Linux Ethernet HOWTO

Paul Gortmaker, Editor. v2.6, 4 October 1996

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 like `ifconfig` and `route`. See the NET2-Howto for that stuff.

Contents

1 Introduction **8**

1.1 New Versions of this Document . . . 9

1.2 Using the Ethernet-Howto . . . 9

1.3 HELP - It doesn't work! . . . 10

2 What card should I buy for Linux? **12**

2.1 So What Drivers are Stable? . . . 12

2.2 Eight bit vs 16 bit Cards . . . 12

2.3 32 Bit / VLB / PCI Ethernet Cards . . . 13

2.4 Available 100Mbs Cards and Drivers . . . 13

2.5 100VG versus 100BaseT . . . 14

2.6 Programmed I/O vs. Shared Memory vs. DMA . . . 14

2.6.1 Programmed I/O (e.g. NE2000, 3c509) . . . 14

2.6.2 Shared memory (e.g. WD80x3, SMC-Ultra, 3c503) . . . 15

2.6.3 Slave (normal) Direct Memory Access (e.g. none for Linux!) . . . 15

2.6.4 Bus Master Direct Memory Access (e.g. LANCE, DEC 21040) . . . 15

2.7 Type of cable that your card should support . . . 15

3 Frequently Asked Questions **16**

3.1 Alpha Drivers – Getting and Using them . . . 16

3.2 Using More than one Ethernet Card per Machine . . . 16

3.3 Poor NE2000 Clones . . . 17

3.4 Problems with NE1000 / NE2000 cards (and clones) . . . 18

3.5 Problems with SMC Ultra/EtherEZ and WD80*3 cards . . . 22

3.6 Problems with 3Com cards . . . 23

3.7 FAQs Not Specific to Any Card. 24
3.7.1 Ethercard is Not Detected at Boot. 24
3.7.2 `ifconfig` reports the wrong i/o address for the card. 24
3.7.3 Shared Memory ISA cards in PCI Machine dont work (0xffff) 24
3.7.4 NexGen machine gets 'mismatched read page pointers' errors. 25
3.7.5 Asynchronous Transfer Mode (ATM) Support 25
3.7.6 FDDI Support 25
3.7.7 Full Duplex Support 25
3.7.8 Ethernet Cards for Linux on Alpha/AXP PCI Boards 26
3.7.9 Linking 10BaseT without a Hub 26
3.7.10 SIOCSIFxxx: No such device 26
3.7.11 SIOCSFFLAGS: Try again 26
3.7.12 Link UNSPEC and HW-addr of 00:00:00:00:00:00 27
3.7.13 Huge Number of RX and TX Errors 27
3.7.14 Entries in `/dev/` for Ethercards 27
3.7.15 Linux and "trailers" 27
3.7.16 Access to the raw Ethernet Device 28

4 Performance Tips **28**

4.1 General Concepts 28
4.2 ISA Bus Speed 29
4.3 Setting the TCP Rx Window 29
4.4 Increasing NFS performance 30

5 Vendor/Manufacturer/Model Specific Information **30**

5.1 3Com 31
5.1.1 3c501 31
5.1.2 3c503, 3c503/16 31
5.1.3 3c505 32
5.1.4 3c507 32
5.1.5 3c509 / 3c509B 33
5.1.6 3c523 34
5.1.7 3c527 34

5.1.8 3c529 . . . 34
5.1.9 3c562 . . . 34
5.1.10 3c579 . . . 35
5.1.11 3c589 / 3c589B . . . 35
5.1.12 3c590 / 3c595 . . . 36
5.1.13 3c592 / 3c597 . . . 36
5.1.14 3c900 / 3c905 . . . 36
5.2 Accton . . . 37
5.2.1 Accton MPX . . . 37
5.2.2 Accton EN1203, EN1207, EtherDuo-PCI . . . 37
5.2.3 Accton EN2212 PCMCIA Card . . . 37
5.3 Allied Telesyn/Telesis . . . 37
5.3.1 AT1500 . . . 37
5.3.2 AT1700 . . . 37
5.3.3 AT2450 . . . 38
5.4 AMD / Advanced Micro Devices . . . 38
5.4.1 AMD LANCE (7990, 79C960, PCnet-ISA) . . . 38
5.4.2 AMD 79C961 (PCnet-ISA+) . . . 39
5.4.3 AMD 79C965 (PCnet-32) . . . 39
5.4.4 AMD 79C970 (PCnet-PCI) . . . 39
5.4.5 AMD 79C974 (PCnet-SCSI) . . . 40
5.5 Ansel Communications . . . 40
5.5.1 AC3200 EISA . . . 40
5.6 Apricot . . . 40
5.6.1 Apricot Xen-II On Board Ethernet . . . 40
5.7 Arcnet . . . 41
5.8 AT&T . . . 41
5.8.1 AT&T T7231 (LanPACER+) . . . 41
5.9 AT-Lan-Tec / RealTek . . . 41
5.9.1 AT-Lan-Tec / RealTek Pocket adaptor . . . 41
5.9.2 RealTek 8029 . . . 42
5.10 Boca Research . . . 42

5.10.1 Boca BEN (PCI, VLB) . . . 42
5.11 Cabletron . . . 43
5.11.1 E10**, E10**-x, E20**, E20**-x . . . 44
5.11.2 E2100 . . . 44
5.12 Cogent . . . 44
5.12.1 EM100-ISA/EISA . . . 45
5.12.2 Cogent eMASTER+, EM100-PCI, EM400, EM960, EM964 . . . 45
5.13 Compaq . . . 45
5.13.1 Compaq Deskpro / Compaq XL (Embedded AMD Chip) . . . 45
5.14 Danpex . . . 46
5.14.1 Danpex EN9400 . . . 46
5.15 D-Link . . . 46
5.15.1 DE-100, DE-200, DE-220-T . . . 46
5.15.2 DE-520 . . . 46
5.15.3 DE-530 . . . 46
5.15.4 DE-600 . . . 47
5.15.5 DE-620 . . . 47
5.15.6 DE-650 . . . 47
5.16 DFI . . . 47
5.16.1 DFINET-300 and DFINET-400 . . . 47
5.17 Digital / DEC . . . 48
5.17.1 DEPCA, DE100/1, DE200/1/2, DE210, DE422 . . . 48
5.17.2 Digital EtherWorks 3 (DE203, DE204, DE205) . . . 48
5.17.3 DE425 (EISA), DE434, DE435, DE500 . . . 48
5.17.4 DEC 21040, 21041, 21140, Tulip . . . 49
5.18 Farallon . . . 49
5.18.1 Farallon Etherwave . . . 49
5.19 Hewlett Packard . . . 50
5.19.1 27245A . . . 50
5.19.2 HP PC Lan+ (27247, 27252A) . . . 50
5.19.3 HP-J2405A . . . 50
5.19.4 HP-Vectra On Board Ethernet . . . 51

5.19.5 HP 10/100 VG Any Lan Cards (27248B, J2573, J2577, J2585) 51
5.20 IBM / International Business Machines 51
5.20.1 IBM Thinkpad 300 51
5.20.2 IBM Credit Card Adaptor for Ethernet 51
5.20.3 IBM Token Ring 52
5.21 ICL Ethernet Cards 52
5.21.1 ICL EtherTeam 16i/32 52
5.22 Intel Ethernet Cards 52
5.22.1 Ether Express 52
5.22.2 Ether Express PRO/10 52
5.22.3 Ether Express PRO/10 EISA/PCI 52
5.22.4 Ether Express PRO 10/100B 53
5.23 LinkSys 53
5.23.1 LinkSys Pocket Ethernet Adapter Plus (PEAEPP) 53
5.23.2 LinkSys PCMCIA Adaptor 53
5.24 Microdyne 53
5.24.1 Microdyne Exos 205T 53
5.25 Mylex 53
5.25.1 Mylex LNP101 54
5.25.2 Mylex LNP104 54
5.25.3 Mylex M390B EISA 54
5.26 Novell Ethernet, NExxxx and associated clones. 54
5.26.1 NE1000, NE2000 54
5.26.2 NE2000-PCI (RealTek-8029) 55
5.26.3 NE1500, NE2100 55
5.26.4 NE3200 55
5.27 Proteon 56
5.27.1 Proteon P1370-EA 56
5.27.2 Proteon P1670-EA 56
5.28 Pure Data 56
5.28.1 PDUC8028, PDI8023 56
5.29 Racal-Interlan 56

5.29.1 ES3210 . . . 56
5.29.2 NI5210 . . . 57
5.29.3 NI6510 (not EB) . . . 57
5.29.4 EtherBlaster (aka NI6510EB) . . . 57
5.30 Sager . . . 57
5.30.1 Sager NP943 . . . 57
5.31 Schneider & Koch . . . 58
5.31.1 SK G16 . . . 58
5.32 SEEQ . . . 58
5.32.1 SEEQ 8005 . . . 58
5.33 SMC (Standard Microsystems Corp.) . . . 58
5.34 Thomas Conrad . . . 58
5.34.1 Thomas Conrad TC-5048 . . . 58
5.35 Western Digital / SMC . . . 58
5.35.1 WD8003, SMC Elite . . . 59
5.35.2 WD8013, SMC Elite16 . . . 59
5.35.3 SMC Elite Ultra . . . 60
5.35.4 SMC EtherEZ (8416) . . . 60
5.35.5 SMC EtherPower PCI (8432) . . . 61
5.35.6 SMC 3008 . . . 61
5.35.7 SMC 3016 . . . 61
5.35.8 SMC-9000 / SMC 91c92/4 . . . 61
5.35.9 SMC 91c100 . . . 62
5.36 Xircom . . . 62
5.36.1 PE1, PE2, PE3-10B* . . . 62
5.37 Zenith . . . 62
5.37.1 Z-Note . . . 62
5.38 Znyx . . . 63
5.38.1 Znyx ZX342 (DEC 21040 based) . . . 63
5.39 Drivers for Non-Ethernet Devices . . . 63

6 Cables, Coax, Twisted Pair **64**

6.1 Thin Ethernet (thinnet) . . . 64

6.2 Twisted Pair . . . 64

6.3 Thick Ethernet . . . 66

7 Software Configuration and Card Diagnostics **66**

7.1 Configuration Programs for Ethernet Cards . . . 66

7.1.1 WD80x3 Cards . . . 66

7.1.2 Digital / DEC Cards . . . 67

7.1.3 NE2000+ or AT/LANTIC Cards . . . 67

7.1.4 3Com Cards . . . 68

7.2 Diagnostic Programs for Ethernet Cards . . . 68

7.3 Identifying an Unknown Card . . . 68

7.3.1 Identifying the Network Interface Controller . . . 68

7.3.2 Identifying the Ethernet Address . . . 69

7.3.3 Tips on Trying to Use an Unknown Card . . . 69

8 Technical Information **70**

8.1 Probed Addresses . . . 70

8.2 Writing a Driver . . . 71

8.3 Driver interface to the kernel . . . 72

8.3.1 Probe . . . 72

8.3.2 Interrupt handler . . . 72

8.3.3 Transmit function . . . 73

8.3.4 Receive function . . . 73

8.3.5 Open function . . . 73

8.3.6 Close function (optional) . . . 73

8.3.7 Miscellaneous functions . . . 73

8.4 Interrupts and Linux . . . 73

8.5 Programming the Intel chips (i82586 and i82593) . . . 75

8.6 Technical information from 3Com . . . 75

8.7 Notes on AMD PCnet / LANCE Based cards . . . 76

8.8 Multicast and Promiscuous Mode . . . 77

8.9 The Berkeley Packet Filter (BPF) . . . 79

9 Networking with a Laptop/Notebook Computer 79

9.1 Using SLIP . . . 79

9.2 Built in NE2000 . . . 79

9.3 PCMCIA Support . . . 80

9.4 ISA Ethercard in the Docking Station. . . . 80

9.5 Pocket / parallel port adaptors. . . . 80

10 Miscellaneous. 81

10.1 Passing Ethernet Arguments to the Kernel . . . 81

10.1.1 The `ether` command . . . 81

10.1.2 The `reserve` command . . . 82

10.2 Using the Ethernet Drivers as Modules . . . 83

10.2.1 8390 Based Cards as Modules . . . 83

10.3 Mailing Lists and the Linux Newsgroups . . . 84

10.4 Related Documentation . . . 84

10.5 Contributors . . . 85

10.6 Disclaimer and Copyright . . . 85

10.7 Closing . . . 86

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 `rtfm.mit.edu` just like all the other newsgroup FAQs.

This present revision covers distribution kernels up to and including v2.0.21. Information pertaining to development kernels up to version 2.1.0 is also documented.

The Ethernet-Howto is edited and maintained by:

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

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

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

who we should 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!

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

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

Please see the Disclaimer and Copying information at the end of this document for information about redistribution of this document and the usual 'we are not responsible for what you do...' legal type mumblings.

