University of Florida's Mach I386 Installation Notes

Carnegie Mellon University
Pittsburgh, PA 15213

Edited for the University of Florida
by
Philip A. Kufeldt
Engineering Computing Services
College of Engineering
pak@eng.ufl.edu

May 3, 1992

1 Introduction

The Mach i386 release is a complete system release that comes with a "Mach 2.5" macro kernel. This system has a newer kernel than the Vax, Sun3 and IBMRT platforms and a system environment that is closer to BSD 4.3-Tahoe. This system can be used to build the Mach 3.0 micro-kernel and Unix server. Most of the user level programs provided in this release will run with either the Mach 2.5 kernel or the Mach 3.0 kernel plus Unix server.

The programs and libraries come from the 4.3BSD-TAHOE release. Additionally, all of the Mach specific programs, include files, and libraries (libmach and libthreads) are provided. The CMUCS library and some of the CMUCS programs (most noteably rfs and sup) are included. This release does not support NFS. Lastly, since the Mach 2.5 kernel does not support file system labels, most file system tools are CMU tools derived from 4.3BSD. This system has been distributed during the 1990 calendar year.

All the releases that are made by UF's College of Engineering are done only by an electronic transfer protocol called SUP (Software Update Protocol). If your site is not on the university network then call UFNET.

This document assumes a working knowledge of the installation and maintainence of a 4.3BSD system. If you are unfamiliar with the 4.3BD system, you should read the system maintainers documentation for 4.3BSD.

2 Hardware Requirements

We only support machines with:

- Either an i386 or i486 processor

- at least 5 Meg of memory. We strongly suggest use 8Meg - 16Meg.
- an AT compatible bus.
- WD 1007 compatible disk controllers. You should get at least 100Meg of disk. We use 300Meg ESDI drives but ST506 drives work [although slowly]. IDE drives work as well.
- E,C,VGA displays in 25 x 80 character mode. X11R5 support for wide variety of color vga adapters at resolutions up to 1024x768.
- high density $3\frac{1}{2}$" and/or $5\frac{1}{4}$" floppies at 1.44Meg and 1.2Meg, respectively.
- several ethernet boards: Intel's iMX586 or PC586/ENET586 ethernet board, 3COM's EtherLink I and EtherLink II (sometimes known as 3c501 and 3c503), and Western Digital's WD8003E and WD8013E ethernet boards.

We support but do not require:

- Intel 80387 hardware floating point. (recommended)
- The built-in serial line: COMi
- The built-in parallel port: LPTi
- Wangtek 1/4" standard 3M streamer.
- Adaptec SCSI adapters and devices.

We Do Not Support:

- The microchannel bus: IBM PS 2/70 and PS 2/80 and some NCR machines

Note: Engineering Computing Services provides this Mach release in electronic form only. Hence, the requirement of an ethernet card. equipment.

3 The Bootstrap Disks

This distribution is intended to be loaded onto an empty disk; it contains all the binaries and sources that are needed. The Mach bootstrap kit consists of a pair of High Density (HD) floppy disks which can be anonymously ftp from eng.ufl.edu. In pub/mach/installation there are two files: disk1 and disk2. These two files are `dd`'ed $3\frac{1}{2}$" high density floppy images. At a nominal cost, ECS will provide these floppies if they cannot be retrieved from the net.

The first disk contains a bootstrap program and a copy of the *Mach Kernel*. The second disk contains a file system with Unix binaries. The file system disk is intended to be used not only during the boot process, but also to recover from problems that might prevent you from being able to boot off the hard disk.

