Platform Notes: The eri FastEthernet Device Driver

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

Part No. 806-5579-10
October 2000, Revision A

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

Please
Recycle

Contents

Tables

Preface

This book describes how to configure the `eri` driver for Sun systems using the Ethernet function of the RIO Application Specific Integrated Chip (ASIC).

How This Book Is Organized

Chapter 1 describes the `eri` device driver and includes topics such as operating speeds and modes, and auto-negotiation.

Chapter 2 describes configuring the `eri` device driver.

Chapter 3 describes the parameters and settings for the `eri` device driver.

Chapter 4 describes how to set the `eri` device driver parameter values using the `ndd` utility and also in the `/etc/system` and `/kernel/drv/eri.conf` files.

Using UNIX Commands

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

See one or more of the following for this information:

- AnswerBook2™ online documentation for the Solaris™ software environment
- Other software documentation that you received with your system

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	#

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 806-5579-10) of your document in the subject line of your email.

CHAPTER 1

The eri Device Driver

The eri device driver handles the SUNW,eri device on Sun systems using the RIO ASIC.

This chapter describes the following:

A hardware overview of the SUNW,eri device

Information on the operating speeds and modes for the SUNW,eri device

Auto-negotiation

The internal transceiver for the eri device driver

Hardware Overview

The SUNW,eri device provides a 100BASE-TX network interface using the Ethernet function of the RIO ASIC. The driver automatically sets the link speed to 10 or 100 Mbps and conforms to the *100BASE-T IEEE 802.3u Ethernet Standard*. The RIO ASIC provides the PCI interface and Media Access Control (MAC) functions. The internal transceiver, which connects to an RJ-45 connector, provides the physical layer functions.

The RIO ASIC is a chip set composed of an I/O chip and a single chip Ethernet transceiver; the eri device driver uses the Ethernet function of this ASIC.

Operating Speeds and Modes

You can operate the link in any of the following speeds and modes with the SUNW, eri device:

- 100 Mbps, full-duplex
- 100 Mbps, half-duplex
- 10 Mbps, full-duplex
- 10 Mbps, half-duplex

The *100BASE-T IEEE 802.3u Ethernet Standard* describes these speeds and modes.

Auto-Negotiation

A key feature of the Sun eri FastEthernet driver is auto-negotiation. The auto-negotiation protocol, as specified by the *100BASE-T IEEE 802.3u Ethernet Standard*, selects the operation mode (half-duplex or full-duplex), and the auto-sensing protocol selects the speed (10 Mbps or 100 Mbps) for the adapter.

The auto-negotiation protocol does the following when the system is booted:

- Identifies all link partner-supported modes of operation
- Advertises its capabilities to the link partner
- Selects the highest common denominator mode of operation based on the following priorities (in decreasing order):
  - 100 Mbps, full-duplex
  - 100 Mbps, half-duplex
  - 10 Mbps, full-duplex
  - 10 Mbps, half-duplex

The link partner is the networking device (system, Ethernet hub, or Ethernet switch) at the other end of the link or cable.

If the SUNW,eri device is connected to a remote system or interface that is not capable of auto-negotiation, your system automatically selects the correct speed and half-duplex mode.

If the Sun eri FastEthernet is connected to a link partner with which the auto-negotiation protocol fails to operate successfully, you can configure the device so it does not use this protocol. This forces the driver to set up the link in the mode and speed of your choice.

Internal (Local) Transceiver

The internal transceiver is a feature supported by the driver and is capable of all the operating speeds and modes listed in the section “Operating Speeds and Modes” earlier in this chapter. The driver automatically sets the link speed to 10 or 100 Mbs, and conforms to the *100BASE-T IEEE 802.3u Ethernet Standard*.

The internal transceiver also supports a forced mode of operation. This is where the user selects the speed and mode using the `ndd` utility, by editing the `/etc/system` file, or by creating an `eri.conf` file in the `kernal/drv/` directory. The `ndd` utility makes calls to the `eri` driver to choose the speed and mode.

CHAPTER 2

Configuring the Driver Software for Sun `eri` FastEthernet Device Drivers

This chapter includes information and instructions for configuring the driver software used by the Sun `eri` FastEthernet PCI adapter.

This chapter includes the following sections:

Installing the Driver Software

The 64-bit driver is included with the Solaris CD.

