Platform Notes: Sun™ GigaSwift Ethernet Device Driver

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA, 95054 U.S.A.
650-960-1300

Part No. 816-2351-10
May 2002, Revision A

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

Contents

Preface

These Platform Notes provide instructions for configuring the software used by the Sun™ GigaSwift™ Ethernet device driver. They also contain information for configuring the network.

Related Documentation

Title	Part Number
Solaris 9 Sun Hardware Platform Guide	816-1664-10

Typographic Conventions

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

Shell Prompts

Shell	Prompt
C shell	*machine_name*%
C shell superuser	*machine_name*#
Bourne shell and Korn shell	$
Bourne shell and Korn shell superuser	#

Accessing Sun Documentation Online

The `docs.sun.com`SM web site enables you to access a select group of 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

Sun is interested in improving its documentation and welcomes your comments and suggestions. You can email your comments to Sun at:

`docfeedback@sun.com`

Please include the part number (816-2351-10) of your document in the subject line of your email.

CHAPTER 1

Configuring Driver Parameters

This chapter describes how to configure the driver parameters used by the Sun GigaSwift Ethernet adapter.

This document contains the following sections:

- GigaSwift Ethernet Device Driver Parameters
- Setting `ce` Driver Parameters
- GigaSwift Ethernet Driver Operating Statistics
- Configuring the Network Host Files

Note – The GigaSwift Ethernet driver supports Virtual Local Area Networks (VLANs). The VLAN packages are installed automatically as part of the Solaris software. However, to use VLANs, you must configure them. Refer to the Solaris on Sun Hardware Platform Guide for instructions.

GigaSwift Ethernet Device Driver Parameters

The `ce` device driver controls the GigaSwift Ethernet devices. The `ce` driver is attached to the UNIX `pci` name property `pci108e,abba` for the Sun GigaSwift Ethernet adapter (`108e` is the vendor ID and `abba` is the PCI device ID).

You can manually configure the `ce` device driver parameters to customize each Sun GigaSwift Ethernet adapter device in your system. This section provides an overview of the capabilities of the GigaSwift Ethernet device used in the adapter, lists the available `ce` device driver parameters, and describes how to configure these parameters.

The Sun GigaSwift Ethernet UTP PCI adapter is capable of all the operating speeds and modes listed in “Setting the Auto-negotiation Mode” on page 112. The ce device performs auto-negotiation with the remote end of the link (link partner) to select a common mode of operation. The ce device also supports a forced mode of operation.

Driver Parameter Values and Definitions

The following table describes the parameters and settings for the ce device driver.

TABLE 1-1 ce Driver Parameters, Status, and Descriptions

Parameter	Status	Description
instance	Read and write	Device instance
adv_autoneg_cap	Read and write	Operational mode parameter
adv_1000fdx_cap	Read and write	Operational mode parameter
adv_1000hdx_cap	Read and write	Operational mode parameter
adv_100T4_cap	Read and write	Operational mode parameter
adv_100fdx_cap	Read and write	Operational mode parameter
adv_100hdx_cap	Read and write	Operational mode parameter
adv_10fdx_cap	Read and write	Operational mode parameter
adv_10hdx_cap	Read and write	Operational mode parameter
adv_asmpause_cap	Read and write	Flow control parameter
adv_pause_cap	Read and write	Flow control parameter
link_master	Read and write	1Gb forced mode parameter
use_int_xcvr	Read and write	
enable_ipg0	Read and write	Enable additional delay before transmitting a packet
ipg0	Read and write	Additional delay before transmitting a packet
ipg1	Read and write	Interpacket Gap parameter
ipg2	Read and write	Interpacket Gap parameter
rx_intr_pkts	Read and write	Receive interrupt blanking values
rx_intr_time	Read and write	Receive interrupt blanking values
red_dv4to6k	Read and write	Random early detection and packet drop vectors
red_dv6to8k	Read and write	Random early detection and packet drop vectors

