

# SunHSI/S™ 3.0 Installation and Administration Guide

---



THE NETWORK IS THE COMPUTER™

**Sun Microsystems, Inc.**  
901 San Antonio Road  
Palo Alto, CA 94303-4900 USA  
650 960-1300 Fax 650 969-9131

Part No. 805-6941-10  
November 1998, Revision A

Send comments about this document to: [docfeedback@sun.com](mailto:docfeedback@sun.com)

Copyright 1998 Sun Microsystems, Inc., 901 San Antonio Road • Palo Alto, CA 94303 USA. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, AnswerBook, Java, the Java Coffee Cup, SunHSI/S, docs.sun.com, SPARCserver, SunSolve, SPARCstation, IPC, SunLink, SunVTS, and Solaris are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

**RESTRICTED RIGHTS:** Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g)(2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015(b)(6/95) and DFAR 227.7202-3(a).

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 1998 Sun Microsystems, Inc., 901 San Antonio Road • Palo Alto, CA 94303 Etats-Unis. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées des systèmes Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, AnswerBook, Java, le logo Java Coffee Cup, SunHSI/S, docs.sun.com, SPARCserver, SunSolve, SPARCstation, IPC, SunLink, SunVTS, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REPENDRE A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.



## Regulatory Compliance Statements

Your Sun product is marked to indicate its compliance class:

- Federal Communications Commission (FCC) — USA
- Department of Communications (DOC) — Canada
- Voluntary Control Council for Interference (VCCI) — Japan

Please read the appropriate section that corresponds to the marking on your Sun product before attempting to install the product.

### FCC Class A Notice

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:

1. This device may not cause harmful interference.
2. This device must accept any interference received, including interference that may cause undesired operation.

**Note:** This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

**Shielded Cables:** Connections between the workstation and peripherals must be made using shielded cables in order to maintain compliance with FCC radio frequency emission limits. Networking connections can be made using unshielded twisted-pair (UTP) cables.

**Modifications:** Any modifications made to this device that are not approved by Sun Microsystems, Inc. may void the authority granted to the user by the FCC to operate this equipment.

### FCC Class B Notice

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:

1. This device may not cause harmful interference.
2. This device must accept any interference received, including interference that may cause undesired operation.

**Note:** This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/television technician for help.

**Shielded Cables:** Connections between the workstation and peripherals must be made using shielded cables in order to maintain compliance with FCC radio frequency emission limits. Networking connections can be made using unshielded twisted pair (UTP) cables.

**Modifications:** Any modifications made to this device that are not approved by Sun Microsystems, Inc. may void the authority granted to the user by the FCC to operate this equipment.

## DOC Class A Notice - Avis DOC, Classe A

This Class A digital apparatus meets all requirements of the Canadian Interference-Causing Equipment Regulations.  
Cet appareil numérique de la classe A respecte toutes les exigences du Règlement sur le matériel brouilleur du Canada.

## DOC Class B Notice - Avis DOC, Classe B

This Class B digital apparatus meets all requirements of the Canadian Interference-Causing Equipment Regulations.  
Cet appareil numérique de la classe B respecte toutes les exigences du Règlement sur le matériel brouilleur du Canada.

## VCCI 基準について


### 第一種 VCCI 基準について

第一種VCCIの表示があるワークステーションおよびオプション製品は、第一種情報装置です。これらの製品には、下記の項目が該当します。

この装置は、第一種情報装置(商工業地域において使用されるべき情報装置)で商工業地域での電波障害防止を目的とした情報処理装置等電波障害自主規制協議会(VCCI)基準に適合しております。したがって、本製品を、住宅地域または住宅地域に隣接した地域でご使用になりますと、ラジオ、テレビジョン受信機等に受信障害を与えることがあります。

取り扱い説明書に従って正しくお取り扱いください。

### 第二種 VCCI 基準について

第二種VCCIの表示  があるワークステーションおよびオプション製品は、第二種情報装置です。これらの製品には、下記の項目が該当します。

この装置は、第二種情報装置(住宅地域または住宅地域に隣接した地域において使用されるべき情報装置)で住宅地域での電波障害防止を目的とした情報処理装置等電波障害自主規制協議会(VCCI)基準に適合しております。しかし、本製品を、ラジオ、テレビジョン受信機に近接してご使用になりますと、受信障害の原因となることがあります。

取り扱い説明書に従って正しくお取り扱いください。

# Contents

---

**Preface** xiii

**1. Overview** 1

Hardware Description 2

SunHSI/S Serial Channels 4

SunHSI/S-to-SBus Interface 4

Aggregate Bandwidth 4

Power Consumption 4

Software Description 5

Network Device Driver 5

Diagnostic Utilities 5

Diagnostics 5

Getting Help 6

**2. Installing the SunHSI/S Adapter and Patch Panel** 7

Tools and Equipment 7

Installing the SunHSI/S Adapter 7

Installing the SunHSI/S Patch Panel 8

▼ To Mount the Patch Panel on a Wall 9

▼ To Install the Patch Panel in a Rack 10

▼	To Install the Patch Panel in a SPARCserver 690MP	11
	Connecting the 96-Pin Cable	13
▼	To Connect the Patch Panel to the SunHSI/S Adapter With the 96-pin Cable	13
	SunHSI/S Cabling	14
	RS-232 to RS-449 Connections	14
	Verifying the Installation and Rebooting the System	14
▼	To Power On Your System	15
<b>3.</b>	<b>Installing the SunHSI/S 3.0 Software</b>	<b>17</b>
▼	To Remove Older Versions of the SunHSI/S Software	19
▼	To Mount the SunHSI/S 3.0 CD-ROM	20
	Files and Directories on the CD-ROM	20
▼	To Install the SunHSI/S Software	21
	Before Operating the SunHSI/S Adapter	22
	Changing the Cabling or Equipment	22
	Checking the MTU and MRU Sizes	22
	Viewing the Man Pages	23
▼	To View the Man Pages in the C Shell Environment	23
▼	To View the Man Pages in Bourne or Korn Shell Environments	24
	Removing the SunHSI/S 3.0 Software	24
	Console Messages	25
	Informational Messages	25
	Error Messages	25
	Warning Messages	26
<b>4.</b>	<b>SunHSI/S Utilities and SunVTS Diagnostic Testing</b>	<b>27</b>
	Software Port Names	28
	The <code>hsi_init</code> Command	29

The <code>hsi_loop</code> Command	34
Test Type Options	36
Test Option 1 — Internal Test	36
Test Option 2 — Test Using Loopback Plugs	36
Test Option 3 — Test Using Local or Remote Modem Loopback	36
Test Option 4 — Use Previously Set Mode	37
<code>hsi_loop</code> Output	38
The <code>hsi_stat</code> Command	39
SunVTS Diagnostic Testing	43
<b>A. Building a Synchronous Null Modem Cable and an X.21-to-RS-449 Converter</b>	<b>45</b>
Null Modem Cable Requirements	45
Configuring Internal or External Clocking	46
Building the Null Modem Cable	46
RS-449 Null Modem Cable	47
X.21-to-RS-449 Converter	50
RS-449 Implementation Example	51
RS-449 Pin Assignments and Descriptions	52
96-Pin Connector Signal and Pin Assignments	53
<b>B. <code>hsi_init</code> Options for T1 Compatibility</b>	<b>57</b>
Inverted Settings	58
Data Signal Inversion	58
Clock Signal Inversion	59
<b>C. <code>hsi_init</code> Options for Operating Modes</b>	<b>61</b>
HDLC Mode	61
IBM (SDLC) Mode	62
IBM Full-Duplex Mode	62

IBM Half-Duplex Mode 62

IBM Multi-Point Mode 63

**D. Hardware Functional Description 65**

Integrated Serial Communications Controller (ISCC) 65

Dual-Ported 32 Kbyte Random Access Memory 66

System Clock 66

Bit Latch Controller 66

EPROM 67

Status Buffers 67

**E. Software Functional Description 69**

Initialization 69

External Interfaces 70

IOCTLs 72

Interrupts 74

Packet Transmission and Reception 75

**Index 77**



# Figures

---

FIGURE 1-1	SunHSI/S Adapter	1
FIGURE 1-2	Hardware Block Diagram	3
FIGURE 2-1	The SunHSI/S Patch Panel	8
FIGURE 2-2	Attaching the Wall Mounting Bracket	9
FIGURE 2-3	Attaching the Rack Mounting Bracket	10
FIGURE 2-4	Attaching the SPARCserver 690MP Mounting Bracket	11
FIGURE 2-5	Installing the Patch Panel Assembly in a SPARCserver 690MP	12
FIGURE 2-6	Connecting the SunHSI/S Adapter to the Patch Panel	13
FIGURE 3-1	SunHSI/S 3.0 Software Directories and Files	18
FIGURE A-1	Null Modem Cable (Both Sun Systems Supply Clocking)	48
FIGURE A-2	Null Modem Cable (Sun System Supplies Clocking for Both Sides)	49
FIGURE A-3	X.21-to-RS-449 Converter	50
FIGURE A-4	RS-449 Implementation Example	51
FIGURE E-1	SunHSI/S 3.0 Driver Software Interface	71



# Tables

---

TABLE 3-1	SunHSI/S 3.0 CD-ROM Files and Directories	20
TABLE 4-1	SunHSI/S Utilities	27
TABLE 4-2	SunHSI/S Hardware and Software Port Numbers	28
TABLE 4-3	hsi_init Parameter Values	30
TABLE 4-4	hsi_init One-Word Commands	33
TABLE 4-5	hsi_loop Options	35
TABLE 4-6	hsi_stat Statistic Descriptions	40
TABLE 4-7	SunVTS Documentation	43
TABLE A-1	RS-449 Signals	47
TABLE A-2	X.21 Signals	47
TABLE A-3	Functional Description of RS-449 Interface Signals	52
TABLE A-4	96-Pin Connector Pin and Signal Assignments	53
TABLE D-1	Status Buffer Bit Assignments	67
TABLE E-1	IOCTL Parameters for the SunHSI/S Driver	72
TABLE E-2	Hardware Interrupts	74



# Preface

---

This document provides information on how to install, configure, and use the SunHSI/S™ adapter, which is a high-speed serial interface SBus adapter.

These instructions are designed for a system administrator with experience installing similar hardware and software.

---

## Using UNIX Commands

This document may not contain information on basic UNIX® commands and procedures such as shutting down the system, booting the system, and configuring devices.

See one or more of the following for this information:

- *Solaris Handbook for Sun Peripherals*
- AnswerBook™ online documentation for the Solaris™ software environment
- Other software documentation that you received with your system

---

# Typographic Conventions

TABLE P-1 Typographic Conventions

Typeface	Meaning	Examples
AaBbCc123	The names of commands, files, and directories; on-screen computer output	Edit your <code>.login</code> file. Use <code>ls -a</code> to list all files. % You have mail.
<b>AaBbCc123</b>	What you type, when contrasted with on-screen computer output	% <b>su</b> Password:
<i>AaBbCc123</i>	Book titles, new words or terms, words to be emphasized	Read Chapter 6 in the <i>User's Guide</i> . These are called <i>class</i> options. You <i>must</i> be superuser to do this.
	Command-line variable; replace with a real name or value	To delete a file, type <code>rm filename</code> .

---

# Shell Prompts

TABLE P-2 Shell Prompts

Shell	Prompt
C shell	<i>machine_name%</i>
C shell superuser	<i>machine_name#</i>
Bourne shell and Korn shell	\$
Bourne shell and Korn shell superuser	#

---

## Related Documentation

You may need to refer to the following manuals when installing the SunHSI/S adapter into your system.

**TABLE P-3** Related Documentation

<b>Manual</b>	<b>Operating Environment</b>
Your system installation and service manuals	All supported versions
<i>Solaris Handbook for SMCC Peripherals</i>	Solaris 2.5.1, 2.6
<i>Solaris Handbook for Sun Peripherals</i>	Solaris 7
<i>SunVTS User's Guide</i>	All supported versions
<i>SunVTS Test Reference Manual</i>	All supported versions

---

## Sun Documentation on the Web

The `docs.sun.comsm` web site enables you to access Sun technical documentation on the Web. You can browse the `docs.sun.com` archive or search for a specific book title or subject at:

`http://docs.sun.com`

---

## Sun Welcomes Your Comments

We are interested in improving our documentation and welcome your comments and suggestions. You can email your comments to us at:

`docfeedback@sun.com`

Please include the part number of your document in the subject line of your email.

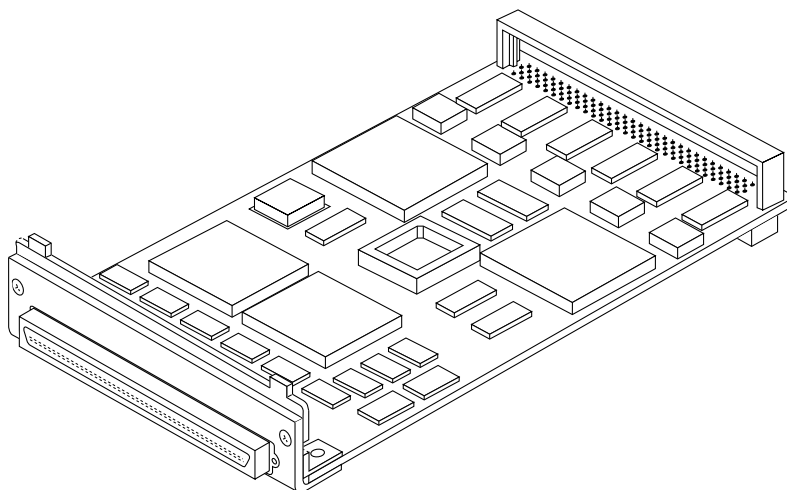




# Overview

---

The SunHSI/S adapter is a four-port high-speed serial communications SBus adapter. The SunHSI/S adapter comes with a 96-pin cable, driver software, patch panel, and documentation. The SunHSI/S adapter occupies one slot of an SBus system.



**FIGURE 1-1** SunHSI/S Adapter

The SunHSI/S adapter supports protocols such as SNA 3270, SNA peer-to-peer, X.25, and PPP.

When used with DSU/CSU equipment (available from third parties) the SunHSI/S adapter can communicate over 1.544 Mbps T1 links or 2.048 Mbps CEPT (E1) links.

The SunHSI/S adapter is connected to the SBus through a standard integral SBus connector. A 96-pin cable is used connect the SunHSI/S adapter to four standard RS-449 connectors on the patch panel.

---

# Hardware Description

The SunHSI/S adapter consists of two Zilog Integrated Serial Communications Controllers (ISCC), 32 KBytes of dual-ported RAM, 32 KBytes of EPROM, 3 bit-addressable latches, and various control and interface logic. A hardware block diagram is shown in FIGURE 1-2.

All interface signals appear at the 96-pin connector on the rear edge of the SBus adapter. The 96-pin cable adapts the 96-pin connector to four RS-449 connectors on the patch panel. The cable and patch panel are shielded to meet FCC Class B EMI regulations.

Appendix A contains instructions for building a null modem cable and an X.21-to-RS-449 converter. An example of a typical RS-449 implementation and the RS-449 signal/pin assignments are also included.

See Appendix D for a more complete description of the hardware.

