Linux Ethernet-Howto

Paul Gortmaker, Editor. v2.3, 11/02/95

This is the Ethernet-Howto, which is a compilation of information about which ethernet devices can be used for Linux, and how to set them up. It hopefully answers all the frequently asked questions about using ethernet cards with Linux. Note that this Howto is focused on the hardware and low level driver aspect of the ethernet cards, and does not cover the software end of things. See the NET2-Howto for that stuff.

Contents

1 Introduction

The Ethernet-Howto covers what cards you should and shouldn't buy; how to set them up, how to run more than one, and other common problems and questions. It contains detailed information on the current level of support for *all of the most common ethernet cards available.* It does *not* cover the software end of things, as that is covered in the NET-2 Howto. Also note that general non-Linux specific questions about Ethernet are not (or at least they should not be) answered here. For those types of questions, see the excellent amount of information in the *comp.dcom.lans.ethernet* FAQ. You can FTP it from `dorm.rutgers.edu` in the directory `/pub/novell/info_and_docs/`

This present revision covers kernels up to and including v1.1.91

The Ethernet-Howto is edited and maintained by:

Paul Gortmaker, `Paul.Gortmaker@anu.edu.au`

The primary source of the information for the Ethernet-Howto is from:

Donald J. Becker, `becker@cesdis.gsfc.nasa.gov`

who we have to thank for writing the vast majority of ethernet card drivers that are presently available for Linux. He also is the original author of the NFS server too. Thanks Donald! We owe ya one! :-)

Net-surfers may wish to check out the following URL:

Donald Becker (`http://cesdis.gsfc.nasa.gov/pub/people/becker/whoiam.html`)

1.1 Using the Ethernet-Howto

As this guide is getting bigger and bigger, you probably don't want to spend the rest of your afternoon reading the whole thing. And you don't *have* to read it all. If you haven't got an ethernet card, then you will want to start with 2 to see what you should buy, and what you should avoid. If you have already got an ethernet card, but are not sure if you can use it with Linux, then you will want to read 3 which contains specific information on each manufacturer, and their cards. If you are having trouble with your card, then you will want to read the specific information about your card mentioned above, and the troubleshooting information in 9. If you are interested in some of the technical aspects of the device drivers, then you can find that information in 7

1.2 Disclaimer and Copyright

This document is *not* gospel. However, it is probably the most up to date info that you will be able to find. Nobody is responsible for what happens to your hardware but yourself. If your ethercard or any other hardware goes up in smoke (...nearly impossible!) we take no responsibility. ie. THE AUTHORS ARE NOT RESPONSIBLE FOR ANY DAMAGES INCURRED DUE TO ACTIONS TAKEN BASED ON THE INFORMATION INCLUDED IN THIS DOCUMENT.

If you are intending to incorporate this document into a published work, please contact me, and I will make an effort to ensure that you have the most up to date information available. In the past, out of date versions of the Linux howto documents have been published, which caused the developers undue grief from being plagued with questions that were already answered in the up to date versions.

1.3 Mailing Lists and the Linux Newsgroups

If you have questions about your ethernet card, please READ this document first. You may also want to join the NET channel of the Linux-activists mailing list by sending mail to

```
linux-activists-request@niksula.hut.fi
```

with the line

```
X-Mn-Admin: join NET
```

at the top of the message body (not the subject). If you want to learn how to use the mailing channels, then send an empty message to the above address, and you will get an instruction manual sent back to you in a few hours. However, it is worth noting that the NET channel is primarily used for discussion of the networking code, and you may not see much discussion about a particular driver.

Furthermore keep in mind that the NET channel is for development discussions only. General questions on how to configure your system should be directed to comp.os.linux.help unless you are actively involved in the development of part of the networking for Linux. We ask that you *please* respect this general guideline for content.

Recently, a similar group of mailing lists has started on the host `vger.rutgers.edu`, using the much more common `majordomo` mailing list server. Send mail to `majordomo@vger.rutgers.edu` to get help with what lists are available, and how to join them.

Also, the news groups *comp.sys.ibm.pc.hardware.networking* and *comp.dcom.lans.ethernet* should be used for questions that are not Linux specific.

1.4 Related Documentation

Much of this info came from saved postings from the comp.os.linux groups, which shows that it is a valuable resource of information. Other useful information came from a bunch of small files by Donald himself. Of course, if you are setting up an Ethernet card, then you will want to read the NET-2 Howto so that you can actually configure the software you will use. And last but not least, the contributions from the individuals and companies listed in 10.3 is greatly appreciated as well. Oh yeah, if you fancy yourself as a bit of a hacker, you can always scrounge some additional info from the driver source files as well. There is usually a paragraph in there describing any important points.

For those looking for information that is not specific in any way to Linux (i.e. what is 10BaseT, what is AUI, what does a hub do, etc.) I strongly recommend the **Ethernet-FAQ** from the newsgroup *comp.dcom.lans.ethernet*. Look on the FTP site `dorm.rutgers.edu` in the directory `/pub/novell/info_and_docs/` or grab it from the following URL:

Ethernet FAQ `(ftp://dorm.rutgers.edu/pub/novell/info_and_docs/Ethernet.FAQ)`

Don't let the fact that it was last revised in 1993 scare you, as not much has happened to Ethernet since then. (Discounting the upcoming 100Base-whatever, of course.)

1.5 New Versions of this Document

New versions of this document can be retrieved via anonymous FTP from sunsite.unc.edu, in /pub/Linux/docs/HOWTO/* and various Linux ftp mirror sites. Updates will be made as new information / drivers becomes available. If this copy that you are reading is more than 2 months old, it is either out of date, or it means that I have been lazy and haven't updated it. This document was produced by using the SGML system that was specifically set up for the Linux Howto project, and there are various output formats available, including, postscript, dvi, ascii, html, and soon TeXinfo.

I would recommend viewing it in the html (via Mosaic) or the Postscript/dvi format. Both of these contain cross-references that are lost in the ascii translation.

If you want to get the official copy off sunsite, here is URL.

Ethernet-HOWTO (`http://sunsite.unc.edu/mdw/HOWTO/Ethernet-HOWTO.html`)

If minor additions and changes have been made, you can view the latest working copy from this URL.

Working Copy (`http://rsphy1.anu.edu.au/ gpg109/Ethernet-HOWTO.html`)

2 What card should I buy for Linux?

For impatient users that just want a quick, cheap answer the summary is: get 16 bit thinnet 8013 cards. For those who want the absolute best performance, get an AMD PC-Net/Lance based card. For more detail as to the who what where and why, read on.

2.1 Eight bit vs 16 bit

Unless you are a light user, or are confined to using the smaller ISA slot, the use of the 8 bit cards like the wd8003, the 3c503 and the ne1000 is usually not worth the cost savings. Get the 8013 or the 3c503/16, or the ne2000 instead. (The 3c501 is not included in this discussion, as it shouldn't be used under any circumstances.)

However, so not to leave you with a bad taste in your mouth if you happen to already have one, you can still expect to get about 500kB/s ftp download speed to an 8 bit wd8003 card (on a 16MHz ISA bus) from a fast host. And if most of your net-traffic is going to remote sites, then the bottleneck in the path will be elsewhere, and the only speed difference you will notice is during net activity on your local subnet.

A note to NFS users: Some people have found that using 8 bit cards in NFS clients causes poorer than expected performance, when using 8kB (native Sun) NFS packet size.

The possible reason for this could be due to the difference in on board buffer size between the 8 bit and the 16 bit cards. The 8 bit cards have an 8kB buffer, and the 16 bit cards have a 16kB buffer. The Linux driver will reserve 3kB of that buffer (for Tx ping-pong buffers), leaving only 5kB for an 8 bit card. The maximum ethernet packet size is about 1500 bytes. Now that 8kB NFS packet will arrive as about 6 back to back maximum size ethernet packets. Both the 8 and 16 bit cards have no problem Rx'ing back to back packets. The problem arises when the machine doesn't remove the packets from the cards buffer in time, and the buffer overflows. The fact that 8 bit cards take an extra ISA bus cycle per transfer doesn't help either.

What you *can* do if you have an 8 bit card is either set the NFS transfer size to 4kB, or try increasing the ISA bus speed in order to get the card's buffer cleared out faster.

2.2 Low price Ethernet cards

The lowest price seen so far was in the March '94 edition of LAN magazine. There was an ad for Addtron AE-200 cards (jumper settable NE2000 clones) for a measly $19 ea! Unfortunately this offer has since expired. However, you might want to check to see what their everyday price is.

You can also call AT-LAN-TEC at 301-948-7070. Ask for their technical support person. As with all purchases, you should indicate you are buying this for a Linux system. NB: Their current NE2000 clone is a model that 'traps' other drivers that probe into their address space. AT-LAN-TEC also carries a clone, non-EEPROM 8013 board for somewhat more, and a NE2100 clone. Either is a better choice if the very lowest price isn't essential.

And a recent addition is the VLB and PCI cards offered by Boca Research. These are selling for around the $70 mark, and these are supported with the latest kernel. These use the new 32 bit versions of the LANCE chip from AMD. See 3.10 for more info.

If you require an ISA card, you can use the the Allied Telesis AT1500 which uses the ISA version of the LANCE chip from AMD. It is offered at a good price by many vendors. Even Inmac, known for their premium markup, has this card for under $100. See 3.3.1 for more info.

2.3 Vendors and Brands to Avoid

These vendors have decided *not* to release programming information about their products, without signing a NDA (non-disclosure agreement). Hence it is strongly advised that you avoid buying products offered from these companies.

(1) Cabletron (see 3.11)

(2) Xircom (see 3.28)

These particular cards should be avoided, as they are obsolete. The reasons as to why they have been classified as such can be found in their respective sections. For your particular application, these reasons may not be a concern, so you should have a read of the reasons listed.

(1) 3c501 (see 3.1.1)

(2) Arcnet based cards (see 3.7)

(3) Any 8-bit cards (see 2.1)

2.4 Type of cable that your card should support

Unless you have to conform to an existing network, you will want to use thinnet or thin ethernet cable. This is the style with the standard BNC connectors. See 5 for other concerns with different types of ethernet cable.

Most ethercards also come in a 'Combo' version for only $10-$20 more. These have both twisted pair and thinnet transceiver built-in, allowing you to change your mind later.

The twisted pair cables, with the RJ-45 (giant phone jack) connectors is technically called 10BaseT. You may also hear it called UTP (Unsheilded Twisted Pair).

The thinnet, or thin ethernet cabling, (RG-58 coaxial cable) with the BNC (metal push and turn-to-lock) connectors is technically called 10Base2.

The older thick ethernet (10mm coaxial cable) which is only found in older installations is called 10Base5.

Large corporate installations will most likely use 10BaseT instead of 10Base2. 10Base2 does not offer an easy upgrade path to the new upcoming 100Base-whatever.

3 Vendor/Manufacturer/Model Specific Information

The only thing that one needs to use an ethernet card with Linux is the appropriate driver. For this, it is essential that the manufacturer will release the technical programming information to the general public without you (or anyone) having to sign your life away. A good guide for the likelihood of getting documentation (or, if you aren't writing code, the likelihood that someone else will write that driver you really, really need) is the availability of the Crynwr (nee Clarkson) packet driver. Russ Nelson runs this operation, and has been very helpful in supporting the development of drivers for Linux. *Net-surfers* can try this URL to look up Russ' software.

Russ Nelson's Packet Drivers (`http://www.crynwr.com/crynwr/home.html`)

Given the documentation, you can write a driver for your card and use it for Linux (at least in theory) and if you intend to write a driver, have a look at 7.2 as well. Keep in mind that some old hardware that was designed for XT type machines will not function very well in a multitasking environment such as Linux. Use of these will lead to major problems if your network sees a reasonable amount of traffic.

Most cards come with drivers for MS-DOS interfaces such as NDIS and ODI, but these are useless for Linux. Many people have suggested directly linking them in or automatic translation, but this is nearly impossible. The MS-DOS drivers expect to be in 16 bit mode and hook into 'software interrupts', both incompatible with the Linux kernel. This incompatibility is actually a feature, as some Linux drivers are considerably better than their MS-DOS counterparts. The '8390' series drivers, for instance, use ping-pong transmit buffers, which are only now being introduced in the MS-DOS world.

Keep in mind that PC ethercards have the widest variety of interfaces (shared memory, programmed I/O, bus-master, or slave DMA) of any computer hardware for anything, and supporting a new ethercard sometimes requires re-thinking most of the lower-level networking code. (If you are interested in learning more about these different forms of interfaces, see 7.5.)

Also, similar product numbers don't always indicate similar products. For instance, the 3c50* product line from 3Com varies wildly between different members.

Enough talk. Let's get down to the information you want.

3.1 3Com

If you are not sure what your card is, but you think it is a 3Com card, you can probably figure it out from the assembly number. 3Com has a document 'Identifying 3Com Adapters By Assembly Number' (ref 24500002) that would most likely clear things up. See 7.7 for info on how to get documents from 3Com.

Also note that 3Com has a FTP site with various goodies: `ftp.3Com.com` that you may want to check out.

3.1.1 3c501

Status – *Semi-Supported*

Too brain-damaged to use. Available surplus from many places. Avoid it like the plague. Again, do not purchase this card, even as a joke. It's performance is horrible, and it breaks in many ways.

Cameron L. Spitzer of 3Com said: "I'm speaking only for myself here, of course, but I believe 3Com advises against installing a 3C501 in a new system, mostly for the same reasons Donald has discussed. You probably won't be happy with the 3C501 in your Linux box. The data sheet is marked '(obsolete)' on 3Com's Developers' Order Form, and the board is not part of 3Com's program for sending free Technical Reference Manuals to people who need them. The decade-old things are nearly indestructible, but that's about all they've got going for them any more."

For those not yet convinced, the 3c501 can only do one thing at a time – while you are removing one packet from the single-packet buffer it cannot receive another packet, nor can it receive a packet while loading a transmit packet. This was fine for a network between two 8088-based computers where processing each packet and replying took 10's of msecs, but modern networks send back-to-back packets for almost every transaction.

Donald writes: 'The driver is now in the std. kernel, but under the following conditions: This is unsupported code. I know the usual copyright says all the code is unsupported, but this is _really_ unsupported. I DON'T want to see bug reports, and I'll accept bug fixes only if I'm in a good mood that day.

I don't want to be flamed later for putting out bad software. I don't know all all of the 3c501 bugs, and I know this driver only handles a few that I've been able to figure out. It has taken a long intense effort just to get the driver working this well.'

AutoIRQ works, DMA isn't used, the autoprobe only looks at `0x280` and `0x300`, and the debug level is set with the third boot-time argument.

Once again, the use of a 3c501 is *strongly discouraged*! Even more so with a IP multicast kernel, as you will grind to a halt while listening to *all* multicast packets. See the comments at the top of the source code for more details.

3.1.2 3c503, 3c503/16

Status – *Supported*

3Com shared-memory ethercards. They also have a programmed I/O mode that doesn't use the 8390 facilities (their engineers found too many bugs!) It should be about the same speed as the same bus width WD80x3, Unless you are a light user, spend the extra money and get the 16 bit model, as the price difference isn't significant. The 3c503 does not have "EEPROM setup", so the diagnostic/setup program isn't needed before running the card with Linux. The shared memory address of the 3c503 is set using jumpers that are shared with the boot PROM address. This is confusing to people familiar with other ISA cards, where you always leave the jumper set to "disable" unless you have a boot PROM.