The second disk contains (approximately):

```
UFMACHBASE

bin:
[               dd              ls              setup           tar
cat             echo            mkdir           setup.new
chmod           ed              mv              sh
cp              hostname        pwd             stty
date            ln              rm              sync

dev:
MAKEDEV and devices

etc:
badblocks       fdisk           ifconfig        passwd          route
badsect         fsck            init            rc              services
clri            group           mkfs            reboot          termcap
diskutil        halt            mknod           resolv.conf     umount
dump            hosts           mount           rfshost         vtoc
restore

mnt:
tmp:

usr/bin:
        rfs     sup

usr/games:
        rain

usr/ucb:
        ftp     telnet
```

The installation process presumes that you are going to boot your target i386 machine from the floppy disks that we have provided and then transfer the files from ECS onto the target i386. In this document, we presume that you are going to access the Mach binaries and sources via the campus network. `tar` and `dump/restore` are also provided, here, to allow you to copy the sources into some other local machine and then to transfer them to the i386 machine by tape, floppies or lan access.

You will be requesting binaries and sources from a database machine at ECS using the sup database update program/protocol. (See the sup manual page). Your machine will authenticate to the database machine; it must explicitly supply an encryption key (CRYPT) in the procotol negotiations.

To get this distribution you must first contact ECS and register your new mach host. At this time the point of contact for this new machine will be added to a campus wide mach mailing list. It is on this list that updates and information dealing with Mach on campus will occur. If you would like there is also a national mailing expressly for Mach on i386's. To register, send a request to machi386-request@mach.cs.cmu.edu.

4 Before You SUP

4.1 Low Level Disk Formatting

You must perform a *low level format* of your hard disk. This procedure partitions the disk into sectors and then looks for bad areas of the disk. It usually takes several hours. Some manufacturers (HP, Toshiba, ...) ship the disks preformatted. Also, some computer stores will format the disk for you before they ship the system.

The documentation you got with your disk or machine or DOS will tell you how to format your disk. The procedures vary widely and are beyond the scope of this discussion. Please do not call us if you are having problems with this step.

4.2 Manual Pages

You should acquaint yourself with the attached manual pages for the programs: diskutil, fdisk, vtoc, badblocks and sup.

4.3 Booting to Single User mode

In the guide below, we presume that nothing will go wrong. In our experience, 80common set of pitfalls. We annotate problem areas with superscripts; these numbers refer to comments in Appendix I.

We will presume for the discussion below that you are performing a "simple" installation. In particular, that you do not want to preserve part of the disk for other systems (like DOS), you do not have excessive bad blocks in inopportune places (where the system would put the bad block table) and that you do not have to override the disk geometry parameters that are maintained internal to BIOS. We will discuss all these matters in the section DONT CLOBBER MY DISK.

You use the floppies to bring up Mach 2.5 in single user mode. Insert the *Mach Kernel* disk (disk 1)and boot (or power toggle) the machine. The system should

1. Print out a 8 hexadecimal digits

2. It then should type `boot:`

   At this prompt, you type a carriage return (or enter). Then, you will see a number printed out, a pause, a `+`, etc. The bootstrap is reading in the kernel.

   Shortly, you will be instructed to `insert file system`. At this point, you remove the *Mach Kernel* disk and insert the *Mach file system* disk (disk 2). Once you have switched floppies, you proceed with the boot process by typing a carriage return (or enter).

3. Next, you will see several lines of printout; the Mach kernel is describing the hardware that it finds.

   Then you will see a lot of activity on the floppy as fsck is run. And finally, you will get the `#` prompt, indicating that you are executing at single user level and may now type commands.

You might want to do an `ls -R` at this point and play with the system a bit. **Note:** operating/loading from floppies is rather slow and not indicative of the true system performance.

4.4 Initializing your Disk

Now you use the Mach procedures on the installation floppy to initialize your hard disk.

type

```
diskutil clobber_my_disk
```

This initializes a disk for use under Mach. It will destroy all/any data on the disk. If you do not wish to trash all the data on your disk, read the section "DONT CLOBBER MY DISK". On an ESDI disk, this procedure takes about 15 minutes per 100Meg. On an ST506 disk, it is much slower. Note: when this operation completes you do not have file systems; you must run the setup program (below).