Before using `eri` as your network interface, you will need to create and edit system host files, as described in the next section.

Configuring the Host File

▼ To Configure the Host File

1. **At the command line, use the** `grep` **command to search the** `/etc/path_to_inst` **file for** `eri` **devices. For example:**

```
# grep eri /etc/path_to_inst
“/pci@8,700000/network@5,1” 0 “eri”
```

2. **Create an** `/etc/hostname.eri`*num* **file, where** *num* **is the instance number of each interface you plan to use.**

   If you want to use the network interface from the example in Step 1, you will need to create a file:

File Name	Instance Number
`/etc/hostname.eri0`	0

   - Do not create `/etc/hostname.eri`*num* files for Sun `eri` FastEthernet network interfaces you plan to leave unused.
   - The `/etc/hostname.eri`*num* file must contain the host name for the appropriate network interface.
   - The host name should have an IP address that will need to be entered in the `/etc/hosts` file.
   - The host name should be different from any other host name of any other interface, for example: `/etc/hostname.hme0` and `/etc/hostname.eri0` cannot share the same host name.

   Using the instance examples in Step 1, the following example shows the two `/etc/hostname.eri`*num* files required for a system called `zardoz` that has a Sun `eri` FastEthernet (`zardoz, zardoz-11`).

```
# cat /etc/hostname.hme0
zardoz
# cat /etc/hostname.eri0
zardoz-11
```

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

   Using the previous example, you will have:

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

Note – The Internet Protocol, version 6 (IPv6), expands the capabilities of IPv4, which is the current version and the default. The Sun `eri` FastEthernet device driver included in this release of the Solaris operating environment supports both IPv4 and IPv6. IPv4 uses the `/etc/hosts` configuration file, but IPv6 uses a different configuration file. To transition to, manage, and implement IPv6, refer to the Solaris 8 System Administration Guide, Volume 3.

---

4. **Reboot your system.**

Booting From the Network

To use a Sun eri interface as the boot device, perform the following tasks:

▼ To Boot From the Network

1. **At the `ok` prompt type:**

```
ok show-nets
```

   The `show-nets` command lists the system devices. You should see the full path name of the `eri` devices, similar to the following examples:

```
/pci@8,700000/network@5,1
```

Note – You need to select only one of these `eri` devices for booting.

2. **At the** `ok` **prompt type:**

```
ok boot full_path_name_of_the_eri_device
```

Optional Post-Installation Procedures

To customize the performance of the Sun `eri` FastEthernet driver, perform the tasks in the following sections.

Setting Driver Parameters

The `eri` device driver, which is loaded from the Solaris CD-ROM, controls the `SUNW,eri` Ethernet devices. The device driver selects the link speed using the auto-negotiation protocol with the link partner. (See "Auto-Negotiation" on page 2.)

You can manually set the `eri` device driver parameters to customize each `SUNW,eri` device in your system in one of three ways:

- Set a parameter on a per-device basis by creating the `eri.conf` file in the `/kernel/drv` directory.
- Use the `ndd` utility to *temporarily* change a parameter. This change is lost when you reboot the system.
- Set the `eri` driver parameters generally for all `SUNW,eri` devices in the system by entering the parameter variables in the `/etc/system` file.

See Chapter 4 "Setting Parameters" for more information.

▼ To Force Network Speed Between 10 Mbps and 100 Mbps

1. **At the** `ok` **prompt, use the** `show-devs` **command to list the system devices.**

   You should see the full path names of the `eri` devices, similar to the following example:

```
/pci@8,700000/network@5,1
```

2. **Type**:

```
ok nvedit
```

3. **Type the following, pressing the Return key at the end of line** `0`:

```
0: probe-all install-console banner
1: apply transfer-speed=10 full_ path_name_of_a_eri_device
```

Note – If you already have commands in NVRAM, append these lines to the end of the file.

---

4. **Press Control-C after typing** *full_ path_name_of_a_*`eri`*_device.*

   Perform Steps 2 to 4 to set the network speed for each `eri` network interface.

Note – In the preceding example, the speed is forced to 10 Mbps. To force the speed to 100 Mbps, replace 10 with 100.

---

5. **At the** `ok` **prompt type**:

```
ok nvstore
ok setenv use-nvramrc? true
```

6. **Reboot your system.**

   See “Setting Forced Mode” on page 22 for more information on forcing network speed.