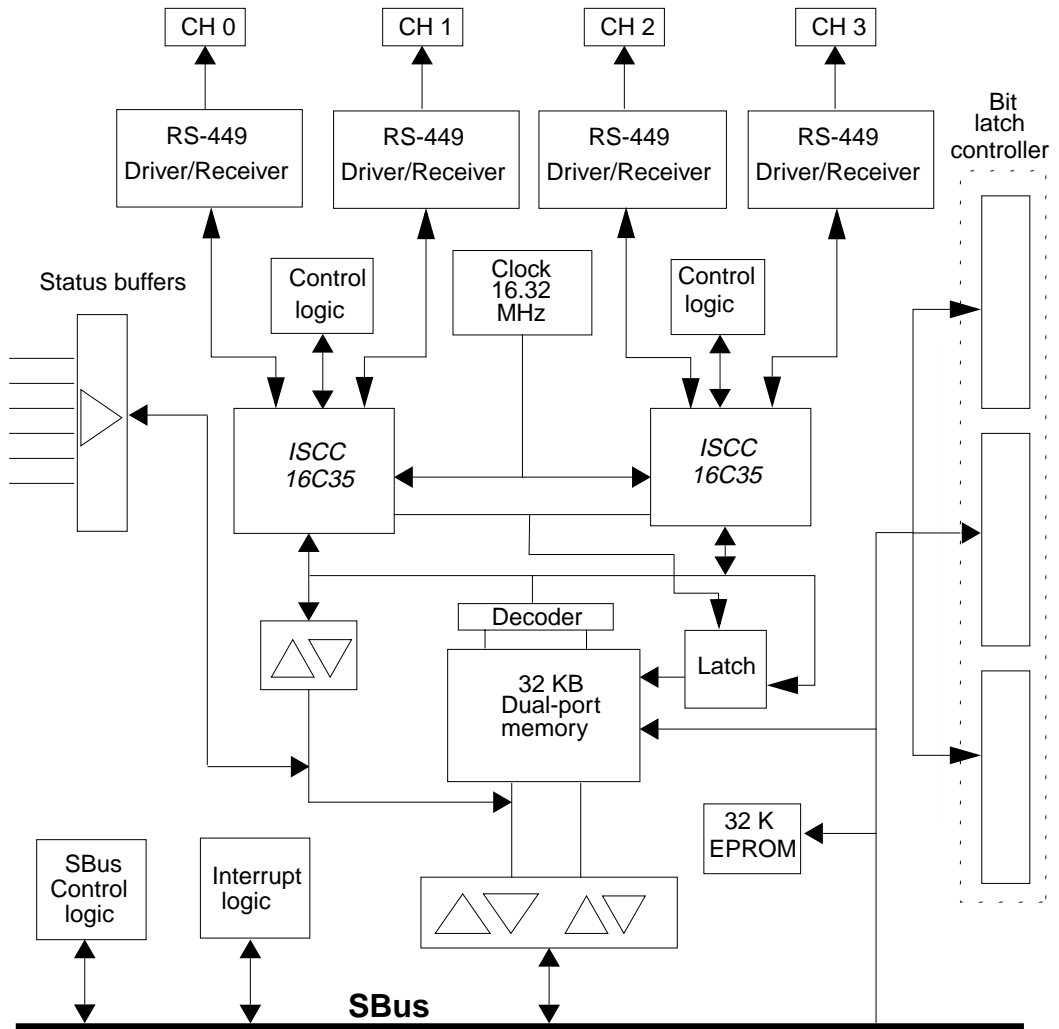


FIGURE 1-2 Hardware Block Diagram

## SunHSI/S Serial Channels

The four serial channels of the SunHSI/S adapter conform to the RS-449 specification for type SR interface circuits. Loopback connectors and test software are provided to verify correct operation of the interface. The SunHSI/S adapter ports are all DTE-type. The optional features for remote and local modem loopback testing as specified by the RS-449 specification are not implemented.

For ease of interface to T1 circuits, each channel can invert its transmitted and received data as well as its transmitted and received clocks (see Appendix B). The inversion is performed under software control. Additionally, all channels can provide their own internal clocks under software control. Internal clocks are adjustable in standard baud rate increments from 300 bps to 2.048 Mbps, under software control. Appendix B also lists the restrictions for using the SunHSI/S adapter with T1 links.

## SunHSI/S-to-SBus Interface

The SunHSI/S-to-SBus interface is a slave interface and can be installed in any vacant SBus slot.

## Aggregate Bandwidth

The aggregate bandwidth of the SunHSI/S adapter is 2.5 Mbps (full-duplex). This implies that you can run one port at T1 (1.536Mbps) or CEPT (2.048 Mbps) speed.

## Power Consumption

The SunHSI/S adapter consumes about 8.0 watts or 1.6 Amps at 5 VDC.

---

# Software Description

The SunHSI/S 3.0 software package includes a network device driver and several utilities to diagnose the functionality of the SunHSI/S device driver. Appendix E contains a functional description of the SunHSI/S software.

## Network Device Driver

The SunHSI/S 3.0 driver provides a streams-based interface to the Solaris operating environment. The streams interface works with other products such as SNA 3270, SNA peer-to-peer, X.25, and PPP protocols.

## Diagnostic Utilities

The SunHSI/S 3.0 software supports bundled Solaris synchronous interface diagnostic utilities (`syncstat`, `syncloop`, and `syncinit`) and also supports an enhanced set of synchronous utilities (`hsi_stat`, `hsi_loop`, and `hsi_init`). These synchronous utilities are described in Chapter 4.

---

# Diagnostics

The SunHSI/S adapter supports a loopback diagnostic command, `hsi_loop`, that is executed as a Solaris command from the workstation where the adapter is installed. Loopback testing allows testing of the path from the processor to the point of the loopback. The point of the loopback can be at the ISCC chip, a loopback plug, or the local/remote modem. See Chapter 4 for further information.

---

# Getting Help

If you have problems installing or using this product after reading this document, call your local service provider and have the following information ready:

- System model and serial numbers
- Solaris release number
- SunHSI/S version number (3.0)
- Type of keyboard
- Number of CPUs
- Number of SunHSI/S adapters

You can display machine and software information needed for help calls by entering the following command:

```
hostname% showrev
```

If you have questions about Sun™ support services or your shipment, visit the SunSolve<sup>SM</sup> website at <http://sunsolve.sun.com/>, or call your authorized service provider.

# Installing the SunHSI/S Adapter and Patch Panel

---

This chapter explains how to install the SunHSI/S adapter in an SBus system and how to install the patch panel.

---

## Tools and Equipment

You will need the following to install the SunHSI/S adapter and the patch panel:

- #1 Phillips screwdriver
- Antistatic wrist strap (provided)

---

## Installing the SunHSI/S Adapter

---

**Note** – Refer to your system installation or service manual for detailed instructions for the following tasks.

---

1. **Power off your system, using the standard shut down procedures described in the *Solaris Handbook for Sun Peripherals* or your system service manual.**  
The *Solaris Handbook for Sun Peripherals* is shipped with the Solaris operating environment software and is available in the on-line AnswerBook documentation.
2. **Open the system unit.**
3. **Attach the adhesive copper strip of the wrist strap to the metal casing of the power supply. Wrap the other end twice around your wrist, with the adhesive side against your skin.**

4. If you are replacing an SBus adapter with the SunHSI/S adapter, remove the old adapter now.
5. Install the SunHSI/S adapter in the selected SBus slot.
6. Detach the wrist strap and close the system unit.

---

## Installing the SunHSI/S Patch Panel

The SunHSI/S patch panel has four RS-449 ports for connecting the SunHSI/S adapter to RS-449 devices as shown in the following illustration.

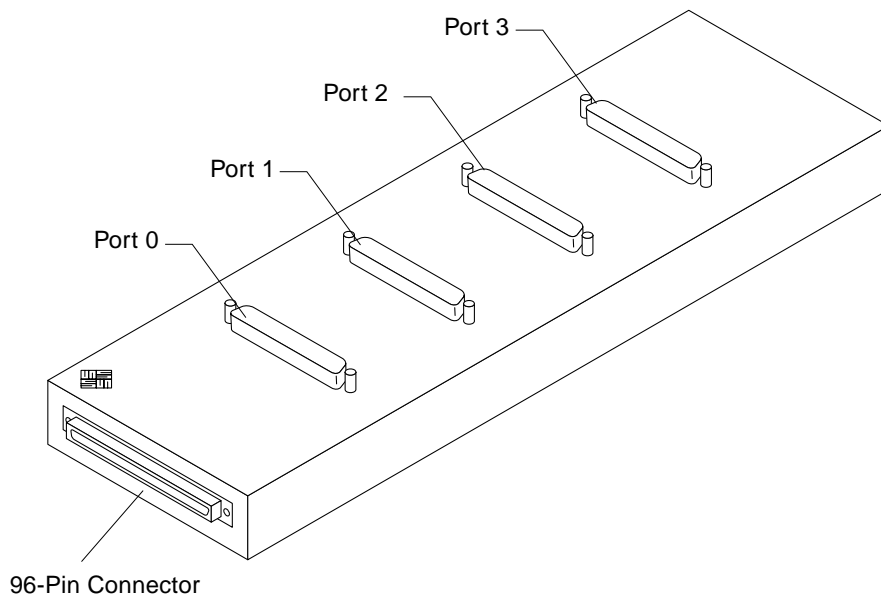


FIGURE 2-1 The SunHSI/S Patch Panel

The patch panel can be installed on a wall, inside a rack, or inside a SPARCserver™ 690MP system.



## ▼ To Mount the Patch Panel on a Wall

1. Remove the screws that secure the four rubber feet to the bottom of the patch panel (FIGURE 2-2.).
2. Secure a wall mounting bracket to each end of the patch panel with the four screws removed in Step 1.
3. Secure the patch panel to a wall with two screws and lockwashers.

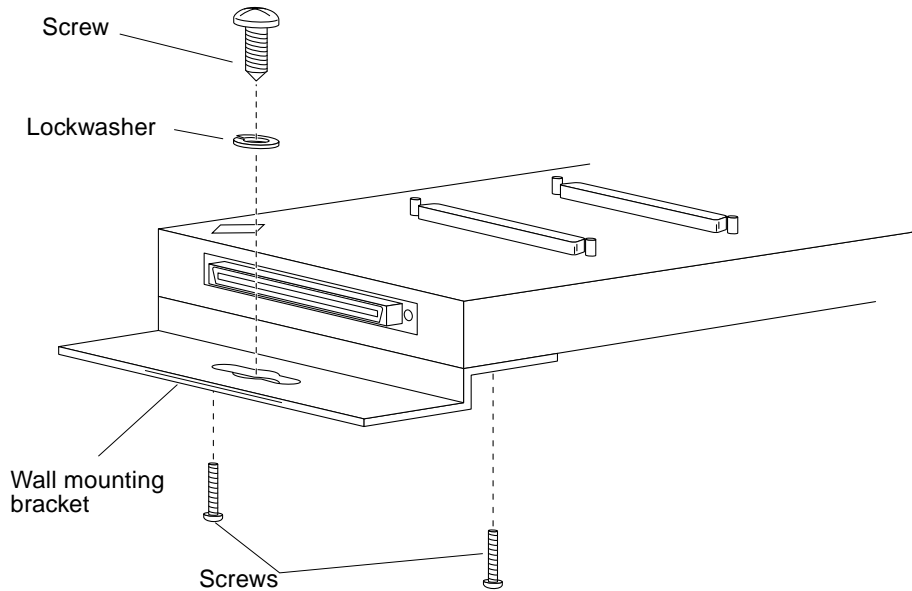


FIGURE 2-2 Attaching the Wall Mounting Bracket

## ▼ To Install the Patch Panel in a Rack

1. Remove the screws that secure the four rubber feet to the bottom of the patch panel (FIGURE 2-3).
2. Secure a mounting bracket to each end of the patch panel using the four screws removed in Step 1.
3. Cut the nylon grommet into four equal lengths.
4. Press the nylon grommet around the edges of the cable openings.
5. Secure the patch panel assembly to the rack RETMA rails with four screws and lockwashers.

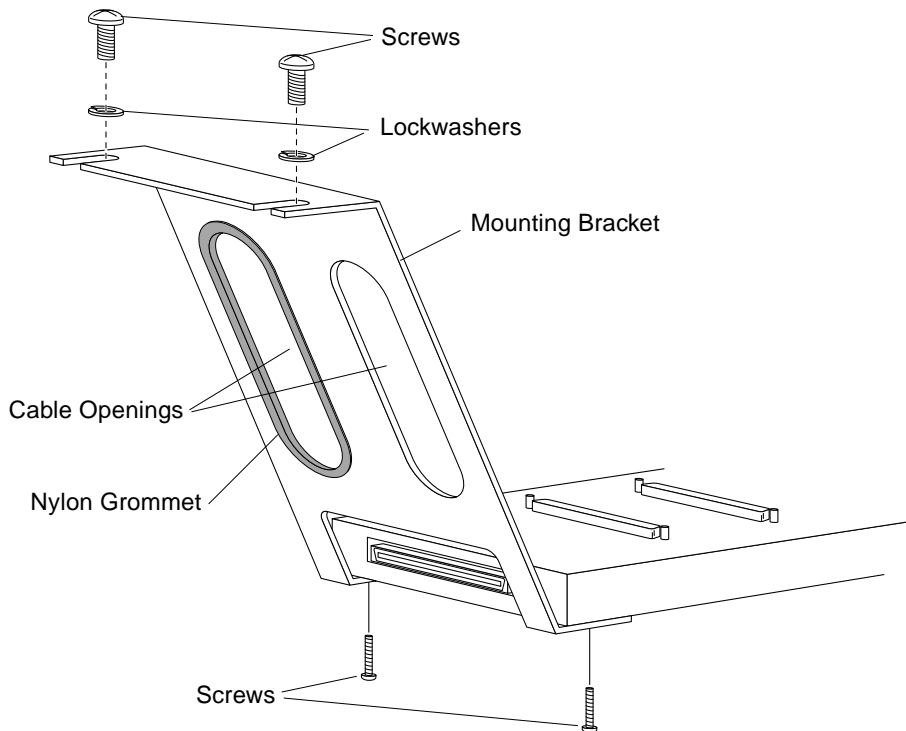


FIGURE 2-3 Attaching the Rack Mounting Bracket

## ▼ To Install the Patch Panel in a SPARCserver 690MP

1. Remove the screws that secure the four rubber feet to the bottom of the patch panel (FIGURE 2-4).
2. Secure a mounting bracket to each end of the patch panel with the four screws removed in Step 1.
3. Loosen the two captive screws at the bottom of the right-side panel. Lift the panel off of the top clips and set it aside.
4. Secure the patch panel assembly to the RETMA rails with four screws and lockwashers at the location(s) shown in FIGURE 2-4 and FIGURE 2-5.
5. Connect one end of each 37-pin RS-449 cable to the patch panel.
6. Connect the other end of the 37-pin RS-449 cable as appropriate for your configuration.
7. If you need to install a second patch, repeat Step 1 through Step 6.

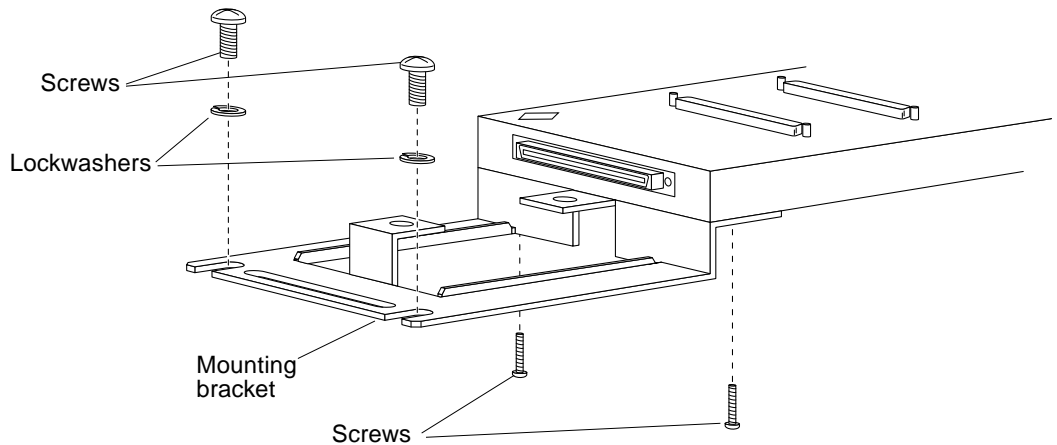
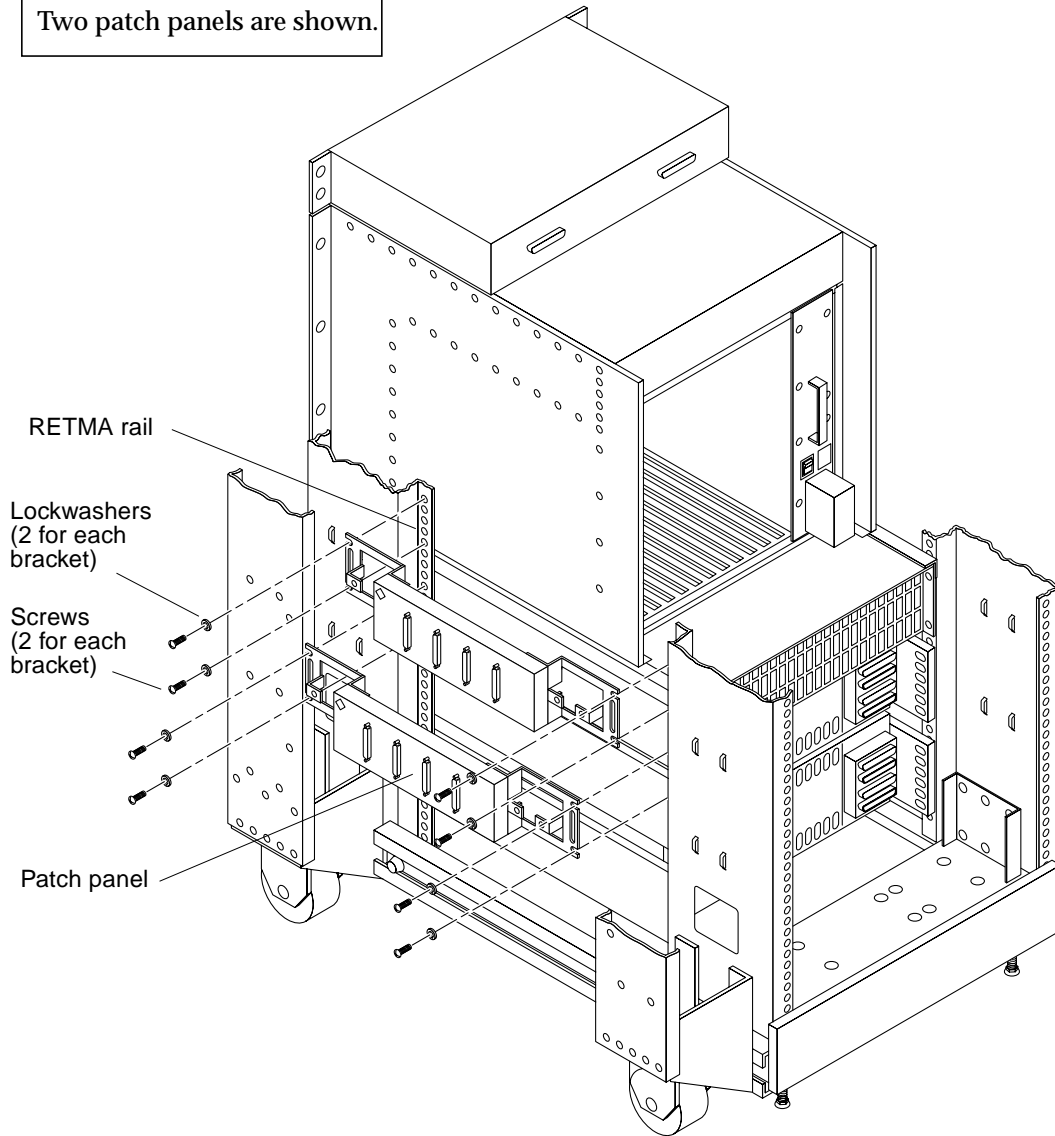