5 Installation and System Setup

5.1 Supping files

Installing Mach is a simple cook book process. You follow the recipe below to bootstrap over the internet.

1. Type

   ```
   setup -init
   ```

   Called with `-init` argument `setup` will create a *root* file system and mount it under /mnt. It will then create a *usr* file system and mount it under /mnt/RFS/.LOCALROOT/usr. The pathname RFS/.LOCALROOT is explained further in the section "Root, Super-root and RFS". If later on, the system needs to be rebooted from the floppies and you do not wish to re–initialize the disk then,

   ```
   setup -mount
   ```

   The `-mount` argument causes the root and usr file systems to be mount under /mnt as above without re–initializing the disk.

2. Type

   ```
   setup -net <machine name> <internet address> [<netmask> <broadcast>]
   setup -gateway <internet_gateway>
   ```

Now you should be on the network. You might try to telnet by IP address somewhere, to verify that your network connection is active. The `-gateway <internet gateway>` is necessary for your machine/site to find a path to the ECS mach server. At this time you might also want to edit the resolv.conf file in the /etc directory to identify the nameserver nearest you. This is not strictly necessary.

3. Type

   ```
   setup -sup
   ```

   This invokes sup on the control file /UFMACHBASE . Your machine will be requesting from ECS all the binaries for the i386 distribution. On our local ethernet, this takes about 1/2 hour; over the internet, we have seen times in the 2 to 3 hour range. If there are failures, just restart this step by retyping

   ```
   setup -sup
   ```

   Repeat this until everything completes. The sup protocol records what it has successfully brought over from previous attempts and transfers any files still required. When sup has successfully completed there will be the following three directories mach.first, mach.root, and mach.usr in /mnt/RFS/.LOCALROOT/sup.

4. Before booting from your hard disk,

   ```
   cd /mnt/RFS/.LOCALROOT/etc
   ```

   and edit the file `rc.custom` (use ed as the editor). Fill in your machine name, ip address and gateway in the appropriate places. You also should verify that your machine name and ip address are in the `hosts` file.

5. Now it is time to reboot off the hard disk. Type

   ```
   reboot
   ```

   After you get the message "syncing disks ...", remove the floppy disk or you will attempt to boot off the floppy again — not the hard disk.

6. Note that the template passwd file allows root to login with no passwd. This is so that you will be able to add yourself as a user and change the root passwd after you boot off the hard disk. You should not leave the root without a password after you have set up your machine.

7. The floppy devices may be wrong for the type of drive on your machine. They are setup for 5 1/4" floppies via the installation process. You make the devices for the 3 1/2" floppy by typing

```
cd /dev
rm -f *floppy *fd0
./MAKEDEV fh0
```

or for the 5 1/4" floppy by

typing

```
cd /dev
rm -f *floppy *fd0
./MAKEDEV fd0
```

MAKEDEV fh1 or MAKEDEV fd1 can be used to make devices for a second disk drive. It will be known by the names fd1 and rfd1.

8. Edit all the site dependent BSD files below. You should look at the BSD installation manual for exact details.

   - /etc/rc.custom
   - /etc/hosts
   - /etc/motd
   - /etc/termcap
   - /etc/sendmail.cf,fc
   - /etc/passwd
   - /etc/group
   - /etc/resolv.conf
   - /etc/printcap - if you have printers

5.2 The Environment

Mach 2.5 I386 has the same kernel features as the Mach 2.5 for all the other supported platforms. The software environment starts with 4.3BSD-TAHOE environment, with one major exception. The Mach 2.5 kernel does not support file system labels; so most file system tools are CMU tools derived from 4.3BSD.

You are also provided with all the Mach programs, include files and libraries and the CMUCS library and some programs (most notably rfs and sup). However, these files are not in the /usr/mach tree or /usr/cs tree as is the case with Mach on other hardware, but are integrated into the BSD tree. (Eventually all Mach systems will have a merged tree.)