TABLE 1-1 ce Driver Parameters, Status, and Descriptions *(Continued)*

Parameter	Status	Description
red_dv8to10k	Read and write	Random early detection and packet drop vectors
red_dv10to12k	Read and write	Random early detection and packet drop vectors
tx_dma_weight	Read and write	PCI Interface parameter
rx_dma_weight	Read and write	PCI Interface parameter
infinit_burst	Read and write	PCI Interface parameter
disable_64bit	Read and write	PCI Interface parameter

Operational Mode Parameters

The following parameters determine the transmit and receive speed and duplex. The following table describes the operational mode parameters and their default values

TABLE 1-2 Operational Mode Parameters

Parameter	Values and Description
adv_autoneg_cap	Local interface capability advertised by the hardware 0 = Forced mode 1 = Auto-negotiation (default)
adv_1000fdx_cap	Local interface capability advertised by the hardware 0 = Not 1000 Mbit/sec full-duplex capable 1 = 1000 Mbit/sec full-duplex capable (default)
adv_1000hdx_cap	Local interface capability advertised by the hardware 0 = Not 1000 Mbit/sec half-duplex capable 1 = 1000 Mbit/sec half-duplex capable (default)
adv_100fdx_cap	Local interface capability advertised by the hardware 0 = Not 100 Mbit/sec full-duplex capable 1 = 100 Mbit/sec full-duplex capable (default)

TABLE 1-2 Operational Mode Parameters *(Continued)*

Parameter	Values and Description
`adv_100hdx_cap`	Local interface capability advertised by the hardware 0 = Not 100 Mbit/sec half-duplex capable 1 = 100 Mbit/sec half-duplex capable (default)
`adv_10fdx_cap`	Local interface capability advertised by the hardware 0 = Not 10 Mbit/sec full-duplex capable 1 = 10 Mbit/sec full-duplex capable (default)
`adv_10hdx_cap`	Local interface capability advertised by the hardware 0 = Not 10 Mbit/sec half-duplex capable 1 = 10 Mbit/sec half-duplex capable (default)

Note – If a parameter's initial setting is 0, it cannot be changed. If you try to change it, it will revert back to 0.

If all these parameters are set to 1, autonegotiation will use the highest speed possible. If all these parameters are set to 0, you will receive the following error message:

```
NOTICE: Last setting will leave ce1 with no link capabilities.
WARNING: ce1: Restoring previous setting.
```

Flow Control Parameters

The `ce` device is capable of sourcing (transmitting) and terminating (receiving) pause frames conforming to the IEEE 802.3x Frame Based Link Level Flow Control Protocol. In response to received flow control frames, the `ce` device can slow down its transmit rate. On the other hand, the `ce` device is capable of sourcing flow control frames, requesting the link partner to slow down, provided that the link partner supports this feature. By default, the driver advertises both transmit and receive pause capability during autonegotiation.

The following table provides flow control keywords and describes their function.

TABLE 1-3 Read-Write Flow Control Keyword Descriptions

Keyword	Description
adv_asmpause_cap	The adapter supports asymmetric pause, which means it can pause only in one direction. 0=Off (default) 1=On
adv_pause_cap	This parameter has two meanings depending on the value of adv_asmpause_cap. (Default=0) If adv_asmpause_cap = 1 while adv_pause_cap = 1 pauses are received. If adv_asmpause_cap = 1 while adv_pause_cap = 0 pauses are transmitted. If adv_asmpause_cap = 0 while adv_pause_cap = 1 pauses are sent and received. If adv_asmpause_cap = 0 then adv_pause_cap determines whether Pause capability is on or off.

Gigabit Forced Mode Parameter

In 10/100 mode it was possible to disable auto-negotiation and force the link to the speed you required. With Gigabit copper interfaces this feature is also available but may require you to decide whether your side of the connection is a master or a slave. The link_master parameter will facilitate that. Usually switches will be enabled as master therefore this parameter can remain unchanged, since slave operation is the default. If this is not the case Then link_master can be used to enable ce as a master.

TABLE 1-4 Forced Mode Parameter

Parameter	Description
link_master	When set to 1 this enables master operation, assuming the link partner is a slave. When set to 0 this enables slave operation, assuming the link partner is a slave. (default)

Interpacket Gap Parameters

The ce device supports a programmable mode called enable_ipg0.