1.1 New Versions of this Document

New versions of this document can be retrieved via anonymous FTP from:

Sunsite HOWTO Archive `<ftp://sunsite.unc.edu/pub/Linux/docs/HOWTO/>`

and various Linux ftp mirror sites. Updates will be made as new information and/or drivers becomes available. If this copy that you are reading is more than 6 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 a WWW browser) 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>`

1.2 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 the good news is that you don't *have* to read it all.

Chances are you are reading this document beacuse you can't get things to work and you don't know what to do or check. The next section (1.3) is aimed at newcomers to linux and will point you in the right direction.

Typically the same problems and questions are asked *over and over* again by different people. Chances are your specific problem or question is one of these frequently asked questions, and is answered in the FAQ portion of this document . (3). Everybody should have a look through this section before posting for help.

If you haven't got an ethernet card, then you will want to start with deciding on a card. (2)

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 the section which contains specific information on each manufacturer, and their cards. (5)

If you are interested in some of the technical aspects of the Linux device drivers, then you can have a browse of the section with this type of information. (8)

1.3 HELP - It doesn't work!

Okay, don't panic. This will lead you through the process of getting things working, even if you have no prior background in linux or ethernet hardware.

First thing you need to do is figure out what model your card is so you can determine if Linux has a driver for that particular card. Different cards typically have different ways of being controlled by the host computer, and the linux driver (if there is one) contains this control information in a format that allows linux to use the card.

If you don't have any manuals or anything of the sort that tell you anything about the card model, then you can either see the section on helping with mystery cards (7.3), or just try a 'kitchen-sink' kernel with nearly every driver built in and hope one of the drivers recognizes your card.

Now that you know what type of card you have, read through the details of your particular card in the card specific section (5) which lists in alphabetical order, card manufacturers, individual model numbers and whether it has a linux driver or not. If it lists it as 'Not Supported' you can pretty much give up here. If you can't find your card in that list, then check to see if your card manual lists it as being 'compatible' with another known card type. For example there are hundreds, if not thousands of different cards made to be compatible with the original Novell NE2000 design.

Assuming you have found out that your card does have a linux driver, you now need to go back to the CD-ROM or whatever you installed from, and find the list of pre-built kernels that comes with it. The kernel is the core operating system that is first loaded at boot, and contains drivers for various pieces of hardware, among other things. Just because linux has a driver for your card does *not* mean that it is built into every kernel. Depending on who made the CD-ROM, there may be only a few pre-built kernels, and a whole bunch of drivers as smaller separate modules, or there may be a whole lot of kernels, covering a vast combination of built-in driver combinations. Hopefully there will also be a text file with them that lists what drivers are included into which kernels. Try and find a kernel that is listed as having the driver you need as built into it, or try and find a module with the name of the driver you need.

If you found a pre-built kernel that has your driver in it, you will want to boot that kernel instead of the one you are presently using. Most linux systems use LILO to boot, and will have installed the LILO documentation on your system. Follow the instructions in that for booting another kernel, as they are beyond the scope of this document.

If you instead found a small module that contains the driver, you will need to attach this module to the kernel after it has booted up. See the information that came with your distribution on installing and using modules, along with the module section in this document. (10.2)

If you didn't find either a pre-built kernel with your driver, or a module form of the driver, chances are you have a typically uncommon card, and you will have to build your own kernel with that driver included. Once you have linux installed, building a custom kernel is not difficult at all. You essentially answer yes or no to what you want the kernel to contain, and then tell it to build it. There is a Kernel-HowTo that will help you along.

At this point you should have somehow managed to be booting a kernel with your driver built in, or be loading it as a module. About half of the problems people have are related to not having driver loaded one way or another, so you may find things work now.

If it still doesn't work, then you need to verify that the kernel is indeed detecting the card. To do this, you need to type `dmesg | more` when logged in after the system has booted and all modules have been loaded. This will allow you to review the boot messages that the kernel scrolled up the screen during the boot process. If the card has been detected, you should see somewhere in that list a message from your card's driver that starts with `eth0`, mentions the driver name and the hardware parameters (interrupt setting, input/output port address, etc) that the card is set for. If you don't see a message like this, then the driver didn't detect your card, and that is why things aren't working. See the FAQ (3) for what to do if your card is not detected. If you have a NE2000 compatible, there is also some NE2000 specific tips on getting a card detected in the FAQ section as well.

If the card is detected, but the detection message reports some sort of error, like a resource conflict, then the driver probably won't have initialized properly and the card still wont be useable. Most common error messages of this sort are also listed in the FAQ section, along with a solution.

If the detection message seems okay, then double check the card resources reported by the driver against those that the card is physically set for (either by little black jumpers on the card, or by a software utility supplied by the card manufacturer.) These must match exactly. For example, if you have the card jumpered or configured to IRQ 15 and the driver reports IRQ 10 in the boot messages, things will not work. The FAQ section discusses the most common cases of drivers incorrectly detecting the configuration information of various cards.

At this point, you have managed to get you card detected with all the correct parameters, and hopefully everything is working. If not, then you either have a software configuration error, or a hardware configuration error. A software configuration error is not setting up the right network addresses for the `ifconfig` and `route` commands, and details of how to do that are fully described in the NET2-HowTo and the 'Network Administrator's Guide' which both probably came on the CD-ROM you installed from.

A hardware configuration error is when some sort of resource conflict or mis-configuration that the driver didn't detect at boot stops the card from working properly. This typically can be observed in one of three different ways. (1) You get an error message when `ifconfig` tries to open the device for use, such as "SIOCSFFLAGS: Try again". (2) The driver reports `eth0` error messages (viewed by `dmesg | more`) or strange inconsistencies for each time it tries to send or receive data. (3) Typing `cat /proc/net/dev` shows non-zero numbers in one of the errs, drop, fifo, frame or carrier columns for `eth0`. Most of the typical hardware configuration errors are also discussed in the FAQ section.

Well, if you have got to this point, read the FAQ section of this document, read the vendor specific section detailing your particular card, *and it still doesn't work* then you may have to resort to posting to an appropriate newsgroup for help. If you do post, please detail all relevant information in that post, such as the card brand, the kernel version, the driver boot messages, the output from `cat /proc/net/dev`, a clear description of the problem, and of course what you have already tried to do in an effort to get things to work.

You would be surprised at how many people post useless things like "Can someone help me? My ethernet doesn't work." and nothing else. Readers of the newsgroups tend to ignore such silly posts, wheras a

detailed and informational problem description may allow a 'linux-guru' to spot your problem right away.

2 What card should I buy for Linux?

The answer to this question depends heavily on exactly what you intend on doing with your net connection, and how much traffic it will see.

If you only expect a single user to be doing the occasional ftp session or WWW connection, then an old 8 bit wd8003 card will keep you happy.

If you intend to set up a server, and you require the CPU overhead of Rx'ing and Tx'ing ether packets to be kept at a minimum, you probably want to look at one of the newer PCI cards with the DEC 21040 chip, or the AMD PCnet-PCI chip.

If you fall somewhere in the middle of the above, then any one of the 16 bit ISA cards with stable drivers will do the job for you.

2.1 So What Drivers are Stable?

Of the 16 bit ISA cards, the following drivers are very mature, and you shouldn't have any problems if you buy a card that uses these drivers.

SMC-Ultra/EtherEZ, WD80x3, 3c509, 3c503/16, Lance, NE2000.

This is not to say that all the other drivers are unstable. It just happens that the above are the oldest and most used of all the linux drivers, making them the safest choice.

Note that some el-cheapo motherboards can have trouble with the bus-mastering that the lance cards do, and some el-cheapo NE2000 clones can have trouble getting detected at boot.

As for PCI cards, the PCnet-PCI cards that use the lance driver are a safe choice (except for the Boca cards as they have hardware flaws). The Allied Telsyn AT2450 is a PCnet-PCI implementation that is known to work well.

The DEC 21040 'tulip' driver and the 3c59x 'vortex' driver are relatively new drivers, but have proven themselves to be quite stable already.

2.2 Eight bit vs 16 bit Cards

You probably can't buy a new 8 bit ISA ethercard anymore, but you will find lots of them turning up at computer swap meets and the like for the next few years, at very low prices. This will make them popular for "home-ethernet" systems.

Some 8 bit cards that will provide adequate performance for light to average use are the wd8003, the 3c503 and the ne1000. The 3c501 provides poor performance, and these poor 12 year old relics of the XT days should be avoided.

The 8 bit data path doesn't hurt performance that much, as you can still expect to get about 500 to 800kB/s ftp download speed to an 8 bit wd8003 card (on a fast 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.

2.3 32 Bit / VLB / PCI 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 a large price increment for the 32 bit interface. Now that 100Mbs networks are becoming more common, this is changing though.

See 2.6 as to why having a 10Mbps ethercard on an 8MHz ISA bus is really not a bottleneck. Even though having the ethercard on a fast bus won't necessarily mean faster transfers, it will usually mean reduced CPU overhead, which is good for multi-user systems.

AMD has the 32 bit PCnet-VLB and PCnet-PCI chips. See 5.4.3 for info on the 32 bit versions of the LANCE / PCnet-ISA chip.

The DEC 21040 PCI chip is another option (see 5.17.4) for power-users. Many manufacturers produce cards that use this chip, and the prices of such no-name cards is usually quite cheap.

3Com's 'Vortex' and 'Boomerang' PCI cards are also another option, and the price is quite cheap if you can get one under their evaluation deal while it lasts. (see 5.1.12)

Various clone manufacturers have started making PCI ne2000 clones based on the RealTek 8029 chip. These cards are also supported by the linux ne2000 driver for v2.0 kernels. However you only benefit from the faster bus interface, as the card is still using the age-old ne2000 driver interface.

2.4 Available 100Mbs Cards and Drivers

The present list of supported 100Mbs hardware is as follows: cards with the DEC 21140 chip; the 3c595 Vortex card; and the HP 100VG ANY-LAN. The drivers for the first two are quite stable, but feedback on the HP driver has been low so far as it has only been around since early 1.3.x kernels.

The EtherExpressPro10/100B is finally supported. However you will have to obtain the driver separately from Donald's FTP or WWW site (see below) for v2.0 kernels.

The 21140 100Base-? chip is supported with the same driver as its 10Mbs counterpart, the 21040. SMC's 100Mbs EtherPower PCI card uses this chip. As with the 21040, you have a choice of two drivers to pick from.

Also have a look at the information on Donald's WWW site, at the following URL:

100Mbs Ethernet `<http://cesdis.gsfc.nasa.gov/linux/misc/100mbs.html>`

Donald had done a fair bit of work with the SMC EtherPower-10/100 cards, and reported getting about 4.6MB/s application to application with TCP on P5-100 Triton machines.

(See 5.1.12 and 5.17.4 for more details.)

For 100VG information, see the following section, and this URL on Donald's Site:

Donald's 100VG Page <http://cesdis.gsfc.nasa.gov/linux/drivers/100vg.html>

You may also be interested in looking at:

Dan Kegel's Fast Ethernet Page <http://alumni.caltech.edu/~dank/fe/>

2.5 100VG versus 100BaseT

The following blurb from yet another one of Donald's informative comp.os.linux postings summarizes the situation quite well:

"For those not in the know, there are two competing 100Mbs ethernet standards, 100VG (aka 100baseVG and 100VG-AnyLAN) and 100baseT (with 100baseTx, 100baseT4 and 100baseFx cable types).

100VG was on the market first, and I feel that it is better engineered than 100baseTx. I was rooting for it to win, but it clearly isn't going to. HP et al. made several bad choices:

1) Delaying the standard so that they could accommodate IBM and support token ring frames. It 'seemed like a good idea at the time', since it would enable token ring shops to upgrade without the managers having to admit they made a very expensive mistake committing to the wrong technology. But there was nothing to be gained, as the two frame types couldn't coexist on a network, token ring is a morass of complexity, and IBM went with 100baseT anyway.

2) Producing only ISA and EISA cards. (A PCI model was only recently announced.) The ISA bus is too slow for 100mbs, and relatively few EISA machines exist. At the time VLB was common, fast, and cheap with PCI a viable choice. But "old-timer" wisdom held that servers would stay with the more expensive EISA bus.

3) Not sending me a databook. Yes, this action was the real reason for the 100VGs downfall :-). I called all over for programming info, and all I could get was a few page color glossy brochure from AT&T describing how wonderful the Regatta chipset was."

2.6 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 (wd80x3, SMC-Ultra, 3c503, ne2000, etc) come pretty close to sending back-to-back packets (depending on the current interrupt latency) and the 3c509 and AT1500 hardware have 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, listed below.

2.6.1 Programmed I/O (e.g. NE2000, 3c509)

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.

2.6.2 Shared memory (e.g. WD80x3, SMC-Ultra, 3c503)

Pro: Simple, faster than programmed I/O, and allows random access to packets. The linux drivers compute the checksum of incoming IP packets as they are copied off the card, resulting in a further reduction of CPU usage vs. an equivalent PIO card.

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

2.6.3 Slave (normal) Direct Memory Access (e.g. none for Linux!)

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.