CHAPTER 3

Parameter Definitions

This chapter describes the parameters and settings for the `eri` device driver.

Driver Parameter Values and Definitions

The following sections describe the `eri` driver parameters, which are listed in TABLE 3-1.

TABLE 3-1 `eri` Driver Parameter, Status, and Descriptions

Parameter	Status	Description
`transceiver_inuse`	Read only	Defines the current status
`link_status`	Read only	Defines the current status
`link_speed`	Read only	Defines the current status
`link_mode`	Read only	Defines the current status
`ipg1`	Read and write	Inter-packet gap parameter
`ipg2`	Read and write	Inter-packet gap parameter
`pace_size`	Read and write	Operational mode parameter
`adv_autoneg_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

TABLE 3-1 `eri` Driver Parameter, Status, and Descriptions *(Continued)*

Parameter	Status	Description
`autoneg_cap`	Read only	Local transceiver auto negotiation capability
`100fdx_cap`	Read only	Local transceiver capability of the hardware
`100hdx_cap`	Read only	Local transceiver capability of the hardware
`10fdx_cap`	Read only	Local transceiver capability of the hardware
`10hdx_cap`	Read only	Local transceiver capability of the hardware
`lp_autoneg_cap`	Read only	Link partner auto negotiation capability
`lp_100fdx_cap`	Read only	Link partner capability
`lp_100hdx_cap`	Read only	Link partner capability
`lp_10fdx_cap`	Read only	Link partner capability
`lp_10hdx_cap`	Read only	Link partner capability
`instance`	Read and write	Device instance
`lance_mode`	Read and write	Additional delay before transmitting a packet
`ipg0`	Read and write	Additional delay before transmitting a packet

Defining the Current Status

The read-only parameters described in TABLE 3-2 explain the operational mode of the interface. These parameters define the current status.

TABLE 3-2 Read-Only Parameters for Defining the Current Status

Parameter	Description	Values
`link_status`	Current link status	0 = Link down 1 = Link up
`link_speed`	Valid only if the link is up	0 = 10 Mbps 1 = 100 Mbps
`link_mode`	Valid only if the link is up	0 = Half duplex 1 = Full duplex

Inter-Packet Gap Parameters

The Ethernet function unit of RIO ASIC supports programmable Inter-Packet Gap (IPG) parameters `ipg1` and `ipg2`. The total IPG is the sum of `ipg1` and `ipg2`. The total IPG is 9.6 microseconds when the link speed set by the auto-negotiation protocol is 10 Mbps. When the link speed is 100 Mbps, the total IPG is 0.96 microseconds.

TABLE 3-3 lists the default values and allowable values for the IPG parameters, `ipg1` and `ipg2`.

TABLE 3-3 Read-Write Inter-Packet Gap Parameter Values and Descriptions

Parameter	Values (Byte-time)	Description
`ipg1`	0, 255	`ipg1` = 8 (default at initialization)
`ipg2`	0, 255	`ipg2` = 4 (default at initialization)

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 either 100 Mbps or 10 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.

Defining an Additional Delay Before Transmitting a Packet Using lance_mode and ipg0

The ethernet function unit of RIO ASIC supports a programmable mode called lance_mode. The ipg0 parameter is associated with lance_mode.

After a packet is received with lance_mode enabled (default), an additional delay is added by setting the ipg0 parameter 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 delay set by ipg0 helps to reduce collisions. Systems that have lance_mode enabled might not have enough time on the network.

If lance_mode is disabled, the value of ipg0 is ignored and no additional delay is set. Only the delays set by ipg1 and ipg2 are used. Disable lance_mode if other systems usually send a large number of back-to-back packets.

You can enable the additional delay by setting the ipg0 parameter from 0 to 31, which is the nibble time delay. Nibble time is the time it takes to transfer four bits on the link. If the link speed is 10 Mbps, nibble time is equal to 400 ns. If the link speed is 100 Mbps, nibble time is equal to 40 ns.

For example, if the link speed is 10 Mbps, and you set ipg0 to 20 nibble times, multiply 20 by 400 ns to get 8000 ns. If the link speed is 100 Mbps, and you set ipg0 to 30 nibble times, multiply 30 by 40 ns to get 1200 ns.

TABLE 3-4 defines the lance_mode and ipg0 parameters.

