NAME

fxp - Intel EtherExpress PRO/100 Ethernet device driver

SYNOPSIS

To compile this driver into the kernel, place the following lines in your kernel configuration file:

```
device miibus
device fxp
```

Alternatively, to load the driver as a module at boot time, place the following line in loader.conf(5):

```
if_fxp_load="YES"
```

DESCRIPTION

The **fxp** driver provides support for Ethernet adapters based on the Intel i82557, i82558, i82559, i82550, and i82562 chips. The driver supports TCP/UDP/IP checksum offload for both transmit and receive on i82550 and i82551. On i82559 only TCP/UDP checksum offload for receive is supported. TCP segmentation offload (TSO) for IPv4 as well as VLAN hardware tag insertion/stripping is supported on i82550 and i82551. Wake On Lan (WOL) support is provided on all controllers except i82557, i82259ER and early i82558 revisions.

The **fxp** driver supports the following media types:

autoselect Enable autoselection of the media type and options. The autoselected mode can be overridden by adding the media options to rc.conf(5).

10baseT/UTP Set 10Mbps operation.

100baseTX Set 100Mbps (Fast Ethernet) operation.

The **fxp** driver supports the following media options:

full-duplex Force full duplex operation.

half-duplex
Force half duplex operation.

Note that 100baseTX media type is not available on the Pro/10. For further information on configuring this device, see ifconfig(8).

The **fxp** driver supports reception and transmission of extended frames for vlan(4). This capability of **fxp** can be controlled by means of the **vlanmtu** parameter to ifconfig(8).

The **fxp** driver also supports a special link option:

link0 Some chip revisions have loadable microcode which can be used to reduce the interrupt load on the host cpu. Not all boards have microcode support. Setting the **link0** flag with ifconfig(8) will download the microcode to the chip if it is available.

HARDWARE

Adapters supported by the **fxp** driver include:

- Intel EtherExpress PRO/10
- Intel InBusiness 10/100
- Intel PRO/100B / EtherExpressPRO/100 B PCI Adapter
- Intel PRO/100+ Management Adapter
- Intel PRO/100 VE Desktop Adapter
- Intel PRO/100 VM Network Connection
- Intel PRO/100 M Desktop Adapter
- Intel PRO/100 S Desktop, Server and Dual-Port Server Adapters
- Many on-board network interfaces on Intel motherboards

LOADER TUNABLES

Tunables can be set at the loader(8) prompt before booting the kernel or stored in loader.conf(5). The following variables are available as both loader(8) tunables and sysctl(8) variables:

dev.fxp.%d.int_delay

Maximum amount of time, in microseconds, that an interrupt may be delayed in an attempt to coalesce interrupts. This is only effective if the Intel microcode is loaded. The accepted range is 300 to 3000, the default is 1000.

dev.fxp.%d.bundle_max

Number of packets that will be bundled, before an interrupt is generated. This is only effective if the Intel microcode is loaded. The accepted range is 1 to 65535, the default is 6.

SYSCTL VARIABLES

The following variables are available as sysctl(8) variables.

dev.fxp.%d.rnr

This is a read-only variable and shows the number of events of RNR (resource not ready).

dev.fxp.%d.stats

This is a read-only variable and displays useful MAC counters maintained in the driver.

DIAGNOSTICS

fxp%d: couldn't map memory A fatal initialization error has occurred.

fxp%d: couldn't map interrupt A fatal initialization error has occurred.

fxp%d: Failed to malloc memory There are not enough mbuf's available for allocation.

fxp%d: device timeout The device has stopped responding to the network, or there is a problem with the network connection (cable).

fxp%d: Microcode loaded, int_delay: %d usec bundle_max: %d The chip has successfully downloaded the microcode, and changed the parameterized values to the given settings.

SEE ALSO

altq(4), arp(4), miibus(4), netintro(4), ng_ether(4), polling(4), vlan(4), ifconfig(8)

HISTORY

The **fxp** device driver first appeared in FreeBSD 2.1.

AUTHORS

The **fxp** device driver was written by David Greenman. It has then been updated to use the busdma API and made endian-clean by Maxime Henrion. This manual page was written by David E. O'Brien.