FIGURE 2-4 Attaching the SPARCserver 690MP Mounting Bracket

**Note**  
Two patch panels are shown.



**FIGURE 2-5** Installing the Patch Panel Assembly in a SPARCserver 690MP

# Connecting the 96-Pin Cable

The 96-pin shielded cable is used to connect the patch panel to the SunHSI/S adapter (FIGURE 2-6).

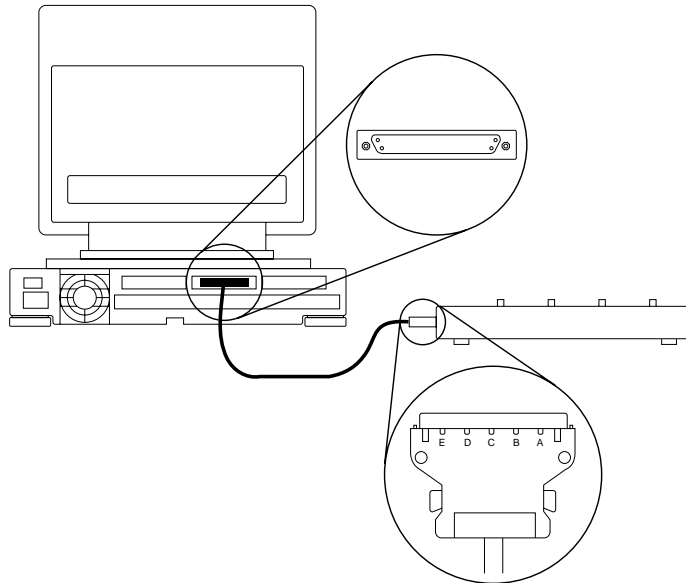
## ▼ To Connect the Patch Panel to the SunHSI/S Adapter With the 96-pin Cable

---

**Note** – When configuring your SunHSI/S cabling, make the connection to your SunHSI/S adapter *after* you have made the connection to the other end.

---

1. Connect one end of the 96-pin cable to the patch panel.
2. Connect the other end of the 96-pin cable to the SunHSI/S adapter.
3. Make sure that the locking mechanisms on each end of the 96-pin connector click closed.



**FIGURE 2-6** Connecting the SunHSI/S Adapter to the Patch Panel

## SunHSI/S Cabling

The SunHSI/S adapter provides connectivity to four RS-449 devices through four DB-37 female connectors (DTE) on a patch panel.

---

**Note** – Always use twisted-pair RS-449 cables with your SunHSI/S adapter.

---

## RS-232 to RS-449 Connections

In order to connect RS-232 devices to the SunHSI/S adapter, you need to install an externally powered RS-449 to RS-232 interface converter to each DB-37 connector on which you intend to connect an RS-232 device. A converter is necessary because of incompatibilities between RS-232 and RS-449 signal levels.

---

**Note** – Use only externally powered RS-449 devices with the SunHSI/S adapter.

---

To obtain an externally powered RS-232 to RS-449 interface converter, contact:

Black Box Corporation  
<http://www.blackbox.com>

---

## Verifying the Installation and Rebooting the System

After you have installed the adapter, but *before* you boot your system, perform the following tasks to verify the installation. Refer to the *Solaris Handbook for Sun Peripherals*, or your Solaris documentation, for more detailed information about the instructions in this procedure.



---

**Caution** – The system cover must be secured before powering on the system. Failure to take this precaution may result in personal injury and damage to the system.

---

## ▼ To Power On Your System

### 1. Make sure all cable connections are secure.

### 2. Power on your system.

Refer to your system installation or service manual for the specific instructions for your system.

When the power is turned on, a banner similar to the following will be displayed on the screen:

```
SPARCstation X. Type-x Keyboard present
ROM Rev. X.XY,X.X MB memory installed, Serial #XXX
Ethernet address X:X:YY:Z:A:BB. Host ID: 01010101.
```

After displaying the banner, the system will then run the power-on self tests. While the power-on tests are running, a number of system messages may be displayed on the screen.

### 3. Observe the power-on self tests.

- If you do not see any error messages, and your system begins the Solaris boot process, press the Stop-A keys to interrupt the boot process and display the OpenBoot™ prompt. Skip to Step 4.
- If you see error messages, and your system cannot begin the boot process, the power-on test may have failed and you need to perform the following sub-steps:

#### a. Shut down and power off the system.

---

**Note** – Refer to your system installation or service manual for the specific instructions for your system.

---

#### b. Verify that the 96-pin cable between the SunHSI/S adapter and the patch panel is installed correctly.

Depending upon the type of malfunction, an error message should have been displayed on the screen when you powered on your system.

#### c. Power on your system again.

If the power-on self tests pass correctly, continue with Step 4. Otherwise, contact your Sun service representative for additional information (see “Getting Help” on page 6).

#### 4. Use the `show-devs` command to list the system devices.

You should see a line in the list of devices, similar to the example below, specific to the SunHSI/S adapter.

---

**Note** – The HSI device entry will vary depending on the system type.

---

```
ok show-devs
...
/iommu@f,e0000000/sbus@f,e0001000/HSI@2,2000
...
```

---

**Note** – If you do not see the device listed, check that the adapter is properly seated and, if necessary, reinstall the adapter.

---

#### 5. Reboot the system.

Refer to the *Solaris on Sun Peripherals* for more information.

After the system reboots, install the SunHSI/S 3.0 software as described in Chapter 3.



## Installing the SunHSI/S 3.0 Software

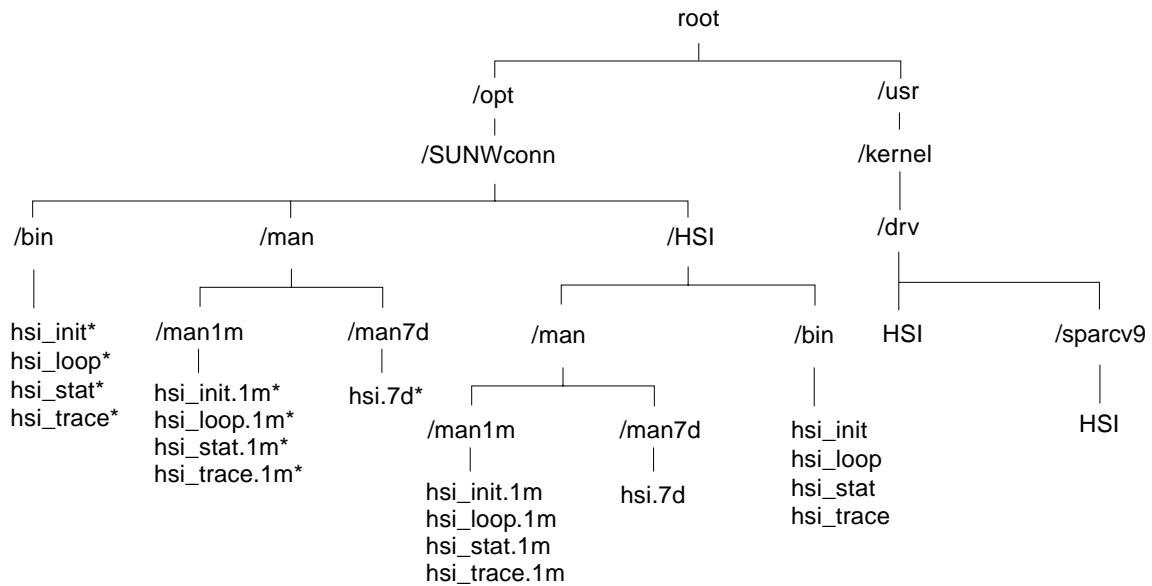
---

This chapter describes how to install the SunHSI/S 3.0 software by:

- Checking for previous versions of the SunHSI/S software
- Mounting the SunHSI/S 3.0 CD-ROM
- Using the `pkgadd(1m)` utility to install the software packages
- Unmounting the CD-ROM once the installation is complete

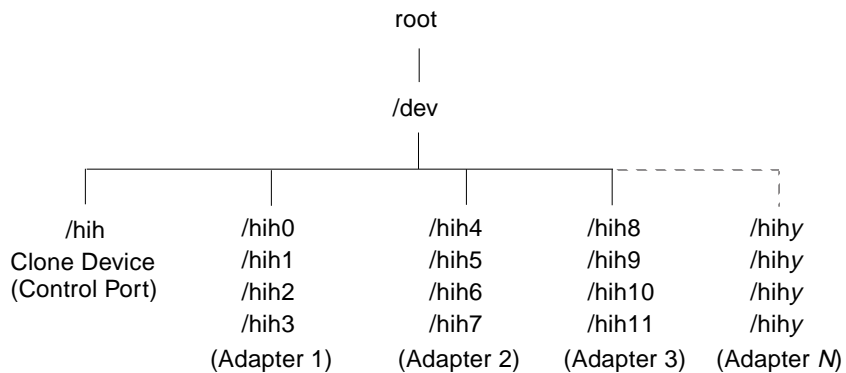
When you have completed the installation of your software, and the `pkgadd` utility has run the post-installation script, you will have created the software directories and files illustrated FIGURE 3-1.

### SunHSI/S 3.0 software files



\* Signifies a symbolic link.

### Devices created by the post-install script



**FIGURE 3-1** SunHSI/S 3.0 Software Directories and Files



---

**Caution** – Do not overwrite any existing SunHSI/S software packages. If you install the SunHSI/S 3.0 software packages over existing SunHSI/S software packages, you will have two instances of the software packages. This may cause problems when installing or backing out of software patches.

---

## ▼ To Remove Older Versions of the SunHSI/S Software

Before installing the SunHSI/S 3.0 software on your system, check your system to see if previous versions of the SunHSI/S software is installed. If older SunHSI/S software exists (before version 3.0), you must remove this software before installing the new SunHSI/S 3.0 software.

1. Use the `pkginfo` command to check the system for an older SunHSI/S software package:

```
# /usr/bin/pkginfo | grep SUNWhsis
system    SUNWhsis    HSI/S Driver/Utilities 2.0 v1.x
```

- If you do not find any SunHSI/S packages, skip to the next section, “To Mount the SunHSI/S 3.0 CD-ROM” on page 20, to continue with the software installation.
  - If you do find a SunHSI/S package you must remove it as described in Step 2.
2. As superuser (root), use the `pkgrm` command to remove the existing SunHSI/S software package:

```
# /usr/sbin/pkgrm SUNWhsis
```

## ▼ To Mount the SunHSI/S 3.0 CD-ROM

1. **Become superuser.**
2. **Insert the SunHSI/S 3.0 CD into a CD-ROM drive that is connected to your system.**
  - If your system is running Volume Manager, it should automatically mount the CD to the `/cdrom/sunhsis_3_0` directory.
  - If your system is not running Volume Manager, mount the CD as follows:

```
# mkdir -p /cdrom/sunhsis_3_0
# mount -F hsfs -o ro /dev/dsk/c0t6d0s2 /cdrom/sunhsis_3_0
```

### Files and Directories on the CD-ROM

You will see the following files and directories in the `/cdrom/sunhsis_3_0` directory.

**TABLE 3-1** SunHSI/S 3.0 CD-ROM Files and Directories

File or Directory	Contents
Copyright	U.S. Copyright file
FR_Copyright	French Copyright file
Product	Contains these SunHSI/S 3.0 software packages: <ul style="list-style-type: none"><li>• SUNWhsis—SunHSI/S 3.0 driver software</li><li>• SUNWhsism—SunHSI/S 3.0 man pages</li><li>• SUNWhsisu—SunHSI/S 3.0 software utilities</li></ul>

## ▼ To Install the SunHSI/S Software

1. As superuser, type the following to install the SunHSI/S 3.0 software packages from the CD:

```
# pkgadd -d /cdrom/sunhsis_3_0/Product SUNWhsis SUNWhsism SUNWhsisu
```

2. When prompted, answer `y` to permit `pkgadd` to launch the post-installation script:

```
This package contains scripts which will be executed with superuser
permission during the process of installing this package.
```

```
Do you want to continue with the installation of this package
[y,n,?] y
```

`pkgadd` will continue to install the software and run the post-installation script to create the `/dev/hih` devices. When the installation is complete, you will see messages saying that the software has been installed successfully.

3. Use the `pkginfo` command to verify that the software has been installed.

```
# pkginfo SUNWhsis
system    SUNWhsis    HSI/S Driver for SBus
```

4. Unmount and eject the SunHSI/S 3.0 CD-ROM.

```
# cd /
# umount /cdrom/sunhsis_3_0
# eject cdrom
```

---

# Before Operating the SunHSI/S Adapter

## Changing the Cabling or Equipment

If you make any cabling or equipment changes on a port (for example, changing modems), you must reset the port with the `hsi_init` reset command (*N* represents the SunHSI/S port number):

```
# hsi_init hihN reset
```

After all the changes have been made, re-initialize the port with the `hsi_init` command. The `hsi_init` command is described in Chapter 4.

## Checking the MTU and MRU Sizes

Before operating an SunHSI/S link, make sure that the MTU and MRU sizes specified on each side of the link are the same on both sides. You can use the `hsi_init` command to check these sizes (replace *N* with the port number of the link you are testing):

```
# hsi_init hihN
```

---

**Note** – Checking the MTU/MRU sizes is especially important if you use SunHSI/S with a different type of hardware other than SunHSI/S (such as the on-board serial port or third-party equipment).

---

---

# Viewing the Man Pages

The following man pages are included with the SunHSI/S software:

- `hsi(7d)`
- `hsi_init(1m)`
- `hsi_loop(1m)`
- `hsi_stat(1m)`
- `hsi_trace(1m)`

If you cannot view these man pages, you need to add the `/opt/SUNWconn/man/` directory to your `MANPATH` environment variable. Depending on the UNIX shell you are using, this variable may be defined in one of a number of startup files.

## ▼ To View the Man Pages in the C Shell Environment

1. **Examine your `$HOME/.login` and `$HOME/.cshrc` files to locate the `MANPATH` variable.**
2. **Using a text editor, add the following line to the end of the file containing the `MANPATH` variable.**

```
setenv MANPATH "/opt/SUNWconn/man/:$MANPATH"
```

If neither of these files contain this variable, add the following line to the end of one of the files, or contact your system administrator for assistance.

```
setenv MANPATH "/opt/SUNWconn/man/"
```

3. **Use the `source` command on the file you edited to make the changes effective in your current window.**

For example, if you added the `MANPATH` line to the `.login` file, you would type:

```
% source $HOME/.login
```

---

**Note** – If you log out and then back into your system, you will update the `MANPATH` variable in all command windows and shells.

