



Solaris on Sun Hardware Reference Manual Supplement

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900
U.S.A. 650-960-1300

Part No. 806-6085-10
October 2000, Revision A

Send comments about this document to: docfeedback@sun.com

Copyright 2000 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, California 94303-4900 U.S.A. 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. For Netscape Communicator™, the following notice applies: (c) Copyright 1995 Netscape Communications Corporation. All rights reserved.

Sun, Sun Microsystems, the Sun logo, AnswerBook2, docs.sun.com, Sun StorEdge, Enterprise Network Array, 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 2000 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, Californie 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. La notice suivante est applicable à Netscape Communicator™: (c) Copyright 1995 Netscape Communications Corporation. Tous droits réservés.

Sun, Sun Microsystems, le logo Sun, AnswerBook2, docs.sun.com, Sun StorEdge, Enterprise Network Array, 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.



Adobe PostScript

Contents

Preface v

1. User Commands (1) 1-1

cdrw 1-1

2. System Administration Commands (1M) 1M-5

envmond 1M-5

hsi_init 1M-6

hsi_loop 1M-9

hsi_stat 1M-12

hsi_trace 1M-15

hsip_init 1M-18

hsip_loop 1M-21

hsip_stat 1M-24

nf_fddidaemon 1M-27

nf_install_agents 1M-28

nf_macid 1M-29

nf_smtmon 1M-30

nf_snmd 1M-32

nf_snmd_kill 1M-34

nf_stat 1M-35

nf_sync 1M-38
pf_fddidaemon 1M-39
pf_install_agents 1M-40
pf_macid 1M-41
pf_smtmon 1M-42
pf_snmd 1M-44
pf_snmd_kill 1M-46
pf_stat 1M-47
rscadm 1M-50
sunvts 1M-54
vts_cmd 1M-55
vtsk 1M-61
vtsprobe 1M-62
vtstty 1M-65
vtsui 1M-67
vtsui.ol 1M-68

3. File Formats (4) 4-69

environ.conf 4-69

4. Device and Network Interfaces (7) 7-71

ge 7-71
hsi 7-75
hsip 7-79
nf 7-83
pf 7-86
smt 7-91

Preface

The *Solaris on Sun Hardware Reference Manual Supplement* contains reference manual pages (man pages) for software provided to Sun hardware customers with the Solaris 8 product. These supplement the man pages provided in the general *Solaris 8 Reference Manual*. This edition has been updated to include man pages found in the Solaris 8 10/00 release.

Before you can access some of the information published in this book through the man command, you may need to install software from the Sun Microsystems Computer Systems Supplement CD for your Solaris release. In most cases, when you install a software product from the Sun Microsystems Computer Systems Supplement CD, a package containing man pages about the software will be automatically installed. For information about installing the man page software, refer to the *Solaris 8 Sun Hardware Platform Guide*.

How This Book Is Organized

This manual contains man pages in alphabetical order within each category: User Commands (1)

- System Administration Commands (1M)
- File Formats (4)
- Device and Network Interfaces (7)

The man pages apply to the following products:

- CD Read/Write drives: cdrw
- SunFDDI™ network adapter software: nf, nf_fddidaemon, nf_install_agents, nf_macid, nf_smtmon, nf_snmd, nf_snmd_kill, nf_stat, nf_sync, pf, pf_fddidaemon, pf_install_agents, pf_macid, pf_smtmon, pf_snmd, pf_snmd_kill, pf_stat, smt

- SunHSI/P™ (PCI bus) network adapter software: `hsip`, `hsip_init`, `hsip_loop`, `hsip_stat`
- SunHSI/S™ (Sbus) network adapter software: `hsi`, `hsi_init`, `hsi_loop`, `hsi_stat`, `hsi_trace`
- Sun Remote System Control (RSC): `rscadm`
- SunVTS™ diagnostic software: `sunvts`, `vts_cmd`, `vtsk`, `vtsprobe`, `vtstty`, `vtsui`, `vtsui.ol`
- Netra™ t server environmental monitoring software: `envmond`, `envmond.conf`

Ordering Sun Documentation

Fatbrain.com, an Internet professional bookstore, stocks select product documentation from Sun Microsystems, Inc. For a list of documents and how to order them, visit the Sun Documentation Center on Fatbrain.com at:

<http://www1.fatbrain.com/documentation/sun>

Accessing Sun Documentation Online

The `docs.sun.com`SM 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 (806-6085-10) of your document in the subject line of your email.

NAME	cdrw – CD read and write
SYNOPSIS	<pre> cdrw -i [-hvSCO] [-d device] [-p speed] [<i>image-file</i>] cdrw -a [-hvSCO] [-d device] [-p speed] [-T audio-type] <i>audio-file1</i> [<i>audio-file2 ...</i>] cdrw -x [-hv] [-d device] [-T audio-type] <i>track-number out-file</i> cdrw -c [-hvSC] [-d device] [-p speed] [-m tmp-dir] [-s src-device] cdrw -b [-hv] [-d device] all session cdrw -M [-hv] [-d device] cdrw -l [-hv] </pre>
AVAILABILITY	SUNWcdrw
DESCRIPTION	<p>The cdrw command provides the ability to create data and audio CDs. It also provides the ability to extract audio tracks from an audio CD. Any MMC-compliant CD-R or CD-RW drive can be used with cdrw.</p> <p>cdrw will search for a CD writer device connected to the system, unless the user specifies a device with the -d option. If it finds a single such writer device, it will use that as the default CD writer device for the command.</p> <p>When more than one CD writer is connected to the system, use the -d option to indicate which device is desired. The device name can be specified in one of the following ways: <code>/dev/rdsk/cNtNdNsN</code>, <code>cNtNdNsN</code>, <code>cNtNdN</code>, or a symbolic name used by volume manager, such as <code>cdrom</code> or <code>cdrom1</code>. The -l option will provide a list of CD writers.</p>
Creating Data CDs	<p>When creating data CDs, cdrw uses the track-at-once mode of writing. With the -i option, the user will specify a file that contains the data to write on CD media. In the absence of such a file, cdrw will read data from standard input.</p> <p>In either case, the data will typically first have been prepared by using the mkisofs (1M) command to convert the file and file information into the High Sierra format used on CDs. See the examples that include use of this command.</p>
Creating Audio CDs	<p>For creating an audio CD, using the -a option, single or multiple audio files can be specified. All of the audio files should be in the supported audio formats. Currently approved formats are:</p> <ul style="list-style-type: none"> sun – Sun .au files with data in Red Book CDDA form wav – RIFF (.wav) files with data in Red Book CDDA form cda – .cda files having raw CD audio data (i.e., 16 bit PCM stereo at 44.1 KHz sample rate in little-endian byteorder)

aur – .aur files having raw CD data in big-endian byteorder

If no audio format is specified, **cdrw** tries to understand the audio file format based on the file extension. The case of the characters in the extension is ignored. If a format is specified using **-T**, it will be assumed as the audio file type for all the files specified. Also, **cdrw** will close the session after writing the audio tracks; therefore, the tracks to be written should be specified in a single command line.

Extracting Audio

cdrw can also be used for extracting audio data from an audio CD with the **-x** option. The CD should have tracks in Red Book CDDA form. By default the output format is based on the file extension. A user can specify a sun, wav, cda, au, or aur output format using the **-T** option.

Copying CDs

cdrw can be used to copy single session data CD-ROMs and Red Book audio CDs. For copying a CD, **cdrw** looks for a specified source device. If no source device is specified when using the **-c** option, the current CD writing device is assumed to be the source. **cdrw** will extract the track or tracks into a temporary file and will look for a blank writable CD-R/RW media in the current CD writing device. If no such media is found, the user will be asked to insert a blank writable CD media in the current CD writing device. If enough space is not available in the default temporary directory, an alternative directory can be specified using the **-m** option.

Erasing CD-RW Media

Users have to erase the CD-RW media before it can be re-written. With the **-b** option, currently, the following flavors of erasing are supported:

session – Erase the last session

all – Erase the entire media

If the **session** erasing type is used, **cdrw** will erase the last session. If there is only one session recorded on the CD-RW (e.g., a data/audio CD-RW created by this tool), then session erasing is useful as it will only erase the portion that is recorded, leaving behind a blank disc; this is faster than erasing the entire media.

The **all** erasing type should be used if it is a multisession disc, or the last session is not closed, or disc status is unknown, and the user wishes to erase the disc. With this type of erase, **cdrw** will erase the entire disc.

Checking device-list/media-status

The user can get a list of CD writing devices currently present in the system with the **-I** option. Also, for a particular media, the user can get the blanking status and table of contents through the **-M** option. **-M** also prints information about the last session start address and the next writable address. This information along with the **-O** option can be used to create multisession CDs. Please refer to **mkisofs(1M)** for more information.

OPTIONS

-a Create an audio disc. At least one *audio-file* name must be specified. A CD can not have more than 99 audio tracks, so no more than 99 audio files can be specified. Also, the maximum audio data that can be written to the media by default is 74 minutes, unless **-C** is specified.

- b Blank a CD-RW media. The type of erasing must be specified by the **all** or **session** argument.
- c Copy a CD. If no other argument is specified, the default CD writing device is assumed to be the source device as well. In this case, the copying operation will read the source media into a temporary directory and will prompt the user to place a blank media into the drive for copying to proceed.
- C Use media stated capacity. Without this option, **cdrw** will use a default value for writable CD media, which is 74 minutes for an audio CD or 681984000 bytes for a data CD.
- d CD writing device.
- h Help, which prints usage message.
- i Image file for creating data CDs. The file size should be less than what can be written on a CD-R or CD-RW media, which is 681984000 bytes by default or the media stated capacity if the **-C** option is used. Also, it is better to have the file locally available instead of having it on an NFS mounted filesystem, because the CD writing process expects data to be available continuously without interruptions.
- l List all the CD writers found in the system.
- m Use an alternate temporary directory instead of system default temporary directory for storing track data while copying a CD. An alternate temporary directory might be required because the amount of data on a CD can be huge (as much as 800 Mbytes for an 80 minute audio CD) and the system default temporary directory might not have that much space.
- M Report media status. **cdrw** will report if the media is blank or not, its table of contents, the last session's start address, and the next writable address if the disc is open.
- O Keep the disc open. **cdrw** will close the session, but it will keep the disc open so that another session can be added later on to create a multisession disc.
- p Set the CD writing speed; e.g., **-p 4** will set the speed to 4X. If this option is not specified, **cdrw** will use the default speed of the CD writer. If this option is specified, **cdrw** will try to set the drive write speed to this value, but there is no guarantee of the speed actually used by the drive.
- s Source device for copying CD.
- S Simulation mode. In this mode, **cdrw** will do everything with the drive laser turned off, so nothing will be written to the media. This can be used to verify if the system can provide data at a rate good enough for CD writing.
- T Audio format to use extracting audio files or reading audio files for audio CD creation. The *audio-type* can be sun, wav, cda, au, or aur.
- v Verbose mode.
- x Extract audio data from an audio track.

EXAMPLES

Example 1: Creating a data CD.

```
cdrw -i /local/iso_image
```

Example 2: Creating a CD from the directory tree /home/foo.

```
$ mkisofs -r /home/foo 2>/dev/null | cdrw -i -p 1
```

Example 3: Extracting audio track number 1 to /home/foo/song1.wav.

```
$ cdrw -x -T wav 1 /home/foo/song1.wav
```

Example 4: Creating an audio CD from wav files on disc.

```
$ cdrw -a song1.wav song2.wav song3.wav song4.wav
```

Example 5: Erasing a CD-RW media in a CD-RW drive.

```
$ cdrw -b all
```

Example 6: Creating a data CD on a system with multiple CD-R/RW drives.

```
$ cdrw -d c1t6d0s2 -i /home/foo/iso-image
```

Example 7: Checking if the system can provide data to a CD-RW drive at a rate sufficient for write operation.

```
$ cdrw -S -i /home/foo/iso-image
```

Example 8: Running **cdrw** at a higher priority (for root user only).

```
$ priocntl -e -p 60 cdrw -i /home/foo/iso-image
```

SEE ALSO

mkisofs(1M), **audioconvert(1)**

NOTES

The CD writing process requires data to be supplied at a constant rate to the drive. It is advised to keep I/O activity to a minimum and shut down the related applications while writing CDs.

When making copies or extracting audio tracks, it is better to use an MMC compliant source CD-ROM drive. The CD writing device can be used for this purpose.

Before writing a CD, it is best to ensure that the media is blank by using the **-M** option and to test that the system can provide data at the required rate by using the **-S** simulation mode. In case the system is not able to provide data at the required rate, try simulation with a slower write speed set through the **-p** option. Users can also try to run **cdrw** at a higher priority using the **priocntl(1)** command.

The **-p** option is provided for users who are aware of the CD-R/RW drive and its capabilities to operate at different write speeds. Some commercially available drives handle the drive speed setting command differently, so use this option judiciously.

Most commercially available drives allow writing beyond 74 minutes as long as the media has the capacity (such as 80 minute media). However, such capability of writing beyond 74 minutes might not be supported by the drive in use. If the drive being used supports such capability, then use the **-C** option to indicate that the tool should rely on the capacity indicated by the media.

NAME	envmond - environmental monitor daemon
SYNOPSIS	<code>/usr/platform/SUNW,UltraSPARC-IIi-Netractal/lib/envmond/sparcv9/envmond [-d] [-f file] [-g granularity]</code>
AVAILABILITY	SUNWcteux
DESCRIPTION	<p>The envmond daemon polls system environment monitoring devices to check for conditions that may require corrective action. In order to do this, the daemon reads a configuration file on startup, during the initial Solaris boot process, to find out which environmental devices will be monitored. Each configuration file entry describing an environmental device is referred to as a policy, and the supported policy entries are described in envmond.conf(4).</p> <p>The envmond daemon logs appropriate messages to a system log file via syslogd(1M). The envmond daemon will reread its configuration information file whenever it receives a hang-up signal, SIGHUP.</p>
OPTIONS	<p>-d Sets Debug mode option. The envmond will not run as a daemon, and will instead run in the foreground, inheriting standard input and output. Error and warning messages will be written to the standard output instead of being logged via syslogd(1M).</p> <p>-f file Provides an alternate file path for the configuration file.</p> <p>-g granularity Defines the finest granularity for the poll interval. The default value is 10 seconds.</p>
FILES	<p><code>/usr/platform/SUNW,UltraSPARC-IIi-Netractal/lib/envmond/sparcv9/envmond</code> The executable daemon</p> <p><code>/usr/platform/SUNW,UltraSPARC-IIi-Netractal/lib/envmond/sparcv9/*.so</code> The envmond policies</p> <p><code>/platform/SUNW,UltraSPARC-IIi-Netractal/lib/envmond.conf</code> The envmond configuration file</p>
SEE ALSO	syslogd (1M), envmond.conf (4)
NOTES	<p>The envmond policies retrieve their environmental information via I2C devices in the system.</p> <p>This daemon is in the PROTOTYPE stage, and is therefore subject to CHANGE WITHOUT NOTICE.</p>

NAME hsi_init – set high speed serial line interface operating parameters.

SYNOPSIS /opt/SUNWconn/bin/hsi_init device [[baud_rate] | [keyword=value, ...] | [single-word option]]

DESCRIPTION The hsi_init utility allows the user to modify some of the hardware operating modes common to high speed synchronous serial lines. This may be useful in troubleshooting a link, or necessary to the operation of a communications package.
If run without options, hsi_init reports the options as presently set on the port. If options are specified, the new settings are reported after they have been made.

OPTIONS Options to hsi_init normally take the form of a keyword, followed by an equal sign and a value. The exception is that a baud rate may be specified as a decimal integer by itself. Keywords must begin with the value shown in the options table, but may contain additional letters up to the equal sign. For example, "loop=" and "loopback=" are equivalent.
Recognized options are listed in the table below.

Keyword	Value	Effect
loopback	no	Disable internal loopback mode. If no other clocking options have been specified, perform the equivalent of txc=txc and rxc=rxc .
	yes	Set the port to operate in internal loopback mode. The receiver is electrically disconnected from the DCE receive data input and tied to the outgoing transmit data line. Transmit data is available to the DCE. If no other clocking options have been specified, perform the equivalent of txc=baud and rxc=baud .
nrzi	no	Set the port to operate with NRZ data encoding. NRZ encoding maintains a constant voltage level when data is present (1) and does not return to a zero voltage (0) until data is absent. The data is decoded as an absolute value based on the voltage level (0 or 1).
	yes	Set the port to operate with NRZI data encoding. NRZI encoding does a voltage transition when data is absent (0) and no voltage transition (no return to zero) when data is present (1). Hence, the name non-return to zero inverted. The data is decoded using relational decoding.
txc	txc	Transmit clock source will be the TxCI signal.
	-txc	Transmit clock source will be the inverted TxCI signal.
	rxc	Transmit clock source will be the RxC signal.
	baud	Transmit clock source will be the internal baud rate generator .

rx	rx	Receive clock source will be the RxC signal.
	-rx	Receive clock source will be the inverted RxC signal.
	baud	Receive clock source will be the internal baud rate generator .
mode	fdx	HDLC Full Duplex mode (Default mode).
	ibm-fdx	IBM Full Duplex mode (SDLC).
	ibm-hdx	IBM Half Duplex mode (SDLC).
	ibm-mpt	IBM Multipoint mode (SDLC).
signal	yes	Notify application of modem signal (RTS and CTS) changes.
	no	Don't notify application of modem signal (RTS and CTS) changes.
speed	<i>integer</i>	Set the baud rate to <i>integer</i> bits per second. The speed can be set from 300 bps to 2048000 bps.
mtu		Set the Maximum Transmission Unit. This is the packet size that is transmitted. The maximum mtu is 1600 bytes.
mru		Set the Maximum Receive Unit. This is the packet size that is received. The maximum mru is 1600 bytes.
txd		This flag is used for inverting transmit data on serial lines. You can switch the polarity of a link by setting this flag to be negative, i.e. -txd.
rx		This flag is used for inverting receive data on serial lines. You can switch the polarity of a link by setting this flag to be negative, i.e. -rx.
reset		Resets the board. Terminates all incoming and outgoing traffic.

There are also several single-word options that set one or more parameters at a time:

Keyword	Equivalent to Options:
external	txc=txc rxc=rx loop=no
sender	txc=baud rxc=rx loop=no
stop	speed=0

EXAMPLES

The following command sets the first CPU port to loop internally, use internal clocking and operate at 38400 bps:

```
example# hsi_init hih0 38400 loop=yes
port=hih0 speed=38309, mode=fdx, loopback=yes, nrzi=no, mtu=1600,
mru=1600, txc=baud, rxc=baud, txd=txd, rxd=rx, signal=no.
```

The following command sets the same port's clocking, local loopback and bit rate settings to their default values:

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

SEE ALSO

hsi_loop(1M), hsi_stat(1M), hsi_trace(1M), Intro(2), hsi(7D)

DIAGNOSTICS**device missing minor device number**

The name *device* does not end in a decimal number that can be used as a minor device number.

bad speed: arg

The string *arg* that accompanied the "speed=" option could not be interpreted as a decimal integer.

Bad arg: arg

The string *arg* did not make sense as an option.

ioctl failure code = errno

An **ioctl(2)** system call failed. The meaning of the value of *errno* may be found in the **Intro(2)** manual page.

WARNINGS

hsi_init should not be used on an active serial link, unless needed to resolve an error condition. It should not be run casually, or if the user is unsure of the consequences of its use.