When a driver receives a packet with `enable_ipg0` set (the default), it adds an additional time delay before transmitting the packet. This delay, set by the `ipg0` parameter, is in addition to the delay set by the `ipg1` and `ipg2` parameters. The additional `ipg0` delay helps to reduce collisions.

If `enable_ipg0` is disabled, the value of `ipg0` is ignored and no additional delay is set. Only the delays set by `ipg1` and `ipg2` will be used. Disable `enable_ipg0` if other systems keep sending a large number of back-to-back packets. Systems that have `enable_ipg0` set might not have enough time on the network.

You can add the additional delay by setting the `ipg0` parameter from 0 to 255, which is the media byte time delay.

The following table defines the `enable_ipg0` and `ipg0` parameters.

TABLE 1-5 Parameters Defining `enable_ipg0` and `ipg0`

Parameter	Values	Description
`enable_ipg0`	0	`enable_ipg0` reset
	1	`enable_ipg0` set (Default=8)
`ipg0`	0 to 255	The additional time delay (or gap) before transmitting a packet (after receiving the packet) (Default=8)

The `ce` device supports the programmable Interpacket Gap (IPG) parameters `ipg1` and `ipg2`. The total IPG is the sum of `ipg1` and `ipg2`. The total IPG is 0.096 microseconds for the link speed of 1000 Mbps.

The following table lists the default values and allowable values for the IPG parameters.

TABLE 1-6 Read-Write Interpacket Gap Parameter Values and Descriptions

Parameter	Values (Byte-time)	Description
`ipg1`	0 to 255	Interpacket gap 1 (Default = 8)
`ipg2`	0 to 255	Interpacket gap 2 (Default = 4)

By default, the driver sets `ipg1` to 8-byte time and `ipg2` to 4-byte time, which are the standard values. (Byte time is the time it takes to transmit one byte on the link, with a link speed of 1000 Mbps.)

If your network has systems that use longer IPG (the sum of `ipg1` and `ipg2`), and if those machines seem to be slow in accessing the network, increase the values of `ipg1` and `ipg2` to match the longer IPGs of other machines.

Interrupt Parameters

The following table describes the receive interrupt blanking values.

TABLE 1-7 RX Blanking Register for Alias Read

Field Name	Values	Description
rx_intr_pkts	0 to 511	Interrupt after this number of packets have arrived since the last packet was serviced. A value of zero indicates no packet blanking. (Default=3)
rx_intr_time	0 to 524287	Interrupt after 4.5 microsecond ticks have elapsed since the last packet was serviced. A value of zero indicates no time blanking. (Default=1250)

Random Early Drop Parameters

The following table describes the RX random early detection 8-bit vectors, which allows you to enable random early drop (RED) thresholds. When received packets reach the RED range packets are dropped according to the preset probability. The probability should increase when the fifo level increases. Control packets are never dropped and are not counted in the statistics.

TABLE 1-8 RX Random Early Detecting 8-Bit Vectors

Field Name	Values	Description
red_dv4to6k	0 to 255	Random early detection and packet drop vectors for when fifo threshold is greater than 4096 bytes and less than 6,144 bytes. Probability of drop can be programmed on a 12.5 percent granularity. For example, if bit 0 is set the first packet out of every eight will be dropped in this region. (Default=0)

TABLE 1-8 RX Random Early Detecting 8-Bit Vectors

Field Name	Values	Description
red_dv6to8k	0 to 255	Random early detection and packet drop vectors for when fifo threshold is greater than 6,144 bytes and less than 8,192 bytes. Probability of drop can be programmed on a 12.5 percent granularity. For example, if bit 8 is set the first packet out of every eight will be dropped in this region. (Default=0)
red_dv8to10k	0 to 255	Random early detection and packet drop vectors for when fifo threshold is greater than 8,192 bytes and less than 10,240 bytes. Probability of drop can be programmed on a 12.5 percent granularity. For example, if bit 16 is set the first packet out of every eight will be dropped in this region. (Default=0)
red_dv10to12k	0 to 255	Random early detection and packet drop vectors for when fifo threshold is greater than 10,240 bytes and less than 12,288 bytes. Probability of drop can be programmed on a 12.5 percent granularity. For example, if bit 24 is set the first packet out of every eight will be dropped in this region. (Default=0)

PCI Bus Interface Parameters

These parameters allow you to modify PCI interface features to gain better PCI interperformance for a given application.

