NAME

le, lebuffer, ledma – Am7990 (LANCE) Ethernet device driver

SYNOPSIS

/dev/le

DESCRIPTION

The Am7990 (LANCE) Ethernet driver is a multi-threaded, loadable, clonable, STREAMS hardware driver supporting the connectionless Data Link Provider Interface, **dlpi**(7P), over a LANCE Ethernet controller. The motherboard and add-in SBus LANCE controllers of several varieties are supported. Multiple LANCE controllers installed within the system are supported by the driver. The **le** driver provides basic support for the LANCE hardware. Functions include chip initialization, frame transmit and receive, multicast and promiscuous support, and error recovery and reporting.

APPLICATION PROGRAMMING INTERFACE

The cloning character-special device **/dev/le** is used to access all LANCE controllers installed within the system.

The **lebuffer** and **ledma** device drivers are bus nexus drivers which cooperate with the **le** leaf driver in supporting the LANCE hardware functions over several distinct slave-only and DVMA LANCE -based Ethernet controllers. The **lebuffer** and **ledma** bus nexi drivers are not directly accessible to the user.

le and DLPI

The **le** 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) on last detach.

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:

- 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 Ethernet/IEEE broadcast address (**0xFFFFFF**).

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 **le** 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 **le** 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* 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 Ethernet by sending **DL_UNITDATA_REQ** messages to the **le** driver. The **le** 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.

le 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.

FILES

/dev/le	**le** special character device.
/kernel/drv/options.conf	System wide default device driver properties

SEE ALSO

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

SPARCstation 10 Twisted-Pair Ethernet Link Test

Twisted-Pair Ethernet Link Test

DIAGNOSTICS

le%d: msg too big: %d
The message length exceeded **ETHERMAX**.

le%d: Babble error – sent a packet longer than 1518 bytes
While transmitting a packet, the LANCE chip has noticed that the packet's length exceeds the maximum allowed for Ethernet. This error indicates a kernel bug.

le%d: No carrier – transceiver cable problem?
The LANCE chip has lost input to its carrier detect pin while trying to transmit a packet.

le%d: Memory Error!
The LANCE chip timed out while trying to acquire the bus for a DVMA transfer.

NOTES

If you are using twisted pair Ethernet (TPE), you need to be aware of the link test feature. The IEEE 10Base-T specification states that the link test should always be enabled at the host and the hub. Complications may arise because:

1. Some older hubs do not provide link pulses
2. Some hubs are configured to not send link pulses

Under either of these two conditions the host translates the lack of link pulses into a link failure unless it is programmed to ignore link pulses. To program your system to ignore link pulses (also known as disabling the link test) do the following at the OpenBoot PROM prompt:

```
<#0> ok setenv tpe-link-test? false
tpe-link-test? = false
```

The above command will work for **SPARCstation-10, SPARCstation-20** and **SPARCclassic** systems that come with built in twisted pair Ethernet ports. For other systems and for add-on boards with twisted pair Ethernet refer to the documentation that came with the system or board for information on disabling the link test.

SPARCstation-10, SPARCstation-20 and **SPARCclassic** systems come with a choice of built in AUI (using an adapter cable) and TPE ports. From Solaris 2.2 onward an auto-selection scheme was implemented in the **le** driver that will switch between AUI and TPE depending on which interface is active. Auto-selection uses the presence or absence of the link test on the TPE interface as one indication of whether that interface is active. In the special case where you wish to use TPE with the link-test disabled you should manually override auto-selection so that the system will use only the twisted pair port.

This override can be performed by defining the *cable-selection* property in the **options.conf** file to force the system to use TPE or AUI as appropriate. The example below sets the cable selection to TPE.

```
example# cd /kernel/drv
example# echo 'cable-selection="tpe";' >> options.conf
```

Note that the standard **options.conf** file contains important information; the only change to the file should be the addition of the *cable-selection* property. Be careful to type this line *exactly* as shown above, ensuring that you append to the existing file, and include the terminating semi-colon. Alternatively you can use a text editor to append the line

```
cable-selection="tpe";
```

to the end of the file.

Please refer to the *SPARCstation 10 Twisted-Pair Ethernet Link Test* (801-2481-10), *Twisted-Pair Ethernet Link Test* (801-6184-10) and the **driver.conf**(4) man page for details of the syntax of driver configuration files.