The Linux 3c503 driver can also work with the 3c503 programmed-I/O mode, but this is slower and less reliable than shared memory mode. Also, programmed-I/O mode is not tested when updating the drivers, the deadman (deadcard?) check code may falsely timeout on some machines, and the probe for a 3c503 in programmed-I/O mode is turned off by default in some versions of the kernel. This was a panic reaction to the general device driver probe explosion; the 3c503 shared memory probe is a safe read from memory, rather than an extensive scan through I/O space. As of 0.99pl13, the kernel has an I/O port registrar that makes I/O space probes safer, and the programmed-I/O 3c503 probe has been re-enabled. You still shouldn't use the programmed-I/O mode though, unless you need it for MS-DOS compatibility.

The 3c503's IRQ line is set in software, with no hints from an EEPROM. Unlike the MS-DOS drivers, the Linux driver has capability to autoIRQ: it uses the first available IRQ line in {5,2/9,3,4}, selected each time the card is ifconfig'ed. (Older driver versions selected the IRQ at boot time.) The ioctl() call in 'ifconfig' will return EAGAIN if no IRQ line is available at that time.

Some common problems that people have with the 503 are discussed in 9.5.

3.1.3 3c505

Status – *Semi-Supported*

This is a driver that was written by Craig Southeren `geoffw@extro.ucc.su.oz.au`. These cards also use the i82586 chip. I don't think there are that many of these cards about. It is included in the standard kernel, but it is classed as an alpha driver. See 9.1 for important information on using alpha-test ethernet drivers with Linux.

There is also the file `/usr/src/linux/drivers/net/README.3c505` that you should read if you are going to use one of these cards. It contains various options that you can enable/disable. Technical information is available in 7.6.

3.1.4 3c507

Status – *Semi-Supported*

This card uses one of the Intel chips, and the development of the driver is closely related to the development of the Intel Ether Express driver. The driver is included in the standard kernel release, but as an alpha driver.

See 9.1 for important information on using alpha-test ethernet drivers with Linux. Technical information is available in 7.6.

3.1.5 3c509 / 3c509B

Status – *Supported*

It's fairly inexpensive and has excellent performance for a non-bus-master design. The drawbacks are that the original 3c509 _requires_ very low interrupt latency. The 3c509B shouldn't suffer from the same problem, due to having a larger buffer. (See below.)

Note that the ISA card detection uses a different method than most cards. Basically, you ask the cards to respond by sending data to an ID_PORT (port `0x100`). Note that if you have some other strange ISA card using an I/O range that includes the ID_PORT of the 3c509, it will probably not get detected. Note that you can change the ID_PORT to `0x110` or `0x120` or... in `3c509.c` if you have a conflicting ISA card, and the 3c509 will still be happy. Also note that this detection method means that it is difficult to predict which card will get detected first in a multiple ISA 3c509 configuration. The card with the lowest hardware ethernet address will end up being `eth0`. This shouldn't matter to anyone, except for those people who want to assign a 6 byte hardware address to a particular interface.

A working 3c509 driver was first included as an alpha-test version in the 0.99pl13 kernel sources. It is now in the standard kernel.

The original 3c509 has a tiny Rx buffer (2kB), causing the driver to occasionally drop a packet if interrupts are masked for too long. To minimize this problem, you can try unmasking interrupts during IDE disk transfers (see **`hdparm(8)`**) and/or increasing your ISA bus speed so IDE transfers finish sooner. (Note that the driver could be completely rewritten to use predictive interrupts, but performance re-writes of working drivers are low priority unless there is some particular incentive or need.)

The newer model 3c509B has 8kB on board, and the driver can set 4, 5 or 6kB for an Rx buffer. This setting can also be stored on the EEPROM. This should alleviate the above problem with the original 3c509. At this point in time, the Linux driver is not aware of this, and treats the 3c509B as an older 3c509.

Apparently, 3c509B users may have to use the supplied DOS utility to disable the *plug and play* support, and to set the output media to what they require.

Cameron Spitzer writes: "Beware that if you put a '509 in EISA addressing mode by mistake and save that in the EEPROM, you'll have to use an EISA machine or the infamous Test Via to get it back to normal, and it will conflict at IO location 0 which may hang your ISA machine. I believe this problem is corrected in the 3C509B version of the board."

3.1.6 3c523

Status – *Not Supported*

This MCA bus card uses the i82586, and now that people are actually running Linux on MCA machines, someone may wish to try and recycle parts of the 3c507 driver into a driver for this card.

3.1.7 3c527

Status – *Not Supported*

Yes, another MCA card. No, not too much interest in it. Better chances with the 3c523 or the 3c529.

3.1.8 3c529

Status – *Not Supported*

This card actually uses the same chipset as the 3c509. Donald actually put hooks into the 3c509 driver to check for MCA cards after probing for EISA cards, and before probing for ISA cards. But it hasn't evolved much further than that. Donald writes:

"I don't have access to a MCA machine (nor do I fully understand the probing code) so I never wrote the `mca_adaptor_select_mode()` or `mca_adaptor_id()` routines. If you can find a way to get the adaptor I/O address that assigned at boot time, you can just hard-wire that in place of the commented-out probe. Be sure to keep the code that reads the IRQ, if_port, and ethernet address."

3.1.9 3c579

Status – *Supported*

The EISA version of the 509. The current EISA version uses the same 16 bit wide chip rather than a 32 bit interface, so the performance increase isn't stunning. The EISA probe code was added to 3c509.c for 0.99pl14. We would be interested in hearing progress reports from any 3c579 users. (Read the above 3c509 section for info on the driver.)

Cameron Spitzer writes: "The 3C579 (Etherlink III EISA) should be configured as an EISA card. The IO Base Address (window 0 register 6 bits 4:0) should be 1f, which selects EISA addressing mode. Logic outside the ASIC decodes the IO address s000, where s is the slot number. I don't think it was documented real well. Except for its IO Base Address, the '579 should behave EXACTLY like the'509 (EL3 ISA), and if it doesn't, I want to hear about it (at my work address)."

3.1.10 3c589 / 3c589B

Status – *Semi-Supported*

Many people have been using this PCMCIA card for quite some time now. Note that support for it is not (at present) included in the default kernel source tree. Note that you will need a supported PCMCIA controller chipset. There are drivers available on Donald's ftp site:

```
cesdis.gsfc.nasa.gov:/pub/linux/pcmcia/README.3c589
cesdis.gsfc.nasa.gov:/pub/linux/pcmcia/3c589.c
cesdis.gsfc.nasa.gov:/pub/linux/pcmcia/dbether.c
```

Or for those that are *net-surfing* you can try:

Don's PCMCIA Stuff (`http://cesdis.gsfc.nasa.gov/linux/pcmcia.html`)

You will still need a PCMCIA socket enabler as well.

See 8.3 for more info on PCMCIA chipsets, socket enablers, etc.

The "B" in the name means the same here as it does for the 3c509 case.

3.2 Accton

3.2.1 Accton MPX

Status – *Supported*

Don't let the name fool you. This is still supposed to be a NE2000 compatible card. The MPX is supposed to stand for MultiPacket Accelerator, which, according to Accton, increases throughput substantially. But if you are already sending back-to-back packets, how can you get any faster...

3.2.2 Accton EN2212 PCMCIA Card

Status – *Semi-Supported*

David Hinds has been working on a driver for this card, and you are best to check the latest release of his PCMCIA package to see what the present status is.

3.3 Allied Telesis

3.3.1 AT1500

Status –*Supported*

These are a series of low-cost ethercards using the 79C960 version of the AMD LANCE. These are bus-master cards, and thus probably the fastest ISA bus ethercards available (although the 3c509 has lower latency thanks to predictive interrupts).

DMA selection and chip numbering information can be found in 3.4.1.

More technical information on AMD LANCE based Ethernet cards can be found in 7.8.

3.3.2 AT1700

Status – *Supported*

The Allied Telesis AT1700 series ethercards are based on the Fujitsu MB86965. This chip uses a programmed I/O interface, and a pair of fixed-size transmit buffers. This allows small groups of packets to sent be sent back-to-back, with a short pause while switching buffers.

A unique feature is the ability to drive 150ohm STP (Shielded Twisted Pair) cable commonly installed for Token Ring, in addition to 10baseT 100ohm UTP (unshielded twisted pair).

The Fujitsu chip used on the AT1700 has a design flaw: it can only be fully reset by doing a power cycle of the machine. Pressing the reset button doesn't reset the bus interface. This wouldn't be so bad, except that it can only be reliably detected when it has been freshly reset. The solution/work-around is to power-cycle the machine if the kernel has a problem detecting the AT1700.

Some production runs of the AT1700 had another problem: they are permanently wired to DMA channel 5. This is undocumented, there are no jumpers to disable the "feature", and no driver dares use the DMA capability because of compatibility problems. No device driver will be written using DMA if installing a second card into the machine breaks both, and the only way to disable the DMA is with a knife.

The at1700 driver is included in the standard kernel source tree.

3.4 AMD / Advanced Micro Devices

3.4.1 AMD LANCE (7990, 79C960, PCnet-ISA)

Status – *Supported*

There really is no AMD ethernet card. You are probably reading this because the only markings you could find on your card said AMD and the above number. The 7990 is the original 'LANCE' chip, but most stuff (including this document) refer to all these similar chips as 'LANCE' chips. (...incorrectly, I might add.)

These above numbers refer to chips from AMD that are the heart of many ethernet cards. For example, the Allied Telesis AT1500 (see 3.3.1) the NE1500/2100 (see 3.22.2) and the Boca-VLB/PCI cards (see 3.10.1)

The 79C960 (a.k.a. PCnet-ISA) contains enhancements and bug fixes over the original 7990 LANCE design.

Chances are that the existing LANCE driver will work with all AMD LANCE based cards. (except perhaps the NI65XX - see 3.24.2 for more info on that one.) This driver should also work with NE1500 and NE2100 clones.

For the ISA bus master mode all structures used directly by the LANCE, the initialization block, Rx and Tx rings, and data buffers, must be accessible from the ISA bus, i.e. in the lower 16M of real memory. If more than 16MB of memory is installed, low-memory 'bounce-buffers' are used when needed.

The DMA channel can be set with the low bits of the otherwise-unused dev-¿mem_start value (a.k.a. PARAM_1). (see 10.1.1) If unset it is probed for by enabling each free DMA channel in turn and checking if initialization succeeds.

The HP-J2405A board is an exception: with this board it's easy to read the EEPROM-set values for the IRQ, and DMA.

See 7.8 for more info on these chips.

3.4.2 AMD 79C961 (PCnet-ISA+)

Status – *Supported*

This is the PCnet-ISA+ – an enhanced version of the 79C960. It has support for jumper-less configuration and Plug and Play. See the info in the above section.

3.4.3 AMD 79C965 (PCnet-32)

Status – *Supported*

This is the PCnet-32 – a 32 bit bus-master version of the original LANCE chip for VL-bus and local bus systems. Minor cleanups were added to the original lance driver around v1.1.50 to support these 32 bit versions of the LANCE chip. The main problem was that the current versions of the '965 and '970 chips have a minor bug. They clear the Rx buffer length field in the Rx ring when they are explicitly documented not to. Again, see the above info.

3.4.4 AMD 79C970 (PCnet-PCI)

Status – *Supported*

This is the PCnet-PCI – similar to the PCnet-32, but designed for PCI bus based systems. Again, see the above info. Donald has modified the LANCE driver to use the PCI BIOS structure that was introduced by Drew Eckhardt for the PCI-NCR SCSI driver. This means that you need to build a kernel with PCI BIOS support enabled.

3.4.5 AMD 79C974 (PCnet-SCSI)

Status – *Supported*

This is the PCnet-SCSI – which is basically treated like a '970 from an Ethernet point of view. A minor '974 specific fix was added to the 1.1.8x kernels, so get a 1.1.90 or newer kernel. Also see the above info. Don't ask if the SCSI half of the chip is supported – this is the *Ethernet-Howto*, not the SCSI-Howto.

3.5 Ansel Communications

3.5.1 AC3200 EISA

Status – *Semi-Supported*

This driver is included in the present kernel as an alpha test driver. Please see 9.1 in this document for important information regarding alpha drivers. If you use it, let Donald know how things work out, as not too many people have this card and feedback has been low.

3.6 Apricot

3.6.1 Apricot Xen-II On Board Ethernet

Status – *Supported*

This on board ethernet uses an i82596 bus-master chip. It can only be at i/o address `0x300`. The author of this driver is Mark Evans. By looking at the driver source, it appears that the IRQ is hardwired to 10.

Earlier versions of the driver had a tendency to think that anything living at `0x300` was an apricot NIC. Since then the hardware address is checked to avoid these false detections.

3.7 Arcnet

Status – *Semi-Supported*

With the very low cost and better performance of ethernet, chances are that most places will be giving away their Arcnet hardware for free, resulting in a lot of home systems with Arcnet.

An advantage of Arcnet is that all of the cards have identical interfaces, so one driver will work for everyone.

Recent interest in getting Arcnet going has picked up again and Avery Pennarun's alpha driver has been put into the default kernel sources for 1.1.80 and above. The arcnet driver uses 'arc0' as its name instead of the usual 'eth0' for ethernet devices. Bug reports and success stories can be mailed to:

`apenwarr@tourism.807-city.on.ca`

3.8 AT&T

Note that AT&T's StarLAN is an orphaned technology, like SynOptics LattisNet, and can't be used in a standard 10Base-T environment.

3.8.1 AT&T T7231 (LanPACER+)

Status – *Not Supported*

These StarLAN cards use an interface similar to the i82586 chip. At one point, Matthijs Melchior (`matthijs.n.melchior@att.com`) was playing with the 3c507 driver, and almost had something useable working. Haven't heard much since that.

3.9 AT-Lan-Tec / RealTek

3.9.1 AT-Lan-Tec / RealTek Pocket adaptor

Status – *Supported*

This is a generic, low-cost OEM pocket adaptor being sold by AT-Lan-Tec, and (likely) a number of other suppliers. A driver for it is included in the standard kernel. Note that there is substantial information contained in the driver source file 'atp.c'. BTW, the adaptor (AEP-100L) has both 10baseT and BNC connections! You can reach AT-Lan-Tec at 1-301-948-7070. Ask for the model that works with Linux, or ask for tech support.

In the Netherlands a compatible adaptor is sold under the name SHI-TEC PE-NET/CT, and sells for about $125. The vendor was Megasellers. They state that they do not sell to private persons, but this doesn't appear to be strictly adhered to. They are: Megasellers, Vianen, The Netherlands. They always advertise in Dutch computer magazines. Note that the newer model EPP-NET/CT appears to be significantly different than the PE-NET/CT, and will not work with the present driver. Hopefully someone will come up with the programming information and this will be fixed up.

In Germany, a similar adaptor comes as a no-brand-name product. Prolan 890b, no brand on the casing, only a roman II. Resellers can get a price of about $130, including a small wall transformer for the power.