---

## ▼ To View the Man Pages in Bourne or Korn Shell Environments

1. Using a text editor, add these two lines to the end of the `$HOME/.profile` file.

```
MANPATH=/opt/SUNWconn/man:$MANPATH
export MANPATH
```

If this file did not already contain this variable, add the following two lines to the end of the file, or contact your system administrator for assistance.

```
MANPATH=/opt/SUNWconn/man
export MANPATH
```

2. Make the changes effective in your current window.

```
$ . $HOME/.profile
```

---

**Note** – If you log out and then back into to your system, you will update the `MANPATH` variable in all command windows and shells.

---

---

## Removing the SunHSI/S 3.0 Software

Use the `pkgrm` command to remove the SunHSI/S 3.0 software from your system.

- As superuser (root), use the `pkgrm` command to remove the SunHSI/S 3.0 software packages:

```
# /usr/sbin/pkgrm SUNWhsis SUNWhsism SUNWhsisu
```



---

# Console Messages

This section lists line error console messages that may be displayed in your console window and a brief description of the error messages.

---

**Note** –  $N$  represents the port number.

---

## Informational Messages

```
hihN up and running at <baud rate>, mode=<mode> txc=<txc>
rxc=<rxc>
```

The SunHSI/S driver just brought up port  $hihN$  with the parameters shown. The baud rate shown in this message may be different from the externally set baud rate when external clocking is used.

```
hihN: reset
```

The SunHSI/S port  $N$  is reset.

## Error Messages

```
ERROR: hih_init: pll and !NRZI.
```

Using the `hsi_init` command, the `txc` parameter was set to `pll`. However, the `nrzi` parameter was not set to `yes`. Setting the transmit clock source to `pll` requires NRZI data encoding.

```
ERROR: hih_init: pll or baud and baud = 0
```

The baud rate specified was 0 and internal clocking was set.

```
hihN: Bad PPA = N
```

SunHSI/S driver received a `DL_ATTACH_REQ`, which has an out-of-range PPA number  $N$ , from upper layers.

```
hihN: port N not installed
```

The SunHSI/S port  $N$ , which is referenced by the PPA number in a received `DL_ATTACH_REQ` message, is not installed to the system.

hihN: out of STREAMS mblocks

Running out of streams mblocks for SunHSI/S port *N*.

hihN: xmit hung

Transmission hung on SunHSI/S port *N*. This usually happens because of cabling problems or due to missing clocks from the CSU/DSU or modem.

hihN: <hih\_rxsoft> no buffers - rxbad

Running out of streams mblocks for SunHSI/S port *N* in `hih_rxsoft()` routine.

## Warning Messages

WARNING: hih\_init: changed baudrate from 100000 to 99512.

The baud rate specified was rounded to a value the SunHSI/S hardware can support.

## SunHSI/S Utilities and SunVTS Diagnostic Testing

---

This chapter describes the utilities associated with SunHSI/S interface driver, and it provides information about the SunVTS™ diagnostic software.

The SunHSI/S software ships with its own version of the serial port utilities. In general, the SunHSI/S utilities provide a superset of the features described in this chapter.

**TABLE 4-1** SunHSI/S Utilities

SunHSI/S Command	Description
<code>hsi_init</code>	Initializes serial ports and allows you to modify and view driver-level parameters.
<code>hsi_loop</code>	Performs loopback testing to check the integrity of your data transmission path.
<code>hsi_stat</code>	Monitors serial port activity, on a “snapshot” or repeating-interval basis.

---

**Note** – You must be superuser (root) in order to run the `hsi_init`, `hsi_stat` or `hsi_loop` utilities.

---

---

## Software Port Names

The port naming conventions are used by initialization and serial port diagnostic commands. Software port names for SunHSI/S ports are of the form `hihN`, where  $N$  is a number in a range starting with 0 and ending at one fewer than the number of serial ports on your machine. For example, on a system with one SunHSI/S adapter installed, the serial ports are named `hih0`, `hih1`, `hih2`, and `hih3`.

SunHSI/S port names have the format: `hihy`, where  $y$  represents the port number, 0, 1, 2, or 3 for the first SunHSI/S adapter and 4, 5, 6, or 7 for the second, and so on. For example, `hih1` is the name for port 1 (the second port) on the first SunHSI/S adapter. The name `hih4` specifies port 0 on the second SunHSI/S adapter.

The software port numbers, as used in SunHSI/S port names, have the following relation to the hardware port numbers used on each adapter:

TABLE 4-2 SunHSI/S Hardware and Software Port Numbers

SunHSI/S adapter Number	Hardware Port Number	Software Port Number
1	0	0
	1	1
	2	2
	3	3
2	0	4
	1	5
	2	6
	3	7
3	0	8
	1	9
	2	10
	3	11
$N$	0	$(N - 1) \times 4 + 1$
	1	$(N - 1) \times 4 + 2$
	2	$(N - 1) \times 4 + 3$
	3	$(N - 1) \times 4 + 4$

---

# The `hsi_init` Command

The `hsi_init` command allows you to display and modify some of the hardware operating modes common to high speed serial lines. These features make the `hsi_init` command valuable when troubleshooting and repairing problematic serial link lines.

Other applications also use the `hsi_init` command. For example, some applications use the `hsi_init` command to initialize serial ports, and the `hsi_loop` command (described in “The `hsi_loop` Command” on page 34) uses the command in a number of loopback tests.

When using `hsi_init` at the command line, the first argument required by the command is always the port name of the link being displayed or modified (for example, `hih0`). With no further arguments, `hsi_init` displays the parameter values as presently set on the selected link:

```
# hsi_init hih0
port=hih0 speed=1536000, mode=fdx, loopback=no, nrzi=no, mtu=1600, mru=1600,
txc=txc, rxc=rxc, txd=txd, rxd=rx, signal=no.
```

---

**Note** – You can use the `hsi_init` command to display the parameter values associated with a serial line, even if the serial device has been initialized through another command. However, you should not use `hsi_init` to modify any parameters on lines that were not initialized by `hsi_init`.

---

You can set all of the `hsi_init` parameters as shown in the usage statement below.

```
# hsi_init
Usage: hsi_init ifname \
    [baudrate] [loopback=[no|yes|echo]] [nrzi=[yes|no]] \
    [txc=[txc|-txc|baud|rx|pll]] [rxc=[rxc|-rxc|baud|txc]] \
    [mode=[fdx|ibm-fdx|ibm-hdx|ibm-mpt]] [signal=[yes|no]] \
    [external|sender|stop|reset] \
    [mtu=<mtu_size>] [mru=<mru_size>] \
    [txd=[txd|-txd]] [rxd=[rxd|-rxd]]
```

To set these parameters, use the syntax `hsi_init portname keyword=value`. For example, to set the maximum transmission unit (mtu) parameter of port hih2 to 1000 bytes, you would type:

```
# hsi_init hih2 mtu=1000
```

TABLE 4-3 displays the possible values for each of these `hsi_init` parameters, and lists the default values as initialized by the SunHSI/S driver for each port. The parameter values are described in greater detail after this table.

**TABLE 4-3** `hsi_init` Parameter Values

Parameter	Default Value	Possible Values
speed	1536000 bps	The line speed can be set from 0 to 2048000 bps.
loopback	no	Can be set to <code>yes</code> , <code>no</code> , and <code>echo</code> . Useful when used with the <code>hsi_loop</code> command.
nrzi	no	Can be set to <code>yes</code> or <code>no</code> , depending on whether the port uses NRZI data encoding.
txc	txc	Sets the port transmit clocking signal to <code>txc</code> , <code>baud</code> , <code>rxr</code> , <code>pll</code> , or <code>-txc</code> .
rxr	rxr	Sets the port receive clocking signal to <code>txc</code> , <code>baud</code> , <code>rxr</code> , or <code>-rxr</code> .
mode	fdx	Sets the network mode to <code>fdx</code> , <code>ibm-fdx</code> , <code>ibm-hdx</code> , or <code>ibm-mpt</code> .
signal	no	Can be set to <code>yes</code> or <code>no</code> . When set to <code>yes</code> , the modem signal (RTS and CTS) state changes are reported by the driver to the application.
mtu	1600 bytes	The maximum transmission unit can be set from 1 to 1600 bytes.
mru	1600 bytes	The maximum receive unit can be set from 1 to 1600 bytes.
txd	txd	The transmit data signal can be inverted ( <code>-txd</code> ) to accommodate certain T1 or CEPT transmission equipment.
rxr	rxr	The receive data signal can be inverted ( <code>-rxr</code> ) to accommodate certain T1 or CEPT transmission equipment.

**Note** – See Appendix B for more information about inverting the `txd`, `rxr`, `txc` and `rxr` options to accommodate the requirements of T1 or CEPT transmission equipment. Also, see Appendix C for more information about the `mode` options.

## speed

The `speed` parameter sets the line speed, or baud rate, of the serial line in bits per second. You can set this parameter to be from 0 to 2048000 bps.

In most situations, the actual line speed is determined by the modems in use, not by the Sun hardware, so the speed set by `hsi_init` is used only for compiling performance statistics for the `hsi_stat` command (see “The `hsi_stat` Command” on page 39). The `speed` parameter *is* significant when you are using the internal (workstation or server) baud generator to generate clocking. You invoke the internal baud generator when you use the `txc=baud` or `rxs=baud` settings with the `hsi_init` command (these parameters are described below).

---

**Note** – When you use `hsi_init` to specify a very high speed, and the `txc` or `rxs` parameters are set to `baud`, the actual speed (as reported by `hsi_stat` or another monitoring tool) can differ from the speed you specify, because the speed is rounded to the nearest integral multiple of the baud-rate generator clocking frequency. For example, after setting the speed parameter to 64000 bits per second, you may see `hsi_stat` report a line speed of, for example, 63750 bits per second.

---

## loopback

Sets and reports the internal loopback state of the serial chip. Setting a link to an internal loopback state (`loopback=yes`) is useful for testing serial ports that are not attached to external loopback equipment.

After testing a port, you can disable the internal loopback state on the port by setting the parameter to `no`:

```
# hsi_init portname loopback=no
```

Selecting `loopback=echo` will also set internal loopback mode, but incoming received data will still be visible.

This parameter is set to `loopback=yes` transparently by the `hsi_loop` command when you run the `hsi_loop -t 1 portname` test. (See “The `hsi_loop` Command” on page 34 for more information.)

## nrzi

Sets the port to operate with NRZI (Non-Return to Zero, Inverted) data encoding. This parameter can be `yes` for NRZI encoding, or `no` for NRZ encoding.

NRZ data encoding maintains a constant voltage level when data is present and does not return to a zero voltage until data is absent. The data is decoded as an absolute value based on the voltage level, which is 1 when data is present and 0 when data is absent.

NRZI data encoding does a voltage transition when data is absent (voltage level 0), and it does not do a voltage transition (no return to 0) when data is present (voltage level 0). With NRZI, the data is decoded using relational decoding.

#### `txc`

Sets the origin of the clocking for the transmitted data. For transmitted data, you can set the clock origin to `txc`, specifying incoming transmit clock (TxCI signal); `-txc`, specifying inverted incoming transmit clock; `rxc`, specifying incoming receive clock (RxC signal); `baud`, specifying the internal (workstation) baud rate generator; or `pll`, for phased lock loop.

The default for SunHSI/S ports is `txc=txc`. When `txc=baud`, the `speed` argument of the `hsi_init` command, not the modem clocking, controls the data rate. To accommodate the requirements of T1 or CEPT transmission equipment, this signal can be inverted (`txc=-txc`). See Appendix B for more information.

#### `rxc`

Sets the origin of the clocking for the received data. You can set the clock origin to `rxc`, specifying incoming receive clock (RxC signal); `-rxc`, specifying inverted incoming receive clock; `txc`, specifying incoming transmit clock (TxCI signal); or `baud`, specifying the internal (workstation) baud rate generator.

The default for SunHSI/S ports is `rxc=rxc`. To accommodate the requirements of T1 or CEPT transmission equipment, this signal can be inverted (`rxc=-rxc`). See Appendix B for more information.

---

**Note** – While `rxc=baud` is supported on most Sun serial port options, it is useful only in conjunction with internal loopback mode (`loopback=yes`) or where the Sun machine is supplying clocking for one or both sides of a link.

---

#### `mode`

The `mode` parameter sets the operating mode of the serial link. The two main operating modes used by the SunHSI/S software are high-level data link control (HDLC) mode and IBM (SDLC) mode. The values `mode` are: `fdx` (HDLC compatible full-duplex), `ibm-fdx` (IBM compatible full-duplex), `ibm-hdx` (IBM compatible half-duplex), and `ibm-mpt` (IBM compatible multi-point multi-drop). The default mode is `fdx`. See Appendix C for more information about operating modes.

#### `signal`

Controls whether modem signal (RTS and CTS) changes are reported back by the driver to the application. When this parameter is set to `yes`, these changes are reported.



mtu/mru

The `mtu` parameter sets the packet size of the maximum transmission unit, and the `mru` parameter sets the packet size of the maximum receive unit. By adjusting these parameters, you may achieve better performance out of the link. Both of these parameters can be set between 1 and 1600 bytes.

txd/rxd

These flags are used for inverting transmit (`txd`) and receive (`rxd`) data on serial lines. You can switch the polarity of a link by setting these flags to be negative (for example, `-txd` and `-rxd`). When `txd=txd`, the transmit data is not inverted, and when `txd=-txd`, the transmit data is inverted. Likewise, when `rxd=rxd`, the receive data is not inverted, and when `rxd=-rxd`, the data is inverted. These flags are useful when you run SunHSI/S over T1 and CEPT lines (see Appendix B for more information).

You can also use the following set of one-word commands to specify useful combinations of `hsi_init` parameters.

**TABLE 4-4** `hsi_init` One-Word Commands

Command	Equivalent <code>hsi_init</code> Parameters
<code>external</code>	<code>txc=txc rxc=rxc loopback=no</code>
<code>sender</code>	<code>txc=baud rxc=rxc loopback=no</code>
<code>reset</code>	(Resets the port and stops its operation.)
<code>stop</code>	<code>speed=0</code> (Stops the port.)

One clocking arrangement, called sender clocking, is useful for testing because it only requires cabling between the two systems under test, without intervening modems or modem eliminators. To configure sender clocking, use the `sender` command (`txc=baud, rxc=rxc, loopback=no`). This causes the transmitting side to generate a clock signal, which can then be routed to the receive clock on the receiving side. In fact, since each direction of data flow has independent clocking, the two directions can have different speeds, each determined by the `speed` option on the transmitting system.

---

# The `hsi_loop` Command

The `hsi_loop` command performs a loopback test that checks the following components of your communications link:

- Port-driver software layering
- CPU-to-port communication
- Correct operation of the serial port
- Cable from port to modem (or modem equivalent)
- Local and remote modems (or modem equivalents)
- Transmission line

When you invoke `hsi_loop`, it runs the `hsi_init` command to initialize the serial port and send out packets. `hsi_loop` then reads the incoming packets to verify that they were received. It also verifies the packet length and checks that the data is correct.

---

**Note** – Do not run `hsi_loop` on a port that is in use. Because certain `hsi_loop` options put a port in loopback mode, its use can prevent communication with a remote host during the time that `hsi_loop` is sending and receiving packets.

---

To stop an active port (in this example, `hih0`) enter the following `hsi_init` command:

```
# hsi_init hih0 0
port=hih0 speed=0, mode=fdx, loopback=no, nrzi=no, mtu=1600, mru=1600,
txc=txc, rxc=rxr, txd=txd, rxd=rxr, signal=no
```

As an alternative to specifying a speed of 0, you can use the `stop` or `reset` commands (see “`hsi_init` One-Word Commands” on page 33). After you finish with `hsi_loop` testing, you must restart your link to reinitialize your serial port.

A `hsi_loop` command takes the following general form:

```
# hsi_loop [options] portname
```

where *portname* is, for example, `hih2`.

The options for `hsi_loop` are described in TABLE 4-5.

**TABLE 4-5** `hsi_loop` Options

Option	Parameter Name	Description
-c	packet count	Specifies the number of packets used for data transfer. The default is 100.
-l	packet length	Specifies the length of the packet in bytes. The default is 100; the maximum is 1600.
-s	line speed	Bit rate in bits/sec. Applies only if local machine supplies clocking. The default line speed is 9600 bps.
-t	test type	Value in the range 1-4 that specifies the type of test <code>hsi_loop</code> performs (see the following subsection).
-d	hex data byte	Specifies a hexadecimal number as the byte content of each packet. The default is to use random data.
-v	verbose	Sets verbose mode. If data errors occur, the expected and received data are displayed.

