NAME

pe – Xircom Pocket Ethernet device driver

SYNOPSIS

```
#include <sys/stropts.h>
#include <sys/ethernet.h>
#include <sys/dlpi.h>
```

AVAILABILITY

x86

DESCRIPTION

The Xircom Pocket Ethernet driver (pe) is a multi-threaded, loadable, clonable, STREAMS hardware driver supporting the connectionless Data Link Provider Interface, **dlpi**(7P), with a Xircom Pocket Ethernet Adapter III (PE3). Multiple PE3 controllers installed within the system are supported by the driver.

PE and DLPI

The **pe** driver provides basic support for the PE3 hardware. Functions include chip initialization, frame transmission and reception, multicast and "promiscuous" support, and error recovery and reporting.

The **pe** driver supports both bi-directional and unidirectional parallel ports. The Port and Adapter type is automatically detected and set when the driver initializes.

It is important not to attempt to use any other driver that may also use the same port when the **pe** driver is operational. This may interfere with network traffic that is currently being sent or received by the PE3 adapter.

The cloning, character-special device **/dev/pe** is used to access all PE3 controllers installed within the system.

The **pe** driver is a “style 2” Data Link Service provider. All M_PROTO and M_PCPROTO type messages are interpreted as DLPI primitives. An explicit DL_ATTACH_REQ message is required by the user to associate the opened stream with a particular device (**ppa**). The **ppa** ID is interpreted as an unsigned long and indicates the corresponding device instance (unit) number. The unit numbers are assigned sequentially to each adapter found. The search order is determined by the order defined in the **pe.conf** 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 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 the DL_INFO_REQ from the user are as follows:

- The max SDU is 1500 (ETHERMTU).
- The min SDU is 0.
- The **dlsap** address length is 8.
- The MAC type is DL_ETHER or DL_CSMACD.
- 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 SAP (Service Access Pointer) with the stream. The **pe** 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 provided by the driver and works as follows. **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 computes the length, not including initial M_PROTO mblk, of all subsequent DL_UNITDATA_REQ messages and transmits 802.3 frames having this value in the MAC frame header length field. All frames received from the media having a "type" field in this range are assumed to be 802.3 frames and are routed up all open streams that 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 **pe** 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 **pe** driver. The **pe** 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.

PE Primitives

In addition to the mandatory connectionless DLPI message set, the driver additionally supports the primitives discussed below.

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 that originally opened this stream must be superuser or 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 on this stream is successful. Once changed, all streams subsequently opened and attached to this device will obtain this new physical address. Once changed, the 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.

CONFIGURATION

The **/kernel/drv/pe.conf** file supports the following options:

intr Specifies the IRQ level for the parallel port that the Xircom Adapter is connected to.

ioaddr Specifies the I/O address for the parallel port that the Xircom Adapter is connected to.

It is important to ensure that there are no conflicts for the adapter's I/O port or IRQ level.

FILES

/dev/pe

SEE ALSO

dlpi(7P)