The adaptor is 'normal size' for the product class, about 57mm wide, 22mm high tapering to 15mm high at the DB25 connector, and 105mm long (120mm including the BNC socket). It's switchable between the RJ45 and BNC jacks with a small slide switch positioned between the two: a very intuitive design.

Donald performed some power draw measurements, and determined that the average current draw was only about 100mA @ 5V. This power draw is low enough that you could buy or build a cable to take the 5V directly from the keyboard/mouse port available on many laptops. (Bonus points here for using a standardized power connector instead of a proprietary one.)

Note that the device name that you pass to `ifconfig` is *not* `eth0` but `atp0` for this device.

3.10 Boca Research

Yes, they make more than just multi-port serial cards. :-)

3.10.1 Boca BEN (PCI, VLB)

Status – *Supported*

These cards are based on AMD's PCnet chips, used in the AT1500 and the like. You can pick up a combo (10BaseT and 10Base2) PCI card for under $70 at the moment.

Supposedly Boca PCI cards can have trouble with Pentium systems that are operating faster than 66MHz. You may want to check with Boca on this one. Note that this is not a driver problem, as it hits DOS/Win/NT users as well. Any additional info on this as it develops would be appreciated.

More information can be found in 3.4.1.

More technical information on AMD LANCE based Ethernet cards can be found in 7.8.

3.11 Cabletron

Donald writes: 'Yes, another one of these companies that won't release its programming information. They waited for months before actually confirming that all their information was proprietary, deliberately wasting my time. Avoid their cards like the plague if you can. Also note that some people have phoned Cabletron, and have been told things like 'a D. Becker is working on a driver for linux' – making it sound like I work for them. This is NOT the case.'

If you feel like asking them why they don't want to release their low level programming info so that people can use their cards, write to support@ctron.com. Tell them that you are using Linux, and are disappointed that they don't support open systems. And no, the usual driver development kit they supply is useless. It is just a DOS object file that you are supposed to link against. Which you aren't allowed to even reverse engineer.

3.11.1 E10**, E10**-x, E20**, E20**-x

Status – *Semi-Supported*

These are NEx000 almost-clones that are reported to work with the standard NEx000 drivers, thanks to a ctron-specific check during the probe. If there are any problems, they are unlikely to be fixed, as the programming information is unavailable.

3.11.2 E2100

Status – *Semi-Supported*

Again, there is not much one can do when the programming information is proprietary. The E2100 is a poor design. Whenever it maps its shared memory in during a packet transfer, it maps it into the *whole 128K region!* That means you **can't** safely use another interrupt-driven shared memory device in that region, including another E2100. It will work most of the time, but every once in a while it will bite you. (Yes, this problem can be avoided by turning off interrupts while transferring packets, but that will almost certainly lose clock ticks.) Also, if you mis-program the board, or halt the machine at just the wrong moment, even the reset button won't bring it back. You will *have* to turn it off and *leave* it off for about 30 seconds.

Media selection is automatic, but you can override this with the low bits of the dev->mem_end parameter. See 10.1.1

Also, don't confuse the E2100 for a NE2100 clone. The E2100 is a shared memory NatSemi DP8390 design, roughly similar to a brain-damaged WD8013, whereas the NE2100 (and NE1500) use a bus-mastering AMD LANCE design.

There is an E2100 driver included in the standard kernel. However, seeing as programming info isn't available, don't expect bug-fixes. Don't use one unless you are already stuck with the card.

3.12 D-Link

Some people have had difficulty in finding vendors that carry D-link stuff. This should help.

```
(714) 455-1688  in the US
(081) 203-9900  in the UK
6196-643011     in Germany
(416) 828-0260  in Canada
(02) 916-1600   in Taiwan
```

3.12.1 DE-100, DE-200, DE-220-T

Status – *Supported*

The manual says that it is 100 % compatible with the NE2000. This is not true. You should call them and tell them you are using their card with Linux, and they should correct their documentation. Some pre-0.99pl12 driver versions may have trouble recognizing the DE2** series as 16 bit cards, and these cards are the most widely reported as having the spurious transfer address mismatch errors. Note that there are cards from Digital (DEC) that are also named DE100 and DE200, but the similarity stops there.

3.12.2 DE-530

Status – *Semi-Supported*

This appears to be a generic DEC 21040 PCI chip implementation, and will most likely work with the generic 21040 driver. However, nobody has verified this yet, and until that has happened, it will remain listed as *Semi-Supported*.

See 3.14.4 for more information on these cards, and the present driver situation.

3.12.3 DE-600

Status – *Supported*

Laptop users and other folk who might want a quick way to put their computer onto the ethernet may want to use this. The driver is included with the default kernel source tree. Bjorn Ekwall `bj0rn@blox.se` wrote the driver. Expect about 80kb/s transfer speed from this via the parallel port. You should read the README.DLINK file in the kernel source tree.

Note that the device name that you pass to `ifconfig` is *now* `eth0` and not the previously used `dl0`.

If your parallel port is *not* at the standard `0x378` then you will have to recompile. Bjorn writes: "Since the DE-620 driver tries to sqeeze the last microsecond from the loops, I made the irq and port address constants instead of variables. This makes for a usable speed, but it also means that you can't change these assignements from e.g. lilo; you _have_ to recompile..." Also note that some laptops implement the on-board parallel port at `0x3bc` which is where the parallel ports on monochrome cards were/are.

Supposedly, a no-name ethernet pocket adaptor marketed under the name 'PE-1200' is DE-600 compatible. It is available in Europe from:

```
SEMCON Handels Ges.m.b.h
Favoritenstrasse 20
A-1040 WIEN
Telephone: (+43) 222 50 41 708
Fax      : (+43) 222 50 41 706
```

3.12.4 DE-620

Status – *Supported*

Same as the DE-600, only with two output formats. Bjorn has written a driver for this model, for kernel versions 1.1 and above. See the above information on the DE-600.

3.12.5 DE-650

Status – *Semi-Supported*

Some people have been using this PCMCIA card for some time now with their notebooks. It is a basic 8390 design, much like a NE2000. The LinkSys PCMCIA card and the IC-Card Ethernet (available from Midwest Micro) are supposedly DE-650 clones as well. Note that at present, this driver is *not* part of the standard kernel, and so you will have to do some patching.

See 8.3 in this document, and if you can, have a look at:

Don's PCMCIA Stuff (`http://cesdis.gsfc.nasa.gov/linux/pcmcia.html`)

3.13 DFI

3.13.1 DFINET-300 and DFINET-400

Status – *Supported*

These cards are now detected (as of 0.99pl15) thanks to Eberhard Moenkeberg `emoenke@gwdg.de` who noted that they use 'DFI' in the first 3 bytes of the prom, instead of using `0x57` in bytes 14 and 15, which is what all the NE1000 and NE2000 cards use. (The 300 is an 8 bit pseudo NE1000 clone, and the 400 is a pseudo NE2000 clone.)

3.14 Digital / DEC

3.14.1 DEPCA, DE100, DE200/1/2, DE210, DE422

Status – *Supported*

As of linux v1.0, there is a driver included as standard for these cards. It was written by David C. Davies. There is documentation included in the source file 'depca.c', which includes info on how to use more than one of these cards in a machine. Note that the DE422 is an EISA card. These cards are all based on the AMD LANCE chip. See 3.4.1 for more info. A maximum of two of the ISA cards can be used, because they can only be set for `0x300` and `0x200` base I/O address. If you are intending to do this, please read the notes in the driver source file `depca.c` in the standard kernel source tree.

3.14.2 Digital EtherWorks 3 (DE203, DE204, DE205)

Status – *Supported*

Included into kernels v1.1.62 and above is this driver, also by David C. Davies of DEC. These cards use a proprietary chip from DEC, as opposed to the LANCE chip used in the earlier cards like the DE200. These cards support both shared memory or programmed I/O, although you take about a 50%performance hit if you use PIO mode. The shared memory size can be set to 2kB, 32kB or 64kB, but only 2 and 32 have been tested with this driver. David says that the performance is virtually identical between the 2kB and 32kB mode. There is more information (including using the driver as a loadable module) at the top of the driver file `ewrk3.c` and also in `README.ewrk3`. Both of these files come with the standard kernel distribution.

Other interesting notes are that it appears that David is/was working on this driver for the unreleased version of Linux for the DEC Alpha AXP. And the standard driver has a number of interesting ioctl() calls that can be used to get or clear packet statistics, read/write the EEPROM, change the hardware address, and the like. Hackers can see the source code for more info on that one.

David has also written a configuration utility for this card (along the lines of the DOS program `NICSETUP.EXE`) along with other tools. These can be found on `sunsite.unc.edu` in the directory `/pub/Linux/system/Network/management` – look for the file `ewrk3tools-X.XX.tar.gz`.

3.14.3 DE425 (EISA), DE434, DE435

Status – *Supported*

These cards are based on the 21040 chip mentioned below. Included into kernels v1.1.86 and above is this driver, also by David C. Davies of DEC. It sure is nice to have support from someone on the inside ;-) Have a read of the 21040 section for extra info.

Note that as of 1.1.91, David has added a compile time option that may allow non-DEC cards (such as the ZYNX cards) to work with this driver. Have a look at `README.de4x5` for details.

3.14.4 DEC 21040, 21140, Tulip

Status – *Supported*

The DEC 21040 is a bus-mastering single chip ethernet solution from Digital, similar to AMD's PCnet chip. The 21040 is specifically designed for the PCI bus architecture. SMC's new EtherPower PCI card uses this chip. The new 21140 recently announced is for supporting 100Base-? and is supposed to be able to work with drivers for the 21040 chip.

You have a choice of *two* drivers for cards based on this chip. There is the DE425 driver discussed above, and the generic 21040 driver that Donald has written.

To use David's `de4x5` driver with non-DEC cards, have a look at `README.de4x5` for details.

Donald is doing his generic 21040 driver development on a SMC EtherPower PCI card at the moment, and this driver is included in the standard kernel source as of 1.1.84. Note that this driver is still considered an *alpha* driver (see 9.1) at the moment, and should be treated as such. To use it, you will have to edit `arch/i386/config.in` and uncomment the line for `CONFIG_DEC_ELCP` support.

3.15 Farallon

Farallon sells EtherWave adaptors and transceivers. This device allows multiple 10baseT devices to be daisy-chained.

3.15.1 Farallon Etherwave

Status – *Supported*

This is reported to be a 3c509 clone that includes the EtherWave transceiver. People have used these successfully with Linux and the present 3c509 driver. They are too expensive for general use, but are a great option for special cases. Hublet prices start at $125, and Etherwave adds $75-$100 to the price of the board – worth it if you have pulled one wire too few, but not if you are two network drops short.

3.16 Hewlett Packard

The 272** cards use programmed I/O, similar to the NE*000 boards, but the data transfer port can be 'turned off' when you aren't accessing it, avoiding problems with autoprobing drivers.

Thanks to Glenn Talbott for helping clean up the confusion in this section regarding the version numbers of the HP hardware.

3.16.1 27245A

Status – *Supported*

8 Bit 8390 based 10BaseT, not recommended for all the 8 bit reasons. It was re-designed a couple years ago to be highly integrated which caused some changes in initialization timing which only affected testing programs, not LAN drivers. (The new card is not 'ready' as soon after switching into and out of loopback mode.)

3.16.2 HP PC Lan+ (27247A, 27247B, 27252A)

Status – *Supported*

The HP PC Lan+ is different to the standard HP PC Lan card. This driver was added to the list of drivers in the standard kernel at about v1.1.3X. Note that even though the driver is included, the entry in 'config.in' seems to have been omitted. If you want to use it, and it doesn't come up in 'config.in' then add the following line to 'config.in' under the 'HP PCLAN support' line:

```
bool 'HP PCLAN Plus support' CONFIG_HPLAN_PLUS n
```

Then run `make config;make dep;make zlilo` or whatever.

The 47B is a 16 Bit 8390 based 10BaseT w/AUI, and the 52A is a 16 Bit 8390 based ThinLAN w/AUI. These cards are high performers (3c509 speed) without the interrupt latency problems (32K onboard RAM for TX or RX packet buffering). They both offer LAN connector autosense, data I/O in I/O space (simpler) or memory mapped (faster), and soft configuration.

The 47A is the older model that existed before the 'B'. Two versions 27247-60001 or 27247-60002 have part numbers marked on the card. Functionally the same to the LAN driver, except bits in ROM to identify boards differ. -60002 has a jumper to allow operation in non-standard ISA busses (chipsets that expect IOCHRDY early.)

3.16.3 HP-J2405A

Status – *Supported*

These are lower priced, and slightly faster than the 27247B/27252A, but are missing some features, such as AUI, ThinLAN connectivity, and boot PROM socket. This is a fairly generic LANCE design, but a minor design decision makes it incompatible with a generic 'NE2100' driver. Special support for it (including reading the DMA channel from the board) is included thanks to information provided by HP's Glenn Talbott.

More technical information on LANCE based cards can be found in 7.8

3.16.4 HP-Vectra On Board Ethernet

Status – *Supported*

The HP-Vectra has an AMD PCnet chip on the motherboard. Earlier kernel versions would detect it as the HP-J2405A but that would fail, as the Vectra doesn't report the IRQ and DMA channel like the J2405A. Get a kernel newer than v1.1.53 to avoid this problem.

DMA selection and chip numbering information can be found in 3.4.1.

More technical information on LANCE based cards can be found in 7.8

3.17 IBM / International Business Machines

3.17.1 IBM Thinkpad 300

Status – *Supported*

This is compatible with the Intel based Zenith Z-note. See 3.29.1 for more info.

Supposedly this site has a comprehensive database of useful stuff for newer versions of the Thinkpad. I haven't checked it out myself yet.

Thinkpad-info (`http://peipa.essex.ac.uk/html/linux-thinkpad.html`)

For those without a WWW browser handy, try `peipa.essex.ac.uk:/pub/tp750/`

3.17.2 IBM Credit Card Adaptor for Ethernet

Status – *Semi-Supported*

People have been using this PCMCIA card with Linux as well. Similar points apply, those being that you need a supported PCMCIA chipset on your notebook, and that you will have to patch the PCMCIA support into the standard kernel.

See 8.3 in this document, and if you can, have a look at:

Don's PCMCIA Stuff (`http://cesdis.gsfc.nasa.gov/linux/pcmcia.html`)

3.18 Intel Ethernet Cards

3.18.1 Ether Express

Status – *Semi-Supported*

This card uses the intel i82586. (Surprise, huh?) The driver is in the standard release of the kernel, as an alpha driver. See 9.1 for important information on using alpha-test ethernet drivers with Linux.

The reason is that the driver works well with slow machines, but the i82586 occasionally hangs from the packet buffer contention that a fast machine can cause. One reported hack/fix is to change all of the outw() calls to outw_p(). Also, the driver is missing promiscuous and multicast modes. (See 7.9)

There is also the standard way of using the chip (read slower) that is described in the chip manual, and used in other i82586 drivers, but this would require a re-write of the entire driver.

There is some technical information available on the i82586 in 7.6 and also in the source code for the driver 'eexpress.c'. Don't be afraid to read it. ;-)

3.18.2 Ether Express PRO

Status – *Not-Supported*

This card uses the Intel 82595. If it is as ugly to use as the i82586, then don't count on anybody writing a driver.

