instead.**

## Displaying the Data

To display the video image you have acquired on the host computer, copy the image from the S1V frame buffer to the host computer frame buffer, using the `sv_videofb_to_hostfb` routine. The S1V transfers the data from the selected bank. After vertical synchronization, you can change the selected bank (using the `sv_set_bank` routine) and display the image in the other bank.

When the video data is in the host frame buffer, it is no longer divided into even and odd fields. Instead, the lines are arrayed in consecutive order.

You may be using the host computer for image processing. After processing, you may then wish to display the image on the video monitor. To do so, copy the image from the application memory space in your host computer to the S1V frame buffer, using the `sv_memory_to_videofb` routine.

## Example Applications

To help you get started, three example applications have been provided: `slvtest.c`, `slvread.c`, and `slvwrite.c`. They are explained below.

- `slvtest.c`     A general-purpose diagnostic program, including tests of the frame buffer memory, color maps, read mask, and registers.
- `slvread.c`     This program reads a frame from the S1V frame buffer and displays it on the host display.
- `slvwrite.c`     This program reads from a file containing an image, places the resulting image in the S1V frame buffer, and displays the results on the video monitor.
- `framecount.c` An example program that outputs frame numbers to the monitor, as well as 'E' when an even field is being displayed and 'O' when an odd field is being displayed. This can be useful for diagnostic purposes.
- `grayscale.c`     An example program that displays a gray scale on the S1V monitor.
- `mread.c`         An example program that places the S1V in continuous mode and acquires the number of frames provided as an argument to the executable, showing the use of the `read()` system call to do so.
- `slvconfig.c`     An example program that configures the S1V driver.
- `xwin.c`          An example program that acquires a frame and displays it in an X window.

The example programs are described in more detail below.

**s1vread**

Reads a frame from the S1V frame buffer and displays it on the host display or the video monitor, or stores it in a file.

**Usage**

```
s1vread [-c n] [-f filename] [-i imagefile] [-p n] [-b n] [-s x y dx dy] [-m] [-k] [-t] [-n] [-u n]
```

**Arguments**

- `-c n` Specifies *n* captures.
- `-f filename` Creates a file consisting of the hexadecimal bytes captured, and saves it to *filename*.
- `-i imagefile`  
Creates a Sun raster file and saves it to *imagefile*.
- `-p n` Sets the input port to 0 or 1. The default is 0.
- `-b n` Sets the bank to use for a single capture. Valid values are 0 or 1. The default is the last selected bank.
- `-s x y dx dy`  
Captures just a portion of the video image. The top left corner of the portion to capture is specified by *x* and *y*. The size of the rectangle to capture is specified by *dx* and *dy*.
- `-m` Saves the captured frames as a movie.
- `-k` Returns the checksum—the sum of all pixel values.
- `-t` Reads the image to a local buffer, in order to time data transfer.
- `-n` Reads the image stored in the selected bank without acquiring a new image.
- `-u n` Sets the unit, if more than one S1V device is installed.

**Notes**

You must invoke this program with at least one of the arguments:  
`[-f filename] [-i imagefile]`

This program starts with bank 0 the first time it is invoked

**mread**

Uses the `read()` system call to acquire the specified number of frames as sequential reads.

**Usage**

```
mread [-u n] [-c n]
```

**Arguments**

- `-u n` Sets the unit, if more than one S1V device is installed.
- `-c n` Specifies the number of frames to acquire. The default is 1.

**s1vwrite**

Reads from a file containing an image, or from the host frame buffer, places the resulting image in the S1V frame buffer, and displays the results on the video monitor.

**Usage**

```
s1vwrite [-s] [-f filename] [-i imagefile] [-c n] [-d x y] [-u n]
```

**Arguments**

- s                   Scrolls the image if it is larger than a frame.
- f *filename*       Reads from *filename*, a file consisting of hexadecimal bytes captured by an S1V.
- i *imagefile*       Reads from a Sun raster file named *imagefile*.
- c *n*               Writes *n* times.
- d *x y*             The *x* and *y* coordinates of the host frame buffer to display on the video monitor.
- u *n*               Sets the unit, if more than one S1V device is installed.

**Notes**

You must invoke this program with one and only one of the arguments:  
[-d *x y*] [-f *filename*] [-i *imagefile*].

**framecount**

Counts the number of times that the S1V has filled.

**Usage**

```
framecount [-u n] [-x n] [-y n] [-l n]
```

**Arguments**

- u *n*               Sets the unit, if more than one S1V device is installed.
- x *n*               Specifies *x* square. Valid values are 0–20. The default is 2.
- y *n*               Specifies *y* square. Valid values are 0–20. The default is 2.
- l *n*               Sets the loop count. A count of 0 loops forever.

**grayscale**

Displays a gray scale on the S1V monitor.

**Usage**

```
grayscale [-u n]
```

**Arguments**

- u *n*               Sets the unit, if more than one S1V device is installed.

## s1vconfig

Configures the S1V.

### Usage

```
s1vconfig [-u n] [-l] [-f] [-b n] [-d] [-v] [-x] [-n] [-i] [-m] [-s]
```

### Arguments

- u *n*            Sets the unit, if more than one S1V device is installed.
- l               Sets the monitor to live input.
- f               Sets the monitor to frame buffer (the default).
- b *n*            Sets the selected bank. Valid values are 0 or 1. The default is bank 0.
- d               Enables digital genlock (the synchronization signal).
- v               Enables VCO genlock (the synchronization signal).
- x               Enables internal genlock (the synchronization signal).
- n               Sets the unit to noninterlaced mode. The S1V reads only one field of each frame.
- i               Sets the unit to interlaced mode. The S1V reads both fields of each frame.
- m               Sets the unit to merge even and odd lines when the S1V reads a frame from a bank to host memory, which provides better output but takes longer.
- s               Sets the unit not to merge even and odd lines when the S1V reads a frame from a bank to host memory, which is faster.

### Notes

If you are using an input device with the S1V, invoke this program with the `-v` argument. If you have trouble synchronizing with the video source, invoke this program with the `-d` argument instead. For output only, invoke it with the `-x` argument instead.

## S1V Video Library

The S1V driver supports the following routines, which are documented on the page specified:

<b>Housekeeping</b>	<b>See page</b>	<b>Acquisition</b>	<b>See page</b>
sv_open	17	sv_enable_continuous	31
sv_close	17	sv_grab	32
sv_set_defaults	18	sv_grab_mem	32
sv_video_format	18	sv_appdone	33
sv_format_height	19	sv_framecnt	33
sv_format_width	19	sv_grabcnt	34
		sv_videofb_checksum	34
<b>Configuration</b>	<b>See page</b>	<b>Data Transfer</b>	<b>See page</b>
sv_set_port	20	sv_merge	35
sv_get_port	20	sv_write_rasfile	35
sv_set_gain	21	sv_read_rasfile	36
sv_get_gain	21	sv_b0_odd	37
sv_set_read_mask	22	sv_b0_even	37
sv_get_read_mask	22	sv_b1_odd	38
sv_set_trigger	23	sv_b1_even	38
sv_get_trigger	23	sv_videofb_to_memory	39
sv_set_mon_out	24	sv_videofb_to_hostfb	40
sv_get_mon_out	24	sv_memory_to_videofb	41
sv_set_bank	25	sv_hostfb_to_videofb	42
sv_get_bank	25	sv_partial_videofb_to_memory	43
sv_set_merge	26	sv_partial_videofb_to_hostfb	43
sv_get_merge	26	sv_partial_memory_to_videofb	45
sv_set_interlace	27	sv_partial_hostfb_to_videofb	45
sv_get_interlace	27		
sv_set_genlock	28		
sv_get_genlock	28		
sv_set_output_cmap	29		
sv_get_output_cmap	30		

In addition to the library routines described above, the S1V under Solaris 2.x also includes an X Imaging Library interface. To use it for grabbing and displaying an image, use `xil_create_from_device` and `xil_copy`, passing the string "ioslv" as the device argument. An example of this is available in the file `xiltake.c`. `xil_create_from_device` and `xil_copy` are standard routines supplied by Solaris. See the Sun AnswerBook documentation for more information about these routines.

## sv\_open(device\_name, lock)

### Description

Opens the device, the first step in accessing the hardware.

If multiple S1V devices are installed, they are named s1v0, s1v1, s1v2, etc.

### Arguments

*device\_name* The pathname of the device to be opened, such as /dev/s1v0.

*lock* Defined values are SV\_LOCKDEV or SV\_NOLOCKDEV. SV\_LOCKDEV locks the device with a nonblocking exclusive lock (see *flock (2)* in the SunOS documentation). SV\_NOLOCKDEV does not check for a lock condition; therefore, the device can be opened even if it has previously been accessed.

### Returns

A pointer to the S1V data structure, if successful. This structure holds the addresses of the S1V frame buffer and registers, and a file descriptor for the open device.

NULL if unsuccessful.

### Example

```
struct SvDev *dv = sv_open("/dev/s1v0", SV_LOCKDEV);
```

### Syntax

```
#include "s1vlib.h"

SvDev * /* The SvDev structure is defined in s1vlib.h */
sv_open(device_name, lock)
char *device_name;
int lock;
```

## sv\_close(sv\_dev)

### Description

Closes the device and unlocks it if it has been locked.

### Arguments

*sv\_dev* The pointer to the device structure.

### Returns

0 if successful. -1 if unsuccessful.

### Example

```
sv_close(sv_dev);
```

### Syntax

```
#include "s1vlib.h"

int
sv_close (sv_dev)
SvDev *sv_dev; /*pointer to the device structure */
```

## **sv\_set\_defaults(sv\_dev)**

### **Description**

Sets the device to its default values. The acquisition mode is set to single-frame. The input port is set to port A. The color map is set to a grayscale color map in which 0 = black, 255 = white, and intermediate gray values are linearly interpolated. The selected bank of the frame buffer is set to bank 0. The OUT output shows the image in the frame buffer bank 0, and the MON output shows the image from the input port A.

### **Arguments**

*sv\_dev*            The pointer to the device structure.

### **Returns**

0 if successful. -1 if unsuccessful.

### **Example**

```
sv_set_defaults(sv_dev);
```

### **Syntax**

```
#include "slvlib.h"

int
sv_set_defaults (sv_dev)
SvDev *sv_dev;    /* pointer to the device structure */
```

## **sv\_video\_format(sv\_dev)**

### **Description**

Returns the video format.

### **Arguments**

*sv\_dev*            The pointer to the device structure.

### **Returns**

SV\_RS170 if the video format is RS-170.  
SV\_CCIR if the video format is CCIR.  
-1 if unsuccessful.