TABLE 3-4 Parameters Defining lance_mode and ipg0

Parameter	Values	Description
lance_mode	0 1	lance_mode disabled lance_mode enabled (default)
ipg0	0-31[1]	Additional IPG before transmitting a packet (after receiving a packet)

1. The default value is 16 nibble-times, which is 6.4 microseconds for 10 Mbps and 0.64 microseconds for 100 Mbps

Operational Mode Parameters

TABLE 3-5 describes the operational mode parameters and their default values.

TABLE 3-5 Operational Mode Parameters

Parameter	Description	Values
adv_autoneg_cap	Local transceiver capability advertised by the hardware	0 = Forced mode 1 = Auto-negotiation (default)
adv_100fdx_cap[1]	Local transceiver capability advertised by the hardware; read/write parameter	0 = Not 100 Mbit/sec full-duplex capable 1 = 100 Mbit/sec full-duplex capable (default)
adv_100hdx_cap[1]	Local transceiver capability advertised by the hardware; read/write parameter	0 = Not 100 Mbit/sec half-duplex capable 1 = 100 Mbit/sec half-duplex capable (default)
adv_10fdx_cap[1]	Local transceiver capability advertised by the hardware; read/write parameter	0 = Not 10 Mbit/sec full-duplex capable 1 = 10 Mbit/sec full-duplex capable (default)
adv_10hdx_cap[1]	Local transceiver capability advertised by the hardware; read/write parameter	0 = Not 10 Mbit/sec half-duplex capable 1 = 10 Mbit/sec half-duplex capable (default)

1. The priority (in descending order) for these parameters is: adv_100fdx_cap, adv_100hdx_cap, adv_10fdx_cap, adv_10hdx_cap.

Defining the Number of Back-to-Back Packets to Transmit

The `pace_size` parameter (see TABLE 3-6) defines the maximum number of back-to-back packets you can transmit at one time. If the value is zero, there is no limit to the number of back-to-back packets that can be transmitted.

TABLE 3-6 Back-to-Back Packet Transmission Capability

Parameter	Values	Description
`pace_size`	1-255	Number of back-to-back packets transmitted at one time
	0	No limit to the number of back-to-back packets that can be transmitted (default)

Reporting Transceiver Capabilities

TABLE 3-7 describes the read-only transceiver capabilities. These parameters define the capabilities of the hardware. The local transceiver can support all of these capabilities.

TABLE 3-7 Read-Only Transceiver Capabilities

Parameter	Description	Values
`autoneg_cap`	Local transceiver capability of the hardware	0 = Not capable of auto-negotiation 1 = Auto negotiation capable
`100fdx_cap`	Local transceiver capability of the hardware; initialized at startup	0 = Not 100 Mbit/sec full-duplex capable 1 = 100 Mbit/sec full-duplex capable
`100hdx_cap`	Local transceiver capability of the hardware; initialized at startup	0 = Not 100 Mbit/sec half-duplex capable 1 = 100 Mbit/sec half-duplex capable
`10fdx_cap`	Local transceiver capability of the hardware; initialized at startup	0 = Not 10 Mbit/sec full-duplex capable 1 = 10 Mbit/sec full-duplex capable
`10hdx_cap`	Local transceiver capability of the hardware; initialized at startup	0 = Not 10 Mbit/sec half-duplex capable 1 = 10 Mbit/sec half-duplex capable

Reporting the Link Partner Capabilities

TABLE 3-8 describes the read-only link partner capabilities.

TABLE 3-8 Read-Only Link Partner Capabilities

Parameter	Values	Description
`lp_autoneg_cap`	0= 1=	No auto-negotiation Auto-negotiation
`lp_100fdx_cap`	0= 1=	No 100Mbit/sec full-duplex transmission 100Mbit/sec full-duplex
`lp_100hdx_cap`	0= 1=	No 100Mbit/sec half-duplex transmission 100Mbit/sec half-duplex
`lp_10fdx_cap`	0= 1=	No 10Mbit/sec full-duplex transmission 10Mbit/sec full-duplex
`lp_10hdx_cap`	0= 1=	No 10Mbit/sec half-duplex transmission 10Mbit/sec half-duplex

If the link partner is not capable of auto-negotiation (when `lp_autoneg_cap` is 0) the information described in TABLE 3-8 is not relevant and the parameter value = 0.