The compile environment/tools, i.e. cc, as, ld, size, nm, strip, ar, and gdb all come from the GNU products of the Free Software Foundation (FSF). There is one exception, cpp; we use a CMU modified 4.3BSD cpp. All of the program and library sources have been compiled without using the "traditional" switch and only a few programs needed "writable-strings". You should be aware of what these two features do; see the gcc manual page. You may need them to successfully port your applications.

Finally, one feature about the load/execution function should be noted. Program text starts at 0x10000 (64K). This means that you can not dereference a pointer to 0 or any small integer value. This will cause programs to core dump that were "accidentally" working on other architectures. Only a small number of 4.3BSD-TAHOE programs that had to be fixed to deal with this. (SunOS and Mach on the Sun Architecture machines have a similar restriction.)

5.3 BSD4.3-TAHOE Source Compilation

Since the system sources, /usr/src, are basically 4.3BSD-TAHOE, the BSD standard compilation procedures applies. The typical way to build a program is to go to the appropriate directory and type make. You install it by typing make install. Note: If the program ¡foo¿.cmucs exists, it was used in preference to the original BSD program ¡foo¿. Also, the compile tools mentioned above, should be taken from /usr/src/gnu; the BSD version has been left for reference purposes.

5.4 Building Kernels

To build a new kernel, go to the kernel source tree. Then type make CONFIG=STD+WS-afs-nfs doconf config This will create a directory, STD+WS-afs-nfs. You can go into this directory and type ./make. This will initiate a standard BSD style kernel build.

5.5 Installing/Booting Kernels

The kernel that you have built should be placed on the root of the file system. Give it some name other than /mach [or /vmunix], initially, so you can fall back to the old kernel if necessary. To invoke your kernel type

```
/etc/reboot
```

The old kernel will shutdown, then the hardware performs self test, and lastly, you are prompted for a kernel to boot. The system types out:

```
boot:
```

If you type nothing, mach is booted. (If you type carriage return (or enter), mach is also booted.) You may type a file name of a kernel to boot. If you had put your kernel in the file /mach.new, you would type

```
/RFS/.LOCALROOT/mach.new
```

to boot this kernel. Note: From the section on Root, Super-root and RFS, you should understand why the /RFS/.LOCALROOT is necessary. (Aside: you may also supply flags on the boot line, above. -s causes the system to boot "single user". -d causes symbols for the debugger to be loaded.)

Once you are happy with your kernel, you want to install it as /mach. The old /mach has a link count of three; the three files /mach, /vmunix and /../../mach have to be linked together. You also want to save the old kernel. My preference for doing this is to:

```
cp -p /mach /mach.old
cp -p /mach.new /mach
```

This way after you set up the links once and never worry about them again. Alternatively, you might

```
rm -f /mach.old  /../../mach.old /../../vmunix.old

mv /mach /mach.old
mv /vmunix /vmunix.old
mv /../../mach /../../mach.old

mv /mach.new /mach
ln /mach /vmunix
ln /mach /../..
```

5.6 Root, Super-Root and RFS

Since 1982, CMUCS kernels have had a facility that lets you access files and devices on other machines. We call this the remote file system, RFS. The client side exists in every Mach kernel; the server is a privileged program, rfsd, that is run on each machine that will permit access to its local files. We have even ported the server to non-Mach operating systems.

RFS refers to remote files/devices in a very location dependent way. The remote file name should not look strange to Mach/Unix programs that parse file names. Thus, the generic name is /../¡remote machine name¿/¡path on that machine¿ for example /../wb1/etc/passwd /../wb1/dev/rmt0 What this implies is that in Mach, the root file system, "/", is not inode 2. The kernel automatically "chroots" (changes the root) to /RFS/.LOCALROOT as it boots. Thus "/.." and "/../.." are well defined and not equal to "/".