NAME	hsi_loop – high speed synchronous serial loopback test program for high speed serial interface.														
SYNOPSIS	<i>/opt/SUNWconn/bin/hsi_loop [-cdlsvt] device</i>														
DESCRIPTION	<p>The hsi_loop command performs several loopback tests that are useful in exercising the various components of a serial communications link.</p> <p>Before running a test, hsi_loop opens the designated port and configures it according to command line options and the specified test type. It announces the names of the devices being used to control the hardware channel, the channel number (ppa) corresponding to the <i>device</i> argument, and the parameters it has set for that channel. It then runs the loopback test in three phases.</p> <p>The first phase is to listen on the port for any activity. If no activity is seen for at least four seconds, hsi_loop proceeds to the next phase. Otherwise, the user is informed that the line is active and that the test cannot proceed, and the program exits.</p> <p>In the second phase, called the "first-packet" phase, hsi_loop attempts to send and receive one packet. The program will wait for up to four seconds for the returned packet. If no packets are seen after five attempts, the test fails with an error message. If a packet is returned, the result is compared with the original. If the length and content do not match exactly, the test fails.</p> <p>The final phase, known as the "multiple-packet" phase, attempts to send many packets through the loop. Because the program has verified the integrity of the link in the first-packet phase, the test will not fail after a particular number of timeouts. If a packet is not seen after four seconds, a message is displayed. Otherwise, a count of the number of packets received is updated on the display once per second. If it becomes obvious that the test is not receiving packets during this phase, the user may wish to stop the program manually. The number and size of the packets sent during this phase is determined by default values, or by command line options. Each returned packet is compared with its original for length and content. If a mismatch is detected, the test fails. The test completes when the required number of packets have been sent, regardless of errors.</p> <p>After the multiple-packet phase has completed, the program displays a summary of the hardware event statistics for the channel that was tested. The display takes the following form:</p> <table border="0" style="margin-left: 20px;"> <tr> <td>Port</td> <td>CRC errors</td> <td>Aborts</td> <td>Overruns</td> <td>Underruns</td> <td>In <-Drops-></td> <td>Out</td> </tr> <tr> <td>hih0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> </table> <p>This is followed by an estimated line speed, which is an approximation of the bit rate of the line, based on the number of bytes sent and the actual time that it took to send them. This is a very rough approximation and should not be used in bechmarking, because elapsed time includes time to print to the display.</p>	Port	CRC errors	Aborts	Overruns	Underruns	In <-Drops->	Out	hih0	0	0	0	0	0	0
Port	CRC errors	Aborts	Overruns	Underruns	In <-Drops->	Out									
hih0	0	0	0	0	0	0									

OPTIONS

The options for hsi_loop are described in the following table:

Option	Parameter	Default	Description
-c	<i>packet_count</i>	100	Specifies the number of packets to be sent in the multiple-packet phase.
-d	<i>hex_data_byte</i>	<i>random</i>	Specifies that each packet will be filled with bytes with the value of <i>hex_data_byte</i> .
-l	<i>packet_length</i>	100	Specifies the length of each packet in bytes.
-s	<i>line_speed</i>	9600	Bit rate in bits per second.
-v			Sets verbose mode. If data errors occur, the expected and received data is displayed.
-t	<i>test_type</i>	<i>none</i>	A number, from 1 to 4, that specifies which test to perform. The values for <i>test_type</i> are as follows: <ol style="list-style-type: none"> 1 Internal loopback test. Port loopback is on. Transmit and receive clock sources are internal (baud rate generator). 2 External loopback test. Port loopback is off. Transmit and receive clock sources are internal. Requires a loopback plug suitable to the port under test. 3 External loopback test. Port loopback is off. Transmit and receive clock sources are external (modem). Requires that one of the local modem, the remote modem, or the remote system (not a Sun) be set in a loopback configuration. 4 Test using predefined parameters. User defines hardware configuration and may select port parameters using the hsi_init(1M) command.

All numeric options except -d are entered as decimal numbers (for example, -s 19200). If you do not provide the -t test_type option, hsi_loop prompts for it.

EXAMPLES

The following command causes hsi_loop to use a packet length of 512 bytes over the first CPU port:

example# hsi_loop -l 512 hih0

In response to the above command, hsi_loop prompts you for the test option you want.

The following command performs an internal loopback test on the first CPU port, using 5000 packets and a bit rate of 56Kbps :

example# hsi_loop -t 1 -s 56000 -c 5000 hih0

SEE ALSO

hsi_init(1M), hsi_stat(1M), hsi_trace(1M), hsi(7d)

DIAGNOSTICS

device missing minor device number

The name *device* does not end in a decimal number that can be used as a minor device number.

invalid packet length: *nnn*

The packet length was specified to be less than zero or greater than 1600.

poll: nothing to read

poll: nothing to read or write.

The **poll(2)** system call indicates that there is no input pending and/or that output would be blocked if attempted.

len *xxx* should be *yyy*

The packet that was sent had a length of *yyy*, but was received with a length of *xxx*.

***nnn* packets lost in outbound queueing**

***nnn* packets lost in inbound queueing**

A discrepancy has been found between the number of packets sent by *hsi_loop* and the number of packets the driver counted as transmitted, or between the number counted as received and the number read by the program.

WARNINGS

To allow its tests to run properly, as well as prevent disturbance of normal operations, *hsi_loop* should only be run on a port that is not being used for any other purpose at that time.

NAME	hsi_stat – report driver statistics from a high speed synchronous serial link port.																								
SYNOPSIS	<pre> /opt/SUNWconn/bin/hsi_stat [-f] -a num_of_ports /opt/SUNWconn/bin/hsi_stat -c [-f] -a num_of_ports /opt/SUNWconn/bin/hsi_stat [-f] device [period] /opt/SUNWconn/bin/hsi_stat -c [-f] device </pre>																								
DESCRIPTION	<p>The hsi_stat command reports the event statistics maintained by a high speed synchronous serial device driver. The report may be a single snapshot of the accumulated totals, or a series of samples showing incremental changes.</p> <p>Event statistics are maintained by a driver for each physical channel that it supports. They are initialized to zero at the time the driver module is loaded into the system when one of the driver's entry points is first called.</p> <p>The device argument is the name of the high speed serial device as it appears in the /dev directory. For example, hih0 specifies the first on-board high speed serial device.</p> <p>As an alternative, you can display or clear the statistics for multiple physical channels using num_of_ports argument. The hsi_stat program will then display statistics accumulated from device hih0 to hih(num_of_ports - 1). Additionally, statistics for all ports can be displayed or cleared by the use of the -a option. In this case, the command will be issued for all the ports on the system. This option is not available for sampling purposes.</p> <p>The following is a breakdown of hsi_stat output:</p> <table border="0"> <tr> <td style="vertical-align: top;">speed</td> <td>The line speed the device has been set to operate at. It is the user's responsibility to make this value correspond to the modem clocking speed when clocking is provided by the modem.</td> </tr> <tr> <td style="vertical-align: top;">ipkts</td> <td>The total number of input packets.</td> </tr> <tr> <td style="vertical-align: top;">opkts</td> <td>The total number of output packets.</td> </tr> <tr> <td style="vertical-align: top;">undrun</td> <td>The number of transmitter underrun errors.</td> </tr> <tr> <td style="vertical-align: top;">ovrrun</td> <td>The number of receiver overrun errors.</td> </tr> <tr> <td style="vertical-align: top;">abort</td> <td>The number of aborted received frames.</td> </tr> <tr> <td style="vertical-align: top;">crc</td> <td>The number of received frames with CRC errors.</td> </tr> <tr> <td style="vertical-align: top;">isize</td> <td>The average size (in bytes) of input packets.</td> </tr> <tr> <td style="vertical-align: top;">osize</td> <td>The average size (in bytes) of output packets.</td> </tr> <tr> <td style="vertical-align: top;">iutil</td> <td>Reports the input line utilization expressed as a percentage.</td> </tr> <tr> <td style="vertical-align: top;">outil</td> <td>Reports the output line utilization expressed as a percentage.</td> </tr> </table> <p>Additional fields for the 'f' flag are listed below.</p> <table border="0"> <tr> <td style="vertical-align: top;">ierror</td> <td>Reports the input error count. Errors can be incomplete frames, empty frames, or receive clock (RxC) problems.</td> </tr> </table>	speed	The line speed the device has been set to operate at. It is the user's responsibility to make this value correspond to the modem clocking speed when clocking is provided by the modem.	ipkts	The total number of input packets.	opkts	The total number of output packets.	undrun	The number of transmitter underrun errors.	ovrrun	The number of receiver overrun errors.	abort	The number of aborted received frames.	crc	The number of received frames with CRC errors.	isize	The average size (in bytes) of input packets.	osize	The average size (in bytes) of output packets.	iutil	Reports the input line utilization expressed as a percentage.	outil	Reports the output line utilization expressed as a percentage.	ierror	Reports the input error count. Errors can be incomplete frames, empty frames, or receive clock (RxC) problems.
speed	The line speed the device has been set to operate at. It is the user's responsibility to make this value correspond to the modem clocking speed when clocking is provided by the modem.																								
ipkts	The total number of input packets.																								
opkts	The total number of output packets.																								
undrun	The number of transmitter underrun errors.																								
ovrrun	The number of receiver overrun errors.																								
abort	The number of aborted received frames.																								
crc	The number of received frames with CRC errors.																								
isize	The average size (in bytes) of input packets.																								
osize	The average size (in bytes) of output packets.																								
iutil	Reports the input line utilization expressed as a percentage.																								
outil	Reports the output line utilization expressed as a percentage.																								
ierror	Reports the input error count. Errors can be incomplete frames, empty frames, or receive clock (RxC) problems.																								

inactiv	Reports the number of input packets received when receive is inactive.
ishort	Reports the number of short input packets. This is the number of input packets 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.
oerror	Reports the output error count. Errors that can be lost are clear to send (CTS) signals or transmit clock (TxC) problems.
olong	Reports the number of long output packets. This is the number of output packets with lengths with lengths larger than the MTU.
ohung	Reports the number of times the transmitter hangs, which is usually due to a missing clock.

OPTIONS

- f** Select full set of accumulated statistics for the device specified. This is useful while debugging the **hsi** driver.
- c** Clear the accumulated statistics for the device specified. This may be useful when it is not desirable to unload a particular driver, or when the driver is not capable of being unloaded.
- num_of_ports** Specify the number of devices that you want to dump the statistics.
- a** Specify all of the ports in the system, regardless of the number of HSI boards.
- interval** Cause **hsi_stat** to sample the statistics every *interval* seconds and report incremental changes. The output reports line utilization for input and output in place of average packet sizes. These are the relationships between bytes transferred and the baud rate, expressed as percentages. The loop repeats indefinitely, with a column heading printed every twenty lines for convenience.

EXAMPLES

```
example# hsi_stat hih0
speed  ipkts  opkts  undrun  ovrrun  abort  crc  isize
9600   15716  17121   0        0        1     3    98

example# hsi_stat 5
speed  ipkts  opkts  undrun  ovrrun  abort  crc  isize
hih0   9600  15716  10100   0        0     1     3
hih1   9600  15234  20100   0        0     1     3
hih2   9600  15123  18254   0        0     1     3
hih3   9600  15378  18234   0        0     1     3

example# hsi_stat -a
speed  ipkts  opkts  undrun  ovrrun  abort  crc  isize  osize
```

```

hih0  9600  15716  10100    0    0    1    3    98
hih1  9600  15234  20100    0    0    1    3    98
hih2  9600  15123  18254    0    0    1    3    98
hih3  9600  15378  18234    0    0    1    3    98
hih4  9600  13900  13000    0    0    1    3    98
hih5  9600  15218  13100    0    0    1    3    98
hih6  9600  15737  22100    0    0    1    3    98
hih7  9600  15143  11254    0    0    1    3    98
    
```

example# **hsi_stat -c hih0**

```

speed  ipkts  opkts  undrun  ovrrun  abort  crc  isize  osize
9600   0     0     0       0       0     0   0     0
    
```

example# **hsi_stat hih0 5**

```

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%
    
```

(In this final example a new line of output is generated every five seconds.)

SEE ALSO
DIAGNOSTICS

hsi_init(1M), hsi_loop(1M), hsi_trace(1M), hsi(7D)

device **missing minor device number**

The name *device* does not end in a decimal number that can be used as a minor device number.

hsi_stat: *Can't sample multiple ports simultaneously.*

Sampling is only available with one specified port, i.e. `hsi_stat hih0 10`.

WARNINGS

Underrun, overrun, frame-abort and CRC errors have a variety of causes. Communication protocols are typically able to handle such errors and initiate recovery of the transmission in which the error occurred. Small numbers of such errors are not a significant problem for most protocols. However, because the overhead involved in recovering from a link error can be much greater than that of normal operation, high error rates can greatly degrade overall link throughput. High error rates are often caused by problems in the link hardware, such as cables, connectors, interface electronics or telephone lines. They may also be related to excessive load on the link or the supporting system.

The percentages for input and output line utilization reported when using the *interval* option 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 baud rate set for the device probably does not reflect the speed of the modem.

NAME	hsi_trace – Dump and Parse the HSI/S driver trace buffer. This is a development/field support only diagnostic utility.
SYNOPSIS	/opt/SUNWconn/bin/hsi_trace
DESCRIPTION	<p>hsi_trace utility id for support and field personnel only. This utility prints out the trace of the incoming and outgoing packets at the hsi driver level.</p> <p>There are two levels of traces that can be captured. This is controlled by setting a variable in the driver in the /etc/system file.</p> <pre>set HSI:hsi_trace=1</pre> <p>The driver maintains an internal circular buffer to store 24K frames (both in and out).</p> <p>Then run hsi_trace on the driver to collect the trace data.</p> <pre># hsi_trace > hsi_trace.log</pre> <p>This trace is useful when the problem occurs rarely (typically a week or so) and we do not have enough file system space.</p> <p>This trace collects the last 24K of frame data.</p> <p>Then there is another trace 'strace' which can be used to collect all the data from the driver. This can be enabled by setting 'hsi_trace' as</p> <pre>set HSI:hsi_trace=2</pre> <p>Then run</p> <pre>#strace 18515 all all > hsi_trace.log</pre> <p>This collects all the data from the driver. This trace is useful when we know that the problem occurs within a short time.</p> <p>The trace output is as follows</p> <p>In the first case ('hsi_trace' utility)</p> <pre>13:26:38 0000004f hih9 len=0100 R: 31323334 35363738 fm: I-FR P/F=1 Nr=1 Ns=1</pre> <p>The fields are as follows</p> <p>1 st field: Time stamp</p> <p>2 nd field: time difference in microsecs between the last frame and current frame.</p> <p>3 rd field: port</p> <p>4 th field: length of the frame.</p> <p>5 th field: R: received data T: transmitted data</p> <p>6 th and 7 th field: First 8 bytes of the data transmitted or received.</p> <p>7 th field: The frame type (SABM, TEST, XID, RR, RNR....)</p>

Some of the frame types are described below.

Keyword	Value	Effect
RR	Receive Ready	This frame is used as a polling command by the primary station to solicit information frames from the secondary station.
RNR	Receive Not Ready	This frame is used as a flow control command or response to indicate that the station transmitting the Receive Not Ready frame is not able to accept any information frames at this time.
REJ	Reject	This frame is sent by a station to indicate that it has received a frame out of the normal sequence. This may indicate the loss of an information frame containing user data.
SABM	Set Async Balanced Mode	An LLC non-data frame requesting the establishment of a connection over which numbered information frames may be sent.
SNRM	Set Normal Response Mode	This command is sent from the primary station to a secondary station to place the secondary in the initialized normal SDLC operating mode.
SNRME	SNRM Extended	SNRM with two more bytes in the control field. Used in SDLC.
DISC	Disconnect	This command is sent from the primary station to the secondary station to place the secondary station in the off-line disconnected mode.
SIM	Set Initialization Mode	This command is sent from the primary station to the secondary station to begin the initialization process.
UA	Unnumbered Ack	This response is sent from the secondary station to the primary station in response to an SNRM, DISC, or SIM command.
DM	Disconnect Mode	This response is sent from the secondary station to the primary station in response to any command other than SNRM or DISC.
RD	Request Disconnect	This response is sent from the secondary to the primary station to request that the secondary station be placed in the off-line or disconnect mode.
RIM	Req Init Mode	This response is sent from the secondary to the primary station to request initialization.

FRMR	Frame Reject	This response is sent from the secondary station to the primary station to indicate that an abnormal condition has been detected or that an invalid frame has been received. It contains bits which indicate the reason for the rejection of the frame.
XID	Exchange Identification	This frame may be either a command sent by the primary station or a response sent by the secondary station. It contains information that is used to identify the secondary station.
TEST	TEST	This command is sent from the primary station to the secondary station and may contain some form of a message that may be used to test the secondary's ability to receive data and transmit the data back to the primary station.
UI	Unnumbered Information	This command allows the primary station to send data to the secondary station and the unnumbered information response allows the secondary station to send data to the primary station.
INFO	Information	This frame contains the information and data relevant to the higher SNA architecture layers. INFO frames consist of several variable-length or optional fields, depending upon the implementation.
UP	unnumbered Poll frame	Used by a primary to poll a secondary.
BCN	Beacon	This is a beacon frame which is usually an indication of a problem.
CFGR	Configure	This is a configuration frame.

'strace' is the normal unix strace output.

```
020809 13:34:31 001c1330 0 ... 18515 0 hih8 len=0100 T: 31323334 35363738 fm: I-FR
P/F=1 Nr=1 Ns=1
```

SEE ALSO **hsi_init(1M), hsi_stat(1M), hsi_loop(1M), hsi(7d)**

DIAGNOSTICS

NAME hsip_init – set high speed serial line interface operating parameters.

SYNOPSIS /opt/SUNWconn/bin/hsip_init device [[baud_rate] | [keyword=value, ...] | [single-word option]]

DESCRIPTION The hsip_init utility allows the user to modify some of the hardware operating modes common to high speed synchronous serial lines. This may be useful in troubleshooting a link, or necessary to the operation of a communications package.

If run without options, hsip_init reports the options as presently set on the port. If options are specified, the new settings are reported after they have been made.

OPTIONS Options to hsip_init normally take the form of a keyword, followed by an equal sign and a value. The exception is that a baud rate may be specified as a decimal integer by itself. Keywords must begin with the value shown in the options table, but may contain additional letters up to the equal sign. For example, "loop=" and "loopback=" are equivalent.

Recognized options are listed in the table below.

Keyword	Value	Effect
loopback	yes	Set the port to operate in internal loopback mode. The receiver is electrically disconnected from the DCE receive data input and tied to the outgoing transmit data line. Transmit data is available to the DCE. If no other clocking options have been specified, perform the equivalent of txc=baud and rxr=baud .
	no	Disable internal loopback mode. If no other clocking options have been specified, perform the equivalent of txc=txc and rxr=rxr .
	echo	Set the port to operate in auto-echo mode. The port will echo incoming receive data on the transmit data pin. When the loopback is set for echo and no clocking option is given the clocking is set txc=txc and rxr=rxr. Other clocking options can be used but line errors may occur due to the loopback=echo implementation.
nrzi	no	Set the port to operate with NRZ data encoding. NRZ encoding maintains a constant voltage level when data is present (1) and does not return to a zero voltage (0) until data is absent. The data is decoded as an absolute value based on the voltage level (0 or 1).

	yes	Set the port to operate with NRZI data encoding. NRZI encoding does a voltage transition when data is absent (0) and no voltage transition (no return to zero) when data is present (1). Hence, the name non-return to zero inverted. The data is decoded using relational decoding.
txc	txc	Transmit clock source will be the TxCI signal.
	rxc	Transmit clock source will be the RxC signal.
	baud	Transmit clock source will be the internal baud rate generator .
	pll	Transmit clock source will be the output of the DPLL circuit. This can only be set with NRZI data encoding.
	-txc	Transmit clock source will be the inverted TxCI signal.
rxc	rxc	Receive clock source will be the RxC signal.
	txc	Receive clock source will be the TxCI signal. This can only be used with transmit clock option txc=txc.
	baud	Receive clock source will be the internal baud rate generator .
	pll	Receive clock source will be the output of the DPLL circuit. This can only be set with NRZI data encoding.
	-rxc	Receive clock source will be the inverted RxC signal.
txd	txd	Transmit data is not inverted.
	-txd	Transmit data is inverted.
rxd	rxd	Receive data is not inverted.
	-rxd	Receive data is inverted.
mode	fdx	HDLC Full Duplex mode (Default mode).
	ibm-fdx	IBM Full Duplex mode (SDLC).
	ibm-hdx	IBM Half Duplex mode (SDLC).
	ibm-mpt	IBM Multipoint mode (SDLC).
signal	yes	Notify application of modem signal (RTS and CTS) changes.
	no	Do not notify application of modem signal (RTS and CTS) changes.
mtu	<i>integer</i>	Set the maximum transmit unit to <i>integer</i> bytes with 2064 bytes maximum.
mrु	<i>integer</i>	Set the maximum receive unit to <i>integer</i> bytes with 2064 bytes maximum.
speed	<i>integer</i>	Set the baud rate to <i>integer</i> bits per second with a minimum rate of 9600 bps and a maximum of 2048000 bps. Zero is also valid when txc is set to txc or -txc.