If the link partner is capable of auto-negotation (when `lp_autoneg_cap` is 1) then the speed and mode information is displayed when you use auto-negotiation and get the link partner capabilities.

CHAPTER 4

Setting Parameters

This chapter describes how to configure the `eri` driver parameters. Use the `ndd` utility to configure parameters that are valid until you reboot the system.

To configure the `eri` driver parameters for all devices in the system so that the parameter values are always in effect (even after rebooting the system), enter the parameter values in the `/etc/system` file. When the system is rebooted, it reads the `/etc/system` file and sets the parameter values in that file.

To set the parameters for a particular device in the system, set the parameters in the `eri.conf` file in the `/kernel/drv` directory. The parameters set in the `eri.conf` file have precedence over the parameters set in the `/etc/system` file and override the parameters set in the `/etc/system` file. The parameters values set in `eri.conf` are always in effect (even after rebooting the system).

Parameter Options

You can set the `eri` device driver parameters in three ways (`ndd`, `/etc/system`, and `eri.conf`), depending on your needs. To set parameters that are valid until you reboot the system, use the `ndd` utility. Using `ndd` is a good way to test parameter settings.

To set parameters so they remain in effect after you reboot the system:

- Add the parameter values to `/etc/system` when you want to configure parameters for all devices in the system.
- Create the `eri.conf` file and add parameter values to `eri.conf` when you need to set a particular parameter for a device in the system.

If you want to test parameter settings, use the ndd utility described in "Setting Parameters Using ndd". With ndd, the parameters are effective until you reboot the system. To make the parameter settings permanent, enter the values in /etc/system or eri.conf as described in this chapter.

Setting Parameters Using ndd

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 eri driver and the ndd utility to modify (with the -set option) or display (without the -set option) the parameters for each SUNW,eri device.

Identifying Device Instances

Before you use the ndd utility to get or set a parameter for the eri device, you must specify the device instance for the utility if there is more than one SUNW,eri device.

Note – If there is only one SUNW,eri device, the device is automatically chosen by the ndd utility.

▼ To Specify the Device Instance for the ndd Utility

1. **Check the /etc/path_to_inst file to identify the instance associated with a particular device.**
2. **Use that instance number to select the device as follows:**

```
% ndd -set /dev/eri instance instance#
```

The device remains selected until you change the selection.

Non-Interactive and Interactive Modes

You can use the ndd utility in two modes:

- Non-interactive
- Interactive

In non-interactive 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 Non-Interactive Mode

- **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/eri` driver instance, and assigns it to the parameter:

```
% ndd -set /dev/eri parameter value
```

- **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/eri parameter
```

Using the ndd Utility in Interactive Mode

- **To modify a parameter value in interactive mode, specify `ndd eri`, as shown below.**

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

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

After you enter the parameter name, the `ndd` utility prompts you for the parameter value (see TABLE 3-1 through TABLE 3-8 for parameter descriptions).

- **To list all the parameters supported by the `eri` driver, type:**

```
% ndd /dev/eri \?
```

(See TABLE 3-1 through TABLE 3-8 for parameter descriptions.)

CODE EXAMPLE 4-1 Example of Listing All Parameters Supported by the `eri` Driver