TABLE 1-9 PCI Bus Interface Parameters

Parameter	Description
`tx_dma_weight`	Determine the multiplication factor for granting credit to the TX side during a weighted round robin arbitration. Values are 0 to 3. (Default=0) Zero means no extra weighting. The other values are power of 2 extra weighting, on that traffic. For example of tx_dma_weight = 0 and rx_dma_weight = 3 then as long as RX traffic is continuously arriving its priority will be 8 times greater than TX to access the PCI

TABLE 1-9 PCI Bus Interface Parameters

Parameter	Description
`rx_dma_weight`	Determine the multiplication factor for granting credit to the RX side during a weighted round robin arbitration. Values are 0 to 3. (Default=0)
`infinite_burst`	allows the infinite burst capability to be utilized. When this is in effect and the system supports infinite burst. The adapter will not free the bus until complete packets are transferred across the bus. Values are 0 or 1. (Default=0)
`disable_64bit`	Switches off 64 bit capability of the adapter. In some cases, it is useful to switch off this feature. Values are 0 or 1. (Default=0, which enables 64 bit capability)

Setting `ce` Driver Parameters

You can set the `ce` device driver parameters in two ways:

- Using the `ndd` utility
- Using the `ce.conf` file

If you use the `ndd` utility, the parameters are valid only until you reboot the system. This method is good for testing parameter settings.

To set parameters so they remain in effect after you reboot the system, create a `/platform/sun4u/kernel/drv/ce.conf` file and add parameter values to this file when you need to set a particular parameter for a device in the system.

Setting Parameters Using the `ndd` Utility

Use the `ndd` utility to configure parameters that are valid until you reboot the system. The `ndd` utility supports any networking driver, which implements the Data Link Provider Interface (DLPI).

The following sections describe how you can use the `ce` driver and the `ndd` utility to modify (with the `-set` option) or display (without the `-set` option) the parameters for each `ce` device.

▼ To Specify Device Instances for the `ndd` Utility

Before you use the `ndd` utility to get or set a parameter for a `ce` device, you must specify the device instance for the utility.

1. **Check the** `/etc/path_to_inst` **file to identify the instance associated with a particular device.**

```
# grep ge /etc/path_to_inst
"/pci@1f,2000/pci@1/network@0" 2 "ce"
"/pci@1f,2000/pci@2/network@0" 1 "ce"
"/pci@1f,2000/pci@4/network@0" 0 "ce"
```

In the example above, the three GigaSwift Ethernet instances are from the installed adapters. The instance numbers are in bold italics for clarity.

2. **Use the instance number to select the device.**

```
# ndd -set /dev/ce instance instance#
```

The device remains selected until you change the selection.

Noninteractive and Interactive Modes

You can use the ndd utility in two modes:

- Noninteractive
- Interactive

In noninteractive mode, you invoke the utility to execute a specific command. Once the command is executed, you exit the utility. In interactive mode, you can use the utility to get or set more than one parameter value. (Refer to the ndd(1M) man page for more information.)

Using the ndd *Utility in Noninteractive Mode*

This section describes how to modify and display parameter values.

1. **To modify a parameter value, use the** `-set` **option.**

   If you invoke the `ndd` utility with the `-set` option, the utility passes *value*, which must be specified, down to the named `/dev/ce` driver instance, and assigns it to the parameter:

```
# ndd -set /dev/ce parameter value
```

   When you change any adv parameter, a message similar to the following appears:

```
xcvr addr:0x00 - link up 1000 Mbps half duplex
```

2. **To display the value of a parameter, specify the parameter name and omit the value.**

   When you omit the `-set` option, a query operation is assumed and the utility queries the named driver instance, retrieves the value associated with the specified parameter, and prints it:

```
# ndd /dev/ce parameter
```

Using the `ndd` *Utility in Interactive Mode*

1. **To modify a parameter value in interactive mode, specify** `ndd /dev/ce`**, as shown below.**

   The `ndd` utility then prompts you for the name of the parameter:

```
# ndd /dev/ce
name to get/set? (Enter the parameter name or ? to view all
parameters)
```

   After typing the parameter name, the `ndd` utility prompts you for the parameter value (see TABLE 1-1through TABLE 1-11).