3.19 LinkSys

3.19.1 LinkSys PCMCIA Adaptor

Status – *Semi-Supported*

This is supposed to be a re-badged DE-650. See the information on the DE-650 in 3.12.5.

3.20 Microdyne

3.20.1 Microdyne Exos 205T

Status – *Not-Supported*

Another i82586 based card. At one point, `dabn100@hermes.cam.ac.uk` had written a driver that "almost worked" that was based on the 3c507 code. More details as they are received...

3.21 Mylex

3.21.1 Mylex LNP101, LNP104

Status – *Semi-Supported*

These are PCI cards that are based on DEC's 21040 chip. The LNP104 uses the 21050 chip to deliver *four* independent 10BaseT ports. The standard LNP101 is selectable between 10BaseT, 10Base2 and 10Base5 output. The LNP101 card *should* work with the generic 21040 driver, but nobody has verified this yet. As for the LNP 104, well...

See the section on the 21040 chip (3.14.4) for more information.

Mylex can be reached at the following numbers, in case anyone wants to ask them anything.

```
MYLEX CORPORATION, Fremont
Sales:  800-77-MYLEX, (510) 796-6100
FAX:    (510) 745-8016.
```

3.22 Novell Ethernet, NExxxx and associated clones.

The prefix 'NE' came from Novell Ethernet. Novell followed the cheapest NatSemi databook design and sold the manufacturing rights (spun off?) Eagle, just to get reasonably-priced ethercards into the market. (The now ubiquitous NE2000 card.)

3.22.1 NE1000, NE2000

Status – *Supported*

The now-generic name for a bare-bones design around the NatSemi 8390. They use programmed I/O rather than shared memory, leading to easier installation but slightly lower performance and a few problems. Again, the savings of using an 8 bit NE1000 over the NE2000 are only warranted if you expect light use. Some recently introduced NE2000 clones use the National Semiconductor 'AT/LANTic' 83905 chip, which offers a shared memory mode similar to the 8013 and EEPROM or software configuration. Some problems can arise with poor clones. See 9.3, and 4.1 In general it is not a good idea to put a NE2000 clone at I/O address `0x300` because nearly *every* device driver probes there at boot. Some poor NE2000 clones don't take kindy to being prodded in the wrong areas, and will respond by locking your machine.

Donald has written a NE2000 diagnostic program, but it is still presently in alpha test. (ne2k) See 6.2 for more information.

3.22.2 NE1500, NE2100

Status – *Supported*

These cards use the original 7990 LANCE chip from AMD and are supported using the Linux lance driver.

Some earlier versions of the lance driver had problems with getting the IRQ line via autoIRQ from the original Novell/Eagle 7990 cards. Hopefully this is now fixed. If not, then specify the IRQ via LILO, and let us know that it still has problems.

DMA selection and chip numbering information can be found in 3.4.1.

More technical information on LANCE based cards can be found in 7.8

3.22.3 NE3200

Status – *Not Supported*

This card uses a lowly 8MHz 80186, and hence you are better off using a cheap NE2000 clone. Even if a driver was available, the NE2000 card would most likely be faster.

3.23 Pure Data

3.23.1 PDUC8028, PDI8023

Status – *Supported*

The PureData PDUC8028 and PDI8023 series of cards are reported to work, thanks to special probe code contributed by Mike Jagdis `jaggy@purplet.demon.co.uk`. The support is integrated with the WD driver.

3.24 Racal-Interlan

3.24.1 NI52**

Status – *Semi-Supported*

Michael Hipp has written a driver for this card. It is included in the standard kernel as an 'alpha' driver. Michael would like to hear feedback from users that have this card. See 9.1 for important information on using alpha-test ethernet drivers with Linux.

Michael says that "the internal sysbus seems to be slow. So we often lose packets because of overruns while receiving from a fast remote host."

This card also uses one of the Intel chips. See 7.6 for more technical information.

3.24.2 NI65**

Status – *Semi-Supported*

There is also a driver for the LANCE based NI6510, and it is also written by Michael Hipp. Again, it is also an 'alpha' driver. For some reason, this card is not compatible with the generic LANCE driver. See 9.1 for important information on using alpha-test ethernet drivers with Linux.

3.25 Sager

3.25.1 Sager NP943

Status – *Semi-Supported*

This is just a 3c501 clone, with a different S.A. PROM prefix. I assume it is equally as brain dead as the original 3c501 as well. Kernels 1.1.53 and up check for the NP943 i.d. and then just treat it as a 3c501 after that. See 3.1.1 for all the reasons as to why you really don't want to use one of these cards.

3.26 Schneider & Koch

3.26.1 SK G16

Status – *Supported*

This driver was included into the v1.1 kernels, and it was written by PJD Weichmann and SWS Bern. It appears that the SK G16 is similar to the NI6510, in that it is based on the first edition LANCE chip (the 7990). Once again, I have no idea as to why this card won't work with the generic LANCE driver.

3.27 Western Digital / SMC (Standard Microsystems Corp.)

The ethernet part of Western Digital has been bought by SMC. One common mistake people make is that the relatively new SMC Elite Ultra is the same as the older SMC Elite16 models – this is **not** the case.

Here is how to contact SMC (not that you should need to.)

> SMC / Standard Microsystems Corp., 80 Arkay Drive, Hauppage, New York, 11788, USA.

Technical Support via phone:

```
800-992-4762 (USA)
800-433-5345 (Canada)
516-435-6250 (Other Countries)
```

Literature requests:

```
800-SMC-4-YOU (USA)
800-833-4-SMC (Canada)
516-435-6255  (Other Countries)
```

Technical Support via E-mail:

```
techsupt@ccmail.west.smc.com
```

FTP Site:

```
ftp.smc.com
```

3.27.1 WD8003, SMC Elite

Status – *Supported*

These are the 8-bit versions of the card. The 8 bit 8003 is slightly less expensive, but only worth the savings for light use. Note that some of the non-EEPROM cards (clones with jumpers, or old *old* old wd8003 cards) have no way of reporting the IRQ line used. In this case, auto-irq is used, and if that fails, the driver silently assings IRQ 5. Information regarding what the jumpers on old non-EEPROM wd8003 cards do can be found in conjunction with the SMC setup/driver disks stored on `dorm.rutgers.edu` in the directory `/pub/novell/nic_drvs/`. Note that some of the newer SMC 'SuperDisk' programs will fail to detect the old EEPROM-less cards. The file `SMCDSK46.EXE` seems to be a good all-round choice. Also the jumper settings for old cards are in an ascii text file in the aforementioned archive. The latest (greatest?) version can be obtained from `ftp.smc.com`.

As these are basically the same as their 16 bit counterparts (WD8013 / SMC Elite16), you should see the next section for more information.

3.27.2 WD8013, SMC Elite16

Status – *Supported*

Over the years the design has added more registers and an EEPROM. Clones usually go by the '8013' name, and usually use a non-EEPROM (jumpered) design. This part of WD has been sold to SMC, so you'll usually see something like SMC/WD8013 or SMC Elite16 Plus (WD8013). Late model SMC cards will have two main PLCC chips on board; the SMC 83c690 and the SMC 83c694. The shared memory design makes the cards 10-20 % faster, especially with larger packets. More importantly, from the driver's point of view, it avoids a few bugs in the programmed-I/O mode of the 8390, allows safe multi-threaded access to the packet buffer, and it doesn't have a programmed-I/O data register that hangs your machine during warm-boot probes.

Non-EEPROM cards that can't just read the selected IRQ will attempt auto-irq, and if that fails, they will silently assign IRQ 10. (8 bit versions will assign IRQ 5)

Also see 4.2 and 9.4.

3.27.3 SMC Elite Ultra

Status – *Supported*

This ethercard is based on a new chip from SMC, with a few new features. While it has a mode that is similar to the older SMC ethercards, it's not compatible with the old WD80*3 drivers. However, in this mode it shares most of its code with the other 8390 drivers, while operating somewhat faster than a WD8013 clone.

Since part of the Ultra *looks like* an 8013, the Ultra probe is supposed to find an Ultra before the wd8013 probe has a chance to mistakenly identify it.

Std. as of 0.99pl14, and made possible by documentation and ethercard loan from Duke Kamstra. If you plan on using an Ultra with Linux send him a note of thanks to let him know that there are Linux users out there!

Donald mentioed that it is possible to write a separate driver for the Ultra's 'Altego' mode which allows chaining transmits at the cost of inefficient use of receive buffers, but that will probably not happen right away. Performance re-writes of working drivers are low priority unless there is some particular incentive or need.

Bus-Master SCSI host adaptor users take note: In the manual that ships with Interactive UNIX, it mentions that a bug in the SMC Ultra will cause data corruption with SCSI disks being run from an aha-154X host adaptor. This will probably bite aha-154X compatible cards, such as the BusLogic boards, and the AMI-FastDisk SCSI host adaptors as well.

Supposedly SMC has acknowledged the problem occurs with Interactive, and older Windows NT drivers. It is supposed to be a hardware conflict that can be worked around in the driver design. More on this as it develops.

Some Linux users with an Ultra + aha-154X compatible cards have experienced data corruption, while others have not. Donald tried this combination himself, and wasn't able to reproduce the problem. You have been warned.

3.27.4 SMC 8416 (EtherEZ)

Status – *Supported*

This card uses SMC's 83c795 chip and supports the Plug 'n Play specification. It also has an *SMC Ultra* compatible mode, which allows it to be used with the Linux Ultra driver. In this compatibility mode, it uses shared memory instead of programmed i/o. Be sure to set your card for this compatibility mode.

Note that the EtherEZ specific checks were added to the SMC Ultra driver in 1.1.84, and hence earlier kernel versions will not handle these cards correctly.

3.27.5 SMC 8432 PCI (EtherPower)

Status – *Supported*

These cards appear to be a basic DEC 21040 implementation, i.e. one big chip and a couple of transceivers. Donald has used one of these cards for his development of the generic 21040 driver. Thanks to Duke Kamstra, once again, for supplying a card to do development on. See 3.14.4 for more details on using one of these cards, and the current status of the driver.

3.27.6 SMC 3008

Status – *Not Supported*

These 8 bit cards are based on the Fujitsu MB86950, which is an ancient version of the MB86965 used in the Linux at1700 driver. Russ says that you could probably hack up a driver by looking at the at1700.c code and his DOS packet driver for the Tiara card (tiara.asm)

3.27.7 SMC 3016

Status – *Not Supported*

These are 16bit i/o mapped 8390 cards, much similar to a generic NE2000 card. If you can get the specifications from SMC, then porting the NE2000 driver would probably be quite easy.

3.27.8 SMC 9000

Status – *Not Supported*

These cards are VLB cards based on the 91c92 chip. They are fairly expensive, and hence the demand for a driver is pretty low at the moment.

3.28 Xircom

Another group that won't release documentation. No cards supported. Don't look for any support in the future unless they release their programming information. And this is highly unlikely, as they *forbid* you from even reverse- engineering their drivers. If you are already stuck with one, see if you can trade it off on some DOS (l)user.

And if you just want to verify that this is the case, you can reach Xircom at 1-800-874-7875, 1-800-438-4526 or +1-818-878-7600. They used to advertise that their products "work with all network operating systems", but have since stopped. Wonder why...

3.28.1 PE1, PE2, PE3-10B*

Status – *Not Supported*

Not to get your hopes up, but if you have one of these parallel port adaptors, you may be able to use it in the DOS emulator with the Xircom-supplied DOS drivers. You will have to allow DOSEMU access to your parallel port, and will probably have to play with SIG (DOSEMU's Silly Interrupt Generator). I have no idea if this will work, but if you have any success with it, let me know, and I will include it here.

3.29 Zenith

3.29.1 Z-Note

Status – *Supported*

The built-in Z-Note network adaptor is based on the Intel i82593 using *two* DMA channels. There is an (alpha?) driver available in the present kernel version. As with all notebook and pocket adaptors, it is under the 'Pocket and portable adaptors' section when running `make config`. See 7.6 for more technical information. Also note that the IBM ThinkPad 300 is compatible with the Z-Note.

3.30 Zynx

3.30.1 Zynx (DEC 21040 based)

Status – *Supported*

You have a choice of *two* drivers for cards based on this chip. There is the DE425 driver written by David, and the generic 21040 driver that Donald has written.

Note that as of 1.1.91, David has added a compile time option that may allow non-DEC cards (such as the Zynx cards) to work with this driver. Have a look at `README.de4x5` for details.

See 3.14.4 for more information on these cards, and the present driver situation.

4 Clones of popular Ethernet cards.

Due to the popular design of some cards, different companies will make 'clones' or replicas of the original card. However, one must be careful, as some of these clones are not 100 % compatible, and can be troublesome. Some common problems with 'not-quite-clones' are noted in 9.

This section used to have a listing of a whole bunch of clones that were reported to work, but seeing as nearly *all* clones will work, it makes more sense to list the ones that don't work 100 % .

4.1 Poor NE2000 Clones

Here is a list of some of the NE-2000 clones that are known to have various problems. Most of them aren't fatal. In the case of the ones listed as 'bad clones' – this usually indicates that the cards don't have the two NE2000 identifier bytes. NEx000-clones have a Station Address PROM (SAPROM) in the packet buffer memory space. NE2000 clones have `0x57,0x57` in bytes `0x0e,0x0f` of the SAPROM, while other supposed NE2000 clones must be detected by their SA prefix.

Accton NE2000 – might not get detected at boot, see 9.3.

Aritsoft LANtastic AE-2 – OK, but has flawed error-reporting registers.

AT-LAN-TEC NE2000 – clone uses Winbond chip that traps SCSI drivers

ShineNet LCS-8634 – clone uses Winbond chip that traps SCSI drivers

Cabletron E10, E20**, E10**-x, E20**-x** – bad clones, but the driver checks for them. See 3.11.1.

D-Link Ethernet II – bad clones, but the driver checks for them. See 3.12.1.

DFI DFINET-300, DFINET-400 – bad clones, but the driver checks for them. See 3.13.1

4.2 Poor WD8013 Clones

I haven't heard of any bad clones of these cards, except perhaps for some chamelion-type cards that can be set to look like a ne2000 card or a wd8013 card. There is really no need to purchase one of these 'double-identity' cards anyway.

5 Cables, Coax, Twisted Pair

If you are starting a network from scratch, it's considerably less expensive to use thin ethernet, RG58 co-ax cable with BNC connectors, than old-fashioned thick ethernet, RG-5 cable with N connectors, or 10baseT, twisted pair telco-style cables with RJ-45 eight wire 'phone' connectors. See 2.4 for an introductory look at cables.

Also note that the FAQ from *comp.dcom.lans.ethernet* has a lot of useful information on cables and such. Look in `dorm.rutgers.edu` for the file `/pub/novell/info_and_docs/Ethernet.FAQ`

5.1 Thin Ethernet (thinnet)

Thin ethernet is the 'ether of choice'. The cable is inexpensive. If you are making your own cables solid-core RG58A is $0.27/m. and stranded RG58AU is $0.45/m. Twist-on BNC connectors are < $2 ea., and other misc. pieces are similarly inexpensive. It is essential that you properly terminate each end of the cable with 50 ohm terminators, so budget $2 ea. for a pair. It's also vital that your cable have no 'stubs' – the 'T' connectors must be attached directly to the ethercards. The only drawback is that if you have a big loop of machines connected together, and some bonehead breaks the loop by taking one cable off the side of his tee, the whole network goes down because it sees an infinite impedance (open circuit) instead of the required 50 ohm termination. Note that you can remove the tee piece from the card itself without killing the whole subnet, as long as you don't remove the cables from the tee itself. Of course this will disturb the machine that you pull the actual tee off of. 8-) And if you are doing a small network of two machines, you *still* need the tees and the 50 ohm terminators – you *can't* just cable them together!