Enter all numeric options except `-d` (hex data byte) as decimal numbers (for example, `-t 3`). If you do not provide the test type option, `hsi_loop` prompts for it, as in the following example:

```
# hsi_loop hih1
[ Using /dev/hih1 ]
Enter test type:
1: Internal Test
    (internal data loop, internal clocking)
2: Test using loopback plugs
    (external data loop, internal clocking)
3: Test using local or remote modem loopback
    (external data loop, external clocking)
4: Other, previously set, special mode
> 2
```

An alternative to the preceding command is to specify the test type on the command line:

```
# hsi_loop -t 2 hih1
```

In the preceding command line, note that `hsi_loop` requires a space between the option switch (`-t`) and the number.

# Test Type Options

Listed below are descriptions of the available test type options. You specify test type options with the `-t` option, as described in TABLE 4-5.

## Test Option 1 — Internal Test

This option uses the internal clocking and internal loopback and runs the following `hsi_init` command:

```
hsi_init portname speed loopback=yes txc=baud rxc=baud
```

The test data packets (100 by default) are sent to the specified serial port and looped back internally. You do not need a loopback plug for this option.

## Test Option 2 — Test Using Loopback Plugs

This option uses the internal clocking and requires a loopback plug. Option 2 runs the following `hsi_init` command:

```
hsi_init portname speed loopback=no txc=baud rxc=rx
```

The test data packet will loop between the CPU and serial port through the loopback plug. Before using this option, install an RS-449 loopback plug (part number: 530-1430-01) on the specified port or the 96-pin loopback plug (part number: 370-1381-01) into the back of the SunHSI/S adapter.

## Test Option 3 — Test Using Local or Remote Modem Loopback

This option uses the external clocking set by the modem and runs the following `hsi_init` command:

```
hsi_init portname speed loopback=no txc=txc rxc=rx
```

Testing with a modem in local loopback mode verifies proper operation of the serial port, the external cable, and the local modem. Testing with the modem in remote loopback mode, checks the components just mentioned, as well as verifying the operation of the communications link.

With test type option 3, `hsi_loop` treats both local and remote modem loopback testing the same, since clocking is provided by the modem in both cases. Whether the data is looped back through the local or remote modem depends on how the modems are set up.

If the test fails on remote modem loopback, but succeeds on local modem loopback, carefully check the transmission line and the setup of both modems.

## Test Option 4 — Use Previously Set Mode

There is no automatic `hsi_init` execution with this option. This allows you to use `hsi_init` before running `hsi_loop` to specify clocking and loopback options that are not possible with the other `hsi_loop` test options.

For example, make the local side supply transmit clock, at a desired speed, for example 19200 bps (`hsi_init hihN speed=19200 txc=baud rxc=rxc`), and run `hsi_loop` on the local side with test option 4.

To run the test for the `hsi_loop` command, enter:

```
# hsi_loop -t4 hihN
```

---

**Note** – You cannot use the `hsi_loop` command to test for the correct operation of a null-modem cable between two Sun systems.

---

## hsi\_loop Output

When the loopback test runs successfully, using any of the test type options, `hsi_loop` reports the statistics shown in the following sample output and then terminates.

```
# hsi_loop hih1
Enter test type:
1: Internal Test
    (internal data loop, internal clocking)
2: Test using loopback plugs
    (external data loop, internal clocking)
3: Test using local or remote modem loopback
    (external data loop, external clocking)
4: Other, previously set, special mode
> 2
speed=9600, loopback=no, nrzi=no, txc=baud, rxc=rx
[ checking for quiet line ]
[ Trying first packet ]
[ Trying many packets ]
100
100 packets sent, 100 received
Port      CRC errors      Aborts    Overruns  Underruns      In <-Drops-> Out
hih1:           0             0          0          0             0             0
hih1:  estimated line speed = 9480 bps
#
```

---

# The `hsi_stat` Command

The `hsi_stat` command gives you information about the packets transmitted and received on a synchronous serial line. The `hsi_stat` command is a valuable tool for monitoring your serial link.

The syntax for `hsi_stat` depends on whether you want to display the statistics of a single port or a number of ports. If you want to display the statistics for a single port, the `hsi_stat` syntax is:

```
# hsi_stat [-c] [-f] device [period]
```

The *device* is the device name of the port (`hih0`, `hih1`, `hih3`, and so on), which is required to display the statistics from one port (see TABLE 4-6 for a description of the `hsi_stat` statistics). You can use the *period* option to display a series of port statistics at a specified interval of seconds.

By default, `hsi_stat` will report the cumulative statistics of the port since the system boot time. However, you can clear the statistics of a port using the `-c` flag, which will reset the port statistics to zero.

With the `-f` flag you can display the full set of statistics of a serial port. This option is useful for debugging purposes.

If you want to display the statistics of a number of ports, the `hsi_stat` syntax is:

```
# hsi_stat [-c] [-f] -a | number_of_ports
```

Replace the *number\_of\_ports* variable with a decimal number. The `hsi_stat` command will then display the statistics of the specified number of ports. For example, if you type the command `hsi_stat 3`, `hsi_stat` will display the statistics of the first three valid ports.

The `-c` and `-f` flag can also be used on a number of ports. For example, `hsi_stat -c 2` will clear the statistics of the first two valid ports. The `-a` flag is used to specify all the valid SunHSI/S ports on the system.

TABLE 4-6 describes the statistics in the `hsi_stat` output.

**TABLE 4-6** `hsi_stat` Statistic Descriptions

<b>Statistic</b>	<b>Description</b>
<code>speed</code>	Reports the line speed as set by <code>hsi_init</code> . It is the system administrator's responsibility to make this value correspond to the modem clocking speed when clocking is provided by the modem.
<code>ipkts</code>	Reports the total number of input packets.
<code>opkts</code>	Reports the total number of output packets.
<code>undrun</code>	Reports the number of transmitter underrun errors. Such errors occur when the local system is too busy to service the serial port hardware. A frame that is not completely sent is aborted, triggering error recovery. Underrun errors can occur when the signaling rate in use on a link is too fast for the local system.
<code>ovrrun</code>	Reports the number of receiver overrun errors. Such errors occur when the local system is unable to accept data fast enough and the port hardware buffers overflow. A frame that is not completely received is aborted, triggering error recovery. Overrun errors can occur when the signaling rate in use on a link is too fast for the local system.
<code>abort</code>	Reports the number of aborted received frames. Occurs when the local serial port received a sequence of eight consecutive ones, in violation of LAPB/SDLC framing rules. Abort errors result from an interruption in the service provided by the link or from clocking problems. Such errors may also be caused by the software running above the driver level. A small number of abort errors probably indicates a software problem rather than a broken link or a persistent clocking problem.
<code>crc</code>	Reports the number of received frames with CRC (Cyclical Redundancy Check, an error detection method) errors. A CRC error is recorded when the checksum on a received frame is incorrect. CRC errors occur when there is a clocking problem (different rates on each side) or a noisy line.
<code>isize</code>	Reports the average size of input packets.
<code>osize</code>	Reports the average size of output packets.
<code>iutil</code>	Reports the input line utilization expressed as a percentage.
<code>outil</code>	Reports the output line utilization expressed as a percentage.
<code>ierror</code>	Reports the input error count. Errors can be incomplete frames, empty frames, or receive clock (RxC) problems.
<code>oerror</code>	Reports the output error count. Errors can be lost clear to send (CTS) signals or transmit clock (TxC) problems.



**TABLE 4-6** hsi\_stat Statistic Descriptions (Continued)

Statistic	Description
inactive	Reports the number of input packets received when receive is inactive.
ishort	Reports the number of short input packets. This is the number of packets received with lengths less than the number of CRC bytes.
ilong	Reports the number of long input packets. This is the number of input packets with lengths larger than the MRU.
olong	Reports the number of long output packets. This is the number of output packets with lengths larger than the MTU.
ohung	Reports the number of times the transmitter hangs, which is usually due to a missing clock.

**Note** – Errors under `abort`, `undrun`, `ovrrun`, and `crc` may indicate a problem with your serial port hardware, connectors, cables, or line-interfacing equipment. If you experience such errors, use `hsi_loop` (or an equivalent loopback diagnostic) to determine which component of your physical link is causing the errors.

The example below shows the `hsi_stat` command displaying the statistics of the `hih1` port:

```
# hsi_stat hih1
speed  ipkts  opkts  undrun  ovrrun  abort   crc   isize  osize
1536000 101    101    0       0       0       0    100   100
```

If you do not enter the optional *interval* parameter, `hsi_stat` terminates after printing the one-line cumulative total of the line statistics.

If you enter a time interval (expressed in seconds), `hsi_stat` operates in an iterative sampling mode, sampling and then displaying line use data for the period specified. In this mode, `hsi_stat` does not output cumulative totals, but displays the incremental changes in the totals between iterations. For example, the command:

```
# hsi_stat hih1 10
```

would produce output similar to the following. Note that in this example, the display will be updated every ten seconds.

```
ipkts opkts undrun ovrrun abort  crc  iutil  outil
12    10    0      0      0    0    5%    4%
22    60    0      0      0    0    3%    90%
36    14    0      0      0    1    51%   2%
Hit Control-C to exit.
```

Using the `hsi_stat` command with an interval adds two fields to the report: `iutil` and `outil`. These two fields report the level of use for the serial line, as a percentage of incoming bandwidth (`iutil`) and outgoing bandwidth (`outil`). These percentages may occasionally be reported as slightly greater than 100% because of inexact sampling times and differences in the accuracy between the system clock and the modem clock. If the percentage of use greatly exceeds 100%, or never exceeds 50%, then the speed value of the `hsi_init` command probably varies greatly from the speed of the modem.

In the following example, the `-a` option is used to display all of the valid ports on the system.

```
# hsi_stat -a
port      speed  ipkts  opkts  undrun  ovrrun  abort  crc  isize  osize
hih8     1536000  0      0      0      0      0      0    0      0
hih9     1536000  0      0      0      0      0      0    0      0
hih10    1536000  0      0      0      0      0      0    0      0
hih11    1536000  0      0      0      0      0      0    0      0
```

If you are experiencing communications problems, leave a `hsi_stat` command running on the console of the machine running an upper-level protocol so that you can see at a glance the recent history of loads and errors. For example, you can run `hsi_stat hih0 60` to get one-minute samples of activity on port 0.

If `hsi_stat` reports that line use is consistently near 100%, then you may need a faster line. Also, watch for errors on the line, especially CRC errors in input packets. A small percentage of such errors can cause severe throughput reduction. These errors are almost always caused by troubles in the communication facilities.

If you see output packets but no input packets, then either the remote system is not initialized, or the line is not properly connected to the remote system.

If you see neither input nor output packets, then the physical layer was not successfully initialized.

---

# SunVTS Diagnostic Testing

The SunVTS software executes multiple diagnostic hardware tests from a single user interface and is used to verify the configuration and the functionality of most hardware controllers and devices. The SunVTS diagnostic primarily operates from a user interface that allows you to control all aspects of the diagnostic test operation.

The `sunlink` diagnostic test, which is shipped with the SunVTS software, checks the functionality of SunHSI/S adapters. This test can be run from the SunVTS user interface, or it can be run from the command line. Refer to the *SunVTS Test Reference Manual* for more information about the `sunlink` test.

Refer to the SunVTS documents for detailed information about the SunVTS software. These documents are available on the *Solaris on Sun Hardware AnswerBook*, which can be viewed on the Sun Documentation website (<http://docs.sun.com/>).

TABLE 4-7 SunVTS Documentation

Title	Description
<i>SunVTS User's Guide</i>	Describes the SunVTS environment; starting and controlling various user interfaces
<i>SunVTS Test Reference Manual</i>	Describes each SunVTS test; provides various test options and command-line arguments
<i>SunVTS Quick Reference Card</i>	Provides an overview of <code>vtsui</code> interface features

The main features of the SunVTS environment include:

- SunVTS kernel

The SunVTS kernel (`vtsk`) controls all facets of the SunVTS environment. When activated, `vtsk` probes the hardware configuration of the system being tested and responds to commands from `vtsui` and `vtstty`. `vtsk` coordinates the operation of individual tests and manages the messages sent by these tests.

- SunVTS user interface

The SunVTS graphical user interface (`vtsui`) diagnostic tool operates on the windowing system. `vtsui` controls `vtsk` and allows you to set user options, start and stop tests, and read log files.

- SunVTS TTY interface

The `vtstty` TTY user interface controls `vtsk` from either a command shell or a terminal attached to a serial port. Most options available in `vtsui` have equivalent options in `vtstty`.

# Building a Synchronous Null Modem Cable and an X.21-to-RS-449 Converter

---

---

## Null Modem Cable Requirements

A synchronous null modem cable is a specially configured cable that simulates back-to-back modems. When the distance between two hosts is not great, you may be able to use a null modem cable instead of a synchronous modem or a synchronous modem eliminator. You can use a null modem cable with any of Sun's RS-449 serial port options.

The maximum distance a null modem cable can work is determined by the specification for your serial port interface.

There are two steps you must perform to use a null modem cable for machine-supplied clocking:

1. Run `hsi_init` with certain parameters set so that the Sun machine, in the absence of a synchronous modem, supplies clocking on the serial line.
2. Configure the cable for the null modem.

---

**Note** – You must run `hsi_init` each time you reboot your machine.

---

---

## Configuring Internal or External Clocking

To configure an RS-449 port to provide transmit clocking for itself as well as receive clocking for the other end of the link, set the `txc` (transmit clock) and `rxr` (receive clock) parameters in `hsi_init` to `baud` and `rxr`, respectively. For example, the following `hsi_init` command, sets the data rate of the first CPU serial port to 9600 bps and sets the clocking as just described:

```
# hsi_init hih0 9600 txc=baud rxr=rxr
```

You enter such a command at both ends of a link if both sides are supplying clocking.

In the situation in which you have Sun systems at both ends of a link and have one machine supply clocking for both sides, on the machine that is not supplying clocking, you enter:

```
# hsi_init hih0 9600 txc=txc rxr=rxr
```

---

## Building the Null Modem Cable

To build a null modem cable, you can configure your own cable or use a standard cable with an adapter box.

---

**Note** – Be sure to use shielded, twisted pair wire when building a null modem cable.

---

If you decide to use an adapter box, be sure to obtain an adapter that allows you to change the pin configurations. Pre-configured adapters generally do not work with synchronous protocols because they do not handle clock signals correctly.

# RS-449 Null Modem Cable

TABLE A-1 and TABLE A-2 list the signals and names for RS-449 and X.21 circuits.

**TABLE A-1** RS-449 Signals

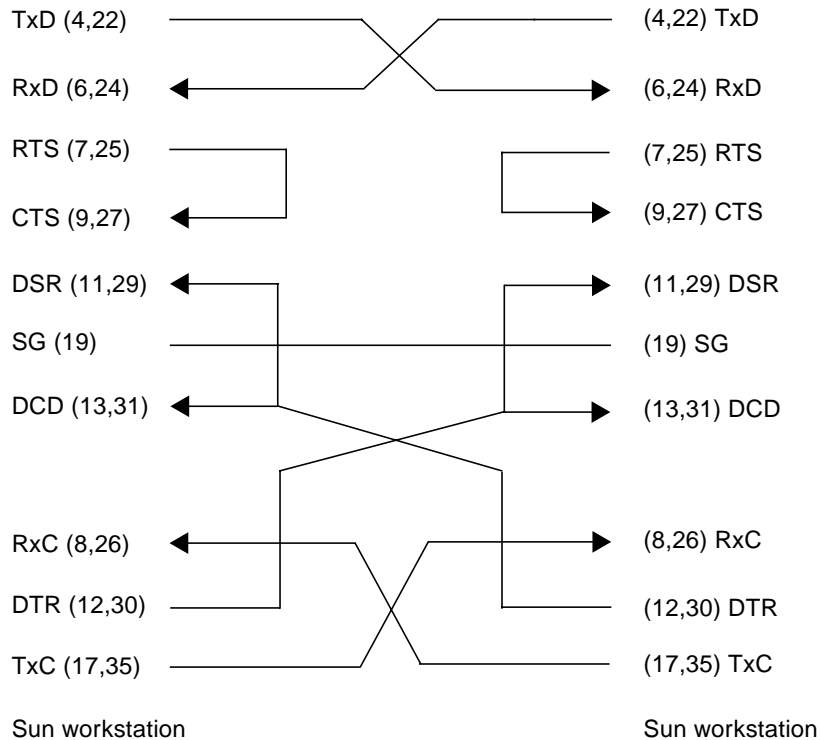
<b>Circuit</b>	<b>Name</b>	<b>Direction</b>	<b>Pin Numbers on DB-37 Connector</b>
TxD	Transmit Data	To DCE	4, 22
RxD	Receive Data	From DCE	6, 24
TxC	Transmit Clock	To DCE	17, 35
TxCI	Transmit Clock In	From DCE	5, 23
RxC	Receive Clock	From DCE	8, 26
RTS	Request to Send	To DCE	7, 25
CTS	Clear to Send	From DCE	9, 27
DCD	Data Carrier Detect	From DCE	13, 31
DTR	Data Terminal Ready	To DCE	12, 30
DSR	Data Set Ready	From DCE	11, 29
SG	Signal Ground		19

**TABLE A-2** X.21 Signals

<b>Circuit</b>	<b>Name</b>	<b>Direction</b>	<b>Pin Numbers</b>
G	Signal Ground		8
T	Transmit	To DCE	2, 9
R	Receive	From DCE	4, 11
C	Control	To DCE	3, 10
I	Indication	From DCE	5, 12
S	Signal Element Timing	From DCE	6, 13

FIGURE A-1 illustrates a synchronous null modem cable that allows you to connect two Sun machines that each supply clocking, using the RS-449 interface. Each Sun supplies clocking on pins 17 and 35. The null modem cable routes this clocking to pins 8 and 26 on the opposite side to provide receive clocking.

