SgVME-PCI Adaptor User's Manual and Installation Guide

olflower
Computer, Inc.

Solflower Computer, Inc
3511 Thomas Road, Ste-2
Santa Clara, CA 95054
(408) 982-8680, Fax (408) 982-8685
http://www.solflower.com

Credit and Trademarks

This equipment generates, uses and can radiate radio frequency energy and if not installed and used in accordance with the Instruction Manual, may cause interference in radio communications.

Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

This document presents information for users of Solflower Computer, Inc.'s Gateway Series.

CHAPTER 1 Overview

1.1 INTRODUCTION

The SgVME-PCI Adaptor provides PCI Bridge and PCI-VME bus functions for a wide range of I/O applications. It can be used as ordinary legacy PCI bridges when the transparent Bridge function is enabled, or they can be used as a Fabric Gateway for advanced features such as multicast or applications used in distributed-computing. Connecting to your I/O peripherals only requires inexpensive CAT5e cables.

In transparent PCI-PCI bridge mode, the existing software can run without changing in drivers and configuration code. All devices in the system are visible and accessible through their unique address ranges. SgVME-PCI Adaptor also provides Bridging function from PCI to VME bus, making expansion from PCI to VME in a seamless fashion.

The Solflower driver supports Solaris 8, 9, and 10, and provides transparent bridging from PCI to VME.

1.2 Features and benefits:

- Can be used to construct low cost large Legacy PCI expansion system
- Compatible to 3.3V and 5V, PCI clock speed 33/66MHz
- Connection is made by inexpensive CAT5e cables which can run up to 40 ft.
- Support Solaris 8, 9, 10, SPARC, SPARC x86 IA
- Support standard PCI address routing, path routing, multicast routing
- Available in multiple board formats, PCI, CompactPCI, VME and PMC
- Easy interface to VMEbus or custom bus for great flexibility
- Fully compliant with Local PCI Bus Specification Rev. 2.2
- Data rate up to 2.5 Gbps
- Integrates with SUN PCI software system, resulting a transparent environment for PCI, cPCI
- Dynamic add/remove software instance of devices
- SUN 4 VME driver compatible

1.3 APPLICABLE DOCUMENTS

- VMEbus Specification Manual, Revision D IEEE STD 1014-9187
- Star-VME-2010PCI VME64x on CompactPCI PICMG 2.2 R1.0
- StarGen SG2010 Hardware Reference Manual

1.4 GENERAL DESCRIPTION

SgVME-PCI board set contains two boards, SOLFpvme driver CD media and 2 x CAT5e cables, and a user manual (this manual). These are the generic names:

- SgPCI-2 (PCI Host adapter board with 2 channels)
- SgVME (PCI VME adapter board)
- Optional SgPCI-LS (for PCI expansion subsystem)
- Solflower Solaris 2.x VME nexus driver
- 15 Ft. CAT5e cable
- SgVME-PCI Adaptor User's Manual and Installation Guide

These boards utilize SG2010 chipsets. SG2010 is capable of 66 MHz, 64-bit PCI bus operation, and provides two (2) 2.5 Gbps full duplex links using two (2) pairs of standard RJ45 connectors. Two links can be bundled to create a 5 Gbps point-to-point link. Links are connected using standard CAT5e cables which can run up to 40 feet.

1.5 Selecting work area

The board set is electronic equipment and must be handled with care. Extremes in temperature and humidity may affect the functionality and performance of the whole system.

1.6 Unpacking instructions

Perform a visual inspection of the shipping carton for any handling damage. If any shipping carton is severely damaged, request that the carrier's agent be present when the carton is opened. Carefully remove the contents and ensure that all the pieces are present. The printed circuit boards are wrapped in electrostatic-safe bags. We recommend that you save the shipping carton and the packing material for future use, in case the product must be reshipped or returned to Solflower.

1.6.1 Identifying the system components

Check the packing slip to make sure that the shipment is complete.

CHAPTER 2 SgVME-PCI Adaptor Specification

2.1 SgPCI Series Description

Based on the StarGen SG2010 chipset, the SgVME-PCI Adaptor is available in two major product categories:

- SgPCI-n: Solflower-gateway-PCI-n card install into any standard PCI workstation (such as SunBlade 1500/2500, Sun CP2140 or any x86 IA system). SgPCI-n supports: 2, 4, 5, or 8, StarFabric ports/channels.
- SgPCI-LS: Solflower-gateway PCI Leaf interface with SYSCON enabled. This card install into a system slot of a PCI expansion box (e.g PCI expansion sub-system)
- SgVME: Solflower-gateway PCI-VME interface board which provides transparent PCI to VME bridging function.

2.1.1 SgPCI-2 Board

The SgPCI-2 board utilizes StarGen SG2010 multi-function device. This device provides PCI to PCI bridge function (PCI-PCI) and Gateway function. SgPCI-2 is capable of 66 MHz, 64-bit PCI bus operation, 3.3 and 5V tolerant, and provides two (2) 2.5 Gbps full duplex link ports. Ports are connected using standard CAT5e cables, which can run up to 40 feet. The Gateway functions can be programmed for up to two paths for expansion to two separated sub-systems. Figure 1 shows a SgPCI-2 board. Figure 2 shows SgPCI-2 board block diagram.

FIGURE 1 SgPCI-2 Board

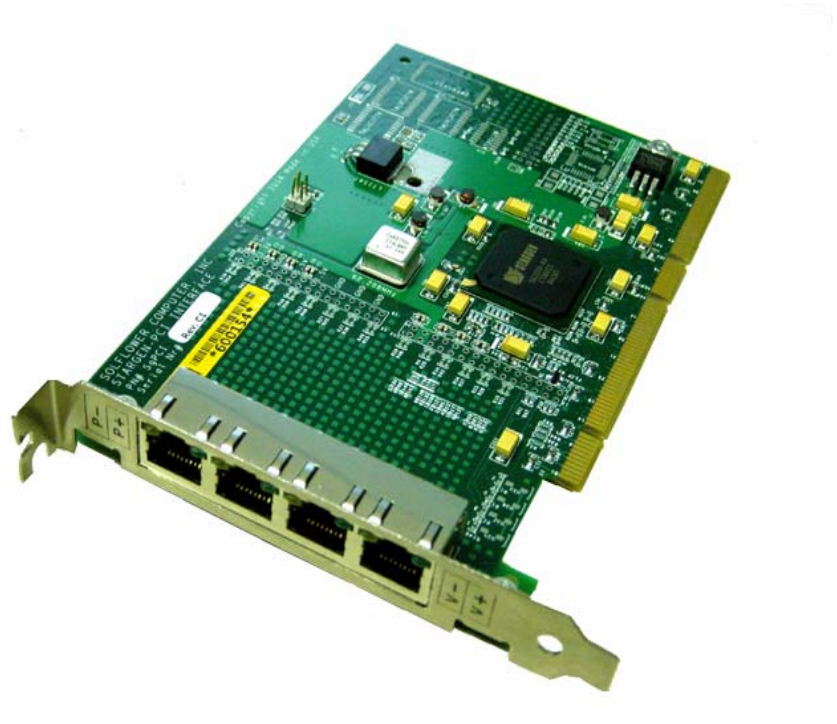
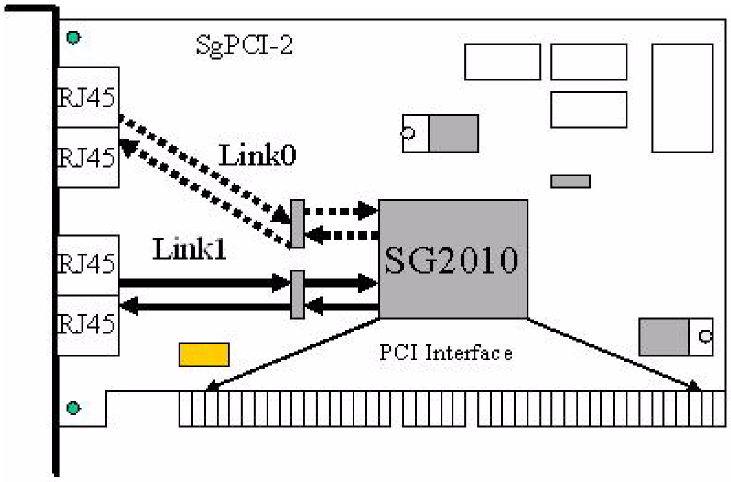
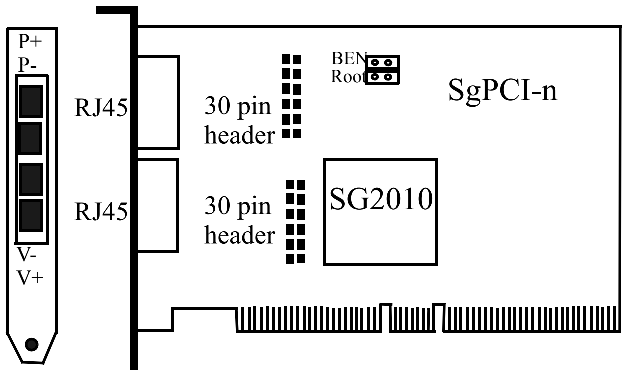


FIGURE 2 **SgPCI-2 block diagram**



The 64/32bit 66/33 MHz PCI interface is compatible with PCI Local Bus Specification 2.2. Ports 0 and 1 operate at 2.5 Gbps full duplex each link. Figure 3 briefly shows a SgPCI-2 board with two pairs of RJ45s at the front panel. The front panel label showing here is for two channels. One channel (link) go to PCI expansion (P+/P-) and the other channel go to VME (V+/V-). CAT5e cables are labeled with P+/P- or VME+/VME-. Simply connect the cables to the RJ45 connectors according to the labels. (i.e.,. VME+ of CAT5e cable go to VME+ RJ45 connector, etc.)