5.2 Twisted Pair

Twisted pair networks require active hubs, which start around $200, and the raw cable cost can actually be higher than thinnet. They are usually sold using the claim that you can use your existing telephone wiring, but it's a rare installation where that turns out to be the case. The claim that you can upgrade to higher speeds is also suspect, as most proposed schemes use higher-grade (read $$) cable and more sophisticated termination ($$$) than you would likely install on speculation. New gizmos are floating around which allow you to daisy-chain machines together, and the like. For example, Farallon sells EtherWave adaptors and transceivers. This device allows multiple 10baseT devices to be daisy-chained. They also sell a 3c509 clone that includes the EtherWave transceiver. The drawback is that it's more expensive and less reliable than a cheap ($100-$150) mini-hub and another ethercard. You probably should either go for the hub approach or switch over to 10base2 thinnet.

On the other hand, hubs are rapidly dropping in price, all 100Mb/sec ethernet proposals use twisted pair, and most new business installations use twisted pair. (This is probably to avoid the problem with idiots messing with the BNC's as described above.)

Also, Russ Nelson adds that 'New installations should use Category 5 wiring. Anything else is a waste of your installer's time, as 100Base-whatever is going to require Cat 5.'

If you are only connecting two machines, it is possible to avoid using a hub, by swapping the Rx and Tx pairs (1-2 and 3-6).

If you hold the RJ-45 connector facing you (as if you were going to plug it into your mouth) with the lock tab on the top, then the pins are numbered 1 to 8 from left to right. The pin usage is as follows:

```
Pin Number              Assignment
----------              ----------
1                       Output Data (+)
2                       Output Data (-)
3                       Input Data (+)
4                       Reserved for Telephone use
5                       Reserved for Telephone use
6                       Input Data (-)
7                       Reserved for Telephone use
8                       Reserved for Telephone use
```

Some cards, like the wd8013 can sense reversed polarity, and will adjust accordingly. Also note that 3 and 6 **must** be a twisted pair. If you make 3-4 a twisted pair, and 5-6 the other twisted pair, your cable may work for lengths less than a metre, but will *fail miserably* for longer lengths.

Note that before 10BaseT was ratified as a standard, there existed other network formats using RJ-45 connectors, and the same wiring scheme as above. Examples are SynOptics's LattisNet, and AT&T's StarLAN. In some cases, (as with early 3C503 cards) you could set jumpers to get the card to talk to hubs of different types, but in most cases cards designed for these older types of networks will not work with standard 10BaseT networks/hubs. (Note that if the cards also have an AUI port, then there is no reason as to why you can't use that, combined with an AUI to 10BaseT transceiver.)

5.3 Thick Ethernet

Thick ethernet is mostly obsolete, and is usually used only to remain compatible with an existing implementation. You can stretch the rules and connect short spans of thick and thin ethernet together with a passive $3 N-to-BNC connector, and that's often the best solution to expanding an existing thicknet. A correct (but expensive) solution is to use a repeater in this case.

6 Software Configuration and Card Diagnostics

In most cases, if the configuration is done by software, and stored in an EEPROM, you will usually have to boot DOS, and use the supplied DOS program to set the cards IRQ, I/O, mem_addr and whatnot. Besides, hopefully it is something you will only be setting once. For those that don't have the DOS utility available, note that a fair number of NIC setup/driver disks (e.g. 3Com, SMC/WD and Allied Telesis NIC's) are available from `dorm.rutgers.edu` in the directory `/pub/novell/nic_drvs/` However, there are some cards for which Linux versions of the config utils exist, and they are listed here.

Also, Donald has written a few small card diagnostic programs that run under Linux. Most of these are a result of debugging tools that he has created while writing the various drivers. Don't expect fancy menu-driven interfaces. You will have to read the source code to use most of these. Even if your particular card doesn't have a corresponding diagnostic, you can still get lots of information just by typing `cat /proc/net/dev` – assuming that your card was at least detected at boot.

In either case, you will have to run most of these programs as root (to allow I/O to the ports) and you probably want to shut down the ethercard before doing so by typing `ifconfig eth0 down` (Note: replace `eth0` with `atp0` or whatever when appropriate.)

6.1 Configuration Programs for Ethernet Cards

For people with wd80x3 cards, there is the program `wdsetup` which can be found in `wdsetup-0.6a.tar.gz` on Linux ftp sites. I am not sure if it is being actively maintained or not, as it has not been updated for quite a while. If it works fine for you then great, if not, use the DOS version that you should have got with your card. If you don't have the DOS version, you will be glad to know that the SMC setup/driver disks are available at the `dorm.rutgers.edu` site mentioned above. Of course, you *have* to have an EEPROM card to use this utility. Old, *old* wd8003 cards, and some wd8013 clones use jumpers to set up the card instead.

The Digital EtherWorks 3 card can be configured in a similar fashion to the DOS program `NICSETUP.EXE`. David C. Davies wrote this and other tools for the EtherWorks 3 in conjunction with the driver. Look on `sunsite.unc.edu` in the directory `/pub/linux/system/Network/management` for the file that is named `ewrk3tools-X.XX.tar.gz`.

Some Nat Semi DP83905 implementations (such as the AT/LANTIC and the NE2000+) are software configurable. (Note that this card can also emulate a wd8013!) You can get the file `/pub/linux/setup/atlantic.c` from Donald's ftp server, `cesdis.gsfc.nasa.gov` to configure this card. Be careful when configuring NE2000+ cards, as you can give them bad setting values which will require you to open the case and switch a jumper to force it back to sane settings.

The 3Com Etherlink III family of cards (i.e. 3c5x9) can be configured by using another config utility from Donald. You can get the file `/pub/linux/setup/3c5x9setup.c` from Donald's ftp server, `cesdis.gsfc.nasa.gov` to configure these cards. (Note that the DOS 3c5x9B config utility may have more options pertaining to the new "B" series of the Etherlink III family.)

6.2 Diagnostic Programs for Ethernet Cards

Any of the diagnostic programs that Donald has written can be obtained from this URL.

Ethercard Diagnostics (`http://cesdis.gsfc.nasa.gov/pub/linux/diag/diagnostic.html`)

Allied Telesis AT1700 – look for the file `/pub/linux/diag/at1700.c` on `cesdis.gsfc.nasa.gov`.

Cabletron E21XX – look for the file `/pub/linux/diag/e21.c` on `cesdis.gsfc.nasa.gov`.

HP PCLAN+ – look for the file `/pub/linux/diag/hp+.c` on `cesdis.gsfc.nasa.gov`.

Intel EtherExpress – look for the file `/pub/linux/diag/eexpress.c` on `cesdis.gsfc.nasa.gov`.

NE2000 cards – look for the file `/pub/linux/diag/ne2k.c` on `cesdis.gsfc.nasa.gov`.

RealTek (ATP) Pocket adaptor – look for the file `/pub/linux/diag/atp-diag.c` on `cesdis.gsfc.nasa.gov`.

All Other Cards – try typing `cat /proc/net/dev` and see what useful info the kernel has on the card in question.

7 Technical Information

For those who want to play with the present drivers, or try to make up their own driver for a card that is presently unsupported, this information should be useful. If you do not fall into this category, then perhaps you will want to skip this section.

7.1 Probed Addresses

While trying to determine what ethernet card is there, the following addresses are autoprobed, assuming the type and specs of the card have not been set in the kernel. The file names below are in /usr/src/linux/drivers/net/

```
3c501.c         0x280, 0x300
3c503.c:        0x300, 0x310, 0x330, 0x350, 0x250, 0x280, 0x2a0, 0x2e0
3c505.c:        0x300, 0x280, 0x310
3c507.c:        0x300, 0x320, 0x340, 0x280
3c509.c:        Special ID Port probe
apricot.c       0x300
at1700.c:       0x300, 0x280, 0x380, 0x320, 0x340, 0x260, 0x2a0, 0x240
atp.c:          0x378, 0x278, 0x3bc
depca.c         0x300, 0x200
de600.c:        0x378
de620.c:        0x378
eexpress.c:     0x300, 0x270, 0x320, 0x340
hp.c:           0x300, 0x320, 0x340, 0x280, 0x2C0, 0x200, 0x240
hp-plus.c       0x200, 0x240, 0x280, 0x2C0, 0x300, 0x320, 0x340
lance.c:        0x300, 0x320, 0x340, 0x360
ne.c:           0x300, 0x280, 0x320, 0x340, 0x360
ni52.c          0x300, 0x280, 0x360, 0x320, 0x340
ni65.c          0x300, 0x320, 0x340, 0x360
smc-ultra.c:    0x200, 0x220, 0x240, 0x280, 0x300, 0x340, 0x380
wd.c:           0x300, 0x280, 0x380, 0x240
```

There are some NE2000 clone ethercards out there that are waiting black holes for autoprobe drivers. While many NE2000 clones are safe until they are enabled, some can't be reset to a safe mode. These dangerous ethercards will hang any I/O access to their 'dataports'. The typical dangerous locations are:

```
Ethercard jumpered base     Dangerous locations (base + 0x10 - 0x1f)
        0x300 *                         0x310-0x317
        0x320                           0x330-0x337
        0x340                           0x350-0x357
        0x360                           0x370-0x377
```

* The 0x300 location is the traditional place to put an ethercard, but it's also a popular place to put other devices (often SCSI controllers). The 0x320 location is often the next one chosen, but that's bad for for the AHA1542 driver probe. The 0x360 location is bad, because it conflicts with the parallel port at 0x378. If you have two IDE controllers, or two floppy controlers, then `0x360` is also a bad choice, as a NE2000 card will clobber them as well.

Note that kernels ¿ 1.1.7X keep a log of who uses which i/o ports, and will not let a driver use i/o ports registered by an earlier driver. This may result in probes silently failing. You can view who is using what i/o ports by typing `cat /proc/ioports` if you have the proc filesystem enabled.

To avoid these lurking ethercards, here are the things you can do:

- Probe for the device's BIOS in memory space. This is easy and always safe, but it only works for cards that always have BIOSes, like primary SCSI controllers.
- Avoid probing any of the above locations until you think you've located your device. The NE2000 clones have a reset range from <base>+0x18 to <base>+0x1f that will read as 0xff, so probe there first if possible. It's also safe to probe in the 8390 space at <base>+0x00 - <base>+0x0f, but that area will return quasi-random values
- If you must probe in the dangerous range, for instance if your target device has only a few port locations, first check that there isn't an NE2000 there. You can see how to do this by looking at the probe code in /usr/src/linux/net/inet/ne.c
- Use the 'reserve' boot time argument to protect volatile areas from being probed. See the information on using boot time arguments with LILO in 10.1.2

7.2 Skeleton / prototype driver

OK. So you have decided that you want to write a driver for the Foobar Ethernet card, as you have the programming information, and it hasn't been done yet. (...these are the two main requirements ;-) You can use the skeleton network driver that is provided with the Linux kernel source tree. It can be found in the file /usr/src/linux/drivers/net/skeleton.c as of 0.99pl15, and later.

It's also very useful to look at the Crynwr (nee Clarkson) driver for your target ethercard, if it's available. Russ Nelson `nelson@crynwr.com` has been actively updating and writing these, and he has been very helpful with his code reviews of the current Linux drivers.

7.3 Driver interface to the kernel

Here are some notes that may help when trying to figure out what the code in the driver segments is doing, or perhaps what it is supposed to be doing.

```
int ethif_init(struct device *dev)
{
```

```
        ...
            dev->send_packet = &ei_send_packet;
            dev->open = &ei_open;
            dev->stop = &ei_close;
            dev->hard_start_xmit = &ei_start_xmit;
            ...
    }

    int ethif_init(struct device *dev)
```

This function is put into the device structure in Space.c. It is called only at boot time, and returns '0' iff the ethercard 'dev' exists.

```
    static int ei_open(struct device *dev)
    static int ei_close(struct device *dev)
```

This routine opens and initializes the board in response to an socket ioctl() usually called by 'ifconfig'. It is commonly stuffed into the 'struct device' by ethif_init().

The inverse routine is ei_close(), which should shut down the ethercard, free the IRQs and DMA channels if the hardware permits, and turn off anything that will save power (like the transceiver).

```
    static int ei_start_xmit(struct sk_buff *skb, struct device *dev)
            dev->hard_start_xmit = &ei_start_xmit;
```

This routine puts packets to be transmitted into the hardware. It is usually stuffed into the 'struct device' by ethif_init().

When the hardware can't accept additional packets it should set the dev->tbusy flag. When additional room is available, usually during a transmit-complete interrupt, dev->tbusy should be cleared and the higher levels informed with mark_bh(INET_BH).

```
        if (dev_rint(buffer, length, is_skb ? IN_SKBUFF : 0, dev))
            stats->rx_dropped++;
```

A received packet is passed to the higher levels using dev_rint(). If the unadorned packet data in a memory buffer, dev_rint will copy it into a 'skbuff' for you. Otherwise a new skbuff should be kmalloc()ed, filled, and passed to dev_rint() with the IN_SKBUFF flag.

```
    int s=socket(AF_INET,SOCK_PACKET,htons(ETH_P_ALL));
```

Gives you a socket receving every protocol type. Do recvfrom() calls to it and it will fill the sockaddr with device type in sa_family and the device name in the sa_data array. I don't know who originally invented SOCK_PACKET for Linux (its been in for ages) but its superb stuff. You can use it to send stuff raw too (both only as root).

7.4 Interrupts and Linux

There are two kinds of interrupt handlers in Linux: fast ones and slow ones. You decide what kind you are installing by the flags you pass to irqaction(). The fast ones, such as the serial interrupt handler, run with _all_ interrupts disabled. The normal interrupt handlers, such as the one for ethercard drivers, runs with other interrupts enabled.

There is a two-level interrupt structure. The 'fast' part handles the device register, removes the packets, and perhaps sets a flag. After it is done, and interrupts are re-enabled, the slow part is run if the flag is set.

The flag between the two parts is set by:

```
mark_bh(INET_BH);
```

Usually this flag is set within dev_rint() during a received-packet interrupt, and set directly by the device driver during a transmit-complete interrupt.

You might wonder why all interrupt handlers cannot run in 'normal mode' with other interrupts enabled. Ross Biro uses this scenario to illustrate the problem:

- You get a serial interrupt, and start processing it. The serial interrupt is now masked.
- You get a network interrupt, and you start transferring a maximum-sized 1500 byte packet from the card.
- Another character comes in, but this time the interrupts are masked!

The 'fast' interrupt structure solves this problem by allowing bounded-time interrupt handlers to run without the risk of leaving their interrupt lines masked by another interrupt request.