The local machine must have an "RFS link" to the remote machine. To make a link in /.., type:

```
cd /..
rfshost <IP ADDRESS> <host name> <host nic names>
```

For example:

```
rfshost 128.2.250.16 wb1.cs.cmu.edu wb1
```

or even

```
rfshost `grep wb1.cs.cmu.edu /etc/hosts`
```

You may create as many entries as you like and as many aliases as you like. A special quoting mechanism is necessary to remove an RFS link. If you are in the directory containing the RFS link, link,

```
rm .///<link>
```

will remove the RFS link, link.

By default, RFS accesses files on the server using the user id, rfsd, and group id, system. However, the server must be told that your machine may access its files. The file /etc/rfsd-hosts, on the server, should contain single lines indicating the internet address of machines allowed to contact this machine. (If you change the file, you must kill and restart rfsd.)

You can also authenticate yourself to RFS, using the program, rfs. rfs will prompt you for a user, group, account, password, and "Identify?" Give the default answer for account and "Identify". After running rfs, you will have a shell that will allow you to access files on any remote RFS hosts using the specified uid and group. Note: you must create a local magic file for authentication to work. You must type

```
cd /..
rfshost 0.0.0.0 CONTROL
```

5.7 Paging

The kernel pages out to inodes. There is no need for special dedicated paging area, but there must be free space on the disk to page out to. The program /etc/swapon can be run at boot time to specify partitions to prefer to swap to and those to never swap to. (see mach_swapon.8)

5.8 Changing Time Zones

As distributed the time zone is set to eastern standard time, i.e. EST5EDT. If you need to change time zones you will need to perform two tasks. First, you will need to re-link /etc/zoneinfo/localtime as follows (do this as root):

```
cd  /etc/zoneinfo
rm  localtime
ln  xxx  localtime   (substitute appropriate timezone file for xxx)
```

Second, you will need to patch time zone variable (i.e. _tz) in the kernel.

Prior to doing this you will need to calculate the appropriate patch value in hex of the difference in minutes between your local zone time and GMT (universal time). The value of this constant for eastern standard time, for example is x12c hex (i.e. 300 decimal). This is the value that you will initially see for _tz. To change to Pacific Daylight Time, as an example, _tz should be patched to x1e0 hex (i.e. 480 decimal).

5.9 The Kernel Debugger

Mach has a version of "adb" in the kernel. The key sequence "control-alt d" will enter the kernel debugger. As you would expect, ":c" gets you back. Once in the kernel debugger, you are on your own. You might look at the BSD documentation on adb and kdb for a list of the commands. A laconic description of the debugger may be found in Appendix V. Symbols have to be loaded with the kernel, for kdb to be able to use them. If the kernel file is executable (determined by the file's permission bits), the boot program always loads the symbols. The boot line flag, -d, will force symbols to be loaded.

6 DONT CLOBBER MY DISK

6.1 Minor Variations

Three simple cases of disk setup arise that can still be handled by diskutil clobber_my_disk. In the first case, you might need/want to override what BIOS believes the disk geometry to be. BIOS allows only 10 bits of cylinder numbers. If you have more, either you can not use them, or you have to lie about the number of sectors per track and number of heads so that the "virtual" number of cylinders falls under 1024. If you specify the "-p" option to diskutil clobber_my_disk, it will interactively prompt you for the geometry information.

In the second case, you have bad blocks in what would be the bad block area. diskutil clobber_my_disk will abort very early on. You fix this by moving the start of the "Mach (BIOS) partition" forward (see section The Disk Layout below). You type:

```
diskutil clobber_my_disk -o <offset in sectors>
```

The Mach (BIOS) partition will start at ¡offset in sectors¿ on the disk. This quantity should be an integral number of cylinders. **NOTE:** You specify the number in SECTORS.

In the last case, you might have a stupid low level format program which marks all the sectors of a track bad if any sector is bad. diskutil clobber_my_disk has a heuristic; it looks for several sector errors before it labels the track as bad. This will cause a lot of needless rattling of the disk (error retries and recalibration), before the track is marked bad by diskutil clobber_my_disk. The "-t" option to diskutil clobber_by_disk (and badblocks) will cause the program to mark the entire track bad if there is one bad sector.

