DEC EtherWORKS Ethernet De4x5 cards

Originally, this driver was written for the Digital Equipment Corporation series of EtherWORKS Ethernet cards:

- DE425 TP/COAX EISA
- DE434 TP PCI
- DE435 TP/COAX/AUI PCI
- DE450 TP/COAX/AUI PCI
- DE500 10/100 PCI Fasternet

but it will now attempt to support all cards which conform to the Digital Semiconductor SROM Specification. The driver currently recognises the following chips:

- DC21040 (no SROM)
- DC21041[A]
- DC21140[A]
- DC21142
- DC21143

So far the driver is known to work with the following cards:

- KINGSTON
- Linksys
- ZNYX342
- SMC8432
- SMC9332 (w/new SROM)
- ZNYX31[45]
- ZNYX346 10/100 4 port (can act as a 10/100 bridge!)

The driver has been tested on a relatively busy network using the DE425, DE434, DE435 and DE500 cards and benchmarked with 'ttcp': it transferred 16M of data to a DECstation 5000/200 as follows:

```
            TCP           UDP
          TX     RX     TX     RX
DE425   1030k  997k   1170k  1128k
DE434   1063k  995k   1170k  1125k
DE435   1063k  995k   1170k  1125k
DE500   1063k  998k   1170k  1125k  in 10Mb/s mode
```

All values are typical (in kBytes/sec) from a sample of 4 for each measurement. Their error is +/-20k on a quiet (private) network and also depend on what load the CPU has.

The ability to load this driver as a loadable module has been included and used extensively during the driver development (to save those long reboot sequences). Loadable module support under PCI and EISA has been achieved by letting the driver autoprobe as if it were compiled into the kernel. Do make sure you're not sharing interrupts with anything that cannot accommodate interrupt sharing!

To utilise this ability, you have to do 8 things:

0. have a copy of the loadable modules code installed on your system.
1. copy de4x5.c from the /linux/drivers/net directory to your favourite temporary directory.
2. for fixed autoprobes (not recommended), edit the source code near line 5594 to reflect the I/O address you're using, or assign these when loading by:

   ```
   insmod de4x5 io=0xghh           where g = bus number
                                         hh = device number
   ```

   Note

   autoprobing for modules is now supported by default. You may just use:

   ```
   insmod de4x5
   ```

   to load all available boards. For a specific board, still use the 'io=?' above.

3. compile de4x5.c, but include -DMODULE in the command line to ensure that the correct bits are compiled (see end of source code).

4. if you are wanting to add a new card, goto 5. Otherwise, recompile a kernel with the de4x5 configuration turned off and reboot.
5. insmod de4x5 [io=0xghh]
6. run the net startup bits for your new eth?? interface(s) manually (usually /etc/rc.inet[12] at boot time).
7. enjoy!

To unload a module, turn off the associated interface(s) 'ifconfig eth?? down' then 'rmmod de4x5'.

Automedia detection is included so that in principle you can disconnect from, e.g. TP, reconnect to BNC and things will still work (after a pause while the driver figures out where its media went). My tests using ping showed that it appears to work....

By default, the driver will now autodetect any DECchip based card. Should you have a need to restrict the driver to DIGITAL only cards, you can compile with a DEC_ONLY define, or if loading as a module, use the 'dec_only=1' parameter.

I've changed the timing routines to use the kernel timer and scheduling functions so that the hangs and other assorted problems that occurred while autosensing the media should be gone. A bonus for the DC21040 auto media sense algorithm is that it can now use one that is more in line with the rest (the DC21040 chip doesn't have a hardware timer). The downside is the 1 'jiffies' (10ms) resolution.

IEEE 802.3u MII interface code has been added in anticipation that some products may use it in the future.

The SMC9332 card has a non-compliant SROM which needs fixing - I have patched this driver to detect it because the SROM format used complies to a previous DEC-STD format.

I have removed the buffer copies needed for receive on Intels. I cannot remove them for Alphas since the Tulip hardware only does longword aligned DMA transfers and the Alphas get alignment traps with non longword aligned data copies (which makes them really slow). No comment.

I have added SROM decoding routines to make this driver work with any card that supports the Digital Semiconductor SROM spec. This will help all cards running the dc2114x series chips in particular. Cards using the dc2104x chips should run correctly with the basic driver. I'm in debt to <mjacob@feral.com> for the testing and feedback that helped get this feature working. So far we have tested KINGSTON, SMC8432, SMC9332 (with the latest SROM complying with the SROM spec V3: their first was broken), ZNYX342 and LinkSys. ZNYX314 (dual 21041 MAC) and ZNYX 315 (quad 21041 MAC) cards also appear to work despite their incorrectly wired IRQs.

I have added a temporary fix for interrupt problems when some SCSI cards share the same interrupt as the DECchip based cards. The problem occurs because the SCSI card wants to grab the interrupt as a fast interrupt (runs the service routine with interrupts turned off) vs. this card which really needs to run the service routine with interrupts turned on. This driver will now add the interrupt service routine as a fast interrupt if it is bounced from the slow interrupt. THIS IS NOT A RECOMMENDED WAY TO RUN THE DRIVER and has been done for a limited time until people sort out their compatibility issues and the kernel interrupt service code is fixed. YOU SHOULD SEPARATE OUT THE FAST INTERRUPT CARDS FROM THE SLOW INTERRUPT CARDS to ensure that they do not run on the same interrupt. PCMCIA/CardBus is another can of worms...

Finally, I think I have really fixed the module loading problem with more than one DECchip based card. As a side effect, I don't mess with the device structure any more which means that if more than 1 card in 2.0.x is installed (4 in 2.1.x), the user will have to edit linux/drivers/net/Space.c to make room for them. Hence, module loading is the preferred way to use this driver, since it doesn't have this limitation.

Where SROM media detection is used and full duplex is specified in the SROM, the feature is ignored unless lp->params.fdx is set at compile time OR during a module load (insmod de4x5 args='eth??:fdx' [see below]). This is because there is no way to automatically detect full duplex links except through autonegotiation. When I include the autonegotiation feature in the SROM autoconf code, this detection will occur automatically for that case.

Command line arguments are now allowed, similar to passing arguments through LILO. This will allow a per adapter board set up of full duplex and media. The only lexical constraints are: the board name (dev->name) appears in the list before its parameters. The list of parameters ends either at the end of the parameter list or with another board name. The following parameters are allowed:

fdx	for full duplex
autosense	to set the media/speed; with the following sub-parameters: TP, TP_NW, BNC, AUI, BNC_AUI, 100Mb, 10Mb, AUTO

Case sensitivity is important for the sub-parameters. They *must* be upper case. Examples:

```
insmod de4x5 args='eth1:fdx autosense=BNC eth0:autosense=100Mb'.
```

For a compiled in driver, in linux/drivers/net/CONFIG, place e.g.:

```
DE4X5_OPTS = -DDE4X5_PARM='"eth0:fdx autosense=AUI eth2:autosense=TP"'
```

Yes, I know full duplex isn't permissible on BNC or AUI; they're just examples. By default, full duplex is turned off and AUTO is the default autosense setting. In reality, I expect only the full duplex option to be used. Note the use of single quotes in the two examples above and the lack of commas to separate items.