FIGURE 3 SgPCI-2 Board



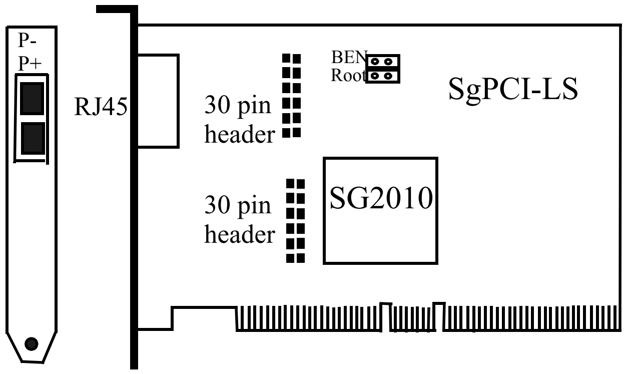
2.1.1.1 SgPCI-2 Electrical Characteristics

Power consumption:	10 Watts
Input power:	5VDC
Power consumptionmax.	5Volt@2000 mA. typical
PCI bus interface:	3.3 and 5V tolerant

2.1.2 SgPCI-LS Board (optional for PCI expansion)

The SgPCI-LS boards provide physical conversion from Fabric signals back to PCI bus signals, and allow the entire system to act like a single unit to the system management. SgPCI-LS is always installed in the system slot of a secondary PCIbus, and re-generates all required signals, including PCI clocks and Request/Grant signals. The Solflower PCI expansion system provides four addition 32/64-bit PCI slots and can be placed up to 40 feet away from the host system. Figure 4 illustrates a SgPCI-LS board.

FIGURE 4 **SgPCI-LS Board**



2.1.2.1 SgPCI-LS Electrical Characteristics

Power consumption:	10 Watts
Input power:	5VDC
Power consumptionmax.	5Volt@2000 mA. typical
PCI bus interface:	3.3 and 5V tolerant

2.2 SgPCI-to-VME Bus Bridge

2.2.1 SgVME board

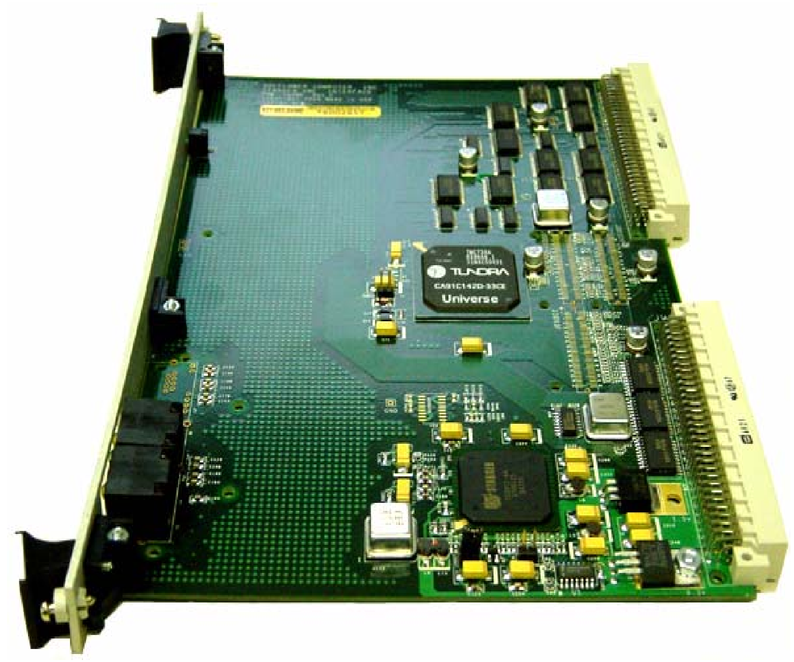
The Solflower SgVME 6U board provides VME64 expansion. It is an 6U VME form factor and installed in the system slot of a standard VME subsystem. The SgVME can drive up to 21 VME slots and supports VME revD specification. The on-board Tundra Universe II chipset provides bridging between PCI and VME bus. Some features of the Tundra Universe II VME-to-PCI bridge include:

- Fully support VME64 Rev D, board requires only 5VDC.
- Block mode data transfer up to 70MB/sec
- Support Posted Writes through 128B FIFO for quick bus release
- Built-in DMA engine for bidirectional PCI-VME and VME-to-PCI transfers
- All VME options such as Request Modes, Bus time-out is software programmable
- Up to seven window-mappings 32MB each for cover larger VME address spaces
- 32 bit or 64 bit, 33 MHz PCI bus interface; Programmable DMA controller;
- Sustained transfer rates up to 70 Mbytes per second;
- Automatic initialization for slave only application;
- Full VMEbus system controller functionality;
- Supports the latest generation of VME application.

Solflower Computer Inc. also provides PCI-VME bridge driver: Solaris 8, 9, 10 SOLFpvme. SOLFpvme allows Sun-4 device drivers to be used with PCI-based Sun workstation and X86 IA system without changing the code in VME applications

Figure 5 shows a SgVME board.

FIGURE 5 **SgVME board**



2.2.1.1 SgVME Electrical Characteristics

Power consumption:	15 Watts
Input power:	5VDC
Power consumptionmax.	5Volt@3000 mA. typical
PCI bus interface:	3.3 and 5V tolerant

2.3 SgPCI-2, SgVME boards Miscellaneous

2.3.1 Operating Voltages (all internally regulated from 5 Volt source)

1.5 Volt

3.3 Volt

5 Volt

2.3.2 RJ45 connectors and LEDs (Figure 6)

There are two pairs of RJ45 connectors on both SgPCI-2 PCI board. Each pair of RJ45s make up one link connects to the outside world. At the top corners of a RJ45 there are two LEDs as shown in Figure6. Each pair of RJ45s has a total of four LEDs indicating four LVDS receiving pairs of that link. These LEDs indicate the link status of that link. Table 1 describes LED's status.

FIGURE 6 **RJ45 and LEDs**

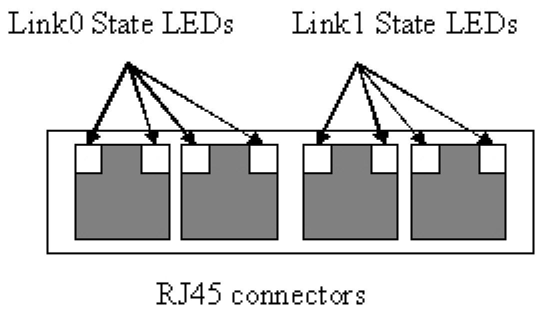


TABLE 1. **LED display**

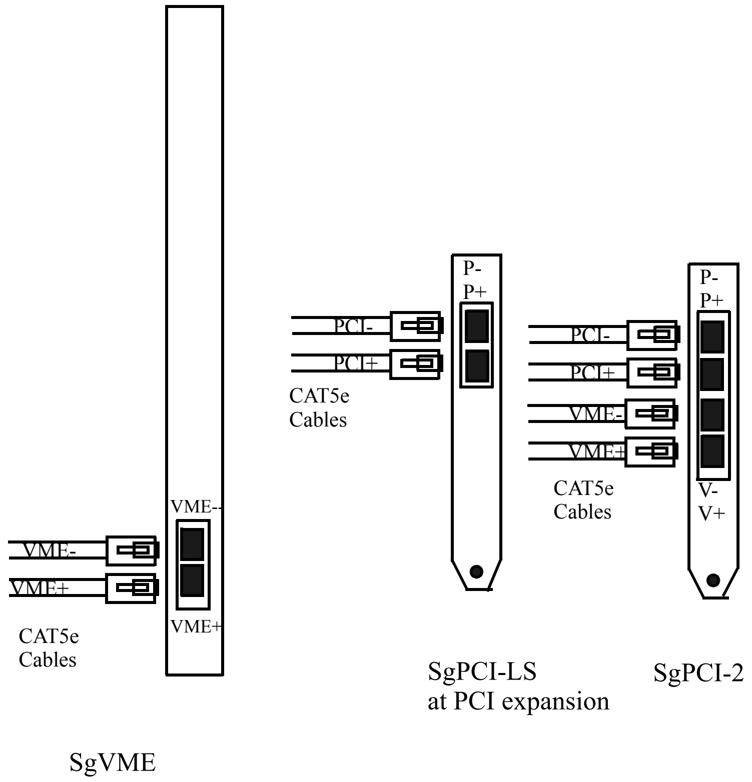
LED States	Description
LED are off	Link is unsynchronized and traffic is disabled
LED are on	Link is synchronized and traffic is enabled
LED are flashing	Link is synchronized but traffic is disabled

During normal operation, LEDs start blinking if CAT5e cable was disconnected or VME power sub-system power was down. This indicates the whole system must be re initialized (or re-boot) to bring up VME hardware. Access to VME address spaces while LEDs blinking may lead to hanging or panicking the system.

2.4 Cable Connection

There are four (4) RJ45 Jacks mounted on the front panel of the SgPCI-2 boards, one pair for each link. Be sure to connect the CAT5e cables according to labels. Please refer to Fig 7. After reset or recycle power to the host, LEDs should be ON indicating the system is ready. Always power on first the leaf system where SgVME or SgPCI-LS was installed BEFORE your host system.

FIGURE 7 **RJ45 connectors at the front panels**



2.5 Sample applications

2.5.1 SgPCI-2, SgVME, and SgPCI-LS Application

The following figure (Figure8) shows a typical application of SgPCI-2, SgVME, and SgPCI-LS. Figure 9, 10,11 are details of cable connections of the three interfaces.