2. **To list all the parameters supported by the** `ce` **driver, type** `ndd /dev/ce`**.**

```
# ndd /dev/ce
name to get/set ? ?
?                               (read only)
instance                        (read and write)
adv_autoneg_cap                 (read and write)
adv_1000fdx_cap                 (read and write)
adv_1000hdx_cap                 (read and write)
adv_100T4_cap                   (read and write)
adv_100fdx_cap                  (read and write)
adv_100hdx_cap                  (read and write)
adv_10fdx_cap                   (read and write)
adv_10hdx_cap                   (read and write)
adv_asmpause_cap                (read and write)
adv_pause_cap                   (read and write)
link_master                     (read and write)
use_int_xcvr                    (read and write)
enable_ipg0                     (read and write)
ipg0                            (read and write)
ipg1                            (read and write)
ipg2                            (read and write)
rx_intr_pkts                    (read and write)
rx_intr_time                    (read and write)
red_dv4to6k                     (read and write)
red_dv6to8k                     (read and write)
red_dv8to10k                    (read and write)
red_dv10to12k                   (read and write)
tx_dma_weight                   (read and write)
rx_dma_weight                   (read and write)
infinite_burst                  (read and write)
disable_64bit                   (read and write)
name to get/set ?
#
```

Setting the Auto-negotiation Mode

By default, autonegotiation is set to `on`. This means that the adapter communicates with its link partner to determine a compatible network speed, duplex mode, and flow control capability.

▼ To Disable Auto-negotiation Mode

If your network equipment does not support autonegotiation, or if you want to specify your network speed, you can set autonegotiation to `off` on the `ce` device.

1. **Set the following driver parameters to the values that are described in the documentation that shipped with your link partner (for example, a switch):**
   - adv_1000fdx_cap
   - adv_1000hdx_cap
   - adv_100fdx_cap
   - adv_100hdx_cap
   - adv_10fdx_cap
   - adv_10hdx_cap
   - adv_asmpause_cap
   - adv_pause_cap

   See TABLE 1-2 for the descriptions and possible values of these parameters.

2. **Set the** adv_autoneg_cap **parameter to** 0**.**

```
# ndd -set /dev/ce adv_autoneg_cap 0
```

When you change any ndd link parameter, a message similar to the following appears:

```
xcvr addr:0x00 - link up 1000 Mbps half duplex
```

Setting Parameters Using the ce.conf File

You can also specify the driver parameter properties on a per-device basis by creating a ce.conf file in the /platform/sun4u/kernel/drv directory. Use a ce.conf file when you need to set a particular parameter for a device in the system. The parameters you set are read and write parameters that are listed in “Driver Parameter Values and Definitions” on page 12.

The man pages for prtconf(1M) and driver.conf(4) include additional details. The next procedure shows an example of setting parameters in a ce.conf file.

▼ To Set Driver Parameters Using a `ce.conf` File

1. **Obtain the hardware path names for the `ce` devices in the device tree.**

   Typically, the path names and the associated instance numbers are in the `/etc/path_to_inst` file.

```
# grep ce /etc/path_to_inst
"/pci108e;abba;/pci@4,4000/network@0" 2 "ce"
"/pci108e;abba;/pci@6,2000/network@0" 1 "ce"
"/pci108e;abba;/pci@4,2000/network@0" 0 "ce"
```

   - In the previous example:
     - The first part within the double quotes specifies the hardware node name in the device tree.
     - The second number is the instance number (shown in bold italics).
     - The last part in double quotes is the driver name.
   - In the device path name, the last component after the last `/` character and before the `@` character is the device name.
   - The path name before the last component is the parent name.
   - The comma separated numbers after the `@` character represent the device and function numbers, which are together referred to as unit-address.

   To identify a PCI device unambiguously in the `ce.conf` file, use the name, parent name, and the unit-address for the device. Refer to the `pci`(4) man page for more information about the PCI device specification.