There is an additional distinction between fast and slow interrupt handlers – the arguments passed to the handler. A 'slow' handler is defined as

```
static void
handle_interrupt(int reg_ptr)
{
    int irq = -(((struct pt_regs *)reg_ptr)->orig_eax+2);
    struct device *dev = irq2dev_map[irq];
...
```

While a fast handler gets the interrupt number directly

```
static void
handle_fast_interrupt(int irq)
{
...
```

A final aspect of network performance is latency. The only board that really addresses this is the 3c509, which allows a predictive interrupt to be posted. It provides an interrupt response timer so that the driver can fine-tune how early an interrupt is generated.

Alan Cox has some advice for anyone wanting to write drivers that are to be used with 0.99pl14 kernels and newer. He says:

'Any driver intended for 0.99pl14 should use the new alloc_skb() and kfree_skbmem() functions rather than using kmalloc() to obtain a sk_buff. The new 0.99pl14 skeleton does this correctly. For drivers wishing to remain compatible with both sets the define 'HAVE_ALLOC_SKB' indicates these functions must be used.

In essence replace

```
skb=(struct sk_buff *)kmalloc(size)
```

with

```
skb=alloc_skb(size)
```

and

```
kfree_s(skb,size)
```

with

```
kfree_skbmem(skb,size) /* Only sk_buff memory though */
```

Any questions should I guess be directed to me (Alan Cox) since I made the change. This is a change to allow tracking of sk_buff's and sanity checks on buffers and stack behaviour. If a driver produces the message'File: ??? Line: ??? passed a non skb!' then it is probable the driver is not using the new sk_buff allocators.'

7.5 Programmed I/O vs. Shared Memory vs. DMA

Ethernet is 10Mbs. (Don't be pedantic, 3Mbs and 100Mbs don't count.) If you can already send and receive back-to-back packets, you just can't put more bits over the wire. Every modern ethercard can receive back-to-back packets. The Linux DP8390 drivers come pretty close to sending back-to-back packets (depending on the current interrupt latency) and the 3c509 and AT1500 hardware has no problem at all automatically sending back-to-back packets.

The ISA bus can do 5.3MB/sec (42Mb/sec), which sounds like more than enough. You can use that bandwidth in several ways:

7.5.1 Programmed I/O

Pro: Doesn't use any constrained system resources, just a few I/O registers, and has no 16M limit.

Con: Usually the slowest transfer rate, the CPU is waiting the whole time, and interleaved packet access is usually difficult to impossible.

7.5.2 Shared memory

Pro: Simple, faster than programmed I/O, and allows random access to packets.

Con: Uses up memory space (a big one for DOS users, only a minor issue under Linux), and it still ties up the CPU.

7.5.3 Slave (normal) Direct Memory Access

Pro: Frees up the CPU during the actual data transfer.

Con: Checking boundary conditions, allocating contiguous buffers, and programming the DMA registers makes it the slowest of all techniques. It also uses up a scarce DMA channel, and requires aligned low memory buffers.

7.5.4 Master Direct Memory Access (bus-master)

Pro: Frees up the CPU during the data transfer, can string together buffers, can require little or no CPU time lost on the ISA bus.

Con: Requires low-memory buffers and a DMA channel. Any bus-master will have problems with other bus-masters that are bus-hogs, such as some primitive SCSI adaptors. A few badly-designed motherboard chipsets have problems with bus-masters. And a reason for not using *any* type of DMA device is using a Cyrix 486 processor designed for plug-in replacement of a 386: these processors must flush their cache with each DMA cycle. (This includes the Cx486DLC, Ti486DLC, Cx486SLC, Ti486SLC, etc.)

7.6 Programming the Intel chips (i82586 and i82593)

These chips are used on a number of cards, namely the 3c507 ('86), the Intel EtherExpress 16 ('86), Microdyne's exos205t ('86), the Z-Note ('93), and the Racal-Interlan ni5210 ('86).

Russ Nelson writes: 'Most boards based on the 82586 can reuse quite a bit of their code. More, in fact, than the 8390-based adapters. There are only three differences between them:

- The code to get the Ethernet address,
- The code to trigger CA on the 82586, and
- The code to reset the 82586.

The Intel EtherExpress 16 is an exception, as it I/O maps the 82586. Yes, I/O maps it. Fairly clunky, but it works.

Garrett Wollman did an AT&T driver for BSD that uses the BSD copyright. The latest version I have (Sep '92) only uses a single transmit buffer. You can and should do better than this if you've got the memory. The AT&T and 3c507 adapters do; the ni5210 doesn't.

The people at Intel gave me a very big clue on how you queue up multiple transmit packets. You set up a list of NOP-> XMIT-> NOP-> XMIT-> NOP-> XMIT-> beginning) blocks, then you set the 'next' pointer of all the NOP blocks to themselves. Now you start the command unit on this chain. It continually processes the first NOP block. To transmit a packet, you stuff it into the next transmit block, then point the NOP to it. To transmit the next packet, you stuff the next transmit block and point the previous NOP to *it*. In this way, you don't have to wait for the previous transmit to finish, you can queue up multiple packets without any ambiguity as to whether it got accepted, and you can avoid the command unit start-up delay.'

7.7 Technical information from 3Com

```
From: Cameron Spitzer 764-6339 {$<$}camerons@nad.3com.com {$>$} Subject: getting 3Com Adapter manuals
        Date: Mon, 27 Sep 1993 21:17:07 +0200
```

Since this is becoming a FAQ, I'm going to tread the thin ice of No Commercial Use and answer it here.

3Com's Ethernet Adapters are documented for driver writers in our 'Technical References' (TRs). These manuals describe the programmer interfaces to the boards but they don't talk about the diagnostics, installation programs, etc that end users can see.

The Network Adapter Division marketing department has the TRs to give away. To keep this program efficient, we centralized it in a thing called 'CardFacts.' CardFacts is an automated phone system. You call it with a touch-tone phone and it faxes you stuff. To get a TR, call CardFacts at 408-727-7021. Ask it for Developer's Order Form, document number 9070. Have your fax number ready when you call. Fill out the order form and fax it to 408-764-5004. Manuals are shipped by Federal Express 2nd Day Service.

If you don't have a fax and nobody you know has a fax, really and truly, *then* send mail to Terry_Murphy@3Mail.3Com.com and tell her about your problem. PLEASE use the fax thing if you possibly can.

After you get a manual, if you still can't figure out how to program the board, try our 'CardBoard' BBS at 1-800-876-3266, and if you can't do that, write Andy_Chan@3Mail.3com.com and ask him for alternatives. If you have a real stumper that nobody has figured out yet, the fellow who needs to know about it is Steve_Lebus@3Mail.3com.com.

There are people here who think we are too free with the manuals, and they are looking for evidence that the system is too expensive, or takes too much time and effort. That's why it's important to try to use CardFacts *before* you start calling and mailing the people I named here.

There are even people who think we should be like Diamond and Xircom, requiring tight 'partnership' with driver writers to prevent poorly performing drivers from getting written. So far, 3Com customers have been really good about this, and there's no problem with the level of requests we've been getting. We need your continued cooperation and restraint to keep it that way.

Cameron Spitzer, 408-764-6339 3Com NAD Santa Clara work: camerons@nad.3com.com home: cls@truffula.sj.ca.us

7.8 Notes on AMD PCnet / LANCE Based cards

The AMD LANCE (Local Area Network Controller for Ethernet) was the original offering, and has since been replaced by the 'PCnet-ISA' chip, otherwise known as the 79C960. A relatively new chip from AMD, the 79C960, is the heart of many new cards being released at present. Note that the name 'LANCE' has stuck, and some people will refer to the new chip by the old name. Dave Roberts of the Network Products Division of AMD was kind enough to contribute the following information regarding this chip:

'As for the architecture itself, AMD developed it originally and reduced it to a single chip – the PCnet(tm)-ISA – over a year ago. It's been selling like hotcakes ever since.

Functionally, it is equivalent to a NE1500. The register set is identical to the old LANCE with the 1500/2100 architecture additions. Older 1500/2100 drivers will work on the PCnet-ISA. The NE1500 and NE2100 architecture is basically the same. Initially Novell called it the 2100, but then tried to distinguish between coax and 10BASE-T cards. Anything that was 10BASE-T only was to be numbered in the 1500 range. That's the only difference.

Many companies offer PCnet-ISA based products, including HP, Racal-Datacom, Allied Telesis, Boca Research, Kingston Technology, etc. The cards are basically the same except that some manufacturers have added 'jumperless' features that allow the card to be configured in software. Most have not. AMD offers a standard design package for a card that uses the PCnet-ISA and many manufacturers use our design without change. What this means is that anybody who wants to write drivers for most PCnet-ISA based cards can just get the data-sheet from AMD. Call our literature distribution center at (800)222-9323 and ask for the Am79C960, PCnet-ISA data sheet. It's free.

A quick way to understand whether the card is a 'stock' card is to just look at it. If it's stock, it should just have one large chip on it, a crystal, a small IEEE address PROM, possibly a socket for a boot ROM, and a connector (1, 2, or 3, depending on the media options offered). Note that if it's a coax card, it will have some transceiver stuff built onto it as well, but that should be near the connector and away from the PCnet-ISA.'

There is also some info regarding the LANCE chip in the file lance.c which is included in the standard kernel.

A note to would-be card hackers is that different LANCE implementations do 'restart' in different ways. Some pick up where they left off in the ring, and others start right from the beginning of the ring, as if just initialised. This is a concern when setting the multicast list.

7.9 Multicast and Promiscuous Mode

Another one of the things Donald has worked on is implementing multicast and promiscuous mode hooks. All of the *released* (i.e. **not** ALPHA) ISA drivers now support promiscuous mode. There was a minor problem with 8390 based cards with capturing multicast packets, in that the promiscuous mode setting in

`8390.c` around line 574 should be `0x18` and not `0x10`. If you have an up to date kernel, this will already be fixed.

Donald writes: 'At first I was planning to do it while implementing either the /dev/* or DDI interface, but that's not really the correct way to do it. We should only enable multicast or promiscuous modes when something wants to look at the packets, and shut it down when that application is finished, neither of which is strongly related to when the hardware is opened or released.

I'll start by discussing promiscuous mode, which is conceptually easy to implement. For most hardware you only have to set a register bit, and from then on you get every packet on the wire. Well, it's almost that easy; for some hardware you have to shut the board (potentially dropping a few packet), reconfigure it, and then re-enable the ethercard. This is grungy and risky, but the alternative seems to be to have every application register before you open the ethercard at boot-time.

OK, so that's easy, so I'll move on something that's not quite so obvious: Multicast. It can be done two ways:

1. Use promiscuous mode, and a packet filter like the Berkeley packet filter (BPF). The BPF is a pattern matching stack language, where you write a program that picks out the addresses you are interested in. Its advantage is that it's very general and programmable. Its disadvantage is that there is no general way for the kernel to avoid turning on promiscuous mode and running every packet on the wire through every registered packet filter. See 7.10 for more info.
2. Using the built-in multicast filter that most etherchips have.

I guess I should list what a few ethercards/chips provide:

```
Chip/card  Promiscuous  Multicast filter
----------------------------------------
Seeq8001/3c501  Yes     Binary filter (1)
3Com/3c509      Yes     Binary filter (1)
8390            Yes     Autodin II six bit hash (2) (3)
LANCE           Yes     Autodin II six bit hash (2) (3)
i82586          Yes     Hidden Autodin II six bit hash (2) (4)
```

1. These cards claim to have a filter, but it's a simple yes/no 'accept all multicast packets', or 'accept no multicast packets'.
2. AUTODIN II is the standard ethernet CRC (checksum) polynomial. In this scheme multicast addresses are hashed and looked up in a hash table. If the corresponding bit is enabled, this packet is accepted. Ethernet packets are laid out so that the hardware to do this is trivial – you just latch six (usually) bits from the CRC circuit (needed anyway for error checking) after the first six octets (the destination address), and use them as an index into the hash table (six bits – a 64-bit table).
3. These chips use the six bit hash, and must have the table computed and loaded by the host. This means the kernel must include the CRC code.

4. The 82586 uses the six bit hash internally, but it computes the hash table itself from a list of multicast addresses to accept.

Note that none of these chips do perfect filtering, and we still need a middle-level module to do the final filtering. Also note that in every case we must keep a complete list of accepted multicast addresses to recompute the hash table when it changes.

My first pass at device-level support is detailed in the new outline driver skeleton.c

It looks like the following:

```
#ifdef HAVE_MULTICAST
static void set_multicast_list(struct device *dev, int num_addrs,
                 void *addrs);
#endif
.
.

ethercard_open() {
...
#ifdef HAVE_MULTICAST
        dev->set_multicast_list = &set_multicast_list;
#endif
...

#ifdef HAVE_MULTICAST
/* Set or clear the multicast filter for this adaptor.
   num_addrs -- -1      Promiscuous mode, receive all packets
   num_addrs -- 0       Normal mode, clear multicast list
   num_addrs > 0        Multicast mode, receive normal and
        MC packets, and do best-effort filtering.
 */
static void
set_multicast_list(struct device *dev, int num_addrs, void *addrs)
{
...
```

Any comments, criticism, etc. are welcome.'

7.10 The Berkeley Packet Filter (BPF)

The general idea of the developers is that the BPF functionality should not be provided by the kernel, but should be in a (hopefully little-used) compatibility library.

For those not in the know: BPF (the Berkeley Packet Filter) is an mechanism for specifying to the kernel networking layers what packets you are interested in. It's implemented as a specialized stack language interpreter built into a low level of the networking code. An application passes a program written in this language to the kernel, and the kernel runs the program on each incoming packet. If the kernel has multiple BPF applications, each program is run on each packet.

The problem is that it's difficult to deduce what kind of packets the application is really interested in from the packet filter program, so the general solution is to always run the filter. Imagine a program that registers a BPF program to pick up a low data-rate stream sent to a multicast address. Most ethernet cards have a hardware multicast address filter implemented as a 64 entry hash table that ignores most unwanted multicast packets, so the capability exists to make this a very inexpensive operation. But with the BFP the kernel must switch the interface to promiscuous mode, receive _all_ packets, and run them through this filter. This is work, BTW, that's very difficult to account back to the process requesting the packets.

8 Networking with a Laptop/Notebook Computer

There are currently only a few ways to put your laptop on a network. You can use the SLIP code (and run at serial line speeds); you can buy one of the few laptops that come with a NE2000-compatible ethercard; you can get a notebook with a supported PCMCIA slot built-in; you can get a laptop with a docking station and plug in an ISA ethercard; or you can use a parallel port Ethernet adapter such as the D-Link DE-600.

8.1 Using SLIP

This is the cheapest solution, but by far the most difficult. Also, you will not get very high transmission rates. Since SLIP is not really related to ethernet cards, it will not be discussed further here. See the NET-2 Howto.

8.2 Built in NE2000

This solution severely limits your laptop choices and is fairly expensive. Be sure to read the specifications carefully, as you may find that you will have to buy an additional non-standard transceiver to actually put the machine on a network. A good idea might be to boot the notebook with a kernel that has ne2000 support, and make sure it gets detected and works before you lay down your cash.

8.3 PCMCIA Support

As this area of Linux development is fairly young, I'd suggest that you join the LAPTOPS mailing channel. See 1.3 which describes how to join a mailing list channel.