FIGURE 8 **SgPCI-2 and SgVME+SgPCI-LS application**

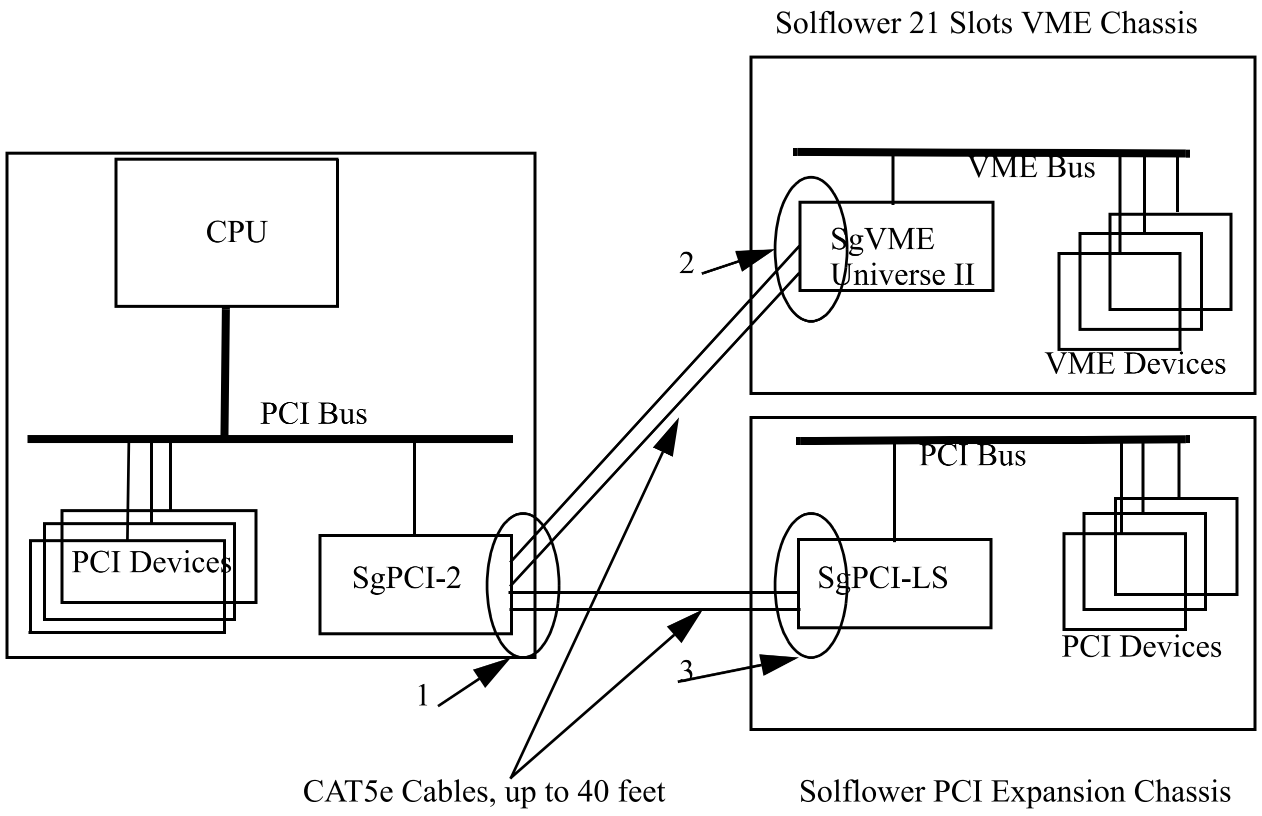


FIGURE 9 **Circle 1: CAT5e cable connections with labels at SgPCI-2**

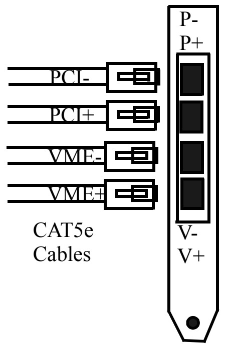


FIGURE 10 **Circle 2: CAT5e Cable connections at SgVME with labels**

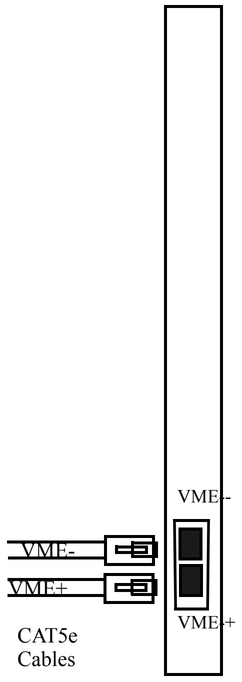
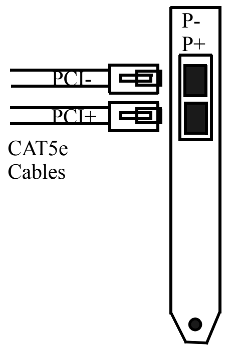


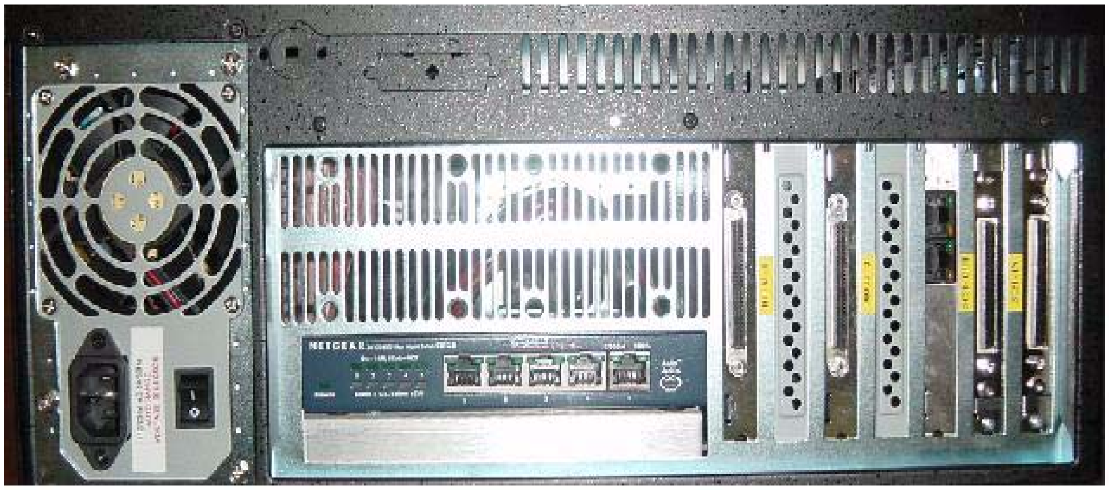
FIGURE 11 **Circle 3: CAT5e Cable connections at SgPCI-LS with labels**



This is a typical application of the SgPCI and SgVME boards using Solflower's PCI Expansion Chassis and 21 Slots VME Chassis.

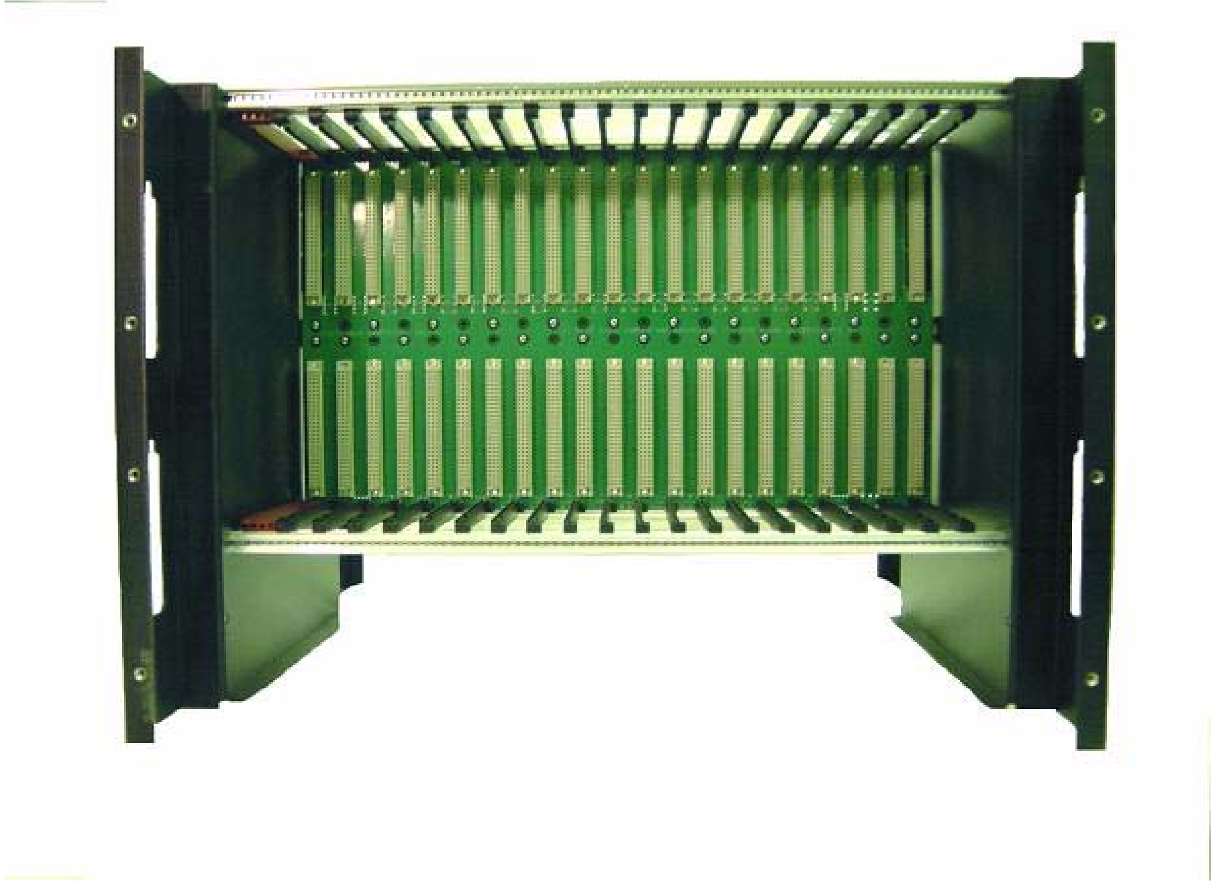
The Solflower PCI Expansion Chassis is a stand-alone PCI Expansion Chassis. It has four 64-bits PCI slots and one 32 bit slots. One of the slots is dedicated for the system controller, such as a SgPCI-LS board. The other four slots are standard PCI slots. Figure 12 briefly shows the back of the PCI Expansion Chassis.