There are also several single-word options that set one or more parameters at a time:

Keyword	Equivalent to Options:
external	txc=txc rxc=rxs loop=no
sender	txc=baud rxc=rxs loop=no
internal	txc=pll rxc=pll loop=no
stop	speed=0

EXAMPLES

The following command sets the first port to loop internally, use internal clocking and operate at 38400 baud:

```
example# hsip_init hihp0 38400 loop=yes
port=hihp0
speed=38400,
mode=fdx, signal=no, loopback=yes, nrzi=no, mtu=2064, mru=2064,
txc=baud, rxc=baud, txd=txd, rxd=rxs
```

The following command sets the same port's clocking, local loopback and baud rate settings to their default values:

```
example# hsip_init hihp0 speed=1536000 loopback=no txc=txc rxc=rxs
port=hihp0
speed=1536000,
mode=fdx, signal=no, loopback=no, nrzi=no, mtu=2064, mru=2064,
txc=txc, rxc=rxs, txd=txd, rxd=rxs
```

SEE ALSO

hsip_loop(1M), hsip_stat(1M), Intro(2), hsip(7D)

DIAGNOSTICS

device missing minor device number

The name *device* does not end in a decimal number that can be used as a minor device number.

bad speed: arg

The string *arg* that accompanied the "speed=" option could not be interpreted as a decimal integer.

Bad arg: arg

The string *arg* did not make sense as an option.

ioctl failure code = errno

An **ioctl(2)** system call failed. The meaning of the value of *errno* may be found in the **Intro(2)** manual page.

WARNINGS

`hsip_init` should not be used on an active serial link, unless needed to resolve an error condition. It should not be run casually, or if the user is unsure of the consequences of its use.

NAME	hsip_loop – high speed synchronous serial loopback test program for high speed serial interface.																
SYNOPSIS	<code>/opt/SUNWconn/bin/hsip_loop [-cdlsvt] device</code>																
DESCRIPTION	<p>The hsip_loop command performs several loopback tests that are useful in exercising the various components of a serial communications link.</p> <p>Before running a test, hsip_loop opens the designated port and configures it according to command line options and the specified test type. It announces the names of the devices being used to control the hardware channel, the channel number (ppa) corresponding to the <i>device</i> argument, and the parameters it has set for that channel. It then runs the loopback test in three phases.</p> <p>The first phase is to listen on the port for any activity. If no activity is seen for at least four seconds, hsip_loop proceeds to the next phase. Otherwise, the user is informed that the line is active and that the test cannot proceed, and the program exits.</p> <p>In the second phase, called the "first-packet" phase, hsip_loop attempts to send and receive one packet. The program will wait for up to four seconds for the returned packet. If no packets are seen after five attempts, the test fails with an error message. If a packet is returned, the result is compared with the original. If the length and content do not match exactly, the test fails.</p> <p>The final phase, known as the "multiple-packet" phase, attempts to send many packets through the loop. Because the program has verified the integrity of the link in the first-packet phase, the test will not fail after a particular number of timeouts. If a packet is not seen after four seconds, a message is displayed. Otherwise, a count of the number of packets received is updated on the display once per second. If it becomes obvious that the test is not receiving packets during this phase, the user may wish to stop the program manually. The number and size of the packets sent during this phase is determined by default values, or by command line options. Each returned packet is compared with its original for length and content. If a mismatch is detected, the test fails. The test completes when the required number of packets have been sent, regardless of errors.</p> <p>After the multiple-packet phase has completed, the program displays a summary of the hardware event statistics for the channel that was tested. The display takes the following form:</p> <table border="0" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Port</th> <th style="text-align: left;">CRC errors</th> <th style="text-align: left;">Aborts</th> <th style="text-align: left;">Overruns</th> <th style="text-align: left;">Underruns</th> <th style="text-align: left;">In</th> <th style="text-align: left;"><-Drops-></th> <th style="text-align: left;">Out</th> </tr> </thead> <tbody> <tr> <td>hihp0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td></td> <td>0</td> </tr> </tbody> </table> <p>This is followed by an estimated line speed, which is an approximation of the bit rate of the line, based on the number of bytes sent and the actual time that it took to send them. This is a very rough approximation and should not be used in benchmarking, because elapsed time includes time to print to the display.</p>	Port	CRC errors	Aborts	Overruns	Underruns	In	<-Drops->	Out	hihp0	0	0	0	0	0		0
Port	CRC errors	Aborts	Overruns	Underruns	In	<-Drops->	Out										
hihp0	0	0	0	0	0		0										

OPTIONS

The options for `hsip_loop` are described in the following table:

Option	Parameter	Default	Description
<code>-c</code>	<i>packet_count</i>	100	Specifies the number of packets to be sent in the multiple-packet phase.
<code>-d</code>	<i>hex_data_byte</i>	<i>random</i>	Specifies that each packet will be filled with bytes with the value of <i>hex_data_byte</i> .
<code>-l</code>	<i>packet_length</i>	100	Specifies the length of each packet in bytes with a maximum of 2064 bytes.
<code>-s</code>	<i>line_speed</i>	9600	Bit rate in bits per second, minimum of 9600 bps and a maximum of 2048000 bps.
<code>-v</code>			Sets verbose mode. If data errors occur, the expected and received data is displayed.
<code>-t</code>	<i>test_type</i>	<i>none</i>	A number, from 1 to 4, that specifies which test to perform. The values for <i>test_type</i> are as follows: <ol style="list-style-type: none"> 1 Internal loopback test. Port loopback is on. Transmit and receive clock sources are internal (baud rate generator). 2 External loopback test. Port loopback is off. Transmit and receive clock sources are internal. Requires a loopback plug suitable to the port under test. 3 External loopback test. Port loopback is off. Transmit and receive clock sources are external (modem). Requires that one of the local modem or the remote modem be set in a loopback configuration. 4 Test using predefined parameters. User defines hardware configuration and may select port parameters using the hsip_init(1M) command.

All numeric options except `-d` are entered as decimal numbers (for example, `-s 19200`). If you do not provide the `-t test_type` option, `hsip_loop` prompts for it.

EXAMPLES

The following command causes `hsip_loop` to use a packet length of 512 bytes over the first CPU port:

```
example# hsip_loop -l 512 hihp0
```

In response to the above command, `hsip_loop` prompts you for the test option you want.

The following command performs an internal loopback test on the first CPU port, using 5000 packets and a bit rate of 56000 bps :

```
example# hsip_loop -t 1 -s 56000 -c 5000 hihp0
```

SEE ALSO

hsip_init(1M), hsip_stat(1M), hsip(7D)

DIAGNOSTICS

device missing minor device number

The name *device* does not end in a decimal number that can be used as a minor device number.

invalid packet length: *nnn*

The packet length was specified to be less than zero or greater than 2064.

poll: nothing to read

poll: nothing to read or write.

The **poll(2)** system call indicates that there is no input pending and/or that output would be blocked if attempted.

len *xxx* should be *yyy*

The packet that was sent had a length of *yyy*, but was received with a length of *xxx*.

***nnn* packets lost in outbound queueing**

***nnn* packets lost in inbound queueing**

A discrepancy has been found between the number of packets sent by **hsip_loop** and the number of packets the driver counted as transmitted, or between the number counted as received and the number read by the program.

WARNINGS

To allow its tests to run properly, as well as prevent disturbance of normal operations, **hsip_loop** should only be run on a port that is not being used for any other purpose at that time.

NAME	hsip_stat – report driver statistics from a high speed synchronous serial link port.																										
SYNOPSIS	<pre> /opt/SUNWconn/bin/hsip_stat [-f] -a num_of_ports /opt/SUNWconn/bin/hsip_stat [-f] device [period] /opt/SUNWconn/bin/hsip_stat -c [-f] -a num_of_ports /opt/SUNWconn/bin/hsip_stat -c [-f] device </pre>																										
DESCRIPTION	<p>The <code>hsip_stat</code> command reports the event statistics maintained by a high speed synchronous serial device driver. The report may be a single snapshot of the accumulated totals, or a series of samples showing incremental changes.</p> <p>Event statistics are maintained by a driver for each physical channel that it supports. They are initialized to zero at the time the driver module is loaded into the system when one of the driver's entry points is first called.</p> <p>The device argument is the name of the high speed serial device as it appears in the <code>/dev</code> directory. For example, hihp0 specifies the first on-board high speed serial device.</p> <p>As an alternative, you can display or clear the statistics for multiple physical channels using num_of_ports argument. The <code>hsip_stat</code> program will then display statistics accumulated for the first n number of ports, where n is num_of_ports.</p> <p>The following is a breakdown of <code>hsip_stat</code> output:</p> <table border="0"> <tr> <td style="vertical-align: top;">speed</td> <td>The line speed the device has been set to operate at. It is the user's responsibility to make this value correspond to the modem clocking speed when clocking is provided by the modem.</td> </tr> <tr> <td style="vertical-align: top;">ipkts</td> <td>The total number of input packets.</td> </tr> <tr> <td style="vertical-align: top;">opkts</td> <td>The total number of output packets.</td> </tr> <tr> <td style="vertical-align: top;">undrun</td> <td>The number of transmitter underrun errors.</td> </tr> <tr> <td style="vertical-align: top;">ovrrun</td> <td>The number of receiver overrun errors.</td> </tr> <tr> <td style="vertical-align: top;">abort</td> <td>The number of aborted received frames.</td> </tr> <tr> <td style="vertical-align: top;">crc</td> <td>The number of received frames with CRC errors.</td> </tr> <tr> <td style="vertical-align: top;">isize</td> <td>The average size (in bytes) of input packets.</td> </tr> <tr> <td style="vertical-align: top;">osize</td> <td>The average size (in bytes) of output packets.</td> </tr> <tr> <td style="vertical-align: top;">ierorr</td> <td>Input error count (errors: Incomplete Frame, Empty frame, Glitch on RxC).</td> </tr> <tr> <td style="vertical-align: top;">oerror</td> <td>Output error count (errors: CTS lost, Glitch on TxC).</td> </tr> <tr> <td style="vertical-align: top;">iutil</td> <td>Input line utilization expressed as a percentage.</td> </tr> <tr> <td style="vertical-align: top;">outil</td> <td>Output line utilization expressed as a percentage.</td> </tr> </table>	speed	The line speed the device has been set to operate at. It is the user's responsibility to make this value correspond to the modem clocking speed when clocking is provided by the modem.	ipkts	The total number of input packets.	opkts	The total number of output packets.	undrun	The number of transmitter underrun errors.	ovrrun	The number of receiver overrun errors.	abort	The number of aborted received frames.	crc	The number of received frames with CRC errors.	isize	The average size (in bytes) of input packets.	osize	The average size (in bytes) of output packets.	ierorr	Input error count (errors: Incomplete Frame, Empty frame, Glitch on RxC).	oerror	Output error count (errors: CTS lost, Glitch on TxC).	iutil	Input line utilization expressed as a percentage.	outil	Output line utilization expressed as a percentage.
speed	The line speed the device has been set to operate at. It is the user's responsibility to make this value correspond to the modem clocking speed when clocking is provided by the modem.																										
ipkts	The total number of input packets.																										
opkts	The total number of output packets.																										
undrun	The number of transmitter underrun errors.																										
ovrrun	The number of receiver overrun errors.																										
abort	The number of aborted received frames.																										
crc	The number of received frames with CRC errors.																										
isize	The average size (in bytes) of input packets.																										
osize	The average size (in bytes) of output packets.																										
ierorr	Input error count (errors: Incomplete Frame, Empty frame, Glitch on RxC).																										
oerror	Output error count (errors: CTS lost, Glitch on TxC).																										
iutil	Input line utilization expressed as a percentage.																										
outil	Output line utilization expressed as a percentage.																										

- OPTIONS**
- f Select a complete set of accumulated statistics for the device specified. This is useful while debugging the **hsip** driver.
 - a Select all devices.
 - c Clear the accumulated statistics for the device specified. This may be useful when it is not desirable to unload a particular driver, or when the driver is not capable of being unloaded.

num_of_ports

Specify the number of devices that you want to dump the statistics.

period

Cause `hsip_stat` to sample the statistics every *period* seconds and report incremental changes. The output reports line utilization for input and output in place of average packet sizes. These are the relationships between bytes transferred and the speed, expressed as percentages. The loop repeats indefinitely, with a column heading printed every twenty lines for convenience.

EXAMPLES

example# **hsip_stat hihp0**

speed	ipkts	opkts	undrun	ovrrun	abort	crc	isize	osize
9600	15716	17121	0	0	1	3	98	89

example# **hsip_stat 5**

	speed	ipkts	opkts	undrun	ovrrun	abort	crc	isize	osize
hihp0	9600	15716	10100	0	0	1	3	98	89
hihp1	9600	15234	20100	0	0	1	3	98	89
hihp2	9600	15123	18254	0	0	1	3	98	89
hihp3	9600	15378	18234	0	0	1	3	98	89
hihp4	9600	13900	13000	0	0	1	3	98	89

example# **hsip_stat -c hihp0**

speed	ipkts	opkts	undrun	ovrrun	abort	crc	isize	osize
9600	0	0	0	0	0	0	0	0

example# **hsip_stat hihp0 5**

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%

(In this final example a new line of output is generated every five seconds.)

SEE ALSO

hsip_init(1M), hsip_loop(1M), hsip(7D)

DIAGNOSTICS

bad interval: *arg*

The argument *arg* is expected to be an interval and could not be understood.

device missing minor device number

The name *device* does not end in a decimal number that can be used as a minor device number.

WARNINGS

Underrun, overrun, frame-abort and CRC errors have a variety of causes. Communication protocols are typically able to handle such errors and initiate recovery of the transmission in which the error occurred. Small numbers of such errors are not a significant problem for most protocols. However, because the overhead involved in recovering from a link error can be much greater than that of normal operation, high error rates can greatly degrade overall link throughput. High error rates are often caused by problems in the link hardware, such as cables, connectors, interface electronics or telephone lines. They may also be related to excessive load on the link or the supporting system.

The percentages for input and output line utilization reported when using the *interval* option 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 baud rate set for the device probably does not reflect the speed of the modem.

NAME	<code>nf_fddidaemon</code> – start/stop the NF FDDI SMT/SNM daemon and its associated processes.
SYNOPSIS	<code>nf_fddidaemon start stop</code>
AVAILABILITY	This command is available with the <i>SunFDDI</i> product.
DESCRIPTION	The <code>nf_fddidaemon</code> script starts/stops the SNM daemon and its associated processes.
OPTIONS	<code>start</code> Starts the SNM daemon <code>stop</code> Stops the SNM daemon You must be root to run this command.
SEE ALSO	<code>nf_snmd</code> (1M)

NAME	<code>nf_install_agents</code> – install SunNet Manager agents for SunFDDI
SYNOPSIS	<code>nf_install_agents</code>
AVAILABILITY	This command is available with the <i>SunFDDI</i> product.
DESCRIPTION	<p>The <code>nf_install_agents</code> script copies the FDDI schema files to the directory in which the standard agents are installed and updates the configuration files for SunNet Manager.</p> <p>The <code>nf_install_agents</code> command takes no arguments.</p> <p>You must be root to run this command.</p>
SEE ALSO	<code>nf_snmd</code> (1M)

NAME	nf_macid – obtain MAC address from specified nf (SunFDDI) interface.
SYNOPSIS	nf_macid <i>interface</i>
AVAILABILITY	This command is available only with the <i>SunFDDI</i> product.
DESCRIPTION	<p>This command queries the IDPROM on the SunFDDI SBus card associated with a nf interface to obtain the MAC address resident there. This address is a globally unique, 48-bit address that is drawn from the same pool from which Ethernet addresses are taken.</p> <p>The nf_macid command does not allow you to set a MAC address, either on the SBus card or for an interface. Use ifconfig with the ether argument to assign the MAC address you obtain with nf_macid to an SunFDDI interface.</p> <p>Normally, you use the host-resident MAC address for all network interfaces on a machine. You would only use the MAC address obtained with nf_macid under unusual circumstances.</p> <p>You can be normal user (not root) to run this command.</p>
OPTIONS	<i>interface</i> Specifies the FDDI interface (nf<num>). The default (which you can omit) is nf0 .
EXAMPLE	<p>Obtain the MAC address for nf0:</p> <pre>% nf_macid 8:0:20:3e:da:5</pre> <p>Set the nf0 interface to have the MAC address in the SBus card IDPROM:</p> <pre># ifconfig nf0 ether 'nf_macid'</pre> <p>You would follow the preceding command with an ifconfig command to assign an IP address to nf0 and bring up that interface. Normally, such ifconfig commands would be run from a startup file.</p>
SEE ALSO	ifconfig (1M)

NAME	<code>nf_smtmon</code> – the SMT monitor.
SYNOPSIS	<code>nf_smtmon</code> [<code>-i interface</code>] [<code>-x</code>] [<code>-h</code>] [<i>frametype</i>]
AVAILABILITY	This command is available with the <i>SunFDDI</i> product.
DESCRIPTION	<p><code>nf_smtmon</code> is used to display received SMT frames. You should run this command on the FDDI proxy system if the Console does not receive a response from a request for SMT MIB information.</p> <p>You must be root to run this command.</p>
OPTIONS	<p><code>-i interface</code> Specifies the FDDI interface (<i>nfnum</i> for SunFDDI). If this option is not specified, frames for all FDDI interfaces are displayed.</p> <p><code>-x</code> Displays the received frames in hex.</p> <p><code>-h</code> Displays the usage of this command.</p> <p><i>frametype</i> Specifies one or more types of SMT frames to be displayed. If this option is not specified, all types of frames are displayed. You can specify the following types of frames to be displayed:</p> <p>ecf Echo Frame. Request and response frames are used for SMT-to-SMT loopback testing on an FDDI ring.</p> <p>esf Extended Service Frame. Request, response, and announcement frames are used to extend new SMT services.</p> <p>nif Neighborhood Information Frame. Request, response, and announcement frames are used to communicate station addresses and descriptions.</p> <p>pmf_get Parameter Management Frame (PMF) Get Request. Request and response frames are used to retrieve SMT Management Information Base (MIB) attribute values.</p> <p>rdf Request Denied Frame (response only). Sent in response to an unsupported or unknown request.</p> <p>sifconfig Status Information Frame (SIF) Configuration. Request and response frames are used to retrieve configuration parameters for one or more stations on the ring.</p> <p>sifoperation Status Information Frame (SIF) Operation. Request and response frames are used to retrieve operation information for one or more stations on the ring.</p> <p>srf Status Report Frame. Announcement frame used to report Station Status. The current version of the SMT</p>

daemon does not send out SRFs; however, any received SRFs are passed on to SNM as traps.