Try and determine exactly what hardware you have (ie. card manufacturer, PCMCIA chip controller manufacturer) and then ask on the LAPTOPS channel. Regardless, don't expect things to be all that simple. Expect to have to fiddle around a bit, and patch kernels, etc. Maybe someday you will be able to type 'make config' 8-)

At present, the two PCMCIA chipsets that are supported are the Databook TCIC/2 and the intel i82365.

There is a number of programs on tsx-11.mit.edu in /pub/linux/packages/laptops/ that you may find useful. These range from PCMCIA Ethercard drivers to programs that communicate with the PCMCIA controller chip. Note that these drivers are usually tied to a specific PCMCIA chip (ie. the intel 82365 or the TCIC/2)

For NE2000 compatible cards, some people have had success with just configuring the card under DOS, and then booting linux from the DOS command prompt via `loadlin`.

For those that are *net-surfing* you can try:

Don's PCMCIA Stuff (`http://cesdis.gsfc.nasa.gov/linux/pcmcia.html`)

Anyway, the PCMCIA driver problem isn't specific to the Linux world. It's been a real disaster in the MS-DOS world. In that world people expect the hardware to work if they just follow the manual. They might not expect it to interoperate with any other hardware or software, or operate optimally, but they do expect that the software shipped with the product will function. Many PCMCIA adaptors don't even pass this test.

Things are looking up for Linux users that want PCMCIA support, as substantial progress is being made. Pioneering this effort is David Hinds. His latest PCMCIA support package can be obtained from `cb-iris.stanford.edu` in the directory `/pub/pcmcia/`. Look for a file like `pcmcia-cs-X.Y.Z.tgz` where X.Y.Z will be the latest version number. This is most likely uploaded to `tsx-11.mit.edu` as well.

Note that Donald's PCMCIA enabler works as a user-level process, and David Hinds' is a kernel-level solution. You may be best served by David's package as it is much more widely used.

8.4 ISA Ethercard in the Docking Station.

Docking stations for laptops typically cost about $250 and provide two full-size ISA slots, two serial and one parallel port. Most docking stations are powered off of the laptop's batteries, and a few allow adding extra batteries in the docking station if you use short ISA cards. You can add an inexpensive ethercard and enjoy full-speed ethernet performance.

8.5 Pocket / parallel port adaptors.

The 'pocket' ethernet adaptors may also fit your need. Until recently they actually costed more than a docking station and cheap ethercard, and most tie you down with a wall-brick power supply. At present, you can choose from the D-Link, or the RealTek adaptor. Most other companies, especially Xircom, (see 3.28) treat the programming information as a trade secret, so support will likely be slow in coming. (if ever!)

Note that the transfer speed will not be all that great (perhaps 100kB/s tops?) due to the limitations of the parallel port interface.

See 3.12.3 and 3.9.1 for supported pocket adaptors.

You can sometimes avoid the wall-brick with the adaptors by buying or making a cable that draws power from the laptop's keyboard port. (See 3.9.1)

9 Frequently Asked Questions

Here are some of the more frequently asked questions about using Linux with an Ethernet connection. Some of the more specific questions are sorted on a 'per manufacturer basis'. However, since this document is basically 'old' by the time you get it, any 'new' problems will not appear here instantly. For these, I suggest that you make efficient use of your newsreader. For example, nn users would type

```
nn -xX -s'3c'
```

to get all the news articles in your subscribed list that have '3c' in the subject. (ie. 3com, 3c509, 3c503, etc.) The moral: Read the man page for your newsreader.

9.1 Alpha Drivers – Getting and Using them

I heard that there is an alpha driver available for my card. Where can I get it?

The newest of the 'new' drivers can be found on Donald's new ftp site: `cesdis.gsfc.nasa.gov` in the `/pub/linux/` area. Things change here quite frequently, so just look around for it. There is still all the stuff on the old ftp site `ftp.super.org` in `/pub/linux`, but this is not being actively maintained, and hence will be of limited value to most people.

As of recent v1.1 kernels, the 'useable' alpha drivers have been included in the standard kernel source tree. When running `make config` you will be asked if you want to be offered ALPHA test drivers.

Now, if it really is an alpha, or pre-alpha driver, then please treat it as such. In other words, don't complain because you can't figure out what to do with it. If you can't figure out how to install it, then you probably shouldn't be testing it. Also, if it brings your machine down, don't complain. Instead, send us a well documented bug report, or even better, a patch!

People reading this while *net-surfing* may want to check out:

Don's Linux Home Page (`http://cesdis.gsfc.nasa.gov/pub/linux/linux.html`)

for the latest dirt on what is new and upcoming.

9.2 Using More than one Ethernet Card per Machine

What needs to be done so that Linux can run two ethernet cards?

The hooks for multiple ethercards are all there. However, note that only *one* ethercard is auto-probed for by default. This avoids a lot of possible boot time hangs caused by probing sensitive cards.

There are two ways that you can enable auto-probing for the second (and third, and...) card. The easiest method is to pass boot-time arguments to the kernel, which is usually done by LILO.Probing for the second card can be achieved by using a boot-time argument as simple as `ether=0,0,eth1`. In this case `eth0` and `eth1` will be assigned in the order that the cards are found at boot. Say if you want the card at `0x300` to be `eth0` and the card at `0x280` to be `eth1` then you could use

```
LILO: linux ether=5,0x300,eth0 ether=15,0x280,eth1
```

The `ether=` command accepts more than the IRQ + i/o + name shown above. Please have a look at 10.1 for the full syntax, card specific parameters, and LILO tips.

These boot time arguments can be made permanent so that you don't have to re-enter them every time. See the LILO configuration option '`append`' in the LILO manual.

The second way (not recommended) is to edit the file `Space.c` and replace the `0xffe0` entry for the i/o address with a zero. The `0xffe0` entry tells it not to probe for that device – replacing it with a zero will enable autoprobing for that device. If you *really* need more than four ethernet cards in one machine, then you can clone the `eth3` entry and change `eth3` to `eth4`.

Note that if you are intending to use Linux as a gateway between two networks, you will have to re-compile a kernel with IP forwarding enabled. Usually using an old AT/286 with something like the 'kbridge' software is a better solution.

If you are viewing this while *net-surfing*, you may wish to look at a mini-howto Donald has on his WWW site. Check out *Multiple Ethercards* (`http://cesdis.gsfc.nasa.gov/linux/misc/multicard.html`).

9.3 Problems with NE1000 / NE2000 cards (and clones)

Problem: NE*000 ethercard at `0x360` doesn't get detected anymore.

Reason: Recent kernels (¿ 1.1.7X) have more sanity checks with respect to overlapping i/o regions. Your NE2000 card is `0x20` wide in i/o space, which makes it hit the parallel port at `0x378`. Other devices that could be there are the second floppy controller (if equipped) at `0x370` and the secondary IDE controller at `0x376--0x377`. If the port(s) are already registered by another driver, the kernel will not let the probe happen.

Solution: Either move your card to an address like `0x280, 0x340, 0x320` or compile without parallel printer support.

Problem: Network 'goes away' every time I print something (NE2000)

Reason: Same problem as above, but you have an older kernel that doesn't check for overlapping i/o regions. Use the same fix as above, and get a new kernel while you are at it.

Problem: NE*000 ethercard probe at 0xNNN: 00 00 C5 ... not found. (invalid signature yy zz)

Reason: First off, do you have a NE1000 or NE2000 card at the addr. 0xNNN? And if so, does the hardware address reported look like a valid one? If so, then you have a poor NE*000 clone. All NE*000 clones are supposed to have the value `0x57` in bytes 14 and 15 of the SA PROM on the card. Yours doesn't – it has 'yy zz' instead.

Solution: The driver (/usr/src/linux/drivers/net/ne.c) has a "Hall of Shame" list at about line 42. This list is used to detect poor clones. For example, the DFI cards use 'DFI' in the first 3 bytes of the prom, instead of using 0x57 in bytes 14 and 15, like they are supposed to.

You can determine what the first 3 bytes of your card PROM are by adding a line like:

```
printk("PROM prefix: %#2x %#2x %#2x\n",SA_prom[0],SA_prom[1],SA_prom[2]);
```

into the driver, right after the error message you got above, and just before the "return ENXIO" at line 227.

Reboot with this change in place, and after the detection fails, you will get the three bytes from the PROM like the DFI example above. Then you can add your card to the bad_clone_list[] at about line 43. Say the above line printed out:

```
PROM prefix:  0x3F 0x2D 0x1C
```

after you rebooted. And say that the 8 bit version of your card was called the "FOO-1k" and the 16 bit version the "FOO-2k". Then you would add the following line to the bad_clone_list[]:

```
{"FOO-1k", "FOO-2k", {0x3F, 0x2D, 0x1C,}},
```

Note that the 2 name strings you add can be anything – they are just printed at boot, and not matched against anything on the card. You can also take out the "printk()" that you added above, if you want. It shouldn't hit that line anymore anyway. Then recompile once more, and your card should be detected.

Problem: Errors like `DMA address mismatch`

Is the chip a real NatSemi 8390? (DP8390, DP83901, DP83902 or DP83905)? If not, some clone chips don't correctly implement the transfer verification register. MS-DOS drivers never do error checking, so it doesn't matter to them.

Are most of the messages off by a factor of 2? If so: Are you using the NE2000 in a 16 bit slot? Is it jumpered to use only 8 bit transfers?

The Linux driver expects a NE2000 to be in a 16 bit slot. A NE1000 can be in either size slot. This problem can also occur with some clones, notably D-Link 16 bit cards, that don't have the correct ID bytes in the station address PROM.

Are you running the bus faster than 8Mhz? If you can change the speed (faster or slower), see if that makes a difference. Most NE2000 clones will run at 16MHz, but some may not. Changing speed can also mask a noisy bus.

What other devices are on the bus? If moving the devices around changes the reliability, then you have a bus noise problem – just what that error message was designed to detect. Congratulations, you've probably found the source of other problems as well.

Problem: The machine hangs during boot right after the '8390...' or 'WD....' message. Removing the NE2000 fixes the problem.

Solution: Change your NE2000 base address to `0x340`. Alternatively, you can use the device registrar implemented in 0.99pl13 and later kernels.

Reason: Your NE2000 clone isn't a good enough clone. An active NE2000 is a bottomless pit that will trap any driver autoprobing in its space. The other ethercard drivers take great pain to reset the NE2000 so that it's safe, but some clones cannot be reset. Clone chips to watch out for: Winbond 83C901. Changing the NE2000 to a less-popular address will move it out of the way of other autoprobes, allowing your machine to boot.

Problem: The machine hangs during the SCSI probe at boot.

Reason: It's the same problem as above, change the ethercard's address, or use the device registrar.

Problem: The machine hangs during the soundcard probe at boot.

Reason: No, that's really during the silent SCSI probe, and it's the same problem as above.

Problem: Errors like `eth0: DMAing conflict in ne_block_input`

This bug came from timer-based packet retransmissions. If you got a timer tick _during_ a ethercard RX interrupt, and timer tick tried to retransmit a timed-out packet, you could get a conflict. Because of the design of the NE2000 you would have the machine hang (exactly the same the NE2000-clone boot hangs).

Early versions of the driver disabled interrupts for a long time, and didn't have this problem. Later versions are fixed. (ie. kernels after 0.99p9 should be OK.)

Problem: NE2000 not detected at boot - no boot messages at all

Donald writes: 'A few people have reported a problem with detecting the Accton NE2000. This problem occurs only at boot-time, and the card is later detected at run-time by the identical code my (alpha-test) ne2k diagnostic program. Accton has been very responsive, but I still haven't tracked down what is going on. I've been unable to reproduce this problem with the Accton cards we purchased. If you are having this problem, please send me an immediate bug report. For that matter, if you have an Accton card send me a success report, including the type of the motherboard. I'm especially interested in finding out if this problem moves with the particular ethercard, or stays with the motherboard.'

Here are some things to try, as they have fixed it for some people:

- Change the bus speed, or just move the card to a different slot.
- Change the 'I/O recovery time' parameter in the BIOS chipset configuration.

9.4 Problems with WD80*3 cards

Problem: A WD80*3 is falsely detected. Removing the sound or MIDI card eliminates the 'detected' message.

Reason: Some MIDI ports happen to produce the same checksum as a WD ethercard.

Solution: Update your ethercard driver: new versions include an additional sanity check. If it is the midi chip at 0x388 that is getting detected as a WD living at 0x380, then you could also use:

```
LILO: linux reserve=0x380,8
```

Problem: You get messages such as the following with your 80*3:

```
eth0: bogus packet size, status = ........ kmalloc called with impossibly large
argument (65400) eth0: Couldn't allocate sk_buff of size 65400 eth0: receiver overrun
```

Reason: There is a shared memory problem.

Solution: If the problem is sporadic, you have hardware problems. Typical problems that are easy to fix are board conflicts, having cache or 'shadow ROM' enabled for that region, or running your bus faster than

8Mhz. There are also a surprising number of memory failures on ethernet cards, so run a diagnostic program if you have one for your ethercard.

If the problem is continual, and you have have to reboot to fix the problem, record the boot-time probe message and mail it to becker@cesdis.gsfc.nasa.gov - Take particular note of the shared memory location.

Problem: WD80*3 will not get detected at boot.

Reason: Earlier versions of the Mitsumi CD-ROM (mcd) driver probe at 0x300 will succeed if just about *anything* is that I/O location. This is bad news and needs to be a bit more robust. Once another driver registers that it 'owns' an I/O location, other drivers (incl. the wd80x3) are 'locked out' and can not probe that addr for a card.

Solution: Recompile a new kernel without any excess drivers that you aren't using, including the above mcd driver. Or try moving your ethercard to a new I/O addr. Valid I/O addr. for all the cards are listed in 7.1 You can also point the mcd driver off in another direction by a boot-time parameter (via LILO) such as:

```
mcd=0x200,12
```

Problem: Old wd8003 and/or jumper-settable wd8013 always get the IRQ wrong.

Reason: The old wd8003 cards and jumper-settable wd8013 clones don't have the EEPROM that the driver can read the IRQ setting from. If the driver can't read the IRQ, then it tries to auto-IRQ to find out what it is. And if auto-IRQ returns zero, then the driver just assigns IRQ 5 for an 8 bit card or IRQ 10 for a 16 bit card.

Solution: Avoid the auto-IRQ code, and tell the kernel what the IRQ that you have jumpered the card to is via a boot time argument. For example, if you are using IRQ 9, using the following should work.

```
LILO: linux ether=9,0,eth0
```

9.5 Problems with 3Com cards

Problem: The 3c503 picks IRQ N, but this is needed for some other device which needs IRQ N. (eg. CD ROM driver, modem, etc.) Can this be fixed without compiling this into the kernel?

Solution: The 3c503 driver probes for a free IRQ line in the order {5, 9/2, 3, 4}, and it should pick a line which isn't being used. Very old drivers used to pick the IRQ line at boot-time, and the current driver (0.99pl12 and newer) chooses when the card is open()/ifconfig'ed.

Alternately, you can fix the IRQ at boot by passing parameters via LILO. The following selects IRQ9, base location 0x300, <ignored value>, and if_port #1 (the external transceiver).

```
LILO: linux ether=9,0x300,0,1,eth0
```

The following selects IRQ3, probes for the base location, <ignored value>, and the default if_port #0 (the internal transceiver)

```
LILO: linux ether=3,0,0,0,eth0
```

Problem: 3c503: Configured interrupt number XX is out of range.