FIGURE 12 PCI Expansion Chassis (back view)



The Solflower 21 Slots VME chassis provides expansion from PCI to VME. The Solflower SgVME board translate traffics between PCI and VME buses with 2.5Gbps bit rate. The SgVME board is seated at the system slot (first slot on your left) of the chassis, and acts as a system controller on behalf of the VME bus.

FIGURE 13 **Solflower 21 Slots VME Chassis**



CHAPTER 3 Installation Procedures

It is important that certain steps should be followed in order to have a trouble-free installation. Certain steps might need to carried out in specific order, otherwise the system might not be able to provide proper services. This chapter details procedures for hardware installation as well as software installation with SUN machines.

To prevent damage to the board and system electronic components: Do not handle the board without ESD precaution. Solflower Computer recommends user to ware an anti-static strap or similar. Do not install or remove the board when your system is powered on.

3.1 New installation

For a new installation of Sg boards, (meaning there is no previous installation of Sg boards and SOLFpvme driver in the same system) the steps list below should be followed.

1. Shut down your host system with “sync; halt”

2. Disconnect system’s AC power.

3. Find the desired PCI slot in your system (please refer to Chapter 4) and install SgPCI-2 board into the PCI slot. Be sure the board is firmly seated.

4a. Install SgVME board to the system slot at Solflower 21 Slot VME Chassis. This VME slot with red card guide rail is located at the right most of the Chassis (or on your left most). Press the board in firmly.

4b. If there is a Solflower PCI expansion system, install SgPCI-LS to the system slot at Solflower expansion box.

5. Connect SOLFlower 21 Slot Chassis and Solflower PCI expansion to your host system with CAT5e cables. For cable connections, please refer to section 2.3 to 2.5.

6. Power on SOLFlower 21 Slot Chassis first, then turn on your host system power. Always follow this power sequence order. At this point you should see the link LEDs are on and solid at the front panels of both SgVME board and SgPCI-2 board.

7. Boot up your host system, and insert PVME Driver CD. Please make sure the CD is for the right machine and the right Solaris version.

8. At unix prompt, cd to your cdrom. e.g., /cdrom/cdrom0 directory. You should see the packages such as SOLFpvme and SOLFlib listed in the directory. Execute “pkgadd -d .” Choose the packages you want to install. Hit enter to install all packages.

9. After SOLFpvme and SOLFlib packages are installed successfully, type q to quit, reboot the system with “reboot” right away. Please do not install any leaf VME device driver at this point.

10. After your host system boots up, you can install your leaf VME device drivers. Edit your leaf VME device driver.conf file with “parent = pvme”.

3.2 Changing PCI slots

When there is a need to re-install SgPCI-2 to a different PCI slot, we need to do a few extra steps. This procedure will work for reinstalling SOLFpvme driver as well.

1. Remove all your VME leaf device driver packages. Then remove SOLFpvme package with “pkgrm SOLFpvme” and “pkgrm SOLFlib”.

2. Do “sync; halt” to bring down your host system.

3. Turn off system power and disconnect AC power source.

4. Reinstall SgPCI-2 board to the new PCI slot you’ve chosen.

5. Repeat the steps listed in section 3.1

CHAPTER 4

Hardware Installation with SUN Machines

When it comes to installing SgPCI-2 boards, it is crucial to know the hardware configurations of the SUN machines that you wish to install your boards. It is desirable to let SgPCI-2 occupy at a PCI bus segment by itself. In a SUN Blade 1500, for example, there are two PCI segments for the total of 5 PCI slots available. The best PCI slot to install SgPCI-2 would be PCI Slot 4. This Slot 4 is in a separate PCI segment other than the other 4 PCI slots, although it's sharing the same PCI segment with the system's network controller. If SgPCI-2 has to share the same PCI Bus segment with other PCI devices, it's important to make sure the SgPCI-2 is assigned to the highest PCI address for normal and optimal operation.

This chapter briefly outlines the optimal PCI slots to install SgPCI-2 boards with some of the SUN SPARC Solaris machines including SUN Blade 1500, SUN Fire V250, and SUN Fire V240, and X86 IA Solaris machines including W1100Z and V20Z.

4.1 SUN Blade 1500

For a Sun Blade 1500 machine, it is best to install SgPCI-2 in PCI Slot 4. This is a 64 bit, 66 MHz PCI slot with a PCI address that's suitable for VME devices address space. This slot is in a separated PCI segment from other 4 PCI slots. CAT5e cables are used to connect from SgPCI-2 to SgVME.(Please refer to Figure 7) Figure 14 shows a snapshot of the back panel of a Blade 1500 machine.

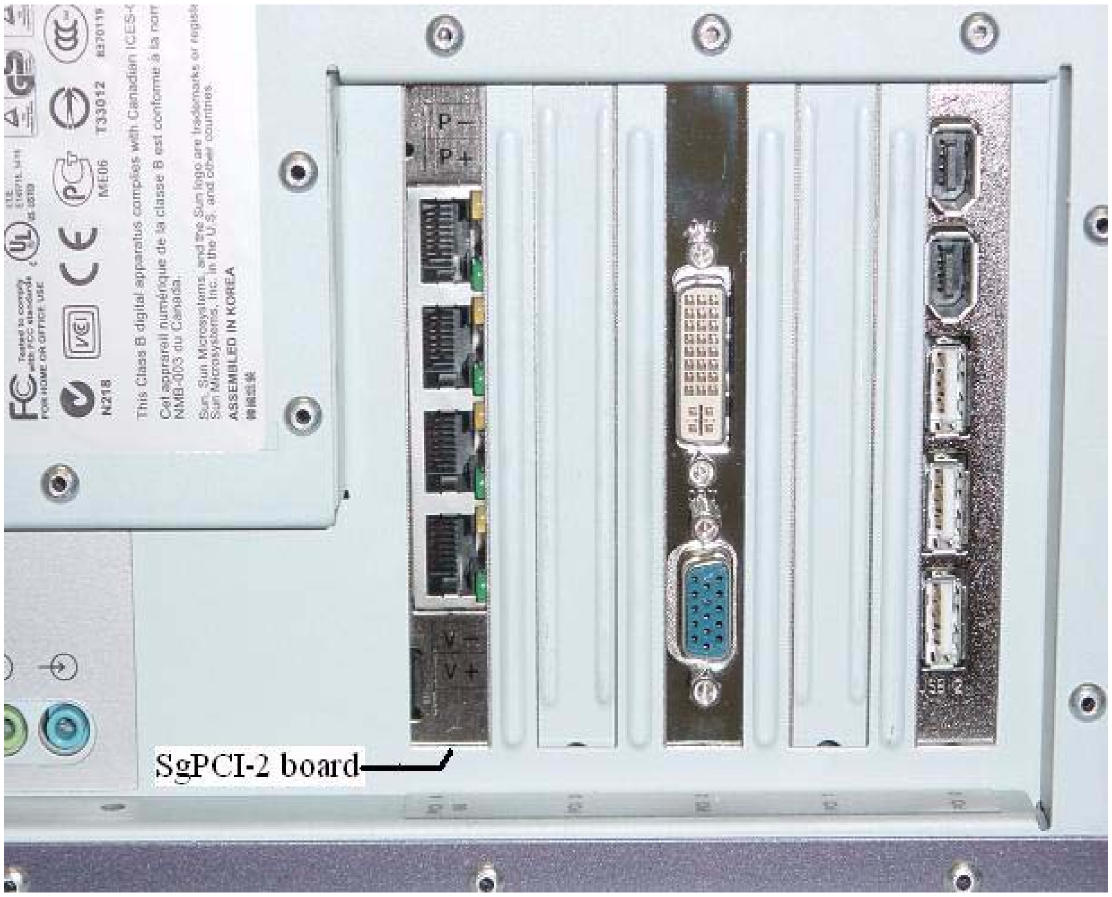
To verify SgPCI-2's PCI address space, go to OBP. Below is a sample printout from Blade 1500's OBP.

```
ok cd /

ok ls

f00c3844 os-io

f0072c50 i2c1f,464000

f006a5f4 pci@1f, 700000

f006a500 ppm@1e,0
```

```
f0061900 pci@1e,600000

f00617c4 memory-conrroller@0,0

ok cd /pci@1f, 700000/pci@3/pci@1

ok ls

pci10e3, 0@c
```

The printout shows there are two PCI segments in the PCI device tree, namely pci@1f, 700000 and pci@1e,600000. There is one PCI slot (PCI slot 4) under pci@1f, 700000 segment and four PCI slots (PCI slot 0, 1, 2, 3) under pci@1e,600000 segment. So the best PCI slot for SgPCI-2 would be slot 4. Walking toward the end node of /pci@1f, 700000/pci@3/pci@1, the end node device is pci10e3,0@c. This is the Universe II PCI-VME bridge on SgVME board.