   In the first line of the previous example:

   - Name = `pci108e,abba`
   - Parent name = `/pci@4,4000`
   - Unit-address = `4,4`

   In the second line in the previous example:

   - Name = `pci108e,abba`
   - Parent name = `/pci@6,2000`
   - Unit-address = 6,2

   In the third line in the previous example:

   - Name = `pci108e,abba`
   - Parent name = `/pci@4,2000`
   - Unit-address = `4,2`

2. **Set the parameters for the above devices in the** `/platform/sun4u/kernel/drv/ce.conf` **file.**

   In the following example, the `adv_autoneg_cap` and `adv_1000fdx_cap` parameters are set for all Sun GigaSwift Ethernet devices. (See the `driver.conf(4)` man page for more information.)

```
adv_autoneg_cap=0 adv_1000fdx_cap=0
```

   In the following example, the `adv_autoneg_cap` and `adv_1000fdx_cap` parameters are set for a single instance of the Sun GigaSwift Ethernet device.

```
name=pci108e,abba parent=pci@4,4000 unit address+4 adv_autoneg_cap=0
adv_1000fdx_cap=0;
```

3. **Save the** `ce.conf` **file.**
4. **Save and close all files and programs, and exit the windowing system.**
5. **Shut down and reboot the system.**

GigaSwift Ethernet Driver Operating Statistics

These statistics are part of the statistics presented by the `netstat -k` command.

The following table describes the read-only Media Independent Interface (MII) capabilities. These parameters define the capabilities of the hardware. The Gigabit Media Independent Interface (GMII) supports all of the following capabilities.

TABLE 1-10 Read-Only `ce` Device Capabilities

Parameter	Description (Local Interface Capabilities)
`cap_autoneg`	0 = Not capable of autonegotiation 1 = Auto-negotiation capable
`cap_1000fdx`	Local interface full-duplex capability 0 = Not 1000 Mbit/sec full-duplex capable 1 = 1000 Mbit/sec full-duplex capable
`cap_1000hdx`	Local interface half-duplex capability 0 = Not 1000 Mbit/sec half-duplex capable 1 = 1000 Mbit/sec half-duplex capable
`cap_100fdx`	Local interface full-duplex capability 0 = Not 100 Mbit/sec full-duplex capable 1 = 100 Mbit/sec full-duplex capable
`cap_100hdx`	Local interface half-duplex capability 0 = Not 100 Mbit/sec half-duplex capable 1 = 100 Mbit/sec half-duplex capable
`cap_10fdx`	Local interface full-duplex capability 0 = Not 10 Mbit/sec full-duplex capable 1 = 10 Mbit/sec full-duplex capable
`cap_10hdx`	Local interface half-duplex capability 0 = Not 10 Mbit/sec half-duplex capable 1 = 10 Mbit/sec half-duplex capable
`cap_asm_pause`	Local interface flow control capability 0 = Not asymmetric pause capable 1 = Asymmetric pause (from the local device) capable
`cap_pause`	Local interface flow control capability 0 = Not Symmetric pause capable 1 = Symmetric pause capable

Reporting the Link Partner Capabilities

The following table describes the read-only link partner capabilities.

TABLE 1-11 Read-Only Link Partner Capabilities

Parameter	Values and Description
`lp_cap_autoneg`	0 = No autonegotiation 1 = Auto-negotiation
`lp_cap_1000fdx`	0 = No 1000 Mbit/sec full-duplex transmission 1 = 1000 Mbit/sec full-duplex
`lp_cap_1000hdx`	0 = No 1000 Mbit/sec half-duplex transmission 1 = 1000 Mbit/sec half-duplex
`lp_cap_100fdx`	0 = No 100 Mbit/sec full-duplex transmission 1 = 100 Mbit/sec full-duplex
`lp_cap_100hdx`	0 = No 100 Mbit/sec half-duplex transmission 1 = 1000 Mbit/sec half-duplex
`lp_cap_10fdx`	0 = No 10 Mbit/sec full-duplex transmission 1 = 10 Mbit/sec full-duplex
`lp_cap_10hdx`	0 = No 10 Mbit/sec half-duplex transmission 1 = 10 Mbit/sec half-duplex
`lp_cap_asm_pause`	0 = Not asymmetric pause capable 1 = Asymmetric pause towards link partner capability
`lp_cap_pause`	0 = Not symmetric pause capable 1 = Symmetric pause capable

If the link partner is *not* capable of autonegotiation (when `lp_cap_autoneg` is 0), the remaining information described in the previous table is not relevant and the parameter value = 0.