Reason: Whoever built your kernel fixed the ethercard IRQ at XX. The above is truly evil, and worse than that, it is not necessary. The 3c503 will autoIRQ when it gets ifconfig'ed, and pick one of IRQ{5, 2/9, 3, 4}.

Solution: Use LILO as described above, or rebuild the kernel, enabling autoIRQ by not specifying the IRQ line.

Problem: The supplied 3c503 drivers don't use the AUI (thicknet) port. How does one choose it over the default thinnet port?

Solution: The 3c503 AUI port can be selected at boot-time with 0.99pl12 and later. The selection is overloaded onto the low bit of the currently-unused dev->rmem_start variable, so a boot-time parameter of:

```
LILO: linux ether=0,0,0,1,eth0
```

should work. A boot line to force IRQ 5, port base 0x300, and use an external transceiver is:

```
LILO: linux ether=5,0x300,0,1,eth0
```

Also note that kernel revisions 1.00 to 1.03 had an interesting 'feature'. They would switch to the AUI port when the internal transciever failed. This is a problem, as it will *never* switch back if for example you momentarily disconnect the cable. Kernel versions 1.04 and newer only switch if the very first Tx attempt fails.

9.6 Problems with Hewlett Packard Cards

Problem: HP Vectra using built in AMD LANCE chip gets IRQ and DMA wrong.

Solution: The HP Vectra uses a different implementation to the standard HP-J2405A. The 'lance.c' driver used to *always* use the value in the setup register of an HP Lance implementation. In the Vectra case it's reading an invalid 0xff value. Kernel versions newer than about 1.1.50 now handle the Vectra in an appropriate fashion.

Problem: HP Card is not detected at boot, even though kernel was compiled with 'HP PCLAN support'.

Solution: You probably have a HP PCLAN+ – note the 'plus'. Support for the PCLAN+ was added to final versions of 1.1, but some of them didn't have the entry in 'config.in'. If you have the file hp-plus.c in /linux/drivers/net/ but no entry in config.in, then add the following line under the 'HP PCLAN support' line:

```
bool 'HP PCLAN Plus support' CONFIG_HPLAN_PLUS n
```

Kernels up tp 1.1.54 are missing the line in 'config.in' still. Do a 'make mrproper;make config;make dep;make zlilo' and you should be in business.

9.7 FAQs Not Specific to Any Card.

9.7.1 `ifconfig` reports the wrong i/o address for the card.

No it doesn't. You are just interpreting it incorrectly. This is *not* a bug, and the numbers reported are correct. It just happens that some 8390 based cards (wd80x3, smc-ultra, etc) have the actual 8390 chip living at an offset from the first assigned i/o port. Try `cd /usr/src/linux/drivers/net;grep NIC_OFFSET *.c|more` to see what is going on. This is the value stored in `dev->base_addr`, and is what `ifconfig` reports. If you want to see the full range of ports that your card uses, then try `cat /proc/ioports` which will give the numbers you expect.

9.7.2 Token Ring

Is there token ring support for Linux?

To support token ring requires more than only a writing a device driver, it also requires writing the source routing routines for token ring. It is the source routing that would be the most time comsuming to write.

Alan Cox adds: 'It will require (...) changes to the bottom socket layer to support 802.2 and 802.2 based TCP/IP. Don't expect anything soon.'

Peter De Schrijver has been spending some time on Token Ring lately, and has patches that are available for IBM ISA and MCA token ring cards. Don't expect miracles here, as he has just started on this as of 1.1.42. You can get the patch from:

`aix13ps2.cc.kuleuven.ac.be:/pub/Linux/TokenRing.patch-1.1.49.gz`

9.7.3 32 Bit / VLB / PCI Ethernet Cards

What is the selection for 32 bit ethernet cards?

There aren't many 32 bit ethercard device drivers because there aren't that many 32 bit ethercards.

There aren't many 32 bit ethercards out there because a 10Mbs network doesn't justify spending the 5x price increment for the 32 bit interface. See 7.5 as to why having an ethercard on an 8MHz ISA bus is really not a bottleneck.

This might change now that AMD has introduced the 32 bit PCnet-VLB and PCnet-PCI chips. The street price of the Boca PCnet-VLB board should be under $70 from a place like CMO (see Computer Shopper). See 3.10.1 for info on these cards.

See 3.4.3 for info on the 32 bit versions of the LANCE / PCnet-ISA chip.

The DEC 21040 PCI chip is another option (see 3.14.4) for power-users. The 21140 100Base-? chip could prove interesting as well, as it is supposedly driver compatible with the 21040. Should be good for uncovering any race-conditions, if nothing else...

9.7.4 FDDI

Is there FDDI support for Linux?

Donald writes: 'No, there is no Linux driver for any FDDI boards. I come from a place with supercomputers, so an external observer might think FDDI would be high on my list. But FDDI never delivered end-to-end throughput that would justify its cost, and it seems to be a nearly abandoned technology now that 100base{X,Anynet} seems imminent. (And yes, I know you can now get FDDI boards for <$1K. That seems to be a last-ditch effort to get some return on the development investment. Where is the next generation of FDDI going to come from?)'

9.7.5 Linking 10BaseT without a Hub

Can I link 10BaseT (RJ45) based systems together without a hub?

You can link 2 machines easily, but no more than that, without extra devices/gizmos. See 5.2 – it explains how to do it. And no, you can't hack together a hub just by crossing a few wires and stuff. It's pretty much impossible to do the collision signal right without duplicating a hub.

9.7.6 SIOCSFFLAGS: Try again

I get 'SIOCSFFLAGS: Try again' when I run 'ifconfig' – Huh?

Some other device has taken the IRQ that your ethercard is trying to use, and so the ethercard can't use the IRQ. You don't necessairly need to reboot to resolve this, as some devices only grab the IRQs when they need them and then release them when they are done. Examples are some sound cards, serial ports, floppy disk driver, etc. You can type `cat /proc/interrupts` to see which interrupts are presently *in use*. Most of the Linux ethercard drivers only grab the IRQ when they are opened for use via 'ifconfig'. If you can get the other device to 'let go' of the required IRQ line, then you should be able to 'Try again' with ifconfig.

9.7.7 Link UNSPEC and HW-addr of 00:00:00:00:00:00

When I run ifconfig with no arguments, it reports that LINK is UNSPEC (instead of 10Mbs Ethernet) and it also says that my hardware address is all zeros.

This is because people are running a newer version of the 'ifconfig' program than their kernel version. This new version of ifconfig is not able to report these properties when used in conjunction with an older kernel. You can either upgrade your kernel, 'downgrade' ifconfig, or simply ignore it. The kernel knows your hardware address, so it really doesn't matter if ifconfig can't read it.

9.7.8 Huge Number of RX and TX Errors

When I run ifconfig with no arguments, it reports that I have a huge error count in both rec'd and transmitted packets. It all seems to work ok – What is wrong?

Look again. It says `RX packets` *big number* **PAUSE** `errors 0` **PAUSE** `dropped 0` **PAUSE** `overrun 0`. And the same for the `TX` column. Hence the big numbers you are seeing are the total number of packets that your machine has rec'd and transmitted. If you still find it confusing, try typing `cat /proc/net/dev` instead.

9.7.9 Entries in /dev/ for Ethercards

I have /dev/eth0 as a link to /dev/xxx. Is this right?

Contrary to what you have heard, the files in /dev/* are not used. You can delete any `/dev/wd0, /dev/ne0` and similar entries.

9.7.10 Linux and "trailers"

Should I disable trailers when I 'ifconfig' my ethercard?

You can't disable trailers, and you shouldn't want to. 'Trailers' are a hack to avoid data copying in the networking layers. The idea was to use a trivial fixed-size header of size 'H', put the variable-size header info at the end of the packet, and allocate all packets 'H' bytes before the start of a page. While it was a good idea, it turned out to not work well in practice. If someone suggests the use of '-trailers', note that it is the equivalent of sacrificial goats blood. It won't do anything to solve the problem, but if problem fixes itself then someone can claim deep magical knowledge.

9.7.11 Non-existent Apricot NIC is detected

I get `eth0: Apricot 82596 at 0x300, 00 00 00 00 00 00 IRQ 10` and `apricot.c:v.0.02 19/05/94` when I boot, but I don't have an "Apricot". And then the card I do have isn't detected.

A few kernel releases had a version of the Apricot driver which only used a simple checksum to detect if an Apricot is present. This would mistakenly think that almost everything was an Apricot NIC. It really should look at the vendor prefix instead. However there is now a check to see if the hardware address is all zeros, so this shouldn't happen. Your choices are to move your card off of `0x300` (the only place the Apricot driver probes), or better yet, get a new kernel.

10 Miscellaneous.

Any other associated stuff that didn't fit in anywhere else gets dumped here. It may not be relevant, and it may not be of general interest but it is here anyway.

10.1 Passing Ethernet Arguments to the Kernel

Here are two generic kernel commands that can be passed to the kernel at boot time. This can be done with LILO, loadlin, or any other booting utility that accepts optional arguments.

For exmaple, if the command was 'blah' and it expected 3 arguments (say 123, 456, and 789) then, with LILO, you would use:

```
LILO: linux blah=123,456,789
```

Note: PCI cards have their i/o and IRQ assigned by the BIOS at boot. This means that any boot time arguments for a PCI card's IRQ or i/o ports are ignored.

10.1.1 The `ether` command

In its most generic form, it looks something like this:

```
ether=IRQ,BASE_ADDR,PARAM_1,PARAM_2,NAME
```

All arguments are optional. The first non-numeric argument is taken as the NAME.

IRQ: Obvious. An IRQ value of '0' (usually the default) means to autoIRQ. It's a historical accident that the IRQ setting is first rather than the base_addr – this will be fixed whenever something else changes.

BASE_ADDR: Also obvious. A value of '0' (usually the default) means to probe a card-type-specific address list for an ethercard.

PARAM_1: It was orginally used as an override value for the memory start for a shared-memory ethercard, like the WD80*3. Some drivers use the low four bits of this value to set the debug message level. 0 – default, 1-7 – level 1..7, (7 is maximum verbosity) 8 – level 0 (no messages). Also, the LANCE driver uses the low four bits of this value to select the DMA channel. Otherwise it uses auto-DMA.

PARAM_2: The 3c503 driver uses this to select between the internal and external transceivers. 0 – default/internal, 1 – AUI external. The Cabletron E21XX card also uses the low 4 bits of PARAM_2 to select the output media. Otherwise it detects automatically.

NAME: Selects the network device the values refer to. The standard kernel uses the names 'eth0', 'eth1', 'eth2' and 'eth3' for bus-attached ethercards, and 'atp0' for the parallel port 'pocket' ethernet adaptor. The arcnet driver uses 'arc0' as its name. The default setting is for a single ethercard to be probed for as 'eth0'. Multiple cards can only be enabled by explicitly setting up their base address using these LILO parameters. The 1.0 kernel has LANCE-based ethercards as a special case. LILO arguments are ignored, and LANCE cards are always assigned 'eth<n>' names starting at 'eth0'. Additional non-LANCE ethercards must be explicitly assigned to 'eth<n+1>', and the usual 'eth0' probe disabled with something like 'ether=0,-1,eth0'. (Yes, this is bug.)

10.1.2 The `reserve` command

This next lilo command is used just like 'ether=' above, ie. it is appended to the name of the boot select specified in lilo.conf

```
reserve=IO-base,extent{,IO-base,extent...}
```

In some machines it may be necessary to prevent device drivers from checking for devices (auto-probing) in a specific region. This may be because of poorly designed hardware that causes the boot to *freeze* (such as some ethercards), hardware that is mistakenly identified, hardware whose state is changed by an earlier probe, or merely hardware you don't want the kernel to initialize.

The `reserve` boot-time argument addresses this problem by specifying an I/O port region that shouldn't be probed. That region is reserved in the kernel's port registration table as if a device has already been found in that region. Note that this mechanism shouldn't be necessary on most machines. Only when there is a problem or special case would it be necessary to use this.

The I/O ports in the specified region are protected against device probes. This was put in to be used when some driver was hanging on a NE2000, or misidentifying some other device as its own. A correct device driver shouldn't probe a reserved region, unless another boot argument explicitly specifies that it do so. This implies that `reserve` will most often be used with some other boot argument. Hence if you specify a `reserve` region to protect a specific device, you must generally specify an explicit probe for that device. Most drivers ignore the port registration table if they are given an explicit address.

For example, the boot line

```
LILO: linux reserve=0x300,32 ether=0,0x300,eth0
```

keeps all device drivers except the ethercard drivers from probing 0x300-0x31f.

As usual with boot-time specifiers there is an 11 parameter limit, thus you can only specify 5 reserved regions per `reserve` keyword. Multiple `reserve` specifiers will work if you have an usually complicated request.

10.2 Using the Ethernet Drivers as Modules

At present, all the modules are put in the subdirectory `modules` in your Linux kernel source tree (usually in the form of symbolic links). To actually generate the modules, you have to type `make modules` after you have finished building the kernel proper. Earlier kernels built them automatically, which wasn't fair to those compiling on 4MB 386sx-16 machines.

If you have an 8390 based card, you will have to insert *two* modules, 8390.o and then the module for your card. You can find out if your card uses an 8390 chip by reading the above documentation for your card, or by just typing something like `grep 8390 my_card_name.c` in the `drivers/net/` directory. If `grep` finds anything, then your card has an 8390 (or compatible) chip.

Once you have figured this out, you can insert the module(s) by typing `insmod mod_name.o` as root. The command `lsmod` will show you what modules are loaded, and `rmmod` will remove them.

Once a module is inserted, then you can use it just like normal, and give `ifconfig` commands. If you set up your networking at boot, then make sure your `/etc/rc*` files run the `insmod` command(s) before getting to the `ifconfig` command.

Also note that a *busy* module can't be removed. That means that you will have to `ifconfig eth0 down` (shut down the ethernet card) before you can remove the modules. Also, if you use an 8390 based card, you will have to remove the card module before removing the 8390 module, as the 8390 module is used by the card module.

10.3 Contributors

Other people who have contributed (directly or indirectly) to the Ethernet-Howto are, in alphabetical order:

```
Ross Biro             <bir7@leland.stanford.edu>
Alan Cox              <iialan@www.linux.org.uk>
David C. Davies       <davies@wanton.enet.dec.com>
Bjorn Ekwall          <bj0rn@blox.se>
David Hinds           <dhinds@allegro.stanford.edu>
Michael Hipp          <mhipp@student.uni-tuebingen.de>
Mike Jagdis           <jaggy@purplet.demon.co.uk>
Duke Kamstra          <kamstra@ccmail.west.smc.com>
Russell Nelson        <nelson@crynwr.com>
Cameron Spitzer       <camerons@NAD.3Com.com>
Dave Roberts          <david.roberts@amd.com>
Glenn Talbott         <gt@hprnd.rose.hp.com>
```

Many thanks to the above people, and all the other unmentioned testers out there.

10.4 Closing

If you have found any glaring typos, or outdated info in this document, please let one of us know. It's getting big, and it is easy to overlook stuff.

Thanks,

Paul Gortmaker, Paul.Gortmaker@anu.edu.au

Donald J. Becker, becker@cesdis.gsfc.nasa.gov