Lastly, you can combine any or all of the options described above.

6.2 Room Enough for DOS

If you are starting out with an emtpy disk and want to leave room for a DOS partition at the front of the disk, there is a very easy way to do this. Use

```
diskutil clobber_my_disk -o <offset in sectors>
```

as noted in the previous section and specify the number of sectors you want to leave for the DOS paritition. (This should be a integral number of cylinders.)

Next, you boot up DOS from a floppy and run DOS's fdisk to create a primary DOS partition that uses ONLY the initial cylinders of the disk that you have set aside. Then you run DOS format to format the DOS c drive. Lastly, you populate the c drive. **NOTE:** in our experience fdisk from DOS 3.3 is very easy to allocate the right number of cylinders. DOS 4.0 fdisk is harder to use for this purpose.

If DOS partitions are already on the disk, or you need several, the next section will explain what you need to do.

6.3 The Disk Layout

At the start of the disk (sector 0), there is a BIOS boot block that has code and a partition table. (**NOTE:** BIOS calls this table a partition table. Unix refers to regions of the disk that are separate file systems as partitions. To try to avoid the obvious confusion, we call to the former a "(BIOS) partition", and the latter a partition.) The table has four entries; one is designated as "active". The boot process finds the active partition and invokes it. This table makes it possible to divide the disk up into several disjoint areas. Thus Mach and DOS can both coexist on the same disk.

The "Mach (BIOS) partition" begins with a boot program that can occupy up to 29 sectors. This is followed by one sector for a vtoc (volume table of contents). The vtoc specifies the override disk geometry and then divides the "Mach (BIOS) partition" into the standard Mach/Unix file system partitions. The vtoc is followed by four sectors that record bad blocks and a number of sectors that are used as bad block replacements. At the next cylinder boundary, the first Mach partition begins; this is typically the "root".

All the structures described above need to be initialized for Mach to be able to use the disk. The programs, fdisk, vtoc, and badblocks, edit the necessary structures. They will let you edit every field in every structure. We will not describe them in detail; see the manual pages for more information.

We also provide a simpler interface:

```
diskutil clobber_my_disk
```

This initializes the structures to use the whole disk for Mach. It is equivalent to the following three commands:

```
diskutil finit
badblocks -w
diskutil fs
```

The first step, diskutil finit, does a quick and basic initialization of the structures to an empty state. Then, you run badblocks to scan the disk for bad spots. Finally, you put the basic Mach partitions, "a" - "e" on the disk, with diskutil fs. (This uses the Mach default layout.)

You may replace diskutil finit with diskutil init. This DOES NOT initialize the sector 0 disk layout information. You MUST use fdisk to pick a "(BIOS) partition" for Mach, before you run this alternate sequence. You should put the sector start address for the "Mach (BIOS) partition" on a cylinder boundary.

7 Known Features/Problems/Bugs

7.1 gas

To be compatible with the AT&T assembler, the gas assembler has been generated to accept "#" and "/" as comment characters. This means that, at present, there is no way to indicate division in assembler expressions.

8 In Trouble?

If during the installation or setup you run into difficulties, ECS will be happy to assist. Just drop a line to ecs-help@eng.ufl.edu and we will get back to you.

Appendix

A A Few More Anomalies

- If you can not low level format your disk, talk to the manufacturer or sales office. This is a routine "BIOS/DOS" procedure. We can not help you.
- If you personally installed a 80387, make sure that there are not switches to change or a setup program, to run to tell the machine about the fpu.
- Unless you are running X, Mach will use a "monochrome 80 x 25" mode of character display. Some vga cards misbehave; you might be better off with a simple monochrome board if you have one. Also, be sure you have initialized your vga board for this mode; some boards require special setup.
- Believe it or not, we have found at least one keyboard that does not work with Mach – it does work with DOS. What will happen is that after you get the single user prompt on the floppy boot, you will not be able to type.
- The X server as currently distributed, cannot return the terminal back to a 80x25 mode. Therefore, once you run X be prepared to always run X (xdm) or to reboot when you log out.