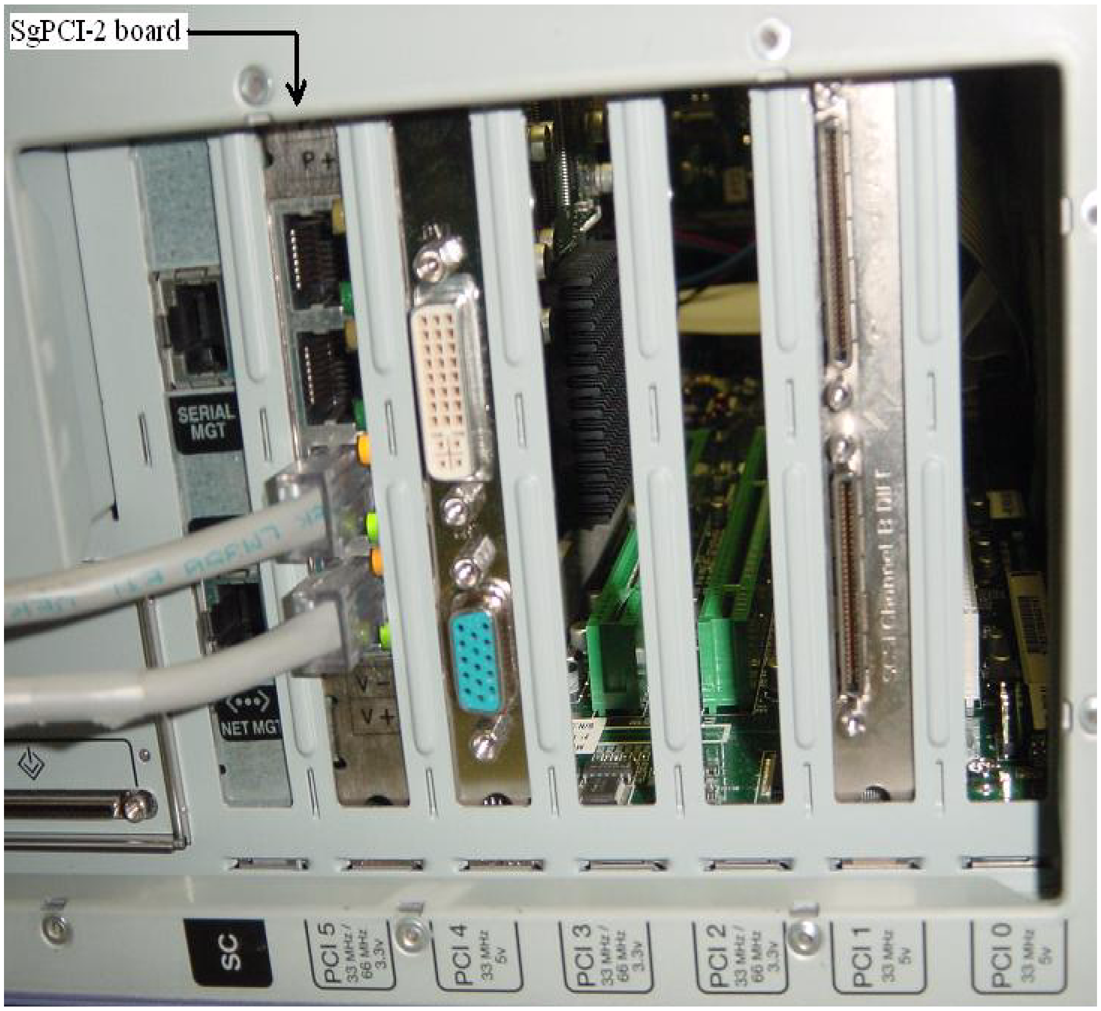
FIGURE 14 SUN Blade 1500 PCI slot 4 for SgPCI-2



4.2 Sun Fire V250 Installation

The best PCI slot to install SgPCI-2 is PCI slot 5 for optimal operation. PCI slot 5 is a stand alone PCI segment under /pci@1f,700000. There is no sharing PCI address space with other devices. Avoid installing SgPCI-2 in PCI2 and PCI3. These slots are in the same PCI segment with on-board SCSI controllers and other devices.

FIGURE 15 V250 PCI Slot 5 for SgPCI-2

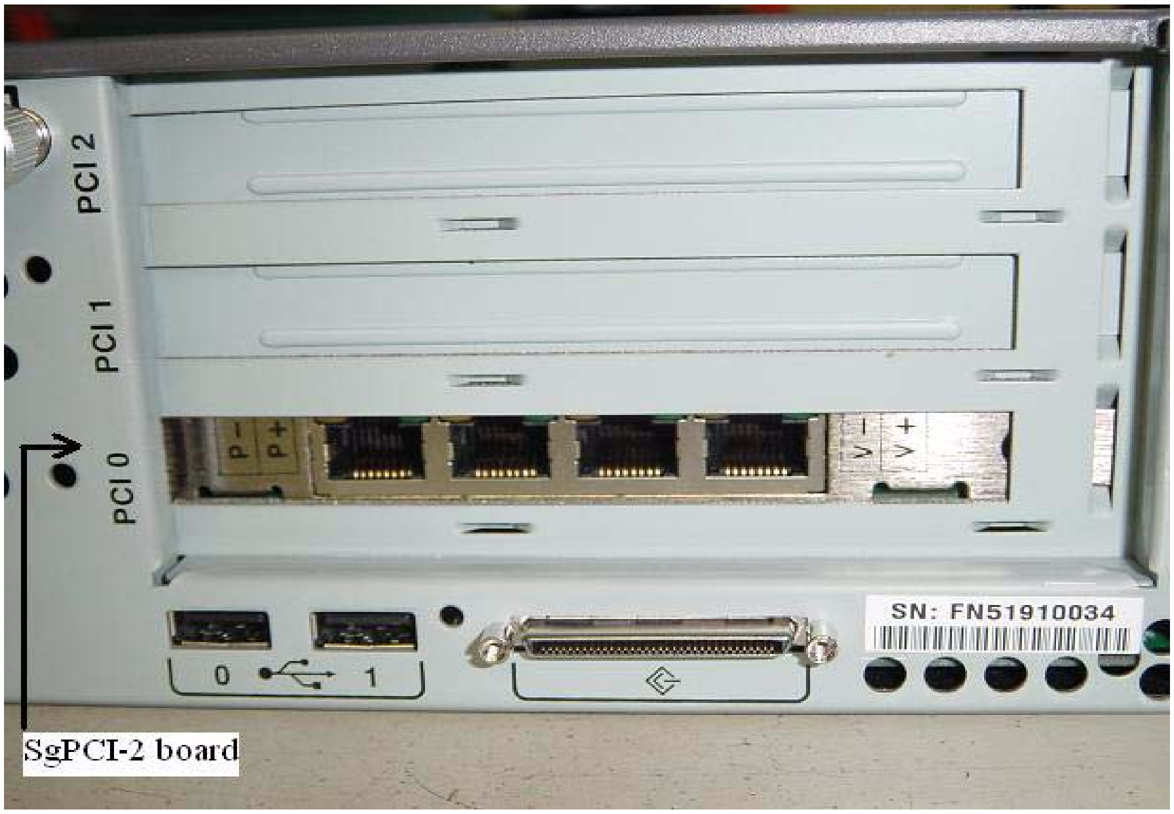

SgPCI-2 board
SERIAL MGT
NET MGT
SC
PCI 5 33 MHz / 66 MHz 3.3v
PCI 4 33 MHz 5v
PCI 3 33 MHz / 66 MHz 3.3v
PCI 2 33 MHz / 66 MHz 3.3v
PCI 1 33 MHz 5v
PCI 0 33 MHz 5v

4.3 SUN Fire V240

For SUN Fire V240, the best PCI slot would be PCI 0. (If you are sharing a pci bus segment between SgPCI-2 and a video graphic card, system might not boot.) Two PCI slots PCI1, and PCI2 are under one PCI segment, while PCI0 is under another PCI segment. From OBP, we can see PCI0 is under /pci@1d,700000. PCI0 is a 64 bit and 66 MHz stand alone PCI slot. It is the best PCI slot for SgPCI-2. Below is a printout from SUN Fire V240.

```
ok cd /

ok ls

pci@1d,700000

pci@1c,600000

pci@1e,600000

pci@1f,700000

memory-controller@1,0

SUNW, UltraSPARC-IIIi@1,0

......

ok cd /pci@1d,700000/pci@1/

ok .properties

....

ranges 82000000, 000000000, 00600000, 82000000, 000000000, 00600000, 00000000,
00100000

...

.
```

FIGURE 16 **SUN Fire V240 PCI Slot for SgPCI-2**



4.4 SUN Java W1100Z work station

For a W1100Z work station, pci slot 3 might not be a suitable choice for SgPCI-2. Experiments shows the fact that when SgPCI-2 is installed in this slot, it won't be assigned enough PCI memory for VME devices. While other PCI slots seem to be suitable for SgPCI-2, PCI slot 1 and 2 seem to be the best choice, because these two slots are in stand-alone PCI segments. Below is a sample printout of SgPCI-2 in PCI slot 1.