2.6.4 Bus Master Direct Memory Access (e.g. LANCE, DEC 21040)

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 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.)

2.7 Type of cable that your card should support

If you are setting up a small "personal" network, you will probably want to use thinnet or thin ethernet cable. This is the style with the standard BNC connectors. See 6 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 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.

3.1 Alpha Drivers – Getting and Using them

I heard that there is an updated or 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.

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!

Note that some of the 'useable' experimental/alpha drivers have been included in the standard kernel source tree. When running `make config` one of the first things you will be asked is whether to "Prompt for development and/or incomplete code/drivers". You will have to answer 'Y' here to get asked about including any alpha/experiemntal drivers.

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.

3.2 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 at the moment only *one* ethercard is auto-probed for by default. This helps to avoid 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.

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>`.

For module users with 8390 based cards, you can have a single module control multiple cards of the same brand. Please see 10.2.1 for module specific information about using multiple cards.

3.3 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.

This is not a comprehensive list of all the NE2000 clones that don't have the `0x57,0x57` in bytes `0x0e,0x0f` of the SAPROM. There are probably hundreds of them. If you get a card that causes the driver to report an 'invalid signature' then you will have to add your cards signature to the driver. The process for doing this is described below.

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

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 5.11.1.

D-Link Ethernet II – bad clones, but the driver checks for them. See 5.15.1.

DFI DFINET-300, DFINET-400 – bad clones, but the driver checks for them. See 5.16.1

EtherNext UTP8, EtherNext UTP16 – bad clones, but the driver checks for them.

3.4 Problems with NE1000 / NE2000 cards (and clones)

Problem: NE*000 card hangs machine, sometimes with a 'DMA conflict' message, sometimes completely silently.

Reason: There were some bugs in the driver and the upper networking layers that caused this. They have been fixed in kernels v1.2.9 and above. Upgrade your kernel.

Problem: NE*000 card hangs machine during NE probe, or can not read station address properly.

Reason: Kernels previous to v1.3.7 did not fully reset the card after finding it at boot. Some cheap cards are not left in a reasonable state after power-up and need to be fully reset before any attempt is made to use them. Also, a previous probe may have upset the NE card prior to the NE probe taking place. In that case, look in to using the "reserve=" boot keyword to protect the card from other probes.

Problem: NE*000 driver reports 'not found (no reset ack)' during boot probe.

Reason: This is related to the above change. After the initial verification that an 8390 is at the probed i/o address, the reset is performed. When the card has completed the reset, it is supposed to acknowedge that the reset has completed. Your card doesn't, and so the driver assumes that no NE card is present.

Solution: You can tell the driver that you have a bad card by using an otherwise unused mem_end hexidecimal value of 0xbad at boot time. You *have* to also supply a non-zero i/o base for the card when using the `0xbad` override. For example, a card that is at `0x340` that doesn't ack the reset would use something like:

```
LILO: linux ether=0,0x340,0,0xbad,eth0
```

This will allow the card detection to continue, even if your card doesn't ACK the reset.

Problem: NE*000 card hangs machine at first network access.

Reason: This problem has been reported for kernels as old as 1.1.57 to the present. It appears confined to a few software configurable clone cards. It appears that they expect to be initialized in some special way.

Solution: Several people have reported that running the supplied DOS software config program and/or the supplied DOS driver prior to warm booting (i.e. loadlin or the 'three-finger-salute') into linux allowed the card to work. This would indicate that these cards need to be initialized in a particular fashion, slightly different than what the present Linux driver does.

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: There are two ways to get around this. The easiest is to use an `0xbad` mem_end value as described above for the 'no reset ack' problem. This will bypass the signature check, as long as a non-zero i/o base is also given. This way no recompilation of the kernel is required.

The second method involves changing the driver itself, and then recompiling your kernel. 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: %2.2x %2.2x %2.2x\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. (Note: The DMA address check is not done by default as of v1.2.4 for performance reasons. Enable it with the 'NE_SANITY' define in `ne.c` if you want the check done.)

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 older 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 something like `0x340`. Alternatively, you can use the "reserve=" boot argument in conjunction with the "ether=" argument to protect the card from other device driver probes.

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. 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 reserve/ether boot arguments.

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

Solution: There is no 'magic solution' as there can be a number of reasons why it wasn't detected. The following list should help you walk through the possible problems.

1) Build a new kernel with only the device drivers that you need. Verify that you are indeed booting the fresh kernel. Forgetting to run lilo, etc. can result in booting the old one. (Look closely at the build time/date reported at boot.) Sounds obvious, but we have all done it before. Make sure the driver is in fact included in the new kernel, by checking the `System.map` file for names like `ne_probe`.

2) Look at the boot messages carefully. Does it ever even mention doing a ne2k probe such as 'NE*000 probe at 0xNNN: not found (blah blah)' or does it just fail silently. There is a big difference. Use `dmesg|more` to review the boot messages after logging in, or hit Shift-PgUp to scroll the screen up after the boot has completed and the login prompt appears.

3) After booting, do a `cat /proc/ioports` and verify that the full iospace that the card will require is vacant. If you are at `0x300` then the ne2k driver will ask for `0x300-0x31f`. If any other device driver has registered even one port anywhere in that range, the probe will not take place at that address and will silently continue to the next of the probed addresses. A common case is having the lp driver reserve `0x378` which stops the ne driver from probing `0x360`.

4) Same as above for `cat /proc/interrupts`. Make sure no other device has registered the interrupt that you set the ethercard for. In this case, the probe will happen, and the ether driver will complain loudly at boot about not being able to get the desired IRQ line.

5) If you are still stumped by the silent failure of the driver, then edit it and add some printk() to the probe. For example, with the ne2k you could add/remove lines (marked with a '+' or '-') in net/ne.c like:

```
    int reg0 = inb_p(ioaddr);

+    printk("NE2k probe - now checking %x\n",ioaddr);
-    if (reg0 == 0xFF)
+    if (reg0 == 0xFF) {
+       printk("NE2k probe - got 0xFF (vacant i/o port)\n");
        return ENODEV;
+    }
```

Then it will output messages for each port address that it checks, and you will see if your card's address is being probed or not.

6) You can also get the ne2k diagnostic from Don's ftp site (mentioned in the howto as well) and see if it is able to detect your card after you have booted into linux. Use the "-p 0xNNN" option to tell it where to look for the card. (The default is 0x300 and it doesn't go looking elsewhere, unlike the boot-time probe.) The output from when it finds a card will look something like:

```
Checking the ethercard at 0x300.
  Register 0x0d (0x30d) is 00
  Passed initial NE2000 probe, value 00.
8390 registers: 0a 00 00 00 63 00 00 00 01 00 30 01 00 00 00 00
SA PROM  0: 00 00 00 00 c0 c0 b0 b0 05 05 65 65 05 05 20 20
SA PROM 0x10: 00 00 07 07 0d 0d 01 01 14 14 02 02 57 57 57 57

        NE2000 found at 0x300, using start page 0x40 and end page 0x80.
```

Your register values and PROM values will probably be different. Note that all the PROM values are doubled for a 16 bit card, and that the ethernet address (00:00:c0:b0:05:65) appears in the first row, and the double `0x57` signature appears at the end of the PROM.

The output from when there is no card installed at `0x300` will look something like this:

```
Checking the ethercard at 0x300.
  Register 0x0d (0x30d) is ff
  Failed initial NE2000 probe, value ff.
8390 registers: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
SA PROM        0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
SA PROM 0x10: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff

 Invalid signature found, wordlength 2.
```

The `0xff` values arise because that is the value that is returned when one reads a vacant i/o port. If you happen to have some other hardware in the region that is probed, you may see some non `0xff` values as well.

7) Try warm booting into linux from a DOS boot floppy (via loadlin) after running the supplied DOS driver or config program. It may be doing some extra (i.e. non-standard) "magic" to initialize the card.

8) Try Russ Nelson's ne2000.com packet driver to see if even it can see your card – if not, then things do not look good. Example:

```
A:> ne2000 0x60 10 0x300
```

The arguments are software interrupt vector, hardware IRQ, and i/o base. You can get it from any msdos archive in pktdrv11.zip – The current version may be newer than 11.

3.5 Problems with SMC Ultra/EtherEZ and WD80*3 cards

Problem: You get messages such as the following:

```
eth0: bogus packet size: 65531, status=0xff, nxpg=0xff
```

Reason: There is a shared memory problem.

Solution: The most common reason for this is PCI machines that are not configured to map in ISA memory devices. Hence you end up reading the PC's RAM (all `0xff` values) instead of the RAM on the card that contains the data from the received packet.

Other typical problems that are easy to fix are board conflicts, having cache or 'shadow ROM' enabled for that region, or running your ISA 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.

Problem: SMC EtherEZ doesn't work in non-shared memory (PIO) mode.

Reason: Older versions of the Ultra driver only supported the card in the shared memory mode of operation.

Solution: The driver in kernel version 2.0 and above also supports the programmed i/o mode of operation. Upgrade to v2.0, or get the drop-in replacement for kernel v1.2.13 from Donald's ftp/www site.

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
```

Problem: SMC Ultra card is detected as wd8013, but the IRQ and shared memory base is wrong.

Reason: The Ultra card looks a lot like a wd8013, and if the Ultra driver is not present in the kernel, the wd driver may mistake the ultra as a wd8013. The ultra probe comes before the wd probe, so this usually shouldn't happen. The ultra stores the IRQ and mem base in the EEPROM differently than a wd8013, hence the bogus values reported.

Solution: Recompile with only the drivers you need in the kernel. If you have a mix of wd and ultra cards in one machine, and are using modules, then load the ultra module first.

3.6 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 X invalid, will use autoIRQ.

Reason: The 3c503 card can only use one of IRQ{5, 2/9, 3, 4} (These are the only lines that are connected to the card.) If you pass in an IRQ value that is not in the above set, you will get the above message. Usually, specifying an interrupt value for the 3c503 is not necessary. The 3c503 will autoIRQ when it gets ifconfig'ed, and pick one of IRQ{5, 2/9, 3, 4}.

Solution: Use one of the valid IRQs listed above, or enable autoIRQ by not specifying the IRQ line at all.

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
```

With kernels 1.3.42 and newer, you can specify the AUI port when loading as a module as well. Just append `xcvr=1` to the `insmod` command line along with your i/o and irq values.

3.7 FAQs Not Specific to Any Card.

3.7.1 Ethercard is Not Detected at Boot.

The usual reason for this is that people are using a kernel that does not have support for their particular card built in. If you are using a pre-compiled kernel that is part of a distribution set, then check the documentation to see which kernel you installed, and if it was built with support for your particular card. If it wasn't, then your options are to try and get one that has support for your card, or build your own.

It is usually wise to build your own kernel with only the drivers you need, as this cuts down on the kernel size (saving your precious RAM for applications!) and reduces the number of device probes that can upset sensitive hardware. Building a kernel is not as complicated as it sounds. You just have to answer yes or no to a bunch of questions about what drivers you want, and it does the rest.

The next main cause is having another device using part of the i/o space that your card needs. Most cards are 16 or 32 bytes wide in i/o space. If your card is set at 0x300 and 32 bytes wide, then the driver will ask for 0x300-0x31f. If any other device driver has registered even one port anywhere in that range, the probe will not take place at that address and the driver will silently continue to the next of the probed addresses. So, after booting, do a `cat /proc/ioports` and verify that the full iospace that the card will require is vacant.

Another problem is having your card jumpered to an i/o address that isn't probed by default. There is a list 8.1 for each card in this document. Even if the i/o setting of your card is not in the list of porbed addresses, you can supply it at boot with the `ether=` command as described in 10.1

3.7.2 `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.

3.7.3 Shared Memory ISA cards in PCI Machine dont work (0xffff)

This will usually show up as reads of lots of `0xffff` values. No shared memory cards of any type will work in a PCI machine unless you have the PCI ROM BIOS/CMOS SETUP configuration set properly. You have to set it to allow shared memory access from the ISA bus for the memory region that your card is trying to use. If you can't figure out which settings are applicable then ask your supplier or local computer

guru. For AMI BIOS, there is usually a "Plug and Play" section where there will be an "ISA Shared Memory Size" and "ISA Shared Memory Base" settings. For cards like the wd8013 and SMC Ultra, change the size from the default of 'Disabled' to 16kB, and change the base to the shared memory address of your card.

3.7.4 NexGen machine gets 'mismatched read page pointers' errors.

A quirk of the NexGen CPU caused all users with 8390 based cards (wd80x3, 3c503, SMC Ultra/EtherEZ, ne2000, etc.) to get these error messages. Kernel versions 2.0 and above do not have these problems. Upgrade your kernel.

3.7.5 Asynchronous Transfer Mode (ATM) Support

Werner Almesberger has been playing around with ATM support for linux. He has been working with the Efficient Networks ENI155p board (*Efficient Networks* `<http://www.efficient.com/>`) and the Zeitnet ZN1221 board (*Zeitnet* `<http://www.zeitnet.com/>`).