### **Example**

```
sv_video_format(sv_dev);
```

### **Syntax**

```
#include "slvlib.h"

int
sv_video_format (sv_dev)
SvDev *sv_dev;    /* pointer to the device structure */
```

## sv\_format\_height(v\_format)

### Description

A macro that returns the number of lines in an RS-170 or a CCIR image.

### Arguments

*v\_format* Specifies the video format, either SV\_RS170 or SV\_CCIR.

### Returns

SV\_RS170\_HEIGHT if *v\_format* is SV\_RS-170.

SV\_CCIR\_HEIGHT if *v\_format* is SV\_CCIR.

-1 if unsuccessful.

### Example

```
int vheight = sv_format_height(
    sv_video_format (sv_dev));
```

### Syntax

```
#include "slvlib.h"

int
sv_format_height (v_format)
```

## sv\_format\_width(v\_format)

### Description

A macro that returns the number of pixels per line in a square-pixel RS-170 or a CCIR image.

### Arguments

*v\_format* Specifies the video format, either SV\_RS170 or SV\_CCIR.

### Returns

SV\_RS170\_WIDTH if *v\_format* is SV\_RS-170.

SV\_CCIR\_WIDTH if *v\_format* is SV\_CCIR.

-1 if unsuccessful.

### Example

```
int vwidth = sv_format_width(sv_video_format(sv_dev));
```

### Syntax

```
#include "slvlib.h"

int
sv_format_width (v_format)
```

## sv\_set\_port(sv\_dev, port)

### Description

Selects one of the two possible input ports, A or B.

### Arguments

*sv\_dev*            The pointer to the device structure.  
*port*                Either SV\_INPUT\_A or SV\_INPUT\_B.

### Returns

0 if successful. -1 if unsuccessful.

### Example

```
sv_set_port(sv_dev, SV_INPUT_A);
```

### Syntax

```
#include "slvlib.h"

int
sv_set_port (sv_dev, port)
SvDev *sv_dev; /* pointer to the device structure */
int port;      /* which port to receive input from */
```

## sv\_get\_port(sv\_dev)

### Description

Determines which of the two possible input ports, A or B, is presently selected to receive input.

### Arguments

*sv\_dev*            The pointer to the device structure.

### Returns

Either SV\_INPUT\_A or SV\_INPUT\_B if successful.  
-1 if unsuccessful.

### Example

```
int port = sv_get_port(sv_dev);
```

### Syntax

```
#include "slvlib.h"

int
sv_get_port (sv_dev)
SvDev *sv_dev; /* pointer to the device structure */
```

## sv\_set\_gain(sv\_dev, value)

### Description

Sets the input gain, a 4-bit value specifying the amount by which to exaggerate or minimize differences in the input signal. Input gain allows you to adjust for differences in particular video cameras.

The minimum input gain is 0, the default is 8, and the maximum value is 15. Values of 0–7 decrease gain by minimizing differences. Values of 9–15 increase gain by exaggerating them. The default is set to be compatible with a typical 700 mV RS-170 or CCIR signal.

### Arguments

*sv\_dev*            The pointer to the device structure.  
*value*             An integer in the range of 0–15.

### Returns

0 if successful, -1 if unsuccessful.

### Example

```
sv_set_gain(sv_dev, 0);
```

### Syntax

```
#include "slvlib.h"
int
sv_set_gain (sv_dev, value)
SvDev *sv_dev;    /* pointer to the device structure */
int value;        /* the amount by which to adjust input gain */
```

## sv\_get\_gain(sv\_dev)

### Description

Gets the current input gain, a 4-bit value specifying the amount by which to exaggerate or minimize differences in the input signal.

The minimum input gain is 0, the default is 8, and the maximum value is 15. Values of 0–7 decrease gain by minimizing differences. Values of 9–15 increase gain by exaggerating them. The default is set to be compatible with a typical 700 mV RS-170 or CCIR signal.

### Arguments

*sv\_dev*            The pointer to the device structure.

### Returns

An integer in the range of 0–15 if successful.  
-1 if unsuccessful.

### Example

```
int gain = sv_get_gain(sv_dev);
```

### Syntax

```
#include "slvlib.h"
int
sv_get_gain (sv_dev)
SvDev *sv_dev;    /* pointer to the device structure */
```

## sv\_set\_read\_mask(sv\_dev, mask)

### Description

Sets the read mask for each of the eight bits representing a pixel. The 8-bit read mask is logically ANDed with the frame buffer pixel data before the pixel data is used to address the color map. Using this register, individual bits of each pixel can be masked.

### Arguments

*sv\_dev*            The pointer to the device structure.  
*mask*              An unsigned char. 0 = mask this bit. 1 = do not mask this bit.

### Returns

0 if successful. -1 if unsuccessful.

### Example

```
sv_set_read_mask(sv_dev, 0xfe); /*sets bits to 1111 1110 */
```

### Syntax

```
#include "slvlib.h"

int
sv_set_read_mask (sv_dev, mask)
SvDev *sv_dev;        /* pointer to the device structure */
unsigned char mask;   /* 8-bit pixel read mask */
```

## sv\_get\_read\_mask(sv\_dev)

### Description

Determines the value of the read mask for each of the eight bits representing a pixel. The 8-bit read mask is logically ANDed with the frame buffer pixel data before the pixel data is used to address the color map. Using this register, individual bits of each pixel can be ignored on the display.

### Arguments

*sv\_dev*            The pointer to the device structure.

### Returns

An unsigned char. 0 = mask this bit. 1 = do not mask this bit. -1 if unsuccessful.

### Example

```
int mask = sv_get_read_mask(sv_dev);
```

### Syntax

```
#include "slvlib.h"

int
sv_get_read_mask (sv_dev)
SvDev *sv_dev;        /* pointer to the device structure */
```

## sv\_set\_trigger(sv\_dev, edge)

### Description

Enables the external trigger from input B and determines whether acquisition is to occur on a positive edge (voltage rising) or a negative edge (voltage falling). Use `sv_grab` to begin acquiring video data when the trigger event occurs.

### Arguments

`sv_dev`            The pointer to the device structure.  
`edge`                `SV_P_EDGE` for a positive edge or `SV_N_EDGE` for a negative edge.

### Returns

0 if successful, -1 if unsuccessful.

### Example

```
sv_set_trigger(sv_dev, SV_N_EDGE);
```

### Syntax

```
#include "slvlib.h"
int
sv_set_trigger (sv_dev, edge)
SvDev *sv_dev; /* pointer to the device structure */
int edge;      /* trigger event occurs when rising or falling */
```

## sv\_get\_trigger(sv\_dev, edge)

### Description

Determines whether the external trigger has been enabled from input B and if so, determines whether acquisition is to occur on a positive edge (voltage rising) or a negative edge (voltage falling).

### Arguments

`sv_dev`            The pointer to the device structure.  
`edge`                `SV_P_EDGE` for a positive edge or `SV_N_EDGE` for a negative edge.

### Returns

1 if trigger is enabled, 0 if trigger is not enabled, -1 if unsuccessful.

**NOTE: If the trigger is not enabled, the value of `*edge` is undefined.**

### Example

```
int ena;
int edge;

ena = sv_get_trigger(sv_dev, &edge);
```

### Syntax

```
#include "slvlib.h"

int
sv_get_trigger (sv_dev, edge)
SvDev *sv_dev; /* pointer to the device structure */
int *edge;     /* pointer to the trigger edge */
```

## sv\_set\_mon\_out(sv\_dev, output)

### Description

Sets the source of the monitor output, either the video input or the S1V frame buffer.

### Arguments

*sv\_dev*            The pointer to the device structure.  
*output*            Either SV\_MON\_IN (video input) or SV\_MON\_FB (S1V frame buffer).

### Returns

0 if successful, -1 if unsuccessful.

### Example

```
sv_set_mon_out(sv_dev, SV_MON_IN);
```

### Syntax

```
#include "slvlib.h"

int
sv_set_mon_out (sv_dev, output)
SvDev *sv_dev;  /* pointer to the device structure */
int output;     /* the active output source */
```

## sv\_get\_mon\_out(sv\_dev)

### Description

Determines whether the source of the monitor output is the video input or the S1V frame buffer.

### Arguments

*sv\_dev*            The pointer to the device structure.

### Returns

SV\_MON\_IN if monitor output is coming from the video input.  
SV\_MON\_FB if monitor output is coming from the S1V frame buffer.  
-1 if unsuccessful.

### Example

```
int output = sv_get_mon_out(sv_dev);
```

### Syntax

```
#include "slvlib.h"

int
sv_get_mon_out (sv_dev)
SvDev *sv_dev;  /* pointer to the device structure */
```

## sv\_set\_bank(sv\_dev, bank)

### Description

Sets whether data from the next frame acquisition is to go to bank 0 or bank 1.

### Arguments

*sv\_dev*            The pointer to the device structure.  
*bank*              Either SV\_BANK0 or SV\_BANK1.

### Returns

0 if successful, -1 if unsuccessful.

### Example

```
sv_set_bank(sv_dev, SV_BANK1);
```

### Syntax

```
#include "s1vlib.h"

int
sv_set_bank (sv_dev, bank)
SvDev *sv_dev; /* pointer to the device structure */
int bank;      /* selected bank of the S1V frame buffer */
```

## sv\_get\_bank(sv\_dev)

### Description

Determines which bank of the S1V frame buffer is to receive data from the next frame acquisition.

### Arguments

*sv\_dev*            The pointer to the device structure.

### Returns

SV\_BANK0, SV\_BANK1, or -1 if unsuccessful.

### Example

```
int bank = sv_get_bank(sv_dev);
```

### Syntax

```
#include "s1vlib.h"

int
sv_get_bank (sv_dev)
SvDev *sv_dev; /* pointer to the device structure */
```

## sv\_set\_merge(sv\_dev, merge)

### Description

Sets whether even and odd fields should be merged or acquired separately.

### Arguments

*sv\_dev*            The pointer to the device structure.  
*merge*             Either 1 for merged (normal acquisition) or 0 for separate (faster).

### Returns

0 if successful, -1 if unsuccessful.

### Example

```
sv_set_merge(sv_dev, 1);
```

### Syntax

```
#include "slvlib.h"

int
sv_set_merge (sv_dev, merge)
SvDev *sv_dev; /* pointer to the device structure */
int merge;     /* merge fields or not */
```

## sv\_get\_merge(sv\_dev)

### Description

Determines whether the S1V is acquiring data merged or unmerged.

### Arguments

*sv\_dev*            The pointer to the device structure.

### Returns

1 for merged, 0 for unmerged, or -1 if unsuccessful.