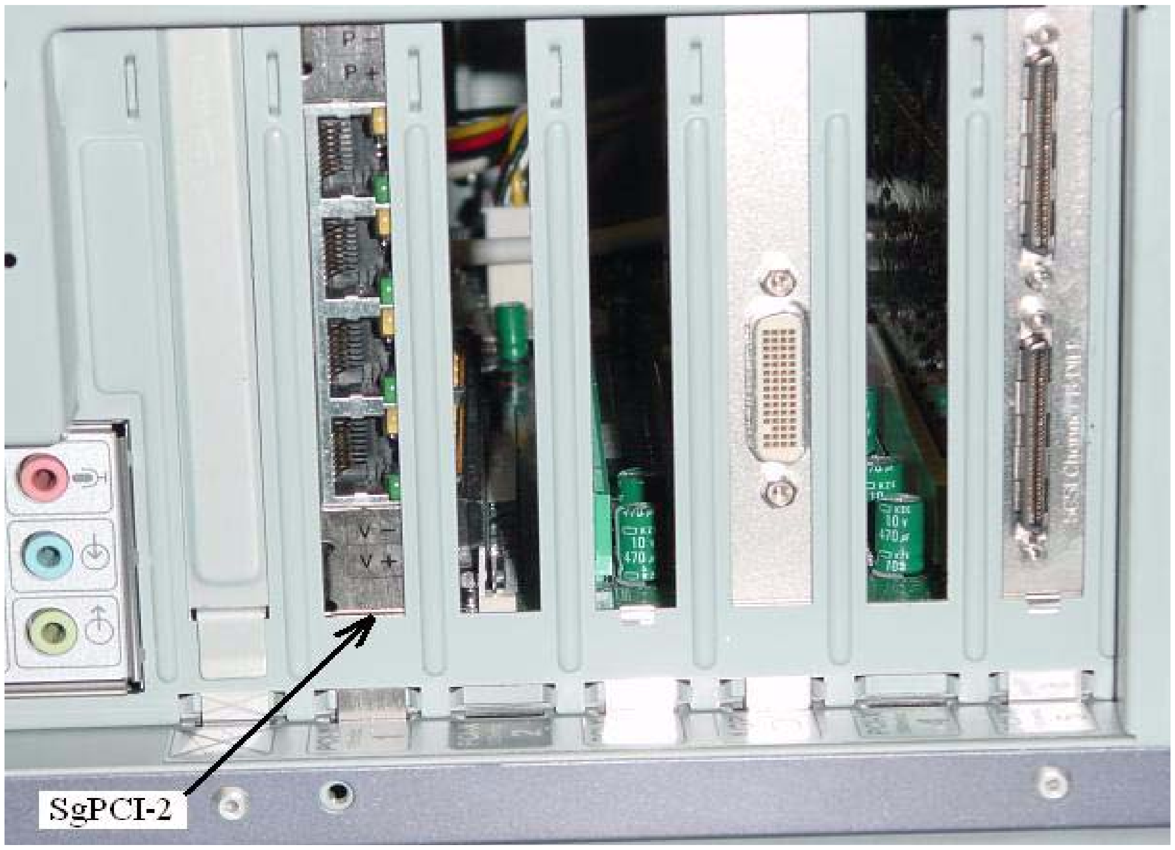
```
/dev/pvme/nexctl> ls

pci1022,7460@6

pci108e,534d

pci-ide@7,1

pci108e,534d

pci108e,534d

pci108e,534d

pci1022,7450@a

pci108e,534d

pci1022,7450@b

pci108e,534d

pci1022,1100

pci1022,1101

pci1022,1102

pci1022,1103

/dev/pvme/nexctl> cd pci1022,7450@a

pci1022,7450@a

/dev/pvme/nexctl> ls

pci9902,1@1

pci9902,2
```

/dev/pvme/nexctl>

As we can see from the printout above, SgPCI-2 board (shown as a dual function device pci9902,1@1 and pci9902,2) is in PCI segment../pci1022,7450@a by itself.

Figure 17 is a snap shot of SgPCI installed in PCI slot 1 with a W1100Z Java station.

FIGURE 17 **W1100Z Java Station**



4.5 SUN Fire V20Z

SUN Fire V20Z server has only two PCI slots. Both of the slots can install SgPCI-2. PCI slot 0 is more preferable over slot 1. The reason is slot 0 doesn’t share its PCI space with other devices in its PCI segment.

CHAPTER 5 Software installation for SgVME

5.1 Introduction

The pvme device driver is a 32- and 64-bit Solaris 8, 9, and 10 nexus driver for the Solflower PCI-to-VME bridge.

The pvme device driver permits the installation, configuration, and use of VME devices on Sun Microsystems UltraSPARC platforms that support the PCI bus, as well as on selected Intel or AMD platforms that run the Solaris 8, 9, and 10 operating system.

Most VME device drivers need only be written to conform to the Device Driver Interface (DDI) definition from Sun, and they should work transparently, with no need to be aware that they are communicating to Solaris through a bus bridge.

5.2 Features of the pvme Device Driver

- Complete Solaris Nexus driver, providing leaf driver management and translation for VME device mappings, interrupts, and DMA.
- Direct VMEbus access, though the export of VME address space device files.
- A special programmable DMA engine that transparently provides high-performance DMA transfers to and from VME address spaces when the VME device files are accessed by read(2) and write(2) operations.
- VMEbus power-off detection and management for user processes that access the pvme device files directly.
- Selectable operation modes for VME block transfer, arbitration, and posted writes.
- Loadable and unloadable Nexus driver.

5.3 Components

The pvme release from Solflower is comprised of binary executable files, libraries, and documentation. The table below provides a list of the major components.

Component Name	Component Description
/usr/kernel/drv/pvme	The loadable pvme device driver binary file.
/usr/kernel/drv/sparcv9/ pvme	(UltraSPARC specific) 64-bit version of pvme device driver.
/usr/kernel/drv/pvme.conf	The pvme driver configuration file. May be used to set a variety of control options governing the pvme nexus driver. Among them are options to assign at which SPARC processor interrupt level the pvme device interrupts, enables for use of posted writes, etc.
/opt/SOLFpvme/etc/ pvme.tab	Description for how to create the symbolic links in the / dev directory for the pvme device files. When you run pkgadd(1M) to install the pvme device driver, the installation scripts automatically appends these lines to the system /etc/devlink.tab file.
/opt/SOLFpvme/vmemem	A directory that contains a simple VME pseudo driver. This driver provides access to a VME memory card. This is not necessarily the way you would want to write such a driver, but it illustrates how to access and configure a VME device under the pvme nexus driver.
/opt/SOLFpvme/drv_test/ pvme/pvme_test	A directory that contains a test program and source code that shows you an example of how to open the pvme device files and perform various operations, such as mmap a chunk of VME memory or dump the contents of the pvme control registers. Included is a simple VME memory verification test.
/opt/SOLFpvme/drv_test/ vmemem	A directory that contains a simple test program and source code that shows you an example of how to interact with the ioctl(2) interface exported by the vmemem device driver.
/opt/SOLFpvme/doc	A directory that contains documentation on the pvme and vmemem drivers, as well as information about test programs.
/opt/SOLFlib/drv_lib	A directory that contains source code and a binary library for building the test programs for the pvme and vmemem drivers.
/opt/SOLFlib/drv_util	A directory that contains include files and a binary library for building the vmemem device driver. Certain of the test programs may use some of the include files, also.
/opt/SOLFpvme/bin	The home of the a_pvme and r_pvme scripts, which are useful for installing and removing the pvme driver on a running system. Also contains the test programs pvme_test.

5.4 Installation

The pvme device driver is delivered on cdrom as a Solaris 2.x package.

The pkgadd(1m) operation should copy and install the pvme driver for you. If a situation occurs where you need to circumvent the installation performed by pkgadd(1m) and perform the installation by hand, the following summarizes the operations performed by the package installation wrapper.

1. Because the pvme driver is not essential for boot-up of the Solaris operating system, by convention, the driver executable, pvme and the driver configuration file, pvme.conf, are copied to the /usr/kernel/drv directory. These components could also have been placed under the /kernel/drv or /platform/sun4u/kernel/drv paths.

2. Entries specifying the device files to create for pvme are appended to the system /etc/devlink.tab file. Before inserting the requisite lines for pvme, we make a backup of /etc/devlink.tab, called /etc/devlink.tab.BAK, so the original file may be recovered if problems occur.

   The SOLFpvme package installation scripts append the lines for pvme automatically. We provide a file with the SOLFpvme package called pvme.tab that contains the lines to be inserted. This file will be resident in the SOLFpvme package directory, once installed. Typically, the pvme.tab file may be found at /opt/SOLFpvme/etc/pvme.tab.

   The following operations mimic the installation operations with respect to the specification of device links:

```
# cp /etc/devlink.tab /etc/devlink.tab.BAK
# cat /etc/devlink.tab.BAK /opt/SOLFpvme/etc/pvme.tab > /etc/devlink.tab
```

3. Once the pvme driver files are placed under /usr/kernel/drv, and the device link specifications have been added to /etc/devlink.tab, the pvme driver is installed and loaded via the Solaris add_drv(1M) command. This is the syntax used by the package installations scripts to install and load the driver:

```
# add_drv -c vme -i '"pci10e3,0"' pvme
```

The system /etc/driver_aliases file is updated by this operation as well.

Assuming that the operations above complete successfully, the pvme driver is now loaded. Henceforth, if the Solaris system is shut down and rebooted, the pvme device driver will be automatically loaded.

To undo this operation requires that the system administrator deinstall the driver, using the rem_drv(1M) command.

4. When the pvme driver is loaded by the Solaris operating system, it prints a banner on the system console. This banner contains information about the pvme product, including version, build date, copyright, and Solflower-specific driver library components employed for this release of the product.

Below is an example of such a banner printed as display on the console. Note that this same information is also sent to the system log file (i.e., syslog).

```
PVME - PCI-to-VME bridge nexus driver, version 8.9(1)
Copyright (c) 1997-2002 by Solflower Computer, Inc.(2)
Built: Wed Aug 28 10:43:23 PDT 2002 (3)
Solflower Driver Kit, Module: DRV-util, Version 1.0(4)
Copyright (c) 2002 by Solflower Computer, Inc.(5)
Built: Tue Aug 27 13:53:10 PDT 2002 (6)
```

Notes:

(1)PVME driver version information

(2)Copyright notice

(3)Build date

(4-6)Solflower Driver Kit module:

version, copyright, and build date

Note: If SOLFpvme driver was installed in a different PCI-VME subsystem than Solflower SgVME-PCI, the installation will stop with an error message " not a Solflower subsystem".

5.5 Theory of Operation

5.5.1 T-of-O: Introduction

The primary tasks for which the pvme device driver is responsible are two-fold.

First, pvme provides memory address translation between the VME bus and PCI bus memory spaces. Second, it manages VME interrupts and vectors. Since the pvme device is a PCI device, it maps associated VME actions and operations into analogous ones for the PCI bus, where possible.

