NAME

dc - DEC/Intel 21143 and clone 10/100 Ethernet driver

SYNOPSIS

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

```
device miibus
device dc
```

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

```
if_dc_load="YES"
```

DESCRIPTION

The **dc** driver provides support for several PCI Fast Ethernet adapters and embedded controllers based on the DEC/Intel 21143 chipset and clones.

All of supported chipsets have the same general register layout, DMA descriptor format and method of operation. All of the clone chips are based on the 21143 design with various modifications. The 21143 itself has support for 10baseT, BNC, AUI, MII and symbol media attachments, 10 and 100Mbps speeds in full or half duplex, built in NWAY autonegotiation and wake on LAN. The 21143 also offers several receive filter programming options including perfect filtering, inverse perfect filtering and hash table filtering.

Some clone chips duplicate the 21143 fairly closely while others only maintain superficial similarities. Some support only MII media attachments. Others use different receiver filter programming mechanisms. At least one supports only chained DMA descriptors (most support both chained descriptors and contiguously allocated fixed size rings). Some chips (especially the PNIC) also have peculiar bugs. The **dc** driver does its best to provide generalized support for all of these chipsets in order to keep special case code to a minimum.

These chips are used by many vendors which makes it difficult to provide a complete list of all supported cards.

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

autoselect Enable autoselection of the media type and options. The user can manually override the autoselected mode by adding media options to the */etc/rc.conf* file.

Note: the built-in NWAY autonegotiation on the original PNIC 82c168 chip is horribly

broken and is not supported by the **dc** driver at this time (see the *BUGS* section for details). The original 82c168 appears on very early revisions of the LinkSys LNE100TX and Matrox FastNIC.

10baseT/UTP Set 10Mbps operation. The **mediaopt** option can also be used to enable **full-duplex** operation. Not specifying **full-duplex** implies **half-duplex** mode.

100baseTX Set 100Mbps (Fast Ethernet) operation. The **mediaopt** option can also be used to enable **full-duplex** operation. Not specifying **full-duplex** implies **half-duplex** mode.

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

full-duplex Force full duplex operation. The interface will operate in half duplex mode if this media option is not specified.

Note that the 100baseTX media type may not be available on certain Intel 21143 adapters which support 10Mbps media attachments only. For more information on configuring this device, see ifconfig(8).

HARDWARE

The **dc** driver provides support for the following chipsets:

- DEC/Intel 21143
- ADMtek AL981 Comet, AN985 Centaur, ADM9511 Centaur II and ADM9513 Centaur II
- ALi/ULi M5261 and M5263
- ASIX Electronics AX88140A and AX88141
- Conexant LANfinity RS7112 (miniPCI)
- Davicom DM9009, DM9100, DM9102 and DM9102A
- Lite-On 82c168 and 82c169 PNIC
- Lite-On/Macronix 82c115 PNIC II
- Macronix 98713, 98713A, 98715, 98715A, 98715AEC-C, 98725, 98727 and 98732
- Xircom X3201 (cardbus only)

The following NICs are known to work with the **dc** driver at this time:

- 3Com OfficeConnect 10/100B (ADMtek AN985 Centaur-P)
- Abocom FE2500
- Accton EN1217 (98715A)
- Accton EN2242 MiniPCI
- Adico AE310TX (98715A)
- Alfa Inc GFC2204 (ASIX AX88140A)

- Built in 10Mbps only Ethernet on Compaq Presario 7900 series desktops (21143, non-MII)
- Built in Ethernet on LinkSys EtherFast 10/100 Instant GigaDrive (DM9102, MII)
- CNet Pro110B (ASIX AX88140A)
- CNet Pro120A (98715A or 98713A) and CNet Pro120B (98715)
- Compex RL100-TX (98713 or 98713A)
- D-Link DFE-570TX (21143, MII, quad port)
- Digital DE500-BA 10/100 (21143, non-MII)
- ELECOM Laneed LD-CBL/TXA (ADMtek AN985)
- Hawking CB102 CardBus
- IBM EtherJet Cardbus Adapter
- Intel PRO/100 Mobile Cardbus (versions that use the X3201 chipset)
- Jaton XpressNet (Davicom DM9102)
- Kingston KNE100TX (21143, MII)
- Kingston KNE110TX (PNIC 82c169)
- LinkSys LNE100TX (PNIC 82c168, 82c169)
- LinkSys LNE100TX v2.0 (PNIC II 82c115)
- LinkSys LNE100TX v4.0/4.1 (ADMtek AN985 Centaur-P)
- Matrox FastNIC 10/100 (PNIC 82c168, 82c169)
- Melco LGY-PCI-TXL
- Microsoft MN-120 10/100 CardBus (ADMTek Centaur-C)
- Microsoft MN-130 10/100 PCI (ADMTek Centaur-P)
- NDC SOHOware SFA110A (98713A)
- NDC SOHOware SFA110A Rev B4 (98715AEC-C)
- NetGear FA310-TX Rev. D1, D2 or D3 (PNIC 82c169)
- Netgear FA511
- PlaneX FNW-3602-T (ADMtek AN985)
- SMC EZ Card 10/100 1233A-TX (ADMtek AN985)
- SVEC PN102-TX (98713)
- Xircom Cardbus Realport
- Xircom Cardbus Ethernet 10/100
- Xircom Cardbus Ethernet II 10/100