### Example

```
int merge = sv_get_merge(sv_dev);
```

### Syntax

```
#include "slvlib.h"

int
sv_get_merge (sv_dev)
SvDev *sv_dev; /* pointer to the device structure */
```

## sv\_set\_interlace(sv\_dev, intlc)

### Description

Sets whether the S1V acquires data in interlaced mode or not. In interlaced mode, the S1V acquires two fields—the odd field and the even field—before switching banks. In noninterlaced mode, the S1V switches banks after acquiring only one field.

### Arguments

*sv\_dev*            The pointer to the device structure.  
*intlc*             Either 0 for noninterlaced or 1 for interlaced.

### Returns

0 if successful, -1 if unsuccessful.

### Example

```
sv_set_interlace(sv_dev, 1);
```

### Syntax

```
#include "slvlib.h"

int
sv_set_interlace (sv_dev, intlc)
SvDev *sv_dev; /* pointer to the device structure */
int intlc;     /* interlace mode */
```

## sv\_get\_interlace(sv\_dev)

### Description

Determines whether the S1V acquires data in interlaced mode or not.

### Arguments

*sv\_dev*            The pointer to the device structure.

### Returns

0 for noninterlaced, 1 for interlaced, or -1 if unsuccessful.

### Example

```
int intlc = sv_get_interlace(sv_dev);
```

### Syntax

```
#include "slvlib.h"

int
sv_get_interlace (sv_dev)
SvDev *sv_dev; /* pointer to the device structure */
```

## sv\_get\_genlock(sv\_dev)

### Description

Determines the source of the S1V clock synchronization signal.

### Arguments

*sv\_dev*            The pointer to the device structure.

### Returns

SV\_LOCK\_VCO, SV\_LOCK\_DIG, SV\_LOCK\_XTL, or -1 if unsuccessful.

### Example

```
int genlock = sv_get_genlock(sv_dev);
```

### Syntax

```
#include "slvlib.h"

int
sv_get_genlock (sv_dev)
SvDev *sv_dev; /* pointer to the device structure */
```

## sv\_set\_genlock(sv\_dev, genlock)

**Description** Sets the source of the S1V clock synchronization signal.

### Arguments

*sv\_dev*            The pointer to the device structure.

*genlock*            Either SV\_LOCK\_VCO, SV\_LOCK\_DIG, or SV\_LOCK\_XTL.  
 SV\_LOCK\_VCO locks the frame buffer and output synchronization to the camera input synchronization using the voltage-controlled oscillator, recommended if an input device is used.  
 SV\_LOCK\_DIG locks the frame buffer and output synchronization to the camera input synchronization using the internal crystal clock, recommended only if the input device has a noisy signal in the time dimension, as the signal will show some jitter.  
 SV\_LOCK\_XTL generates an internal synchronization signal, which is useful for output only.

### Returns

0 if successful, -1 if unsuccessful.

### Example

```
sv_set_genlock(sv_dev, SV_LOCK_VCO);
```

### Syntax

```
#include "slvlib.h"

int
sv_set_genlock (sv_dev, genlock)
SvDev *sv_dev; /* pointer to the device structure */
int genlock; /* source of the clock synchronization signal */
```

## sv\_set\_output\_cmap(sv\_dev, map)

### Description

Sets the color map to use for output to the video monitor or host frame buffer by loading the specified values from the *map* buffer into the S1V output color maps.

The routine begins loading the color map at entry 0 and ends when it has loaded the last specified value. The color map contains 256 entries (0-255). If more than 256 values are specified, the superfluous ones are ignored.

The green color map (`map[1]` in the `colormap_t` structure) goes to the output labeled OUT. The blue color map (`map[2]` in the `colormap_t` structure) goes to the output labeled MON. The red color map (`map[0]` in the `colormap_t` structure) is unused.

**NOTE: This routine does not modify the Brooktree red color map register. Instead, it uses the Sun *pixrect* color map structure to enable you to set the S1V color map without regard to the hardware issues discussed in the section entitled Registers. Use the `sv_set_gain` routine to modify the Brooktree red color map.**

### Arguments

*sv\_dev*            The pointer to the device structure.  
*map*                Pointer to the color map structure.

### Returns

0 if successful, -1 if unsuccessful.

### Example

```
#include "slvlib.h"
#include <pixrect/pixrect.h>
#include <pixrect/pr_io.h>

/* create a color map with two entries: black and white */
colormap_t outmap;
unsigned char bwmap[2] = {0x00, 0xff};

/* both outputs get the same color map */
outmap.map[1] = outmap.map[2] = bwmap;
outmap.length = 2;

sv_set_output_cmap(cv, &outmap);
```

### Syntax

```
#include "slvlib.h"
int
sv_set_output_cmap (sv_dev, map)
SvDev *sv_dev;     /* pointer to the device structure */
colormap_t *map;   /* the S1V color map buffer */
```

## sv\_get\_output\_cmap(sv\_dev, map)

### Description

Determines the color map currently being used for output to the video monitor or host frame buffer by loading the specified values from the S1V output color maps into the *map* buffer.

The routine begins loading the *map* buffer at entry 0 and ends when it has loaded the last specified value. The *map* buffer can contain up to 256 entries (0-255).

The green color map (*map[1]* in the *colormap\_t* structure) goes to the output labeled OUT. The blue color map (*map[2]* in the *colormap\_t* structure) goes to the output labeled MON. The red color map (*map[0]* in the *colormap\_t* structure) is unused.

**NOTE: This routine does not read the Brooktree red color map register. Instead, it uses the Sun *pixrect* color map structure to enable you to read the S1V color map without regard to the hardware issues discussed in the section entitled Registers. Use the *sv\_set\_gain* routine to read the Brooktree red color map.**

### Arguments

<i>sv_dev</i>	The pointer to the device structure.
<i>map</i>	Pointer to the color map structure.

### Returns

0 if successful, -1 if unsuccessful.

### Example

```
colormap_t cmap;
unsigned char out_map[256];
unsigned char mon_map[256];

/* both outputs have the same color map */
cmap.map[0] = NULL; /*red color map is ignored */
cmap.map[1] = out_map;
cmap.map[2] = mon_map;
cmap.length = 2;

sv_get_output_cmap(cv, &cmap);
```

### Syntax

```
#include "slvlib.h"
int
sv_get_output_cmap (sv_dev, cmap)
SvDev *sv_dev; /* pointer to the device structure */
colormap_t *cmap; /* the S1V color map buffer */
```

## **sv\_enable\_continuous(sv\_dev, flag)**

### **Description**

Enables or disables continuous acquisition mode. When continuous acquisition is enabled, the S1V acquires frame after frame of video data. The bank changes automatically to the next selected bank upon vertical synchronization.

When continuous acquisition mode is disabled, the S1V is in single-frame mode, acquiring one frame of video data in the selected bank when acquisition is enabled. Use `sv_grab` to acquire video data.

`sv_grab` always returns the bank into which it copied. However, you can use `sv_set_bank` to set the selected bank, and you can also use `sv_get_bank` to determine the selected bank.

In single-frame mode, you can use the external trigger from input B to acquire video data upon receiving an external stimulus. See `sv_set_trigger` and `sv_get_trigger`.

See `sv_grab` for further information.

### **Arguments**

<i>sv_dev</i>	The pointer to the device structure.
<i>flag</i>	1 = enables continuous acquisition 0 = disables continuous acquisition, placing the device in single-frame mode

### **Returns**

0 if successful, -1 if unsuccessful.

### **Example**

```
sv_enable_continuous(sv_dev, 1);
```

### **Syntax**

```
#include "slvlib.h"

int
sv_enable_continuous(sv_dev, flag)
SvDev *sv_dev; /* pointer to the device structure */
int flag;      /* enables continuous acquisition */
```

## sv\_grab(sv\_dev)

### Description

Returns the bank of the next frame of digitized video data. In single-frame mode, this routine begins acquisition of one frame of video data after the next vertical synchronization. In continuous mode, returns the last complete frame.

### Arguments

*sv\_dev*            The pointer to the device structure

### Returns

0 if copied into bank 0; 1 if copied into bank 1.

### Example

```
int bank = sv_grab(sv_dev);
```

### Syntax

```
#include "slvlib.h"

int
sv_grab (sv_dev)
SvDev *sv_dev; /* pointer to the device structure */
```

## sv\_grab\_mem(sv\_dev, addr)

### Description

Performs *sv\_grab*, copies data to the specified address, and performs *sv\_appdone*.

### Arguments

*sv\_dev*            The pointer to the device structure.  
*addr*                The address in application memory to which to copy the data.

### Returns

0 if successful, -1 if unsuccessful.

### Example

```
sv_grab_mem(sv_dev, buf);
```

### Syntax

```
#include "slvlib.h"

int
sv_grab_mem (sv_dev, addr)
SvDev *sv_dev; /* pointer to the device structure */
caddr_t addr; /* the address to which to copy the data */
```

## **sv\_appdone(sv\_dev)**

### **Description**

Tells the S1V that the application has finished processing the last bank acquired.

### **Arguments**

*sv\_dev*            The pointer to the device structure.

### **Returns**

0 if the event has occurred, -1 if an error has occurred.

### **Example**

```
sv_appdone;
```

### **Syntax**

```
#include "slvlib.h"

int
sv_appdone (sv_dev)
SvDev *sv_dev; /* pointer to the device structure */
```

## **sv\_framecnt(sv\_dev)**

### **Description**

Returns an unsigned integer representing the number of frames that have been seen since the S1V device was opened.

### **Arguments**

*sv\_dev*            The pointer to the device structure.

### **Returns**

The number of frames counted since *sv\_open*, or -1 if unsuccessful.

### **Example**

```
u_int frames = sv_framecnt(sv_dev);
```

### **Syntax**

```
#include "slvlib.h"

u_int
sv_framecnt (sv_dev)
SvDev *sv_dev; /* pointer to the device structure */
```

## sv\_grabcnt(sv\_dev)

### Description

Returns an unsigned integer representing the number of frames that have been acquired since the S1V device was opened.

### Arguments

*sv\_dev*            The pointer to the device structure.

### Returns

The number of frames acquired since *sv\_open*, or -1 if unsuccessful.

### Example

```
u_int grabs = sv_grabcnt(sv_dev);
```

### Syntax

```
#include "slvlib.h"

u_int
sv_grabcnt (sv_dev)
SvDev *sv_dev; /* pointer to the device structure */
```

## sv\_videofb\_checksum(sv\_dev, bank)

### Description

Returns the sum of all bytes in the specified bank.

### Arguments

*sv\_dev*            The pointer to the device structure.  
*bank*              Either SV\_BANK0 or SV\_BANK1.

### Returns

0 if successful, -1 if unsuccessful.