- EXAMPLES** `nf_smtmon -i nf0 nif sifconfig`
displays the NIF and SIF configuration frames received in non-hex format on the **nf0** (SunFDDI) interface.
- `nf_smtmon -i nf1 -x ecf`
displays, in hex, ECF frames received on the **nf1** (SunFDDI) interface.
- SEE ALSO** `smttd(1M)`

NAME	<code>nf_snmd</code> – start the station management (SMT) to SunNet Manager daemon.
SYNOPSIS	<code>nf_snmd [-d] [-v5]</code>
AVAILABILITY	This command is available with the <i>SunFDDI</i> product.
DESCRIPTION	<p>Upon invocation, the SNM daemon starts up station management processes that allow the station to communicate with other stations using the SMT protocol, and collect and return FDDI statistics to a SunNet Manager (SNM) Console. The daemon also receives SMT requests and SMT responses. The daemon also sends out SMT requests to other stations on the ring on behalf of SNM. The SMT daemon also forwards received Status Report Frames (SRFs) to the SNM management station in the form of traps.</p> <p>The processes started by the SNM daemon include two SNM agents: a local agent (<code>fddi</code>) and a proxy agent (<code>fddisnt</code>). Like other SNM agents, the local agent and proxy agent communicate with the SNM management station using RPC. The local agent responds to SNM requests with FDDI statistics gathered on the local machine. These statistics are equivalent to those displayed with the <code>nf_stat</code> and <code>nf_stat -m</code> commands.</p> <p>The proxy agent can return two types of SMT information to the SNM Console: actual SMT frames (ECF, ESF, NIF, SIF Configuration, or SIF Operation), and attribute values for selected SMT MIB groups. The proxy agent gathers information from target stations by issuing SMT request frames and receiving SMT response frames. The proxy uses PMF Get request and response frames to retrieve MIB attribute values from the target station.</p> <p>If the target station does not support PMF Get frames, it returns an RDF response to the proxy system. If a Console request for MIB attributes values is not successful, run the SMT monitor on the proxy system to see if an RDF frame has been received from the target station. If PMF Get frames are not supported by the target station, you may be able to use NIF, SIF Configuration or SIF Operation frames to return the desired attribute values.</p> <p>The SMT MIB attributes groups MAC, PATH, and PORT contain index parameters. If you send a Quick Dump request from the Console for attribute values from one of these groups, only the values associated with the first index are returned (from the Console's point of view, the key value associated with the request is 1). If you want to see attribute values associated with other indexes, you must send a Data Report request with the Key field in the request set to the desired index.</p> <p>If you make any changes to the <code>/etc/opt/snm/snm.conf</code> file on the station (for example, you add an additional hostname to the <code>na.fddi.trap-rendez</code> entry), you must kill the SNM daemon with <code>nf_snmd_kill</code> and then restart it in order for the change(s) to take effect.</p> <p>You must be root to run this command.</p>

OPTIONS **-d** (debug mode) Displays a one-line entry in the window where **nf_snmd** is started for each frame that the station sends or receives. If this option is not specified, you are returned to the system prompt and there is no display. Use of this option is not recommended if the **nf_snmd** command is included in **/etc/rc2.d/S98nf_fddidaemon** .

SEE ALSO **nf_snmd_kill (1M), nf_stat (1M)**

NAME	nf_snmd_kill – kill the station management (SMT) to SunNet Manager daemon and its associated processes.
SYNOPSIS	nf_snmd_kill
AVAILABILITY	This command is available with the <i>SunFDDI</i> product.
DESCRIPTION	<p>The nf_snmd_kill script kills the SNM daemon and its associated processes. This command also kills the two SNM agents which are started by the SNM daemon: the local agent (<i>fddi</i>) and the proxy agent (<i>fddismt</i>). This command should not be used if the SNM daemon is not already running.</p> <p>The nf_snmd_kill command takes no arguments.</p> <p>You must be root to run this command.</p>
SEE ALSO	nf_snmd (1M)

NAME	<code>nf_stat</code> – display SunFDDI interface statistics.
SYNOPSIS	<code>nf_stat [-m][<i>interface</i>][<i>interval</i>][<i>count</i>]</code>
AVAILABILITY	This command is available with the <i>SunFDDI</i> product.
DESCRIPTION	<p>The <code>nf_stat</code> utility displays statistics for the SunFDDI interface. Some statistics relate to the SunFDDI implementation of the ANSI FDDI Connection Management standard (CMT), while others contain packet throughput, or station neighbor information.</p> <p>This utility can report, on a periodic basis, packet throughput statistics, reconfiguration events, and interface exceptions. It also reports the identity of neighboring stations, information on its PHYs, and some FORMAC error counters. Several of the counters and status variables are periodically passed to the host from the hardware during the heartbeat signal. These statistics are available when invoking the command without the <code>-m</code> option. Issuing the command without an <i>interval</i> value displays the accumulated statistics; issuing the command with an <i>interval</i> value displays any differences between values since the previous display.</p>
OPTIONS	<p>-m Dumps the current nearest neighbor information and FDDI/S timer settings (described below). The <i>interval</i> and <i>count</i> arguments have no effect when used with this option. Note that you must be root to invoke <code>nf_stat</code> with the <code>-m</code> option.</p> <p><i>interface</i> Specifies which SunFDDI interface, <i>nfnum</i>.</p> <p><i>interval</i> Specifies the interval in seconds at which to display the statistics.</p> <p><i>count</i> Specifies the number of times to display the statistics. If no count is provided, the utility runs forever. It can be terminated by typing ^C (Control-C).</p>
USAGE	<p>You invoke <code>nf_stat</code> with the <code>-m</code> option to display information about neighboring stations. It generates a columnar display containing the following categories of data:</p> <p>PhyA On a machine running SunFDDI Dual, shows the PHY type of the neighboring station that is connected to PHYA. Values are A, B, S, M, and None (if no connection). This column does not appear on a machine running SunFDDI SAS - Single Attached Station. (See Chapter 9 of the document ANSI/FDDI Station Management (SMT) Rev7.2 (25 June 1992)).</p> <p>PhyB On a machine running SunFDDI Dual, shows the PHY type of the neighboring station that is connected to PHYB. Values are A, B, S, M, and None (if no connection). This column does not appear on a machine running SunFDDI SAS. (See Chapter 7 of the document ANSI/FDDI Station Management (SMT) Rev7.2 (25 June 1992)).</p> <p>PhyS On a machine running SunFDDI SAS, shows the PHY type of the neighboring station that is connected to PHYS. Values are A, B, S, M, and None (if no connection). If connected to a concentrator, this will be M. This column</p>

does not appear on a machine running SunFDDI Dual.

Frame	FDDI MAC standard counter, frames received.
Error	FDDI MAC standard counter, frame with the E bit first detected at this station.
Lost	Frames whose reception is aborted.
SA	MAC address; the unique 48-bit address of the SunFDDI interface. Where an IP hostname exists, it is displayed; otherwise, the 48-bit MAC address is used.
UNA	The address of this station's upstream neighbor, using the SMT NIF protocol.
DNA	The address of this station's downstream neighbor, using the SMT NIF protocol.

Display status information : You invoke **nf_stat** without the *-m* option, or with values for *interface* or *interval*, to display status information. Issuing the command without an *interval* value displays the accumulated statistics; issuing the command with an *interval* value displays any differences between values since the previous display.

One use of **nf_stat** without the *-m* option is to monitor the **Ring_OP** (Ring Operational) column; if it indicates more than one ring_op per second, there are media problems that must be fixed.

When invoked without the *-m* option, **nf_stat** generates a columnar display containing the following categories of data:

Ring	Indicates whether the ring is up or down (that is, the Claim has succeeded). Note: The following five fields use terms described in the SMT document, Chapter 9.
ECM	(<i>ec_state</i>). Shows the current state of the ECM state machine. Valid values are: Out , In , Trace , Leave , Path_Test , Insert , Check , and Deinsert .
RMT	(<i>rmt_state</i>). Shows the current state of the RMT state machine. Valid values are: Isolated , Non_Op , Ring_Op , Detect , Non_Op_Dup , Ring_Op_Dup , Directed , and Rm_Trace .
PCMA/PCMB (for SunFDDI Dual) PCMS (for SunFDDI SAS)	(<i>pc_state</i>). Is a variable from PCM to other management entities containing the current state of the PCM state machine. Current valid values are: Off (O), Break (B), Reject (R), Connect (C), Next (N), Signal (S), Join (J), Verify (V), Active (A), and Maint (M).
Ring_OP (Ring Operational).	Indicates the number of times the ring has come up (and therefore implies the number of times the ring has gone down).
XmitP	The number of packets transmitted.
RecvP	The number of packets received.

SEE ALSO

netstat (1M)

NAME	<code>nf_sync</code> – configure SunFDDI interface to operate in synchronous mode.
SYNOPSIS	<code>nf_sync nf<inst> [<i>tsync sap</i>]</code>
AVAILABILITY	This command is available with the <i>SunFDDI</i> product.
DESCRIPTION	The <code>nf_sync</code> utility is used to configure SunFDDI interfaces to operate in synchronous mode. By default, the SunFDDI interface configure to carry asynchronous traffic only.
OPTIONS	<p><code>nf<inst></code> Specifies the FDDI interface,</p> <p><code>tsync</code> Specifies synchronous timer in nanoseconds, 400000 nanoseconds minimum,</p> <p><code>sap</code> Specifies the service access point (SAP) for synchronous operation.</p>
USAGE	<p>Running <code>nf_sync</code> without specifying values for <code>tsync</code> and <code>sap</code> returns current configuration of the interface.</p> <p>To reconfigure SAP for asynchronous operations, specify <code>tsync=0</code></p>
EXAMPLES	<p><code>nf_sync nf0</code> displays current configuration on the nf0 (SunFDDI) interface.</p> <p><code>nf_sync nf0 1000000 800</code> configures SAP 800 for synchronous operation with a clock rate 1000000 nanoseconds (1ms)</p>

NAME	<code>pf_fddidaemon</code> – start/stop the PF FDDI SMT/SNM daemon and its associated processes.
SYNOPSIS	<code>pf_fddidaemon start stop</code>
AVAILABILITY	This command is available with the <i>SunFDDI</i> product.
DESCRIPTION	The <code>pf_fddidaemon</code> script starts/stops the SNM daemon and its associated processes.
OPTIONS	<code>start</code> Starts the SNM daemon <code>stop</code> Stops the SNM daemon You must be root to run this command.
SEE ALSO	<code>pf_snmd</code> (1M)

NAME	<code>pf_install_agents</code> – install SunNet Manager agents for SunFDDI
SYNOPSIS	<code>pf_install_agents</code>
AVAILABILITY	This command is available with the <i>SunFDDI</i> product.
DESCRIPTION	The <code>pf_install_agents</code> script copies the FDDI schema files to the directory in which the standard agents are installed and updates the configuration files for SunNet Manager. The <code>pf_install_agents</code> command takes no arguments. You must be root to run this command.
SEE ALSO	<code>pf_snmd</code> (1M)

NAME	pf_macid – obtain MAC address from specified pf (SunFDDI/P) interface.
SYNOPSIS	pf_macid <i>interface</i>
AVAILABILITY	This command is available only with the <i>SunFDDI</i> product.
DESCRIPTION	<p>This command queries the IDPROM on the SunFDDI card associated with a pf interface to obtain the MAC address resident there. This address is a globally unique, 48-bit address that is drawn from the same pool from which Ethernet addresses are taken.</p> <p>The pf_macid command does not allow you to set a MAC address, either on the PCI card or for an interface. Use ifconfig with the ether argument to assign the MAC address you obtain with pf_macid to an SunFDDI interface.</p> <p>Normally, you use the host-resident MAC address for all network interfaces on a machine. You would only use the MAC address obtained with pf_macid under unusual circumstances.</p> <p>You can be normal user (not root) to run this command.</p>
OPTIONS	<i>interface</i> Specifies the FDDI interface (pf<num>). The default (which you can omit) is pf0 .
EXAMPLE	<p>Obtain the MAC address for pf0:</p> <pre>% pf_macid 8:0:20:3e:da:5</pre> <p>Set the pf0 interface to have the MAC address in the PCI card IDPROM:</p> <pre># ifconfig pf0 ether 'pf_macid'</pre> <p>You would follow the preceding command with an ifconfig command to assign an IP address to pf0 and bring up that interface. Normally, such ifconfig commands would be run from a startup file.</p>
SEE ALSO	ifconfig (1M)

NAME	pf_smtmon – the SMT monitor.																								
SYNOPSIS	pf_smtmon [-i <i>interface</i>] [-x] [-h] [<i>frametype</i>]																								
AVAILABILITY	This command is available with the <i>SunFDDI/P</i> product.																								
DESCRIPTION	<p>pf_smtmon is used to display received SMT frames. You should run this command on the FDDI proxy system if the Console does not receive a response from a request for SMT MIB information.</p> <p>You must be root to run this command.</p>																								
OPTIONS	<table border="0"> <tr> <td style="vertical-align: top;">-i <i>interface</i></td> <td>Specifies the FDDI interface (pfnum for SunFDDI/P). If this option is not specified, frames for all FDDI interfaces are displayed.</td> </tr> <tr> <td style="vertical-align: top;">-x</td> <td>Displays the received frames in hex.</td> </tr> <tr> <td style="vertical-align: top;">-h</td> <td>Displays the usage of this command.</td> </tr> <tr> <td style="vertical-align: top;"><i>frametype</i></td> <td>Specifies one or more types of SMT frames to be displayed. If this option is not specified, all types of frames are displayed. You can specify the following types of frames to be displayed:</td> </tr> <tr> <td style="vertical-align: top;">ecf</td> <td>Echo Frame. Request and response frames are used for SMT-to-SMT loopback testing on an FDDI ring.</td> </tr> <tr> <td style="vertical-align: top;">esf</td> <td>Extended Service Frame. Request, response, and announcement frames are used to extend new SMT services.</td> </tr> <tr> <td style="vertical-align: top;">nif</td> <td>Neighborhood Information Frame. Request, response, and announcement frames are used to communicate station addresses and descriptions.</td> </tr> <tr> <td style="vertical-align: top;">pmf_get</td> <td>Parameter Management Frame (PMF) Get Request. Request and response frames are used to retrieve SMT Management Information Base (MIB) attribute values.</td> </tr> <tr> <td style="vertical-align: top;">rdf</td> <td>Request Denied Frame (response only). Sent in response to an unsupported or unknown request.</td> </tr> <tr> <td style="vertical-align: top;">sifconfig</td> <td>Status Information Frame (SIF) Configuration. Request and response frames are used to retrieve configuration parameters for one or more stations on the ring.</td> </tr> <tr> <td style="vertical-align: top;">sifoperation</td> <td>Status Information Frame (SIF) Operation. Request and response frames are used to retrieve operation information for one or more stations on the ring.</td> </tr> <tr> <td style="vertical-align: top;">srf</td> <td>Status Report Frame. Announcement frame used to report Station Status. The current version of the SMT</td> </tr> </table>	-i <i>interface</i>	Specifies the FDDI interface (pfnum for SunFDDI/P). If this option is not specified, frames for all FDDI interfaces are displayed.	-x	Displays the received frames in hex.	-h	Displays the usage of this command.	<i>frametype</i>	Specifies one or more types of SMT frames to be displayed. If this option is not specified, all types of frames are displayed. You can specify the following types of frames to be displayed:	ecf	Echo Frame. Request and response frames are used for SMT-to-SMT loopback testing on an FDDI ring.	esf	Extended Service Frame. Request, response, and announcement frames are used to extend new SMT services.	nif	Neighborhood Information Frame. Request, response, and announcement frames are used to communicate station addresses and descriptions.	pmf_get	Parameter Management Frame (PMF) Get Request. Request and response frames are used to retrieve SMT Management Information Base (MIB) attribute values.	rdf	Request Denied Frame (response only). Sent in response to an unsupported or unknown request.	sifconfig	Status Information Frame (SIF) Configuration. Request and response frames are used to retrieve configuration parameters for one or more stations on the ring.	sifoperation	Status Information Frame (SIF) Operation. Request and response frames are used to retrieve operation information for one or more stations on the ring.	srf	Status Report Frame. Announcement frame used to report Station Status. The current version of the SMT
-i <i>interface</i>	Specifies the FDDI interface (pfnum for SunFDDI/P). If this option is not specified, frames for all FDDI interfaces are displayed.																								
-x	Displays the received frames in hex.																								
-h	Displays the usage of this command.																								
<i>frametype</i>	Specifies one or more types of SMT frames to be displayed. If this option is not specified, all types of frames are displayed. You can specify the following types of frames to be displayed:																								
ecf	Echo Frame. Request and response frames are used for SMT-to-SMT loopback testing on an FDDI ring.																								
esf	Extended Service Frame. Request, response, and announcement frames are used to extend new SMT services.																								
nif	Neighborhood Information Frame. Request, response, and announcement frames are used to communicate station addresses and descriptions.																								
pmf_get	Parameter Management Frame (PMF) Get Request. Request and response frames are used to retrieve SMT Management Information Base (MIB) attribute values.																								
rdf	Request Denied Frame (response only). Sent in response to an unsupported or unknown request.																								
sifconfig	Status Information Frame (SIF) Configuration. Request and response frames are used to retrieve configuration parameters for one or more stations on the ring.																								
sifoperation	Status Information Frame (SIF) Operation. Request and response frames are used to retrieve operation information for one or more stations on the ring.																								
srf	Status Report Frame. Announcement frame used to report Station Status. The current version of the SMT																								

daemon does not send out SRFs; however, any received SRFs are passed on to SNM as traps.

- EXAMPLES** **pf_smtmon -i pf0 nif sifconfig**
displays the NIF and SIF configuration frames received in non-hex format on the **pf0** (SunFDDI/P) interface.
- pf_smtmon -i pf1 -x ecf**
displays, in hex, ECF frames received on the **pf1** (SunFDDI/P) interface.
- SEE ALSO** **smttd (1M)**

NAME	pf_snmd – start the station management (SMT) to SunNet Manager daemon.
SYNOPSIS	pf_snmd [-d] [-v5]
AVAILABILITY	This command is available with the <i>SunFDDI/P</i> product.
DESCRIPTION	<p>Upon invocation, the SNM daemon starts up station management processes that allow the station to communicate with other stations using the SMT protocol, and collect and return FDDI statistics to a SunNet Manager (SNM) Console. The daemon also receives SMT requests and SMT responses. The daemon also sends out SMT requests to other stations on the ring on behalf of SNM. The SMT daemon also forwards received Status Report Frames (SRFs) to the SNM management station in the form of traps.</p> <p>The processes started by the SNM daemon include two SNM agents: a local agent (fddi) and a proxy agent (fddisnt). Like other SNM agents, the local agent and proxy agent communicate with the SNM management station using RPC. The local agent responds to SNM requests with FDDI statistics gathered on the local machine. These statistics are equivalent to those displayed with the pf_stat and pf_stat -m commands.</p> <p>The proxy agent can return two types of SMT information to the SNM Console: actual SMT frames (ECF, ESF, NIF, SIF Configuration, or SIF Operation), and attribute values for selected SMT MIB groups. The proxy agent gathers information from target stations by issuing SMT request frames and receiving SMT response frames. The proxy uses PMF Get request and response frames to retrieve MIB attribute values from the target station.</p> <p>If the target station does not support PMF Get frames, it returns an RDF response to the proxy system. If a Console request for MIB attributes values is not successful, run the SMT monitor on the proxy system to see if an RDF frame has been received from the target station. If PMF Get frames are not supported by the target station, you may be able to use NIF, SIF Configuration or SIF Operation frames to return the desired attribute values.</p> <p>The SMT MIB attributes groups MAC, PATH, and PORT contain index parameters. If you send a Quick Dump request from the Console for attribute values from one of these groups, only the values associated with the first index are returned (from the Console's point of view, the key value associated with the request is 1). If you want to see attribute values associated with other indexes, you must send a Data Report request with the Key field in the request set to the desired index.</p> <p>If you make any changes to the <code>/etc/opt/snm/snm.conf</code> file on the station (for example, you add an additional hostname to the <code>na.fddi.trap-rendez</code> entry), you must kill the SNM daemon with pf_snmd_kill and then restart it in order for the change(s) to take effect.</p> <p>You must be root to run this command.</p>

OPTIONS **-d** (debug mode) Displays a one-line entry in the window where **pf_snmd** is started for each frame that the station sends or receives. If this option is not specified, you are returned to the system prompt and there is no display. Use of this option is not recommended if the **pf_snmd** command is included in **/etc/rc2.d/S98pf_fddidaemon** .