DIAGNOSTICS

dc%d: couldn’t map ports/memory A fatal initialization error has occurred.

dc%d: couldn’t map interrupt A fatal initialization error has occurred.

dc%d: watchdog timeout A packet was queued for transmission and a transmit command was issued, but the device failed to acknowledge the transmission before a timeout expired. This can happen if the device is unable to deliver interrupts for some reason, of if there is a problem with the network

connection (cable or network equipment) that results in a loss of link.

dc%d: no memory for rx list The driver failed to allocate an mbuf for the receiver ring.

dc%d: TX underrun -- increasing TX threshold The device generated a transmit underrun error while attempting to DMA and transmit a packet. This happens if the host is not able to DMA the packet data into the NIC's FIFO fast enough. The driver will dynamically increase the transmit start threshold so that more data must be DMAed into the FIFO before the NIC will start transmitting it onto the wire.

dc%d: TX underrun -- using store and forward mode The device continued to generate transmit underruns even after all possible transmit start threshold settings had been tried, so the driver programmed the chip for store and forward mode. In this mode, the NIC will not begin transmission until the entire packet has been transferred into its FIFO memory.

dc%d: chip is in D3 power state -- setting to D0 This message applies only to adapters which support power management. Some operating systems place the controller in low power mode when shutting down, and some PCI BIOSes fail to bring the chip out of this state before configuring it. The controller loses all of its PCI configuration in the D3 state, so if the BIOS does not set it back to full power mode in time, it will not be able to configure it correctly. The driver tries to detect this condition and bring the adapter back to the D0 (full power) state, but this may not be enough to return the driver to a fully operational condition. If you see this message at boot time and the driver fails to attach the device as a network interface, you will have to perform a second warm boot to have the device properly configured.

Note that this condition only occurs when warm booting from another operating system. If you power down your system prior to booting FreeBSD, the card should be configured correctly.

SEE ALSO

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

ADMtek AL981, AL983 and AL985 data sheets, http://www.admtek.com.tw.

ASIX Electronics AX88140A and AX88141 data sheets, http://www.asix.com.tw.

Davicom DM9102 data sheet,
http://www.davicom.com.tw/userfile/24247/DM9102H-DS-F01-021508.pdf.

Intel 21143 Hardware Reference Manual, http://developer.intel.com.

Macronix 98713/A, 98715/A and 98725 data sheets, http://www.macronix.com.

Macronix 98713/A and 98715/A app notes, http://www.macronix.com.

HISTORY

The **dc** device driver first appeared in FreeBSD 4.0.

AUTHORS

The **dc** driver was written by Bill Paul <*wpaul@ee.columbia.edu*>.

BUGS

The Macronix application notes claim that in order to put the chips in normal operation, the driver must write a certain magic number into the CSR16 register. The numbers are documented in the app notes, but the exact meaning of the bits is not.

The 98713A seems to have a problem with 10Mbps full duplex mode. The transmitter works but the receiver tends to produce many unexplained errors leading to very poor overall performance. The 98715A does not exhibit this problem. All other modes on the 98713A seem to work correctly.

The original 82c168 PNIC chip has built in NWAY support which is used on certain early LinkSys LNE100TX and Matrox FastNIC cards, however it is horribly broken and difficult to use reliably. Consequently, autonegotiation is not currently supported for this chipset: the driver defaults the NIC to 10baseT half duplex, and it is up to the operator to manually select a different mode if necessary. (Later cards use an external MII transceiver to implement NWAY autonegotiation and work correctly.)

The **dc** driver programs 82c168 and 82c169 PNIC chips to use the store and forward setting for the transmit start threshold by default. This is to work around problems with some NIC/PCI bus combinations where the PNIC can transmit corrupt frames when operating at 100Mbps, probably due to PCI DMA burst transfer errors.

The 82c168 and 82c169 PNIC chips also have a receiver bug that sometimes manifests during periods of heavy receive and transmit activity, where the chip will improperly DMA received frames to the host. The chips appear to upload several kilobytes of garbage data along with the received frame data, dirtying several RX buffers instead of just the expected one. The **dc** driver detects this condition and will salvage the frame; however, it incurs a serious performance penalty in the process.

The PNIC chips also sometimes generate a transmit underrun error when the driver attempts to download the receiver filter setup frame, which can result in the receive filter being incorrectly programmed. The **dc** driver will watch for this condition and requeue the setup frame until it is transferred successfully.

The ADMtek AL981 chip (and possibly the AN985 as well) has been observed to sometimes wedge on

transmit: this appears to happen when the driver queues a sequence of frames which cause it to wrap from the end of the transmit descriptor ring back to the beginning. The **dc** driver attempts to avoid this condition by not queuing any frames past the end of the transmit ring during a single invocation of the **dc_start**() routine. This workaround has a negligible impact on transmit performance.