### Example

```
sv_videofb_checksum(sv_dev, SV_BANK0)
```

### Syntax

```
#include "slvlib.h"

int
sv_videofb_checksum (sv_dev, bank)
SvDev *sv_dev; /* pointer to the device structure */
int bank;       /* the bank in which to find the bytes to sum */
```

## sv\_merge(buf, width, height, outbuf)

### Description

Merges a video frame that was acquired in unmerged mode. See `sv_set_merge`, `sv_get_merge`.

### Arguments

<i>buf</i>	The pointer to the unmerged data.
<i>width</i>	The width of the video image, in pixels.
<i>height</i>	The height of the video image, in lines.
<i>outbuf</i>	The pointer to the memory in which to store the merged data.

### Returns

0 if successful, -1 if unsuccessful.

### Example

```
sv_merge(inbuf, 768, 510, outbuf);
```

### Syntax

```
#include "slvlib.h"

sv_merge (buf, width, height, outbuf)
caddr_t buf;      /* pointer to the unmerged data */
int width;        /* width of frame */
int height;       /* height of frame */
caddr_t outbuf;   /* pointer to memory for merged data */
```

## sv\_write\_rasfile(fname, addr, width, height)

### Description

Writes a Sun raster file to the specified address in application memory.

### Arguments

<i>fname</i>	The name of the file to write to.
<i>addr</i>	The address of the file to write to, in application memory.
<i>width</i>	The width of the source image or images, in pixels.
<i>height</i>	The height of the source image or images, in lines.

### Returns

0 if successful, -1 if unsuccessful.

### Example

```
sv_write_rasfile("rasterfile", buf, 640, 478);
```

### Syntax

```
#include "slvlib.h"

sv_write_rasfile (fname, addr, width, height)
char *fname; /* string representing the file name */
caddr_t addr; /* address in which to store the file */
int width;    /* width of image(s) */
int height;   /* height of image(s) */
```

## sv\_read\_rasfile(fname, width, height, red, green, blue)

### Description

Reads and returns a video image from the specified Sun raster file.

### Arguments

<i>fname</i>	The name of the file containing the image or images.
<i>width</i>	Pointer to the width of the source image or images, in pixels.
<i>height</i>	Pointer to the height of the source image or images, in lines.
<i>red</i>	256-byte character array for the red color map.
<i>blue</i>	256-byte character array for the blue color map.
<i>green</i>	256-byte character array for the green color map.

### Returns

A pointer to the address to which the image has been read, or -1 if unsuccessful.

### Example

```
#include <sys/types.h>
char mapR[256], char mapG[256], char mapB[256]
caddr_t pic1 = sv_read_rasfile("rasterfile", &width, &height, mapR, mapG,
mapB);
```

### Syntax

```
#include "slvlib.h"

caddr_t
sv_read_rasfile (fname, width, height, red, green, blue)
char *fname; /* string representing the file name */
caddr_t addr; /* address in which to store the file */
int *width; /* width of image(s) */
int *height; /* height of image(s) */
char *red; /* char array for red color map */
char *green; /* char array for green color map */
char *blue; /* char array for blue color map */
```

## **sv\_b0\_odd(sv\_dev)**

### **Description**

Returns a pointer to the odd field of bank 0.

### **Arguments**

*sv\_dev*            The pointer to the device structure.

### **Returns**

The pointer to the specified field, or -1 if unsuccessful.

### **Example**

```
bank0odd = sv_b0_odd(sv_dev);
```

### **Syntax**

```
#include "s1vlib.h"

caddr_t
sv_b0_odd (sv_dev)
SvDev *sv_dev; /* pointer to the device structure */
```

## **sv\_b0\_even(sv\_dev)**

### **Description**

Returns a pointer to the even field of bank 0.

### **Arguments**

*sv\_dev*            The pointer to the device structure.

### **Returns**

The pointer to the specified field, or -1 if unsuccessful.

### **Example**

```
bank0even = sv_b0_even(sv_dev);
```

### **Syntax**

```
#include "s1vlib.h"

caddr_t
sv_b0_even (sv_dev)
SvDev *sv_dev; /* pointer to the device structure */
```

## sv\_b1\_odd(sv\_dev)

### Description

Returns a pointer to the odd field of bank 1.

### Arguments

*sv\_dev*            The pointer to the device structure.

### Returns

The pointer to the specified field, or -1 if unsuccessful.

### Example

```
bank1odd = sv_b1_odd(sv_dev);
```

### Syntax

```
#include "s1vlib.h"

caddr_t
sv_b1_odd (sv_dev)
SvDev *sv_dev; /* pointer to the device structure */
```

## sv\_b1\_even(sv\_dev)

### Description

Returns a pointer to the even field of bank 1.

### Arguments

*sv\_dev*            The pointer to the device structure.

### Returns

The pointer to the specified field, or -1 if unsuccessful.

### Example

```
bank1even = sv_b1_even(sv_dev);
```

### Syntax

```
#include "s1vlib.h"

caddr_t
sv_b1_even (sv_dev)
SvDev *sv_dev; /* pointer to the device structure */
```

## sv\_videofb\_to\_memory(sv\_dev, bank, memp)

### Description

Transfers an image in the S1V frame buffer to the user-defined portion of host memory. This routine observes the settings set by `sv_set_merge` and `sv_set_interlace`. It is faster than `sv_videofb_to_hostfb`, but assumes that the source and destination have the same number of rows and pixels.

### Arguments

<i>sv_dev</i>	The pointer to the device structure.
<i>bank</i>	Either <code>SV_BANK0</code> or <code>SV_BANK1</code> .
<i>memp</i>	Address of start of destination host memory.

### Returns

0 if successful, -1 if unsuccessful.

### Example

The following example copies an RS-170 640 x 480 image to host memory.

```
u_char image [640*480] /* allocate memory for image */

/* now transfer entire image from S1V bank 0 */
sv_videofb_to_memory(sv_dev, SV_BANK0, image)
```

### Syntax

```
#include "slvlib.h"

int
sv_videofb_to_memory (sv_dev, bank, memp)
SvDev *sv_dev; /* pointer to the device structure */
int bank;      /* take image from bank 0 or bank 1? */
u_char *memp; /* starting address of destination memory */
```

## sv\_videofb\_to\_hostfb(sv\_dev, bank, memp, x, y, rowinc, skip)

### Description

Transfers an image in the S1V frame buffer to the specified portion of the host display, or to any image, such as a Sun pixrect or an X image. The source—the specified bank of the S1V frame buffer—is the size specified by the RS-170 or CCIR format. The *x* and *y* arguments specify the coordinates at which to place the upper left pixel of the image. The coordinates 0,0 specify the upper left corner of the host frame buffer.

The *rowinc* argument specifies the number of pixels between the start of a line and the start of the next line in the destination image (or from column *n* in one line to the same column in the next). For example, for a host frame buffer of 1024 x 768, set *rowinc* to 1024.

The *skip* argument allows you to speed image transfer by skipping *n* lines when reading the image from the specified S1V bank. It skips the same number of lines when writing the image to memory, not changing the previous values of those lines in the image. Because a video image is unlikely to change significantly from frame to frame, this is useful for updating the screen faster.

A value of 0 to *skip* copies each line. A value of 1 to *skip* copies one field the first time it is called, and the other field the next time.

### Arguments

<i>sv_dev</i>	The pointer to the device structure.
<i>bank</i>	Either SV_BANK0 or SV_BANK1.
<i>memp</i>	Address of start of destination host memory.
<i>x</i>	Horizontal coordinate on host frame buffer at which to display the first pixel.
<i>y</i>	Vertical coordinate on host frame buffer at which to display the first pixel.
<i>rowinc</i>	Number of pixels in one row of destination rectangle.
<i>skip</i>	Number of lines to skip to speed the transfer. Must be 0 or 1.

### Returns

0 if successful, -1 if unsuccessful.

### Example

This example copies the S1V frame to the upper left corner of a host display 1024 pixels wide.

```
u_char *image = host_pixrect; /* specify the destination image */

/* now transfer the image from S1V bank 0, skipping every other line */
sv_videofb_to_hostfb(sv_dev, SV_BANK0, image, 0, 0, 1024, 1)
```

### Syntax

```
#include "slvlib.h"

int
sv_videofb_to_hostfb (sv_dev, bank, rowinc, memp, skip)
SvDev *sv_dev; /* pointer to the device structure */
int bank;      /* take image from bank 0 or bank 1? */
u_char *memp; /* starting address of destination memory */
int x;        /* horiz position of 1st image pixel on host display */
int y;        /* vert position of 1st image pixel on host display */
int rowinc;   /* number of pixels per line in destination */
int skip;     /* speed transfer by skipping lines? */
```

## sv\_memory\_to\_videofb(sv\_dev, bank, memp)

### Description

Transfers an image in the host memory to the S1V frame buffer. For example, for the RS-170 format, the source image is assumed to be 640 x 480. This routine observes the settings set by `sv_set_merge` and `sv_set_interlace`. It is faster than `sv_hostfb_to_videofb`, but assumes that the source and destination have the same number of rows and pixels.

### Arguments

<i>sv_dev</i>	The pointer to the device structure.
<i>bank</i>	Either <code>SV_BANK0</code> or <code>SV_BANK1</code> .
<i>memp</i>	Address of start of source host memory.

### Returns

0 if successful, -1 if unsuccessful.

### Example

The following example copies a 640 x 480 image from host memory to bank 1.

```
u_char image [640*480]/* image in host memory */

/* transfer image to S1V bank 1 */
sv_memory_to_videofb(sv_dev, SV_BANK1, image)
```

### Syntax

```
#include "slvlib.h"

int
sv_memory_to_videofb (sv_dev, memp, bank)
SvDev *sv_dev; /* pointer to the device structure */
int bank;      /* take image from bank 0 or bank 1? */
u_char *memp; /* starting address of source (host) memory */
```

## sv\_hostfb\_to\_videofb(sv\_dev, bank, memp, x, y, rowinc)

### Description

Transfers an image to the S1V frame buffer from the specified portion of the host display or other image, such as a Sun pixrect or an X image. The destination—the specified bank of the S1V frame buffer—is the size specified by the RS-170 or CCIR format. The *x* and *y* arguments specify the coordinates at which to begin reading the image. The coordinates 0,0 specify the upper left corner of the host frame buffer.

The *rowinc* argument specifies the number of pixels between the start of a line and the start of the next line in the source image (or from column *n* in one line to the same column in the next). For example, for a host frame buffer of 1024 x 768, set *rowinc* to 1024.

### Arguments

<i>sv_dev</i>	The pointer to the device structure.
<i>bank</i>	Either SV_BANK0 or SV_BANK1.
<i>memp</i>	Address of start of source host memory.
<i>x</i>	Horizontal coordinate on host frame buffer at which the first pixel is displayed.
<i>y</i>	Vertical coordinate on host frame buffer at which the first pixel is displayed.
<i>rowinc</i>	Number of pixels in one row of source rectangle.