If the link partner *is* capable of autonegotiation (when `lp_cap_autoneg` is 1), then the speed and mode information is displayed when you use autonegotiation and the link partner capabilities.

The following table describes the netstat -k transmit and receive parameters:

TABLE 1-12 Transmit and Receive Parameters

Parameter	Description
xcvr_inits	Number of Physical layer re-initializations every time you change link parameters using NDD this increments.
rev_id	Revision ID of the GigaSwift Ethernet device useful for recognition of device being used in the field.
xcvr_addr	GMII/MII Physical layer device address for management interface.
xcvr_id	GMII/MII Physical layer device Identification Decimal copy of MII registers 2 and 3.
lb_mode	Copy of the Loopback mode the device is in, if any.
qos_mode	When zero, the TX queues operate in a simple round robin queueing scheme, based on TCP/UDP destination port number. If set the TX queues operate in a scheme designed to provide VLAN priorities.
tx_starts	Number of times that the driver attempted to transmit a packet.
tx_dma_bind_fail	Number of times a page table entry was not available to allow the driver to map the kernel memory to device accessible memory for transmission.
tx_queue0	Number of packets queued for transmission on the first hardware transmit queue.
tx_queue1	Number of packets queued for transmission on the second hardware transmit queue.
tx_queue2	Number of packets queued for Transmission on the third hardware transmit queue.
tx_queue3	Number of packets queued for Transmission on the fourth hardware transmit queue.
tx_max_pend	Maximum number of transmits pending on any of the four queues.
rx_hdr_pkts	Number of packets received that were less than 256 bytes.
rx_mtu_pkts	Number of packets received that were greater than 256 bytes and less than 1514 bytes.
rx_split_pkts	Number of packets that were split across two pages.
rx_no_comp_wb	Number of times the hardware cannot post completion entries for received data.
rx_no_buf	Number of times the hardware cannot receive data because there is no more receive buffer space.
rx_new_pages	Number of pages that got replaced during reception.

TABLE 1-12 Transmit and Receive Parameters

Parameter	Description
rx_new_hdr_pgs	Number of pages that were filled with packets less than 256 bytes that got replaced during reception.
rx_new_mtu_pgs	Number of pages that were filled with packets greater than 256 bytes and less than 1514 that got replaced during reception.
rx_new_nxt_pgs	Number of pages that contained packets that were split across pages that got replaced during reception.
rx_hdr_drops	Number of times a whole page of packets less than 256 bytes was dropped because the driver was unable to map a new one to replace it.
rx_mtu_drops	Number of times a whole page of packets greater than 256 bytes and less than 1514 was dropped because the driver was unable to map a new one to replace it.
rx_nxt_drops	Number of times a page with a split packet was dropped because the driver was unable to map a new one to replace it.
rx_rel_flow	Number of times the driver was told to release a flow.