Werner says that the driver for the ENI155p is rather stable, while the driver for the ZN1221 is presently unfinished.

Check the latest/updated status at the following URL:

Linux ATM Support `<http://lrcwww.epfl.ch/linux-atm/>`

3.7.6 FDDI Support

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?)'

However, Larry Stefani (stefani@lkg.dec.com) recently announced a beta driver for v2.0 with DEC's DEFEA and DEFPA cards. You will have to contact him for more details.

3.7.7 Full Duplex Support

Will Full Duplex give me 20MBps? Does Linux support it?

Full Duplex is at the wiring level of things, and has nothing to do with the drivers in linux.

Cameron Spitzer writes the following about full duplex 10Base-T cards: "If you connect it to a full duplex switched hub, and your system is fast enough and not doing much else, it can keep the link busy in both directions. There is no such thing as full duplex 10BASE-2 or 10BASE-5 (thin and thick coax). Full

Duplex works by disabling collision detection in the adapter. That's why you can't do it with coax; the LAN won't run that way. 10BASE-T (RJ45 interface) uses separate wires for send and receive, so it's possible to run both ways at the same time. The switching hub takes care of the collision problem. The signalling rate is 10 Mbps."

3.7.8 Ethernet Cards for Linux on Alpha/AXP PCI Boards

As of v2.0, only the 3c509, depca, de4x5 and all the 8390 drivers (wd, smc-ultra, ne, 3c503, etc.) have been made 'architecture independent' so as to work on the DEC Alpha CPU based systems.

Note that the changes that are required aren't that complicated. You only need to do the following:

-multiply all `jiffies` related values by HZ/100 to account for the different HZ value that the Alpha uses. (i.e `timeout=2;` becomes `timeout=2*HZ/100;`)

-replace any i/o memory (640k to 1MB) pointer dereferences with the appropriate readb() writeb() readl() writel() calls, as shown in this example.

```
-       int *mem_base = (int *)dev->mem_start;
-       mem_base[0] = 0xba5eba5e;
+       unsigned long mem_base = dev->mem_start;
+       writel(0xba5eba5e, mem_base);
```

-replace all memcpy() calls that have i/o memory as source or target destinations with the appropriate one of `memcpy_fromio()` or `memcpy_toio()`.

3.7.9 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 6.2 – 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.

3.7.10 SIOCSIFxxx: No such device

I get a bunch of 'SIOCSIFxxx: No such device' messages at boot, followed by a 'SIOCADDRT: Network is unreachable' What is wrong?

Your ethernet device was not detected at boot, and when `ifconfig` and `route` are run, they have no device to work with. Use `dmesg | more` to review the boot messages and see if there are any messages about detecting an ethernet card.

3.7.11 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 necessarily 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.

3.7.12 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.

3.7.13 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.

3.7.14 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.

3.7.15 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.

3.7.16 Access to the raw Ethernet Device

How do I get access to the raw ethernet device in linux, without going through TCP/IP and friends?

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

This gives you a socket receiving 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 via `sendto()` calls. You have to have root access to do either of course.

4 Performance Tips

Here are some tips that you can use if you are suffering from low ethernet throughput, or to gain a bit more speed on those ftp transfers.

The `ttcp.c` program is a good test for measuring raw throughput speed. Another common trick is to do a `ftp> get large_file /dev/null` where `large_file` is > 1MB and residing in the buffer cache on the Tx'ing machine. (Do the 'get' at least twice, as the first time will be priming the buffer cache on the Tx'ing machine.) You want the file in the buffer cache because you are not interested in combining the file access speed from the disk into your measurement. Which is also why you send the incoming data to `/dev/null` instead of onto the disk.

4.1 General Concepts

Even an 8 bit card is able to receive back-to-back packets without any problems. The difficulty arises when the computer doesn't get the Rx'd packets off the card quick enough to make room for more incoming packets. If the computer does not quickly clear the card's memory of the packets already received, the card will have no place to put the new packet.

In this case the card either drops the new packet, or writes over top of a previously received packet. Either one seriously interrupts the smooth flow of traffic by causing/requesting re-transmissions and can seriously degrade performance by up to a factor of 5!

Cards with more onboard memory are able to "buffer" more packets, and thus can handle larger bursts of back-to-back packets without dropping packets. This in turn means that the card does not require as low a latency from the the host computer with respect to pulling the packets out of the buffer to avoid dropping packets.

Most 8 bit cards have an 8kB buffer, and most 16 bit cards have a 16kB buffer. Most Linux drivers will reserve 3kB of that buffer (for two Tx buffers), leaving only 5kB of receive space for an 8 bit card. This is room enough for only three full sized (1500 bytes) ethernet packets.

4.2 ISA Bus Speed

As mentioned above, if the packets are removed from the card fast enough, then a drop/overrun condition won't occur even when the amount of Rx packet buffer memory is small. The factor that sets the rate at which packets are removed from the card to the computer's memory is the speed of the data path that joins the two – that being the ISA bus speed. (If the CPU is a dog-slow 386sx-16, then this will also play a role.)

The recommended ISA bus clock is about 8MHz, but many motherboards and peripheral devices can be run at higher frequencies. The clock frequency for the ISA bus can usually be set in the CMOS setup, by selecting a divisor of the mainboard/CPU clock frequency.

For example, here are some receive speeds as measured by the TTCP program on a 40MHz 486, with an 8 bit WD8003EP card, for different ISA bus speeds.

ISA Bus Speed (MHz)	Rx TTCP (kB/s)
6.7	740
13.4	970
20.0	1030
26.7	1075

You would be hard pressed to do better than 1075kB/s with *any* 10Mb/s ethernet card, using TCP/IP. However, don't expect every system to work at fast ISA bus speeds. Most systems will not function properly at speeds above 13MHz. (Also, most PCI systems have the ISA bus speed fixed at 8MHz, so that the end user does not have the option of increasing it.)

In addition to faster transfer speeds, one will usually also benefit from a reduction in CPU usage due to the shorter duration memory and i/o cycles. (Note that hard disks and video cards located on the ISA bus will also usually experience a performance increase from an increased ISA bus speed.)

Be sure to back up your data prior to experimenting with ISA bus speeds in excess of 8MHz, and thouroughly test that all ISA peripherals are operating properly after making any speed increases.

4.3 Setting the TCP Rx Window

Once again, cards with small amounts of onboard RAM and relatively slow data paths between the card and the computer's memory run into trouble. The default TCP Rx window setting is 32kB, which means that a fast computer on the same subnet as you can dump 32k of data on you without stopping to see if you received any of it okay.

Recent versions of the `route` command have the ability to set the size of this window on the fly. Usually it is only for the local net that this window must be reduced, as computers that are behind a couple of routers or gateways are 'buffered' enough to not pose a problem. An example usage would be:

```
route add <whatever> ... window <win_size>
```

where `win_size` is the size of the window you wish to use (in bytes). An 8 bit 3c503 card on an ISA bus operating at a speed of 10MHz or less would work well with a window size of about 4kB. Too large a window will cause overruns and dropped packets, and a drastic reduction in ethernet throughput. You can check the operating status by doing a `cat /proc/net/dev` which will display any dropped or overrun conditions that occurred.

4.4 Increasing NFS performance

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 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.

5 Vendor/Manufacturer/Model Specific Information

The following lists many cards in alphabetical order by vendor name and then product identifier. Beside each product ID, you will see either 'Supported', 'Semi-Supported' or 'Not Supported'.

Supported means that a driver for that card exists, and many people are happily using it and it seems quite reliable.

Semi-Supported means that a driver exists, but at least one of the following descriptions is true: (1) The driver and/or hardware are buggy, which may cause poor performance, failing connections or even crashes. (2) The card is fairly uncommon, and hence the driver has seen very little use/testing and the driver author has had very little feedback. Obviously (2) is preferable to (1), and the individual description of the card/driver should make it clear which one holds true. In either case, you will probably have to answer 'Y' when asked "Prompt for development and/or incomplete code/drivers?" when running `make config`. For the older v1.2 kernels, the option was known as `CONFIG_NET_ALPHA`. It was changed to avoid confusion with the Alpha-AXP line of processors made by Digital, and to encompass all experimental drivers, not just net drivers.

Not Supported means there is not a driver currently available for that card. This could be due to a lack of interest in hardware that is rare/uncommon, or because the vendors won't release the hardware documentation required to write a driver.

Note that the difference between 'Supported' and 'Semi-Supported' is rather subjective, and is based on user feedback observed in newsgroup postings and mailing list messages. (After all, it is impossible for one person to test all drivers with all cards for each kernel version!!!) So be warned that you may find a card

listed as semi-supported works perfectly for you (which is great), or that a card listed as supported gives you no end of troubles and problems (which is not so great).

5.1 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 8.6 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.

For those of you browsing this document by a WWW browser, you can try 3Com's WWW site

5.1.1 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.

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.

5.1.2 3c503, 3c503/16

Status – *Supported*

If you have a 3c503/16 you may be interested to know that as of 1.3.37 the driver has the facility to use the full 16kB RAM on your card. Previous versions treated the 16bit cards as 8bit cards, and only used half of the available RAM. This update also detects the newer 3Com prefix found on newly manufactured cards mentioned below.

Recently made 3c503/16 cards have a new base hardware address because 3Com ran out of numbers (they made too many cards!) The cards used to start with 02 60 8C and the newer ones use 00 20 AF. Up to 1.3.37, the driver will only check for the old address, and skip over the newer cards. You can upgrade to a kernel newer than 1.3.37, or change the numbers in 3c503.c for older kernels.

These shared-memory ethercards 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, but it is actually a bit slower. The 3c503 does not have "EEPROM setup", so a 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 in programmed-I/O mode, but this is slower and less reliable than shared memory mode. Also, programmed-I/O mode is not as well tested when updating the drivers. 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 3.6.

If you intend on using this driver as a loadable module you should probably see 10.2 and also 10.2.1 for module specific information.

5.1.3 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. There are not that many of these cards about. It is included in the standard kernel, but it is classed as an alpha driver. See 3.1 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 8.5.

5.1.4 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 3.1 for important information on using alpha-test ethernet drivers with Linux. Technical information is available in 8.5.

5.1.5 3c509 / 3c509B

Status – *Supported*

This card is fairly inexpensive and has good 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.) These cards use PIO transfers, similar to a ne2000 card, and so a shared memory card such as a wd8013 will be more efficient in comparison.

The original 3c509 has a small packet buffer (4kB total, 2kB Rx, 2kB Tx), 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.

The newer model 3c509B has 8kB on board, and the buffer can be split 4/4, 5/3 or 6/2 for Rx/Tx. This setting is changed with the DOS configuration utility, and is 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.

3c509B users should use the supplied DOS utility to disable the *plug and play* support, *and* to set the output media to what they require. The linux driver currently does *not* support the Autodetect media setting, so you *have* to select 10Base-T or 10Base-2 or AUI. With regards to the media detection features, Cameron said: "Autoselect is a feature of the commercial drivers for 3C509(B). AFAIK nobody ever claimed the Linux driver attempts it. When drivers/net/3c509.c recognizes my 3C509B at boot time, it says: `eth0: 3c509 at 0x300 tag 1, 10baseT port, ...` revealing that the card is configured for 10BASE-T. It finds that out by reading the little EEPROM, which IMHO is the Right Way To Do It."

As for the plug-and-pray stuff, Cameron adds: "The 3C509B has 3Com's relocatable I/O port scheme, and Microsoft*tm* Plug-and-play ("PnP"). You can't use them both at the same time. Some (broken, IMHO) BIOSes begin a PnP sequence by writing to the PnP address (0x279 ?), which causes PnP adapters like 3C509B to enter the PnP state, but then they (these funny BIOSes) never come back to finish the job. The 3C509Bs hang there in the middle of the PnP ID Sequence, where they have no idea you didn't mean it and you're going to use the 3Com ID sequence after all. 3C5X9CFG /PNPRST clears this hang. Disable PnP if your drivers (eg., Linux) don't use it.

It was a marketing decision to turn PnP on as a factory default setting. If it caused you a hassle, or not, please take the time to say so when you mail in your warranty card. The more info they have, the better decisions they can make. Also, check with your motherboard supplier to see if you need a BIOS upgrade."

It has been reported that you have to do a hard reset after doing the '3C5X9CFG /PNPRST' for the change to take effect.

Some people ask about the "Server or Workstation" and "Highest Modem Speed" settings presented in the DOS configuration utility. Donald writes "These are only hints to the drivers, and the Linux driver does not use these parameters: it always optimizes for high throughput rather than low latency ('Server'). Low latency was critically important for old, non-windowed, IPX throughput. To reduce the latency the

MS-DOS driver for the 3c509 disables interrupts for some operations, blocking serial port interrupts. Thus the need for the 'modem speed' setting. The Linux driver avoids the need to disable interrupts for long periods by operating only on whole packets e.g. by not starting to transmit a packet until it is completely transferred to the card."