### Returns

0 if successful, -1 if unsuccessful.

### Example

This example copies an image from a host display 1024 pixels wide to the S1V frame. The upper left (first) pixel in the image is located at coordinates 48, 20.

```
u_char *image = host_pixrect; /* specify the source image */

/* now transfer the image to S1V bank 0 */
sv_hostfb_to_videofb(sv_dev, SV_BANK0, image, 48, 20, 1024)
```

### Syntax

```
#include "slvlib.h"

int
sv_hostfb_to_videofb (sv_dev, bank, memp, x, y, rowinc)
SvDev *sv_dev; /* pointer to the device structure */
int bank;      /* copy image to bank 0 or bank 1? */
u_char *memp; /* starting address of destination memory */
int x;        /* horiz position of 1st image pixel on host display */
int y;        /* vert position of 1st image pixel on host display */
int rowinc;   /* number of pixels per line in source */
```

## **sv\_partial\_videofb\_to\_memory(sv\_dev, bank, memp, sx, sy, dx, dy)**

### **Description**

Transfers the specified rectangular portion of an image in the S1V frame buffer to the user-defined portion of host memory.

### **Arguments**

<i>sv_dev</i>	The pointer to the device structure.
<i>bank</i>	Either SV_BANK0 or SV_BANK1.
<i>memp</i>	Address of start of destination host memory.
<i>sx</i>	x coordinate of the top left corner of the rectangle to transfer.
<i>sy</i>	y coordinate of the top left corner of the rectangle to transfer.
<i>dx</i>	Number of pixels by which to increment <i>sx</i> .
<i>dy</i>	Number of lines by which to increment <i>sy</i> .

### **Returns**

0 if successful, -1 if unsuccessful.

### **Example**

The following example copies a 40-pixel by 80-line portion of an RS-170 640 x 480 image to host memory.

```
u_char image [40*80] /* allocate memory for subimage */

/* now transfer part of image from S1V bank 0 */
sv_partial_videofb_to_memory(sv_dev, SV_BANK0, image, 0, 0, 40, 80)
```

### **Syntax**

```
#include "slvlib.h"

int
sv_partial_videofb_to_memory (sv_dev, bank, memp, sx, sy, dx, dy)
SvDev *sv_dev; /* pointer to the device structure */
int bank; /* take image from bank 0 or bank 1? */
u_char *memp; /* starting address of destination memory */
int sx, sy; /* starting x and y coordinates for partial image
int dx, dy; /* amounts by which to increment starting x & y */
```

## **sv\_partial\_videofb\_to\_hostfb(sv\_dev, bank, memp, x, y, rowinc, sx, sy, dx, dy)**

### **Description**

Transfers the specified rectangular portion of an image in the S1V frame buffer to the specified portion of the host display, or to any image, such as a Sun pixrect or an X image. The source—the specified bank of the S1V frame buffer—is the size specified by the RS-170 or CCIR format. The *x* and *y* arguments specify the coordinates at which to place the upper left pixel of the image. The coordinates 0,0 specify the upper left corner of the host frame buffer.

The *rowinc* argument specifies the number of pixels between the start of a line and the start of the next line in the destination image (or from column *n* in one line to the same column in the next). For example, for a host frame buffer of 1024 x 768, set *rowinc* to 1024.

The *skip* argument allows you to speed image transfer by skipping *n* lines when reading the image from the specified S1V bank. It skips the same number of lines when writing the image to memory, not changing the previous values of those lines in the image. Because a video image is unlikely to change significantly from frame to frame, this is useful for updating the screen faster.

### Arguments

<i>v_dev</i>	The pointer to the device structure.
<i>bank</i>	Either SV_BANK0 or SV_BANK1.
<i>memp</i>	Address of start of destination host memory.
<i>x</i>	Horizontal coordinate on host frame buffer at which to display the first pixel.
<i>y</i>	Vertical coordinate on host frame buffer at which to display the first pixel.
<i>rowinc</i>	Number of pixels in one row of destination rectangle.
<i>sx</i>	<i>x</i> coordinate of the top left corner of the rectangle to transfer.
<i>sy</i>	<i>y</i> coordinate of the top left corner of the rectangle to transfer.
<i>dx</i>	Number of pixels by which to increment <i>sx</i> .
<i>dy</i>	Number of lines by which to increment <i>sy</i> .

**NOTE: The current release supports only values of 0 (complete image) or 1 (one field) for *skip*.**

### Returns

0 if successful, -1 if unsuccessful.

### Example

This example copies a 100-pixel by 100-line portion of the image starting at 20,30 of the S1V frame to the upper left corner of a host display 1024 pixels wide.

```
u_char *image = host_pixrect; /* specify the destination image */

/* now transfer the subimage from S1V bank 0, skipping every other line */
sv_partial_videofb_to_hostfb(sv_dev,
                             SV_BANK0, image, 0, 0, 1024, 1, 20, 30, 100, 100)
```

### Syntax

```
#include "slvlib.h"

int
sv_partial_videofb_to_hostfb (sv_dev, bank, rowinc, mem, skip)
SvDev *sv_dev; /* pointer to the device structure */
int bank; /* take image from bank 0 or bank 1? */
u_char *mem; /* starting address of destination memory */
int x; /* horiz position of 1st image pixel on host display */
int y; /* vert position of 1st image pixel on host display */
int rowinc; /* number of pixels per line in destination */
int skip; /* speed transfer by skipping lines? */
int sx, sy; /* starting x and y coordinates for partial image
int dx, dy; /* amounts by which to increment starting x & y */
```

## **sv\_partial\_memory\_to\_videofb(sv\_dev, bank, memp, sx, sy, dx, dy)**

### **Description**

Transfers the specified rectangular portion of an image in the host memory to the S1V frame buffer.

### **Arguments**

<i>sv_dev</i>	The pointer to the device structure.
<i>bank</i>	Either SV_BANK0 or SV_BANK1.
<i>memp</i>	Address of start of source host memory.
<i>sx</i>	x coordinate of the top left corner of the rectangle to transfer.
<i>sy</i>	y coordinate of the top left corner of the rectangle to transfer.
<i>dx</i>	Number of pixels by which to increment <i>sx</i> .
<i>dy</i>	Number of lines by which to increment <i>sy</i> .

### **Returns**

0 if successful, -1 if unsuccessful.

### **Example**

The following example copies a 40 x 40 image starting at coordinate 100, 110 from host memory to bank 1.

```
u_char image [40*40] /* image in host memory */

/* transfer subimage to S1V bank 1 */
sv_partial_memory_to_videofb(sv_dev, SV_BANK1, image, 100, 110, 40, 40)
```

### **Syntax**

```
#include "slvlib.h"

int
sv_partial_memory_to_videofb (sv_dev, memp, bank)
SvDev *sv_dev; /* pointer to the device structure */
int bank;      /* take image from bank 0 or bank 1? */
u_char *memp; /* starting address of source (host) memory */
int sx, sy;   /* starting x and y coordinates for partial image
int dx, dy;   /* amounts by which to increment starting x & y */
```

## **sv\_partial\_hostfb\_to\_videofb(sv\_dev, bank, memp, x, y, rowinc, sx, sy, dx, dy)**

### **Description**

Transfers the specified rectangular portion of an image to the S1V frame buffer from the specified portion of the host display or other image, such as a Sun pixrect or an X image. The destination—the specified bank of the S1V frame buffer—is the size specified by the RS-170 or CCIR format. The *x* and *y* arguments specify the coordinates at which to begin reading the image. The coordinates 0,0 specify the upper left corner of the host frame buffer.

The *rowinc* argument specifies the number of pixels between the start of a line and the start of the next line in the source image (or from column *n* in one line to the same column in the next). For example, for a host frame buffer of 1024 x 768, set *rowinc* to 1024.

### Arguments

<i>sv_dev</i>	The pointer to the device structure.
<i>bank</i>	Either SV_BANK0 or SV_BANK1.
<i>memp</i>	Address of start of source host memory.
<i>x</i>	Horizontal coordinate on host frame buffer at which the first pixel is displayed.
<i>y</i>	Vertical coordinate on host frame buffer at which the first pixel is displayed.
<i>rowinc</i>	Number of pixels in one row of source rectangle.
<i>sx</i>	<i>x</i> coordinate of the top left corner of the rectangle to transfer.
<i>sy</i>	<i>y</i> coordinate of the top left corner of the rectangle to transfer.
<i>dx</i>	Number of pixels by which to increment <i>sx</i> .
<i>dy</i>	Number of lines by which to increment <i>sy</i> .

### Returns

0 if successful, -1 if unsuccessful.

### Example

This example copies a 75 X 50 image starting at coordinate 125, 45 from a host display 1024 pixels wide to the S1V frame. The upper left (first) pixel in the image is located at coordinates 48, 20.

```
u_char *image = host_pixrect; /* specify the source image */

/* now transfer the subimage to S1V bank 0 */
sv_partial_hostfb_to_videofb(sv_dev,
                             SV_BANK0, image, 48, 20, 1024, 125, 45, 75, 50)
```

### Syntax

```
#include "s1vlib.h"

int
sv_partial_hostfb_to_videofb (sv_dev, bank, memp, x, y, rowinc)
SvDev *sv_dev; /* pointer to the device structure */
int bank; /* copy image to bank 0 or bank 1? */
u_char *memp; /* starting address of destination memory */
int x; /* horiz position of 1st image pixel on host display */
int y; /* vert position of 1st image pixel on host display */
int rowinc; /* number of pixels per line in source */
int sx, sy; /* starting x and y coordinates for partial image
int dx, dy; /* amounts by which to increment starting x & y */
```

# Registers

Figure 4 describes the S1V registers in detail. Registers labeled (R) are read-only. Those labeled (W) are write-only. Unlabeled registers can be both read and written.

0x002F.FFFF	frame buffer, bank 1			
0x0028.0000	frame buffer, bank 0			
0x0020.0000	not used			
0x0010.0020	cmd(0)	status (R) reset int (W)(1)	cfg(2)	vc(3)
0x0010.000C	not used			
0x0010.0004	cmowm*(4)	cmop*(5)	not used(6)	cmorm*(7)
0x0010.0000	cmrwm*(0)	cmcp* (1)	cmrdm*(2)	cmrrm*(3)
0x0008.0000	not used			
				ROM byte 0x7FFF
	ROM			
0x0000.0000	ROM byte 0			
Byte	0	1	2	3
Word	0		1	

**Figure 4. S1V Memory**

Registers marked with an asterisk (\*) are color map setup registers for the Brooktree Bt478. For details, see the Brooktree documentation listed in the **References** section.

The other four registers are specific to the S1V. They are all 8-bit registers; all eight bits must be read or written simultaneously. Detailed descriptions are provided below.

