NAME

riles – device driver for the Racal Interlan ES-3210 Ethernet Adapter

SYNOPSIS

/dev/riles

AVAILABILITY

x86

DESCRIPTION

The **riles** Ethernet driver is a multi-threaded, loadable, **gld** (Generic LAN Driver) compliant, clonable, STREAMS hardware driver that supports the connectionless service mode of the Data Link Provider Interface, **dlpi**(7P), over a Racal Interlan ES-3210 (ES-3210) controller. The driver can support multiple ES-3210 controllers on the same system. It provides basic support for the controller such as chip initialization, frame transmission and reception, multicasting and promiscuous mode support, maintenance of error statistic counters, and automatic detection of missing links.

APPLICATION PROGRAMMING INTERFACE

The **riles** driver uses the **gld** (Generic LAN Driver) module of Solaris, which handles all the STREAMS and DLPI specific functions of the driver. It is a *style 2* DLPI driver and supports only the connectionless mode of data transfer. Thus, a DLPI user should issue a **DL_ATTACH_REQ** primitive to select the device to be used. Valid DLPI primitives are defined in **<sys/dlpi.h>**. Refer to **dlpi**(7P) for more information.

The device is initialized on the first attach and de-initialized (stopped) on the last detach.

The values returned by the driver in the **DL_INFO_ACK** primitive in response to a **DL_INFO_REQ** from the user are as follows:

- 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 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 the Ethernet/IEEE broadcast address (**FF:FF:FF:FF:FF:FF**).

Once in the **DL_ATTACHED** state, the user must send a **DL_BIND_REQ** to associate a particular SAP (Service Access Point) with the stream.

CONFIGURATION

riles.conf is the configuration file for the **riles** device driver. The **riles.conf** file should contain the following properties.

intr This specifies the spl level and the IRQ level for which the board has been configured

reg A list of sets where each set contains 3 values specifying the entry number in the **riles.conf** file, and the starting address and size (in bytes) of the onboard RAM (see **sysbus**(4)).

It is the system administrator's responsibility to ensure that these values do not conflict with those used by any other board in the system.

The module **gld**, on which the **riles** driver depends, should be present in the **/kernel/misc** directory.

EXAMPLES

An example of an entry in the **riles.conf** file is given below.

name="riles" class="sysbus" intr=5,3 reg=0,0xc0000,0x4000;

This indicates that the I/O address for the board is **0x300**, the spl level is **5**, the IRQ level is set to **3**, the board's shared memory is configured to begin at **0xc0000** and its memory size is **0x4000** bytes.

FILES

/dev/riles	character special device
/kernel/drv/riles	**riles** device driver
/kernel/drv/riles.conf	**riles** device driver configuration file
/kernel/misc/gld	Generic LAN Driver

SEE ALSO

netstat(1M), **sysbus**(4), **dlpi**(7P), **streamio**(7I)

Writing Device Drivers

STREAMS Programming Guide

Standards Conformance Guide

DIAGNOSTICS

The following diagnostic messages are output by the driver.

riles%d: cannot allocate private data structure (attach failed)
riles%d: cannot allocate transmit buffer (attach failed)
The driver cannot allocate resources needed for its operation. Consequently, the driver does not get loaded.

riles%d: cannot read EISA nvm for slot %d (attach failed)
The driver failed to read the configured settings of the specified board from the EISA non-volatile memory. The driver does not get loaded.

riles%d: no intr property in riles.conf (attach failed)
The **intr** property is not present in the **riles.conf** file. The driver does not get loaded.

riles%d: no match for board irq in riles.conf (attach failed)
No match was found between the configured IRQ value of the board and the value(s) of the **intr** property in the **riles.conf** file.

riles: cannot map RAM address(attach failed)
The driver cannot map the configured RAM address to kernel space. The driver does not get loaded.

riles%d: NIC initialization failed
riles: cannot start NIC of board in slot %d
riles: unable to reset board in slot %d
The driver's attempt to initialize/start the board's network interface controller or reset the board was unsuccessful, probably due to malfunctional hardware.

riles%d: gld_register() unsuccessful (attach failed)
The driver could not register itself with the **gld** module on which it depends. Consequently, the driver does not get loaded.

riles driver(ver. 1.0) for Racal Interlan ES-3210 in slot %d loaded
The driver was loaded after successful initialization of the hardware.

riles: cannot stop board in slot %d
The driver failed to stop the NIC (Network Interface Controller) on the board, probably due to malfunctional hardware.

riles: illegal irq value(%d) for board in slot %d
riles: illegal dma value(%d) for board in slot %d
The IRQ/DMA channel value for the board is not supported. Driver functionality is likely to fail.

riles%d: multicast table full
The number of multicast addresses for which the board has been programmed equaled the maximum allowed by the driver (**16**).

riles%d: no matching multicast entry found
An attempt was made to delete a multicast address for which the board had not been previously programmed.

riles%d: abnormal sized packet; not sent
The driver received an abnormally large packet from the upper layer to be transmitted on the network. Such packets are automatically discarded by the driver.

riles%d: no STREAMS buffers; dropping packet
A STREAMS buffer could not be allocated for passing a received frame upstream. However, the received frame continues to be stored in the board's memory and will be passed upstream when a STREAMS buffer can be allocated.

riles%d: ring buffer overflow
The board ran out of buffers while receiving frames from the network. Although the hardware would recover gracefully, it is a symptom of very high network activity.

riles%d: serious NIC error (RST set)
The reset bit was set in the network interface controller's interrupt status register.

riles%d: board in slot %d failed-- resetting
The ES-3210 hardware has failed and is being reset in an attempt to bring it up.

riles%d: lost carrier (cable or transceiver problem?)
This indicates that the transceiver/cable is faulty or improperly connected to the board.

riles%d: packet received in error (dropped)
The ES-3210 received a bad frame that had to be dropped by the driver.

riles%d: transmit retry count exceeded
The driver's attempts to transmit a packet on the network exceeded the maximum number of retry attempts (usually **5**).

WARNINGS

The driver does not program the chip to receive bad packets. Thus the statistics of bad input frames on a **riles** network interface displayed by **netstat**(1M) is incorrect since it does not take into account the number of bad packets that were discarded by the board's network interface controller.

The **riles** protected mode driver does *not* support I/O-mapped I/O. Hence, the driver will not work with the default configuration of the board; that is, with shared memory disabled. To get around this, you should explicitly choose a memory area for the board at configuration time.

The driver does not use DMA channels 0 through 3, since 32-bit burst mode DMA transfers cannot be accomplished on these channels. Therefore, the driver forcibly uses memory-mapped I/O even when one of the DMA channels 0 through 3 are configured.