NAME

eepro – Intel EtherExpress-Pro Ethernet device driver

SYNOPSIS

/dev/eepro

AVAILABILITY

x86

DESCRIPTION

The **eepro** Ethernet driver is a multi-threaded, loadable, clonable, STREAMS hardware driver supporting the connectionless Data Link Provider Interface, **dlpi**(7P), over Intel EtherExpress-Pro Ethernet controllers. The EtherExpress-Pro Ethernet adapter is based on the Intel 82595TX high integration controller. Multiple EtherExpress-Pro controllers installed within the system are supported by the driver.

The **eepro** driver provides basic support for the EtherExpress-Pro hardware. Functions including chip initialization, frame transmit and receive, multicast and "promiscuous" support, and error recovery and reporting. It also supports an ioctl to perform a time domain reflectometry test (i.e. detect open or short circuits on the link). Refer to **IOCTLS** below.

APPLICATION PROGRAMMING INTERFACE

The cloning, character-special device **/dev/eepro** is used to access all EtherExpress-Pro devices installed within the system.

eepro and DLPI

The **eepro** driver is a "style 2" Data Link Service provider (see **dlpi**(7P)). Valid DLPI primitives are defined in **<sys/dlpi.h>**. All **M_PROTO** and **M_PCPROTO** type messages 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 integer and indicates the corresponding device instance (unit) number. The unit numbers are assigned sequentially to each board found. The search order is determined by the order defined in the **eepro.conf** configuration file. 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 driver will pad to the mandatory 60-byte minimum packet size.
- The **dlsap** address length is **8**.
- The Media Access Control (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 (**FF:FF:FF:FF:FF:FF**).

Once in the **DL_ATTACHED** state, the user must send a **DL_BIND_REQ** to associate a particular Service Access Pointer (SAP) with the stream. The **eepro** 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 also provided by the driver. In this mode, **sap** values in the range [**0-1500**] are 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 within this range, then the driver expects that the destination **DLSAP** in a **DL_UNITDATA_REQ** will contain the *length* of the data rather than a **sap** value. All frames received from the media that have a "type" field in this range are assumed to be 802.3 frames. These frames are routed up all open streams which are bound to any **sap** value within this range. 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 **eepro** 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 should instead 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** request 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 **eepro** driver. The **eepro** driver will route received Ethernet frames up all open and bound streams that have 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.

eepro Primitives

In addition to the mandatory connectionless **DLPI** message set, the driver also 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 the stream or other streams.

The **DL_PHYS_ADDR_REQ** primitive returns the 6-byte 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-byte Ethernet address currently associated (attached) to this stream. The credentials of the process which originally opened this stream must be superuser or an **EPERM** error 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** message 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. The new physical address will remain in effect until this primitive is used to change the physical address again or the system is rebooted, whichever comes first.

IOCTLS

The following ioctl request is supported:

TDR_TEST Performs the time domain reflectometry test. This test checks for network faults. The argument should be **NULL**. The values returned are interpreted as follows:

- **0x8000** No network fault was detected. The transmission link that is connected to the network controller is functioning properly.
- **0x2000** The transmission line connected to the hardware controller is open (i.e. The line has not been properly terminated).
- **0x1000** A short exists on the transmission line.

This ioctl request is defined in **<sys/eepro.h>**.

CONFIGURATION

The **eepro.conf** file supports the following options:

intr Specifies the IRQ level for the board (see **sysbus**(4)).

Auto-detect of the media type is supported and the board need not be explicitly configured for which media connector it is using. It is important to ensure that there are no conflicts between the board's I/O port or IRQ level and other hardware installed in the system.

FILES

/dev/eepro	**eepro** character special device
/kernel/drv/eepro.conf	configuration file of **eepro** driver

SEE ALSO

ioctl(2), **driver.conf**(4), **sysbus**(4), **dlpi**(7P)