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`). This detection method means that a particular card will *always* get detected first in a multiple ISA 3c509 configuration. The card with the lowest hardware ethernet address will *always* 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. If you have multiple 3c509 cards, it is best to append `ether=0,0,ethN` commands without the i/o port specified (i.e. use i/o=zero) and allow the probe to sort out which card is first, otherwise it may not detect all your cards.

5.1.6 3c523

Status – *Not Supported*

This MCA bus card uses the i82586, and now that people are actually running Linux on MCA machines, people have tried to recycle parts of the 3c507 driver into a driver for this card. More details can be found on the MCA-Linux page at `http://glycerine.cetmm.uni.edu/mca/`

5.1.7 3c527

Status – *Not Supported*

Yes, another MCA card. No, not too much interest in it. Better chances with the 3c529 if you are stuck with MCA.

5.1.8 3c529

Status – *Semi-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."

Darrell Frappier (aa822@detroit.freenet.org) reports that you can get the i/o address from running the PS/2 reference diskette, and once you put that directly into the driver, it does actually work.

5.1.9 3c562

Status – *Supported*

This PCMCIA card is the combination of a 3c589B ethernet card with a modem. The modem appears as a standard modem to the end user. The only difficulty is getting the two separate linux drivers to share one interrupt. There are a couple of new registers and some hardware interrupt sharing support. You need to use a v2.0 or newer kernel that has the support for interrupt sharing.

As a side note, the modem part of the card has been reported to be not well documented for the end user (the manual just says 'supports the AT command set') and it may not connect as well as other name brand modems. The recommendation is to buy a 3c589B instead, and then get a PCMCIA modem card from a company that specializes in modems.

Thanks again to Cameron for getting a sample unit and documentation sent off to David Hinds. Look for support in David's PCMCIA package release.

5.1.10 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)."

5.1.11 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. You will also 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 9.3 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.

5.1.12 3c590 / 3c595

Status – *Supported*

These "Vortex" cards are for PCI bus machines, with the '590 being 10Mbps and the '595 being 3Com's 100Mbs offering. Also note that you can run the '595 as a '590 (i.e. in a 10Mbps mode). The driver is included in the v2.0 kernel source, but is also continually being updated. If you have problems with the driver in the v2.0 kernel, you can get an updated driver from the following URL:

Vortex `<http://cesdis.gsfc.nasa.gov/linux/drivers/vortex.html>`

Note that there are two different 3c590 cards out there, early models that had 32kB of on-board memory, and later models that only have 8kB (eeccch!) of memory. Chances are you won't be able to buy a new 3c59x for much longer, as it is being replaced with the 3c90x card. If you are buying a used one off somebody, try and get the 32kB version. The 3c595 cards have 64kB, as you can't get away with only 8kB RAM at 100Mbps!

A thanks to Cameron Spitzer and Terry Murphy of 3Com for sending cards and documentation to Donald so he could write the driver.

5.1.13 3c592 / 3c597

Status – *Supported*

These are the EISA versions of the 3c59x series of cards. The 3c592/3c597 (aka Demon) should work with the vortex driver discussed above.

5.1.14 3c900 / 3c905

Status – *Supported*

These cards (aka 'Boomerang', aka EtherLink III XL) have been recently released to take over the place of the 3c590/3c595 cards. Cameron Spitzer of 3Com writes that the "3C900 has a scatter gather bus master controlled by a descriptor ring in main memory. Aside from that, it's a lot like 3C590."

You may still be able to get a couple of these cards at a reduced price through one of 3Com's evaluation deals, if you are quick.

To use this card with v2.0 kernels, you must obtain the updated `3c59x.c` driver from Donald's site at:

Vortex-Page `<http://cesdis.gsfc.nasa.gov/linux/drivers/vortex.html>`

This updated 3c59x driver allows you to use the 3c900 in a 3c59x compatible mode, and has been reported to be quite stable. Note that this updated driver may be snuck into the v2.0 source tree at a later date.)

On the same WWW page, you will also find the experimental `boomerang.c` driver which uses some of the enhancements of the 3c900 over that which is available on the 3c59x cards. Since this is a new/experimental driver, you may be better off in using the updated `3c59x.c` if system stability is a primary concern.

5.2 Accton

5.2.1 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...

5.2.2 Accton EN1203, EN1207, EtherDuo-PCI

Status – *Supported*

This is another implementation of the DEC 21040 PCI chip. Someone once mentioned that they had a bit of trouble with one of the above cards and the drivers at that point in time, so as with all purchases, you should try and make sure you can return it if it doesn't work for you.

See 5.17.4 for more information on these cards, and the present driver situation.

5.2.3 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.

5.3 Allied Telesyn/Telesis

5.3.1 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 hence one of the faster ISA bus ethercards available.

DMA selection and chip numbering information can be found in 5.4.1.

More technical information on AMD LANCE based Ethernet cards can be found in 8.7.

5.3.2 AT1700

Status – *Supported*

Note that to access this driver during `make config` you still have to answer 'Y' when asked "Prompt for development and/or incomplete code/drivers?" at the first. This is simply due to lack of feedback on the driver stability due to it being a relatively rare card. This will probably be changed for v2.1 kernels.

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 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). A fibre optic version of the card (AT1700FT) exists as well.

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.

5.3.3 AT2450

Status – *Supported*

This is the PCI version of the AT1500, and it doesn't suffer from the problems that the Boca 79c970 PCI card does. Allied Telsyn was still 'beta testing' the card in early/mid 1995, so it may not have spread to various retailers yet (but it doesn't hurt to ask.)

DMA selection and chip numbering information can be found in 5.4.1.

More technical information on AMD LANCE based Ethernet cards can be found in 8.7.

5.4 AMD / Advanced Micro Devices

5.4.1 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 5.3.1) the NE1500/2100 (see 5.26.3) and the Boca-VLB/PCI cards (see 5.10.1)

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

One common problem people have is the 'busmaster arbitration failure' message. This is printed out when the LANCE driver can't get access to the bus after a reasonable amount of time has elapsed (50us). This usually indicates that the motherboard implementation of bus-mastering DMA is broken, or some other device is hogging the bus, or there is a DMA channel conflict. If your BIOS setup has the 'GAT option' (for Guaranteed Access Time) then try toggling/altering that setting to see if it helps.

Chances are that the existing LANCE driver will work with all AMD LANCE based cards. (except perhaps some of the original 7990 designs with shared memory.) 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 8.7 for more info on these chips.

5.4.2 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. Also see the info in the above section.

Dave Platt writes: "The Lance driver may report recent versions of this chip as 'PCnet (unknown)', as the ID number in newer '961s seems to have been revised (it's now 0x2261 rather than 0x2260). This misidentification shouldn't prevent the driver from working with it, though. I've been told of a problem with the '961 - it will work correctly the first time you boot Linux after a hard reset, but will not work correctly after a soft reboot. From the data sheet, it looks as if the '961 disables itself upon reset, and won't "talk" again until the motherboard BIOS goes through the Plug+Play probe-and-enable sequence, and this might not be happening during a soft reboot. I do not yet know of a good workaround for this problem."

5.4.3 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.

5.4.4 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.

Note that the Boca implementation of the 79C970 fails on fast Pentium machines. This is a hardware problem, as it affects DOS users as well. See the Boca section for more details.

5.4.5 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.

5.5 Ansel Communications

5.5.1 AC3200 EISA

Status – *Semi-Supported*

Note that to access this driver during `make config` you still have to answer 'Y' when asked "Prompt for development and/or incomplete code/drivers?" at the first. This is simply due to lack of feedback on the driver stability due to it being a relatively rare card.

This driver is included in the present kernel as an alpha test driver. It is based on the common NS8390 chip used in the ne2000 and wd80x3 cards. Please see 3.1 in this document for important information regarding alpha drivers.

If you use it, let one of us know how things work out, as feedback has been low, even though the driver has been in the kernel since v1.1.25.

If you intend on using this driver as a loadable module you should probably see 10.2 and also 10.2.1 for module specific information.

5.6 Apricot

5.6.1 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.

5.7 Arcnet

Status – *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. It also has built in error handling so that it supposedly never loses a packet. (Great for UDP traffic!)

Avery Pennarun's arcnet driver has been in the default kernel sources since 1.1.80. 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@foxnet.net`

There are information files contained in the standard kernel for setting jumpers and general hints.

Supposedly the driver also works with the 100Mbs ARCnet cards as well!

5.8 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, without a hub that 'speaks' both.

5.8.1 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.

5.9 AT-Lan-Tec / RealTek

5.9.1 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. Apparently there are various clones of this adaptor being sold here and there throughout Europe as well.

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.

5.9.2 RealTek 8029

Status – *Supported*

This is a PCI single chip implementation of a NE2000 clone. Various vendors are now selling cards with this chip. See 5.26.2 for information on using any of these cards.

5.10 Boca Research

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

5.10.1 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.

Perspective buyers should be warned that many users have had endless problems with these cards. Owners of fast Pentium systems have been especially hit. Note that this is not a driver problem, as it hits DOS/Win/NT users as well. Boca's technical support number is (407) 241-8088, and you can also reach them at `75300.2672@compuserve.com`.

Donald did a comparitive test with the above Boca PCI card and a similar Allied Telsyn PCnet/PCI implementation, which showed that the problem lies in Boca's implementation of the PCnet/PCI chip. These test results can be accessed on Don's www server.

Linux at CESDIS `<http://cesdis.gsfc.nasa.gov/linux/>`

Also, Dave Platt has compared the recommended implementation given on the AMD data sheet with the Boca implementation, and has determined that Boca has left out a substantial number of important filtering capacitors. At the risk of being verbose, here is a quote from Dave, which will allow you to assess if the problem has been addressed yet, given a card for visual inspection.

"I just reviewed Appendix B in the 79c970 data sheet. It recommends a _minimum_ of 8 high-frequency bypassing caps (.1 uF multilayer ceramic) around the chip, to keep ground and power bounce from causing unreliable operation. Looking at the card I have here, I see a total of 5 such caps on the entire card - only a couple of them are close enough to the chip to do a decent job of high-frequency bypassing.

If you hold the card with chips facing up, and the PCI pins pointed towards you, the lower-left corner of the '970 has the positioning dimple. The upper-left corner of the chip is the 'analog corner', where the analog power and ground pins are. Appendix B calls these 'the most cricical pins in the layout of a PCnet-PCI card'. There are 4 analog power pins, and 2 analog ground pins... all of them are supposed to be connected, and properly bypassed.

On the left side of the chip, the uppermost pin is AVSS1 (analog ground 1). On the top edge, the fourth pin from the left is AVDD3 (analog power 3). The Appendix specifically recommends having a .1 uF bypass cap tied directly to these two pins (*not* going through the common ground plane). On the card I have, there is no such cap - the nearest bypass cap is about half-an-inch away over by the crystal, and is tied to the ground plane.

The Appendix also specifically recommends 'low-frequency bulk capacitors' (by which I assume that they mean multi-uF tantalum or aluminum electrolytics) as well as high-frequency bypass caps. I see only two bulk capacitors (one 10 uF in the AVSS2/AVDD2 filter circuit, and one 4.7 uF up above the chip which also appears to be part of a filter circuit). The Appendix recommends 'at least one low-frequency bulk (e.g. 22 uF) bypass capactor... connected directly to the power and ground planes.' There is no capacitor on the board which matches this description.

It appears that Boca ignored _several_ of AMD's recommendations, regarding the number, size, placement, and wiring of the power supply bypass capacitors."

(Thanks Dave.) Boca is offering a 'warranty repair' for affected owners, which involves adding one of the missing capacitors, but it appears that this fix doesn't work 100 percent for most people, although it helps some.

If you are *still* thinking of buying one of these cards, then at least try and get a 7 day unconditional return policy, so that if it doesn't work properly in your system, you can return it.

More general information on the AMD chips can be found in 5.4.1.

More technical information on AMD LANCE based Ethernet cards can be found in 8.7.

5.11 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.

5.11.1 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.

5.11.2 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. Module users can specify an `xcvr=N` value on the `insmod` command line to do the same.

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.

If you intend on using this driver as a loadable module you should probably see 10.2 and also 10.2.1 for module specific information.

5.12 Cogent

Here is where and how to reach them:

```
Cogent Data Technologies, Inc.
175 West Street, P.O. Box 926
Friday Harbour, WA 98250, USA.

Cogent Sales
15375 S.E. 30th Place, Suite 310
Bellevue, WA 98007, USA.

Technical Support:
```

```
Phone (360) 378-2929 between 8am and 5pm PST
Fax (360) 378-2882
Compuserve GO COGENT
Bulletin Board Service (360) 378-5405
Internet: support@cogentdata.com
```

5.12.1 EM100-ISA/EISA

Status – *Semi-Supported*

These cards use the SMC 91c100 chip and may work with the SMC 91c92 driver, but this has yet to be verified.