## Brooktree Color Map Registers

The S1V uses a Brooktree Bt478 digital-to-analog converter to produce the video output signal from the frame buffer and to set the input gain. The Bt478 is designed with three DACs and look-up tables of 256 levels each. Ordinarily, color applications use these three DACs and look-up tables for red, green, and blue color data. However, the S1V supports only grayscale applications. The S1V uses the three color maps as follows:

1. The red color map is used for controlling input gain.
2. The green color map goes to the output labeled OUT.
3. The blue color map goes to the output labeled MON, when it is taking input from the frame buffer.

Because MON and OUT use two distinct color maps, they need not be set up identically. For example, with two monitors side by side, you could compare various methods for highlighting or correcting intensities. By default, entry 0 in these color maps corresponds to 0 (black) in the output. Entry 255 in these color maps corresponds to 255 (white) in the output.

The output of the red DAC is used to set a DC level that controls the input gain of the selected video input. The output of red DAC must not change no matter what input it gets from frame buffer. To set the input gain, all 256 red color map entries and 16 red overlay entries are set to the same value, corresponding to the desired input gain value.

**NOTE: If any of these values differ, different images will yield different input gain levels, producing potentially odd and undesirable results.**

Because the red DAC is eight bits wide, 256 levels of gain are possible. However, such fine control of input gain is normally unnecessary. Therefore, the S1V tool and the `slvlib.h` gain routines use only 16 levels. Use the `sv_set_gain` library routine provided to set the input gain. This routine also serves as an example of how to set the input gain. If you use the library routines `sv_set_gain`, `sv_get_gain`, `sv_set_output_cmap`, and `sv_get_output_cmap`, you need not worry about how the color maps are set up.

The following table is provided to help you determine the proper input gain.

Input Gain Value (Red DAC) (Value in Hexadecimal)	Video Input Voltage Measured from Clamp Level Required for Maximum Output
00	1.15
10	1.10
20	1.05
30	1.00
40	.95
50	.90
60	.85
70	.80
80	.75
90	.70
A0	.65
B0	.60
C0	.55
D0	.50
E0	.45
F0	.40

**Table 1. Input Gain Required to Saturate the A–D Converter**

The Brooktree color map registers are described below.

Address	Name	Description
000 (0)	cmrwm	color map RAM write mode
001 (1)	cmcp	color map RAM
010 (2)	cmrdm	color map read mask
011 (3)	cmrrm	color map RAM read mode
100 (4)	cmowm	color map overlay write mode
101 (5)	cmop	color map overlay
110 (6)	unused	unused
111 (7)	cmorm	color map overlay read mode

**Table 2. Brooktree Color Map Register Definitions**

The following information comes from pages 4–372 and 4–373 of the Brooktree manual listed in **References** on page 58.

## Color Map RAM Data

To write data to the color map, the host loads the color map RAM write mode address register with the address of the color map RAM location to be modified. The host performs three successive write cycles. The S1V uses the first byte (intended for the red color map) to set the input gain. The second byte modifies the OUT output color map. The third byte modifies the MON output color map. After the third write cycle, the three bytes are concatenated into a 24-bit word and written to the location specified by the address register. The address register then automatically increments to point to the next location, which the host can then modify by writing another three-byte sequence. A block of color values in consecutive locations can be written by writing the start address and performing continuous three-byte write cycles until the entire block has been written.

To read color map RAM data, the host loads the color map RAM read mode address register with the address of the color map RAM location to be read. The contents of the color map RAM at the specified location are copied into the color map RAM, and the address register is incremented to point to the next RAM location. The host performs three successive read cycles. The first byte (from the red color map) represents the input gain level. The second byte reads the OUT output color map. The third byte reads the MON output color map. After the third read cycle, the contents of the color map RAM at the address specified by the address register are copied into the color map register and the address register again increments. A block of color values in consecutive locations can be read by writing the start address and performing continuous three-byte read cycles until the entire block has been read.

When accessing the color map RAM, the address register resets to 00 (hex) after reading or writing the third byte to RAM location FF (hex).

Use the `sv_set_output_cmap` library routine provided to set the color map register. This routine also serves as an example of how to set the color map.

## Color Map Overlays

The Bt478 has an overlay feature, which the S1V uses to set the blank level for the MON and OUT outputs, and the input gain level. Set the overlay entries for the green and blue color maps to `SV_BLANK_LEVEL`, as defined in the file `s1v.h`. Set the overlay entries for the red color map to the desired input gain value. This value must be the same as the red color map RAM data values.

To write overlay color data, the host loads the color map overlay write mode address register with the address of the overlay location to be modified. The host performs three successive write cycles. The S1V uses the first byte (intended for the red color map) to set the input gain. The second byte modifies the OUT output color map. The third byte modifies the MON output color map. After the third write cycle, the three bytes are concatenated into a 24-bit word and written to the overlay location specified by the address register. The address register then automatically increments to point to the next location, which the host can then modify by writing another three-byte sequence. A block of color values in consecutive locations can be written by writing the start address and performing continuous three-byte write cycles until the entire block has been written.

To read overlay color data, the host loads the color map overlay read mode address register with the address of the overlay location to be read. The contents of the overlay at the specified location are copied into the color map RAM, and the address register is incremented to point to the next overlay location. The host performs three successive read cycles. The first byte (from the red color map) represents the input gain level. The second byte reads the OUT output color map. The third byte reads the MON output color map. After the third read cycle, the contents of the overlay location at the address specified by the address register are copied into the color map registers and the address register again increments. A block of color values in consecutive locations can be read by writing the start address and performing continuous three-byte read cycles until the entire block has been read.

When accessing the overlay registers, the four most significant bits of the address register are ignored.

## Read Mask

The 8-bit read mask register is logically ANDed with the frame buffer pixel data before the pixel data is used to address the color map. Using this register, individual bits of each pixel can be ignored on the display.

Use the `sv_set_read_mask` library routine provided to set the read mask register. This routine also serves as an example of how to set the read mask.

## Command Register

This 8-bit readable and writable register controls a number of the input and output functions of the S1V.

Bit	S1V_	Description
7	VERT_ENA	A value of one enables an interrupt upon completion of vertical synchronization. A value of zero disables it. This is useful for continuous acquisition, or for switching banks upon acquisition or output.
6	AQDE_ENA	A value of one enables an interrupt when acquisition of both interlaced fields has been completed. A value of zero disables it. This is useful for determining when acquisition of an interlaced image is complete.
5	AQ_ENA	A value of one enables an interrupt when acquisition of the first interlaced field has been completed. A value of zero disables it. This is useful when acquiring an interlaced image. Processing can begin on the first field, even before acquisition of the second field is complete.
4	NINTER	A value of one enables interlacing. A value of zero disables it. If interlacing is disabled, the S1V acquires or writes the same information in both fields.
3	SYNC_ENA	A value of one enables internal synchronization. A value of zero disables it. To view or acquire an image, this bit must be set to one.
2	UNBLANK	A value of one enables the video signal to go to MON or OUT. A value of zero disables video output. To view the image, this bit must be set to one.
1	AQUIRE	A value of one means that the board starts acquiring the frame present at the selected input at the next vertical synchronization (or, if external triggering is being used, at the next vertical synchronization after an external trigger occurs). A value of zero means that the board is not acquiring video data.
0	AQCONT	A value of one means that the board is in continuous acquisition mode. A value of zero means that the board is in single-frame acquisition mode.

**Table 3. Command Register Bit Definitions**

## Status Register

This 8-bit read-only register provides status information. This register shares the same address as the write-only Reset Interrupt Register.

The FIFO referred to in bits 1 and 3 is a buffer that stores video data between the time it is captured and when it is stored in the frame buffer. These bits are principally useful for testing.

Bit	S1V_	Description
7	VINT	Indicates whether a vertical synchronization has occurred since the last time this bit was reset. 1 = true, 0 = false.
6	AQ_DONE	Indicates whether both fields have been completely acquired since the last time this bit was reset. 1 = true, 0 = false.
5	AQ1_DONE	Indicates whether the first field has been completely acquired since the last time this bit was reset. 1 = true, 0 = false.
4	AQ_START	Tells whether acquisition has started. 1 = true, 0 = false. When using external trigger, this information is useful for determining if the trigger has occurred.
3	OFIFO_NHALF	Tells whether the output FIFO is more than half full. 1 = less than half full, 0 = half full or more.
2	SBUS_INT	Tells whether any S1V interrupt is enabled and currently asserted. 1 = true, 0 = false. This allows the <i>sv_intr</i> routine to easily determine if the S1V requires service.
1	IFIFO_NEMPT	Tells whether the input FIFO is empty. 1 = not empty, 0 = empty.
0	AQ_ODD	Tells whether the S1V is currently acquiring or displaying the odd field of an interlaced frame. Useful for determining when an acquisition will start (if a frame is interlaced, the odd field is always acquired first). 1 = true (odd), 0 = false(even).

**Table 4. Status Register Bit Definitions**

## Reset Interrupt Register

This 8-bit write-only register allows you to reset the interrupts enabled by bits 5, 6, and 7 of the command register.

This register shares the same address as the read-only Status Register.

Bit	S1V_	Description
7	VINT	Writing a 1 resets VINT (bit 7 of the status register). Writing a 0 has no effect.
6	AQ_DONE	Writing a 1 resets AQ_DONE (bit 6 of the status register). Writing a 0 has no effect.
5	AQ1_DONE	Writing a 1 resets AQ1_DONE (bit 5 of the status register). Writing a 0 has no effect.
4-0	UNUSED	unused

**Table 5. Reset Interrupt Register Bit Definitions**

## Configuration Registers

This 8-bit readable and writeable register allows you to configure the S1V.

Bit	S1V_	Description
7	CCIR	This read-only bit tells you what kind of S1V board you have. A value of 1 means the board is configured for CCIR. A value of 0 means the board is configured for RS-170.
6	DIG_GNLK	A value of one locks frame buffer and output synchronization to the camera input synchronization using the internal crystal clock—recommended only if the input device has a noisy signal (in the time dimension), as the signal will show some jitter.  A value of zero generates an internal synchronization signal unless bit 1 is set—useful for output only.  <i>NOTE: Do not set both bit 1 and bit 6 to high.</i>
5	EVEN_FIRST	Sets which field of an interlaced frame is acquired first in memory. 0 = odd first, the usual case. 1 = even first. In either case, the first line displayed (as opposed to acquired) is line 0 of the even field.  <i>NOTE: If you ask for the even field to be acquired first, you will get the even field of the first frame and then the odd field of the next.</i>
4	DSBL_BURST	Reserved—must be set to one.
3	XTRG_LVL	Sets edge on which external trigger will trigger. 0 = positive edge (voltage rising), 1 = negative edge (voltage falling).
2	XTRG_ENA	A value of zero disables external trigger; a value of one enables it.
1	VCO_ENA	A value of one locks frame buffer and output synchronization to the camera input synchronization using the voltage-controlled oscillator—recommended if an input device is used.  A value of zero generates an internal synchronization signal unless bit 6 is set—useful for output only.
0	CLAMP_SYNC	Clamp level. 0 = clamp at blank level. 1 = clamp at sync tip. A value of zero is more useful for grayscale video data.