B Ethernet

In the presentation below, you will see that we prefer IRQ 9 for the ethernet interrupt level. On some cards, mostly 8 bit cards, there is no option for IRQ 9; you should IRQ 2 instead – which just happens to map to IRQ 9 on an AT BUS machine.

B.1 PC586

We can accept:

PORT	IRQ	MEMORY
0x0d0000	9	0x0d0000
0x0c0000	5	0x0c0000
0xf00000	12	0xf00000

A schematic of the board jumpers is shown below with one acceptible/prefered jumper arrangement. NOTE: This is not the specification for an iMX586 which is the equivalent board but layed out differently.

Jumper 1: E27-E32 selects 8 Mhz bus.

The board static ram must be jumpered to address 0xD0000

Jumper 2: E28-E33

Jumper 3: E29-E24

Jumper 4: E30-E25

Jumper 5: E31-E26

Jumper 6 thru jumper 13 choose one: The board must be configured for interrupt request line 9. This is jumper 10 set to E41-E49.

Jumper 14: E54-E55 selects 0WS mode.

Jumper 15: is left alone.

Jumpers 16 thru Jumpers 21: E1-E18 control whether the board uses cheapnet or standard ethernet. The figure below shows it using cheapernet. Move the jumpers down for standard ethernet.

The figure below visually shows the jumper pins of interest. Those pins marked with X's are the jumpered pins. E56 — E58 may be missing from older cards.

```
                           J J J J J J
                           1 1 1 1 2 2
                           6 7 8 9 0 1

                         E1 X X X X X X E6
                         E7 X X X X X X E12
                        E13 . . . . . . E18
 J15

X E19
X E20
. E21

                J J J J J
                1 2 3 4 5

         E56 X   . . X X X E22-E26
         E57 X   X X X X X E27-E31
         E58 .   X X . . . E32-E36
          . . . . X . . . E37-E44
          . . . . X . . . E45-E52

          J J J J J J J J
          0 0 0 0 1 1 1 1
          6 7 8 9 0 1 2 3

                           J14

                    E53 . X X E55
```

B.2 WD8003 & Etherlink II

Both these boards use the NS 8390 ethernet chip; they are programmed "basically" the same with a few localized differences to account for the board logic.

B.2.1 WD8003 & WD8013

We can accept:

port	irq	memory
0x280	9	0xd0000
0x2A0	9	0xd0000
0x2E0	5	0xd0000
0x300	5	0xd0000

The WD8013 is the 16 bit card. It is very nicely layed out and all the jumpers are labelled. Select a set of options consistent with the above table.

For the WD8003, there are five sets of jumpers that must be set. A schematic of the board jumpers is shown later with one acceptible/prefered jumper arrangement. NOTE: with the DELL 320SLT, you must use a setting with IRQ 5.

W1: Sets the Port address. We use 280 which is indicated in the figure below. If you interpret (jumper) shorts as 1 then the map of 16 possible port ranges is given below: (The *'d ports are the ones we support.)

port	0x200	0x220	0x240	0x260
pattern	0xF	0x7	0xB	0x3
port	0x280*	0x2A0*	0x2C0	0x2E0*
pattern	0xD	0x5	0x9	0x1
port	0x300*	0x320	0x340	0x360
pattern	0xE	0x6	0xA	0x2
port	0x380	0x3A0	0x3C0	0x3E0
pattern	0xC	0x4	0x8	0x0

The jumper labeled 2 in the W1 set, reduces the number of board wait states from 4 to 2.