5.5.2 VME-to-PCI address translation

When the pvme driver is installed, it examines the PCI bus device tree to determine which addresses are already occupied and thus how much memory is available for accessing VMEbus devices. If other PCI devices are installed in the same PCI space, the amount of PCI space available for VMEbus access is reduced.

However, even when the free PCI memory space is heavily fragmented, the pvme driver can, in many cases, work around holes in the PCI memory space caused by other devices. The pvme driver accomplishes this by using slave image windows.

Slave image windows are a PCI-to-VME address translation mechanism provided by the pvme hardware; they are in many respects analogous to memory management mapping hardware. A slave image window operates much like an MMU page table entry: it translates memory accesses on a PCI memory address range into corresponding memory accesses on a VME memory address range.

The pvme hardware provides up to 8 slave image windows for mapping address ranges on the VMEbus. Once initialized, a slave image window may provide PCI-to-VME address translation for any VME A32, A24, or A16 space.

By using the pvme slave image windows, the driver can hide the effects of fragmentation of the PCI address space.

For example, PCI frame buffers may occupy multiple megabytes of space for their bit-planes. The pvme device and driver CANNOT use the PCI space occupied by these bit-planes for mapping or accessing the VMEbus. However, using the Solaris device tree, the pvme driver is able to detect the presence of such a device, and can use the PCI memory ranges that are lower or higher than the occupied regions of memory.

The default sizes assigned to each slave image window are predefined as a function of the type of VME address space being mapped. (Note that the default size for each VME address space slave image window may be overridden by adding directives to the pvme.conf configuration file.)

The availability of multiple slave image windows provides flexibility. If a mapping operation requires more than 32 MBytes of contiguous A32 memory space, more than one pvme slave image window will be used. Mapping requests made to the pvme driver will silently detect whether a new slave image window is required, and if so, will allocate and initialize the window so that the desired VMEbus addresses can be mapped. Unless numerous "sparse" mappings are made, the user should not be aware that mapping windows are being used.

The pvme device driver assigns the PCI memory physical addresses for accessing VMEbus address space by using hardware base and bound registers of the slave image windows. These base and bound registers permit us to define space on the PCI bus for our use that did not show up when our device was initialized by the Sun Open Boot PROM.

The pvme device driver must allocate the PCI memory space to use for these slave image windows from that memory space not used by other PCI devices on the same bus as the pvme hardware. So, in a sense, the pvme is competing for space against its sibling PCI devices. Consequently, if your system needs for VME address space are large, it is best to put the pvme hardware on the least occupied PCI bus available. If there is only one PCI bus on the system, of course, this is not a consideration.

5.5.3 VME interrupt management

The VME bus supports 7 interrupt priority levels. The PCI nexus can support only 1 interrupt. Thus, all VME interrupts are in a sense "funneled" into a single PCI interrupt. When an VME interrupt occurs, the pvme driver scans for pending VME interrupts from highest to lowest priorities. Thus, in theory, higher priority VME interrupts are therefore given preference over lower priority VME interrupts.

However, short term priority inversion of interrupt handlers is possible with this scheme. There is a timing window where a lower priority VME interrupt may arrive and activate its interrupt handler, and thereby block the servicing of a higher priority VME interrupt until the low priority interrupt handler completes. When the interrupt handler completes, the higher priority interrupt will occur and be serviced. The duration of potential interrupt priority inversion is therefore limited by the duration of the other VME interrupt handlers.

VME devices use an interrupt vector scheme to determine the source of any VME interrupt. The VME vector is a unique number between 1 and 255. Each device, when an interrupt acknowledgement cycle is run by the pvme hardware, emits its unique identifying vector on the VME bus. The pvme hardware receives this vector, and the pvme driver uses it to find the appropriate handler for that VME vector.

This implies that there can be only one handler for each unique VME vector.

5.5.4 Bus Error Monitoring

If there is no "Target Abort" or "Bus Error" signal terminate a non-existing-memory cycle, there are two options to generate Bus Error from pvme driver level:

The first option is to call one of pvme driver's ioctl after each memory access cycle. Here is an example of how to implement it:

```
/*
* vaddr is the virtual address of VME memory.
* out is the virtual address of a user buffer.
*/
static int
dev_peek32(int fd, uint32_t * vaddr, uint32_t * out)
{
    int tabort = 0;

    *out = *vaddr;
    if (dev_ioctl(fd, PVME_TARGET_ABORT, &tabort) == 0) {
            return (tabort ? -1 : 0);
    }
    return (0);
}
```

For more detail, please take a look at the files pvme_dev.c and pvme_mem.c in the directory SOLFpvme/drv_test/pvme/pvme_test. The pvme_test program also implements this as the "-b" option. For example, "pvme_test -b -i -s 100 -o 30000000" reads 100 bytes at address 0x30000000 and checks if bus error occurs. “pvme_test -b -W -s 100 -o 55000000” writes 100 bytes at address 0x55000000 and checks if bus error occurs.

The second option is to call pvme_test program before any memory access: "pvme_test -t <ms>". This needs to be called only once to start a daemon monitoring bus error. "ms" is how many milliseconds the daemon should wait before checking status of the bus again, and it must be between 100ms to 1000ms. The default wait time is 500ms. Use "pvme_test -t 0" to stop the daemon.

Note: Reading from a memory location to determine if it is valid is safer than writing to it since writing to an invalid address can panic the system. Writing also has posted-write option to be considered. To detect bus error correctly while writing, posted-write must be disable. Posted-write can be disable or enable by an entry in the file pvme.conf: post-writes=”yes|no”;

5.6 PVME Device Files

The pvme nexus driver creates 13 device files in /dev to permit access to the VMEbus and the pvme control registers.

In general, the user should not need to access the pvme control registers.

The VMEbus device files are potentially useful when you wish to mmap(2) a chunk of VME address space and access it directly with a user program.

5.6.1 VME Device Files

There are 12 device files exported by the pvme driver that represent the various VME address spaces, plus associated data widths:

Device File	Description
/dev/pvme/vme16d64	Used to access A16 space with D64 width.
/dev/pvme/vme24d64	Used to access A24 space with D64 width.
/dev/pvme/vme32d64	Used to access A32 space with D64 width.
/dev/pvme/vme16d32	Used to access A16 space with D32 width.
/dev/pvme/vme24d32	Used to access A24 space with D32 width.
/dev/pvme/vme32d32	Used to access A32 space with D32 width.
/dev/pvme/vme16d16	Used to access A16 space with D16 width.
/dev/pvme/vme24d16	Used to access A24 space with D16 width.
/dev/pvme/vme32d16	Used to access A32 space with D16 width.
/dev/pvme/vme16d8	Used to access A16 space with D8 width.
/dev/pvme/vme24d8	Used to access A24 space with D8 width.
/dev/pvme/vme32d8	Used to access A32 space with D8 width.

5.6.2 PVME Control Registers

The pvme nexus driver exports another device file for direct access to the nexus's control registers:

Device	Description
/dev/pvme/nexctl	Used to access the pvme control registers.

This file is present for diagnostic purposes, such as to examine the current control register settings of the pvme. It is not ordinarily something that the user has a need to use. Since it permits direct access to the pvme device registers, it is possible to change pvme control register settings with this mechanism. The user is advised to exercise great caution if he desires to try this, since an incorrect value inserted into one of the pvme control registers has the potential to put the pvme into a mode that could result in a system panic.

5.6.3 Device File Example

A typical program would access one of these VME spaces from a user program in the following manner:

```
char * vme_map(char * vme_device, int vme_off, int bytes)

{

int fd;


fd = open(vme_device, O_RDWR);

printf("[Mapping 0x%x bytes at VME offset 0x%x]\n", bytes, vme_off);

return (mmap((caddr_t)0, bytes, PROT_RW, MAP_SHARED, fd, vme_off));

}


#defineVME_OFFSET0x00100000

#defineVME_BYTES0x00010000
```

```
int main()
{
char *vme_mem;
charvme_byte;

/*
 * map VME_BYTES of VME memory at location
 * VME_OFFSET in VME A32 space.
 */

vme_mem = vme_map("/dev/pvme/vme32d32", VME_OFFSET, VME_BYTES);

/*
 * Now read the first byte at the
 * specified VME offset.
 */

vme_byte = vme_mem[0];
}
```

5.7 PVME Test Program

The tests included in the drv_test/pvme directory include

pvme_test

The pvme_test program allows users verify the state of VME mapping and interrupt activities by examining the register state of the Universe chip. A few examples are given below.

pvme_pages

This is a simple VME memory mapping and access test.