SEE ALSO **pf_snmd_kill (1M), pf_stat (1M)**

NAME	pf_snmd_kill – kill the station management (SMT) to SunNet Manager daemon and its associated processes.
SYNOPSIS	pf_snmd_kill
AVAILABILITY	This command is available with the <i>SunFDDI/P</i> product.
DESCRIPTION	<p>The pf_snmd_kill script kills the SNM daemon and its associated processes. This command also kills the two SNM agents which are started by the SNM daemon: the local agent (fddi) and the proxy agent (fddismt). This command should not be used if the SNM daemon is not already running.</p> <p>The pf_snmd_kill command takes no arguments.</p> <p>You must be root to run this command.</p>
SEE ALSO	pf_snmd (1M)

NAME	pf_stat – display SunFDDI/P interface statistics.
SYNOPSIS	pf_stat [-m][<i>interface</i>][<i>interval</i>][<i>count</i>]
AVAILABILITY	This command is available with the <i>SunFDDI/P</i> product.
DESCRIPTION	<p>The pf_stat utility displays statistics for the SunFDDI/P interface. Some statistics relate to the SunFDDI/P implementation of the ANSI FDDI Connection Management standard (CMT), while others contain packet throughput, or station neighbor information.</p> <p>This utility can report, on a periodic basis, packet throughput statistics, reconfiguration events, and interface exceptions. It also reports the identity of neighboring stations, information on its PHYs, and some FORMAC error counters. Several of the counters and status variables are periodically passed to the host from the hardware during the heartbeat signal. These statistics are available when invoking the command without the <i>-m</i> option. Issuing the command without an <i>interval</i> value displays the accumulated statistics; issuing the command with an <i>interval</i> value displays any differences between values since the previous display.</p>
OPTIONS	<p>-m Dumps the current nearest neighbor information and FDDI/S timer settings (described below). The <i>interval</i> and <i>count</i> arguments have no effect when used with this option. Note that you must be root to invoke pf_stat with the -m option.</p> <p><i>interface</i> Specifies which SunFDDI/P interface, pfnum.</p> <p><i>interval</i> Specifies the interval in seconds at which to display the statistics.</p> <p><i>count</i> Specifies the number of times to display the statistics. If no count is provided, the utility runs forever. It can be terminated by typing ^C (Control-C).</p>
USAGE	<p>You invoke pf_stat with the -m option to display information about neighboring stations. It generates a columnar display containing the following categories of data:</p> <p>PhyA On a machine running SunFDDI/P Dual, shows the PHY type of the neighboring station that is connected to PHYA. Values are A, B, S, M, and None (if no connection). This column does not appear on a machine running SunFDDI/P SAS - Single Attached Station. (See Chapter 9 of the document ANSI/FDDI Station Management (SMT) Rev7.2 (25 June 1992)).</p> <p>PhyB On a machine running SunFDDI/P Dual, shows the PHY type of the neighboring station that is connected to PHYB. Values are A, B, S, M, and None (if no connection). This column does not appear on a machine running SunFDDI/P SAS. (See Chapter 7 of the document ANSI/FDDI Station Management (SMT) Rev7.2 (25 June 1992)).</p> <p>PhyS On a machine running SunFDDI/P SAS, shows the PHY type of the neighboring station that is connected to PHYS. Values are A, B, S, M, and None</p>

(if no connection). If connected to a concentrator, this will be **M**. This column does not appear on a machine running SunFDDI/P Dual.

Frame	FDDI MAC standard counter, frames received.
Error	FDDI MAC standard counter, frame with the E bit first detected at this station.
Lost	Frames whose reception is aborted.
SA	MAC address; the unique 48-bit address of the SunFDDI/P interface. Where an IP hostname exists, it is displayed; otherwise, the 48-bit MAC address is used.
UNA	The address of this station's upstream neighbor, using the SMT NIF protocol.
DNA	The address of this station's downstream neighbor, using the SMT NIF protocol.

Display status information : You invoke **pf_stat** without the *-m* option, or with values for *interface* or *interval*, to display status information. Issuing the command without an *interval* value displays the accumulated statistics; issuing the command with an *interval* value displays any differences between values since the previous display.

One use of **pf_stat** without the *-m* option is to monitor the **Ring_OP** (Ring Operational) column; if it indicates more than one ring_op per second, there are media problems that must be fixed.

When invoked without the *-m* option, **pf_stat** generates a columnar display containing the following categories of data:

Ring	Indicates whether the ring is up or down (that is, the Claim has succeeded). Note: The following five fields use terms described in the SMT document, Chapter 9.
ECM	(<i>ec_state</i>). Shows the current state of the ECM state machine. Valid values are: Out , In , Trace , Leave , Path_Test , Insert , Check , and Deinsert .
RMT	(<i>rmt_state</i>). Shows the current state of the RMT state machine. Valid values are: Isolated , Non_Op , Ring_Op , Detect , Non_Op_Dup , Ring_Op_Dup , Directed , and Rm_Trace .
PCMA/PCMB (for SunFDDI/P Dual) PCMS (for SunFDDI/P SAS)	(<i>pc_state</i>). Is a variable from PCM to other management entities containing the current state of the PCM state machine. Current valid values are: Off (O), Break (B), Reject (R), Connect (C), Next (N), Signal (S), Join (J), Verify (V), Active (A), and Maint (M).
Ring_OP (Ring Operational).	Indicates the number of times the ring has come up (and therefore implies the number of times the ring has gone down).
XmitP	The number of packets transmitted.
RecvP	The number of packets received.

SEE ALSO

netstat (1M)

NAME	rscadm – administer SUN(tm) Remote System Control (RSC)
SYNOPSIS	<pre> rscadm help rscadm resetrsc [-s] rscadm set <i>variable value</i> rscadm download [boot] <i>file</i> rscadm show [variable] rscadm date [-s] [[mmdd]HHMM mmddHHMM[cc]yy][.SS] rscadm send_event [-c] <i>message</i> rscadm modem_setup rscadm useradd <i>username</i> rscadm userdel <i>username</i> rscadm usershow [username] rscadm userpassword <i>username</i> rscadm userperm <i>username</i> [cuar] </pre>
DESCRIPTION	<p>rscadm administers the SUN(tm) Remote System Console (RSC). It allows the host server to interact with the RSC. The following operations are supported:</p> <p>rscadm help Displays a usage screen.</p> <p>rscadm resetrsc Reset the RSC. There are two types of reset allowed, a "hard" reset and a "soft" reset. The hard reset is done by default. The soft reset can be selected by using the -s option.</p> <p>rscadm set Set RSC configuration variables. Examples of RSC configuration variables include RSC IP address and RSC hostname. See the RSC documentation for a complete list of RSC configuration variables.</p> <p>rscadm download Program the RSC's firmware. There are two parts to the firmware, the boot monitor and the main image. By default, rscadm download programs the main firmware image. The boot option selects programming of the boot monitor.</p> <p>rscadm show View the current RSC configuration variable settings. If no variable is specified, rscadm shows all variable settings.</p> <p>rscadm date Show or set RSC's time and date. The -s options can be used to set RSC's time and date to the hosts time and date.</p> <p>rscadm send_event Send a text based event to RSC. RSC may forward the event based on its event configuration.</p>

rscadm modem_setup
Direct connection to the RSC modem. This allows the user to enter AT commands to configure the modem. "~." returns to prompt.

rscadm useradd
Add user account to RSC. RSC can support up to four separate users.

rscadm userdel
Delete a user account from RSC.

rscadm usershow
Show details on the specified user account. If a username is not specified, all user accounts will be shown.

rscadm userpassword
Set a password for the user account specified. This password overrides any existing password currently set. There is no verification of the old password before setting the new password. See the RSC documentation on valid password formats.

rscadm userperm
Set the authorization profile for the user. See the userperm options section in this man page for more detail.

OPTIONS

The following options are supported for rscadm:

rscadm resetrsc

[-s] Perform a "soft" reset instead of a "hard" reset. A hard reset physically resets the RSC hardware. The RSC software jumps to the boot firmware, simulating a reset, for a soft reset.

rscadm download

[boot] Program the boot monitor portion of the flash. The main portion of the flash is usually programmed.

rscadm show

[variable] Show the value of that particular variable.

rscadm date

[-s] Set the date to the hosts time and date.

[[mmdd]HHMM | mmddHHMM[cc]yy][.SS]
the date.

mm - month

dd - day

HH - hour

MM - minute

cc - the first two digits of the four digit year

yy - last 2 digits of the year number

SS - seconds

rscadm send_event

[-c] Send a critical event. Without the **-c**, **send_event** sends a warning. Warnings are only logged in the RSC event log and not forwarded further.

rscadm usershow

[username]

RSC account name to display info on. If no username is given, all accounts will be displayed.

rscadm userperm

[cuar] Set permissions for RSC account. If no permissions are specified, all four permissions will be disabled. The options are **t**o; allow user to connect to (c)onsole, allow user to use the (u)ser commands to modify RSC accounts, allow user to (a)dmminister/change the RSC configuration variables, allow the user to (r)eset RSC and to power on/off the host.

OPERANDS

The following operands are supported for **rscadm**:

rscadm set

variable RSC configuration variable to set. See the RSC documentation for a list of configuration variables.

value Value to set RSC configuration variable to. See the RSC documentation for a list of valid values.

rscadm download

file Firmware file to download. The file should contain the RSC boot monitor image or RSC main image.

rscadm send_event

message Text message to describe event. Should be enclosed in quotes.

rscadm useradd

username Username for new RSC account.

rscadm userdel

username RSC account to be removed.

rscadm userpassword

username RSC account to have password set.

rscadm userperm

username RSC account to have permissions changed.

EXIT STATUS

= 0 on success

!= 0 on failure (with status message)

EXAMPLES

```
# rscadm date
# rscadm date -s
# rscadm date 050113101998
```

```
# rscadm set hostname rsc15
# rscadm show
# rscadm show hostname
# rscadm send_event -c "The UPS signaled a loss in power!"
# rscadm send_event "The disk is close to full capacity"
# rscadm useradd rscroot
# rscadm userdel olduser
# rscadm usershow
# rscadm usershow rscroot
# rscadm userperm rscroot cuar
# rscadm userperm newuser c
# rscadm userperm newuser
```

NOTES rscadm modem_setup - "~." will only work after a new line.
rscadm MUST be run as root.

BUGS None known.

NAME	sunvts – Invokes the SunVTS kernel and its user interface
SYNOPSIS	sunvts [-lepqstv] [-o <i>option_file</i>] [-f <i>log_dir</i>] [-h <i>hostname</i>]
AVAILABILITY	SUNWvts
DESCRIPTION	The sunvts command is used to invoke the SunVTS user interface and kernel on the same system. It could be used to start the user interface on the local system and connect to the SunVTS kernel on the remote system. By default, it displays CDE Motif graphic interface for CDE environment, OpenLook graphic interface for OpenWindows environment, or TTY interface for non-windowing system.
OPTIONS	<ul style="list-style-type: none"> -l Displays SunVTS OpenLook graphic interface. -e Disables the security checking feature. -f <i>log_dir</i> Specifies an alternative <i>log_file</i> directory. The default <i>log_file</i> directory is /var/opt/SUNWvts/logs. -h <i>hostname</i> Starts the SunVTS user interface on the local system, which connects to or invokes the SunVTS kernel on the specified host after security checking succeeds. -o <i>option_file</i> Starts the SunVTS kernel with the test options loaded from the specified <i>option_file</i>, which by default is located in /var/opt/SUNWvts/options. -p Starts the SunVTS kernel vtsk (1M) such that it does not probe the test system's devices. -q Automatically quits both the SunVTS kernel and the user interface when testing stops. -s Automatically starts testing from a selected group of tests. The flag must be used with the -o <i>option_file</i> flag. -t Starts vtstty (1M), a TTY based interface, instead of CDE or OpenLook interface. -v Displays version information from vt sui(1M) and vt sk(1M).
NOTES	If vt sk (1M) is already running on the test system, the sunvts command ignores the -e , -o , -f , -q , -p , and -s options.
SEE ALSO	vt sk(1M) , vt stty(1M) , vt sui(1M) , vt sui.ol(1M) , vt sprobe(1M)

NAME	vts_cmd – Send a command to the SunVTS kernel (vtsk)
SYNOPSIS	vts_cmd [<i>command</i>] [<i>argument</i>]
AVAILABILITY	SUNWvts
DESCRIPTION	<p>vts_cmd is a UNIX shell application that allows you to send a single command to the SunVTS kernel (vtsk). The test machine's SunVTS kernel will send the response to the standard output.</p> <p>The SunVTS application programming interface (API) is character based, which means that a string of characters (in the form of a <i>command</i>) can be sent to the SunVTS kernel, which then returns a reply back in the form of a string of characters.</p> <p>vts_cmd(1M) allows the user to send commands and receive replies from a UNIX command line.</p>
OPTIONS	<p>vts_cmd uses the commands listed below. In all cases, the commands (and any of the command's arguments) must follow vts_cmd. See the EXAMPLES section for reference.</p> <p>Some of the command descriptions listed below refer to a testnode. In the SunVTS API, there is a hierarchy of testnodes, with the system being on the top, the test groups below the system, and the tests themselves at the bottom. In the commands below, use a slash "/" to refer to the system. A test group can be one of the following: Processor(s), Memory, Network, SCSI-Devices(esp0), Comm.Ports, Graphics, OtherDevices, or any user specified group. When referring to a test, you must mention the device name and the test name [for example, sound0(audio)].</p> <p>list testnode Displays all the testnodes under the specified testnode.</p> <p>config testnode Displays the configuration information of the testnode.</p> <p>status [testnode] [-r] Displays the testing status information of the system. If a testnode is specified, status will display the status information of that testnode. If you use the -r argument, the status information of all of testnodes recursive to the testnode will be displayed.</p> <p>option [testnode] [-l] [-h n s t a] Either displays all the options associated with the specified testnode, or sets a specific option in a testnode.</p> <p>To display a testnode's options, type option followed by the testnode and one of the categories:</p> <p>-h Threshold -n Notify category -s Scheduling category</p>

- t** Test execution category
- a** Advanced category

vts_cmd will print all options, as well as the setting of each option. Use the **-l** option to display the options in long form. In long form, the options will be displayed with all their settings.

option [testnode] [test_option] [-g|s|x|y|z]

-g is used to pass all of the current option settings, for a given instance of a given test, to all of the same instances and tests that are in the same group (will not affect the same tests that are in different groups).

-s is used to pass all of the current option settings for a given instance of a given test, to all of the same instances for all of the same tests on the system (rather than for a group, as with **-g**).

-x is used to pass all of the current option settings for a given instance of a given test, to all the instances of that test.

-y is used to pass all of the current option settings for a given instance of a given test, to all the instances of all the same tests in a particular group.

-z is used to pass all of the current option settings for a given instance of a given test, to all the instances of all the same tests in the whole system.

To set an option, you must state the testnode immediately followed by the option and the new setting. You must use this format when setting an option:

vts_cmd option testnode[option:setting]

Once the option has been successfully changed, vts_cmd will display the word "DONE".

select testnode

Selects a testnode. If a testnode is selected, all the tests associated with the testnode will be enabled and run when testing begins.

For example, if you select the Graphics testnode, all the tests in Graphics will be enabled for testing. If you select just the "fpu(fputest)" test, then you will only enable this test.

deselect testnode

Deselects a testnode. If a testnode is deselected, all the tests associated with the testnode will be disabled and will not be run when testing begins.

For example, if you deselect the OtherDevices testnode, all the tests in the OtherDevices will be disabled. If you select just the "cgsix0(cg6)" test, then you will only enable this test.

start

Starts all enabled (selected) SunVTS tests.

stop

Stops all running SunVTS tests.

suspend

Suspends (or pauses) all running SunVTS tests. When you are ready to resume testing, type "resume".

resume

Resumes any suspended tests.

reset

Resets all the SunVTS pass and error counts to zero.

probe

Probes all the devices on the test machine and updates the SunVTS kernel's device list.

If a device is listed in the device list, but it is not found during the probe, it will be removed from the list. Conversely, if a device does not exist in a previous device list and is found during the probe, it will be added to the list.

load option_file

Loads an option file. Once loaded, the system and test options will be changed to reflect the settings listed in the option file.

Option files are stored in the /var/opt/SUNWvts/options directory.

store option_file

Creates an option file, listing all the system and test options, and save it in the /var/opt/SUNWvts/options directory.

- quit**
Terminates the SunVTS kernel (vtsk).
- invokeds**
Starts the deterministic scheduler.
- quitds**
Terminates the deterministic scheduler.
- loadseq sequence_file**
Loads a sequence file. Once loaded, the deterministic scheduler UI will reflect the tasks in the loaded sequence file.
- storeseq sequence_file**
Creates sequence_file, listing all the tasks in the directory /var/opt/SUNWvts/sequences.
- statusseq**
Returns a string containing the status information of the currently running sequence. The string consists of four fields separated by commas (","). The fields are: current status of SunVTS, current loop count of the sequence, total loop count of the sequence, and currently running task's position.
- startseq**
Starts the execution of the deterministic scheduler.
- stopseq**
Stops the execution of the currently running task in the sequence file. Upon starting again, the execution will start from the task that was stopped.
- resumeseq**
Restarts the execution of the sequence file. Execution will start at the point where the sequence was stopped, unless the sequence was reset, in which case it would start at the beginning of the sequence file.
- resetseq**
Sets the starting point of the execution to the start of the sequence file. Will also reset the passes and error count.
- suspendseq**
Suspends the execution of the currently running task in the sequence file.
- removeseq sequence_file**
Removes sequence_file from the list of sequence files in the directory /var/opt/SUNWvts/sequences.
- listtask**
Lists the tasks that are present in the currently loaded

sequence file.

addtask task_name [i]

Adds task_name at the ith position in the sequence file. If no index is passed then the task would be added to the end of the list.

deletetask [i]

Removes the task at the specified index from the selected sequence.

loadtask task_name

Loads a task file. Once loaded, the system and test options will be changed to reflect the settings listed in the task file.

setloopcount count

Sets the number of loops to run in the current sequence to count.

getvtsmode

Gets the current mode of SunVTS kernel.

EXAMPLES

To list out the configuration information of the test machine, you would use the config command:

```
sample% vts_cmd config /
/[Hostname:sample,Model:SPARCstation-10,SunVTS version:1.0]:idle
```

To load an option file, you would use the load command:

```
sample% ls /var/adm/sunvtslog/options
CPU_options      sample      options
sbus_standard
sample% vts_cmd load sbus_standard
DONE
```

To print all the system options in the Comm.Ports testnode, you would use the option command and pipe the output to your local printer:

```
sample% vts_cmd option Comm.Ports -l | lp
request id is printer-213 (standard input)
```

ENVIRONMENT

VTS_CMD_HOST=hostname

The hostname of the test machine running the SunVTS kernel (**vtsk**). If this environment variable is not set, **vts_cmd** will attempt to send the commands to

the local machine's SunVTS kernel.