5.12.2 Cogent eMASTER+, EM100-PCI, EM400, EM960, EM964

Status – *Supported*

These are yet another DEC 21040 implementation that should work fine with the standard 21040 driver.

The EM400 and the EM964 are four port cards using a DEC 21050 bridge and 4 21040 chips.

See 5.17.4 for more information on these cards, and the present driver situation.

5.13 Compaq

Compaq aren't really in the business of making ethernet cards, but a lot of their systems have embedded ethernet controllers on the motherboard.

5.13.1 Compaq Deskpro / Compaq XL (Embedded AMD Chip)

Machines such as the XL series have an AMD 79c97x PCI chip on the mainboard that can be used with the standard LANCE driver. But before you can use it, you have to do some trickery to get the PCI BIOS to a place where Linux can see it. Frank Maas was kind enough to provide the details:

" The problem with this Compaq machine however is that the PCI directory is loaded in high memory, at a spot where the Linux kernel can't (won't) reach. Result: the card is never detected nor is it usable (sideline: the mouse won't work either) The workaround (as described thoroughly in http://www-c724.uibk.ac.at/XL/) is to load MS-DOS, launch a little driver Compaq wrote and then load the Linux kernel using LOADLIN. Ok, I'll give you time to say 'yuck, yuck', but for now this is the only working solution I know of. The little driver simply moves the PCI directory to a place where it is normally stored (and where Linux can find it)."

More general information on the AMD chips can be found in 5.4.1.

5.14 Danpex

5.14.1 Danpex EN9400

Yet another card based on the DEC 21040 chip, reported to work fine, and at a relatively cheap price.

See 5.17.4 for more information on these cards, and the present driver situation.

5.15 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
(416) 828-0260  in Canada
(02) 916-1600   in Taiwan
```

5.15.1 DE-100, DE-200, DE-220-T

Status – *Supported*

Some of the early D-Link cards didn't have the `0x57` PROM signature, but the ne2000 driver knows about them. For the software configurable cards, you can get the config program from `www.dlink.com`. The DE2** cards were the most widely reported as having the spurious transfer address mismatch errors with early versions of linux. Note that there are also cards from Digital (DEC) that are also named DE100 and DE200, but the similarity stops there.

5.15.2 DE-520

Status – *Supported*

This is a PCI card using the PCI version of AMD's LANCE chip. DMA selection and chip numbering information can be found in 5.4.1.

More technical information on AMD LANCE based Ethernet cards can be found in 8.7.

5.15.3 DE-530

Status – *Supported*

This is a generic DEC 21040 PCI chip implementation, and works with the generic 21040 driver.

See 5.17.4 for more information on these cards, and the present driver situation.

5.15.4 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 180kb/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.

5.15.5 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.

5.15.6 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 9.3 in this document, and if you can, have a look at:

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

5.16 DFI

5.16.1 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.)

5.17 Digital / DEC

5.17.1 DEPCA, DE100/1, 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 5.4.1 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.

This driver will also work on Alpha CPU based machines, and there are various ioctl()s that the user can play with.

5.17.2 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.

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`.

The next release of this driver (v0.40) will have Alpha CPU support like depca.c does and is available from David now if you require it.

5.17.3 DE425 (EISA), DE434, DE435, DE500

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 ;-) The DE500 uses the newer 21140 chip to provide 10/100Mbs ethernet connections. Have a read of the 21040 section below for extra info.

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

All the Digital cards will autoprobe for their media (except, temporarily, the DE500 due to a patent issue).

This driver is also ALPHA CPU ready and supports being loaded as a module. Users can access the driver internals through ioctl() calls - see the 'ewrk3' tools and the de4x5.c sources for information about how to do this.

5.17.4 DEC 21040, 21041, 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.

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.

The updated 21041 chip is also found in place of the 21040 on newly produced SMC EtherPower cards. The 21140 is for supporting 100Base-? and works with the Linux drivers for the 21040 chip. To use David's `de4x5` driver with non-DEC cards, have a look at `README.de4x5` for details.

Donald has used SMC EtherPower-10/100 cards to develop the 'tulip' driver. Note that the driver that is in the standard kernel tree at the moment is not the most up to date version. If you intend on using this driver, you should get the newest version from Donald's ftp/WWW site.

Tulip Driver `<http://cesdis.gsfc.nasa.gov/linux/drivers/tulip.html>`

The above URL also contains a (non-exhaustive) list of various cards/vendors that use the 21040 chip.

Also note that the tulip driver is still considered an *alpha* driver (see 3.1) 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.

5.18 Farallon

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

5.18.1 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.

5.19 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.

5.19.1 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.)

If you intend on using this driver as a loadable module you should probably see 10.2 and also 10.2.1 for module specific information.

5.19.2 HP PC Lan+ (27247, 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 during the v1.1.x development cycle. It can be operated in either a PIO mode like a ne2000, or a shared memory mode like a wd8013.

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 have 32K onboard RAM for Tx/Rx packet buffering instead of the usual 16KB, and they both offer LAN connector autosense.

If you intend on using this driver as a loadable module you should probably see 10.2 and also 10.2.1 for module specific information.

5.19.3 HP-J2405A

Status – *Supported*

These are lower priced, and slightly faster than the 27247/27252A, 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 8.7

5.19.4 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 5.4.1.

More technical information on LANCE based cards can be found in 8.7

5.19.5 HP 10/100 VG Any Lan Cards (27248B, J2573, J2577, J2585)

Status – *Supported*

As of early 1.3.x kernels, this driver was made available by Jaroslav Kysela, (perex@pf.jcu.cz). Due to the newness of the driver and the relatively small number of VG cards in use, feedback on this driver has been low.

Donald has also written a driver for these cards. Unlike the above, it is not presently in the standard kernel source tree. Check out the following URL for more information on Donald's 100VG work.

Donald's 100VG Page `<http://cesdis.gsfc.nasa.gov/linux/drivers/100vg.html>`

5.20 IBM / International Business Machines

5.20.1 IBM Thinkpad 300

Status – *Supported*

This is compatible with the Intel based Zenith Z-note. See 5.37.1 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/`

5.20.2 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 9.3 in this document, and if you can, have a look at:

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

5.20.3 IBM Token Ring

Status – *Semi-Supported*

To support token ring requires more than only 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.

Peter De Schrijver has been spending some time on Token Ring lately. and has worked with IBM ISA and MCA token ring cards.

The present token ring code has been included into the first of the 1.3.x series kernels.

Peter says that it was originally tested on an MCA 16/4 Megabit Token Ring board, but it should work with other Tropic based boards.

5.21 ICL Ethernet Cards

5.21.1 ICL EtherTeam 16i/32

Status – *Supported*

Mika Kuoppala (miku@pupu.elt.icl.fi) wrote this driver, and it was included into early 1.3.4x kernels. It uses the Fujitsu MB86965 chip that is also used on the at1700 cards.

5.22 Intel Ethernet Cards

5.22.1 Ether Express

Status – *Supported*

This card uses the intel i82586. (Surprise, huh?) Earlier versions of this driver (in v1.2 kernels) were classed as alpha-test, as it didn't work well for most people. The driver in the v2.0 kernel seems to work much better for those who have tried it. The comments at the top of the driver source list some of the problems associated with these cards.

There is also some technical information available on the i82586 in 8.5 and also in the source code for the driver 'eexpress.c'. Don't be afraid to read it. ;-)

5.22.2 Ether Express PRO/10

Status – *Supported*

Bao Chau Ha has written a driver for these cards that has been included into early 1.3.x kernels. It may also work with some of the Compaq built-in ethernet systems that are based on the i82595 chip.

5.22.3 Ether Express PRO/10 EISA/PCI

Status – *Not Supported*

These PCI and EISA cards use a different chip that the ISA PRO/10 cards do.

5.22.4 Ether Express PRO 10/100B

Status – *Supported*

A driver for this card is available on Donald's www/ftp site for v2.0 kernels. It is not included in the v2.0 kernel source tree, so you have to get it separately. Note that this driver will *not* work with the older 100A cards.

Drivers-Page `<http://cesdis.gsfc.nasa.gov/linux/drivers/>`

Apparently Donald had to sign a non-disclosure agreement that stated he could actually disclose the driver source code! How is that for sillyness on intel's part?

This driver will be included into the v2.1 source tree sometime in the future.

5.23 LinkSys

5.23.1 LinkSys Pocket Ethernet Adapter Plus (PEAEPP)

Status – *Supported*

This is supposedly a DE-620 clone, and is reported to work well with that driver. See 5.15.5 for more information.

5.23.2 LinkSys PCMCIA Adaptor

Status – *Supported*

This is supposed to be a re-badged DE-650. See 5.15.6 for more information.

5.24 Microdyne

5.24.1 Microdyne Exos 205T

Status – *Semi-Supported*

Another i82586 based card. Dirk Niggemann `dabn100@hermes.cam.ac.uk` has written a driver that he classes as "pre-alpha" that he would like people to test. Mail him for more details.

5.25 Mylex

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.
```

5.25.1 Mylex LNP101

Status – *Supported*

This is a PCI card that is based on DEC's 21040 chip. It is selectable between 10BaseT, 10Base2 and 10Base5 output. The LNP101 card has been verified to work with the generic 21040 driver.

See the section on the 21040 chip (5.17.4) for more information.

5.25.2 Mylex LNP104

Status – *Not Supported*

The LNP104 uses the DEC 21050 chip to deliver *four* independent 10BaseT ports. It may work with 21040 drivers, but nobody has tried yet.

5.25.3 Mylex M390B EISA

Status – *Not Supported*

Russ writes that "...it's a fairly vanilla 8390-based adapter. Only really tricky thing is that, like the Interlan es3210, you MUST do only 32-bit transfers to/from the board. Look at the Crynwr packet driver and try making the changes yourself..."

5.26 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.)

5.26.1 NE1000, NE2000

Status – *Supported*

NOTE: If you are using a kernel that is older than v1.2.9, it is *strongly* recommended that you upgrade to a newer version. There was an important bugfix made to the ne driver in 1.2.7, and another important bugfix made to the upper layers (dev.c) in 1.2.9. Both of these bugs can cause a ne2000 card to hang your computer.

The ne2000 is now a generic name for a bare-bones design around the NatSemi 8390 chip. 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 problems can arise with poor NE2000 clones. You should see 3.4, and 3.3

Some recently introduced NE2000 clones use the National Semiconductor 'AT/LANTic' 83905 chip, which offers a shared memory mode similar to the wd8013 and EEPROM software configuration. The shared memory mode will offer less CPU usage (i.e. more efficient) than the programmed i/o mode.

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 kindly to being prodded in the wrong areas, and will respond by locking your machine. Also `0x320` is bad because SCSI drivers probe into `0x330`.

Donald has written a NE2000 diagnostic program (ne2k.c) for all ne2000 cards. See 7.2 for more information.

If you intend on using this driver as a loadable module you should probably see 10.2 and also 10.2.1 for module specific information.

5.26.2 NE2000-PCI (RealTek-8029)

Status – *Supported*

Yes, believe it or not, people are making PCI cards based on the ten year old interface design of the ne2000. At the moment nearly all of these cards are based on the RealTek 8029 chip, and linux kernel v2.0 has support to automatically detect these cards at boot and use them.

If you have a NE2000 PCI card that is *not* based on the RealTek 8029 chip, please contact the maintainer of the NE2000 driver as listed in `/usr/src/linux/MAINTAINERS`. That way the ID of your card can also be added to the driver.

Note that you have to say 'Y' to the 'Other ISA cards' option when running `make config` as you are actually using the same NE2000 driver as the ISA cards use.

If you are using the driver as a module, you will have to supply the i/o address of the card (obtained from doing a `cat /proc/pci`) when loading the module. Note that this will not be necessary for future kernels.

5.26.3 NE1500, NE2100

Status – *Supported*

These cards use the original 7990 LANCE chip from AMD and are supported using the Linux lance driver. Newer NE2100 clones use the updated PCnet/ISA chip from AMD.

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 5.4.1.

More technical information on LANCE based cards can be found in 8.7

5.26.4 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.

5.27 Proteon

5.27.1 Proteon P1370-EA

Status – *Supported*

Apparently this is a NE2000 clone, and works fine with Linux.

5.27.2 Proteon P1670-EA

Status – *Supported*

This is yet another PCI card that is based on DEC's Tulip chip. It has been reported to work fine with Linux.

See the section on the 21040 chip (5.17.4) for more driver information.

5.28 Pure Data

5.28.1 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.

5.29 Racal-Interlan

Racal Interlan can be reached via WWW at `www.interlan.com`.

5.29.1 ES3210

Status – *Semi-Supported*

This is an EISA 8390 based shared memory card. An experimetal driver for v2.0 is available from me. It is reported to work fine, but the EISA IRQ and shared memory address detection currently is non-operational, so you have to supply them at boot; e.g. `ether=5,0,0xd0000,eth0` for IRQ 5 and shared memory at `0xd0000`. The i/o base is automatically detected and hence a non-zero value need not be supplied.