▼ To Check Link Partner Settings

1. **As superuser, type the** `netstat -k` **command:**

```
# netstat -k ce0
ce0:
ipackets 0 ipackets64 0 ierrors 0 opackets 0 opackets64 0
oerrors 0 collisions 0 rbytes 0 rbytes64 0 obytes 0 obytes64 0
multircv 0 multixmt 0 brdcstrcv 0 brdcstxmt 0 norcvbuf 0
noxmtbuf 0 first_collision 0 excessive_collisions 0 late_collisions 0
peak_attempts 0 length_err 0 alignment_err 0 crc_err 0 code_violations 0
ifspeed 0 rev_id 1 xcvr_inits 1 xcvr_inuse 3 xcvr_addr 0
xcvr_id 0 cap_autoneg 1 cap_1000fdx 1 cap_1000hdx 0 cap_100T4 0
cap_100fdx 0 cap_100hdx 0 cap_10fdx 0 cap_10hdx 0 cap_asmpause 0
cap_pause 1 lp_cap_autoneg 0 lp_cap_1000fdx 0 lp_cap_1000hdx 0
lp_cap_100T4 0 lp_cap_100fdx 0 lp_cap_100hdx 0 lp_cap_10fdx 0
lp_cap_10hdx 0 lp_cap_asmpause 0 lp_cap_pause 0 link_T4 0
link_speed 0 link_duplex 0 link_asmpause 0 link_pause 0
link_up 0 lb_mode 0 qos_mode 0 tx_inits 0 tx_starts 0 tx_nocanput 0
tx_msgdup_fail 0 tx_allocb_fail 0 tx_no_desc 0 tx_dma_bind_fail 0
tx_uflo 0 tx_queue0 0 tx_queue1 0 tx_queue2 0 tx_queue3 0
tx_max_pend 0 rx_inits 0 rx_hdr_pkts 0 rx_mtu_pkts 0 rx_split_pkts 0
rx_no_buf 0 rx_no_comp_wb 0 rx_ov_flow 0 rx_len_mm 0 rx_bad_descs 0
rx_nocanput 0 rx_msgdup_fail 0 rx_allocb_fail 0 rx_new_pages 0
rx_new_hdr_pgs 0 rx_new_mtu_pgs 0 rx_new_nxt_pgs 0 rx_hdr_drops 0
rx_mtu_drops 0 rx_nxt_drops 0 rx_rel_flow 0 rx_pkts_dropped 0
pci_err 0 pci_rta_err 0 pci_rma_err 0 pci_parity_err 0 pci_bad_ack_err 0
pci_drto_err 0 ipackets_cpu00 0 ipackets_cpu01 0 ipackets_cpu02 0
ipackets_cpu03 0
```

Configuring the Network Host Files

After installing the driver software, you must create a `hostname.ce`*number* file for the adapter's Ethernet interface. You must also create both an IP address and a host name for its Ethernet interface in the `/etc/hosts` file.

1. **At the command line, use the** `grep` **command to search the** `/etc/path_to_inst` **file for** `ce` **interfaces.**

```
# grep ce /etc/path_to_inst
"/pci@1f,4000/pci@1/network@4" 0 "ce"
```

In the example above, the device instance is from a Sun GigaSwift Ethernet adapter installed in slot 1. For clarity, the instance number is in bold italics.

2. **Use the** `ifconfig` **command to setup the adapter's** `ce` **interface.**

Use the `ifconfig` command to assign an IP address to the network interface. Type the following at the command line, replacing *ip_address* with the adapter's IP address:

```
# ifconfig ce0 plumb ip_address up
```

Refer to the `ifconfig(1M)` man page and the Solaris documentation for more information.

- If you want a set-up that will remain the same after you reboot, create an `/etc/hostname.ce`*number* file, where *number* corresponds to the instance number of the `ce` interface you plan to use.

To use the adapter's `ce` interface in the Step 1 example, create an `/etc/hostname.ce0` file, where `0` is the number of the `ce` interface. If the instance number were *1*, the filename would be `/etc/hostname.ce1`.

- Do not create an `/etc/hostname.ce`*number* file for a Sun GigaSwift Ethernet adapter interface you plan to leave unused.
- The `/etc/hostname.ce`*number* file must contain the hostname for the appropriate `ce` interface.
- The host name should have an IP address and should be listed in the `/etc/hosts` file.
- The host name should be different from any other host name of any other interface, for example: `/etc/hostname.ce0` and `/etc/hostname.ce1` cannot share the same host name.

The following example shows the `/etc/hostname.ce`*number* file required for a system called `zardoz` that has a Sun GigaSwift Ethernet adapter (`zardoz-11`).

```
# cat /etc/hostname.hme0
zardoz
# cat /etc/hostname.ce0
zardoz-11
```

3. **Create an appropriate entry in the `/etc/hosts` file for each active `ce` interface.**

   For example:

```
# cat /etc/hosts
#
# Internet host table
#
127.0.0.1     localhost
129.144.10.57 zardoz    loghost
129.144.11.83 zardoz-11
```