W2: selects the IRQ level that the board is to use. We use IRQ2 that gets remapped to IRQ9 on ISA bus machines. The possiblilties are:

- jumper 11 → IRQ2
- jumper 9 → IRQ3
- jumper 7 → IRQ5
- jumper 5 → IRQ5
- jumper 3 → IRQ6

- jumper 1 → IRQ7

W3: Is shorted for thin ether, otherwise all the jumpers are open.
W4: Is open for thin ether, otherwise shorted.
W5: Is only meaningful for thin ether and we wanted it shorted.
Below is a picture of the board and jumpers set up for thin ether.

```
               W4

                                                             E
                                                             T
     2468a W1                _ W3                            H
      || |                   _                               E
                             _                               R
                             _
                             _                               T
                             _                               A
                                                             P
                                                             S
                                             | W5

             b97531 W2
             |
```

B.2.2 EtherLinkII

This board, like the WD8013, has nicely labeled jumpers. There are only two jumpers: one for the memory address and one for the port. You may use either 0xd8000 or 0xdc000 for the memory address. Unfortunately, the thick/thin ether setting and IRQ are made by software. So we use the port address to imply a setting of IRQ and thick vs thin ether.

PORT	IRQ	Ether Tranceiver
0x280	9	thin
0x2A0	9	thick
0x2E0	5	thin
0x300	5	thick

B.2.3 Etherlink I

We can accept:

port	irq
0x300	2

A schematic of the board jumpers is shown below with one acceptible/prefered jumper arrangement.

```
I/O ADDR        MEM
4 5 6 7 8 9     EN

. . . . X X      X
X X X X X X      X
X X X X . .      .


MEM     1 1 1 1 1 1 1 1
ADDR    2 3 4 5 6 7 8 9

        . . X X . X X X
        X X X X X X X X
        X X . . X . . .


INTERRUPT           DMA
2 3 4 5 6 7     1 2 3 1 2 3
|               |     |
```

C KDB commands

The following is a brief list of the kdb commands. kdb uses the same command format as adb and expertise in adb is assumed. To enter the debugger:

```
ctrl-alt-d
```

The format commands: [addr]/[letters]

and write commands *[addr]/w, W* values function exactly as you would expect from adb. (And note that / and ? are treated the same.) So you can use kdb to disassemble code, look at memory locations and change memory locations.

$r	will print out the register values
<rname/var	accesses the value of register_name/variable
>rname/var	sets the value of register_name/variable

The neat features are the "step/breakpoint" commands, which are again analogous to the adb functions:

:s	single step one instruction
:c,C	continue until the next breakpoint (these both takes counts ala adb syntax)
:p,P	print the instructions while single stepping
:j	step until the next call or return and count the instructions
:J	step until we return to this nesting level (show functions entered and instruction counts.)
:S	step over the function call you are stopped at
:r,R	set a temporary break at the return address, so you effectively step out of this function. (Note: make sure you have done the push %ebp, movl %esp, %ebp before you try this.)

Breakpoints are set with [addr]:B and [addr]:D and listed with $B, just like adb. You may not provide commands to execute when a breakpoint is hit. [addr]:b #, [addr]:d and $b are very different breakpoints. They use the i386 hardware match facilities. You can set up to 4 breakpoints and you have to specify which one to ":b". Right now they work analogous to the "B" breakpoints. But someday, I will implement access to the hardware break on read and break on write features; all that is supported now is break on execute.

$l	lists the state of all the threads
$L	lists the state and stack of all the threads
$k	does a stack trace
$c	does a stack trace
<thread>$K	does a stack trace for the give thread
<addres>$C	does a stack trace at the given frame.
<map>$m	prints the map at address <map>
<addres>$Pp	prints port
<addres>$PO	prints object
<addres>$PM	prints map
<addres>$PP	prints "page structure"
<function>! arg0 arg1 ...	invokes function with the given arguments
R!	invokes the function to sync the disk and reboot
Q!	invokes the function to reboot
pb!	invokes the playback function. The screen is small compared to the info you might want to display; pb will play back each line and wait for a space or return before continuing. Any other character terminates playback. The buffer is 64k.

D Manual Pages