This driver will appear in the v2.1 kernels at some time in the future.

5.29.2 NI5210

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 3.1 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 8.5 for more technical information.

5.29.3 NI6510 (not EB)

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 3.1 for important information on using alpha-test ethernet drivers with Linux.

5.29.4 EtherBlaster (aka NI6510EB)

Status – *Supported*

As of kernel 1.3.23, the generic LANCE driver had a check added to it for the `0x52, 0x44` NI6510EB specific signature. Others have reported that this signature is not the same for all NI6510EB cards however, which will cause the lance driver to not detect your card. If this happens to you, you can change the probe (at about line 322 in lance.c) to printk() out what the values are for your card and then use them instead of the `0x52, 0x44` defaults.

The cards should probably be run in 'high-performance' mode and not in the NI6510 compatible mode when using the lance driver.

5.30 Sager

5.30.1 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 5.1.1 for all the reasons as to why you really don't want to use one of these cards.

5.31 Schneider & Koch

5.31.1 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, it appears as though this card won't work with the generic LANCE driver.

5.32 SEEQ

5.32.1 SEEQ 8005

Status – *Supported*

This driver was included into early 1.3.x kernels, and was written by Hamish Coleman. There is little information about the card included in the driver, and hence little information to be put here. If you have a question, you are probably best off e-mailing hamish@zot.apana.org.au

5.33 SMC (Standard Microsystems Corp.)

Please see 5.35 for information on SMC cards. (SMC bought out Western Digital's network card section quite a while ago.)

5.34 Thomas Conrad

5.34.1 Thomas Conrad TC-5048

This is yet another PCI card that is based on DEC's 21040 chip.

See the section on the 21040 chip (5.17.4) for more information.

5.35 Western Digital / SMC

The ethernet part of Western Digital has been bought out 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. They have separate drivers.

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
```

WWW Site: *SMC* `<http://www.smc.com>`

5.35.1 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. You can get the SMC setup/driver disks from SMC's ftp site. Note that some of the newer SMC 'SuperDisk' programs will fail to detect the real old EEPROM-less cards. The file `SMCDSK46.EXE` seems to be a good all-round choice. Also the jumper settings for all their 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.

5.35.2 WD8013, SMC Elite16

Status – *Supported*

Over the years the design has added more registers and an EEPROM. (The first wd8003 cards appeared about ten years ago!) Clones usually go by the '8013' name, and usually use a non-EEPROM (jumpered) design. Late model SMC cards will have the SMC 83c690 chip instead of the original Nat Semi DP8390 found on earlier cards. The shared memory design makes the cards a bit faster than PIO cards, 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)

Cards with a non standard amount of memory on board can have the memory size specified at boot (or at 'insmod' time if using modules). The standard memory size is 8kB for an 8bit card and 16kB for a 16bit card. For example, the older WD8003EBT cards could be jumpered for 32kB memory. To make full use of that RAM, you would use something like (for i/o=0x280 and IRQ 9):

```
LILO: linux ether=9,0x280,0xd0000,0xd8000,eth0
```

Also see 3.5 for some of the more common problems and frequently asked questions that pop up often.

If you intend on using this driver as a loadable module you should probably see 10.2 and also 10.2.1 for module specific information.

5.35.3 SMC Elite Ultra

Status – *Supported*

This ethercard is based on a new chip from SMC, the 83c790, which has a few new features. While it has a mode that is similar to the older SMC ethercards, it's not entirely compatible with the old WD80*3 drivers. However, in this mode it shares most of its code with the other 8390 drivers, while operating slightly 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.

Donald mentioned 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.

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.

SMC has acknowledged the problem occurs with Interactive, and older Windows NT drivers. It is a hardware conflict with early revisions of the card that can be worked around in the driver design. The current Ultra driver protects against this by only enabling the shared memory during data transfers with the card. Make sure your kernel version is at least 1.1.84, or that the driver version reported at boot is at least `smc-ultra.c:v1.12` otherwise you are vulnerable.

If you intend on using this driver as a loadable module you should probably see 10.2 and also 10.2.1 for module specific information.

5.35.4 SMC EtherEZ (8416)

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. Be sure to set your card for this compatibility mode. See the above information for notes on the Ultra driver.

For v1.2 kernels, the card had to be configured for shared memory operation. However v2.0 kernels can use the card in shared memory or programmed i/o mode. Shared memory mode will be slightly faster, and use considerably less CPU resources as well.

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

5.35.5 SMC EtherPower PCI (8432)

Status – *Supported*

These cards are 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 (aka `tulip.c`). Thanks to Duke Kamstra, once again, for supplying a card to do development on.

Some of the newer revisons of this card use the newer DEC 21041 chip, which may cause problems with older versions of the tulip driver. If you have problems, make sure you are using the latest driver release, which may not yet be included in the current kernel source tree.

See 5.17.4 for more details on using one of these cards, and the current status of the driver.

5.35.6 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). They are not very common.

5.35.7 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. They are not very common.

5.35.8 SMC-9000 / SMC 91c92/4

Status – *Supported*

The SMC9000 is a VLB card based on the 91c92 chip. The 91c92 appears on a few other brand cards as well, but is fairly uncommon. Erik Stahlman (erik@vt.edu) has written this driver which is in v2.0 kernels,

but not in the older v1.2 kernels. You may be able to drop the driver into a v1.2 kernel source tree with minimal difficulty.

5.35.9 SMC 91c100

Status – *Semi-Supported*

The SMC 91c92 driver is supposed to work for cards based on this 100Base-T chip, but at the moment this is unverified.

5.36 Xircom

For the longest time, Xircom wouldn't release the programming information required to write a driver, unless you signed your life away. Apparently enough linux users have pestered them for driver support (they claim to support all popular networking operating systems...) so that they have changed their policy to allow documentation to be released without having to sign a non-disclosure agreement, and apparently they will release the source code to the SCO driver as well. If you 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.

However, at the moment nobody has rushed forth offering to write any drivers, so all their products are still unsupported.

5.36.1 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).

5.37 Zenith

5.37.1 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 8.5 for more technical information. Also note that the IBM ThinkPad 300 is compatible with the Z-Note.

5.38 Znyx

5.38.1 Znyx ZX342 (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 Znyx cards) to work with this driver. Have a look at `README.de4x5` for details.

See 5.17.4 for more information on these cards, and the present driver situation.

5.39 Drivers for Non-Ethernet Devices

There are a few other drivers that are in the linux source, that present an *ethernet-like* device to network programs, while not really being ethernet. These are briefly listed here for completeness.

`dummy.c` - The purpose of this driver is to provide a device to point a route through, but not to actually transmit packets.

`eql.c` - Load Equalizer, enslaves multiple devices (usually modems) and balances the Tx load across them while presenting a single device to the network programs.

`ibmtr.c` - IBM Token Ring, which is not really ethernet. Broken-Ring requires source routing and other uglies.

`loopback.c` - Loopback device, for which all packets from you machine and destined for your own machine go. It essentially just moves the packet off the Tx queue and onto the Rx queue.

`pi2.c` - Ottawa Amateur Radio Club PI and PI2 interface.

`plip.c` - Parallel Line Internet Protocol, allows two computers to send packets to each other over two joined parallel ports in a point-to-point fashion.

`ppp.c` - Point-to-Point Protocol (RFC1331), for the Transmission of Multi-protocol Datagrams over a Point-to-Point Link (again usually modems).

`slip.c` - Serial Line Internet Protocol, allows two computers to send packets to each other over two joined serial ports (usually via modems) in a point-to-point fashion.

`tunnel.c` - Provides an IP tunnel through which you can tunnel network traffic transparently across subnets

`wavelan.c` - An Ethernet-like radio transceiver controlled by the Intel 82586 coprocessor which is used on other ethercards such as the Intel EtherExpress.

6 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 2.7 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 *Usenet FAQs* `<ftp://rtfm.mit.edu/pub/usenet-by-hierarchy/>` for the FAQ for that newsgroup.

6.1 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!

Note that there are a few cards out there with 'on-board termination'. These cards have a jumper which when closed, puts a 50 ohm resistor across the BNC input. With these cards, you can use a BNC T and terminator like normal, or put the cable directly onto the card and close the jumper to enable the on-board termination.

There are also some fancy cable systems which *look like* a single lead going to the card, but the lead is actually a loop, with the two runs of cable laying side-by-side covered by an outer sheath, giving the lead an oval shaped cross-section. At the turnaround point of the loop, a BNC connector is spliced in which connects to your card. So you have the equivalent of two runs of cable and a BNC T, but in this case, it is impossible for the user to remove a cable from one side of the T and disturb the network.

6.2 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

If you want to make a cable, the following should spell it out for you. Differential signal pairs must be on the same twisted pair to get the required minimal impedance/loss of a UTP cable. If you look at the above table, you will see that 1+2 and 3+6 are the two sets of differential signal pairs. Not 1+3 and 2+6 !!!!!! At 10MHz, with short lengths, you *may* get away with such errors, if it is only over a short length. Don't even think about it at 100MHz.

For a normal patch cord, with ends 'A' and 'B', you want straight through pin-to-pin mapping, with the input and output each using a pair of twisted wires (for impedance issues). That means 1A goes to 1B, 2A goes to 2B, 3A goes to 3B and 6A goes to 6B. The wires joining 1A-1B and 2A-2B must be a twisted pair. Also the wires joining 3A-3B and 6A-6B must be another twisted pair.

Now if you don't have a hub, and want to make a 'null cable', what you want to do is make the input of 'A' be the output of 'B' and the output of 'A' be the input of 'B', without changing the polarity. Tha means connecting 1A to 3B (out+ A to in+ B) and 2A to 6B (out- A to in- B). These two wires must be a twisted pair. They carry what card/plug 'A' considers output, and what is seen as input for card/plug 'B'. Then connect 3A to 1B (in+ A to out+ B) and also connect 6A to 2B (in- A to out- B). These second two must also be a twisted pair. They carry what card/plug 'A' considers input, and what card/plug 'B' considers output.

So, if you consider a normal patch cord, chop one end off of it, swap the places of the Rx and Tx twisted pairs into the new plug, and crimp it down, you then have a 'null' cable. Nothing complicated. You just want to feed the Tx signal of one card into the Rx of the second and vice versa.

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.)

6.3 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.

7 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. If you don't have the DOS software for your card, try looking on the WWW site of your card manufacturer. If you don't know the site name, take a guess at it, i.e. 'www.my_vendor.com' where 'my_vendor' is the name of your card manufacturer. This works for SMC, 3Com, and many *many* other manufacturers.

There are some cards for which Linux versions of the config utils exist, and they are listed here. 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 some 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.)

7.1 Configuration Programs for Ethernet Cards

7.1.1 WD80x3 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 SMC's ftp site. 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.

7.1.2 Digital / DEC Cards

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`.

7.1.3 NE2000+ or AT/LANTIC Cards

Some Nat Semi DP83905 implementations (such as the AT/LANTIC and the NE2000+) are software configurable. (Note that these cards can also emulate a wd8013 card!) You can get the file `/pub/linux/setup/atlantic.c` from Donald's ftp server, `cesdis.gsfc.nasa.gov` to configure this card. In addition, the configuration programs for the Kingston DP83905 cards seem to work with all cards, as they don't check for a vendor specific address before allowing you to use them. Follow the following URL: *Kingston Software* `<http://www.kingston.com/download/etherx/etherx.htm>` and get `20XX12.EXE` and `INFOSET.EXE`.

Be careful when configuring NE2000+ cards, as you can give them bad setting values which can cause problems. A typical example is accidentally enabling the boot ROM in the EEPROM (even if no ROM is installed) to a setting that conflicts with the VGA card. The result is a computer that just beeps at you (AMI beep eight times for VGA failure) when you turn it on and nothing appears on the screen.

You can typically recover from this by doing the following: Remove the card from the machine, and then boot and enter the CMOS setup. Change the 'Display Adapter' to 'Not Installed' and change the default boot drive to 'A:' (your floppy drive). Also change the 'Wait for F1 if any Error' to 'Disabled'. This way, the computer should boot without user intervention. Now create a bootable DOS floppy ('format a: /s /u') and copy the program `default.exe` from the `20XX12.EXE` archive above onto that floppy. Then type `echo default > a:autoexec.bat` so that the program to set the card back to sane defaults will be run automatically when you boot from this floppy. Shut the machine off, re-install the ne2000+ card, insert your new boot floppy, and power it back up. It will still probably beep at you, but eventually you should see the floppy light come on as it boots from the floppy. Wait a minute or two for the floppy to stop, indicating that it has finished running the `default.exe` program, and then power down your computer. When you then turn it on again, you should hopefully have a working display again, allowing you to change your CMOS settings back, and to change the card's EEPROM settings back to the values you want.

Note that if you don't have DOS handy, you can do the whole method above with a linux boot disk that automatically runs Donald's `atlantic` program (with the right command line switches) instead of a DOS boot disk that automatically runs the `default.exe` program.

7.1.4 3Com Cards

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.)

7.2 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 `dmesg` to see what useful info the kernel has on the card in question.

7.3 Identifying an Unknown Card

Okay, so your uncle's cousin's neighbour's friend had a brother who found an old ISA ethernet card in the AT case he was using as a cage for his son's pet hampster. Somehow you ended up with the card and want to try and use it with linux, but nobody has a clue what the card is and there isn't any documentation.

First of all, look for any obvious model numbers that might give a clue. Any model number that contains 2000 will most likely be a NE2000 clone. Any cards with 8003 or 8013 on them somewhere will be Western/Digital WD80x3 cards or SMC Elite cards or clones of them.

7.3.1 Identifying the Network Interface Controller

Look for the biggest chip on the card. This will be the network controller (NIC) itself, and most can be identified by the part number. If you know which NIC is on the card, the following might be able to help you figure out what card it is.

Probably still the most common NIC is the National Semiconductor DP8390 aka NS32490 aka DP83901 aka DP83902 aka DP83905 aka DP83907. And those are just the ones made by National! Other companies such as Winbond and UMC make DP8390 and DP83905 clone parts, such as the Winbond 89c904 (DP83905 clone) and the UMC 9090. If the card has some form of 8390 on it, then chances are it is a

ne1000 or ne2000 clone card. The second most common 8390 based card are wd80x3 cards and clones. Cards with a DP83905 can be configured to be an ne2000 *or* a wd8013. Never versions of the genuine wd80x3 and SMC Elite cards have an 83c690 in place of the original DP8390. The SMC Ultra cards have an 83c790, and use a slightly different driver than the wd80x3 cards. The SMC EtherEZ cards have an 83c795, and use the same driver as the SMC Ultra. All BNC cards based on some sort of 8390 or 8390 clone will usually have an 8392 (or 83c692, or XXX392) 16 pin DIP chip very close to the BNC connector.

Another common NIC found on older cards is the Intel i82586. Cards having this NIC include the 3c505, 3c507, 3c523, Intel EtherExpress-ISA, Microdyne Exos-205T, and the Racal-Interlan NI5210.

The original AMD LANCE NIC was numbered AM7990, and newer revisions include the 79c960, 79c961, 79c965, 79c970, and 79c974. Most cards with one of the above will work with the Linux LANCE driver, with the exception of the old Racal-Interlan NI6510 cards that have their own driver.

Newer PCI cards having a DEC 21040, 21041, 21140, or similar number on the NIC should be able to use the linux tulip or de4x5 driver.

Other PCI cards having a big chip marked RTL8029 are ne2000 clone cards, and the ne driver in linux version v2.0 and up should automatically detect these cards at boot.

7.3.2 Identifying the Ethernet Address

Each ethernet card has its own six byte address that is unique to that card. The first three bytes of that address are the same for each card made by that particular manufacturer. For example all SMC cards start with `00:00:c0`. The last three are assigned by the manufacturer uniquely to each individual card as they are produced.

If your card has a sticker on it giving all six bits of its address, you can look up the vendor from the first three. However it is more common to see only the last three bytes printed onto a sticker attached to a socketed PROM, which tells you nothing.

You can determine which vendors have which assigned addresses from RFC-1340. Apparently there is a more up to date listing available in various places as well. Try a WWW or FTP search for `EtherNet-codes` or `Ethernet-codes` and you will find something.

7.3.3 Tips on Trying to Use an Unknown Card

If you are still not sure what the card is, but have at least narrowed it down some, then you can build a kernel with a whole bunch of drivers included, and see if any of them autodetect the card at boot.

If the kernel doesn't detect the card, it may be that the card is not configured to one of the addresses that the driver probes when looking for a card. In this case, you might want to try getting `scanport.tar.gz` from your local linux ftp site, and see if that can locate where your card is jumpered for. It scans ISA i/o space from 0x100 to 0x3ff looking for devices that aren't registered in `/proc/ioports`. If it finds an unknown device starting at some particular address, you can then explicity point the ethernet probes at that address with an `ether=` boot argument.

If you manage to get the card detected, you can then usually figure out the unknown jumpers by changing them one at a time and seeing at what i/o base and IRQ that the card is detected at. The IRQ settings

can also usually be determined by following the traces on the back of the card to where the jumpers are soldered through. Counting the 'gold fingers' on the backside, from the end of the card with the metal bracket, you have IRQ 9, 7, 6, 5, 4, 3, 10, 11, 12, 15, 14 at fingers 4, 21, 22, 23, 24, 25, 34, 35, 36, 37, 38 respectively. Eight bit cards only have up to finger 31.

Jumpers that appear to do nothing usually are for selecting the memory address of an optional boot ROM. Other jumpers that are located near the BNC or RJ-45 or AUI connectors are usually to select the output media. These are also typically near the 'black box' voltage converters marked YCL, Valor, or Fil-Mag.

A nice collection of jumper settings for various cards can be found at the following URL:

Ethercard Settings `<ftp://ftp.syd.dit.csiro.au/pub/ken/NIC/index.html>`

8 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.

8.1 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

8.2 Writing a Driver

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). 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.

(Ping-pong Tx buffers means using at least 2 max-size packet buffers for Tx packets. One is loaded while the card is transmitting the other. The second is then sent as soon as the first finished, and so on. In this way, most cards are able to continuously send back-to-back packets onto the wire.)

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 should start with 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 in all recent kernels. Also have a look at the Kernel Hackers Guide, at the following URL: *KHG* <http://www.redhat.com:8080/HyperNews/get/khg.html>

8.3 Driver interface to the kernel

Here are some notes on the functions that you would have to write if creating a new driver. Reading this in conjunction with the above skeleton driver may help clear things up.

8.3.1 Probe

Called at boot to check for existence of card. Best if it can check un-obtrsively by reading from memory, etc. Can also read from i/o ports. Initial writing to i/o ports in a probe is *not good* as it may kill another device. Some device initialization is usually done here (allocating i/o space, IRQs,filling in the dev->??? fields etc.) You need to know what io ports/mem the card can be configured to, how to enable shared memory (if used) and how to select/enable interrupt generation, etc.

8.3.2 Interrupt handler

Called by the kernel when the card posts an interrupt. This has the job of determining why the card posted an interrupt, and acting accordingly. Usual interrupt conditions are data to be rec'd, transmit completed, error conditions being reported. You need to know any relevant interrupt status bits so that you can act accordingly.

8.3.3 Transmit function

Linked to dev->hard_start_xmit() and is called by the kernel when there is some data that the kernel wants to put out over the device. This puts the data onto the card and triggers the transmit. You need to know how to bundle the data and how to get it onto the card (shared memory copy, PIO transfer, DMA?) and in the right place on the card. Then you need to know how to tell the card to send the data down the wire, and (possibly) post an interrupt when done. 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)`.

8.3.4 Receive function

Called by the kernel interrupt handler when the card reports that there is data on the card. It pulls the data off the card, packages it into a sk_buff and lets the kernel know the data is there for it by doing a netif_rx(sk_buff). You need to know how to enable interrupt generation upon Rx of data, how to check any relevant Rx status bits, and how to get that data off the card (again sh mem, PIO, DMA, etc.)

8.3.5 Open function

linked to dev->open and called by the networking layers when somebody does `ifconfig eth0 up` - this puts the device on line and enables it for Rx/Tx of data. Any special initialization incantations that were not done in the probe sequence (enabling IRQ generation, etc.) would go in here.

8.3.6 Close function (optional)

This puts the card in a sane state when someone does `ifconfig eth0 down`. It should free the IRQs and DMA channels if the hardware permits, and turn off anything that will save power (like the transceiver).

8.3.7 Miscellaneous functions

Things like a reset function, so that if things go south, the driver can try resetting the card as a last ditch effort. Usually done when a Tx times out or similar. Also a function to read the statistics registers of the card if so equipped.

8.4 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.

8.5 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.'

8.6 Technical information from 3Com

If you are interested in working on drivers for 3Com cards, you can get technical documentation from 3Com. Cameron has been kind enough to tell us how to go about it below:

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.

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
```