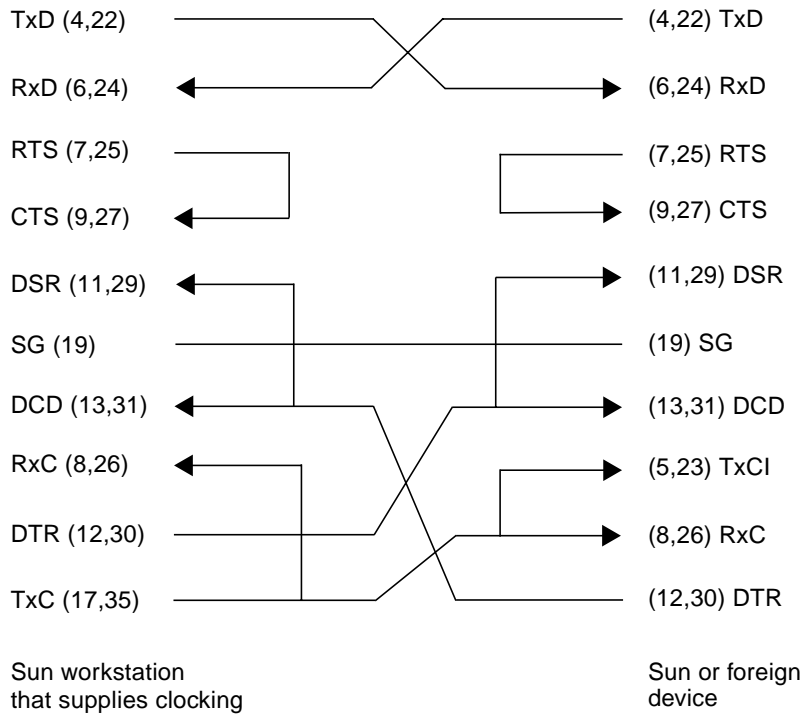
Because the RS-449 interface is balanced, there are two pins for each signal. For example, Transmit Data (TxD), pins 4 and 22, is connected to Receive Data (RxD), pins 6 and 24. This means that pin 4 is connected to pin 6 and pin 22 is connected to pin 24.



**FIGURE A-1** Null Modem Cable (Both Sun Systems Supply Clocking)

FIGURE A-2 illustrates a synchronous null modem cable that allows you to connect a Sun machine to another machine, Sun or not Sun, using the RS-449 interface. The Sun supplies both the transmit and receive clocks for the other machine. Note that this null modem cable is not symmetrical.

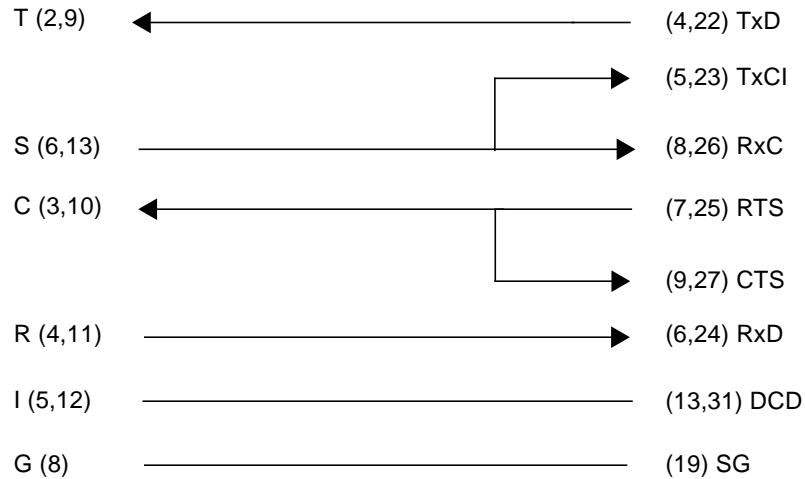




**FIGURE A-2** Null Modem Cable (Sun System Supplies Clocking for Both Sides)

## X.21-to-RS-449 Converter

FIGURE A-3 illustrates the pin connections required for an X.21-to-RS-449 converter.



X.21 Interface  
(15-pin connector)

RS-449 Interface  
(37-pin connector)

FIGURE A-3 X.21-to-RS-449 Converter

When using an X.21 conversion you must perform the following `hsi_init` operation:

```
# hsi_init hiho 9600 txc=txc rxc=rxc
```

---

**Note** – Both Receive and Transmit Clock Inputs (RxC and TxCI) require a clock signal if the `hsi_init` parameter `txc` is set to `txc` (`txc=txc`) and `rxc` is set to `rxc` (`rxc=rxc`).

---

# RS-449 Implementation Example

The RS-449 implementation illustrated in FIGURE A-4 is not necessarily a functional design. Instead, its purpose is to show basic design considerations.

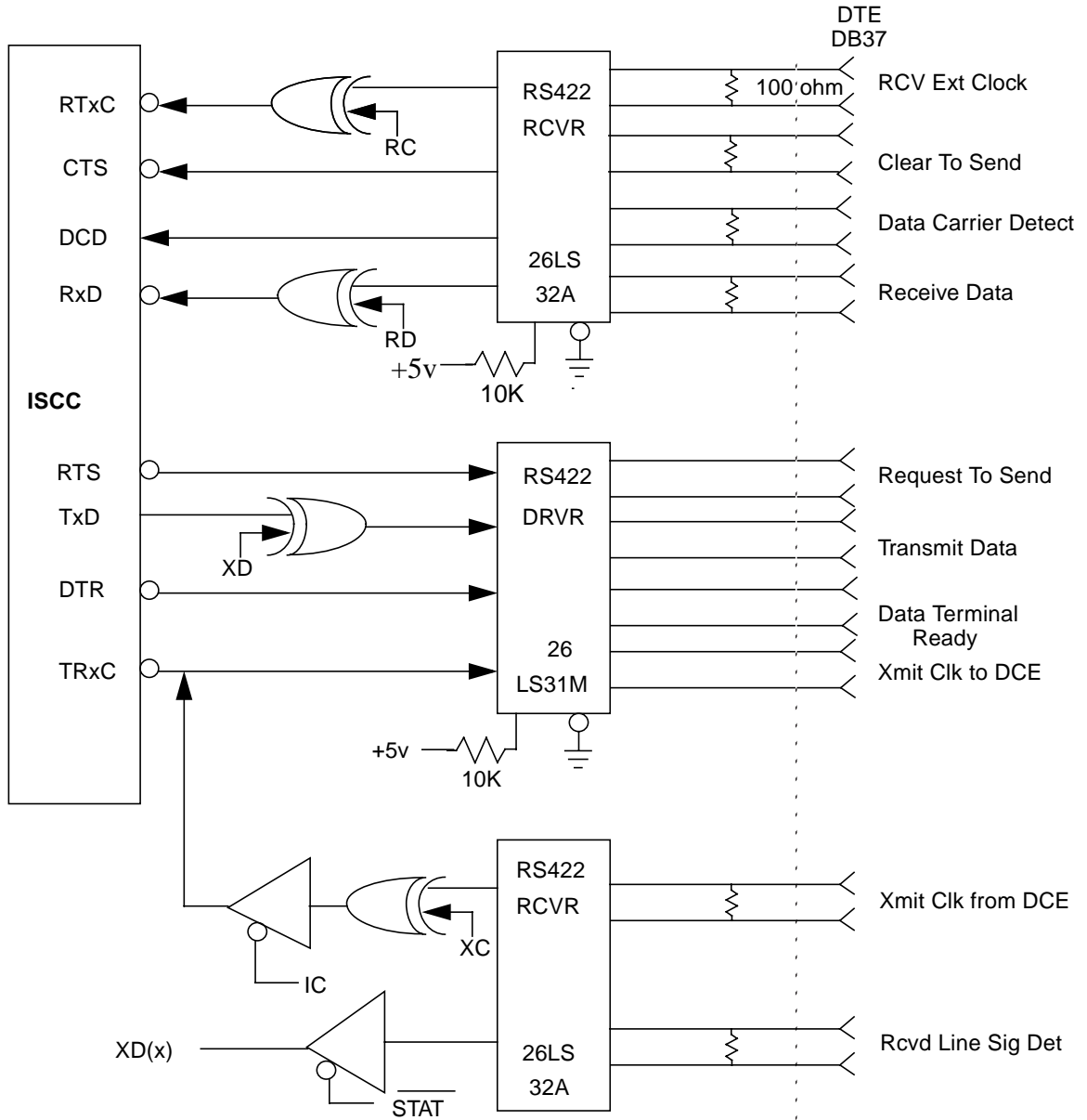


FIGURE A-4 RS-449 Implementation Example

# RS-449 Pin Assignments and Descriptions

TABLE A-3 describes the functions of the RS-449 signals and their pinouts.

**TABLE A-3** Functional Description of RS-449 Interface Signals

RS-449 Pin #	Signal Name	Function
1	Shield Ground	Allows tandem sections of shielded cable to retain continuity through the connector.
19	Signal Ground (SG)	Directly connects the DTE circuit ground to the DCE circuit ground, providing a path for DTE and DCE signal commons.
4/22	Transmit Data (TxD)	Used by the DTE to pass binary data to the DCE for transmission over the communications channel.
6/24	Receive Data (RxD)	Used by the DCE to pass binary data received from the communications channel to the DTE.
5/23	Transmit Clock In (TxCI)	Allows the DCE to transmit signal element timing to the DTE. This allows the DTE Transmit Data signal on circuit TxD to be in synchronization with On/Off transitions on this lead.
8/26	Receive Clock (RxC)	Transitions on this lead allow the DTE to time data received over circuit RxD.
17/35	Transmit Clock (TxC)	Allows the DTE to provide transmit timing information to the DCE so that it can synchronize with data sent over the TxD lead.
7/25	Request to Send (RTS)	Used by the DTE to advise the DCE it wishes to transmit data.
9/27	Clear to Send (CTS)	Used by the DCE to advise the DTE that the DCE is ready to send data over the communications channel.
11/29	Data Set Ready (DSR)	Used to advise the DTE of the Ready status on the DCE. In most cases, it simply implies the unit is powered on.
12/30	Data Terminal Ready (DTR)	Used by the DTE to advise the DCE it is ready to transmit or receive.
13/31	Data Carrier Detect (DCD)	The DCE uses this lead to advise the DTE that an incoming signal on the communications channel is present. When first initialized it is an indication to the DTE to expect data momentarily.

# 96-Pin Connector Signal and Pin Assignments

TABLE A-4 lists the pin and signal descriptions of the 96-pin connector. Note that there are no connections to pins 2, 23, 26, 47, 50, 71, or 95.

**TABLE A-4** 96-Pin Connector Pin and Signal Assignments

From J1	To Channel	Pin	Direction	Signal Description
NC	-	1	None	Shield
1,49	-	19	-	AB Signal Ground
3	A	4	To DCE	+BA Transmit Data
51	A	22	To DCE	-BA Transmit Data
4	A	6	From DCE	+BB Receive Data
52	A	24	From DCE	-BB Receive Data
5	A	17	To DCE	+DA Transmit Clock
53	A	35	To DCE	-DA Transmit Clock
6	A	5	From DCE	+DB Transmit Clock In
54	A	23	From DCE	-DB Transmit Clock In
7	A	8	From DCE	+DD Receive Clock
55	A	26	From DCE	-DD Receive Clock
8	A	7	To DCE	+CA Request-to-Send
56	A	25	To DCE	-CA Request-to-Send
9	A	9	From DCE	+CB Clear-to-Send
57	A	27	From DCE	-CB Clear-to-Send
10	A	11	From DCE	+CC Data Set Ready
58	A	29	From DCE	-CC Data Set Ready
11	A	12	To DCE	+CD Data Terminal Ready
59	A	30	To DCE	-CD Data Terminal Ready
12	A	13	From DCE	+CF Data Carrier Detect
60	A	31	From DCE	-CF Data Carrier Detect
NC	B	1	-	Shield
24,72	-	19	-	AB Signal Ground
13	B	4	To DCE	+BA Transmit Data
61	B	22	To DCE	-BA Transmit Data

**TABLE A-4** 96-Pin Connector Pin and Signal Assignments (*Continued*)

From J1	To Channel	Pin	Direction	Signal Description
14	B	6	From DCE	+BB Receive Data
62	B	24	From DCE	-BB Receive Data
15	B	17	To DCE	+DA Transmit Clock
63	B	35	To DCE	-DA Transmit Clock
16	B	5	From DCE	+DB Transmit Clock In
64	B	23	From DCE	-DB Transmit Clock In
17	B	8	From DCE	+DD Receiver Clock
65	B	26	From DCE	-DD Receiver Clock
18	B	7	To DCE	+CA Request-to-Send
66	B	25	To DCE	-CA Request-to-Send
19	B	9	From DCE	+CB Clear-to-Send
67	B	27	From DCE	-CB Clear-to-Send
20	B	11	From DCE	+CC Data Set Ready
68	B	29	From DCE	-CC Data Set Ready
21	B	12	To DCE	+CD Data Terminal Ready
69	B	30	To DCE	-CD Data Terminal Ready
22	B	13	From DCE	+CF Data Carrier Detect
70	B	31	From DCE	-CF Data Carrier Detect
NC	C	1	-	Shield
25,73	-	19	-	AB Signal Ground
27	C	4	To DCE	+BA Transmit Data
75	C	22	To DCE	-BA Transmit Data
28	C	6	From DCE	+BB Receive Data
76	C	24	From DCE	-BB Receive Data
29	C	17	To DCE	+DA Transmit Clock
77	C	35	To DCE	-DA Transmit Clock
30	C	5	From DCE	+DB Transmit Clock In
78	C	23	From DCE	-DB Transmit Clock In
31	C	8	From DCE	+DD Receiver Clock
79	C	26	From DCE	-DD Receiver Clock

**TABLE A-4** 96-Pin Connector Pin and Signal Assignments (*Continued*)

From J1	To Channel	Pin	Direction	Signal Description
32	C	7	To DCE	+CA Request-to-Send
80	C	25	To DCE	-CA Request-to-Send
33	C	9	From DCE	+CB Clear-to-Send
81	C	27	From DCE	-CB Clear-to-Send
34	C	11	From DCE	+CC Data Set Ready
82	C	29	From DCE	-CC Data Set Ready
35	C	12	To DCE	+CD Data Terminal Ready
83	C	30	To DCE	-CD Data Terminal Ready
36	C	13	From DCE	+CF Data Carrier Detect
84	C	31	From DCE	-CF Data Carrier Detect
NC	D			Shield
48,96		19		AB Signal Ground
37	D	4	To DCE	+BA Transmit Data
85	D	22	To DCE	-BA Transmit Data
38	D	38	From DCE	+BB Receive Data
86	D	24	From DCE	-BB Receive Data
39	D	17	To DCE	+DA Transmit Clock
87	D	35	To DCE	-DA Transmit Clock
40	D	5	From DCE	+DB Transmit Clock In
88	D	23	From DCE	-DB Transmit Clock In
41	D	8	From DCE	+DD Receiver Clock
89	D	26	From DCE	-DD Receiver Clock
42	D	7	To DCE	+CA Request-to-Send
90	D	25	To DCE	-CA Request-to-Send
43	D	9	From DCE	+CB Clear-to-Send
91	D	27	From DCE	-CB Clear-to-Send
44	D	11	From DCE	+CC Data Set Ready
92	D	29	From DCE	-CC Data Set Ready
45	D	12	To DCE	+CD Data Terminal Ready

**TABLE A-4** 96-Pin Connector Pin and Signal Assignments (*Continued*)

<b>From J1</b>	<b>To Channel</b>	<b>Pin</b>	<b>Direction</b>	<b>Signal Description</b>
93	D	30	To DCE	-CD Data Terminal Ready
46	D	13	From DCE	+CF Data Carrier Detect
94	D	31	From DCE	-CF Data Carrier Detect



## hsi\_init Options for T1 Compatibility

---

The version of the `hsi_init` command shipped with the SunHSI/S 3.0 software has options that allow you to invert data and clock signals to accommodate the requirements of T1 or CEPT transmission equipment.