```
example# ndd /dev/eri \?
?                               (read only)
transceiver_inuse               (read only)
link_status                     (read only)
link_speed                      (read only)
link_mode                       (read only)
ipg1                            (read and write)
ipg2                            (read and write)
use_int_xcvr                    (read and write)
pace_size                       (read and write)
adv_autoneg_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)
autoneg_cap                     (read only)
100T4_cap                       (read only)
100fdx_cap                      (read only)
100hdx_cap                      (read only)
10fdx_cap                       (read only)
10hdx_cap                       (read only)
lp_autoneg_cap                  (read only)
lp_100fdx_cap                   (read only)
lp_100hdx_cap                   (read only)
lp_10fdx_cap                    (read only)
lp_10hdx_cap                    (read only)
instance                        (read and write)
lance_mode                      (read and write)
ipg0                            (read and write)
example#
```

Setting Forced Mode

The procedure that follows describes how to set forced mode (not capable of auto-negotiation).

▼ To Select a Transceiver Capability and Set Forced Mode

1. **Select one of the following capabilities:** `adv_100fdx_cap`**,** `adv_100hdx_cap`**,** `adv_10fdx_cap`**, or** `adv_10hdx_cap`**, and set its value to** `1`**.**

   If you select more than one of the local transceiver capabilities, the driver selects the one that is highest in the priority order.

2. **Set the local transceiver capabilities advertised by the hardware to forced mode = 0, which is not capable of auto-negotiation:** `adv_autoneg_cap 0`

   Use the `ndd` utility as described in "Using the `ndd` Utility in Interactive Mode" in this chapter.

Auto-Negotiation Mode

▼ To Set the Mode to Auto-Negotiation

1. **Select** *at least one* **of the four capabilities (** `adv_100fdx_cap`**,** `adv_100hdx_cap`**,** `adv_10fdx_cap`**,** `adv_10hdx_cap`**) that you want to advertise to the remote system, and set its value to 1.**

2. **Set the local transceiver capabilities advertised by the hardware to 1, the auto-negotiation setting:** `adv_autoneg_cap 1`

   Use the `ndd` utility as described in "Using the `ndd` Utility in Interactive Mode" on page 21 in this chapter.

Setting Parameters in the `/etc/system` File

To configure the `eri` driver parameters for all `SUNW,eri` devices in the system so that the parameter variables are always effective (even after rebooting the system), enter the parameter variables in the `/etc/system` file. When you reboot the system, the system reads the `/etc/system` file and sets these parameter variables in the `eri` module in the operating system kernel.

TABLE 4-1 lists the variables you need to set in the `/etc/system` file.

TABLE 4-1 Setting Variables in the `/etc/system` File

Parameter	Variable
`ipg1`	`eri_ipg1`
`ipg2`	`eri_ipg2`
`pace_size`	`eri_pace_size`
`adv_autoneg_cap`	`eri_adv_autoneg_cap`
`adv_100fdx_cap`	`eri_adv_100fdx_cap`
`adv_100hdx_cap`	`eri_adv_100hdx_cap`
`adv_10fdx_cap`	`eri_adv_10fdx_cap`
`adv_10hdx_cap`	`eri_adv_10hdx_cap`
`lance_mode`	`eri_lance_mode`
`ipg0`	`eri_ipg0`

These parameter values, described in Chapter 3, are applicable to all `SUNW,eri` devices on the system. See TABLE 3-2 through TABLE 3-8 for parameter descriptions. An example follows.

▼ To Set the `ipg1` to 10 and `ipg2` to 5 When Rebooting

1. **Become superuser.**
2. **Add the following lines to the `/etc/system` file:**

```
set eri:ipg1 = 10
set eri:ipg2 = 5
```

3. **Save the `/etc/system` file.**
4. **Save all files and exit all programs. Exit the windowing system.**
5. **Reboot the system by typing `init 6` at the superuser prompt.**

   The system is halted and then rebooted.

Setting Parameters Using the `eri.conf` File

You can also specify the properties described in the section, “Setting Parameters in the `/etc/system` File”,” in this chapter on a per-device basis by creating the `eri.conf` file in the `/kernel/drv` directory. The properties set in the `eri.conf` file will override the parameters set in the `/etc/system` file. Use `eri.conf` 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 Chapter 3.

The man pages for `prtconf` (1M), `system` (4) and `driver.conf` (4) include additional details. An example follows:

▼ To Configure Driver Parameters Using `eri.conf`

1. **Obtain the hardware path name for the device in the device tree.**

   Typically this path name and the associated instance number will be present in the `/etc/path_to_inst` file. For example, on a Sun Blade 1000 PCI system, the `/etc/path_to_inst` file will have the following entry:

```
“/pci@8,700000/network@5,1” 0 “eri”
```

   - In the entry:
     - The first part within the double quotes specifies the hardware node name in the device tree.
     - The second number is the instance number.
     - 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 at the end represent the device and function numbers, which are together referred to as unit-address.

   To identify a PCI device unambiguously in the `eri.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 PCI device specification.

   In the first line of the previous example:

   - Name = `eri`

- Parent = `/pci@8,700000`
- Unit-address = `5,1`

2. **Set the `ipg1` and `ipg2` parameters for the above device in the `/kernel/drv/eri.conf` file:**

```
name = “eri” parent = “/pci@8,700000” unit-address = “5,1” ipg1=10 ipg2=5;
```