SEE ALSO

SunVTS User's Guide

NAME	vtsk – SunVTS diagnostic kernel
SYNOPSIS	vtsk [-epqsv] [-o options_file] [-f logfile_directory]
AVAILABILITY	SUNWvts
DESCRIPTION	<p>The vtsk command starts up the SunVTS diagnostic kernel as a background process. There can only be one copy of vtsk running at a time. Only the superuser can execute this command.</p> <p>Normally, vtsk is automatically started up by the sunvts (1M) command if it is not already running. vtsk will also be invoked by inetd (1M) when there is a connection request from vtsui or vtsui.ol. In that case, the security file, .sunvts_sec, will be checked for the permission before running vtsk on the target host specified by vtsui(1M) or vtsui.ol(1M).</p>
OPTIONS	<ul style="list-style-type: none"> -e Enables the security checking for all connection requests. -p Starts SunVTS diagnostic kernel, but does not probe system configuration. -q Quits both the SunVTS diagnostic kernel and the attached User Interfaces when the testing is completed. -s Runs enabled tests immediately after started. -v Display SunVTS diagnostic kernel's version information only. -o options_file Starts the SunVTS diagnostic kernel and sets the test options according to the option file named <i>options_file</i>. -f logfile_directory Specifies an alternative logfile directory, other than the default.
EXIT STATUS	<p>The following exit values are returned:</p> <ul style="list-style-type: none"> 0 Successful completion. -1 An error occurred.
FILES	<ul style="list-style-type: none"> /var/opt/SUNWvts/options default option file directory. /var/opt/SUNWvts/logs default log file directory.
SEE ALSO	sunvts(1M), vtsui(1M), vtsui.ol(1M), vtstty(1M), vtsprobe(1M)

NAME	vtsprobe – prints the device probe information from the SunVTS kernel
SYNOPSIS	vtsprobe [-m] [-h <i>hostname</i>]
AVAILABILITY	SUNWvts
DESCRIPTION	vtsprobe is a utility that displays the device and configuration information contained in the SunVTS kernel. The output includes the SunVTS assigned group for the device, the device name, the device instance, the testname attached to this device, and the configuration information obtained from the device-specific test probe.
OPTIONS	<p>-m Specifies manufacturing mode, which displays the probe information in a format that is easy to read using script files.</p> <p>-h <i>hostname</i> Specifies the <i>hostname</i> to connect to and get the device and configuration information. If not specified, the current host will be used.</p>
USAGE	After the SunVTS kernel is up and running, you may type vtsprobe at the shell prompt to get the probe output. (See the sunvts (1M) man page for more information on how to start up SunVTS.
EXAMPLE	<p>Running vtsprobe on a sun4m SPARCclassic produces the following output:</p> <pre> % vtsprobe Processor(s) system(system) System Configuration=sun4m SPARCclassic System clock frequency=50 MHz SBUS clock frequency=25 MHz fpu(fputest) Architecture=sparc Type=TI TMS390S10 or TMS390S15 microSPARC chip Memory kmem(vmem) Total: 143120KB mem(pmem) Physical Memory size=24 Mb SCSI-Devices(esp0) c0t2d0(rawtest) Capacity: 638.35MB Controller: esp0 Vendor: MICROP SUN Id: 1588-15MBSUN0669 Firmware Rev: SN0C </pre>

```
Serial Number: 1588-15MB103
c0t2d0(fstest)
  Controller: esp0
c0t3d0(rawtest)
  Capacity: 404.65MB
  Controller: esp0
  Vendor: SEAGATE
  SUN Id: ST1480 SUN0424
  Firmware Rev: 8628
  Serial Number: 00836508
c0t3d0(fstest)
  Capacity: 404.65MB
  Controller: esp0
  Vendor: SEAGATE
  SUN Id: ST1480 SUN0424
  Firmware Rev: 8628
  Serial Number: 00836508
c0t3d0(fstest)
  Controller: esp0
c0t6d0(cdtest)
  Controller: esp0
tape1(tapetest)
  Drive Type: Exabyte EXB-8500 8mm Helical Scan
Network
  isdn0(isdntest)
    NT Port TE Port
  le0(nettest)
    Host_Name: ctech84
    Host Address: 129.146.210.84
    Host ID: 8001784b
    Domain Name: scsict.Eng.Sun.COM
Comm.Ports
  zs0(sptest)
    Port a -- zs0 /dev/term/a : /devices/ ... a
    Port b -- zs1 /dev/term/b : /devices/ ... b
Graphics
  cgthree0(fbtest)

OtherDevices
  bpp0(bpptest)
    Logical name: bpp0
  sound0(audio)
    Audio Device Type: AMD79C30
  sound1(audio)
    Audio Device Type: DBRI Speakerbox
```

spd0(spctest)
Logical name: spd0

NOTES The output of **vtsprobe** is highly dependent on the device being correctly configured into the system (so that a SunVTS probe for the device can be run successfully on it) and on the availability of a device-specific test probe.

If the device is improperly configured or if there is no probing function associated with this device, **vtsprobe** cannot print any information associated with it.

SEE ALSO **sunvts(1M)**, **vtsk(1M)**, **vtsui(1M)**, **vtsui.ol(1M)**, **vtstty(1M)**

NAME	vtstty – TTY interface for SunVTS																
SYNOPSIS	vtstty [-qv] [-h <i>hostname</i>]																
AVAILABILITY	SUNWvts																
DESCRIPTION	vtstty is the default interface for SunVTS in the absence of a windowing environment. It can be used in a non-windowing environment such as a terminal connected to the serial port of the system. However, its use is not restricted to this; vtstty can also be used from shell window.																
OPTIONS	<p>-q The "auto-quit" option automatically quits when the conditions for SunVTS to quit are met.</p> <p>-v Prints the vtstty version. The interface is not started when you include this option.</p> <p>-h <i>hostname</i> Connects to the SunVTS kernel running on the host identified by <i>hostname</i>.</p>																
USAGE	<p>The vtstty screen consists of four panels: main control, status, test groups, and console. The panels are used to display choices that the user can select to perform some function and/or to display information. A panel is said to be "in focus" or in a "selected" state when it is surrounded by asterisks and the current item is highlighted. In order to choose from the items in a panel, the focus should be shifted to that panel first.</p> <p>The following are the different types of selection items that can be present in a panel:</p> <table border="0"> <tr> <td style="padding-right: 20px;">Text string</td> <td>Describes a choice that, when selected, either pops up another panel or performs a function. For example, "stop" will stop the SunVTS testing.</td> </tr> <tr> <td>Data entry field</td> <td>To enter or edit numeric or textual data.</td> </tr> <tr> <td>Checkbox</td> <td>Represented as "[]". Checkboxes are associated with items and indicate whether the associated item is selected or not. A checkbox can be in one of the following two states: Deselected [] or Selected [*].</td> </tr> </table> <p>The key assignments given below describe the keys for shifting focus, making a selection, and performing other functions:</p> <table border="0"> <tr> <td style="padding-right: 20px;">TAB or <CTRL>W</td> <td>Shift focus to another panel</td> </tr> <tr> <td>RETURN</td> <td>Select current item</td> </tr> <tr> <td>Spacebar</td> <td>Toggle checkbox</td> </tr> <tr> <td>Up arrow or <CTRL>U</td> <td>Move up one item</td> </tr> <tr> <td>Down arrow or <CTRL>N</td> <td>Move down one item</td> </tr> </table>	Text string	Describes a choice that, when selected, either pops up another panel or performs a function. For example, "stop" will stop the SunVTS testing.	Data entry field	To enter or edit numeric or textual data.	Checkbox	Represented as "[]". Checkboxes are associated with items and indicate whether the associated item is selected or not. A checkbox can be in one of the following two states: Deselected [] or Selected [*].	TAB or <CTRL>W	Shift focus to another panel	RETURN	Select current item	Spacebar	Toggle checkbox	Up arrow or <CTRL>U	Move up one item	Down arrow or <CTRL>N	Move down one item
Text string	Describes a choice that, when selected, either pops up another panel or performs a function. For example, "stop" will stop the SunVTS testing.																
Data entry field	To enter or edit numeric or textual data.																
Checkbox	Represented as "[]". Checkboxes are associated with items and indicate whether the associated item is selected or not. A checkbox can be in one of the following two states: Deselected [] or Selected [*].																
TAB or <CTRL>W	Shift focus to another panel																
RETURN	Select current item																
Spacebar	Toggle checkbox																
Up arrow or <CTRL>U	Move up one item																
Down arrow or <CTRL>N	Move down one item																

Left arrow	or	<CTRL>P	Move left one item
Right arrow	or	<CTRL>R	Move right one item
Backspace			Delete text in a data entry field
ESC			Dismiss a pop-up
<CTRL>F			Scroll forward in a scrollable panel
<CTRL>B			Scroll backward in a scrollable panel
<CTRL>X			Quit vtstty but leave the SunVTS kernel running
<CTRL>L			Refresh the vtstty screen

NOTES

1. To run **vtstty** from a telnet session, carry out the following steps:
 - a. Before telnet-ing, determine the values for "rows and "columns". (See **stty(1)**).
 - b. Set term to the appropriate type after telnet-ing(for example, **set term=vt100**).
 - c. Set the values of columns and rows to the value noted above. (See **stty(1)**).
2. Before running **vtstty** ensure that the environment variable describing the terminal type is set correctly.

SEE ALSO

sunvts(1M), **vtsk(1M)**, **vtsui(1M)**, **vtsui.ol(1M)**, **vtsprobe(1M)**

NAME	vtsui – SunVTS Graphic User Interface (CDE)
SYNOPSIS	vtsui [-qv] [-h <i>hostname</i>]
AVAILABILITY	SUNWvts
DESCRIPTION	<p>The vtsui command starts up the CDE Motif version of SunVTS graphic user interface. There can be multiple instances of vtsui running at the same time, all connected to one SunVTS diagnostic kernel, vtsk(1M). The name of the host machine running the diagnostic kernel, vtsk(1M), will be displayed in the title bar of the graphical user interface window.</p> <p>vtsui is automatically started up by the sunvts (1M) command. vtsui can be also used to start vtsk (1M) if inetd (1M) is in operation. In that case, the security file, sunvts_sec, will be checked for the permission before running vtsk on the target host. See the "SunVTS User's Guide" for a complete description on using the graphical user interface.</p>
OPTIONS	<p>-q Quits the SunVTS graphic user interface when testing has terminated.</p> <p>-v Displays graphic user interface version information only.</p> <p>-h <i>hostname</i> Starts the SunVTS graphic user interface and connects to the SunVTS diagnostic kernel running on <i>hostname</i>, or invokes the kernel if not running, after security checking succeeds. If <i>hostname</i> not specified, the local host is assumed.</p>
EXIT STATUS	<p>The following exit values are returned:</p> <p>0 Successful completion.</p> <p>1 An error occurred.</p>
SEE ALSO	sunvts (1M), vtsk (1M), vtsui.ol (1M), vtstty (1M), vtsprobe (1M)

NAME	vtsui.ol – SunVTS Graphic User Interface (OpenLook)
SYNOPSIS	vtsui.ol [-qv] [-h <i>hostname</i>]
AVAILABILITY	SUNWvts
DESCRIPTION	<p>The vtsui.ol command starts up the OpenLook version of SunVTS graphic user interface. There can be multiple instances of vtsui.ol running at the same time, all connected to one SunVTS diagnostic kernel, vtsk(1M). The name of the host machine running the diagnostic kernel, vtsk(1M), will be displayed in the title bar of the graphic user interface window.</p> <p>vtsui.ol can be used to start vtsk(1M) if inetd(1M) is in operation. In that case, the security file, .sunvts_sec, will be checked for the permission before running vtsk on the target host. vtsui.ol is also automatically started up by the sunvts(1M) command. See the "SunVTS User's Guide" for a complete description on using the graphic user interface.</p>
OPTIONS	<p>-q Quits the SunVTS graphic user interface when testing has terminated.</p> <p>-v Displays graphic user interface version information only.</p> <p>-h <i>hostname</i> Starts the SunVTS graphic user interface and connects to the SunVTS diagnostic kernel running on <i>hostname</i>, or invokes the kernel if not running, after security checking succeeds. If <i>hostname</i> not specified, the local host is assumed.</p>
EXIT STATUS	<p>The following exit values are returned:</p> <p>0 Successful completion.</p> <p>1 An error occurred.</p>
SEE ALSO	sunvts(1M), vtsk(1M), vtsui(1M), vtstty(1M), vtsprobe(1M)

NAME	envmond.conf - configuration file for environment monitor daemon
SYNOPSIS	<code>/usr/platform/SUNW,UltraSPARC-III-Netractor/envmond.conf</code>
DESCRIPTION	<p>The envmond.conf file is the configuration file for envmond(1M), the system environment monitor daemon. The daemon monitors environmental devices to check for conditions that may require some action. The envmond (1M) daemon logs appropriate messages to a system log file via syslogd(1M).</p> <p>Each configuration file entry provides the daemon information about a shared object library, referred to as a policy, which has the knowledge to monitor a device. Each policy entry describes an interface between the envmond daemon and the policy. The policy entry in the envmond.conf file can contain configurable parameters in the <i>policy-args</i> field.</p> <p>All policy entries have the same format:</p> <p><i>poll-interval policy-name policy-args</i></p> <p>The three fields shown above are separated by whitespace. Use the backslash (/) at the end of a line to continue <i>policy-args</i> to the line following.</p> <p>The fields in the envmond.conf file are described as follows:</p> <p><i>poll-interval</i></p> <p>Given in seconds as a decimal number, specifies how often to invoke the policy check function. If <i>poll-interval</i> is 0, the policy check function will never be called.</p> <p><i>policy-name</i></p> <p>The file name, with optional path, of the file implementing the policy. The default location for the policy files is <code>/usr/platform/SUNW,UltraSPARC-III-Netractor/lib/envmond/sparcv9</code></p> <p><i>policy-args</i></p> <p>An optional list of whitespace-separated arguments to be passed to the policy during initialization. The number and format of these arguments is policy-dependent.</p> <p>The following sections describe policies shipped with the implementation of envmond(1M).</p> <p>fancpu Policy</p> <p>The fancpu policy polls I2C slave devices every <i>poll-interval</i> seconds to get the current CPU temperature and the fantray status. If the CPU temperature reaches a warning temperature threshold, a warning message is printed on the system console and to the system log file specified in syslog.conf(4). If the CPU temperature reaches the shutdown temperature, a critical error message is printed on the system console by syslogd(1M). The system is then halted by the shutdown(1M) command. The fan status will be reflected by the corresponding LEDs on the System Status Board, and with log messages sent to syslogd.</p>

powersupply Policy

The powersupply policy sets and clears the power supply LEDs on the System Status Board to reflect power supply status. The policy also handles an interrupt event if a power supply fails.

scsb Policy

The System Controller and Status Board Policy is primarily to configure the scsb driver for cPCI Slot Status LED control. The default **scsb_led_ctrl** setting is false, meaning that the scsb driver controls the cPCI slot LEDs. If **scsb_led_ctrl** is set to true, then some application is responsible for slot LED updates.

EXAMPLES**Example 1: Sample Entries**

The first entry, below, invokes the powersupply shared library every 60 seconds. The second entry specifies that the scsb policy controls the cPCI Slot Status LED.

```
60 powersupply.so
scsb.so scsb_led_ctrl=false
```

FILES

/usr/platform/SUNW,UltraSPARC-III-Netract/
Installation directory.

The following relative pathnames are all beneath the directory named above.

lib/envmond/sparcv9/envmond

Executable for the environmental daemon.

lib/envmond/sparcv9/fancpu.so

Policy for CPU temperature and fan speed control.

lib/envmond/sparcv9/powersupply.so

Policy for power supply monitoring.

SEE ALSO

envmond(1M), **syslogd(1M)**, **syslogd.conf(4)**

NAME	ge – GEM Gigabit-Ethernet device driver
SYNOPSIS	/dev/ge
DESCRIPTION	<p>The ge Sun Gigabit-Ethernet driver is a multi-threaded, loadable, clonable, STREAMS hardware driver supporting the connectionless Data Link Provider Interface, dlpi(7P), over GEM SBus and PCI Gigabit-Ethernet add-in Adapters. Multiple GEM based adapters installed within the system are supported by the driver. The ge driver provides basic support for the GEM based Ethernet hardware and it is used to handle the SUNW,sbus-gem (SBus GEM) and pci108e,2bad (PCI GEM) devices. Functions include chip initialization, frame transmit and receive, multicast and promiscuous support, and error recovery and reporting. The GEM device provides 1000BASE-SX networking interfaces using the GEM ASIC, external SERDES and Fiber optical Transceiver. The GEM ASIC provides the appropriate bus interface, MAC functions and the Physical code sub-layer (PCS) functions. The external SERDES connects to a fiber transceiver and provides the physical connection.</p> <p>The 1000Base-SX standard specifies an “auto-negotiation” protocol to automatically select the mode of operation. In addition to to the duplex mode of the operation, the GEM ASIC can auto-negotiate for IEEE 802.3x Frame Based Flow Control capabilities. The GEM PCS is capable of doing “auto-negotiation” with the remote-end of the link (Link Partner) and receives the capabilities of the remote end. It selects the Highest Common Denominator mode of operation based on the priorities. It also supports forced-mode of operation where the driver can select the mode of operation.</p>
APPLICATION PROGRAMMING INTERFACE ge and DLPI	<p>The cloning character-special device /dev/ge is used to access all ge controllers installed within the system.</p> <p>The ge driver is a “style 2” Data Link Service provider. All M_PROTO and M_PCPROTO type messages are interpreted as DLPI primitives. Valid DLPI primitives are defined in <sys/dlpi.h>. Refer to dlpi(7P) for more information. An explicit DL_ATTACH_REQ message by the user is required to associate the opened stream with a particular device (ppa). The ppa ID is interpreted as an unsigned long data type and indicates the corresponding device instance (unit) number. An error (DL_ERROR_ACK) is returned by the driver if the ppa field value does not correspond to a valid device instance number for this system. The device is initialized on first attach and de-initialized (stopped) at last detach.</p> <p>The values returned by the driver in the DL_INFO_ACK primitive in response to the DL_INFO_REQ from the user are as follows:</p> <ul style="list-style-type: none"> • The maximum SDU is 1500 (ETHERMTU - defined in <sys/ethernet.h>). • The minimum SDU is 0. • The dlsap address length is 8. • The MAC type is DL_ETHER. • The sap length values is -2 meaning the physical address component is followed immediately by a 2 byte sap component within the DLSAP address.

- The service mode is **DL_CLDLS**.
- No optional quality of service (QOS) support is included at present so the QOS fields are **0**.
- The provider style is **DL_STYLE2**.
- The version is **DL_VERSION_2**.
- The broadcast address value is Ethernet/IEEE broadcast address (**0xFFFFF**).

Once in the **DL_ATTACHED** state, the user must send a **DL_BIND_REQ** to associate a particular SAP (Service Access Pointer) with the stream. The **ge** driver interprets the **sap** field within the **DL_BIND_REQ** as an Ethernet “type” therefore valid values for the **sap** field are in the [**0-0xFFFF**] range. Only one Ethernet type can be bound to the stream at any time.

If the user selects a **sap** with a value of **0**, the receiver will be in “802.3 mode”. All frames received from the media having a “type” field in the range [**0-1500**] are assumed to be 802.3 frames and are routed up all open Streams which are bound to **sap** value **0**. If more than one Stream is in “802.3 mode” then the frame will be duplicated and routed up multiple Streams as **DL_UNITDATA_IND** messages.

In transmission, the driver checks the **sap** field of the **DL_BIND_REQ** if the **sap** value is **0**, and if the destination type field is in the range [**0-1500**]. If either is true, the driver computes the length of the message, not including initial **M_PROTO** mblk (message block), of all subsequent **DL_UNITDATA_REQ** messages and transmits 802.3 frames that have this value in the MAC frame header length field.

The **ge** driver **DLSAP** address format consists of the 6 byte physical (Ethernet) address component followed immediately by the 2 byte **sap** (type) component producing an 8 byte **DLSAP** address. Applications should *not* hard code to this particular implementation-specific **DLSAP** address format but use information returned in the **DL_INFO_ACK** primitive to compose and decompose **DLSAP** addresses. The **sap** length, full **DLSAP** length, and **sap**/physical ordering are included within the **DL_INFO_ACK**. The physical address length can be computed by subtracting the **sap** length from the full **DLSAP** address length or by issuing the **DL_PHYS_ADDR_REQ** to obtain the current physical address associated with the stream.

Once in the **DL_BOUND** state, the user may transmit frames on the Ethernet by sending **DL_UNITDATA_REQ** messages to the **ge** driver. The **ge** driver will route received Ethernet frames up all those open and bound streams having a **sap** which matches the Ethernet type as **DL_UNITDATA_IND** messages. Received Ethernet frames are duplicated and routed up multiple open streams if necessary. The **DLSAP** address contained within the **DL_UNITDATA_REQ** and **DL_UNITDATA_IND** messages consists of both the **sap** (type) and physical (Ethernet) components.