**Table 6. Configuration Register Bit Definitions**

## Video Control Register

This 8-bit readable and writeable register controls video aspects of the S1V.

Bit	S1V_	Description
4-7	UNUSED	Reserved. Must be set to zero for future compatibility.
3	BANK1	Selects from which bank to acquire or display video data after next vertical synchronization. 0 = bank 0, 1 = bank 1.
2	MON_SRC	Determines the source for the MON output—whether monitor displays image from video camera or from S1V frame buffer. 0 = input from S1V frame buffer, 1 = input from video camera.
1	UNUSED	unused
0	INPUT_B	Determines which input is used for acquiring video data. 0 = A input, 1 = B input.

**Table 7. Video Control Register Bit Definitions**

## Interfaces

Interfaces to the SBus and monitor are described below.

### SBus Interface

The S1V connects to the Sun workstation host through the SPARCstation SBus. The SBus is a general-purpose input-output bus for Sun SPARC-based products. Through this bus, the host can access all the S1V registers. The S1V board physically connects to the host using a 96-pin high-density card interconnect-style connector.

For further details on the SBus interface, refer to the Sun *SBus Specification B.0* mentioned in **References** on page 58.

### Monitor Interface

The S1V module supplies the monitor's video inputs with pixel intensity and SYNC information, transmitted through a 75-ohm coaxial cable which physically connects to the monitor with a standard BNC connector. Timing information is provided in the table below.

Parameter	RS-170	CCIR
Lines per frame	525	625
Field rate	60 Hz	50 Hz
Horizontal interval	63.556 s	64 s
Horizontal blanking	10.9 s	10.24–11.52 s
Horizontal sync pulse	4.7 s	4.22–5.76 s
Vertical blanking	1271 s	1280 s
Vertical sync pulse	190.7 s	192 s

**Table 8. Monitor Interface Timing Characteristics**

## Glossary

A–D converter	The mechanism that converts the analog signals from the video camera into the digital signals used by the S1V. Depending upon the strength of the analog signals, they are converted into one of 256 different levels when digitized. The starting point (0) is the clamp level. The ending point (255) is the gain. See <i>clamp</i> , <i>gain</i> .
bank	One of the two regions of the frame buffer able to store one complete video image. See <i>frame buffer</i> .
CCIR	Comité Consultatif International Radio; the international organization responsible for standards set for 50 Hz (PAL) monochrome television signals. See <i>PAL</i> .
clamp	A mechanism that sets the zero level of the A–D converter to a specified point on the incoming video signal. For the S1V, this is either the blank level or the sync level. See <i>A–D converter</i> .
DC	direct current.
even field	A field consisting of all the even-numbered lines on a video monitor. See <i>field</i> , <i>interlacing</i> .
field	Half of the lines of a video image—either all the even-numbered lines or all the odd-numbered lines. Interlaced video images repaint one field at a time. See <i>even field</i> , <i>interlacing</i> , <i>odd field</i> .
field rate	The number of times per second (expressed in Hz) that one field of a video image is repainted on the display. Compare <i>refresh rate</i> .
frame	One complete video image. For interlaced video images, a frame equals two fields.
frame rate	See <i>refresh rate</i> .
frame buffer	A region of memory and associated registers representing pixels in a video image. The S1V frame buffer can store two frames (two video images), one in each of its two banks. See <i>bank</i> , <i>frame</i> .
gain	The amount by which differences in the input signal are exaggerated or minimized.
grayscale	A system of encoding intensities such that pixels appear in black, dark gray, then increasingly lighter shades of gray, and finally white.
HSYNC	Horizontal synchronization.
interlacing	A technique to eliminate flicker by doubling the apparent refresh rate on a video monitor. A video image is composed of lines on the monitor—525 lines for NTSC images and 625 lines for PAL images. First, all the even-numbered lines are repainted; then, all the odd-numbered lines are repainted. See <i>field</i> , <i>NTSC</i> , <i>refresh rate</i> , <i>PAL</i> .
monochrome	An image without color information, but only differing intensities. See <i>grayscale</i> .
NTSC	National Television System Committee; the standards organization responsible for determining the characteristics of 60 Hz video signals, used in North America and Japan.
odd field	A field consisting of all the odd-numbered lines on a video monitor. See <i>field</i> , <i>interlacing</i> .
PAL	Phase Alternation Line; the standard for determining the characteristics of 50 Hz video signals, used in northern Europe and much of Asia.
pixel	The smallest independently addressable unit on the display.

refresh rate	The number of times per second (expressed in Hz) that a video image is repainted on the display. For RS-170 signals, this is 60 Hz. For CCIR signals, this is 50 Hz. Also called <i>frame rate</i> . Compare <i>field rate</i> .
RS-170	The international standard for 60 Hz (NTSC) monochrome television signals. See <i>NTSC</i> .
VCO	Voltage-controlled oscillator, a variable oscillator used to lock the internal synchronization signal to the synchronization signal coming from the video camera.
VSYNC	Vertical synchronization.

## Specifications

The S1V conforms to the following specifications.

### Video Interface

RS-170	525 lines video input at 60 Hz 640 x 480 digitized data output
CCIR	625 lines video input at 50 Hz 768 x 512 digitized data output
Data memory	1 MB 256 gray levels (8 bits per pixel) NTSC: (640 x 480 x 8) x 2 PAL: (768 x 512 x 8) x 2

### Software

Drivers for Sun OS Version 4.1 and System V Version 4 (Solaris 2.0)  
10 registers  
32-bit read and write access modes  
Random access to video data

### SBus Compliance

Master/slave	Slave only
Transfer size	Byte and word
Clock rate	25 MHz maximum
Interrupt	Level 5
Connector	Fujitsu FCN-2341PO96-GO or Honda PCS-96MD

### Power

5 V at 1 A

### Environmental

Temperature	Operating: 10 to 40 C Nonoperating: -20 to 60 C
Humidity	Operating: 20 to 80% noncondensing at 40 C Nonoperating: 95% noncondensing at 40 C

### Physical

Occupies one standard SBus slot	
Dimensions	3.3" x 5.87" x 0.5"
Weight	6 oz.
Connectors	4 standard BNC connectors 2 video input, display and monitor output

## References

The following additional documentation may prove helpful.

Brooktree Bt478 product specification, in *Brooktree Graphics and Imaging Product Databook*, pp. 4-371 to 4-394. Available from Brooktree Corporation, (800) VIDEO IC.

Sun *SBus Specification B.0*, part number 800-5323-05, available from Sun Microsystems, Inc., (415) 960-1300.

EIA Standard EIA-170-A (Revision of RS-170 standard), *Electrical Performance Standards Monochrome Television Studio Facilities*, Electronic Industries Association, November, 1957. Available from the Electronic Industries Association, (202) 457-4900. Also available from Global Engineering Documents, (800) 854-7179.

*Recommendations of the CCIR, 1990*, Vol. XI-Part 1, *Broadcasting Service (Television)*. See especially Recommendation 470-2, *Television Systems*. Available from the International Telecommunications Union International Radio Consultative Committee. Also available from Global Engineering Documents, (800) 854-7179.

*Recommendations of the CCIR, 1990*, Annex to Vol. XI-Part 1, *Broadcasting Service (Television)*. See especially Report 624-4, *Characteristics of Television Systems*. Available from the International Telecommunications Union International Radio Consultative Committee. Also available from Global Engineering Documents, (800) 854-7179.

## Free Imaging Software

The following imaging software may prove helpful. It is available free from the sources specified.

- ALV** This Sun-specific image toolkit is available through e-mail from:  
alv-users-request@cs.bris.ac.uk
- Fuzzy Pixmap** This software for converting image formats and manipulating images is available through anonymous ftp from: nl.cs.cmu.edu in /usr/mlm/ftp. It consists of three files: fb.tar.Z, utah.tar.Z, and tiff.tar.Z.
- IMG Software Set**  
This software was developed by Paul Raveling (raveling@unify.com). It writes specific image formats that display under X windows and can be manipulated. It is available through anonymous ftp from: venera.isi.edu in /pub/img\*
- TIFF Library** This portable library for reading and writing TIFF files (as well as other tools) is available through anonymous ftp from: sgi.com in /graphics/tiff/v3.0.tar.Z or from: ucbvax.berkeley.edu in /pub/tiff/v3.0.tar.Z  
A companion file, v3.0pics.tar.Z contains sample TIFF image files.
- xloadimage** This program reads images in a wide variety of formats and displays them under X windows. Version 3.01 is available through anonymous ftp from: export.lcs.mit.edu in /contrib/xloadimage.3.01.tar.Z  
It was developed by Jim Frost (jimf@centerline.com). Those wishing Version 3.02 fixes can e-mail him a request.
- xv** This program reads images in a wide variety of formats and displays them under X windows. As of this writing, version 3.10a is available through anonymous ftp from: ftp.cis.upenn.edu in /pub/xv/vx-3.10a.tar.Z.