8.7 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.

8.8 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.

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 8.9 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 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.'

8.9 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.

9 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.

9.1 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.

9.2 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.

9.3 PCMCIA Support

As this area of Linux development is fairly young, I'd suggest that you join the LAPTOPS mailing channel. See 10.3 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.

9.4 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.

9.5 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 5.36) 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 5.15.4 and 5.9.1 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 5.9.1)

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 example, 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 usually ignored.

For more information on (and a complete list of) boot time arguments, please see the *BootPrompt-HOWTO* <http://sunsite.unc.edu/mdw/HOWTO/BootPrompt-HOWTO.html>

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 unusually complicated request.

10.2 Using the Ethernet Drivers as Modules

See the `insmod(8)` manual page for information on passing arguments to the module as it is being loaded. The command `lsmod` will show you what modules are loaded, and `rmmod` will remove them.

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.

Most modules accept parameters like `io=0x340` and `irq=12` on the `insmod` command line. It is *STRONGLY ADVISED* that you supply these parameters to avoid probing for the card. Unlike PCI and EISA devices, there is no real safe way to do auto-probing for ISA devices, and so it should be avoided when using drivers as modules.

A list of all the parameters that each module accepts can be found in the file:

`/usr/src/linux/Documentation/networking/net-modules.txt`

It is recommended that you read that to find out what options you can use for your particular card.

Once you have figured out the arguments/options you are going to use, you can insert the module by typing as root:

```
insmod mod_name.o [io=val1[,val2,...]] [irq=val7[,val8,...]]
```

The comma separated value lists are used for modules that have the capability to handle multiple devices from a single module, such as all the 8390 drivers, and the PLIP driver.

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 module(s).

10.2.1 8390 Based Cards as Modules

The present list of 8390 based drivers is: 3c503, ac3200, e2100, hp, hp-plus, ne, smc-ultra and wd. These cards were not supported as modules for kernel versions prior to 1.3.42. (This does not include some of the separately distributed PCMCIA drivers (e.g. de-650) that are also 8390 based, that have had module support for quite some time now.)

If you have an 8390 based card, you may have to insert *two* modules, 8390.o and then the module for your card. If 8390 support has been built into your kernel, then you will not need to insert the 8390 module. (8390 support is built in whenever an 8390 based card is selected to be built into the kernel.) Doing a `cat /proc/ksyms | grep 8390` will tell you if 8390 support is in your kernel.

For 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, and thus marked as *busy*.

The 8390 series of network drivers now support multiple card systems without reloading the same module multiple times (memory efficient!) This is done by specifying multiple comma separated values, such as:

```
insmod 3c503.o io=0x280,0x300,0x330,0x350 xcvr=0,1,0,1
```

The above would have the one module controlling four 3c503 cards, with card 2 and 4 using external transcievers.

It is *STRONGLY RECOMMENDED* that you supply "io=" instead of autoprobing. If an "io=" argument is not supplied, then the ISA 8390 drivers will complain about autoprobing being not recommended, and begrudgingly autoprobe for a *SINGLE CARD ONLY* – if you want to use multiple cards you *have* to supply an "io=0xNNN,0xQQQ,..." argument.

The ne module is an exception to the above. A NE2000 is essentially an 8390 chip, some bus glue and some RAM. Because of this, the ne probe is more invasive than the rest, and so at boot we make sure the ne probe is done last of all the 8390 cards (so that it won't trip over other 8390 based cards) With modules we can't ensure that all other non-ne 8390 cards have already been found. Because of this, the ne module *REQUIRES* an `io=0xNNN` argument passed in via insmod. It will *refuse* to autoprobe.

It is also worth noting that auto-IRQ probably isn't as reliable during the flurry of interrupt activity on a running machine. Cards such as the ne2000 that can't get the IRQ setting from an EEPROM or configuration register are probably best supplied with an `irq=M` argument as well. The file

`/usr/src/linux/Documentation/networking/net-modules.txt`

also lists how the interrupt settings are determined for the various cards if an `irq=N` value is not given.

10.3 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 mailing lists by sending mail to `majordomo@vger.rutgers.edu` to get help with what lists are available, and how to join them.

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.setup 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.

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.

10.4 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. Also, 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 or two in there describing any important points before any actual code starts..

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** that is posted regularly to the newsgroup *comp.dcom.lans.ethernet*. You can grab it from RTFM which holds all the newsgroup FAQs at the following URL:

Usenet FAQs <ftp://rtfm.mit.edu/pub/usenet-by-hierarchy/>

You can also have a look at the 'Ethernet-HomePage' so to speak, which is at the following URL:

Ethernet-HomePage <http://wwwhost.ots.utexas.edu/ethernet/ethernet-home.html>

10.5 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.

10.6 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.

This document is Copyright (c) 1993, 1994, 1995 1996 by Donald Becker and Paul Gortmaker. Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this document under the conditions for verbatim copying, provided that this copyright notice is included exactly as in the original, and that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this document into another language, under the above conditions for modified versions.

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.

10.7 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`