In addition to the mandatory connectionless **DLPI** message set the driver additionally supports the following primitives.

ge Primitives

The **DL_ENABMULTI_REQ** and **DL_DISABMULTI_REQ** primitives enable/disable reception of individual multicast group addresses. A set of multicast addresses may be iteratively created and modified on a per-stream basis using these primitives. These

primitives are accepted by the driver in any state following **DL_ATTACHED**.

The **DL_PROMISCON_REQ** and **DL_PROMISCOFF_REQ** primitives with the **DL_PROMISC_PHYS** flag set in the **dl_level** field enables/disables reception of all (“promiscuous mode”) frames on the media including frames generated by the local host.

When used with the **DL_PROMISC_SAP** flag set this enables/disables reception of all **sap** (Ethernet type) values. When used with the **DL_PROMISC_MULTI** flag set this enables/disables reception of all multicast group addresses. The effect of each is always on a per-stream basis and independent of the other **sap** and physical level configurations on this stream or other streams.

The **DL_PHYS_ADDR_REQ** primitive returns the 6 octet Ethernet address currently associated (attached) to the stream in the **DL_PHYS_ADDR_ACK** primitive. This primitive is valid only in states following a successful **DL_ATTACH_REQ**.

The **DL_SET_PHYS_ADDR_REQ** primitive changes the 6 octet Ethernet address currently associated (attached) to this stream. The credentials of the process which originally opened this stream must be superuser. Otherwise **EPERM** is returned in the **DL_ERROR_ACK**. This primitive is destructive in that it affects all other current and future streams attached to this device. An **M_ERROR** is sent up all other streams attached to this device when this primitive is successful on this stream. Once changed, all streams subsequently opened and attached to this device will obtain this new physical address. Once changed, the physical address will remain until this primitive is used to change the physical address again or the system is rebooted, whichever comes first.

ge DRIVER

By default, the ge driver performs “auto-negotiation” to select the **mode** and **flow control capabilities** of the link.

The link can be in one of the 4 following modes:

- 1000 Mbps, full-duplex
- 1000 Mbps, half-duplex
- Symmetric Pause
- Asymmetric Pause

These speeds and modes are described in the 1000Base-TX standard.

The *auto-negotiation* protocol automatically selects:

- Operation mode (half-duplex or full-duplex)
- Flow Control Capability (Symmetric and/or Asymmetric)

The *auto-negotiation* protocol does the following:

- Gets all the modes of operation supported by the Link Partner
- Advertises its capabilities to the Link Partner
- Selects the highest common denominator mode of operation based on the priorities

The *GEM Hardware* is capable of all of the operating modes listed above, when by *default*, auto-negotiation is used to bring up the link and select the common mode of operation with the Link Partner. The PCS also supports *forced-mode* of operation in

which the driver can select the mode of operation and the flow control capabilities, using the *ndd* utility.

The **GEM** device also supports programmable “**IPG**” (Inter-Packet Gap) parameters **ipg1** and **ipg2**. By default, the driver sets **ipg1** to 8 **byte-times** and **ipg2** to 4 **byte-times** (which are the standard values). Sometimes, the user may want to alter these values from the standard 1000 Mbps **IPG** set to 0.096 microseconds.

ge Parameter List

The ge driver provides for setting and getting various parameters for the **GEM** device. The parameter list includes **current transceiver status**, **current link status**, **inter-packet gap**, **PCS capabilities** and **link partner capabilities**.

The PCS has two set of capabilities: one set reflects the capabilities of the **hardware**, which are **read-only (RO)** parameters and the second set reflects the values chosen by the user and is used in **speed selection**. There are **read/write (RW)** capabilities. At boot time, these two sets of capabilities will be the same. The Link Partner capabilities are also read only parameters because the current default value of these parameters can only be read and cannot be modified.

FILES

<code>/dev/ge</code>	ge special character device.
<code>/kernel/drv/ge.conf</code>	System wide default device driver properties

SEE ALSO

ndd(1M), **netstat(1M)**, **driver.conf(4)**, **dlpi(7P)**, **ie(7D)**, **le(7D)** **hme(7D)** **qfe(7D)**

NAME	hsi – S-Bus based high speed serial line interface.
SYNOPSIS	<pre>#include <fcntl.h> open(/dev/hihn, mode); open(/dev/hih, mode);</pre>
DESCRIPTION	<p>The hsi module is a loadable and unloadable STREAMS driver that implements the sending and receiving of data packets such as HDLC frames over synchronous serial lines. The hsi driver is a standalone driver that supports HSI/S S-Bus based serial interface hardware and provides physical level data transfer services for upper data link layer protocols (e.g. HDLC or SDLC).</p> <p>The hihn devices provide what is known as a data path which supports the transfer of data via read(2) and write(2) system calls, as well as ioctl(2) calls. Data path opens are exclusive in order to protect against injection or diversion of data by another process.</p> <p>The hih device provides a separate control path for use by programs that need to configure or monitor a connection independent of any exclusive access restrictions imposed by data path opens. Up to three control paths may be active on a particular serial channel at any one time. Control path accesses are restricted to ioctl(2) calls only; no data transfer is possible.</p> <p>When used in synchronous modes, the Z16C35 ISCC supports several options for clock sourcing and data encoding. Both the transmit and receive clock sources can be set to be the external receive clock (RTxC) and the internal baud rate generator (BRG). Additionally, the transmit clock source can be set to the external transmit clock (TRxC).</p> <p>The baud rate generator is a programmable divisor that derives a clock frequency from the PCLK input signal to the ISCC. A programmed baud rate is translated into a 16-bit time constant that is stored in the ISCC. When using the BRG as a clock source the driver may answer a query of its current speed with a value different from the one specified. This is because baud rates translate into time constants in discrete steps, and reverse translation shows the change. If an exact baud rate is required that cannot be obtained with the BRG, an external clock source must be selected.</p> <p>A local loopback mode is available, primarily for use by the hsi_loop(1M) utility for testing purposes, and should not be confused with SDLC loop mode, which is not supported on this interface. This option should be selected casually, or left in use when not needed.</p> <p>The hsi driver keeps running totals of various hardware generated events for each channel. These include numbers of packets and characters sent and received, abort conditions detected by the receiver, receive CRC errors, transmit underruns, receive overruns, input errors and output errors, and message block allocation failures. Input errors are logged whenever an incoming message must be discarded, such as when an abort or CRC error is detected, a receive overrun occurs, or when no message block is available to store incoming data. Output errors are logged when the data must be discarded due to underruns, CTS drops during transmission, CTS timeouts, or excessive watchdog timeouts caused by a cable break.</p>

IOCTLS

The **hsi** driver supports several **ioctl()** commands, including:

- S_IOCGETMODE** Return a **struct scc_mode** containing parameters currently in use. These include the transmit and receive clock sources, boolean loop-back and NRZI mode flags and the integer baudrate.
- S_IOCSETMODE** The argument is a **struct scc_mode** from which the ISCC channel will be programmed.
- S_IOCGETSTATS** Return a **struct hs_stats** containing the current totals of hardware-generated events. These include numbers of packets and characters sent and received by the driver, aborts and CRC errors detected, transmit underruns, and receive overruns.
- S_IOCCLRSTATS** Clear the hardware statistics for this channel.
- S_IOCGETSPEED** Returns the currently set baudrate as an integer. This may not reflect the actual data transfer rate if external clocks are used.
- S_IOCGETMCTL** Returns the current state of the CTS and DCD incoming modem interface signals as an integer.

The following structures are used with **hsi** **ioctl()** commands:

```
struct scc_mode {
    charsm_txclock; /* transmit clock sources */
    charsm_rxclock; /* receive clock sources */
    charsm_iflags; /* data and clock inversion flags (non-zsh) */
    u_char      sm_config; /* boolean configuration options */
    int  sm_baudrate; /* real baud rate */
    int  sm_retval; /* reason codes for ioctl failures */
};

struct hs_stats {
    unsigned int ipack; /* input packets */
    unsigned int opack; /* output packets */
    unsigned int ichar; /* input bytes */
    unsigned int ochar; /* output bytes */
    int  abort; /* abort received */
    int  crc; /* CRC error */
    int  cts; /* CTS timeouts */
    int  dcd; /* Carrier drops */
    int  overrun; /* receive overrun */
    int  underrun; /* transmit underrun */
    int  ierror; /* input error */
    int  oerror; /* output error */
    int  nobuffers; /* rcv side memory allocation failure */
    int  ishort; /* input packet too short (< CRC-bytes+1) */
    int  ilong; /* input packet too long (> mru) */
    int  inactive; /* input packet rcvd when rcv is inactive */
    int  idma; /* receive dma error */
};
```

```

int olong;    /* output packet too long (> mtu) */
int ohung;    /* transmit hung (usually missing clock) */
int odma;     /* transmit dma error */
};

```

ERRORS

An **open()** will fail if a STREAMS message block cannot be allocated, or:

ENXIO The unit being opened does not exist.
EBUSY The device is in use by another serial protocol.

An **ioctl()** will fail if:

EINVAL An attempt was made to select an invalid clocking source.
EINVAL The baud rate specified for use with the baud rate generator would translate to a null time constant in the ISCC's registers.

FILES

/dev/hih[0-n], /dev/hih
Character-special devices.
/usr/include/sys/ser_sync.h
Header file specifying synchronous serial communication definitions.

SEE ALSO

hsi_init(1M), hsi_loop(1M), hsi_stat(1M), hsi_trace(1M)

Refer to the *Zilog Z16C35 ISCC Serial Communications Controller Technical Manual* for details of the ISCC's operation and capabilities.

DIAGNOSTICS

hih data open failed, no memory, rq=nnn

hih clone open failed, no memory, rq=nnn A kernel memory allocation failed for one of the private data structures. The value of *nnn* is the address of the read queue passed to **open(2)**.

hih_open: can't alloc message block

The open could not proceed because an initial STREAMS message block could not be made available for incoming data.

hih: clone device *d* must be attached before use!

An operation was attempted through a control path before that path had been attached to a particular serial channel.

hihn: invalid operation for clone dev.

An inappropriate STREAMS message type was passed through a control path. Only **M_IOCTL** and **M_PROTO** message types are permitted.

hihn: not initialized, can't send message

An **M_DATA** message was passed to the driver for a channel that had not been programmed at least once since the driver was loaded. The ISCC's registers were in an unknown state. The **S_IOCSETMODE** **ioctl** command performs the programming operation.

hihn: transmit hung

The transmitter was not successfully restarted after the watchdog timer

expired.

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 in 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: hih_init: changed baudrate from 100000 to 99512.

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

NAME	hsip – PCI-Bus based high speed serial line interface.
SYNOPSIS	<pre>#include <fcntl.h> #include </usr/include/sys/ser_sync.h> open(/dev/hihp<i>n</i>, mode); open(/dev/hihp, mode);</pre>
DESCRIPTION	<p>The hsip module is a loadable and unloadable STREAMS driver that implements the sending and receiving of data packets such as HDLC frames over synchronous serial lines. The hsip driver is a standalone driver that supports HSI/P PCI-Bus based serial interface hardware and provides physical level data transfer services for upper data link layer protocols (e.g. HDLC or SDLC).</p> <p>The hihp<i>n</i> devices provide what is known as a data path which supports the transfer of data via read(2) and write(2) system calls, as well as ioctl(2) calls. Data path opens are exclusive in order to protect against injection or diversion of data by another process.</p> <p>The hihp device provides a separate control path for use by programs that need to configure or monitor a connection independent of any exclusive access restrictions imposed by data path opens. Up to three control paths may be active on a particular serial channel at any one time. Control path accesses are restricted to ioctl(2) calls only; no data transfer is possible.</p> <p>The HSIP ports support several options for clock sourcing and data encoding. Both the transmit and receive clock sources can be set to be the external transmit clock (TxC), external receive clock (RxC), the internal baud rate generator (BRG), or the output of the SCC's Digital Phase-Lock Loop (DPLL).</p> <p>The baud rate generator is a programmable divisor that derives a clock frequency from the PCLK input signal to the SCC. A programmed baud rate is translated into a 16-bit time constant that is stored in the SCC. When using the BRG as a clock source the driver may answer a query of its current speed with a value different from the one specified. This is because baud rates translate into time constants in discrete steps, and reverse translation shows the change. If an exact baud rate is required that cannot be obtained with the BRG, an external clock source must be selected.</p> <p>Use of the DPLL option requires the selection of NRZI data encoding and the setting of a non-zero value for the baud rate, because the DPLL uses the BRG as its reference clock source.</p> <p>A local loopback mode is available, primarily for use by the hsip_loop(1m) utility for testing purposes, and should not be confused with SDLC loop mode, which is not supported on this interface. Also, an auto-echo feature may be selected that causes all incoming data to be routed to the transmit data line, allowing the port to act as the remote end of a digital loop. Neither of these options should be selected casually, or left in use when not needed.</p>

The **hsip** driver keeps running totals of various hardware generated events for each channel. These include numbers of packets and characters sent and received, abort conditions detected by the receiver, receive CRC errors, transmit underruns, receive overruns, input errors and output errors. Input errors are logged whenever an incoming message must be discarded, such as when an abort or CRC error is detected, a receive overrun occurs, or when no message block is available to store incoming data. Output errors are logged when the data must be discarded due to underruns, CTS drops during transmission, CTS timeouts, or excessive watchdog timeouts caused by a cable break.

IOCTLS

The **hsip** driver supports several **ioctl()** commands, including:

- S_IOCGETMODE** Return a **struct scc_mode** containing parameters currently in use. These include the transmit and receive clock sources, boolean loop-back and NRZI mode flags and the integer baudrate.
- S_IOCSETMODE** The argument is a **struct scc_mode** from which the SCC channel will be programmed.
- S_IOCGETSTATS** Return a **struct sl_stats** containing the current totals of hardware-generated events. These include numbers of packets and characters sent and received by the driver, aborts and CRC errors detected, transmit underruns, and receive overruns.
- S_IOCCLRSTATS** Clear the hardware statistics for this channel.
- S_IOCGETSPEED** Returns the currently set baudrate as an integer. This may not reflect the actual data transfer rate if external clocks are used.
- S_IOCGETMCTL** Returns the current state of the CTS and DCD incoming modem interface signals as an integer.

The following structures are used with **hsip** **ioctl()** commands:

```

struct scc_mode {
    char    sm_txclock;    /* transmit clock sources */
    char    sm_rxclock;    /* receive clock sources */
    char    sm_iflags;     /* data and clock inversion flags (non-zsh) */
    u_char  sm_config;     /* boolean configuration options */
    int     sm_baudrate;   /* real baud rate */
    int     sm_retval;     /* reason codes for ioctl failures */
};

struct sl_stats {
    int     ipack;         /* input packets */
    int     opack;         /* output packets */
    int     ichar;         /* input bytes */
    int     ochar;         /* output bytes */
    int     abort;         /* abort received */
    int     crc;           /* CRC error */
    int     cts;           /* CTS timeouts */
};

```

```

int    dcd;           /* Carrier drops */
int    overrun;      /* receive overrun */
int    underrun;     /* transmit underrun */
int    ierror;       /* input error */
int    oerror;       /* output error */
int    nobuffers;    /* receive side memory allocation failure */
};

```

ERRORS

An **open()** will fail if a STREAMS message block cannot be allocated, or:

ENXIO The unit being opened does not exist.

EBUSY The device is in use by another serial protocol.

An **ioctl()** will fail if:

EINVAL An attempt was made to select an invalid clocking source.

EINVAL The baud rate specified for use with the baud rate generator would translate to a null time constant in the SCC's registers.

FILES

/dev/hihp[0-n], /dev/hihp

Character-special devices.

/usr/include/sys/ser_sync.h

Header file specifying synchronous serial communication definitions.

SEE ALSO

hsip_init(1M), hsip_loop(1M), hsip_stat(1M),

Refer to the *Motorola MC68360 Quad Integrated Communications Controller Technical Manual* for details of the SCC's operation and capabilities.

DIAGNOSTICS

hihp data open failed, no memory, rq=nnn

hihp clone open failed, no memory, rq=nnn A kernel memory allocation failed for one of the private data structures. The value of *nnn* is the address of the read queue passed to **open(2)**.

hihp_open: can't alloc message block

The open could not proceed because an initial STREAMS message block could not be made available for incoming data.

hihp: clone device *d* must be attached before use!

An operation was attempted through a control path before that path had been attached to a particular serial channel.

hihpn: invalid operation for clone dev.

An inappropriate STREAMS message type was passed through a control path. Only **M_IOCTL** and **M_PROTO** message types are permitted.

hihpn: not initialized, can't send message

An **M_DATA** message was passed to the driver for a channel that had not been programmed at least once since the driver was loaded. The **S_IOCSETMODE** **ioctl** command performs the programming operation.

hihp:n: transmit hung

The transmitter was not successfully restarted after the watchdog timer expired.

NAME	nf – FDDI device driver
SYNOPSIS	<pre>#include <sys/nf.h> #include <sys/dlpi.h></pre>
DESCRIPTION	<p>nf is a multi-threaded, loadable, clonable, STREAMS hardware device driver supporting the connectionless Data Link Provider Interface, dlpi(7), over DP83265A (BSI-2) FDDI controller in the SBus card. There is no fixed limitation on the number of FDDI cards supported by the driver. The nf driver provides basic support for the BSI-2, BMAC and PLAYER+ hardware. Functions include chip initialization, frame transmit and receive, multicast and promiscuous support, and error recovery and reporting. The cloning character-special device <code>/dev/nf</code> is used to access BSI-2 controller installed within the system.</p>
nf and DLPI	<p>The nf driver is a “style 2” Data Link Service provider. All <code>M_PROTO</code> and <code>M_PCPROTO</code> type msgs are interpreted as DLPI primitives. An explicit <code>DL_ATTACH_REQ</code> message by the user is required to associate the opened stream with a particular device (ppa). The ppa ID is interpreted as an unsigned long and indicates the corresponding device instance (unit) number. An error (<code>DL_ERROR_ACK</code>) is returned by the driver if the ppa field value does not correspond to a valid device instance number for this system. The device is initialized on first attach and de-initialized (stopped) on last detach.</p> <p>The values returned by the driver in the <code>DL_INFO_ACK</code> primitive in response to the <code>DL_INFO_REQ</code> from the user are as follows:</p> <ul style="list-style-type: none"> • The max SDU is 4352 (FDDIMTU). • The min SDU is 0. • The dlsap address length is 8. • The MAC type is <code>DL_FDDI</code>. • The sap length value is -2 meaning the physical address component is followed immediately by a 2 byte sap component within the DLSAP address. • The service mode is <code>DL_CLDLS</code>. • No optional quality of service (QOS) support is included at present so the QOS fields are 0. • The provider style is <code>DL_STYLE2</code>. • The version is <code>DL_VERSION_2</code>. • The broadcast address value is Ethernet/IEEE broadcast address (0xFFFFFFFF).

Once in the DL_ATTACHED state, the user must send a DL_BIND_REQ to associate a particular SAP (Service Access Pointer) with the stream. The **nf** driver interprets the **sap** field within the DL_BIND_REQ as an Ethernet “type” therefore valid values for the **sap** field are in the [0-0xFFFF] range. Only one Ethernet type can be bound to the stream at any time.

In addition to Ethernet V2 service, an “802.3 mode” is provided by the driver and works as follows. **sap** value 0 is treated as equivalent and represent a desire by the user for “802.3 mode”. If the value of the **sap** field of the DL_BIND_REQ is 0, then the driver computes the length of the message, not including initial M_PROTO mblk, of all subsequent DL_UNITDATA_REQ messages and transmits 802.3 frames having this value in the MAC frame header length field and a value of 0xaaaa030000 in the snap header. All frames received from the media having a “type” field in the range [0-1500] are assumed to be 802.3 frames and are routed up all open streams which are bound to **sap** value 0. If more than one stream is in “802.3 mode” then the frame will be duplicated and routed up multiple streams as DL_UNITDATA_IND messages.