# Index

## A

accessing video data 8  
 acquiring  
   continuously 7  
   series of frames 7  
   single frame 7  
   single pixels 9  
   small rectangles 9  
 acquisition 6, 23, 24, 25, 26, 28, 33  
   completion 33  
   counting 35  
   enabling interrupt upon completion 53  
   merged 27  
   started 54  
 acquisition mode  
   continuous 1, 14  
   default 18  
   determining 53  
   enabling continuous 32  
   enabling single-shot 32  
   single-shot 1, 14  
 ALV software 60  
 application status, communicating to S1V 34  
 AQ\_DONE 54  
 AQ\_ENA 53  
 AQ\_ODD 54  
 AQ\_START 54  
 AQ1\_DONE 54  
 AQCONT 53  
 AQDE\_ENA 53  
 ACQUIRE 53  
 A-to-D converter  
   defined 57

## B

bank 1, 15  
   default 18  
   defined 57  
   selected 9, 26  
   selected with continuous acquisition 32  
   selecting 25, 56  
   switching 27, 53  
   use with continuous acquisition 14  
 BANK1 56  
 black level 50  
 blank level 52, 55  
 blue color map 30, 31, 50  
   reading 52  
   writing 52

blue color map overlay  
   reading 52  
   writing 52  
 bytes in bank, summing 35

## C

cable 2, 56  
 capture 1  
 CCIR 1, 8, 15, 19, 21, 22, 55  
   defined 57  
   height of image 19  
   specifications 59  
   width of image 20  
 checksum 35  
 clamp  
   defined 57  
   level 51, 55  
 CLAMP\_SYNC 55  
 clock rate  
   specifications 59  
 closing S1V device 6, 18  
 color map  
   overlay 52  
   RAM 52  
 color map registers  
   defined 50  
 command register 53  
 configuration register 55  
 configuring the driver 5, 6  
 connector 2, 56  
   specifications 59  
 continuous acquisition 6, 7  
 continuous mode 6  
 copying data to application memory 33  
 counting  
   acquisitions 35  
   frames 34

## D

data transfer size  
   specifications 59  
 DC  
   defined 57  
 default values 18  
   gain 15, 21, 22  
   input 2  
   output 2  
 device  
   closing 6, 18

- locking 6, 17
- opening 6, 17
- resetting 6
- unlocking 18
- device driver 1, 3, 4
  - executable file 4
  - header file 4
  - library header file 4
  - library routines 16
    - executable 4
    - source 4
  - multiple 3, 17
  - README file 4
  - utility file
    - executable 4
    - source 4
- diagnostic program 5, 9
- digital-to-analog converter 50
- displaying an image from a file 37
- displaying video data 9
- distribution diskette 3, 4
- DSBL\_BURST 55

**E**

- EINVAL error 7
- enabling
  - interlacing 53
  - internal synchronization 53
  - interrupt 53
  - output 53
  - trigger 23, 55
- environmental
  - specifications 59
- even field
  - defined 57
- EVEN\_FIRST 55
- example program 5, 9
  - acquiring sequential frames 10
  - configuring the S1V 12
  - counting frames 11
  - displaying a file 11
  - displaying a gray scale 11
  - displaying the host frame buffer 11
  - displaying the S1V frame buffer 10
  - reading from a file 11
  - reading from the host frame buffer 11
  - reading the S1V frame buffer 10
  - using read() system call 10
  - writing to a file 10

**F**

- field 8, 9
  - both acquired 54
  - defined 57
  - determining which is being acquired 54
  - enabling interrupt upon acquiring 53
  - even, returning pointer to 38, 39
  - merging 26, 27
  - odd, returning pointer to 38, 39
  - one acquired 54
  - setting which to acquire first 55
- field rate
  - defined 57
- FIFO 54
- file offset error 7
- frame
  - counting 34
  - defined 57
- frame buffer 15
  - defined 57
  - host 9, 30, 31, 41, 43, 45, 47
  - S1V 8, 9, 22, 23, 24, 25, 26, 40, 41, 42, 43, 44, 45, 46, 47, 54
- frame rate
  - defined 57
- framecount 5, 9, 11
- fuzzy pixmap manipulation software 60

**G**

- gain 15, 21, 22, 50, 52
  - defined 57
  - determining proper 51
- grabbing partial frames 6
- gray levels 1, 13, 50
- grayscale 5, 9, 11
  - defined 57
- green color map 30, 31, 50
  - reading 52
  - writing 52
- green color map overlay
  - reading 52
  - writing 52

**H**

- HSYNC
  - defined 57
- humidity
  - specifications 59

**I**

IFIFO\_NEMPT 54  
 IFIFO\_NHALF 54  
 imaging software 60  
 IMG software 60  
 input 15  
     A 1, 2  
     B 1, 23, 24, 32  
     default port 18  
     determining gain 22  
     gain 15, 50, 52  
         determining proper 51  
     selected port 20, 21  
     selecting 56  
     setting gain 21  
     signal 15  
     viewing 24, 25, 55  
 INPUT\_B 56  
 installing  
     installation command 3  
     S1V board 2  
     S1V driver 3  
     S1V tool 3  
 intensity correction 50  
 interfaces 56  
     monitor 56  
     SBus 56  
 interlacing 6, 8, 27, 28, 41, 53, 55  
     defined 57  
     enabling 53  
 interrupt  
     determining if any are pending 54  
     level 59  
     resetting 54  
     resetting acquisition complete 54  
     resetting field acquisition complete 54  
     resetting vertical 54  
     vertical synchronization 53

**J**

jitter 29, 55

**L**

locking S1V device 6, 17

**M**

makefile 3, 4  
 memory  
     host 40, 42, 44, 46  
     S1V 1, 8

    specifications 59  
 merged  
     determining status 27  
 merging  
     fields 6, 26  
     frame acquired unmerged 36  
 MON\_SRC 56  
 monitor  
     interface 56  
     timing 56  
 monochrome  
     defined 57  
 mread 5, 9, 10

**N**

network 13  
 NINTER 53  
 noninterlaced 27  
 NTSC 1, 59  
     defined 57

**O**

odd field  
     defined 57  
 OFIFO\_NHALF 54  
 opening S1V device 6, 17  
 output 15, 24, 25  
     blue color map 30, 31  
     default 18  
     enabling 53  
     green color map 30, 31  
     MON 1, 2, 50, 52, 56  
     OUT 1, 2, 50, 52  
     palette 30, 31  
     producing 50  
     red color map 30, 31  
 overlay 52

**P**

PAL 1, 59  
     defined 57  
 palette 22, 23  
     default 18  
     grayscale 13  
     host display 13  
     output 30, 31  
 physical  
     specifications 59  
 pixel 1, 23  
     defined 57

- masking bits of 22, 53
- pixrect 41, 43, 45, 47
- pkgadd command 4
- pointer to even field 38, 39
- pointer to odd field 38, 39
- power
  - specifications 59

## R

RAM

- application 9
- host 40, 42, 44, 46
- S1V 1, 8
- specifications 59

read mask 22, 23, 53

read() system call 7

reading

- from file 37

red color map 30, 31, 50

references 60

- Brooktree specification 60
- CCIR specification 60
- RS-170 specification 60
- SBus specification 60

refresh rate

- defined 57

registers 49

- color map 30, 31, 50
- command 53
- configuration 55
- header file 5
- read mask 53
- reset interrupt 54
- S1V 53
- status 53
- video control 56

reset interrupt register 54

resetting device 6

resetting interrupt 54

ROM 56

RS-170 1, 8, 15, 19, 21, 22, 55

- defined 58
- height of image 19
- specifications 59
- width of image 20

## S

S1V data structure 17

S1V tool 4

- appearance 14
- invoking 13

- quitting 14
  - using 13
- s1vconfig 5, 9, 12
- s1vread 5, 9, 10
- s1vtest 5, 9
- s1vwrite 5, 9, 11
- SBus 2
  - connector 59
  - interface 56
  - specifications 59
- SBus video library 4, 16
- screen updating, fast 41
- setdebug 5
- single-shot acquisition 7
- size
  - specifications 59
- software
  - specifications 59
- specifications 59
  - Brooktree 60
  - CCIR 59, 60
  - clock rate 59
  - connector 59
  - data transfer size 59
  - environmental 59
  - humidity 59
  - interrupt level 59
  - memory 59
  - physical 59
  - power 59
  - RAM 59
  - RS-170 59, 60
  - SBus 59, 60
  - size 59
  - software 59
  - temperature 59
  - video interface 59
  - weight 59
- status 54
  - signaling to S1V 34
- Sun raster file 1, 14, 36, 37
  - saving 14
  - viewing 14
- Sun SPARCstation 1, 2
- SunOS 1, 6
  - Solaris 2.0 4, 59
  - SVR 4 4, 59
  - Version 4.1 3, 59
- sv\_appdone 33, 34
- sv\_b0\_even 9, 38
- sv\_b0\_odd 9, 38
- sv\_b1\_even 9, 39

sv\_b1\_odd 9, 39  
 sv\_close 18  
 sv\_enable\_continuous 32  
 sv\_format\_height 19  
 sv\_format\_width 20  
 sv\_framecnt 7, 34  
 sv\_get\_bank 26, 32  
 sv\_get\_gain 22, 50  
 sv\_get\_genlock 28  
 sv\_get\_interlace 28  
 sv\_get\_merge 27  
 sv\_get\_mon\_out 25  
 sv\_get\_output\_cmap 31, 50  
 sv\_get\_port 21  
 sv\_get\_read\_mask 23  
 sv\_get\_trigger 24, 32  
 sv\_grab 23, 32, 33  
 sv\_grab\_mem 33  
 sv\_grabcnt 7, 35  
 sv\_hostfb\_to\_videofb 43  
 sv\_memory\_to\_videofb 9, 42  
 sv\_merge 36  
 sv\_open 6, 17  
 sv\_partial\_hostfb\_to\_videofb 47  
 sv\_partial\_memory\_to\_videofb 46  
 sv\_partial\_videofb\_to\_hostfb 45  
 sv\_partial\_videofb\_to\_memory 44  
 sv\_set\_bank 7, 9, 25, 32  
 sv\_set\_defaults 18  
 sv\_set\_gain 21, 50  
 sv\_set\_genlock 29  
 sv\_set\_interlace 6, 27  
 sv\_set\_merge 26  
 sv\_set\_mon\_out 24  
 sv\_set\_output\_cmap 30, 50  
 sv\_set\_port 20  
 sv\_set\_read\_mask 22, 53  
 sv\_set\_trigger 23, 32  
 sv\_video\_format 19  
 sv\_videofb\_checksum 35  
 sv\_videofb\_to\_hostfb 9, 41  
 sv\_videofb\_to\_memory 9, 40  
 sv\_write\_rasfile 36  
 SYNC\_ENA 53  
 synchronization  
   enabling interrupt upon completion of vertical 53  
   internal 53, 55  
   locking 55  
   signal 29  
   vertical 33, 53, 54, 56  
 synchronization signal 28

**T**

tar command 3  
 temperature  
   specifications 59  
 test program 5, 9  
 testing 54  
 TIFF images 60  
 TIFF library software 60  
 transferring data  
   as image 41, 43, 45, 47  
   fast 40, 42  
   partial 44, 46  
 trigger 1, 15, 53  
   enabling 23, 55  
   setting edge 23, 55

**U**

UNBLANK 53  
 unlocking S1V device 18

**V**

VCO  
   defined 58  
 VCO\_ENA 55  
 VERT\_ENA 53  
 video camera 2, 15  
 video control register 56  
 video format 19, 55  
   height 19  
   width 20  
 VIDEO IN connector 2  
 video interface  
   specifications 59  
 video monitor 2, 9, 15, 30, 31  
 VINT 54  
 VSYNC  
   defined 58

**W**

weight  
   specifications 59  
 white level 50  
 writing  
   from file to memory 36

**X**

X image 41, 43, 45, 47  
 X window server 13  
 xloadimage software 60

XTRG\_ENA 55  
XTRG\_LVL 55  
xv software 60  
xwin 5, 9