The `hsi_init` parameters that allow for inversion are:

- `txd` - transmit data signal
- `rxn` - receive data signal
- `txc` - transmit clock signal
- `rxn` - receive clock signal

The effect of the default settings for all of these parameters is that SunHSI/S 3.0 software does *not* invert the data or clock signal controlled by the parameter. To invert a signal, you specify a setting of the form `param_name=-paramname`, for example, `txc=-txc`.

As an example, suppose you want to invert the transmit and receive data signals on the first SunHSI/S port (port 0) on the second SunHSI/S adapter in your system. To do so, enter the following command:

```
# hsi_init hih4 txd=-txd rxn=-rxn
```

To invert both clock and data signals, enter:

```
# hsi_init hih4 txd=-txd rxn=-rxn txc=-txc rxn=-rxn
```

The following section discusses the background and requirements for these inverted settings.

---

# Inverted Settings

The reason for inverting data signals is distinct from the reason for inverting clock signals. The background for data signal inversion is explained first, then the background for clock signal inversion is explained.

## Data Signal Inversion

The requirement for inverting data signals arises from the “ones density” problem you encounter with most T1 transmission lines in North America. The T1 transmission scheme uses a signaling mechanism known as Alternate Mark Inversion (AMI), in which one bits are represented by a positive or negative pulse, while zero bits are represented by the absence of a pulse. In this scheme, the polarity of each pulse must be the opposite of the polarity of the pulse which immediately preceded it. This signaling scheme makes it possible to embed a reference clock for the data into the data stream itself.

Various types of T1 transmission equipment, such as Data Service Units (DSU), Channel Service Units (CSU), repeaters, and various telephone central office equipment, must be able to keep a phase locked loop (PLL) circuit locked on to this reference clock. This PLL circuit uses the pulses generated when one bits are transmitted to lock the embedded clock to a local reference oscillator. To keep the PLL circuit locked on the extracted clock, a certain density of pulses (one bits) must be guaranteed. For North American T1 lines, the density requirement dictates that at least one out of every 16 bits must be a one (see *AT&T Technical Publication 62411*). Another way of stating this is that no more than 15 consecutive zero bits can occur anywhere in the data stream.

T1 lines were originally intended to carry voice traffic, wherein the digitized voice signals could be altered to meet the ones-density requirement by forcing every eighth bit of a voice channel to be a one. This practice introduces a small—but virtually inaudible—amount of distortion in the voice signal. Digital data streams between two computers are another matter, since the corruption of even one data bit causes a packet to be rejected. Note that in a typical data packet it is quite easy to produce bit patterns that violate the ones-density requirement. A random file could easily contain a sequence of bytes that would produce 16 or more consecutive zero bits if transmitted serially.

There are many different schemes for circumventing the ones-density requirement. The most common technique simply reserves every eighth bit of the signal for a “density bit” and forces this bit to be a one. Obviously, these bits are not available

for data transmission, which means that 12.5 percent of the bandwidth of the T1 line is wasted. When you consider that the lease cost for a coast-to-coast T1 line can be exceedingly expensive, this waste of bandwidth can be unacceptable.

There are alternatives. One of them uses a special code that transmission equipment can generate when using the AMI signalling scheme. This special code depends on the fact that two successive one bits that are represented by pulses of the same polarity result in a signal known as a “Bipolar Violation.” A CSU can be designed so that it will automatically replace any string of eight consecutive zeros with a special code pattern that contains two of Bipolar Violations. A compatible, receiving CSU recognizes this special code and converts it back to a pattern of eight zeros. This technique is known by the acronym B8ZS, which stands for Bipolar with 8-Zero Substitution.

All CEPT lines (the European equivalent of T1) mandate the use of a variant of B8ZS that holds the density requirement down to no more than three consecutive zeros. However, telephone companies in North America have been slow to adopt B8ZS, because it would entail a significant capital investment. Therefore, the B8ZS solution will not solve the ones-density problem in the short term.

An alternative to B8ZS—an alternative used by the SunHSI/S product—makes use of the fact that the HDLC framing rules specify that any data stream that contains five or more consecutive one bits requires that the transmitting end insert a zero bit after the fifth one bit. This guarantees that the HDLC flag pattern 01111110 (hex 7E) does not occur randomly inside a frame. The receiving end must automatically discard the zero bit that follows a pattern of five consecutive ones. So, HDLC framing, which is used by SunHSI/S, guarantees that, except for the flag pattern, in any set of six bits, at least one bit will be a zero. If you include the flag pattern, you can say that in any set of seven bits, at least one bit will be a zero.

By inverting the data signal with HDLC framing on both ends of a link, the HDLC zero insertion algorithm becomes a ones insertion algorithm. This guarantees that in any set of seven bits, at least one bit will be a one. Thus, the HDLC data stream meets the density requirements of North American T1 lines without sacrificing any bandwidth.

## Clock Signal Inversion

The need to invert clock lines is separate from the need to invert data lines. Most computer, modem, and terminal vendors adhere to an industry standard specification known as RS-334. This specification defines the relationship between a data bit and a reference clock on a synchronous serial link. The specification also says that a device should transmit data with reference to the rising edge of the clock signal and that data should be received with reference to the falling edge of the clock signal.

When using long cables or cables not carrying a clock signal, a phase shift may occur causing a high number of errors. In such cases, inverting the clock signal may correct the phase shift. You may also need to invert the clock signal when connecting a SunHSI/S port to equipment not adhering to the RS-334 standard.

## hsi\_init Options for Operating Modes

---

This appendix describes the operating modes that can be set by the `hsi_init` utility (see “The `hsi_init` Command” on page 29 for instructions on how to use this utility).

The SunHSI/S driver operates in two main operating modes, the high-level data link control (HDLC) mode and the IBM (SDLC) mode. The HDLC mode always operates in a full-duplex, point-to-point fashion. While the IBM mode defaults to a full-duplex, point-to-point, operation, you can also set this mode to be either a half-duplex or a multi-point operation.

---

### HDLC Mode

The default operating mode used by the SunHSI/S driver is the HDLC full-duplex protocol (`mode=fdx`). In this mode the transmitter is always enabled and it sends flag bytes continuously when it is not sending a data frame.

If no message is currently being transmitted, the driver will attempt to start sending its next message. At this point the driver indicates that it is busy transmitting, in order to prevent the transmission of another message concurrently. The driver also activates a mechanism that ensures that the transmit operation will not hang if the hardware is not responding.

When the transmission is completed, the busy mechanism previously set is cleared and the next message can be transmitted. If the transmission is hung, an abort sequence is sent instead of the CRC so that the receiver will not interpret the frame as valid data. The message is discarded, and the output error statistic is incremented, which allows for a proper recovery by higher level protocols.

The received data is buffered until a complete frame has been received. If any error occurs during the reception of a frame, the appropriate statistic is incremented and the frame is discarded.

---

## IBM (SDLC) Mode

This mode is designed to support IBM system network architecture (SNA) communications. It uses most of the same protocols used in HDLC mode, with two major exceptions:

- When the line is idle, instead of sending flag bytes, the transmitter is disabled.
- The request-to-send (RTS) and clear-to-send (CTS) signals are used to gate transmission.

## IBM Full-Duplex Mode

When the SunHSI/S software is set to this mode (`mode=ibm-fdx`), the software uses a full-duplex point-to-point communication protocol. Both ends of the link are expected to have RTS and CTS signals asserted at all times when data is being exchanged. When starting a message transmission, the interface raises the RTS signal and expects the CTS signal to be asserted immediately. If this is not done, all messages currently queued for transmission are discarded, and the write operation returns an error.

If the CTS signal drops before the frame transmission is complete, the frame is discarded and the abort error statistic is incremented. If the transmission underruns, an abort sequence is *not* sent and the frame is silently discarded. The RTS signal remains asserted until the data transmission is complete.

## IBM Half-Duplex Mode

Half-duplex is a sub-mode of the IBM mode (`mode=ibm-hdx`). Half-duplex mode operates in the same manner as full-duplex mode except that transmission cannot occur while receiving, and vice-versa. When a transmission is completed, the RTS signal is dropped to “turn the line around.” Dropping the RTS signal tells the remote station to begin transmitting if it wishes.

## IBM Multi-Point Mode

In a multi-point configuration (`mode=ibm-mpt`), more than two stations “share” a link. This configuration is accomplished by designating one station as a primary station and the rest as secondary stations. In this mode, the port acts as a secondary station. The primary station arbitrates traffic on the link by polling the secondary stations, asking them all if they are ready to transmit.

If a secondary station has data to transmit, it will raise its RTS signal and check for CTS signals. When a CTS signal comes up the station may begin transmitting, following the same rules for RTS and CTS signals used in half-duplex mode. When the transmission is complete the secondary drops the RTS signal, which allows another station to respond to a poll and begin transmitting. The RTS signal cannot be dropped until the transmission is complete.





## Hardware Functional Description

---

This appendix contains a detailed description of the hardware block diagram (FIGURE 1-2) in Chapter 1.

SunHSI/S provides four synchronous serial communications channels for any slave host. Each channel can operate at a maximum speed of 2.048 Mbps, and each supports a separate RS-449 interface. Furthermore, the aggregate bandwidth for all four ports is limited to 2.5 Mbps (full-duplex).

---

### Integrated Serial Communications Controller (ISCC)

Each ISCC provides two synchronous serial channels and a four-channel DMA controller for full-duplex operation. All serial channels are connected to the SBus interface through RS-422 drivers and receivers. The RS-449 signals, with the exception of the CF (Receive Line Detect), are supported by the ISCC. The status of the CF signal for each channel is obtained through a status register.

The signals BA (Transmit Data), BB (Receive Data), DB (Transmit Clock), and DD (Receive Clock) for both channels can be inverted under software control. This flexibility provides user-configurable options and allows you to achieve optimum bit density on T1 lines. The commands that control the inversion are described in Appendix B. At reset, all channels are set to the non-inverting state.

The master clock for the ISCC is 16.32 MHz. This clock rate is used because it is divisible into standard bit rates from 300 bps to 2.5 Mbps. It is possible to program the ISCC to divide and use the internal clock as a programmable baud rate source. The commands that control the clock selection are listed in Chapter 4. At reset, the internal clocks are selected. Normally, in synchronous mode, the transmit and receive clocks are provided by the DCE.

An internal four-channel DMA provides full-duplex operation for both ISCC serial channels. Bus utilization is optimized by incorporating the internal packet status FIFO and the internal DMA, which reduces host intervention.

---

## Dual-Ported 32 Kbyte Random Access Memory

The use of dual-ported random access memory (RAM) minimizes bus contention between the host and the serial controllers. The dual-ported design provides two ports to the memory; one port is provided for the ZD bus and one port is provided for the SBus.

The two ports provide asynchronous access to the memory, thereby allowing the host to access the RAM without any delay caused by contention from the eight DMA channels. The controller provides a slave interface to the SBus. The host can access the dual-ported RAM in word, half-word, or byte mode.

---

## System Clock

The system clock is provided as separately buffered clocks that are derived from one 16.32 MHz clock driver. The frequency of 16.32 MHz is used to provide a clock that is divisible into the standard baud rates frequencies.

---

## Bit Latch Controller

Three addressable-bit latches are used to control the inversion of interface signals and to control the selection of the internal clocks for each of the serial channels.

The latches are set, or reset, by writing a one or a zero to the physical address. When a bit is set, it causes the interface line associated with it to be inverted. In the case of clocks, the set bit causes selection of the clock associated with the bit position.

---

# EPROM

SunHSI/S supports up to 32 Kbytes of read-only memory. This EPROM is only used for supplying the required ID string of data that begins at location zero of the physical address for the slot in which the SunHSI/S adapter is installed.

---

## Status Buffers

A buffer is used to report the status of the CF signal (Receive Line Detector) from the DCE. The buffer is also used to report the status of the interrupt lines. TABLE D-1 describes the bit assignments for the Status Buffers.

**TABLE D-1** Status Buffer Bit Assignments

Bit	Description	Direction
LA(0)	Interrupt from Receive, Channel A	-
LA(1)	Interrupt from Receive, Channel B	-
LA(2)	Interrupt from Receive, Channel C	-
LA(3)	Interrupt from Receive, Channel D	-
LA(4)	CF Receive Line Signal Detect, Chan A	From DCE
LA(5)	CF Receive Line Signal Detect, Chan B	From DCE
LA(6)	CF Receive Line Signal Detect, Chan C	From DCE
LA(7)	CF Receive Line Signal Detect, Chan D	From DCE



# Software Functional Description

---

This appendix contains a functional description of the SunLink SunHSI/S 3.0 software.

---

## Initialization

The SunHSI/S 3.0 driver software is dynamically loadable and unloadable to help conserve memory resources. During software installation, the driver is installed to the system with `add_drv(1m)`, which temporarily loads the driver into the kernel and uses the attach routine of the driver to dynamically create the device nodes for each hardware instance installed. This autoconfiguration feature is provided in the Solaris software and it is also supported in the SunHSI/S 3.0 driver software.

After the autoconfiguration, the module is unloaded. The driver module is loaded to the system again when the driver is first referenced. The autoconfiguration and initialization of the SunHSI/S driver software is performed through a set of standard SBus device driver routines:

```
static int hsidentify (dev_info_t *dip)
```

This routine is called at initialization time to find out whether the driver controls the device specified by parameter `dip`. The driver compares `ddi_get_name(9E)` with a hard-coded string "HSI" with `strcmp(3C)`. This routine returns the `DDI_IDENTIFIED` message if both strings match. Otherwise, the `DDI_NOT_IDENTIFIED` message is returned.

```
static int hsprobe (dev_info_t *dip)
```

This routine is called at initialization when the calling of `hsidentify` succeeded. Since SunHSI/S hardware is a self-identifying device, the system performs the probe function. This routine always returns the `DDI_PROBE_SUCCESS` message.

```
static int hsattach (dev_info_t *dip, ddi_attach_cmd_t cmd)
```

This routine is called at autoconfiguration time when the driver module is loaded into the system and the calling of `hsprobe` routine succeeded. In turns it calls the protocol-dependent routine `hih_attach` to create device nodes that driver needs in order to access the hardware and `hih_init` to set up a control structure for each port on the board.

The structure includes information such as hardware address, transmission state, minor number and port configuration parameters. The `hsattach` routine also calls the standard DDI routines to map SunHSI/S hardware registers and to add device interrupt service routine to the kernel.

```
static int hsdetach (dev_info_t *dip, ddi_detach_cmd_t cmd)
```

This routine is called when the driver module is unloaded from the system. It calls the protocol-dependent routine `hih_detach` to remove the device node and reset the hardware. It also calls standard DDI routines to un-map the hardware registers and delete the device interrupt service routine from the system.