5.7.1 A pvme_test Example

To print out all available options of the pvme_test program, use the -h option to pvme_test:

```
# pvme_test -h

[PVME_TEST - Version: 2.0 Date: Sun Sep 9 13:05:55 PDT 2001 ]

-A - tests don't ask the user for input

-a - set VME address space for memory and DMA tests

-B - print pvme_test build date and version

-b - VME memory accesses check for bus errors

-C - run a memory comparison test

-c - reset the VME bus

-D - disconnect busops

-d - run the DMA test

-g - generate a VME software interrupt

-h - get this message

-H - get ownership of the VME bus
```

-I - clear pending interrupts

-i - read VME memory

-L - run the memory test in store/load mode

-l - Parent memory limit check

-m - run the memory test

-M - run VME mapping tests

-n - toggled interrupt recording switch

-N - print interrupt record

-O - Check the power on state of the VME bus

-o - set the starting address offset for memory and DMA tests

-P - VME switched on: toggle state

-p - print register state

-R - reconnect busops

-r - PCI resource map limits

-s - set the size to use for DMA and memory tests

-S - set the stride length for memory tests

-T - memory test mode: increment or walking ones

-t - monitor target abort with the specified time interval

-U - release ownership of the VME bus

-V - device registers

-v - set the initial test pattern for memory tests

-W - write VME memory

-w - CPU uses 64, 32, 16, or 8 bit memory operations (DEFAULT 32)

-X - select the VME HW-enforced Data Width to use for memory & DMA tests

5.7.2 A pvme_test example: Examining Universe Registers

```
# /opt/SOLFpvme/bin/pvme_test -p

[opening /dev/pvme/nexctl]

VID 10e3 DID 0 COMM 107 <IO><MAE><ME><SERR> STATUS 200
<DSTMED>

CLASS revid 2 prog 0 sub 80 base 6

Cache line size 0 Latency Timer 0 Header Type 0 BIST 0

BAR[0] ffafd000 BAR[1] 0000d001 BAR[2] 00000000

BAR[3] 00000000 BAR[4] 00000000 BAR[5] 00000000

CIS Pointer 0 SUBSYSTEM: VID 0 ID 0 CAP 0 ROM 0

MAX LAT 0 MIN GNT 3 ILINE b IPIN 1

PCI SI[0] CTL c0400000 BAR ffa00000 BD ffa10000 TO 00600000

PCI SI[1] CTL c0820000 BAR 20000000 BD 22000000 TO 78000000

PCI SI[2] CTL 00000000 BAR 00000000 BD 00000000 TO 00000000

PCI SI[3] CTL 00000000 BAR 00000000 BD 00000000 TO 00000000

-------------

PCI SI[4] CTL 00000000 BAR 00000000 BD 00000000 TO 00000000

PCI SI[5] CTL 00000000 BAR 00000000 BD 00000000 TO 00000000

PCI SI[6] CTL 00000000 BAR 00000000 BD 00000000 TO 00000000

PCI SI[7] CTL 00000000 BAR 00000000 BD 00000000 TO 00000000

-------------

VME SI[0] CTL 80f20000 BAR 00000000 BD 00fff000 TO 00000000

VME SI[1] CTL 80f10000 BAR 00000000 BD 00ff0000 TO 00000000

VME SI[2] CTL 00000000 BAR 00000000 BD 00000000 TO 00000000

VME SI[3] CTL 00f00000 BAR 00000000 BD 00000000 TO 00000000

-------------
```

```
  SPEC CYCL CTL 00000000 ADDR 00000000 EN 00000000 CMP 00000000 SWP
00000000

  PCI LMISC 05 SSI 00000000 ERR_LOG CMD 70000000 ADDR 00000000

  PCI INTR ENABLE 000081fe STATUS 00004000 MAP 0..1 00000000..00000000

  -------------

  VME INTR ENABLE 00000000 STATUS 00000000 MAP 0..1 76543210..00000001

  VME INTR STATUS-ID fe000000

  IRQ1 00000000 IRQ2 00000000 IRQ3 00000000 IRQ4 00000000

  IRQ5 00000000 IRQ6 00000000 IRQ7 00000000

  -------------

  DMA DCTL 00000000 DTBC 00000000 DLA 00000000 DVA 00000000

  DMA CPP 00000000 DGCS 00000000 DLLUE 00000000

  -------------

  VME CTL MAST 84e01000 MISC 76060000 MSTAT 00260000 USER_AM
40400000

  -------------

  VME RAI CTL 00000000 BAR 00000000

  VME CSR CTL 00000000 CSR_TO 00000000 AM_ER 24000000 ADDR_ER
992fff84

  VME CSR CLR 40000000 SET 40000000 BAR 00000000

  -------------
```

5.8 Known Limitations

If a VME master transfers data to PCI memory at the same time as the CPU attempts to access VME space, a PCI timeout may occur that causes the system to panic with the message "too many retries".

This problem arises because Solaris initializes the PCI bus arbiter to use an insufficient number of retries when contention occurs. The result is that if a VME master holds the bus for a long time when the CPU is trying to access the PCI space, the PCI bus arbiter may gives up too early.

Before you attempt to access the VMEbus with the CPU, you should make sure that no VMEbus device is performing a transfer of data to PCI memory. Typically, a way to trigger this problem is to use mmap(2) to map a chunk of VME memory into a program's address space and then to access the mmapped VME memory it while a VME device is performing a DMA transfer. Such simultaneous attempts to access the VMEbus should be avoided by the user as much as possible.

CHAPTER 6 Trouble shooting

In trouble shooting section, we assume SgPCI-2, SgVME and SgPCI-LS were properly installed and seated securely.

Link LEDs are off

Make sure the cables are connected correctly according to the label on the CAT5e cables and board front panels. Check if the +/- CAT5e cables are swapped. Turn on the Solflower 21 Slots Chassis (and PCI expansion system) first, then power up the Host. Make sure power supplies of the Solflower 21 Slot VME Chassis is good. If the LEDs are still OFF, replace the parts one at a time: CAT5e cable, SgPCI-LS (if there is any), SgVME, then SgPCI-2.

Link LEDs are blinking, Links were not established.

Be sure to turn on the Solflower 21 Slots VME Chassis (and PCI expansion system) before the Host. The Host is responsible for enumerating the SgPVME and SgPCI-LS (if any) and all PCI buses behind the SgPCI-2 card. If the Host is ON before the expansion box, the enumeration will complete without seeing that expansion PCI path. Issue reset-all command at OBP to force new Host enumeration cycle.

At SUN machine's OBP level, one or more leaves are not seen.

Try “reset-all” at the OK prompt. Due to power up delay or some other reasons, the leaves were in some unknown state when the OBP is probing the device tree. Thus OBP will not recognize them. “reset-all” will reset the devices tree, and restart a new probing.

Cable connections are correct, power sequence was correct, tried “reset-all”, but the link LEDs are still blinking.

Check the board model # used in the system. SgPCI-2 (P/N 621-001-2CN) can be connected with SgPCI-LS (P/N 519-001-2CN) and SgVME (P/N621-002-SVME), but not SgPCI-2(P/N 621-001-2CN) . These boards physically are almost identical but configuration is set differently from factory as default, and should not be altered on the field.

My application program cannot communicate with my leaf VME devices

a. Solflower SgVME driver SOLFpvme is written strictly for used with Solflower SgVME-PCI Adaptor board set. SOLFpvme will not attach to if the hardware configuration is different than Solflower suggested setting.

b. Make sure your device driver.conf file is set to “parent = pvme”

c. Check and see if the PCI slot the installed SgPCI-2 is a suitable slot. Please refer to Chapter 4.

d. if a, b,c wasn't the case, try to reinstall SOLFpvme driver. Please refer to section 3.2

System get stuck at boot up

In some rare instances at boot up, after SOLFpvme driver is loaded by the system at OBP, but it's not ready to attach the leaf VME device drivers on the VME bus, the system will get stuck at that point trying to allocate resource for leaf VME device(s). In this case we need to do the following:

Disconnect CAT5e cables from SgPCI-2 board; reboot the system; after the system boot up, remove all your leaf VME device drivers; remove SOLFpvme driver by "/opt/SOLFpvme/bin/r_pvme"; reconnect the CAT5e cables to SgPCI-2 board; reboot again; add SOLFpvme driver by "/opt/SOLFpvme/bin/a_pvme"; add your leaf device drivers.

CHAPTER 7

Physical and Environmental Specifications

7.1 Mechanical Dimensions

SgPCI-2:
Board Dimension:6.5 in x 4 in (L x W)
Front Panel Dimension: 4.75 in x 0.72 in (L x H)

SgPCI-LS:
Board Dimension:6.5 in x 4 in (L x W)
Front Panel Dimension: 4.75 in x 0.72 in (L x H)

SgVME:
Board Dimension:9.2 in x 6.8 in (L x W)
Front Panel Dimension: 10.3 in x 0.8 in (L x H)

7.2 Operating Environment

7.2.1 SgVME Board

Operational Humidity: 5-95% RH non condensing at 40 C

Non-Operational Humidity: 5-95% at 40 C

Operating Altitude: 10,000ft

Non-Operating Altitude: 40,000 ft.

Cooling Requirement: 80 LFM minimum

Power Consumption: 5V only @4.5A max

MTBF 100,000 poh

7.2.2 SgPCI-2 and SgPCI-LS

Operational Humidity: 5-95% RH non condensing at 40 C

Non-Operating Humidity: 5-95% at 40 C

Operating Altitude: 10,000ft

Non-Operational Altitude: 40,000 ft.

Cooling Requirement: 80 LFM minimum

Power Consumption: 5V only @4.5A max
MTBF 100,000 poh

CHAPTER 8

Ordering Informations

- 2 Channels PCI/VME host card (SgPCI-2)--P/N:621-001-2CN
- StarPCI-VME combo bridge (SgVME)--P/N: 621-002-SVME
- PCI Expansion System Controller board (SgPCI-LS)--P/N:519-001-2CN
- CBL-CAT5e,7 7Ft CATe 5 cable--P/N: CBL-CAT67
- CBL-CAT5e,15 15Ft CATe 5 cable--P/N: CBL-CAT615
- CBL-CAT5e,40 40Ft CATe 5 cable--P/N: CBL-CAT640
- Solaris nexus driver for SPARC--P/N: 621-PVME-006 VME-4
- VME-4 Solaris nexus driver for X86--P/N:621-PVMEX86-007
- User Manual--P/N: 621-PVME-UM01

CHAPTER 9

Warranty, Maintenance, and Technical Support

Solflower products are warranted against defective materials and workmanship within the warranty period of one year from date of invoice. Within the warranty period, Solflower will, free of charge, repair or replace any defective unit covered by this warranty, shipping prepaid. A Return Material Authorization number (RMA # must be obtained from Solflower prior to the return of any defective product.

Solflower's warranty is limited to the repair or replacement policy described above and neither Solflower or it agent shall be responsible for consequential or special damages related to the use of their products.

Any questions or requests for repair/technical support should be directed to:

Solflower Computer, Inc.
3511 Thomas Road, Ste-2
Santa Clara, CA 95054
Phone: (408) 982-8680
Fax: (408) 982-8685
E-Mail: info@solflower.com
URL: http://www.solflower.com