The **nf** driver DLSAP address format consists of the 6 byte physical (FDDI) address component followed immediately by the 2 byte **sap** (type) component producing an 8 byte DLSAP address. Applications should *not* hardcode to this particular implementation-specific DLSAP address format but use information returned in the DL_INFO_ACK primitive to compose and decompose DLSAP addresses. The **sap** length, full DLSAP length, and **sap**/physical ordering are included within the DL_INFO_ACK. The physical address length can be computed by subtracting the **sap** length from the full DLSAP address length or by issuing the DL_PHYS_ADDR_REQ to obtain the current physical address associated with the stream.

Once in the DL_BOUND state, the user may transmit frames on the FDDI ring by sending DL_UNITDATA_REQ messages to the **nf** driver. The **nf** driver will route received FDDI frames up all those open and bound streams having a **sap** which matches the type as DL_UNITDATA_IND messages. Received FDDI frames are duplicated and routed up multiple open streams if necessary. The DLSAP address contained within the DL_UNITDATA_REQ and DL_UNITDATA_IND messages consists of both the **sap** (type) and physical (FDDI) components.

nf Primitives

In addition to the mandatory connectionless DLPI message set the driver additionally supports the following primitives.

The DL_ENABMULTI_REQ and DL_DISABMULTI_REQ primitives enable/disable reception of individual multicast group addresses. A set of multicast addresses may be iteratively created and modified on a per-stream basis using these primitives. These primitives are accepted by the driver in any state following DL_ATTACHED.

The DL_PROMISCON_REQ and DL_PROMISCOFF_REQ primitives with the DL_PROMISC_PHYS flag set in the dl_level field enables/disables reception of all (“promiscuous mode”) frames on the media including frames generated by the local host. When used with the DL_PROMISC_SAP flag set this enables/disables reception of all **sap** (Ethernet type) values. When used with the DL_PROMISC_MULTI flag set this enables/disables reception of all multicast group addresses. The effect of each is

always on a per-stream basis and independent of the other **sap** and physical level configurations on this stream or other streams.

The `DL_PHYS_ADDR_REQ` primitive return the 6 octet MAC address currently associated (attached) to the stream in the `DL_PHYS_ADDR_ACK` primitive. This primitive is valid only in states following a successful `DL_ATTACH_REQ`.

The `DL_SET_PHYS_ADDR_REQ` primitive changes the 6 octet MAC address currently associated (attached) to this stream. The credentials of the process which originally opened this stream must be superuser or `EPERM` is returned in the `DL_ERROR_ACK`. This primitive is destructive in that it affects all other current and future streams attached to this device. An `M_ERROR` is sent up all other streams attached to this device when this primitive on this stream is successful. Once changed, all streams subsequently opened and attached to this device will obtain this new physical address. Once changed, the physical address will remain so until this primitive is used to change the physical address again or the system is rebooted, whichever comes first.

By default the first interface will use the systems MAC address but subsequent interfaces will use the FDDI local address.

FILES

`/dev/nf`

SEE ALSO

`smt(7)`, `dlpi(7)`,

NAME	pf – FDDI device driver
SYNOPSIS	#include <sys/pf.h> #include <sys/dlpi.h>
DESCRIPTION	<p>pf is a multi-threaded, loadable, clonable, STREAMS hardware device driver supporting the connectionless Data Link Provider Interface, dlpi(7), over PBS FDDI controller in the PCI card. The driver also provides support for Applications to get statistics and status of Station Management. There is no fixed limitation on the number of FDDI cards supported by the driver. The pf driver provides basic support for the PBS, BMAC and PLAYER+ hardware. Functions include chip initialization, LLC/SMT frame transmit and receive, multicast and promiscuous support, and error recovery and reporting.</p> <p>The cloning character-special device /dev/pf is used to access PBS controller installed within the system.</p>
pf and DLPI	<p>The pf driver is a “style 2” Data Link Service provider. All M_PROTO and M_PCPROTO type msgs are interpreted as DLPI primitives. An explicit DL_ATTACH_REQ message by the user is required to associate the opened stream with a particular device (ppa). The ppa ID is interpreted as an unsigned long and indicates the corresponding device instance (unit) number. An error (DL_ERROR_ACK) is returned by the driver if the ppa field value does not correspond to a valid device instance number for this system. The device is initialized on first attach and de-initialized (stopped) on last detach.</p> <p>The values returned by the driver in the DL_INFO_ACK primitive in response to the DL_INFO_REQ from the user are as follows:</p> <ul style="list-style-type: none"> • The max SDU is 4352 (FDDIMTU). • The min SDU is 0. • The dlsap address length is 8. • The MAC type is DL_FDDI. • The sap length value is -2 meaning the physical address component is followed immediately by a 2 byte sap component within the DLSAP address. • The service mode is DL_CLDLS. • No optional quality of service (QOS) support is included at present so the QOS fields are 0. • The provider style is DL_STYLE2. • The version is DL_VERSION_2. • The broadcast address value is Ethernet/IEEE broadcast address (0xFFFFFFFF).

Once in the DL_ATTACHED state, the user must send a DL_BIND_REQ to associate a particular SAP (Service Access Pointer) with the stream. The **pf** driver interprets the **sap** field within the DL_BIND_REQ as an Ethernet “type” therefore valid values for the **sap** field are in the [0-0xFFFF] range. Only one Ethernet type can be bound to the stream at any time.

In addition to Ethernet V2 service, an “802.3 mode” is provided by the driver and works as follows. **sap** value 0 is treated as equivalent and represent a desire by the user for “802.3 mode”. If the value of the **sap** field of the DL_BIND_REQ is 0, then the driver computes the length of the message, not including initial M_PROTO mblk, of all subsequent DL_UNITDATA_REQ messages and transmits 802.3 frames having this value in the MAC frame header length field and a value of 0xaaaa030000 in the snap header. All frames received from the media having a “type” field in the range [0-1500] are assumed to be 802.3 frames and are routed up all open streams which are bound to **sap** value 0. If more than one stream is in “802.3 mode” then the frame will be duplicated and routed up multiple streams as DL_UNITDATA_IND messages.

The **pf** driver DLSAP address format consists of the 6 byte physical (FDDI) address component followed immediately by the 2 byte **sap** (type) component producing an 8 byte DLSAP address. Applications should *not* hardcode to this particular implementation-specific DLSAP address format but use information returned in the DL_INFO_ACK primitive to compose and decompose DLSAP addresses. The **sap** length, full DLSAP length, and **sap**/physical ordering are included within the DL_INFO_ACK. The physical address length can be computed by subtracting the **sap** length from the full DLSAP address length or by issuing the DL_PHYS_ADDR_REQ to obtain the current physical address associated with the stream.

Once in the DL_BOUND state, the user may transmit frames on the FDDI ring by sending DL_UNITDATA_REQ messages to the **pf** driver. The **pf** driver will route received FDDI frames up all those open and bound streams having a **sap** which matches the type as DL_UNITDATA_IND messages. Received FDDI frames are duplicated and routed up multiple open streams if necessary. The DLSAP address contained within the DL_UNITDATA_REQ and DL_UNITDATA_IND messages consists of both the **sap** (type) and physical (FDDI) components.

pf Primitives

In addition to the mandatory connectionless DLPI message set the driver additionally supports the following primitives.

The DL_ENABMULTI_REQ and DL_DISABMULTI_REQ primitives enable/disable reception of individual multicast group addresses. A set of multicast addresses may be iteratively created and modified on a per-stream basis using these primitives. These primitives are accepted by the driver in any state following DL_ATTACHED.

The DL_PROMISCON_REQ and DL_PROMISCOFF_REQ primitives with the DL_PROMISC_PHYS flag set in the dl_level field enables/disables reception of all (“promiscuous mode”) frames on the media including frames generated by the local host. When used with the DL_PROMISC_SAP flag set this enables/disables reception of all **sap** (Ethernet type) values. When used with the DL_PROMISC_MULTI flag set this enables/disables reception of all multicast group addresses. The effect of each is

always on a per-stream basis and independent of the other **sap** and physical level configurations on this stream or other streams.

The `DL_PHYS_ADDR_REQ` primitive return the 6 octet MAC address currently associated (attached) to the stream in the `DL_PHYS_ADDR_ACK` primitive. This primitive is valid only in states following a successful `DL_ATTACH_REQ`.

The `DL_SET_PHYS_ADDR_REQ` primitive changes the 6 octet MAC address currently associated (attached) to this stream. The credentials of the process which originally opened this stream must be superuser or `EPERM` is returned in the `DL_ERROR_ACK`. This primitive is destructive in that it affects all other current and future streams attached to this device. An `M_ERROR` is sent up all other streams attached to this device when this primitive on this stream is successful. Once changed, all streams subsequently opened and attached to this device will obtain this new physical address. Once changed, the physical address will remain so until this primitive is used to change the physical address again or the system is rebooted, whichever comes first.

By default the first interface will use the systems MAC address but subsequent interfaces will use the FDDI local address.

pf and SMT

The driver provides information on its PHYs and some FORMAC error counters.

The user has to include these two lines in the program before the line `'#include <pfsmt.h>'`

```
#define      SMT7_2      0
#define      CFG_YES    1
```

The cloning character special device `/dev/pf` is used to access the driver. An explicit `DL_ATTACH_REQ` message by the user is required to associate the opened stream with a particular device(`ppa`) where `ppa` corresponds to the interface instance number.

Once in the `DL_ATTACHED` state, the user need not send a `DL_BIND_REQ`. The user can interact with the driver with `ioctl(2)` calls. The arguments for the `ioctl` are

```
ioctl (int fd, int request, SMTCB *smtcb)
```

The request is **smt** driver specific and can be `SMT_GET` or `SMT_ACT`. `SMTCB` is defined as follows in the header file `pfsmt.h`

```
typedef struct {
    int      command;
    int      sub_command;
    int      param1;
    int      param2;
    int      param3;
    char     *where;
    int      length;
} SMTCB;
SMT_GET:
```

SMT_GET provides a variety of functions such as to read the HPC registers and to get the smt status. command field of smtp should be initialized to one of the following values

```
HPC_BMAC1_REGS      : To read the BMAC registers
HPC_READ            : To read the HPC registers
HPC_PORT1_REGS     : To read RMT port1
and HPC_PORT2_REGS and port2 registers
```

Some of the commands provide sub commands. The field sub_command should be initialized to these sub commands.

1. HPC_BMAC1_REGS

HPC_BMAC1_REGS enables the user to read the BMAC registers. HPC_BMAC1_REGS provides two sub commands GET_COUNTER_GROUP and GET_NEIGHBOR_ADDR. GET_COUNTER_GROUP is used to get various SMT counter values.

GET_COUNTER_GROUP needs the SMTCB *smtp to be initialized as follows

```
COUNTER_GROUP ct;
smtp->command = HPC_BMAC1_REGS;
smtp->sub_command = GET_COUNTER_GROUP;
smtp->where = (char *) &ct;
smtp->length = sizeof (ct);
```

GET_NEIGHBOR_ADDR enables the user to get the MAC address of the Neighbour station. GET_NEIGHBOR_ADDR needs the SMTCB *smtp to be initialized as follows

```
char      addr_buf[12];
smtp->command = HPC_BMAC1_REGS;
smtp->sub_command = GET_NEIGHBOR_ADDR;
smtp->where = addr_buf;
smtp->length = 12;
```

2. HPC_READ

HPC_READ enables the user to read the HPC registers. HPC_READ does not provide any sub commands. HPC_READ needs the SMTCB *smtp to be initialized as follows

```
smtp->command = HPC_READ;
smtp->param1 = HPC_READ | HPC_SIZE_BYTE
             | <HPC_reg_offset>;
smtp->where = (char *) smtp;
```

where HPC_register_offset offset is set of register space provided by the HPC. For the set of reister offsets refer to the file pfsmt.h

3. HPC_PORT1_REGS and HPC_PORT2_REGS

HPC_PORT1_REGS enables the user to get the status of the Connection Management. HPC_PORT2_REGS is for the second port if the interface is a DAS. The sub command for HPC_PORT1_REGS is GET_PORT_GROUP. HPC_PORT1_REGS needs the SMTCB *smtp to be initialized as follows

```
FDDI_PORT_GROUP port; smtp->command = HPC_PORT1_REGS;
smtp->sub_command = GET_PORT_GROUP;
smtp->where = (char *) &port;
smtp->length = sizeof (port);
```

The two important status returned in the structure port are port.ecm_state and port.pcm_state. port.ecm_state corresponds to the current state of the ECM state machine. The valid values are OUT, IN, TRACE, PATHTEST, INSERT, CHECK and DEINSERT. The value returned in port.ecm_state is the index into the list of the ECM States. port.pcm_state corresponds to the current state of the PCM state machine. The Valid values are OFF, BREAK, TRACE, CONNECT, NEXT, SIGNAL, JOIN, VERIFY, ACTIVE, MAINT. The value returned in port.pcm_state in an index into the list of PCM States.

SMT_ACT:

SMT_ACT is supported to set the state of the smt driver. The command field should always be set to SMT_CTL. SMT_ACT provides two sub commands SMT_ACCEPT_FRAME and SMT_CLOSE. SMT_ACCEPT_FRAME needs to be used when any SMT API client is active.

```
smtp->command = SMT_CTL;
smtp->sub_command = SMT_ACCEPT_FRAME;
```

SMT_CLOSE needs to be used when the API client exits.

```
smtp->command = SMT_CTL;
smtp->sub_command = SMT_CLOSE;
```

To transmit SMT NSA frames the user should bind to FDDI_NSA sap. To transmit other SMT frames the user may bind to FDDI_SMTINFO sap.

FILES

/dev/pf

SEE ALSO

dlpi(7)

NAME	smt – FDDI SMT Apps Interface device driver
SYNOPSIS	#include <sys/nfsmt.h>
DESCRIPTION	<p>smt is a multi-threaded, loadable, clonable, STREAMS device driver supporting Data Link Provider Interface, dlpi(7), for Application programs to get the statistics and status of the Station Management. smt driver provides packet throughput statistics, reconfiguration events and interface exceptions. It also provides the information on its PHYs and some FORMAC error counters.</p> <p>The user has to include these two lines in the program before the line '#include <nfsmt.h>'</p> <pre>#define SMT7_2 0 #define CFG_YES 1</pre> <p>The cloning character special device /dev/smt is used to access the driver. An explicit DL_ATTACH_REQ message by the user is required to associate the opened stream with a particular device(ppa) where ppa corresponds to the interface instance number. Once in the DL_ATTACHED state, the user need not send a DL_BIND_REQ. The user can interact with the driver with ioctl(2) calls. The arguments for the ioctl are</p> <pre>ioctl (int fd, int request, SMTCB *smtp)</pre> <p>The request is smt driver specific and can be SMT_GET or SMT_ACT. SMTCB is defined as follows in the header file nfsmt.h</p> <pre>typedef struct { int command; int sub_command; int param1; int param2; int param3; char *where; int length; } SMTCB;</pre> <p>SMT_GET:</p> <p>SMT_GET provides a variety of functions such as to read the HPC registers and to get the smt status. command field of smtp should be initialized to one of the following values</p> <pre>HPC_BMAC1_REGS : To read the BMAC registers HPC_READ : To read the HPC registers HPC_PORT1_REGS : To read RMT port1 and HPC_PORT2_REGS and port2 registers</pre> <p>Some of the commands provide sub commands. The field sub_command should be initialized to these sub commands.</p>

1. HPC_BMAC1_REGS

HPC_BMAC1_REGS enables the user to read the BMAC registers. HPC_BMAC1_REGS provides two sub commands GET_COUNTER_GROUP and GET_NEIGHBOR_ADDR. GET_COUNTER_GROUP is used to get various SMT counter values.

GET_COUNTER_GROUP needs the SMTCB *smtp to be initialized as follows

```
COUNTER_GROUP ct;
smtp->command = HPC_BMAC1_REGS;
smtp->sub_command = GET_COUNTER_GROUP;
smtp->where = (char *) &ct;
smtp->length = sizeof (ct);
```

GET_NEIGHBOR_ADDR enables the user to get the MAC address of the Neighbour station. GET_NEIGHBOR_ADDR needs the SMTCB *smtp to be initialized as follows

```
char addr_buf[12];
smtp->command = HPC_BMAC1_REGS;
smtp->sub_command = GET_NEIGHBOR_ADDR;
smtp->where = addr_buf;
smtp->length = 12;
```

2. HPC_READ

HPC_READ enables the user to read the HPC registers. HPC_READ does not provide any sub commands. HPC_READ needs the SMTCB *smtp to be initialized as follows

```
smtp->command = HPC_READ;
smtp->param1 = HPC_READ | HPC_SIZE_BYTE
             | <HPC_reg_offset>;
smtp->where = (char *) smtp;
```

where HPC_register_offset offset is set of register space provided by the HPC. For the set of reister offsets refer to the file nfsmt.h

3. HPC_PORT1_REGS and HPC_PORT2_REGS

HPC_PORT1_REGS enables the user to get the status of the Connection Management. HPC_PORT2_REGS is for the second port if the interface is a DAS. The sub command for HPC_PORT1_REGS is GET_PORT_GROUP. HPC_PORT1_REGS needs the SMTCB *smtp to be initialized as follows

```
FDDI_PORT_GROUP port;
smtp->command = HPC_PORT1_REGS;
smtp->sub_command = GET_PORT_GROUP;
smtp->where = (char *) &port;
smtp->length = sizeof (port);
```

The two important status returned in the structure port are port.ecm_state and port.pcm_state. port.ecm_state corresponds to the current state of the ECM state machine. The valid values are OUT, IN, TRACE, PATHTEST, INSERT, CHECK and DEINSERT. The value returned in port.ecm_state is the index into the list of the ECM

States. `port.pcm_state` corresponds to the current state of the PCM state machine. The Valid values are OFF, BREAK, TRACE, CONNECT, NEXT, SIGNAL, JOIN, VERIFY, ACTIVE, MAINT. The value returned in `port.pcm_state` is an index into the list of PCM States.

SMT_ACT:

SMT_ACT is supported to set the state of the smt driver. The command field should always be set to SMT_CTL. SMT_ACT provides two sub commands

SMT_ACCEPT_FRAME and SMT_CLOSE. SMT_ACCEPT_FRAME needs to be used when any SMT API client is active.

```
smtp->command = SMT_CTL;
smtp->sub_command = SMT_ACCEPT_FRAME;
```

SMT_CLOSE needs to be used when the API client exits.

```
smtp->command = SMT_CTL;
smtp->sub_command = SMT_CLOSE;
```

FILES /dev/smt

SEE ALSO nf(7), dlpi(7),

Index

C

cdrw (1), 1-1

D

device and network interfaces, 7-71 to 7-93

E

envmond (1M), 1M-5
envmond.conf (4), 4-69

F

file formats, 4-69 to 4-70

G

ge (7D), 7-71

H

hsi (7D), 7-75
hsi_init (1M), 1M-6
hsi_loop (1M), 1M-9
hsi_stat (1M), 1M-12
hsi_trace (1M), 1M-15
hsip (7D), 7-79
hsip_init (1M), 1M-18

hsip_loop (1M), 1M-21
hsip_stat (1M), 1M-24

M

maintenance commands, 1M-5 to 1M-68

N

nf (7), 7-83
nf_fddidaemon (1M), 1M-27
nf_install_agents (1M), 1M-28
nf_macid (1M), 1M-29
nf_smtmon (1M), 1M-30
nf_snmd (1M), 1M-32
nf_snmd_kill (1M), 1M-34
nf_stat (1M), 1M-35
nf_sync (1M), 1M-38

P

pf (7), 7-86
pf_fddidaemon (1M), 1M-39
pf_install_agents (1M), 1M-40
pf_macid (1M), 1M-41
pf_smtmon (1M), 1M-42
pf_snmd (1M), 1M-44
pf_snmd_kill (1M), 1M-46
pf_stat (1M), 1M-47

R

rscadm (1M), 1M-50

S

smt (7), 7-91

sunvts (1M), 1M-54

system administration commands, 1M-5 to 1M-68

U

user commands, 1-1 to 1-4

V

vts_cmd (1M), 1M-55

vtsk (1M), 1M-61

vtsprobe (1M), 1M-62

vtstty (1M), 1M-65

vtsui (1M), 1M-67

vtsui.ol (1M), 1M-68