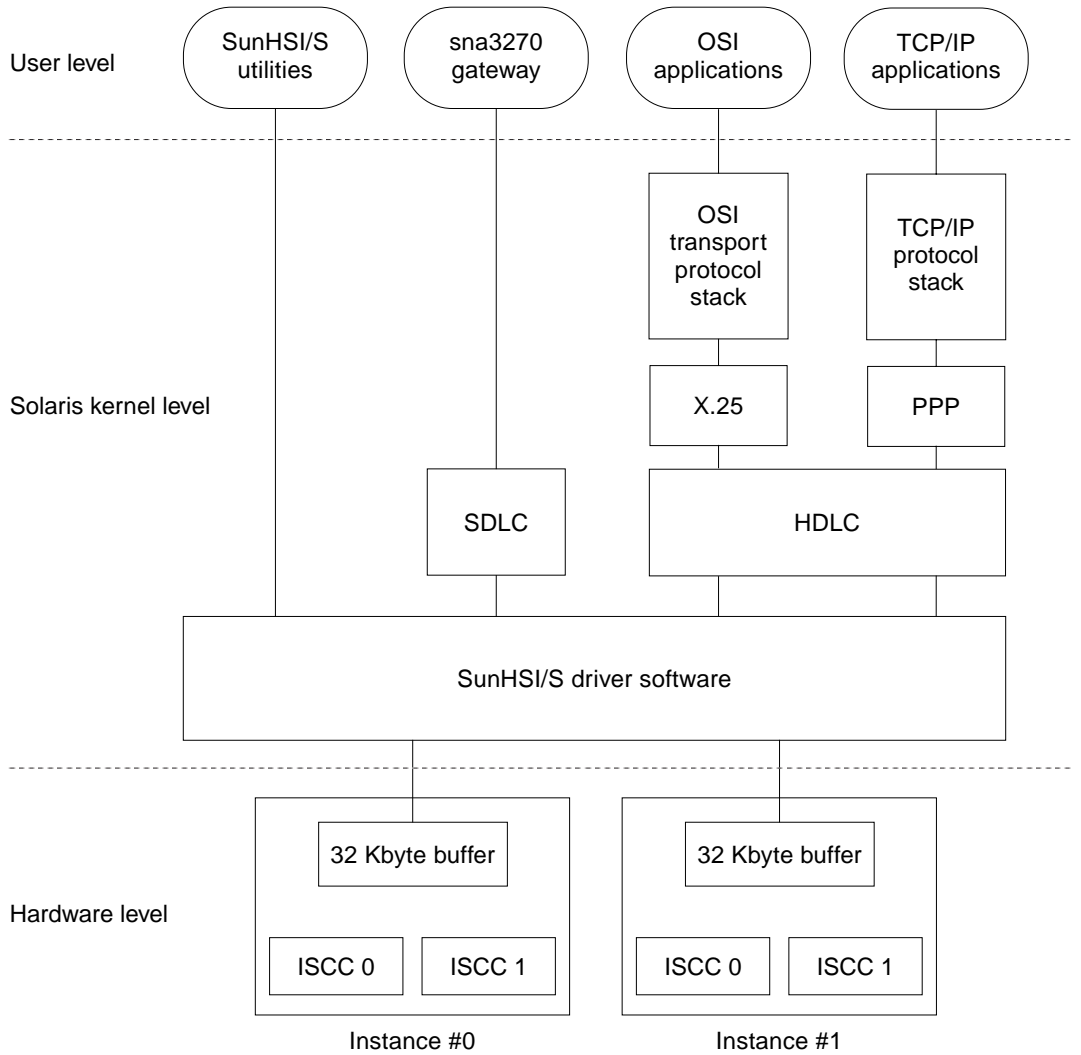
---

## External Interfaces

The SunHSI/S 3.0 driver provides a streams-based interface to the Solaris kernel and user program. The driver software can be reached from user program through standard `open(2)`, `close(2)`, `putmsg(2)`, `getmsg(2)`, and `ioctl(2)` system calls. The driver software communicates with other kernel resident (upper protocol) streams modules using the standard utility `putnext(9F)`.

The driver software can also be reached through hardware interrupts from the SunHSI/S hardware. Hardware interrupts, both standard serial control interrupts and on-chip DMA interrupts, are received through the SunHSI/S hardware from the Integrated Serial Communications Controller (ISCC).

The SunHSI/S driver software interface diagram, shown in FIGURE E-1, shows the external interfaces of the driver.



**FIGURE E-1** SunHSI/S 3.0 Driver Software Interface

# IOCTLs

All driver control is achieved through IOCL(2) system call. TABLE E-1 lists the IOCTL parameters for the SunHSI/S Driver.

TABLE E-1 IOCTL Parameters for the SunHSI/S Driver

IOCTL Name	Purpose and Structure
S_IOCGETMODE	Retrieves the current transmission parameters setting for a particular port. Structure required: <pre>struct scc_mode {     char    sm_txclock;    /*transmit clock sources */     char    sm_rxclock;    /*receive clock sources */     char    sm_iflags;    /*data and clock invert flags */     u_char  sm_config;    /*boolean configuration options*/     int     sm_baudrate;   /*real baud rate */     int     sm_retval;    /*SMERR codes go here,                           query with GETMODE */ };</pre>
S_IOCSETMODE	Reinitializes a particular port with new transmission parameters setting. Structure required: Same as S_IOCGETMODE
S_IOCGETSPEED	Retrieves the current baud rate setting for a particular port. Structure required: <pre>int speed;</pre>
S_IOCGETMRU	Retrieves the current Maximum Receiving Unit (MRU) setting for a particular port. Structure required: <pre>int mru;</pre>
S_IOCSETMRU	Sets to a new Maximum Receiving Unit (MRU) for a particular port. Structure required: <pre>int mru;</pre>
S_IOCGETMTU	Retrieves the current Maximum Transmission Unit (MTU) setting for a particular port. Structure required: <pre>int mru;</pre>
S_IOCSETMTU	Sets to a new Maximum Transmission Unit (MTU) setting for a particular port. Structure required: <pre>int mru;</pre>
S_IOCGETMCTL	Retrieves the current DCD/CTS state for a particular port. Structure required: <pre>u_char mctl;</pre>



**TABLE E-1** IOCTL Parameters for the SunHSI/S Driver (*Continued*)

IOCTL Name	Purpose and Structure
S_IOCGETSTATS	<p>Retrieves the data or errors statistics that SunHSI/S driver has accumulated for a particular port. Structure Required:</p> <pre> struct hs_stats {     long    ipack;        /*input packets*/     long    opack;        /*output packets*/     long    ichar;        /*input bytes*/     long    ochar;        /*output bytes*/     long    abort;        /*abort received*/     long    crc;          /*CRC error*/     long    cts;          /*CTS timeouts*/     long    dcd;          /*Carrier drops*/     long    overrun;      /*receiver overrun*/     long    underrun;     /*xmitter underrun*/     long    ierror;       /*input error (rxbad)*/     long    oerror;       /*output error                           (watchdog timeout)*/     long    nobuffers;    /*no active receive block available                           (&lt;CRC-bytes+1)*/     long    ishort;       /*input packet too short                           (&gt; mru)*/     long    ilong;        /*input packet received                           when inactive*/     long    idma;         /*receive dma error*/     long    olong;        /*output packet too long (&gt; mtu)*/     long    ohung;        /*transmit hung (usually                           missing clock)*/     long    odma;         /*transmit dma error*/ }; </pre> <p>or:</p> <pre> struct sl_stats {     long    ipack;        /*input packets*/     long    opack;        /*output packets*/     long    ichar;        /*input bytes*/     long    ochar;        /*output bytes*/     long    abort;        /*abort received*/     long    crc;          /*CRC error*/     long    cts;          /*CTS timeouts*/     long    dcd;          /*Carrier drops*/     long    overrun;      /*receiver overrun*/     long    underrun;     /*xmitter underrun*/     long    ierror;       /*input error (rxbad)*/     long    oerror;       /*output error (watchdog timeout)*/     long    nobuffers;    /*no active receive block available*/ }; </pre>

**TABLE E-1** IOCTL Parameters for the SunHSI/S Driver (*Continued*)

<b>IOCTL Name</b>	<b>Purpose and Structure</b>
S_IOCCLRSTATS	Clears the data and error statistics that SunHSI/S driver has accumulated for a particular port. Structure Required: Same as S_IOCGETSTATS

The `scc_mode` and `sl_stats` structures defined in the system include file (`/usr/include/sys/ser_sync.h`).

---

## Interrupts

Hardware interrupts are serviced through the `hsintr` interrupt service routine. This routine determines the source of the interrupt by means of an interrupt vector read from the ISCC chip. If the interrupt is not from the SunHSI/S adapter(s), the procedure returns a zero value. If the interrupt is from the ISCC chip, the interrupt is serviced. The possible hardware interrupts are listed in TABLE E-2.

**TABLE E-2** Hardware Interrupts

<b>Interrupt</b>	<b>Cause</b>
External Status	Caused by either a Break or an Abort transmitted over the DLC line. An Abort is series of fifteen successive ones sent over the line by the transmitting side.
Transmit Interrupt	Generally occurs when a full packet has been transmitted over the line.
Receive Interrupt	Special circuitry on the SunHSI/S adapter detects the receipt of a complete frame and interrupts the system processor(s).
DMA Transmit Terminal Count	Occurs when a complete packet has been generated. In addition to this interrupt, a Transmit Interrupt occurs at transmission.
DMA Receive Terminal Count	This is an error condition implying that the received packet is larger than the DMA Receive Buffer.

---

# Packet Transmission and Reception

When an upper-protocol layer or a user program has a packet ready for transmission by the interface, it calls either the `putnext(9F)` utility or the `putmsg(2)` system call to pass the packet to the SunHSI/S driver. If the SunHSI/S driver transmission buffer is empty, the `hih_wput` routine of the driver copies the packet into the RAM buffer on the SunHSI/S adapter. From there, the packet is transmitted across the serial line. If the transmit buffer is full, the packet is queued at the local `WRITE` queue for later transmission by the `hih_wsrv` routine.

If the local `WRITE` queue has too many packets (beyond the high-water mark), the upper layers will detect the congestion by calling `canputnext` and slowing down the traffic until the congestion is resolved.

When a correct packet is received, it is copied from the SunHSI/S board RAM to a stream message buffer. When a complete packet is received, the `hih_rsrv` routine of the driver is called to send the packet to upper layers.

The Z16C35 ISCC support an internal status FIFO of approximately ten packets. As a result, it is possible to queue many packets during reception without servicing an interrupt. Since most synchronous protocols require relatively fast reception of control packets, it is detrimental to queue up packets at the driver level for long periods. To alleviate this possible problem, an algorithm based on the receive queue size and a timer is used. Either event causes the packets to be sent to the upper level.



# Index

---

## NUMERICS

### 96-pin

- cable, connecting, 13
- connector, 53
- pin assignments, 53
- signals, 53

## A

### adapter

- connecting cable, 13
- functional description, 65
- illustrated, 1
- installing, 7
- power consumption, 4
- SBus interface, 4
- verifying installation, 15

alternate mark inversion (AMI), 58

## B

bandwidth, aggregate, 4

baud rate, setting, 31

### bipolar

- 8-Zero Substitution (B8ZS), 59
- violation, 59

booting, reconfiguration, 16

## C

### cables

- 96-pin, 13
- changing, 22
- connecting, 13
- null modem cable, 45 to 49
- RS-232 to RS-449 converter, 14

### CD-ROM

- files and directories, 20
- mounting, 20
- unmounting, 21

### CEPT

- inverting
  - incoming receive clock, 32, 59
  - incoming transmit clock, 32, 59
  - receive data, 33, 59
  - transmit data, 33, 59
- lines, 59

channel service unit (CSU), 58

clock signal inversion, 57, 59

## D

data signal inversion, 57

density bit, 59

### device driver

- see* software, 17

### diagnostics

- hsi\_loop, 5, 34
- Solaris utilities, 5
- sunlink, 43

- directory structure, software, 18
- DMA interrupts, 70
- DOC notice, iv
- documentation
  - feedback, xv
  - related, xv
  - SunVTS, 43
  - website, xv

## E

- error messages
  - error, 25
  - informational, 25
  - warning, 26

## F

- FCC notice, iii
- full-duplex mode
  - HDLC, default, 61
  - IBM compatible, 62

## H

- half-duplex mode, IBM compatible, 62
- hardware
  - bit latches, 66
  - block diagram, 3
  - description, 2
  - dual-ported RAM, 66
  - EPROM, 67
  - functional description, 65
  - installation, 7 to 16
    - adapter, 7
    - patch panel, 8
    - verifying, 16
  - interrupts, 70, 74
  - ISCC, 65
  - port numbers, 28
  - status buffers, 67
  - system clock, 66
- HDLC, 59, 61
  - framing, 59
  - setting, 32

- help, getting, 6
- hihN console messages, 25
- HSI device entry, 16
- hsi\_init
  - configuring
    - external clocking, 46
    - internal clocking, 46
  - inverting
    - clock signals, 59
    - data signals, 58
  - setting
    - baud rate, 31
    - maximum receive unit (MRU), 33
    - maximum transmission unit (MTU), 33
    - modem signals, 32
    - NRZI data encoding, 31
    - operating modes, 32
    - receive clock origin, 32
    - transmit clock origin, 32
- hsi\_init
  - arguments, 29
  - checking MTU and MRU sizes, 22
  - description, 27, 29
  - internal loopback state, 31
  - man page, 23
  - one-word commands, 33
  - operating modes, 61 to 63
  - options for T1 compatibility, 57
  - parameters, 30
  - re-initializing ports, 22
  - reset command, 22
  - syntax, 30
  - troubleshooting, 29
  - X.21 conversion, 50
- hsi\_loop
  - test type options
    - internal test, 36
    - local/remote modem loopback, 36
    - loopback plugs, 36
    - previously set mode, 37
- hsi\_loop
  - description, 27, 34
  - man page, 23
  - options, 35
  - output, 38
  - parameters, 35
  - syntax, 34
- hsi\_stat

- description, 27, 39
- man page, 23
- output description, 40
- syntax, 39

hsi\_trace, man page, 23

## I

installing

- adapter, 7
- patch panel, 8 to 12
  - in a rack, 10
  - in a SPARCserver 690MP, 11
  - on wall, 9
- software packages, 21

integrated serial communications controller (ISCC), 70

interface signals, 2

internal clock, 4

interval parameter, 41

IOCTL parameters, 72

## K

kernel, SunVTS, 43

## L

line speed, setting, 31

## M

man pages

- bourne shell environment, 24
- C shell environment, 23
- korn shell environment, 24
- listed, 23

MANPATH variable, setting, 23 to 24

modem

- null modem cable, 45
- reporting signal changes, 32

mounting brackets

- rack, 10
- SPARCserver 690MP, 11

- wall, 9

mounting CD-ROM, 20

MRU

- setting, 33
- size, checking, 22

MTU

- setting, 33
- size, checking, 22

## N

non-return to zero (NRZ), default setting, 31

non-return to zero, inverted (NRZI), setting, 31

null modem cable

- building, 46
- illustrated, 48, 49
- requirements, 45

nylon grommet, 10

## O

ones density, 58

operating modes

- HDLC, 61
- IBM full-duplex, 62
- IBM half-duplex, 62
- IBM multi-point, 63
- SDLC, 62
- setting, 32

## P

packet

- reception, 75
- transmission, 75

patch panel

- connecting cable, 13
- illustrated, 8
- installation, 8
  - in a rack, 10
  - in a SPARCserver 690MP, 11
  - on wall, 9
- RS-449 ports, 8

phased lock loop (PLL), 58

ports

- displaying statistics, 39
- naming conventions, 28
- numbers, 28
- on-board serial port, 22
- re-initializing, 22

post-installation script, 21

power on self tests (POST), 15

protocols, supported, 1

## R

- receive data, inverting, 33
- regulatory compliance statements, iii
- re-initializing ports, 22
- removing old software versions, 19
- RS-232 to RS-449 connection, 14
- RS-334 EIA specification, 59
- RS-449
  - implementation example, 51
  - null modem cable, 45, 47
  - patch panel ports, 8
  - pin assignments, 52
  - RS-232 converter, 14
  - signal description, 52
  - signals, 47, 52

## S

script, post-installation, 21

SDLC

- described, 62
- setting, 32

sender clocking, 33

show-devs command output, 16

software

- description, 5
- device driver routines, 69
- diagnostic utilities, 5
- directory structure, 18
- functional description, 69
- initialization, 69
- installation, 17 to 21
  - mounting CD-ROM, 20
  - packages, 21
- interface, illustrated, 71

- interrupts, service routine, 74
- IOCTL parameters, 72
- man pages, 23
- network device driver, 5
- packages, 20
- packet
  - reception, 75
  - transmission, 75
- port names, 28
- removing
  - current version, 24
  - older versions, 19
  - utilities, bundled, 27

SPARCserver 690MP, installing patch panel, 11

status buffer bit assignments, 67

sunlink diagnostic, 43

SunSolve, website, 6

SunVTS

- documentation, 43
- kernel, 43
- sunlink diagnostic, 43
- tty interface, 44
- user interface, 43

support, requesting, 6

system network architecture (SNA), 62

## T

T1

- circuits, 4
- inverting
  - incoming receive clock, 32, 59
  - incoming transmit clock, 32, 59
  - receive data, 33, 58
  - transmit data, 33, 58
- requirements, 57

third-party equipment, 22

transmit data, inverting, 33

troubleshooting

- using `hsi_init`, 29
- using `hsi_loop` utility, 34

tty user interface, SunVTS, 44



## **U**

user interface, SunVTS, 43

## **V**

VCCI statements, iv

verifying hardware installation, 15

viewing man pages, 23

## **X**

X.21 signals, 47

X.21 to RS-449 converter, 50

